Loughborough University

This item was submitted to Loughborough University as a PhD thesis by the author and is made available in the Institutional Repository (https://dspace.lboro.ac.uk/) under the following Creative Commons Licence conditions.

creative commons

COMMONS DEED

Attribution-NonCommercial-NoDerivs 2.5

You are free:

- to copy, distribute, display, and perform the work

Under the following conditions:

Attribution. You must attribute the work in the manner specified by the author or licensor.

Noncommercial. You may not use this work for commercial purposes.

No Derivative Works. You may not alter, transform, or build upon this work.

- For any reuse or distribution, you must make clear to others the license terms of this work.
- Any of these conditions can be waived if you get permission from the copyright holder.

Your fair use and other rights are in no way affected by the above.

This is a human-readable summary of the Legal Code (the full license).

Disclaimer

For the full text of this licence, please go to:
http://creativecommons.org/licenses/by-nc-nd/2.5/

LOUGHBOROUGH
UNIVERSITY OF TECHNOLOGY
LIBRARY

AUTHOR/FILING TITLE
LAW, H.W.

ACCESSION/COPY NO.
040101530

VOL. NO.

CLASS MARK
LOAN COPY

28 JUN 1996
21 NOV 1995
26 MAR 1999

0401015300

BADMINTON PRESS
10 THE HALFORDS
SYSTON
LEICESTER LE7 8LD
ENGLAND
TEL 0533 302917
FAX 0533 336696

KNOWLEDGE-BASED

COMPUTER AIDED PROCESS PLANNING SYSTEM

FOR

THE MANUFACTURE OF BARE PRINTED CIRCUIT BOARD

by

Hang-Wai LAW, MSc (Loughborough), MIEE, CEng

A Doctoral Thesis
submitted in partial fulfillment of the requirements
for the award of Doctor of Philosophy
of the Loughborough University of Technology

June 1994

© by Hang-Wai LAW 1994

Loughborough University
of Technology Library

Date Feb 91

Class

Acc. No. 040101530

V8912516

CONTENTS

DECLARATION 1
ACKNOWLEDGMENTS 2
ABSTRACT 3
1. INTRODUCTION 5
1.1 Overview and Problem Description 5
1.2 The Area of Investigation 6
1.3 Research Approach 6
1.4 Structure of the Thesis 7
1.5 Summary of Contribution 9

2. THE MANUFACTURE OF BARE PRINTED CIRCUIT BOARDS AND PROCESS PLANNING 10
2.1 Introduction 10
2.2 Manufacturing Processes for Printed Circuit Boards (PCBs) 10
2.2.1 Printed Circuit Board (PCB) Manufacturing Technology 11
2.2.2 Some Essential PCB Manufacturing Processes 13
2.3 Circuit Board Specification Representation Method 18
2.4 Circuit Board Manufacturing Methods 19
2.5 Planning and Process Planning 22
2.6 Computer Aided Process Planning (CAPP) 24
2.6.1 The Importance of CAPP in Modern Manufacture 25
2.6.2 CAPP Approaches 26
2.7 CAPP Researches in Various Manufacturing Operations 29
2.7.1 Machining 29
2.7.2 Assembly 30
2.7.3 Process 30
2.8 Significant Functions in an Ideal CAPP System 31
2.9 Summary 32

3. KNOWLEDGE-BASED CAPP SYSTEMS AND OBJECT ORIENTED MODELLING TECHNIQUES 33
3.1 Introduction 33
3.2 Knowledge-Based System 33
3.2.1 User Interface 34
3.2.2 Inference Engine and Working Memory 35
3.2.3 The Knowledge Base and Knowledge Representation 36
3.2.4 External Interfaces Unit 38
3.3 Knowledge-Based Computer Aided Process Planning System 39
3.3.1 The Planning Problem description 39
3.3.2 The Inference Engine 40
3.3.3 The Planning Knowledge Base 41
3.3.4 User Interface 41
3.3.5 Planning Knowledge Acquisition and Up-dating 42
3.3.6 External Interface 42
3.4 Object-Orientation 43
3.4.1 Abstraction 43
3.4.2 Encapsulation 43
3.4.3 Polymorphism 43
3.4.4 Inheritance 44
3.5 Object Oriented Analysis and Design Techniques 44
3.6 Object Modelling Technique (OMT) 45
3.6.1 The Three Models of OMT 46
3.6.2 The Four Phases of OMT 48
3.6.3 Traditional Software Engineering Approaches and OMT 49
3.7 The Use of OMT Methodology in this Project 51
3.8 Summary 53

4. OMT MODELLING OF THE KNOWLEDGE BASED PCB CAPP SYSTEM 54
4.1 Introduction 54
4.2 Problem Statement 54
4.3 Object Model of the System 55
4.4 Dynamic Model of the System 60
4.5 Functional Model of the System 64
4.6 Hardware and Software Architecture of the System 66
4.7 Summary 68

5. MODELLING & IMPLEMENTATION OF THE INTERFACE MODULE AND THE AUTOMATIC FEATURE EXTRACTION MODULE 69
5.1. Introduction 69
5.2 The Three Models of the Interface Module 70
5.3 Object Modelling and Implementation of the In-house Process Capability Interface Module 73
5.3.1 Object Classes of the Process Capability Knowledge Base 74
5.3.2 Implementation of the In-house Process Capability Interface Module 77
5.4 Modelling and Implementation of the Customer General Specification Interface Module 80
5.4.1 Object Classes of the Customer General Specification Data Base 81
5.4.2 Implementation of the Customer Specification Interface Module 84
5.5 Object Modelling and Implementation of the Product Requirements Interface Module 87
5.5.1 Object Classes of the Product Requirement Data Base 87
5.5.2 Implementation of the Customer Product Requirements Interface module 91
5.6 Modelling and Implementation of Features Extraction Modules 95
5.6.1 Modelling of the Circuit Feature Extraction Module 95
5.6.2 Modelling of the Solder Masking Feature Extraction Module 98
5.9. Summary 99

6. PLANNING KNOWLEDGE USED IN THE KB PCB CAPP SYSTEM 101
6.1 Introduction 101
6.2 Knowledge Representation in the KB PCB CAPP System 101
6.3 Rules to Check Product Requirements against Customer General Specifications 104
6.3.1 Rules to Check Circuit Features 106
6.3.2 Rules to Check Copper Plating Requirement 108
6.3.3 Rules to Check Solder Masking Clearance 108
6.3.4 Rules to Check Gold Finger Requirement 110
6.3.5 Adding New Specification Checking Rules 110
6.4 Process Selection Knowledge 111
6.4.1 Rules to Select NC Drilling Process 112

6.4.2 Rules to Select Electroless Copper Plating Process 113
6.4.3 Rules to Select Circuit Image Transfer Dry Film Process 114
6.4.4 Rules to Select Bare Copper Plating Process 114
6.4.5 Other Rules to Select Process 115
6.5 Process Sequencing Knowledge 116
6.6 Machine Selection Knowledge 119
6.7 Process Recipe Generation Knowledge 122
6.8 Summary 123

7. IMPLEMENTATION AND OPERATION OF THE KB PCB CAPP SYSTEM 125
7.1. Introduction 125
7.2. Implementation of the Process Selection and Sequencing Session 125
7.2.1 Loading of Data Objects 125
7.2.2 Process Selection and Sequencing 127
7.3 Machine Selection Session 134
7.4 Process Recipe Generation Session 139
7.5 Summary 146

8. EVALUATION OF THE DEVELOPED KB PCB CAPP SYSTEM 147
8.1 Introduction 147
8.2 Description of the Evaluation 147
8.2.1 In-house Process Capability and In-house Machine Capability 149
8.2.2 Customer General Specification 153
8.2.3 Product Requirements 154
8.3 Results of the Evaluation and Discussion 166
8.3.1 Circuit Features 166
8.3.2 Product Requirements Checking Against Customer General Specifications 167
8.3.3 Process Selection and Sequencing 169
8.3.4 Machine Selection 169
8.3.5 Process Recipe 171
8.3.6 Planning Time 173
8.4 Overall Discussion 174
8.5 Summary 175

9. DISCUSSION AND CONCLUSION 176
9.1 Introduction 176
9.2 Contributions of this Research Project 176
9.2.1 Pioneer CAPP Project in PCB Manufacture 176
9.2.2 OMT Modelling Methodology in KB CAPP System 176
9.2.3 Systemic Structuring of the Knowledge in the System 177
9.2.4 Superiority of the KB PCB CAPP System 179
9.3 Suggestions for Future Work 180
9.4 Conclusions 182

REFERENCES 183

APPENDIX 1 KAL Representation of the In-house Process Capability A-1

APPENDIX 2 Customer General Specification
2.1 KAL Representation of the CustomerA General Specification A-11
2.2 KAL Representation of the CustomerB General Specification A-15

APPENDIX 3 Product Requirements
3.1 KAL Representation of the CircuitA Requirements A-19
3.2 KAL Representation of the CircuitB Requirements A-25
3.3 KAL Representation of the CircuitC Requirements A-31
3.4 KAL Representation of the CircuitD Requirements A-37
3.5 KAL Representation of the CircuitE Requirements A-43

APPENDIX 4 Feature Extraction Program
4.1 Listing of the Circuit Feature Extraction Program A-49
4.1 Listing of the Solder Masking Feature Extraction Program A-93

APPENDIX 5 Product Image Data File
5.1 CircuitA Circuit Image Data File A-105
5.2 CircuitA Solder Masking Image Data File A-107

APPENDIX 6 Feature Data File
6.1 CircuitA Circuit Feature Data File A-109
6.2 CircuitA Solder Masking Feature Data File A-110

APPENDIX 7 Knowledge Base
7.1 Customer General Specification Checking Rules A-111
7.2 Process Selection Rules A-114
7.3 Process Sequencing Rules (Methods) A-120
7.4 Process Capability Checking Rules A-121
7.5 Machine Selection Rules (Methods) A-124
7.6 Process Recipe Generation Rules A-126

APPENDIX 8 Developed System Support Functions A-129

APPENDIX 9 System Support Objects
9.1 In-house Process Capability Interface Objects A-141
9.2 Customer General Specification Interface Objects A-142
9.3 Product Requirements Interface Objects A-144
9.4 "ProNameSeqNo" Objects A-148
9.5 "SuggProInHou" Objects A-149
9.6 "MCChar" Objects A-160
9.7 "MCNameList" Objects A-164
9.8 "ProcessRecipe" Objects A-166
9.9 "MessageFile" Objects A-169
9.10 "Session" Windows Objects A-170
9.11 "Button" Objects A-176

APPENDIX 10 Process Planning Results

10.1 CircuitA Process Planning Results A-190

10.2 CircuitB Process Planning Results A-193

10.3 CircuitC Process Planning Results A-197

10.4 CircuitD Process Planning Results A-201

10.5 CircuitE Process Planning Results A-205

DECLARATION

I declare that the work is original and of my own presented in this thesis, unless specified. Neither the thesis nor the original work presented therein has been submitted to this or any other institution for a higher degree.

Hang-Wai LAW

June1994.

ACKNOWLEDGMENTS

My sincere appreciation goes to my supervisor, Professor David Williams, for his intellectual stimulus, encouragement, valuable advice and support throughout the project.

My thanks also go to Mr. Hang. C. Law for his proof-reading this thesis. The greatest help was provided by my wife, Isabel, who gave me concrete support and never lost her patience with me or enthusiasm for my research project.

Hang-wai LAW

June 1994

ABSTRACT

This thesis focuses on the use of a knowledge-based computer aided system for the task of bare printed circuit board (PCB) process planning. To achieve this task, a knowledge-based computer system has been developed in which process plans can be generated automatically. The planning decisions are based on board requirements, customer general specifications and product quality standards.

Object modelling technique (OMT) methodology is used extensively to model the system. Different levels of details in object view, dynamic view and functional view have been established and developed. Object oriented knowledge-based software shell Kappa-PC and object oriented language C++ are used to implement the system. Findings of the test show that the effects of applying OMT methodology to modelling and implementation of the process planning system are pioneer and have advantages over the traditional ad hoc approach.

The developed process planning system has a product planning features extraction module. Pattern plating area, minimum circuit line spacing and solder masking pad clearance features can be identified and extracted from the required circuit board. The planning system is knowledge-based. A board-based domain of knowledge which includes global board manufacturing knowledge, in-house (local) process capability knowledge, process sequence knowledge, machine selection knowledge and process recipe generation knowledge has been acquired and represented by declarative facts and production rules. The knowledge is segmented and structured to facilitate searching and up-dating. The

system can suggest feasible processes, arrange them in appropriate manufacture sequence, select appropriate machine and generate process recipe.

The performance of the developed knowledge-based computer aided process planning system has been evaluated. It has been found that technical people with no PCB working experience can use this system to generate process plans comparable to those produced by experienced planners. In addition, the system can provide better functionality, use less planning time and generate more consistent process plans than those done manually by experienced planners. The thesis concludes that the author's pioneer project has been proved successful in using a knowledge-based computer aided process planning system to automate the printed circuit boards process planning activities.

CHAPTER 1
INTRODUCTION

1.1 Overview and Problem Description

The development of electronic industry is quality, technology and information driven. This is specially true in the production of printed circuit boards (PCB) which are one of the most crucial component in any electronic product. The electronic products differ greatly in their board designs. Due to business competition, circuit board manufacturers constantly pursue for higher board quality, shorter throughput time, smaller batch size and better process yield. In view of shortening life cycle of the electronic products and the more precise circuit requirements, those circuit board shops that use traditional and labour intensive means of production technology and processing manufacturing information are finding it difficult to survive in such a highly competitive electronic manufacturing environment. The use of computer technology to assist design, planning and manufacture of PCBs is a logical and timely development.

At present, in most circuit board shops, process plans (also called instruction sheets or job cards) are prepared manually by process planners. The process planners first have to study the particular circuit board specifications, customer requirements (such as order quantity, delivery time, functional specifications and quality level), in-house working guidelines and in-house process capabilities. Based on their own experience and understanding, the planners prepare the process plans. This method of generating the process plans is very labour intensive and requires a lengthy lead time; the quality of the process plan also relies very much on the individual process planner's experience and expertise. Training new process planners is another difficult task faced by circuit board

manufacturers because the in-house planning knowledge is not consolidated and the manual decision making procedure is not structured. New process planners usually acquire their planning knowledge on real jobs and learn through planning mistakes. Experienced process planners also face the problem of updating their own circuit board manufacturing knowledge because of the seemingly incessant advancement of circuit board manufacturing technology. It can be said that the manual method of process planning is time consuming, error-prone and inconsistent. Development of a computer system to automate, at least, part of the process planning is worth pursuing.

1.2 Area of Investigation

This is a project aimed to develop a structured, computer aided printed circuit board process planning system with a capability for generating process plans for the manufacture of bare printed circuit boards and capturing formally the knowledge used in the process planning process. The structure of such a system should include functions such as a user interface for problem extraction and formulation, process selection, process sequencing, facility or equipment selection, process recipe generation and planning knowledge up-dating.

1.3 Research Approach

This research is focused on capturing the problems of using a computer system for the task of process planning in bare circuit board manufacture. First is to identify a suitable framework for the development of such a computer aided process planning system. The limitations of the process planning system developed using such a computer framework

are explored. A modelling methodology is, then, identified and the advantages of using such a modelling methodology are justified. A planning system is developed by using the selected modelling methodology. Based on the developed model, a system is implemented and tested. The system's performance in process planning is compared with those of the experienced process planners.

1.4 Structure of the Thesis

There are eight further chapters in this thesis. Chapter two introduces PCBs' manufacturing technology and their process planning. It further discusses the importance of developing a computer aided process planning (CAPP) system in a computer integrated manufacturing environment. Finally, the CAPP researches in different manufacturing areas are reviewed and the functionality of an ideal CAPP system is discussed.

Chapter three examines knowledge-based systems (KBS) and knowledge-based computer aided process planning system (KB CAPP). Object oriented concepts are described and the different object oriented analyses and design methodologies reviewed. The chapter then elaborates the advantages of using the object modelling technique (OMT) and the justifications for adopting OMT methodology for this research.

Chapter four contains the results of the analysis, the overall system design and the object design phases of the OMT methodology. The scope of the system is defined by formulating a formal problem statement and the requirement specifications. The three different views of the system using the object model, the dynamic model and the functional model are introduced. The chapter finally describes the implementation of the hardware and software architecture of the KB CAPP system using the developed models.

Chapter five describes the work of the interfacing module and the automatic feature extraction module of the system. Object models of the three sub-level interface modules are then presented and their implementation illustrated. Then, the circuit feature and solder masking feature extraction functions are modelled and their implementations discussed.

Chapter six explains how the object oriented model is used to model the knowledge structure of the system. The represented knowledge is categorised into five groups and discussed with examples.

Chapter seven deals with the modelling and implementation of the knowledge-based printed circuit board process planning system. The loading of data session is first discussed. Then, the object model and dynamic model of the selection and sequence session, the machine selection session and the process recipe session are presented with results of the implementation.

Chapter eight presents the measurements of the system performance and compares the system performance with those of the experienced process planners. It also analyses the evaluation results.

Chapter nine concludes the thesis with an overall discussion about the research. It reviews the results and summaries the contribution of the research work. The chapter closes with an identification of the possible areas for further research work.

1.5 Summary of Contribution

The contribution of this research can be summarised into four broad areas.

A knowledge-based framework for the development of a computer aided printed circuit board process planning system has been identified. An evaluation on the developed system has showed that the developed circuit board process planning system could perform well under the identified framework.

The Object Oriented Technique Methodology has been identified and justified to be a better modelling and implementation methodology for a computer aided circuit board process planning system. The research has demonstrated that using such a methodology could avoid the drawback of the commonly used ad hoc approach in knowledge-based system development. It could provide a structural means for the system developers to improve the planning system progressively.

Circuit board process planning knowledge which includes a global board manufacturing knowledge, in-house (local) process capability facts, customer general specifications and board requirements has been successfully acquired and represented using the declarative facts and the production rules structure. An object-oriented knowledge inference mechanism for the planning system has been developed. It has been demonstrated that the performance of such structure is superior than that of simple rule based knowledge inference mechanism.

An evaluation of the system has shown that a pioneer knowledge-based circuit board process planning system could provide better functionality, use less planning time and give more consistent planning results.

CHAPTER 2

THE MANUFACTURE OF BARE PRINTED CIRCUIT BOARDS

AND

PROCESS PLANNING

2.1 Introduction

This chapter discusses the various manufacturing processes for Printed Circuit Boards (PCBs), with a brief account of the usual industrial practice in circuit board specification representation methods. Different types of circuit boards and the manufacture of such boards in industry are explained. Planning and process planning are discussed; Computer Aided Process Planning (CAPP) is introduced together with relevant research in the area. Functions of an ideal CAPP system are then proposed at the end of the chapter.

2.2 Manufacturing Processes for Printed Circuit Boards (PCBs)

As a result of the fast development of the electronics industry, electronic product developments are constantly driven by the goals of high quality, short throughput time, small batch size and short product life cycle. In electronic products, printed circuit boards are perhaps the most crucial components and each product requires a totally different circuit board. Traditional labour intensive methods of manufacture information preparation and manufacturing of circuit boards are no longer acceptable to board manufacturers. With the rapid introduction of new electronic products, the need for fast

product turn over is especially crucial. Therefore, the use of computer technology to assist design, planning and manufacture of PCBs is a logical and timely development.

2.2.1 Printed Circuit Board (PCB) Manufacturing Technology

PCB is used here to mean a board with a printed pattern of electrical interconnects. It may have one or multiple layers (Figure 2.1) of patterned metal circuits on an insulated and laminated board such as fibreglass. Complex PCBs can have as many as 24 layers of circuits, although PCBs of two to eight layers are most common. Metal circuits (usually copper) are used for electrical connection while the laminated fibreglass serves as electrical insulation and gives the board strength. Usually in a multi-layer board, one or two layers are used as ground planes to improve the noise characteristics of the circuitry and reduce its susceptibility to the electromagnetic interference.

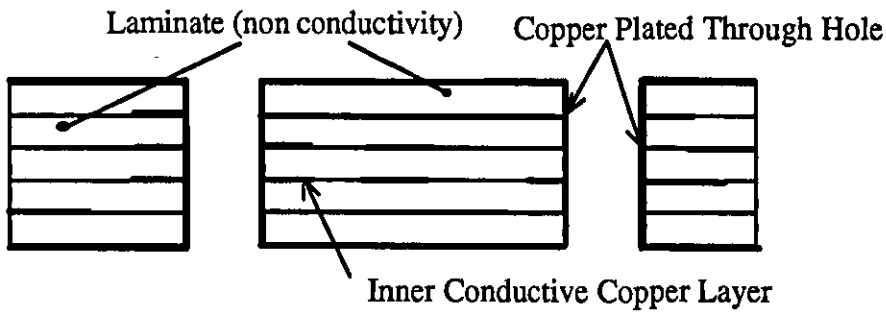


Figure 2.1 A Six-Layer Circuit Board

The processes of the PCB manufacture can be conveniently grouped into six categories: the preparation of production tooling, machining processes, the image transfer processes, chemical processes, testing/inspection and other miscellaneous processes (Table 2.1). The manufacture of PCBs usually starts with laminated copper sheets.

Preparation of Production Tooling
1. Drill Program Preparation
2. Routing Program Preparation
3. Blanking Tool Preparation
4. Tester Preparation
5. Silk Screen Tool
6. Circuit Production Film

Machining Processes
1. Board Drilling
2. Board Routing
3. Board Punching
4. Board Blanking
5. Board Laminating
6. Laminated Board Trimming
7. Board Deburring
8. Board Scrubbing
9. Board "V-cutting"

Image Transfer Processes
1. Circuit Image Transfer
2. Solder Masking
3. Component Marking

Chemical Processes
1. Etching
2. Etch back
3. Electroless Copper Plating
4. Copper Etching
5. Copper Plating
6. Pattern Plating
7. Edge Gold Plating
8. Stripping of Plating Resist

Testing and Inspection
1. Computer Vision Inspection
2. Open-Short Testing
3. Micro Sectioning
4. Chemical Analysis
5. Manual Visual Inspection

Other Miscellaneous Processes
1. Reflow of Tin-Lead
2. Hole Plugging
3. Hole Tenting
4. Board Cleaning
5. Packaging

Table 2.1 Six Categories of PCB Manufacturing Process

Different types of boards can be manufactured by different combinations of the above processes. Some of these processes such as electroless copper plating and edge gold plating may involve as many as ten or more sub-processing steps. Some other processes involve very expensive production equipment (e.g. one NC drilling machine can cost US$200,000 - $300,000 and an automatic plating line can cost US$300,000 - $600,000). In addition, process parameter settings need to be precisely controlled because the manufacturing processes of circuit boards are susceptible to contamination; boards with slight defects will be rejected by board users.

Because of the complexity of the circuit and the size and scale of the board, electronic Computer Aided Design (CAD) systems are usually used in the design of PCBs.

However, these computer designed data have to be interpreted by the production engineers who are responsible for the preparation of the board manufacturing data. The manufacture of PCBs still depends very much on manual intervention, both physical (material movement and machine/equipment loading/unloading) and mental (data logging, judgment and decision making). Following automation in industry, material movement and equipment loading/unloading are now mostly handled by automatic equipment and data logging by automatic data logging devices. However, work tasks involving human judgment are difficult to automate as a comprehensive understanding of the production processes and consolidated board manufacturing experience on shop floors is still not available. The existing computer technology is capable to provide real-time on line tracking of board locations and monitor manufacturing process status. An application of the computer technology to dynamically changing the product process routing and adjusting the process parameters looks possible, but such application is still rare in board shops because an automatic process planning system in PCB industry is still unavailable.

2.2.2 Some Essential PCB Manufacturing Processes

The production of a good-quality PCB requires precise control of the manufacturing processes. Some essential PCB manufacturing processes are discussed below.

Hole Drilling. Holes in PCBs are created either by punching or by drilling. Holes prepared by drilling have better wall quality and narrower tolerances than those prepared by punching. Drilling also does not require expensive tooling and a consequent long tooling fabrication lead time. Moreover, the use of CNC control drilling machines gives a greater degree of flexibility; hence most Plated Through Holes (PTH) are drilled. Holes in PCBs are for insertion of electronic components, for mounting of mechanical components, for electrical connection of different circuit layers and for the mounting of the circuit

board in the product housing. A precise and burr-free hole is a prerequisite to the good quality of a printed circuit board. To obtain a consistently high quality of holes, a balance of proper drill bits, drilling machines, cutting speeds, cutting feed rates, entry and backup materials and copper-clad materials are required.

Electroless Copper Deposition. This is commonly known as the "PTH" process in PCB manufacture. By means of an appropriately controlled chemical reaction, copper is deposited on the hole surface to allow the electrical connection of the hole walls. After the PTH process, the whole PC board is conductive (Figure 2.2) and any other required electrical plating processes can be performed.

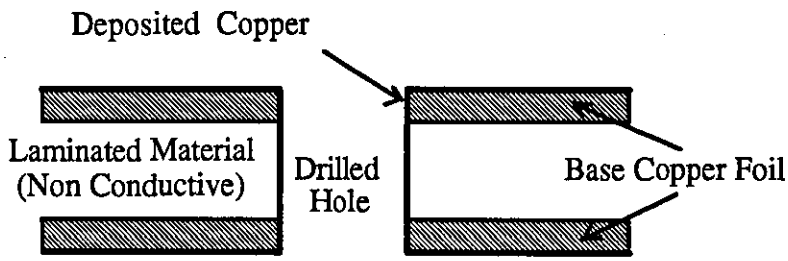


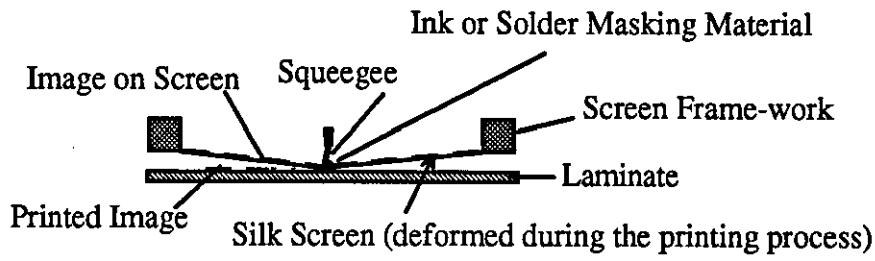
Figure 2.2 Electroless Copper Deposition

Electrical Copper Plating. Because of its high conductivity, low cost and ease of handling, copper is almost invariably used as the base metal for PCBs. Plating takes place in a tank of electrolyte. Boards are submerged into the tank which is connected to the cathode of an electrical source. Anode electrodes made of copper are placed on each side of the boards and connected to a rectifier. The electrode copper releases copper ions that migrate through the electrolyte (a mixture of copper sulfate and sulfuric acid) onto the circuit board surface. As the circuit board is the cathode, copper ions become metallic copper as they deposit on the surface of the board and the hole walls. Usually, boards are

oscillated forwards and backwards in the plating bath during the plating process in order to achieve an even copper thickness. Electrical copper plating will be done after the PTH process and after the circuit image transfer process. The former is known as panel plating in which the whole board surface is plated with copper while the later is known as pattern copper plating where only the circuit patterns exposed are plated with copper.

Image Transfer. The image transfer method is used to transfer the circuit image, solder masking pattern or component marking onto the circuit board surface. There are two main types of image transfer method: silk screen transfer and photo image transfer (Figure 2.3). Both types of method can be used for the transfer of circuit image and solder masking pattern. However, for component marking, the silk screen method is commonly used because of its relatively cheaper material and production cost. In the silk screen method, a silk screen with an image pattern is placed on the top of the board surface, ink or chemical is then applied onto the surface of the screen and then squeezed through the opening of the silk mesh by a squeegee. The pattern on the screen will be transferred onto the surface of the board. As the screen will be deformed during printing and there is a limitation in the screen opening tolerance, a sharp and precise pattern (e.g. circuit width under 0.1mm) is difficult to produce using the silk screen method. Also, two different screens will be required for a double sided board and two separate processing steps will be involved. Therefore, the silk screen image method is used only for producing wide tolerance circuits, solder mask images and component marking images. In photo image transfer, an ultra violet (UV) light sensitive ink or chemical is applied onto the board surface either by laminating a film (the dry film method) or by silk screen printing (the wet film method). The master film pattern is then placed on the board surface and the board is exposed to UV light. Patterns can be transferred onto the UV light sensitive material on both sides of the board surface at the same time. The image on the board is then developed. Because of its tight tolerance image capability, the dry film method is

commonly used for circuit image transfer while the wet film method is adopted in solder mask image transfer.



Silk Screen Image Transfer Method

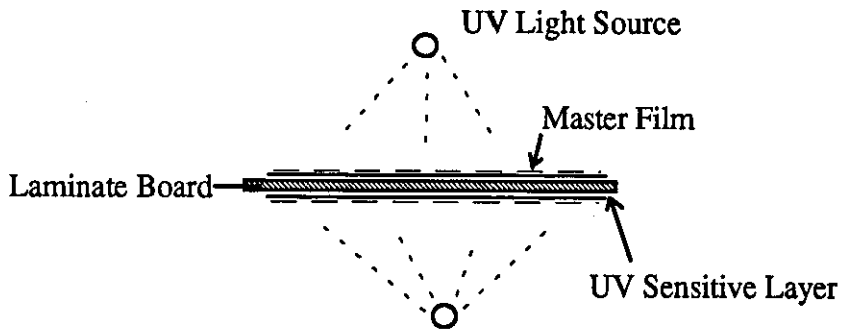


Photo Image Transfer Method

Figure 2.3 Silk Screen Image Transfer and Photo Image Transfer

Tin/lead Plating. Tin/lead is commonly plated on the circuit surface because of its low oxidation properties. It provides a good protective layer for the copper circuit of the board. In addition, tin/lead is resistant to any etching which is applied to remove un-wanted copper from the boards. Tin/lead is also compatible to solder in the subsequent soldering process during PCB assembly. The tin/lead plating process is similar to copper

plating as described above. The tin/lead alloy in the anode is usually a combination of 63% tin and 37% lead by weight; a thickness of minimum 0.007mm of tin/lead is required to have an effective resistance to any etching during the copper etching process.

Copper Etching. Copper etching is a subtractive process in which the exposed copper is removed to establish the required circuit pattern on the boards. The etching is done by a spray etching machine which sprays copper etcher on the board surface to achieve the best etching result. The etch temperature, the spray pressure and the etching time are controlled through the etching machine.

Gold Plating. Gold, a chemically stable and electrically conductive metal, is plated either on the circuit board edge connector or on the circuit pad surface for wire bonding. Because its high cost, only a very thin layer of gold is plated. In order to enhance the hardness of the gold layer, nickel is always plated under the gold.

Tin/lead Reflow. Electrically plated tin/lead does not form a metallic alloy. Thus a high temperature process must be applied to the plated tin/lead. This is known as the tin/lead reflow process in the PCB manufacturing industry. Reflowed tin/lead produces a dense tin/lead alloy with less porosity and helps prevent oxidation of the copper circuit. Successfully reflowed tin/lead board has a bright metallic surface which has a sales appeal. The bright metallic surface is also an indication that the copper circuit surface is adequately cleaned before tin/lead plating. Tin/lead can be reflowed using infrared light, high temperature solvent or oil.

The Printed Circuit Handbook [Coombs, 1990] and the Newnes Electronics Assembly Handbook [Brindley, 1990] give comprehensive descriptions of the PCB manufacturing processes.

2.3 Circuit Board Specification Representation Method

The specifications of the circuit board is usually represented by: an art-work, a mechanical drawing and a purchasing order.

Artwork. Artwork representations are appropriate for circuit patterns, solder masking patterns and images for component marking. The number of circuit artworks corresponds to the number of layers required in the final circuit board. For instance, a double sided board has two circuit artworks. Solder masking artworks specify the solder masking pattern for the circuit board. For double sided or multi-layer boards, only two solder masking artworks are required to specify the component side and the solder side of the solder masking pattern. Component marking artworks capture the component markings necessary on the circuit board. The artworks can be produced manually by laying sheet of paper or by using a plotter to plot the patterns on the artwork. The plotter may be based on either photographic or laser technology. Following the development of computer software, electronic circuits are now designed and generated by using an electronic CAD system; and the images are represented by electronic data files. The data files appear in different data formats among which the Gerber format is the most common. The Gerber format is commonly accepted as a standard in the circuit board manufacturing industry.

Mechanical Drawing. The other technical requirements of the circuit boards are captured on mechanical drawings. Mechanical drawings are important documents for the fabrication and processing of a PCB; the usual data included in mechanical drawings are:

- board outline dimensions
- types of laminate to be used
- hole specification details (including specifications for plated through holes, component holes, registration holes and mounting holes and their respective locations, diameters and tolerances)
- surface finish requirements (such as types of surface finish, plating thickness, thickness tolerance, etc.)
- solder masking requirements (such as required materials and colours)
- component marking requirements (including material and colour)
- other general specifications

Purchase Order. A purchase order usually contains the part number of the circuit board, the quantity of boards required, the delivery schedule, acceptance standard, unit cost, packing specifications and other critical specifications. It is a piece of formal purchasing documentation from the board users to the board shops.

2.4 Circuit Board Manufacturing Methods

The PCB manufacturing methods are diversified and vary greatly. For example, in the manufacture of a specific type of multi-layer boards, more than 40 processes may be involved (Table 2.2) and some of these processes each may have more than 10 sub-processing steps.

1. Raw material inspection
2. Shear C stage material into panel sizes
3. Material identification marking
4. Drill racking holes in panel
5. Extra material cure
6. Registration pinholes
7. Raw material inventory
8. Raw material inventory release
9. Material scrub
10. Apply photoresist
11. Artwork image transfer
12. Develop artwork image
13. Etch inner circuit patterns
14. Remove exposed photoresist
15. Inner circuit inspection
16. Scrub for lamination
17. Bake prior to lamination
18. Shear B stage and punch registration pinholes
19. Lay up for lamination
20. Lamination
21. Remove laminate from fixtures
22. Laminate trim
23. Mark identification no. on laminated blank
24. Post-laminate bake
25. Drill through holes
26. Debur holes
27. Etchback
28. Electroless copper plating
29. Scrubbing
30. Apply photoresist
31. Circuit image transfer
32. Develop image
33. Pattern copper plating
34. Pattern tin-lead plat
35. Strip photoresist
36. Copper etching
37. Masking for edge plating
38. Edge tin-lead plating
39. Edge nickel & gold plating
40. Edge tape removal
41. Reflow tin-lead
42. Board cleaning & scrubbing
43. Solder masking of board
44. Component marking
45. Drill non-plated holes
46. Contour board
47. Board cleaning
48. Electrical test & inspection

Table 2.2 Typical Manufacturing Method of Reflow Tin/lead Surface Finish Multi-layer Circuit Board (modified from Coombs, pp. 22-26)

In general, two main processes are used for the manufacture of PCBs: the subtractive method and the additive method.

The Subtractive Method. The subtractive method starts with copper foil laminated material and etchant is used to dissolve away the unwanted part of the copper foil, leaving tracks of circuit on the base surface. It is also called the "print-and-etch" technique, which refers to the two main steps in the subtractive printed circuit board production:

1. printing an etch resist pattern on the copper surface corresponding to the required track, and

2. removing the unwanted areas of copper from the board surface by etching to produce the required pattern.

Usually, single sided PCBs are manufactured using the subtractive method.

The Additive Method. The additive method starts with a base laminate board without copper foil on the surface, adding circuit tracks where required. Additive method appears to be the most economical option because copper is only added to the required tracks. However, technically, adding copper to the base laminate is not an easy task. It involves rather complicated and time-consuming processes and gives tracks of variable electrical properties. It is therefore rarely economically justified. In fact, the manufacture of double sided and multi layer printed circuit boards uses a mixture of additive and subtractive processes and starts with copper foil laminate.

As well as choosing the most appropriate processes for the manufacture of a typical circuit board, setting different process parameters in the light of the board's specifications is also required. Most of the equipment involved is very specific and suited for a narrow range of applications. Thus the manufacture of a new circuit board can be a rather perplexing task that requires effective process planning. The process planner must thoroughly and carefully explore the various alternatives of the manufacturing processes and equipment that are available for his possible use before making his decisions. In some cases, he may even suggest re-designing the board or relaxing some of the requirements. The quality of such a decision process depends heavily on the process planner's work knowledge, the extent of his expertise and his experience. The diversity of electronic products, a shortage of experts in printed circuit board manufacture and inconsistency in manually generated process plans now give rise to a pressing need for research into computer aided process planning systems for printed circuit board manufacture. In addition, the knowledge now used in the process planning of PCB manufacture may be too heuristic and empirical,

rendering the traditional manual process planning method incapable of coping with the fast development of the electronics industry.

2.5 Planning and Process Planning

Planning is "the activity of devising means to achieve desired goals under given constraints and with limited resources" [Ham, 1988]. In other words, planning is a multi-perspective problem solving process aimed to reach a pre-defined goal under a set of constraints and limited resources. The panning process may involve: problem definition, constraint-reasoning, goal-achieving, resource utilization and conflict-resolution. In manufacturing industry there are different types of planning activities such as human resource planning, material planning, process planning and production planning.

Process planning in manufacturing is a particular planning function in which decision are made on how to translate a part design into the "best" method of manufacture. It is formally defined by the Society of Manufacturing Engineers as "the systematic determination of the methods by which a product is to be manufactured economically and competitively" [White, 1987]. In other words, process planning is a set of planning functions that establishes what manufacturing processes, what parameters and what type of methods are to be used to convert input materials from their initial state into a final state. Such a planning function is also known as: manufacturing planning, material process planning, process engineering, process routing, operation planning, etc. Correspondingly, the person who performs the process planning is called a process planner, process planning engineer, industrial engineer, manufacturing engineer or product planning engineer. In a general sense, the "process" in process planning refers to any operations performed on a part automatically, manually or a mix of both.

Process plans are prepared by process planners based on their understanding and of the product requirements, customer general quality specifications, customer requirements (such as order quantity and delivery time), processing methods and the company's own manufacturing guidelines. The process plans give details of the operation procedures including the types of processes to be used, process routings, required materials and machines and their specifications, parameter settings for each process and machine, the tools and fixtures required, operational methods (including operation instructions, operation time, numerical control part program and how the fixtures are to be set up) [Sutton, 1989] [Srinivasan, 1991]. Based on the generated process plans, production planning personnel then schedule the production and the production operators carry out the actual production. Simultaneously, the costing people use the information in the process plans to carry out costing. In summary, the process plan dictates the method, cost, quality, and rate of production of a product and therefore is the most important component in any manufacturing system. In conventional manufacturing environment, process planning is done by "experts" in the field (usually a very experienced person in the factory). What experience and knowledge that the planners possess is crucial to the quality of the process plan that they can produce.

A capable process planner should be versed in :

- interpretation of engineering documents
- all related manufacturing processes
- in house operations/practices/procedures
- resources available in the shop floor
- characteristics of different raw materials and their availability
- the relative costs of different processes, tooling, and materials
- "optimization" of the various processes under different situations

- acquisition of relevant knowledge from reference books or other experts
- presentation of process plans in human or machine readable formats

Thus, manual process planning activities rely heavily on empirical knowledge acquired mainly through practical experience. A high level of expertise is almost a perquisite for an efficient process plan. For example, Nolen reported [Nolen, 1989] that in the aerospace industry, the average age of an aerospace process planner ranges from 51 to 55 years old. As large numbers of senior process planners are approaching their retirement age, their experience and knowledge is likely to be lost. It is also reported that in the U.S. industry, about 200,000 to 300,000 process planners are required, but only 150,000 to 200,000 are currently available [Chang, 1991, pp. 404]. A shortage of process planning experts, the long time required to train up new planners, the long lead time in developing "good" process plans and the unavoidable subjectivity of individual planners, have rendered the manual method of process planning undesirable because of its time consumption, inconsistency and waste of scarce resources [Joseph, 1991]. Against such background, the timely emergence of powerful computers for storage and manipulation of vast amounts of the acquired data derived from empirical knowledge has prompted the proliferation of automated process planning systems in recent years.

2.6 Computer Aided Process Planning (CAPP)

In Computer-Aided Process Planning (CAPP) systems, computer systems are employed to code human planning knowledge and information in a specific manufacturing domain and the inference procedures to produce process plans satisfying a set of desired goals that work under given constraints and resources. Although the ultimate aim of using CAPP system is to completely automate all the activities involved in generating the whole

process plan, it has not been reported that any realistic CAPP system can generate process plans without the assistance of any human planners.

The use of computers to assist the process planning task was first proposed by Nieble [Niebel, 1965] and the "feasibility of automated process planning" was studied by Schenk and the results were reported in his Ph.D. dissertation in year 1966. The two scholars' basic idea was to extract operations and processing sequences from the part geometry of mechanical parts. The CAPP concept was not explored further until the beginning of the 70s. In the early 80s, most CAPP systems were developed by adopting the variant approach (variant and generative process planning approaches will be discussed later). In the mid 80s, CAPP systems using generative approaches began to emerge. In 1988, Alting reported that more than 156 CAPP systems were developed [Alting, 1989].

2.6.1 The Importance of CAPP in Modern Manufacture

In an investigation conducted in 1980, the U.S. Air Force Integrated Computer Aided Manufacturing (ICAM) group identified that automated process planning and design retrieval were the two items toped a list of "areas most in need of improved information handling methods within the manufacturing environment". Another study by Granville [Granville, 1990] showed that process planning could affect as much as 75% of the production costs. Automating the process planning task can have the following advantages: saving time and effort for the process planners thus shortening the product lead time, increasing the consistency of the process plans, raising the product quality, reducing the production cost, etc. Such advantages are particularly pertinent in today's manufacturing environment because of the short product life cycle, fast evolution of technology, small production batch sizes, the high expectation of customers and global competition. Studies on successful applications of CAPP systems to process planning

indicated a 47% reduction in part throughput time, a 35% improvement in process planners' productivity and a 7% reduction in total design costs [Granville, 1990]. There is an important synergy effect from the integration of a CAPP system with a computer aided design (CAD) system and the computer aided manufacturing (CAM) system. The use of CAD or CAM systems alone cannot achieve the anticipated savings in time and money. Therefore, automating the process planning functions forms a crucial link between the product design and the manufacturing functions. This automation not only eliminates the disadvantages of the manual planning methods but also bridges the gap between CAD system and CAM system to achieve some of the major integration tasks in elements of Computer Integrated Manufacture (CIM) [Chang, 1985] [Alting, 1989].

Several Delphi forecasts done in 1978 and 1980 by the Society of Manufacturing Engineers (SME) predicted that by 1988, computers would automatically generate process plans in 30% of all manufacturing industries and by 1990, 20% of the manufacturing industries would utilize an integrated system of MRP and CAPP and 50% of the process plans used to produce parts or assemblies would be computer generated [Wilson, 1980] [Colding, 1978] [Merchant, 1987]. However, all these forecasts turned out to be over ambitious and many challenges still exist for the automatic process planning research.

2.6.2 CAPP Approaches

Two CAPP approaches, namely, the variant approach and the generative approach, are commonly used [Alting, 89] [Liu, 87][Iwata, 84] [Kusiak, 90] [Chang, 1991].

The Variant Approach. The variant approach is based on group technology (GT) concepts and uses similarities among the parts to be manufactured to retrieve existing process plans. In general, variant process-planning systems have two stages: preparatory

and production (Figure 2.4) [Chang, 1991. pp. 413]. At the preparatory stage, existing parts are grouped into families and then coded according to their similarities, attributes, features, specifications, and types of raw materials. The coding can be a tedious task, but the success of the future system depends very much on the quality of the initial coding system. Standard process plans for each part family are then prepared and stored in a database. The coded parts and their standard process plans are indexed by a family matrix. Standard plans of the parts generally contain manufacturing methods such as the type of processes to be used and the sequence of fabrication steps or operations. The production stages of the process planning function then follow. When a process plan for a new part is required, the part is first classified and coded. The code is used to retrieve a process plan from the family matrix. A human planner then helps to verify and modify the plan to meet the part design and other requirements. Optiz, MICLASS and DCLASS [Chang, 1991, pp. 368] are some classification and coding systems used in the variant approach. Among the 156 CAPP systems recorded by Alting [Alting, 1989], 108 systems use this approach.

The variant approach is particularly useful in handling process planning which involves products with a lot of similarities as the development effort is minimal. However, one of the limitations of this approach is that the quality of the generated process plan depends on the respective standards of the existing plans and the experience of the human process planners. As a result of the rapid advance and complexity of manufacturing technology, it is very difficult for process planners to keep their knowledge constantly up-dated and to take all the relevant factors into account during the final process plan formulation stage. Also, the task of updating the standard plans in data-base to reflect the dynamically changing production environment is tedious and time consuming. In view of such drawbacks, the variant approach is not suitable for use in a fast moving and dynamic production environment.

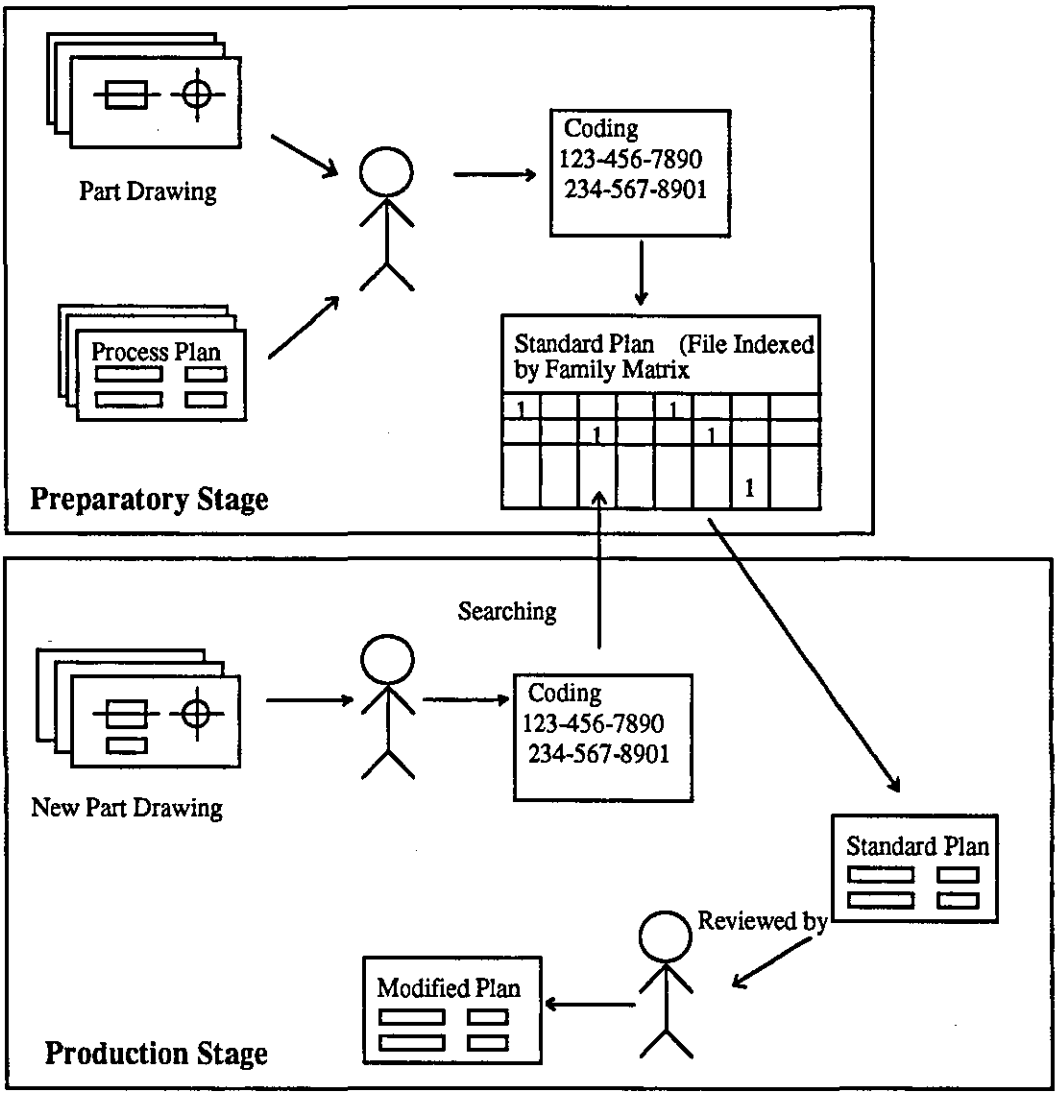


Figure 2.4 Variant Process Planning System (modified from Chang 1991 pp. 413, 414)

The Generative Approach. In the generative approach, no process plan is stored in the data base. Instead, new plans for the product are generated individually based on planning knowledge and heuristics such as decision logic, formulae, algorithms, equipment capabilities, process specifications and customer requirements. The development of such systems is difficult, time consuming and extremely complex because all manufacturing and planning knowledge must be efficiently captured, encoded into and retrieved from

software. However, the significant advantage of this approach is that process plans with a high degree of consistency for completely new parts can be generated without reference to existing plans. Furthermore, modification on the knowledge and heuristics being used can be performed to reflect the dynamically changing manufacture environment.

However, the involvement of the human process planner in the transformation of part data and the decision rules into a computer readable format needs to be overcome before achieving a fully operational generative planning system. In fact, most of the generative process planning systems developed is not truly generative in the strict sense because the product features extraction process is only partially automated. Human intervention and interaction is still required for the generation of the final process plan. Also, a lot of "local" knowledge such as information on in-house machines and tooling in use, process limitation and capability, in-house practice and cost data needs to be extracted and added to the knowledge base before such a system can be workable in the industry environment.

2.7 CAPP Researches in Various Manufacturing Operations

According to Powers [Powers, 1987], major manufacturing operations can be classified into three main types, namely, machining, assembly and process.

2.7.1 Machining

Machining is basically a subtraction fabrication process. Typical machining operations are characterised by the involvement of a variety of materials, precision specifications and complicated curves or surfaces as the sizes of the parts vary greatly. Most CAPP researchers concentrate on the machining operation such as milling, drilling and sheet

metal work. Of the 156 CAPP systems recorded by Alting, 150 of them were related to machining [Alting, 1989].

2.7.2 Assembly

Assembly is the largest single type of operation in manufacturing industry because most end products have to undergo some assembly before their completion. Assembly operations usually involve many different types of parts and material handling. Parts have various features; difference in design and bills of materials and testing/inspection operations are not uncommon. Due to the complicated process of assembly, assembly is generally performed by people as they are more flexible and cost effective. Industrial engineers usually assume the responsibility for the generation of the assembly methods/plans because of their knowledge in time and method study. As more assembly operations are now undertaken by automatic pick and place machines and robots, automating the generation of assembly plans is desirable. Some researches in the generation of assembly plans, particularly in printed circuit assembly, have begun [Browne, 1991] [Chiu, 1991] [Rondeau, 1990] [Segre, 1988] [Srinivasan, 1991] [Wong, 1993] [Lam, 1994]. However, there is still much to be done before the complete generation of process plans from product design data and their direct integration into the manufacturing system.

2.7.3 Process

Manufacture and production of chemical, pharmaceutical, metal, food, electronic components all involve process operations. Although these products have their individual differences, manufacturing processes involved have very similar features: supply of raw materials, continuous material flow through the various process steps, use of different,

complex production equipment, good quality material and required tight control process parameters. Plating, welding, soldering, etching, casting and heat treatment are some typical process operations. Although researches are being carried out in recipe generation and process control in integrated circuit (IC) processing in the processing industry [Budge, 1990] [Chandraker, 1990] [Fu, 1988] [Fu, 1989] [Guldi, 1989]) [Pillai, 1990] [Sachs, 1991], research on process planning in other process operations such as PCB manufacturing process is unusual.

2.8 Significant Functions in an Ideal CAPP System

Using an artificial intelligence system (AI) framework, the process planning problem can be formulated into a sequence of goal searching actions based on sets of initial state and constraints. Various researches and studies have identified the following functions of an ideal CAPP system [Ham, 1988] [Lee, 1991] [Tsang, 1986] [Kusiak, 1990, 1991]:

- interpretation of part design data
- representation of part data
- selection of manufacturing processes
- selection of machine tools and fixtures
- process/machine optimisation
- sequencing operations
- determination of jig and fixtures
- determination of production specifications and tolerances
- determination of the machining/processing conditions
- calculation of the machining/processing/operation time
- generation of process recipes/process sheets/NC codes

As the knowledge required for the planning process is usually qualitative in nature, a knowledge-based systems approach [Kumara, 1986] [Kanumury, 1991] [Kusiak, 1988] is identified to offer an appropriate framework for the problem. This thesis presents such a system for the process planning of the manufacture of bare printed circuit boards.

2.9 Summary

This chapter has discussed the various manufacturing processes for Printed Circuit Board and explained the circuit board manufacturing methods. The possible use of CAPP systems to assist the process planning of circuit boards has been identified for further exploration in the following chapters. Then, the importance of developing CAPP systems in the computer integrated manufacturing environment is justified. CAPP researches in different manufacturing areas are revised and the functionality of an ideal CAPP system is discussed.

CHAPTER 3

KNOWLEDGE-BASED CAPP SYSTEMS AND OBJECT ORIENTED MODELLING TECHNIQUES

3.1 Introduction

This chapter introduces what a knowledge-based system is and explains how knowledge-based systems can be used in computer aided process planning. Object orientation and object oriented analysis and design techniques are discussed. The Object Modelling Technique (OMT) approach for system modelling and development is, then, presented. Use of OMT for modelling and development of a knowledge-based computer aided printed circuit board process planning system is justified.

3.2 Knowledge-Based System [Badiru, 1992] [Kusiak, 1990]

A knowledge-based system is a computer program that simulates the thought process of human expert to solve complex decision problems in a specific domain. In this project, the following definition (modified from Badiru [Badiru, 1992]) of a knowledge-based system is adopted.

> "A knowledge-based system is an interactive computer-based decision tool that uses both facts and heuristics to solve difficult decision problems based on knowledge acquired from experts and other logical sources."

A typical structure of a knowledge-based system is shown in Figure 3.1. The components include: user interface, inference engine and working memory, knowledge base and external interfaces unit.

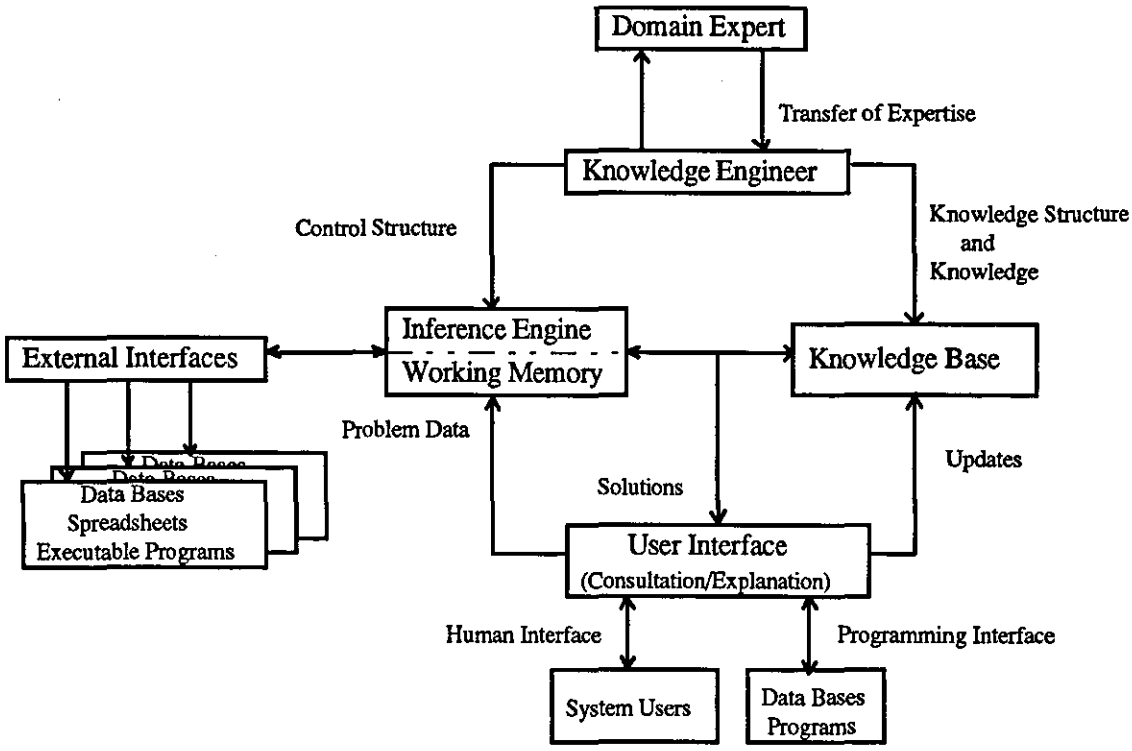


Figure 3.1 A Typical Architecture of a Knowledge-Based System
(Modified from Badiru 1992, p. 21)

3.2.1 User Interface

The user interface allows the user to communicate with the Knowledge-Based System (KBS). Using the interface, users can define the problem to the system. The system captures the problem description in a pre-defined and precise way. The user interface also

presents solutions to the user. Some KBSs have programming interface ability in the user interface units and can extract problems directly from data bases.

3.2.2 Inference Engine and Working Memory

An inference engine incorporates reasoning methods, which act upon the input problem data and the specific knowledge in the knowledge base to solve the particular problem and produce an explanation for the solution presented if it is required. In a knowledge-based system, the inference engine examines facts and executes rules in the knowledge base according to selected inference and control procedures. Rules usually take the form of "IF _____ THEN _____" structure which means that if the conditions after the "IF" portion are fulfilled, the statements after the "THEN" portion are then executed. There are basically two different types of control procedures that invoke the inference rules: forward chaining and backward chaining.

Forward chaining. The procedure starts with a set of facts (or data) and looks for those rules in the knowledge base for which the "IF" portion matches the facts. When such rules are found, one of them is selected based upon an appropriate resolution criterion and executed. This generates new facts in the knowledge base, which, in turn causes other rules to fire. The reasoning operations stop when no more new rules can be fired. This kind of reasoning is known as forward chaining or data-driven reasoning.

Backward chaining. Backward chaining is an alternative approach which begins with the goal to be proven. It tries to establish the facts needed to prove this goal by examining those rules which have the desired goal in the "THEN" portion. If such facts are not available in the knowledge base, they are set up as sub-goals. The process continues until all the required facts are found, in which case the original goal is proved; or the situation is

reached when one of the sub-goals cannot be satisfied, in which case the original goal is not proved. This method of reasoning is called backward chaining or goal-directed reasoning.

The working memory in the inference engine contains the specific problem data and the intermediates of the final results produced by the system during the reasoning process.

3.2.3 The Knowledge Base and Knowledge Representation

The knowledge base contains the knowledge of the particular problem domain, sometimes including facts, way of practices, and heuristics which are unique to individual experts. Knowledge can be divided into four different categories [Michalski, 1983]:

- knowledge about objects:	facts about objects in the world around us (e.g. fish can swim)
- knowledge about events:	including a sequence of events and their cause and effect relation (e.g. it will rain tomorrow)
- knowledge about performance:	how to do things (e.g. solving mathematical equations, picture interpretation)
- knowledge about knowledge:	or "meta-knowledge". This is knowledge about what we know (e.g. the reliability and relative importance of facts, knowledge about the inference mechanism).

Effective representation of knowledge is one of the key issues in knowledge-based systems. Representation schemes can be classified into [Kusiak, 1990] [Badiru, 1992] [Chang, 1990]: state space representations, logical representations, semantic networks,

frames, procedural representations, production systems and object oriented representations.

State Space Representation. State apace representation is the use of a set of conditions or values to describe the system during the problem solving process [Badiru, 1992, pp.272]. However, this representation method is not particularly suited for most expert systems applications as the requisite number of states will be too large to be identified completely.

Logic Representation. First-order predicate calculus is used as a means of logic representation. The description of the real world is given in terms of logic clauses. They are well defined and are easily understood. Logic representation schemes, however, have difficulty in representing procedural knowledge and limitations in managing large knowledge base because of restricted organizational structure and data manipulation [Badiru, 1992, pp. 87].

Semantic Networks. Semantic networks were developed by Quillian in 1968 as an explicit psychological model for human associative memory. Semantic nets attempt to describe the world in terms of nodes and relations. In semantic networks representation, knowledge is a collection of objects and associations represented by a labeled, directed graph. Semantic networks are easy to understand, but are difficult to implement, especially when the number of associations are complex and large in quantity. Another problem of semantic networks is the lack of a formal definitive structure in the semantic networks which makes it difficult to implement especially in complex system [Badiru, 1992, pp. 80].

Frames. A frame is a data structure for representing a stereotyped situation. It is analogous to a standard form or fixed format data file that has slots (attributes) in which

data are put. Each slot will only accept a predetermined type of data. Frames thus permits the representation of both declarative and procedural information using attributes and hierarchical relations with other frames.

Procedural Representation. In procedural representation, knowledge is contained in procedures. The procedures can be small programs that know how to do specific things or to proceed in well-specified situations. The problem associated with this scheme is the difficulties in verifying and changing a procedural representation if modifications are required.

Production Systems. Production systems are a representation scheme that is finding increasing popularity in large Artificial Intelligence (AI) programs. The basic idea of these systems is that the data base consists of rewriting rules, called productions, in the form of "IF_____ THEN_____" condition-action pairs. The production systems have been found useful as a mechanism for controlling the interaction between frames.

Object Oriented Representation. Object oriented representation provides an excellent data structure for symbolic manipulation of conceptual information. Each object is an entity which can represent both procedural and declarative knowledge. The object oriented representation is more flexible than frame representation due to its use of data abstraction, inheritance, modularity and inheritance.

3.2.4 External Interfaces Unit

The external interfaces unit allows the knowledge-based system a better integration with external programs and data bases. External customer programs may be needed to perform unique analyses such as optimization, complex computations and graphics manipulation

while data bases may be used to store passive data sets as needed during knowledge-based system consultation.

3.3 Knowledge-Based Computer Aided Process Planning System

A Knowledge-Based Computer Aided Process Planning (KB CAPP) system is a computer program that performs process planning tasks similar to human process planners. It operates from a heuristic knowledge base (not necessarily extracted from an expert) which has explicit process planning knowledge. A typical architecture for knowledge-based CAPP systems (Figure. 3.2) [Funakoshi, 1990] [Joneja, 1991] [Jumara, 1986] [Srihari, 1990] [Srinivasan, 1991] [Wang, 1988] includes the following functional blocks:

- A user/data base interface module for the planning problem description, input and output
- A knowledge base on the specific planning domain
- An inference engine of generic planning problem solving knowledge of specific domain
- A knowledge acquisition/up-dating mechanism

3.3.1 The Planning Problem Description

The planning problem description/definition stage is crucial in the use of a KB CAPP system. The user interface module allows users to communicate with the system and creates a detailed description of a specific process planning problem in a particular domain to the system. This information, any intermediate solutions, and the final results produced by the system will be stored in the working memory and interactively accessed by the inference engine until the final process plan is achieved.

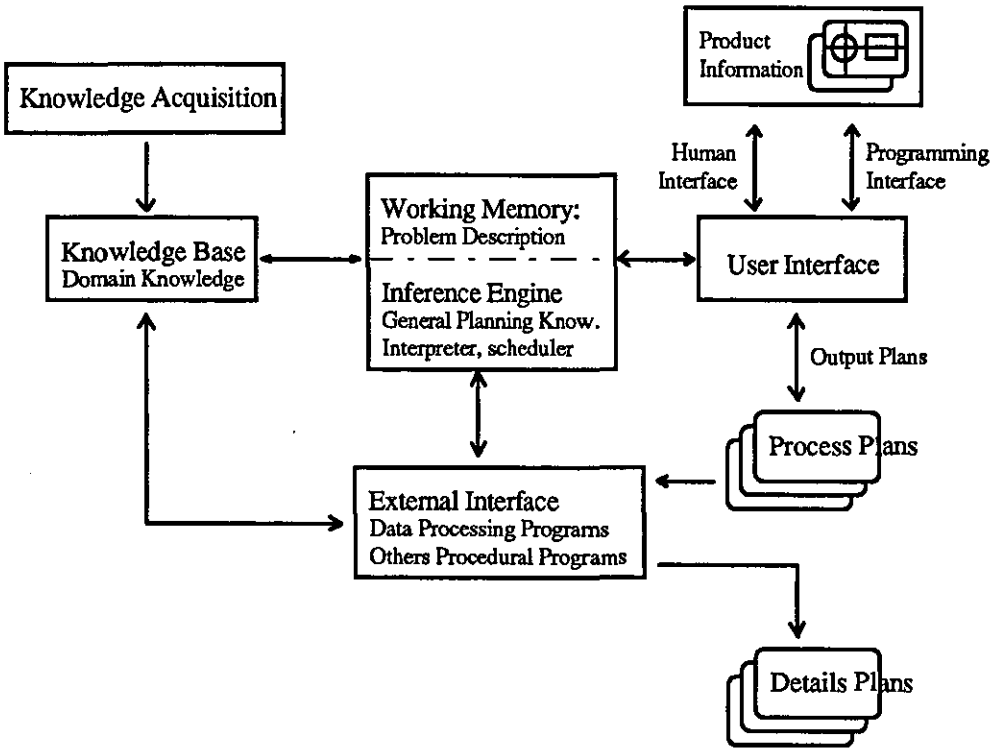


Figure. 3.2 Architecture of a Knowledge-Based CAPP System

3.3.2 The Inference Engine

The inference engine of a KB CAPP system incorporates the generic planning problem solving reasoning methods, which act upon the problem and its intermediate solutions in turn to produce the final process plan. The sequence of rule firings in the inference engine is governed by the conditions of the rules within the system. The reasoning mechanisms commonly used to interpret the domain knowledge include backward and forward inferencing.

3.3.3 The Planning Knowledge Base

The knowledge base in a KB CAPP system serves as a common repository of planning knowledge. The knowledge may be a combination of data structures (facts) and interpretive procedures (rules), including the following elements:

1. Analytical, empirical, and/or experimental models of processes, machines, equipment, etc.
2. Machine tool specification including abilities, capabilities, etc.
3. Representation of the end product.
4. Process data such as machinability data and process capability data.
5. Optimization techniques and economic models as applied to the process.
6. Manufacturing process constraints and process monitoring methods.
7. Other related planning knowledge.

The above list has been drawn from the work of Amstead [Amstead, 1979], Atling [Atling, 1982], Boothroyd [Boothroyd, 1975], Denney [Denney, 1988], Doyel [Doyel, 1969], Follette [Follette, 1980] and Yankee [Yankee, 1979].

3.3.4 User Interface

The user interface is where the KB systems interact with process planners. Planners define planning problems to the systems and/or the systems can directly extracted planning problems from product data base through programming interfaces. For example, in a PCB process planning system, part of the planning problem can be represented as circuit features such as minimum circuit width, minimum line spacing and total circuit area which

can be extracted directly from the circuit data file using a computer program. The user interface can also be used by the system to present the planning results. The results may be files that contain working procedures or operation instructions on how the product can be manufactured. Some explanation functions may also be included so that the users know the internal reasoning carried out in the KB system and how the process plans are generated.

3.3.5 Planning Knowledge Acquisition and Up-dating

The knowledge acquisition facility provides the system with the ability to acquire new planning knowledge or to up-date existing planning knowledge. Usually, the heuristic type of knowledge in a CAPP system is up-dated manually.

3.3.6 External Interface

Although the basic framework of a KB CAPP system adopts a knowledge-based approach, other problem solving approaches and programming techniques may also be embedded in the system especially in the detail process planning generation stage. Some of these approaches and programming techniques are as follows:

- determining process optimization conditions (such as the optimization of drill path [Law, 1990])
- process recipe generation (such as the generation of plating current and plating time) and,
- generation of machine coding (such as drilling and routing NC codes).

3.4 Object-Orientation

Object orientation is a software modelling concept rather than an alternative programming coding or programming language approach. The following are main themes underlying object orientation modelling and are well supported in object-oriented software systems.

3.4.1 Abstraction

Abstraction denotes the extracting of the essential details of an item or a group of items while ignoring the unessential details [Berard, 1993]. Thus, the abstraction focuses on the important aspects of the entity and ignores its "accidental" properties. Using the concept of abstraction, decisions on the implementation of the system in software can be deferred until the problem is completely understood.

3.4.2 Encapsulation

Encapsulation means the act of enclosing one or more items within a physical or logical container [Berard, 1993]. The encapsulation separates the external aspects from the internal implementation details of the object and eliminates the propagation effects of changes through the system. In other words, the application will not be affected if changes in the internal implementation of an object are made.

3.4.3 Polymorphism

Polymorphism is a measure of the degree of difference in how each item in a specified collection of items must be treated at a given level of abstraction. The polymorphism is increased when any unnecessary differences, at any levels of abstraction, within a

collection of items are eliminated [Berard, 1993]. Thus, the concept of polymorphism allows data and behavior of the sub-systems to be grouped under the same class hierarchy. As a result, it is unnecessary to write the same procedures again for similar sub-systems under the same class hierarchy. Adoption of the polymorphic approach in software development greatly reduces the burden of maintenance work because of the high degree of software reuse.

3.4.4 Inheritance

Inheritance means one object acquires (gets, receives) characteristics such as operations, knowledge of state, exceptions and constraints from another object [Berard, 1993]. For example, an inheritance of data structure and operations allows subclasses of an object to share a commonly defined structure. This simplifies the procedures for handling different objects under the same subclass.

Compared to the traditional system development approach, a system built using an object-oriented concept is more stable because the features of the object are always the same. Encapsulation is the foundation of object-oriented approach. It shifts the emphasis from coding technique to packaging in the system development stage. Inheritance built on encapsulation makes reuse of the code possible.

3.5 Object Oriented Analysis and Design Techniques

According to Lubars [Lubars, 1992] and Rumbaugh [Rumbaugh, 1991], some of the common Object Oriented Analysis Techniques include: Rumbaugh's Object Modeling Technique (OMT) [Rumbaugh, 1991], Coad and Yourdon's Object Oriented Analysis

[Coad, 1991], Shlaer and Mellor's Object Oriented Systems Analysis [Shlaer, 1988], Booch's [Booch, 1991] and Jacobsen's [Jacobsen, 1987] approaches. The following is a comparison between different object oriented techniques [Rumbaugh, 1991, pp. 266].

Coad and Yourdon's object oriented analysis is similar to OMT (to be discussed in detail later) approach but concentrates on analysis and only touches on design briefly. Shlaer and Mellor's Object Oriented Systems Analysis is also similar to the OMT approach, dividing the analysis into three phases: static modelling of objects, dynamic modelling of states and events and functional modelling. However, Shlaer and Mellor's approach is excessively preoccupied with relational database tables and database keys. The approach is also concentrated on analysis, hence a big difference may be found between the final stage and the implementation stage. Compared with the OMT approach, Booch's approach places less emphasis on the analysis stage and association characteristics in object modelling. Jacobsen's methodology is not commonly known and have limited publication.

Comparatively, the OMT approach has permeated the entire software development process - from analysis to design and to implementation. It is the most comprehensive, mature and well documented methodology which is the most suitable methodology for this project.

3.6 Object Modelling Technique (OMT)

Object Modelling Technique (OMT) [Rumbaugh 1991] is one of the object-oriented, analysis and design methodologies for object-oriented development, using graphical notations to represent object-oriented concepts. The OMT supports the entire software

life cycle, including analysis of problem requirements, building of a model for the problem domain, designing a solution for the problem and gradually implementing details in the system through a programming language or a database. This software engineering methodology provides clear guidelines for what to do at each stage of the system development.

3.6.1 The Three Models of OMT

The OMT methodology uses three models, namely an object model, a dynamic model and a functional model to describe a system (Figure 3.3). The object model describes the objects and their relationships of the system, the dynamic model the interactions among objects of the system and the functional model the data transformations of the system. Each model is applicable to all stages of the system development and acquires implementation details during the development progress.

The following is a description of the three models and the four phases of the development process based on the "Object-Oriented Modelling and Design" by Rumbaugh 1991.

The Object Model. The object model describes the static structure of the objects of a system: identity, attributes, operations of the objects and relationships among the objects. The descriptions are presented graphically in object diagrams containing object classes. The classes are arranged into hierarchies sharing a common structure and behavior and are associated with other classes. The classes also define attribute values for each object instance and the operations which each object performs or undergoes.

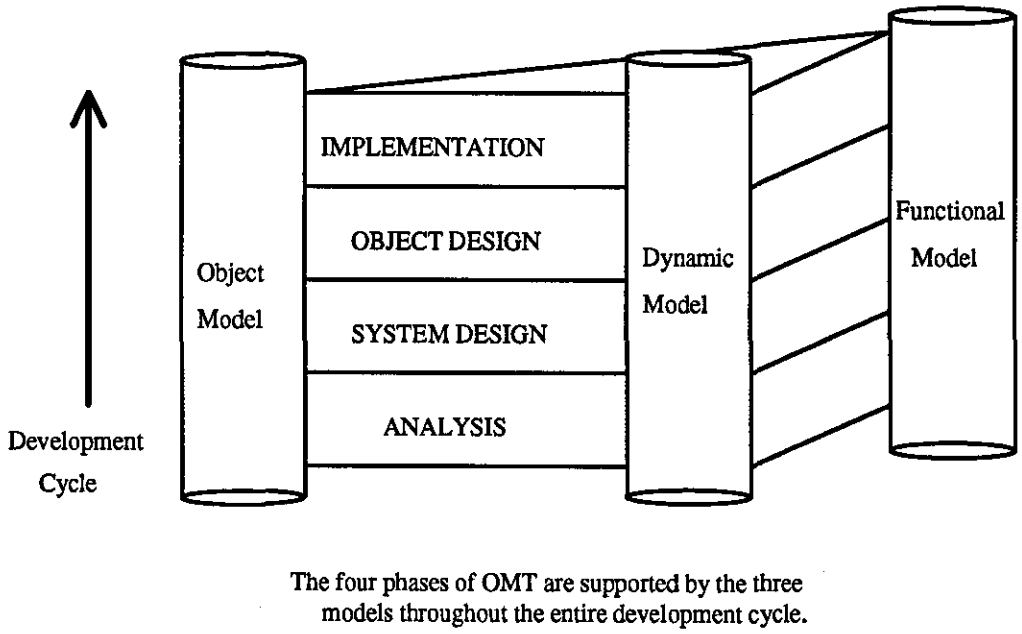


Figure 3.3 The Three Models and Four Phases of Object Modelling Technique (OMT)

The Dynamic Model. The dynamic model describes those aspects of a system concerned with time and the sequencing of operations - events that make changes, sequences of events, states that define the context for events and the organization of events and states. The dynamic model assists the developer to specify and implement the control aspects of a system. In the dynamic model, the scenario is developed first; it is a sequence of events taking place during one particular execution cycle of the system. An event is something that happens at a single point in time and has no duration. The scope of a scenario, however, can vary. It can include all events in the system or only those events impinging on or generated by certain objects in the system. After the scenario is developed, the sender and the receiver of each event are then identified. The sequence of the events and the objects exchanging events are both shown in an augmented scenario which is called an event trace diagram. In the event trace diagram, each object is shown as a vertical line and each event by a horizontal arrow linking the sender object to the receiver object. The time increases from top to bottom, but the spacing is irrelevant. Only the sequences of events,

but not their exact timing, are shown. After completion of the event trace diagram, a state diagram is developed. A state corresponds to the interval between two events received by an object and has duration. The state often involves the value of an object and the satisfaction of some conditions; those attributes that do not affect the behavior of the object are often ignored. In short, a state diagram specifies the state sequence in light of the event sequence. The actions correspond to functions in the functional model and the events become operations on objects in the object model.

Functional Model. The functional model describes those aspects of a system concerning the transformations of values - functions, mappings, constraints and functional dependencies. It captures what a system does by utilising a data flow diagram to describe the processes and the data flow. Functions are invoked as actions in the dynamic model and are shown as operations on objects in the object model.

3.6.2 The Four Phases of OMT

The OMT methodology divides the software development process into four phases, namely Analysis, System Design, Object Design and Implementation as shown in Figure 3.3. However, it does not include maintenance and enhancement after implementation in the software life cycle.

Analysis Phase. The analysis phase is the first software development process where the application expert analyses and models the application under the domain which it operates. Having stated the problem, a model is established for use throughout the analysis phase. The model describes what a desired system would do, but not how it would be done.

System Design Phase. In the system design phase, the system designer makes high-level decisions on the overall architecture. Based on the analysis model and the tentative architecture, the target system is modularised into sub-systems. The strategy built in the overall system performance is carried out.

Object Design Phase. In the object design phase, the object designer refines and "optimises" the analysis model to produce a practical design with implementation details. It can be said that the practical design is a cumulative result of the analysis model and the system design with the detail of the data structures and the algorithms required for the subsequent implementation.

Implementation Phase. In the implementation phase, the object design is translated into a particular programming language, a database or a hardware. In contrast to the previous phases, this phase is a more mechanical process than the others within the development cycle.

3.6.3 Traditional Software Engineering Approaches and OMT Approach

The most widely used, traditional software engineering development methodologies are those based on data flow diagrams. The Structured Analysis/Structured Design (SA/SD) is a typical representation of these methodologies [Rumbaugh, 1991, pp. 266]. Both OMT and SA/SD methodologies incorporate the three views of a system - the object, dynamic and functional models. However, the SA/SD methodology stresses functional decomposition and is considered primarily as providing functions to the end user. In contrast, the OMT designs are dominated by the object model where the real world paradigm of objects and relationships provides the context for understanding the dynamic and functional behavior. The SA/SD method organises a system around procedures while

the object-oriented design method organises a system around real-world object or conceptual objects existing in the user's view of the world. Therefore, changes in function can be readily incorporated into the object-oriented design by adding or changing operations, but leaving the basic object structure remains unchanged. Another weakness of the SA/SD methodology is its process of decomposing of a system into sub-systems is somewhat arbitrary. Different people can produce different system decomposition because of their background and experience. In the OMT, the decomposition is based on objects in the problem domain, so developers in the same domain tend to discover similar objects. This increases the reusability of the project components. The objects in the OMT also allow the model of data and programming code under one uniform paradigm while in the traditional procedural design approach, it is difficult to merge programming code with data. Another characteristics of the OMT are its ready resilience to change and its easy extensible ability. A software system developed by this method can be easily extended by adding objects and relationships near the system boundary.

Another traditional software development methodology is the Jackson Structure Development (JSD) approach. However, this approach is ill-suited for high level analysis and data base design [Rumbaugh, 1991, pp. 271]. The detailed abstraction of the system comprising hundreds or thousands of processes is only justified in applications where timing is important. In view of the said limitations, common software development approaches such as SA/SA and JSD are not used in this project.

3.7 The Use of OMT Methodology in this Project

According to Embley [Embley, 1992, pp. 2] (Figure 3.4), there are three approaches of system analysis: natural language analysis, process-oriented analysis and object-oriented analysis. The natural language approach's short-coming is that it relies heavily on natural language descriptions. As a result, the information about objects and their behaviors is spread over the specifications, making it difficult to match documented concepts with the real-world objects in the system. In the process-oriented approach, the system is described as a network of interacting processes, with established specifications on how these processes can be performed. This approach has the problem of steering an analyst away from studying the system components and their interrelationships. On the other hand, the object-oriented analysis approach encourages an analyst to concentrate on "what" rather than on "how" and the information is organised around objects.

As discussed in Section 3.5, the OMT methodology covers the entire software life cycle: analysis, design and implementation. The analysis is the most important phase because it shifts the design effort to the implementation again. The object model produced in the analysis phase is applied at later stages, but the boundaries of each stage of the development is invisible to the developer. The model is refined iteratively throughout the whole development life cycle, adding details to and clarifying some uncertainties of the system. The refinement of the model continues until a final satisfactory model is produced.

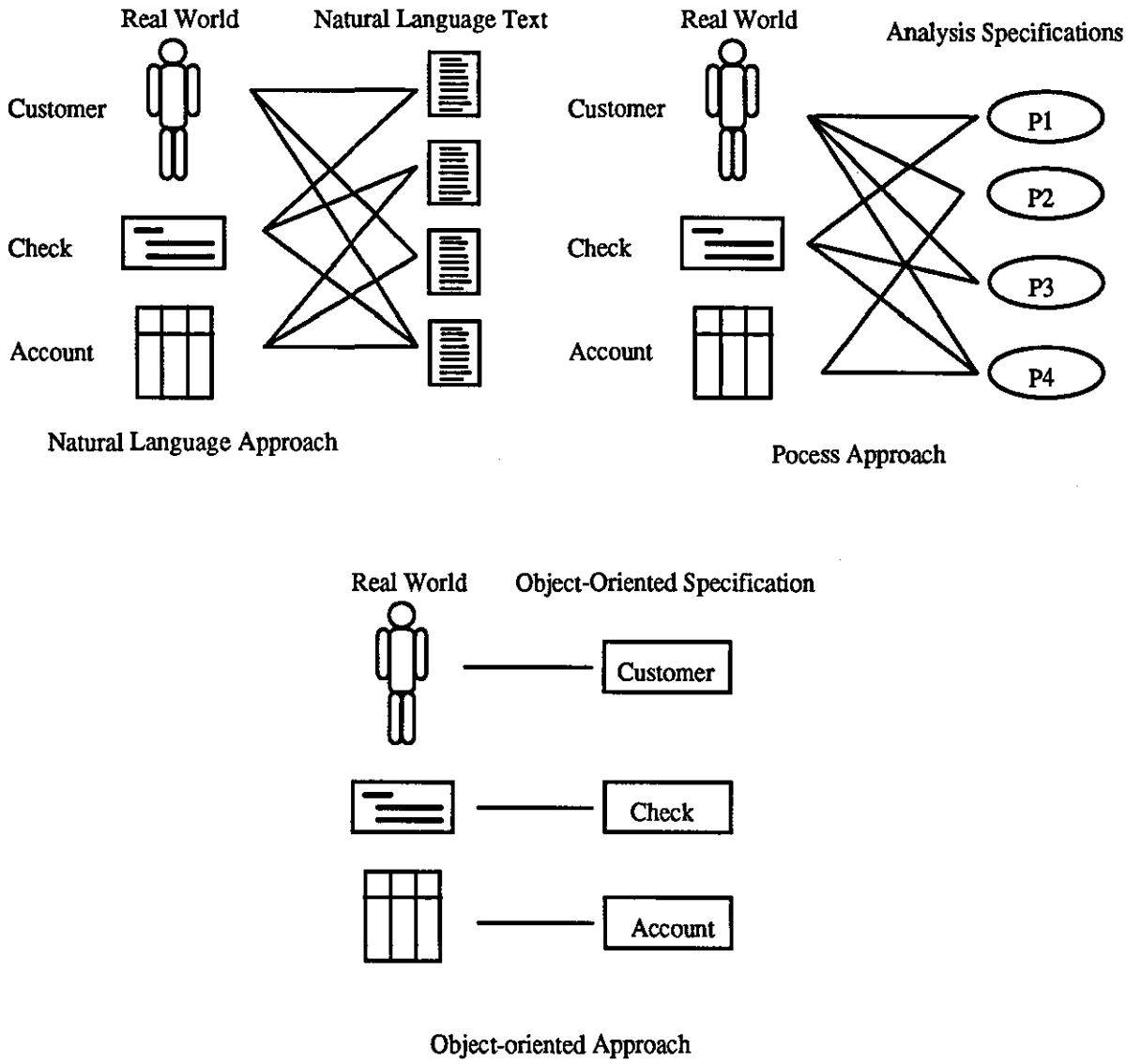


Figure 3.4 Different Types of System Analysis Approaches [Embley, 1992]

The object model of the OMT approach places emphasis on the data structure of an object and its functions, providing a more stable base for the design and allowing the unique concept of the object to be used in the entire development. Because of the superiority in the use of the OMT methodology against other approaches in the software development, the OMT methodology has been adopted in this project to develop a knowledge-based

system for the process planning of the manufacture of bare printed circuit boards. The four development phases and the three models are followed in designing the system and a foundation is provided for the knowledge engineer or the domain expert to update or review the knowledge contained in the system. The OMT methodology also facilitates the organization of the knowledge in a hierarchy way. Furthermore, applying the OMT methodology to the development of a KB CAPP system allows improvements of the system progressively; the system achieves its mature state gradually. This project is a pioneer one in adopting the OMT methodology to design a knowledge-based process planning system for the manufacture of bare circuit board.

3.8 Summary

In this chapter, a knowledge-based system has been described and how it can be used in a computer aided process planning has been discussed. Major themes of the object-orientation have been presented with an introduction of the different analysis and design techniques and a discussion about their limitations. The three models and the four phases of the OMT methodology have been described in detail. Adoption of the OMT methodology in this project is justified at the end of the chapter.

CHAPTER 4
OVERVIEW OF THE OMT MODEL OF THE KNOWLEDGE-BASED PCB CAPP SYSTEM

4.1 Introduction

This chapter gives an account of the process of developing the KB CAPP system using the OMT methodology. Firstly, the problem statement is presented. Then the three models of the system intended to solve the problem: the object model, the dynamic model and the functional model are described. The object model describes the structure of the objects of the process planning system. The dynamic model describes the time and the sequence of operations of the system while the functional model concerns with the transformations of data. The models intended to solve the problem described have been constructed by a process of refinement and modification in an iterative manner. However, only the final models are presented. The chapter ends with a discussion on the hardware and software architecture of the system developed based on the models.

4.2 Problem Statement

The problem statement is the first document produced at the analysis stage, and has the following scope and requirements:

Scope:

A computer aided process planning system that can generate process plans for the manufacture of bare printed circuit boards (PCB).

Requirements:

1. The system can be operated by technical people (not necessary experienced PCB process planners) who have minimal working experience in PCB manufacture.
2. The system can suggest feasible processes and a processing sequence for the manufacture of a specific PCB.
3. For process with more than one machine, the system can suggest the most appropriate machine(s) for the process.
4. The system can then generate a process recipe and details operation instructions accordingly.
5. Customers' general quality specifications, in-house process capability, customers' product requirements can be input into the system and stored within a data base for the system to make decisions.
6. Planning features can be extracted from circuit board CAD data files.
7. The planning knowledge or decision knowledge can be input into the system by knowledge engineers and the planning system can use the knowledge to make logical decision.

4.3 Object Model of the System

The final object model of the system is shown in Figure 4.1. Each object class is further divided into sub-classes and will be discussed in detail in later chapters. The scope and properties of each object class are described by the following data dictionary.

Customer General Specification Interface Module:

This module is used by planners to create a new customer general specification data base, to up-date any existing specification data bases, to store the defined or up-dated data into a specific data file and to load specific specification file into the system before carrying out process planning.

Product Requirements Interface Module:

This module is used for planners to create, up-date, store and load the product requirements in a similar manner to the previous interface module. In addition, this object has one more operation - the "Check with General Specification" operation which is used to check product requirements against customer general specifications.

In-house Process Capability Interface Module

This module is used to build up and review the in-house process capability data base. As the data base is embedded in the planning software system, there is no load operation in this object.

Feature Extraction Module:

This module is used to extract planning features from customer circuit and solder mask artwork data files. The data files should be presented in Gerber format. The extracted features are stored in circuit and solder mask features data files which can be accessed by the product requirements interface module automatically during each product requirement input stage.

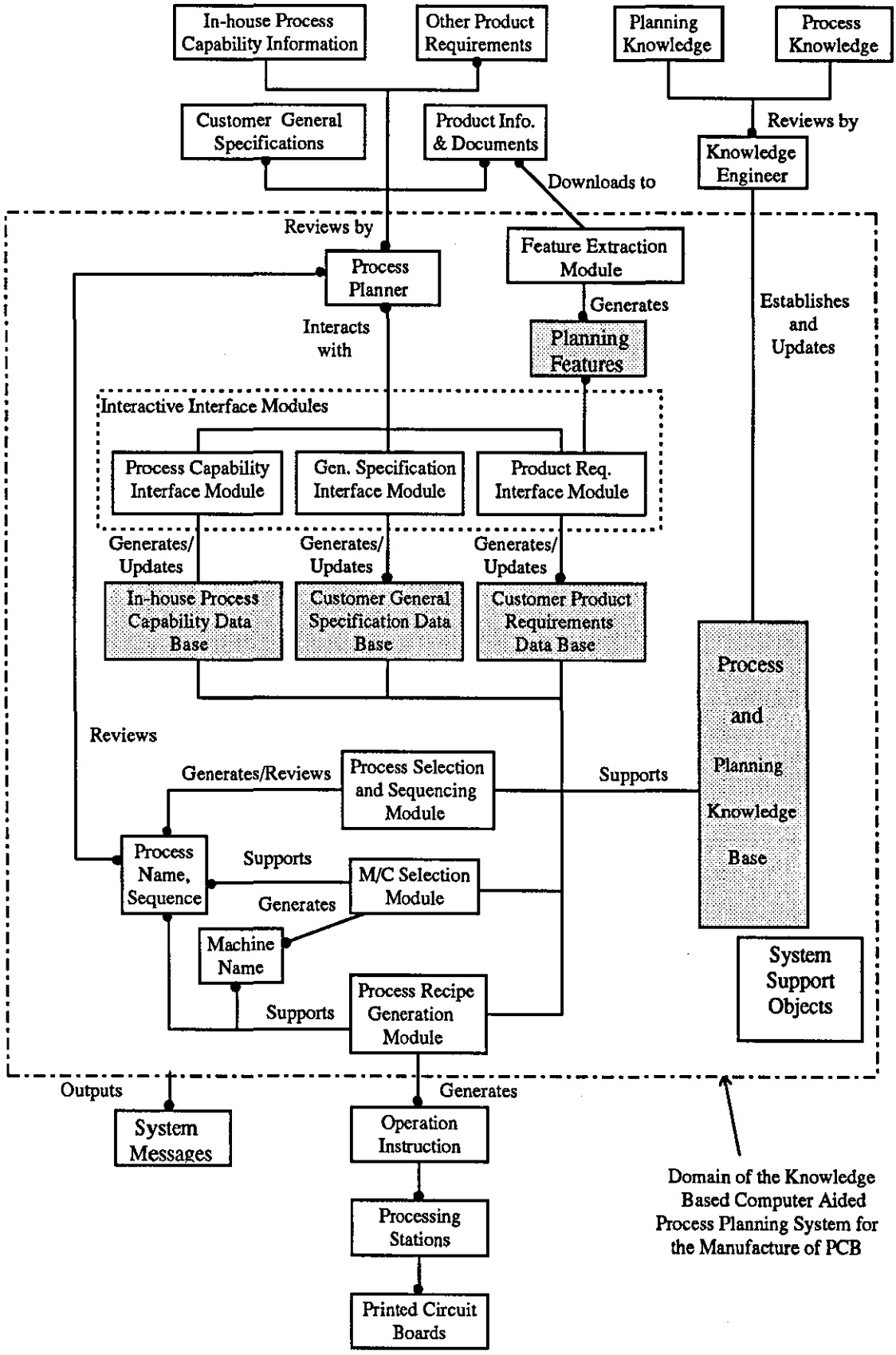


Figure 4.1 Object Model of the KB CAPP System for the Manufacture of Bare Printed Circuit Board

Process Selection and Sequencing Module:

This module generates a feasible process list with an appropriate manufacturing sequence to manufacture the required circuit board. It is capable of producing warning message(s) if the process list is not practical due to some in-house limitations. After generating a list of feasible processes, the module, then, prompts the user(s) to review and approve the suggested process list. Based on this process listing, the appropriate manufacturing sequence can be generated by the system. The process and the manufacturing sequence can be stored in a data file.

Machine Selection Module:

This module automatically selects the most appropriate machine for the processes that can be carried out on more than one machines based on machine capability, product requirements and machine selection knowledge.

Process Recipe Generation Module:

Based on the process characteristics, the type of machine and other requirements, this module generates process recipe accordingly. The process recipe is given in the form of operation instructions. The module can store the generated operation instructions in data files.

Planning Knowledge:

The planning knowledge supports the process selection and sequencing module, the machine selection module, the process recipe generation module and the general specification checking operations. The knowledge is implemented using the rule representation function of the specific software shell. The knowledge is also in modular form for effective up-dating and decision making during system execution.

Customer General Specification Data Base:

This data base contains the customer's general acceptance specification which is the data acquired via the customer general specification interface module. The data are used by all planning modules of the process planning system. The format and contents of the data base will be discussed in a later chapter.

Customer Product Requirements Data Base:

This data base contains the customer's product requirements which are the data acquired by the customer product requirements interface module. The data are also used by all planning modules in the process planning system. The format and contents of this type of data base are also covered in a later chapter.

In-house Process Capability Data Base:

This data base stores the in-house process capability of the board shop and is created, modified and up-dated using the system interactive interface module. The data are used by all modules in the process planning system. The format and contents of this data base will be fully discussed later. As all the three data bases are created, manipulated and accessed by other planning modules which are implemented by a specific knowledge-based software shell, the data base formats are in the same software syntax.

Planning Features File:

This file stores the planning features generated by the feature extraction module. Details on how the feature extraction module functions and how the planning feature file is orgainsed will be covered separately in the feature extraction section later.

Process Name and Manufacturing Sequence Data File:

This file stores up the feasible process names and manufacturing sequence generated by the process selection and sequencing module of the system. Users are allowed to review the processes suggested by the system. The manufacturing sequence of the approved processes will be generated by the planning system automatically.

Operation Instruction Data File:

This file stores the board requirements, suggested and reviewed process with manufacturing sequence to manufacture the board, appropriate machine(s) in processes and the process settings.

4.4 Dynamic Model of the System

The KB CAPP system has few dynamic features and during the system execution, there is very little real time interaction with the user. The states of the in-house process capability data base, the customer general specification data base and the planning knowledge are stable, but the states of the customer product requirements data base and the planning features data file will be different from product to product. However, once the customer product requirements are loaded into the system and the planning features are extracted by the feature extraction module, the respective modules, data bases and data files can be classified as passive objects which do not exchange events. Their state transition diagram therefore need not be constructed. However, if the whole KB CAPP system is considered, a scenario can be constructed (Figure 4.2).

The user selects customer general specification data base.

The user selects product requirements data base.

The system displays system messages to user.

The system displays the recommended process name.

The user reviews and approves the recommended process name.

The system displays the recommended process sequence.

The user asks the system to select machine for the process.

The user asks the system to generate process recipe and operation instructions.

The user creates/up-dates various data bases.

The user asks the system to read planning features from data file.

The user executes external feature extraction program.

Figure 4.2 Scenario of the KB CAPP System

After that, the event flow diagram is constructed and is shown in Figure 4.3.

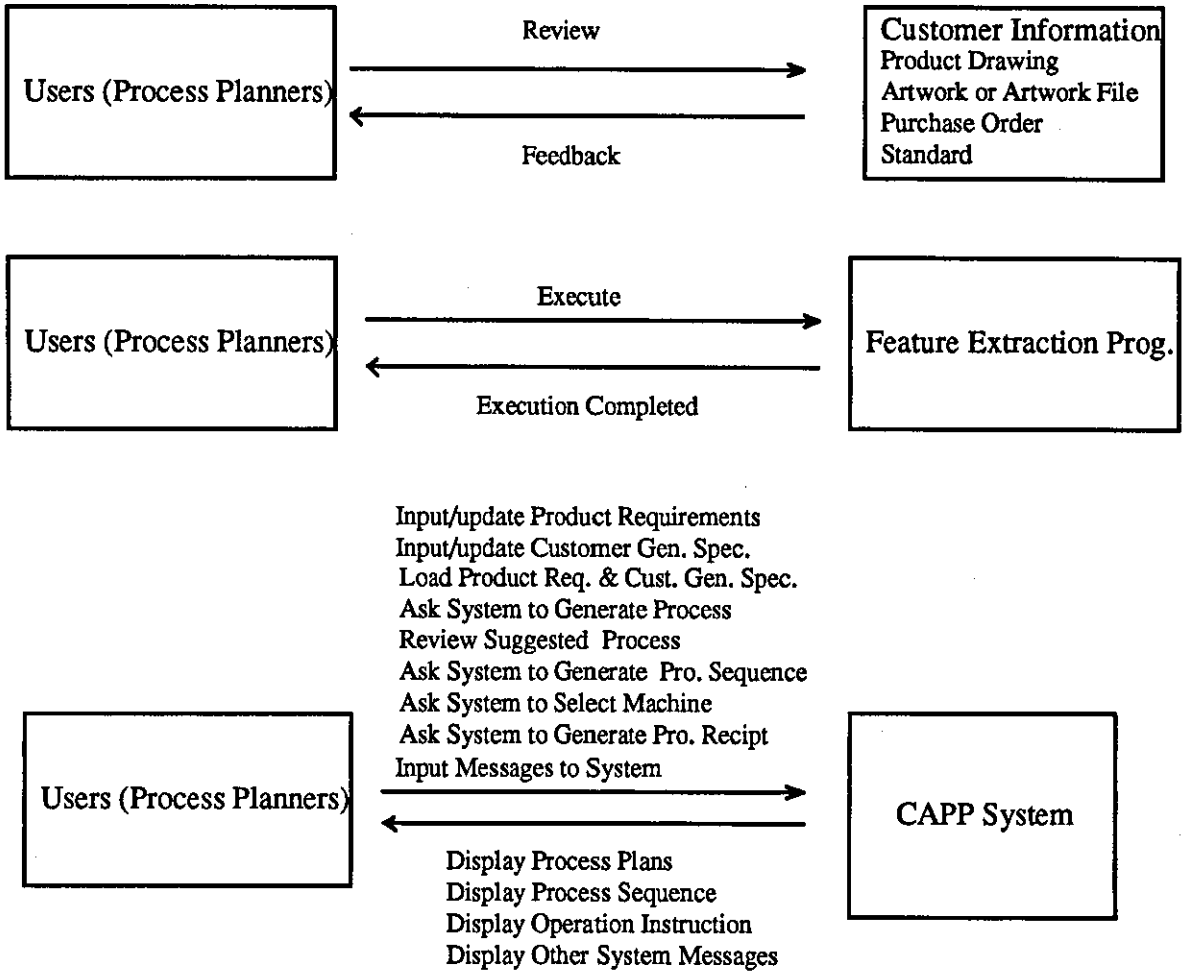


Figure 4.3 Event Flow Diagram of the KB PCB CAPP System

After the construction of the event flow diagram, the state diagram of the KB PCB CAPP system is then established (Figure 4.4). The state diagram relates the events and the states of the system. The sequence of actions performed by the actors of the system is also represented by this diagram.

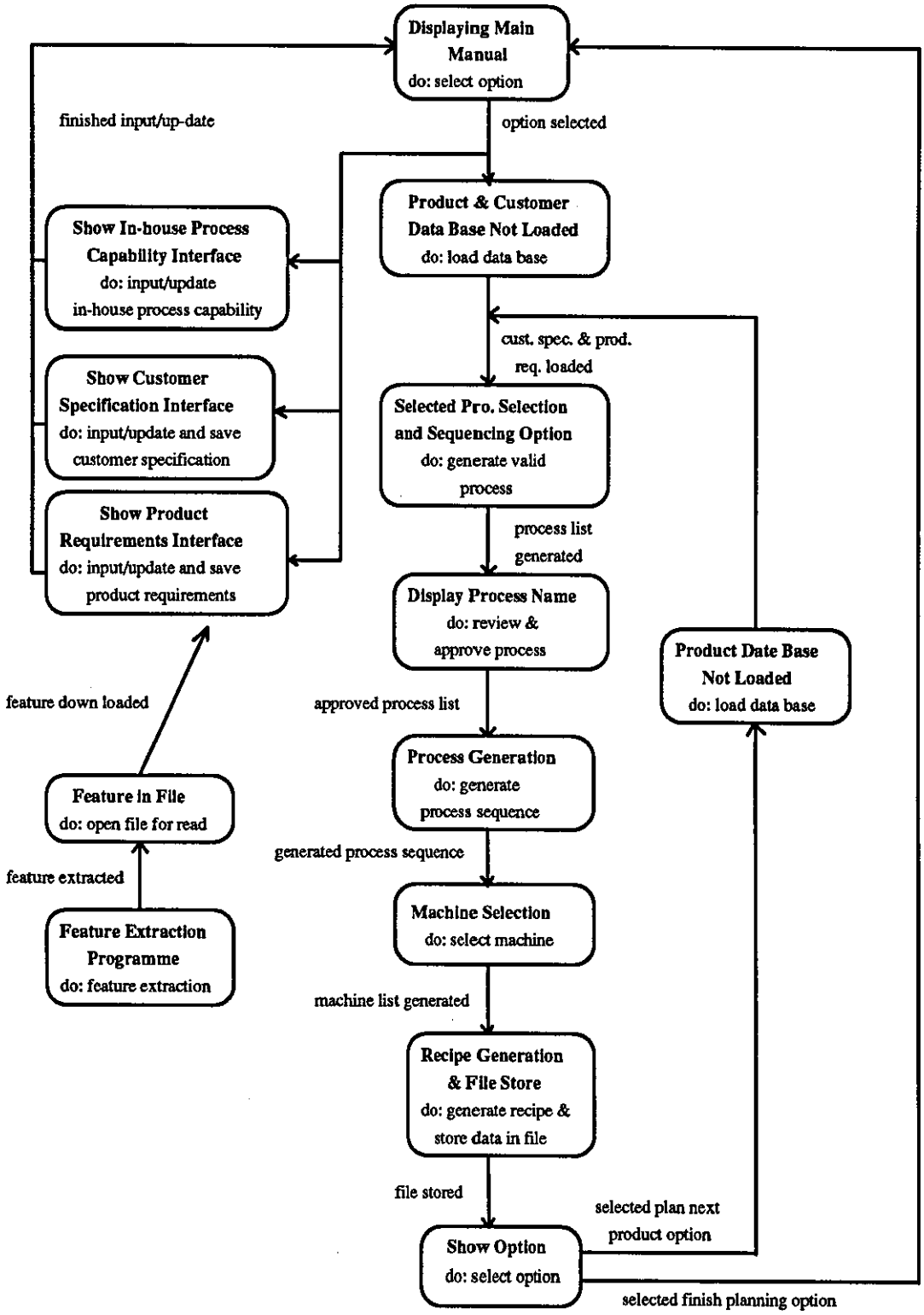


Figure 4.4 State Diagram of the KB PCB CAPP System

The state names are shown in boldface in the rounded boxes while the activities are indicated in the state boxes by the keyword "do:" followed by the names or the descriptions of the activities. Event names are indicated alongside the transition arrows. The input/update of the interface modules, load data base, generate valid process name, review and approve process name, generate process sequence, select machine, generate process recipe, store instruction in data file and extract feature are all the major actors that exchange events.

4.5 Functional Model of the System

The functional model of the system is modelled using Data Flow Diagram (DFD). It is established after building up the object and dynamic models. Each of the processes in the DFD corresponds to the actions in the state diagram of the KB PCB CAPP system (Figure 4.5). The processes are represented by ellipses containing descriptions of the data transformation. Each process has a fixed number of input and output arrows (data flow) which carries value of a given type. A pair of parallel lines is used to represent the name of the data stored (for example the customer specification and product requirements data bases). Data storage does not generate any operations on its own but merely responds to requests for storing and accessing data. The data of the customer general specification and product requirements in respective documents and files are reviewed by respective planners. Data base objects are then created or up-dated using the respective interface modules. They are stored as data files in specific knowledge-based software language format. The dates are then retrieved from the data files and loaded into the planning system for the generation of process plan. Files can be used to store the process names and their sequence after approved by the system users. Then appropriate machine are selected

and a process recipe is generated. The operation instructions are stored in the operation instruction data file.

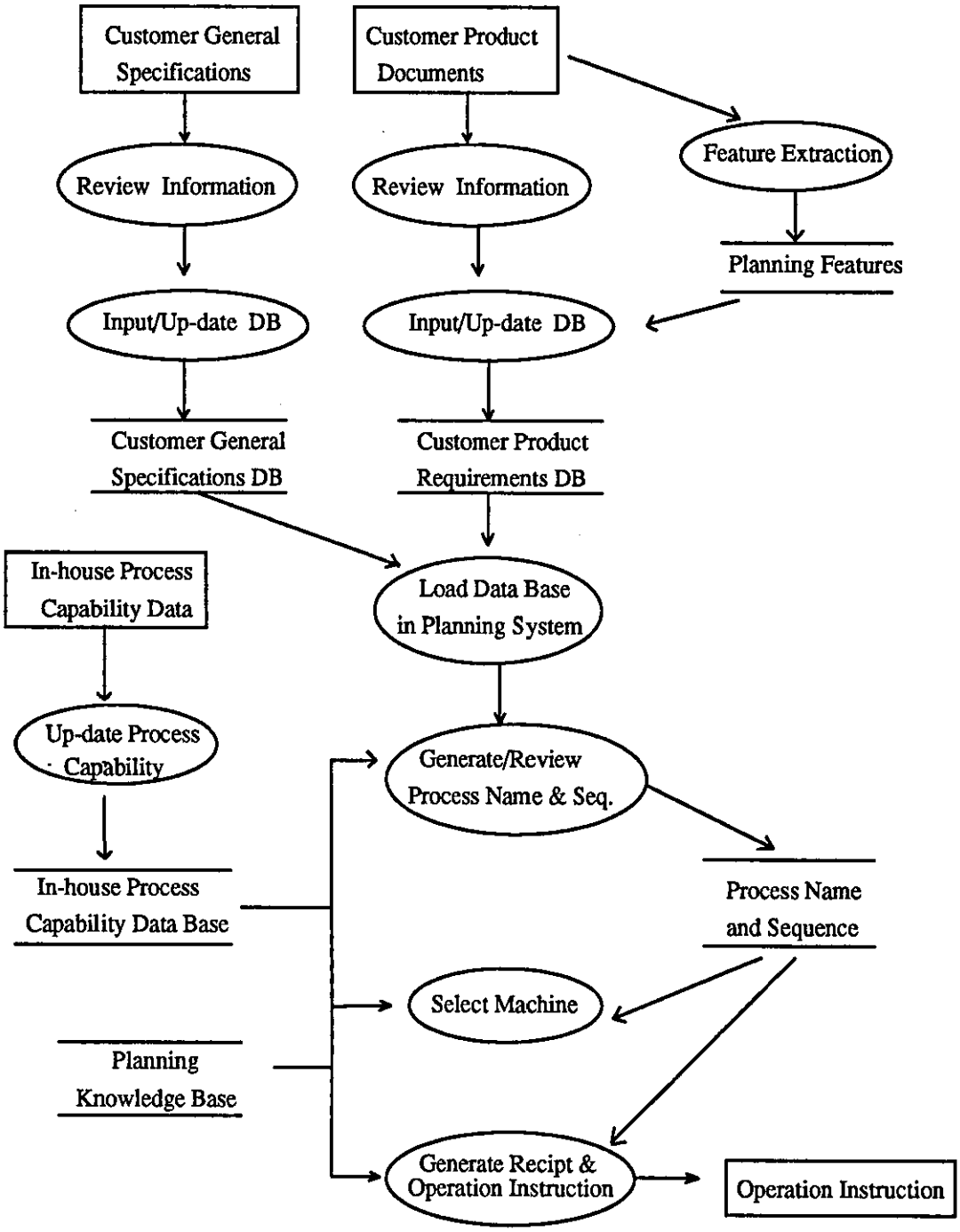


Figure 4.5 Data Flow Diagram of the KB PCB CAPP System

4.6 Hardware and Software Architecture of the System

Based on the above models, the system software architecture is developed and shown in Figure 4.6. The hardware platform selected is a 486 type IBM PC. The software in use includes "Microsoft Window 3.1", knowledge-based shell "Kappa-PC" and Borland C++ programming language.

The hardware platform used is a 486 type IBM PC because it supports the two selected software. A PC is more affordable than a workstation and it is commonly accepted and used in industry.

The knowledge-based shell Kappa-PC is selected because it supports object oriented programming [Intellicorp, 1992]. The implementation of the software system is nearly a one-to-one mapping from the object model as described. The software also supports rule-based reasoning in that expertise, heuristics and rules of thumb can be incorporated into the software system. A set of powerful graphical developer interfaces including browsers, editors, layout tools, a language interpreter and a debugger is provided to speed up the software development cycle. Another advantage of the software is its excellent graphical tools for communicating solutions to the end users. A set of object-oriented graphical tools including forms, images, meter, dialog boxes and various button is available to make software development can be done at high level. The Kappa-PC also provides a set of more than 300 pre-defined high level function libraries so that the program can be developed at a high level of abstraction. Interfacing with external data sources and software such as databases, spreadsheets, programming languages, graphics packages and ASCII files is also supported. The software also allows integration with Dynamic Link Libraries that are written in C or C++ language and can be translated into "ProKappa"

which can be run on the workstation level perform such as Sun Sparc. In view of the said strengths, the Kappa-PC is used as the knowledge-based software shell for the system.

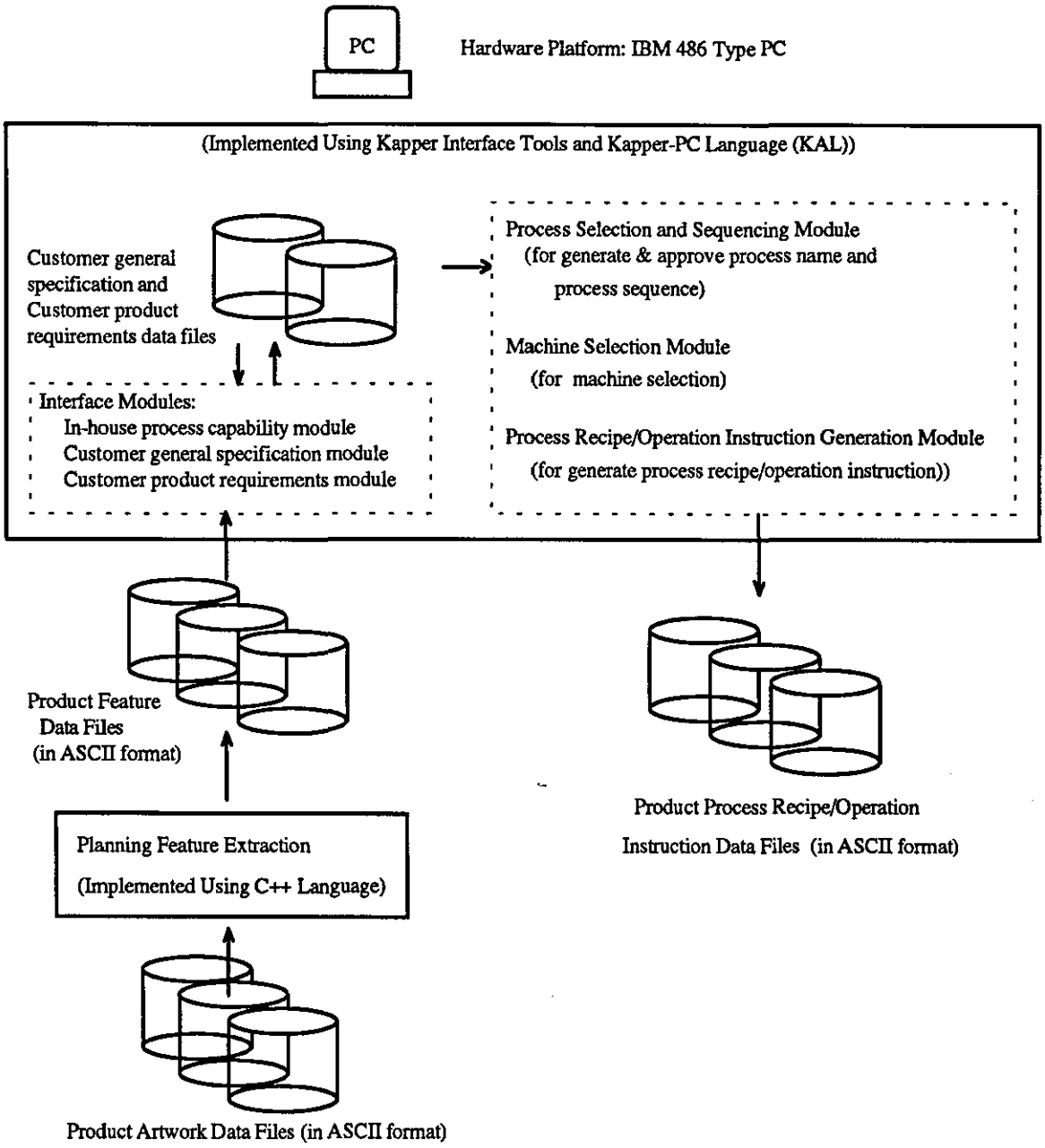


Figure 4.6 Hardware and Software Architecture of the KB PCB CAPP System

For the feature extraction program, a number manipulation language C++ is chosen because most of the algorithms are of number crunching type. Furthermore, the C++ language is more portable and supports object oriented programming concept.

4.7 Summary

The process of developing the KB PCB CAPP system using the OMT methodology is described in this chapter. The problem statement is stated first. The object model, the dynamic model and the functional model of the system are presented together with detail discussion. Based on the models, the hardware and software architecture of the planning system are established and justified.

CHAPTER 5

MODELLING & IMPLEMENTATION OF THE INTERFACE MODULE AND THE AUTOMATIC FEATURE EXTRACTION MODULE

5.1 Introduction

The chapter starts with a discussion on the object model, the dynamic model and the functional model of the interface module. Then the low level object models of the three sub-modules: the in-house process capability interface module, the customer general specification interface module and the customer product requirements interface module will be described. The definitions of the three respective data base structures will be defined in detail. Implementation of the three interface modules using the Kappa software shell will be presented and the operation of these modules will be discussed with the support of examples. The chapter follows with a description of the modelling of planning features and the feature extraction module. Various planning feature objects are discussed and presented. Then the data flow model of the features in the features extraction module is described. The chapter closes with an explanation of how the features extraction module is implemented in the C++ programming language and how the automatic feature extraction module successfully extracts features from circuit and solder mask data file.

5.2 The Three Models of the Interface Module

The object model of the interface module is presented in Figure 5.1. The in-house process capability, customer's general specification and requirements of the products are all modelled as object classes. The other object classes include engineers and technical people who operate the interface modules and the three data base structures which are generated and manipulated by the interface modules. There are three sub-level interface object classes: the in-house process capability interface object class, the customer general specification interface object class and the product requirements interface object class. Each of them will be discussed in detail in the following sections. As the data in customer general specification and the in-house process capability is not frequently changed and in order to maintain the integrity of data, only selected engineers are authorized to execute these two modules and change the data in the data base. On the other hand, the customer product requirements interface module is expected to be executed by planners or technicians with limited knowledge of PCB process planning. The three interface object classes will have different operations in order to manipulate the respective date base objects.

The dynamic model of the interface module is relative simple and as a result, only the state diagram of it is developed and presented (Figure 5.2). The first major event in this model is the initial input of the data base. If the data base needs up-dating subsequently, the interface module will be executed again.

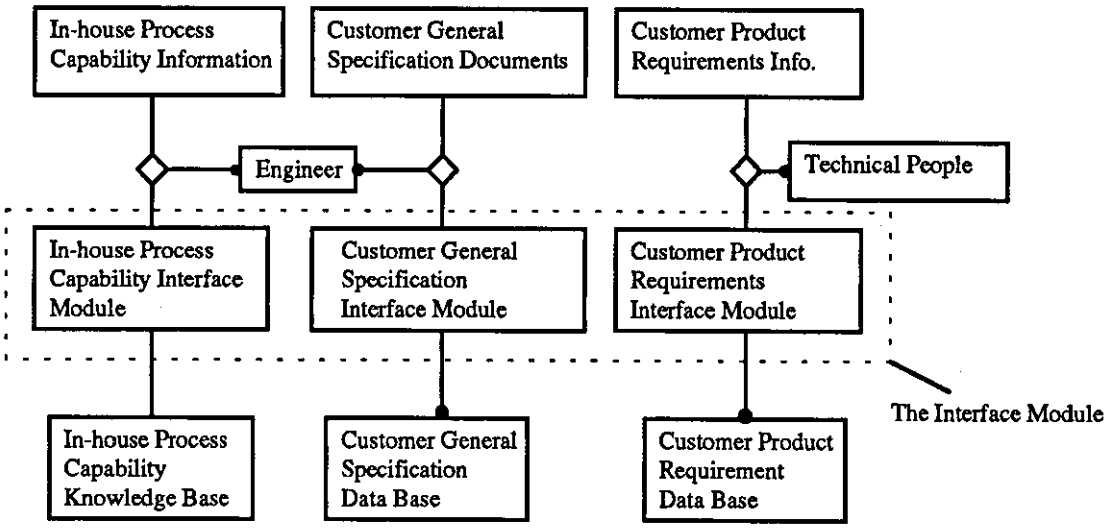


Figure 5.1 Object Model of the Interface Module

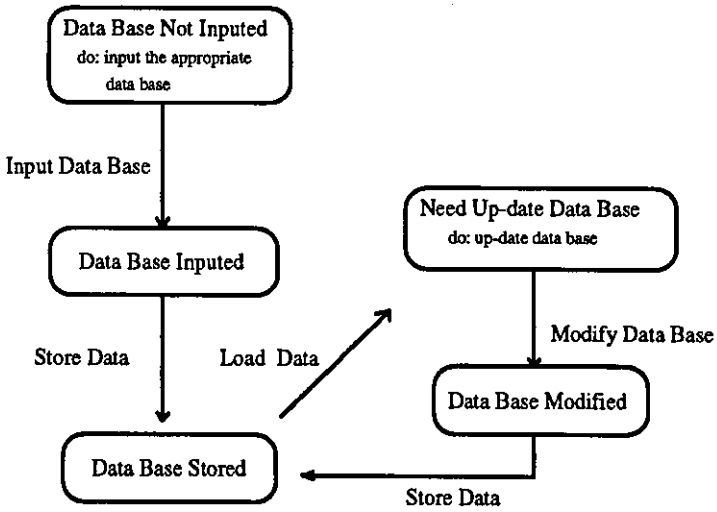


Figure 5.2 Dynamic Model (Using State Diagram Notation) of the Interface Module

The functions of the interface module are modelled using a data flow diagram (Figure 5.3). The actors of the model are the in-house process capability data, the customer general specification data and the customer product requirements data. The processes include inputting data, storing of data and up-dating of data. The processed data will flow through the interface module and then be stored as different data base for later access. When there are changes in these data actors, data will be reviewed and flow through the up-dating process and be stored in the data base again.

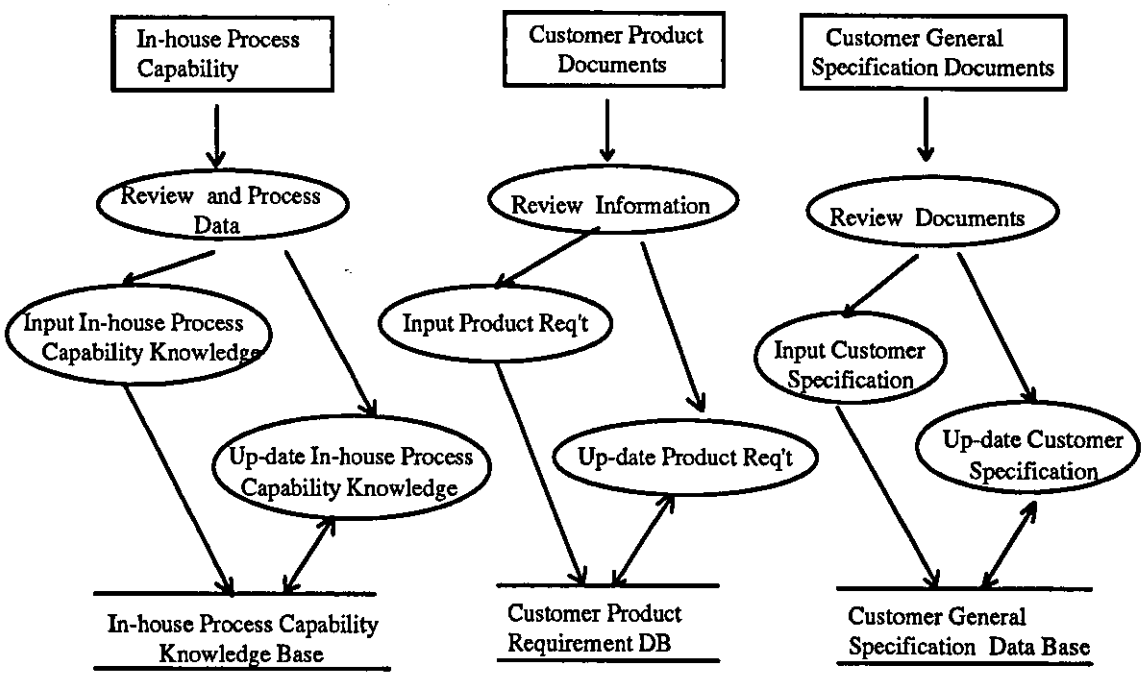


Figure 5.3 Functional Model (Data Flow Diagram) of the Interface Module

5.3 Object Modelling and Implementation of the In-house Process Capability Interface Module

The object models of the in-house process capability interface module and its knowledge base are developed and presented in Figure 5.4. There are five object classes: the in-house process capability facts object class, the engineer object class, the graphic interface object class, the in-house process capability knowledge interface object class and the knowledge base structure object class. The graphic interface object class is used to interact with the process engineer to extract process capability facts and establish/modify the knowledge base. The process capability knowledge will be stored up by respective knowledge object classes, each of which will be discussed in detail in the following sections. The "up-date" operation in the interface object will be used to up-date individual in-house process capability knowledge object class as needed.

5.3.1 Object Classes of the In-house Process Capability Knowledge Base

Each object class of the process capability knowledge base is discussed in detail in the following:

Documentation Control Object Class (Abbreviated as "*DocCont*")

This object class is used to store the data describing by who and when the knowledge base is modified. The two attributes in this object class are: the revision date ("*RevDate*") and the name of the person who revises the capability information ("*RevBy*").

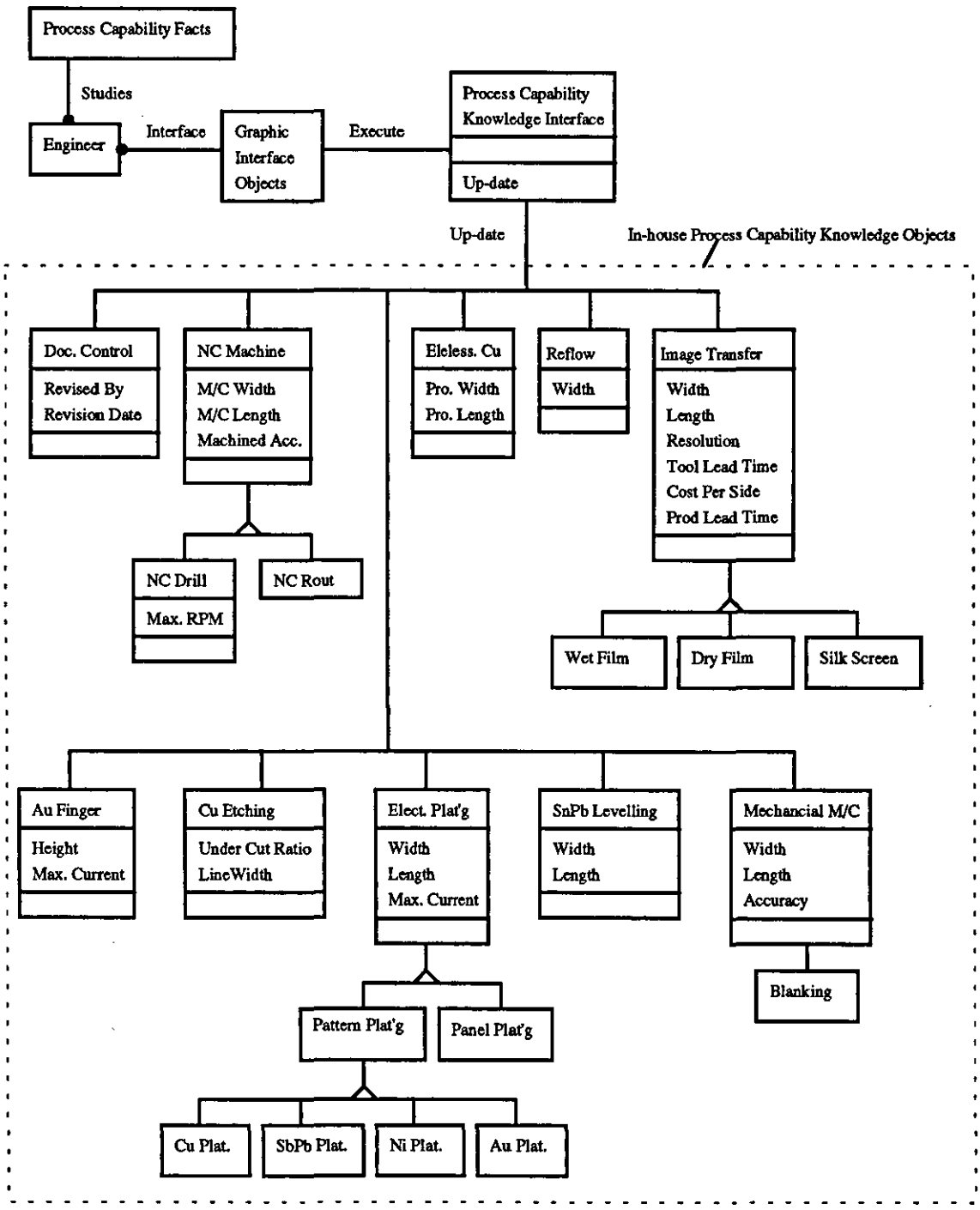


Figure 5.4 Object Model of the In-house Process Capability Interface Module

NC Machine Capability Object Class (Abbreviated as "*InNCMachine*")

This object class stores the in-house machine capability and has three attributes: the maximum machine width ("*NCWidth*"), the maximum machine length ("*NCLength*") and the machined part accuracy ("*MachinedAcc*"). Under the NC machine object, two more object classes are created. They are the NC drilling machine object class ("*InNCDrill*") and the NC routing machine class ("*InNCRout*"). Only one attribute, the maximum revolution per minute (RPM) ("*MaxPRM*"), is in the NC drilling machine object class. Other attributes are inherited from the parent NC machine capability object class.

Electroless Copper Plating Capability Object Class (Abbreviated as "*InElelessCu*")

This object class stores up the in-house electroless copper plating capability. The maximum width ("*Width*") and the maximum height ("*Height*") of the copper plating process are the two attributes of this object class.

Electrical Plating Capability Object Class (Abbreviated as "*InElecPlat*")

This object class is created to provide attributes for the low-level objects. The attributes of this object are: the maximum width of the process ("*Width*"), the maximum length of the process ("*Length*") and the maximum plating current the process can provide ("*MaxCur*"). There are two object classes under the "*InElecPlat*" object class: the pattern platting object class and the panel platting object class. Four more sub-object classes are in the pattern plating object. They are: the copper plating object class ("*InCuPlat*"), the tin-lead plating object class ("*InSbPbPlat*"), the nickel plating object class ("*InNiPlat*") and the gold plating object class ("*InAuPlat*"). All these object classes inherit the parent class attributes.

Image Transfer Process Capability Object Class (Abbreviated as "*InImageTran*")

This object class is created to provide attributes for the three low level object classes: the wet film object class ("*InWetFilm*"), the dry film object class ("*InDryFilm*") and the silk screen object class ("*InSilkScreen*"). Attributes of the "*InImageTran*" object class include the maximum width of the process ("*Width*"), the maximum length of the process ("*Length*"), the best resolution of the process ("*Resolution*"), the initial production lead time of the process ("*FirstLeadTime*"), the production cost per side of the process ("*CostPerSide*") and the production lead time of the process ("*ProdLeadTime*").

Copper Etching Capability Object Class (Abbreviated as "*InCuEtching*")

Copper etching capability details are stored in this object class. The attributes of this object class include the under cut ratio ("*UnCutRatio*") and the minimum line width the process can produce ("*LineWidth*").

Tin/Lead Reflow Process Capability Object Class (Abbreviated as "*InReflow*")

In this object class, the only attribute is the width of the reflow machine ("*Width*").

Gold Finger Plating Process Capability Object Class (Abbreviated as "*InAuFing*")

The attributes of this object class include: the maximum height of the gold finger the process can produce ("*Height*") and the maximum current the process can provide ("*MaxCur*").

Tin/Lead Levelling Process Capability Object Class (Abbreviated as "*InSnPbLevelling*")

The maximum width ("*Width*") and the maximum length ("*Length*") are the two attributes of this tin/lead levelling process capability object class.

Mechanical Machine Process Capability Object Class (Abbreviated as "*InMechMC*")

There are three attributes in this object class: the maximum width of the machine ("*Width*"), the maximum length of the machine ("*Length*") and the accuracy of the machined parts ("*Accuracy*"). At present, there is only one blanking process object class ("*InBlank*") under this object. Other mechanical machine classes such as slotting machine and V-cut machine classes can be added as required later.

5.3.2 Implementation of the In-house Process Capability Interface Module

The interface object and the knowledge base objects have been implemented using the Kappa software shell. The implementation is based on the object model, the dynamic model and the functional model as presented. The process capability interface object and the knowledge base objects hierarchy are created and presented in Figure 5.5. It shows the hierarchical relationships between the parent object and the different child(ren) object(s).

The implementation of the knowledge object "*InNCDrill*" is shown in full in Figure 5.6. The hierarchy of the object, the attributes of the object, the prompt messages when the attributes of the object are accessed and the instance values of the attributes are all shown. The whole Kappa Application Language (KAL) implementations of the "In-house Process Capability" objects are shown at Appendix 1.1. The KAL implementations of the "In-house Process Capability" interface objects is shown at Appendix 9.1

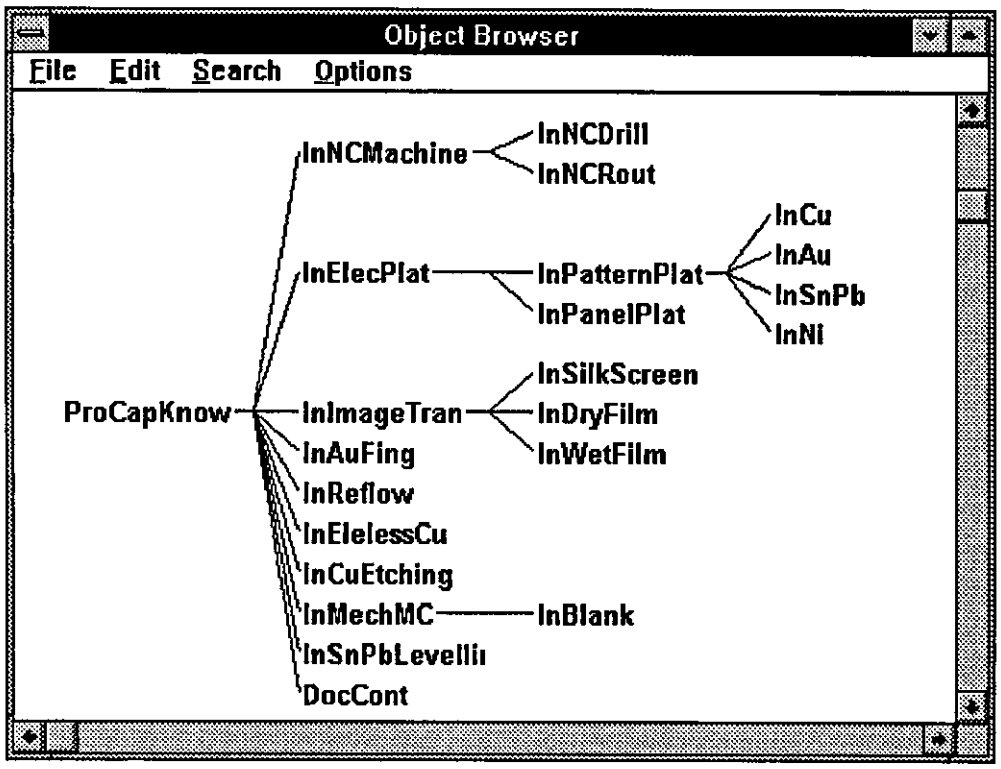


Figure 5.5 Hierarchy of Subclasses under the Class "*ProCapKnow*"

```
/*************************************
 **** CLASS: InNCDrill
 *************************************/
MakeClass( InNCDrill, InNCMachine );
MakeSlot( InNCDrill:MaxRPM );
SetSlotOption( InNCDrill:MaxRPM, VALUE_TYPE, NUMBER );
InNCDrill:MaxRPM = 70000;
SetSlotOption( InNCDrill:MaxRPM, PROMPT, "Enter/Modify NC Drilling Machine Max. RPM
Available." );
InNCDrill:NCWidth = 300;
SetSlotOption( InNCDrill:NCWidth, PROMPT, "Enter/Modify NC Drilling Machine Max. Width." );
InNCDrill:NCLength = 300;
SetSlotOption( InNCDrill:NCLength, PROMPT, "Enter/Modify NC Drilling Machine Max. Length." );
InNCDrill:MachinedAcc = 0.1;
SetSlotOption( InNCDrill:MachinedAcc, PROMPT, "Enter/Modify NC Drilling Machine Best Accuracy."
);
```

Figure 5.6 KAL Implementation of the "*InNCDrill*" Object

The operation "Update" in the interface object "*ProCapKnow*" which is used to up-date the in-house process capability knowledge objects is shown in Figure 5.7. It is implemented in Kappa KAL syntax. The "EnumSubClasses" command will loop over all children objects of the "*ProCapKnow*" object and the "GetSlotList" command will put all the slot names of the objects into temporary slot list under the "*Global*" object. "EnumList" command will ask the user to update all the slot values of all the slot names in the temporary slot lists recursively.

```
    /*************************************
     **** CLASS: ProCapKnow
     *************************************/
MakeClass( ProCapKnow, Root );
/************** METHOD: Update  **************/
MakeMethod( ProCapKnow, Update, [],
{ EnumSubClasses(ProCapKnow, dummyname,
  {
     GetSlotList(dummyname, Global:TempList);
     EnumList(Global:TempList, SlotName, AskValue(dummyname:SlotName));
  } );
 } );
```

Figure 5.7 Operation "Update" in the "*Object ProCapKnow*" Object

The user interface action of the interface object is implemented using the "*Button*" instance object provided by the Kappa software shell. When the "Up-date In-house Process Capability Knowledge" interface button ("*Button22*") (Figure 5.8 shows the "KAL" implementation of it.) is activated, function "UpProCap" will be executed. Message will be sent to the "*ProCapKnow*" object and the "Update" operation will be executed. Users will be allowed to up date the slot values of all the process knowledge objects using the graphic interface dialog box provided by the Kappa software.

```
/*************************************
 **** INSTANCE: Button22
 *************************************/
MakeInstance( Button22, Button );
Button22:SessionNumber = 0;
Button22:Title = "Up-date In-house Process Capability Knowledge";
SetValue( Button22:ForegroundColor, 0, 0, 0 );
SetValue( Button22:BackgroundColor, 255, 255, 0 );
SetValue( Button22:ForegroundColor2, 0, 0, 0 );
SetValue( Button22:BackgroundColor2, 255, 255, 255 );
Button22:Visible = TRUE;
Button22:X = 19;
Button22:Y = 10;
Button22:Width = 165;
Button22:Height = 53;
Button22:Action = UpProCap;
Button22:TabStop = 1;
ResetImage ( Button22 );
```

Figure 5.8 The "Up-date In-house Process Capability Knowledge" Button Instance Object ("Button22")

5.4 Modelling and Implementation of the Customer General Specification Interface Module

The object model of the customer general specification interface module is shown in Figure 5.9. Customer general specification documents, the interface module, graphical interface objects, customer general specification data objects and engineer objects are the five object classes of this model. In the interface object class, four operations are created to provide four functions: define data for new customer, up-date existing customer data base, store the defined or modified data and load the data to the process planning system.

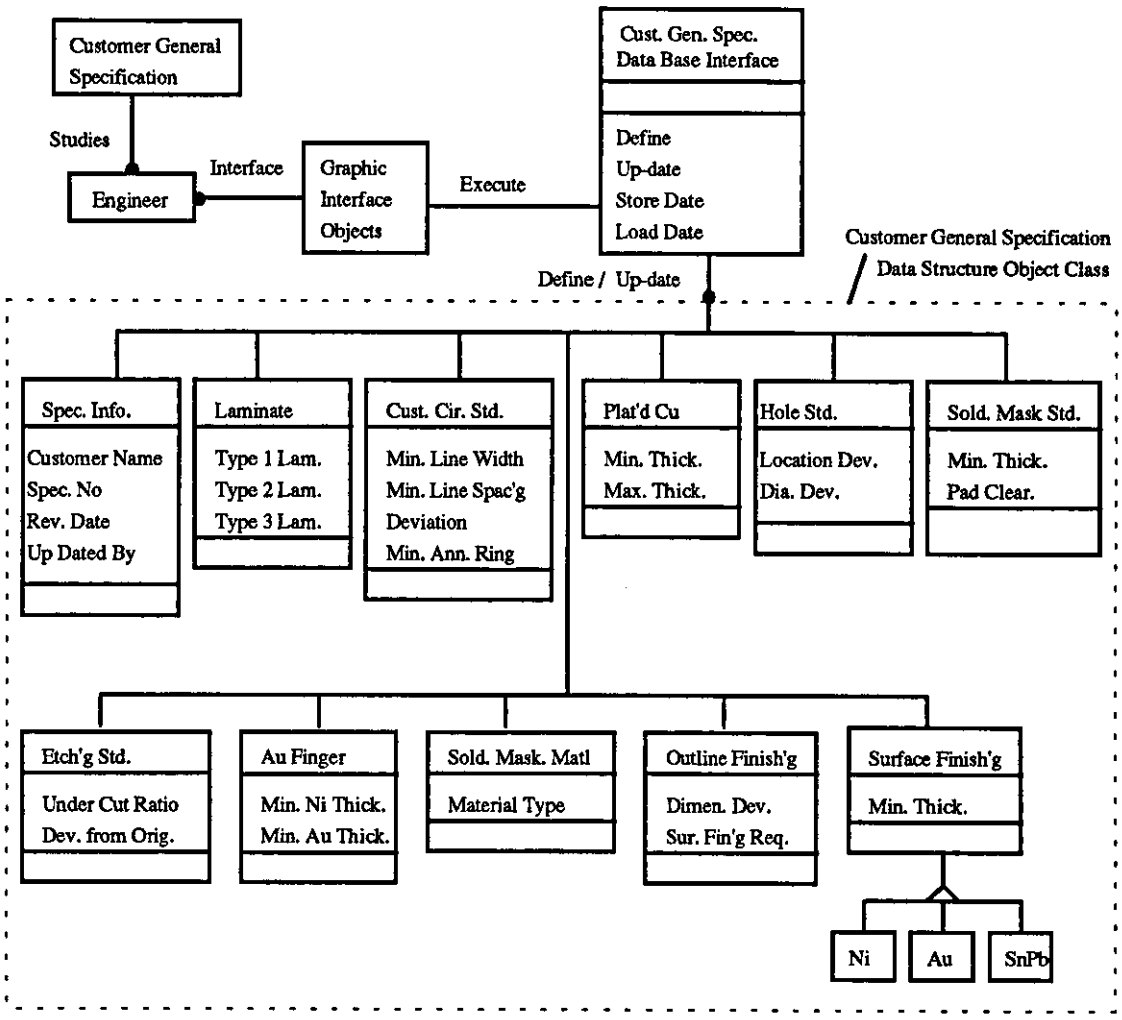


Figure 5.9 Object Model of the Customer General Specification Interface Module

5.4.1 Object Classes of the Customer General Specification Data Base

All the object classes of the customer general specification data base are discussed in the following sections.

Customer Specification Information Object Class (Abbreviated as "*SpecInfo*")

This object class is created to store the records of customer's specification. The customer name ("*CustName*"), the specification number ("*SpecNo*"), the revision date ("*RevDate*")

and the name of person who updated the specifications ("*UpDatedBy*") are the attributes of this object class.

Laminate Object Class (Abbreviated as "*CustLaminate*")

Types of laminate ("*TypeOfLaminate*") are the attributes of this object class. This object class is for the user to store the type of laminate approved by the customer and three types of laminate can be stored. More slots can be added to store more laminate types as required.

Customer Circuit Standard Object Class (Abbreviated as "*CustCirStd*")

All the circuit specifications will be stored in this object class. The attributes of this object class include the allowed minimum line width ("*MinLnWid*"), the allowed minimum line space between circuits ("*MinLnSpc*") and the allowable minimum annular ring of the circuit pad ("*MinAnnRing*").

Outline Finishing Object Class (Abbreviated as "*CustOutFin*")

The specifications of the physical outline dimension and the surface finish of the board will be stored in this object class. The allowable physical deviation of the outline dimension ("*DevDimen*") and the board outline surface finish requirement ("*SurFinReq*") are the two attributes of this object class.

Hole Specification Object Class (Abbreviated as "*CustHoleStd*")

The allowable deviation of the hole diameter ("*DevDia*") from the specified hole diameter is the only attribute of this object class.

Surface Finishing Object Class (Abbreviated as "*CustAppSurFin*")

The allowed minimum plated metal surface thickness ("*Max*") on the board surface is the attribute of this object class. The three sub-classes "*Ni*", "*Au*" and "*SnPb*" will inherit this attribute.

Etching Standard Object Class (Abbreviated as "*CustEtchStd*")

Specifications related to etching will be stored in this object class. It includes the allowed under-cut ratio ("*UCutRatio*"), the deviation of the width of the etched circuit from the original artwork ("*DevOrig*"), the minimum allowable line spacing of circuit ("*MinLnSpac*") and the minimum line width of etched circuit ("*MinLnWid*").

Gold Finger Standard Object Class (Abbreviated as "*CustAuFing*")

The minimum nickel thickness requirement ("*MinNiThick*") under the gold finger and the minimum plated gold thickness requirement ("*MinAuThick*") are the two attributes of this object class.

Solder Masking Material Standard Object Class (Abbreviated as "*CustSMMatl*")

All the approved solder masking materials ("*SMMaterial*") allowed on the circuit board will be stored under this attribute.

Copper Plating Standard Object Class (Abbreviated as "*CustPlatCu*")

The minimum allowed plated copper thickness ("*MinThick*") and the maximum plated copper thickness ("*MaxThick*") are the two attributes of this object class.

Solder Masking Standard Object Class (Abbreviated as "*CustSMStd*")

The specifications of the solder masking are the attributes of this object class. They are the minimum allowed solder masking thickness ("*MinThick*") and the minimum allowed solder pad clearance ("*PadClear*").

5.4.2 Implementation of the Customer Specification Interface Module

Based on the object model, the dynamic model and the functional model, the interface module and the specification data objects have been implemented using the Kappa "KAL" language. The hierarchy of the object classes is shown in Figure 5.10.

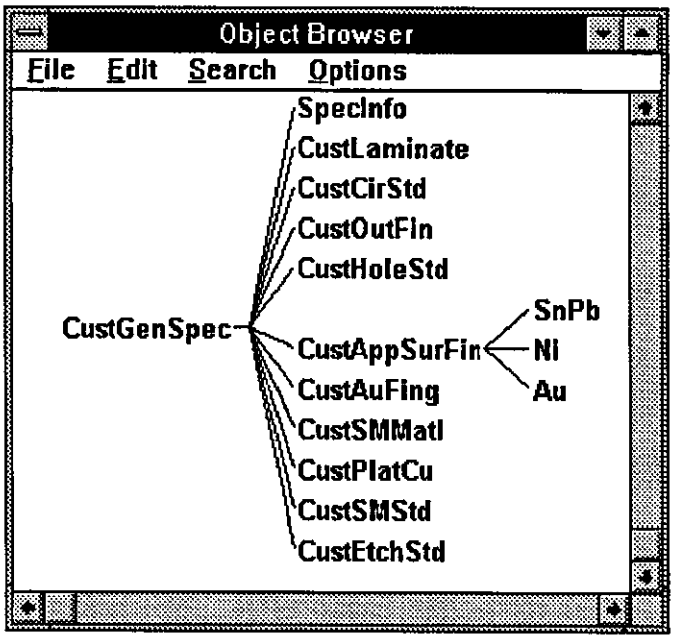


Figure 5.10 Hierarchy of the Object Class "*CustGenSpec*" and Different Sub-classes of the Customer General Specification Data Base

Four operations: "Define", "Update", "StoreData" and "LoadData" are implemented in the interface object. Operation "Define" is used to create a new specification object for a new customer, operation "Update" to up-date the existing customer specification objects, operation "StoreData" to store data base objects in a file after the operation "Update" or "Define" and operation "LoadData" to load the specification data objects in the planning system before doing process planning. The KAL implementation of the operations is shown in Figure 5.11.

When the "Create/Update Customer General Specification" instance object button is activated, users are given two options: "inputting new customer specification" or "up-dating existing specification". When the required option is selected, a message "Define" or "Update" is sent to the "CustGenSpec" object to activate the appropriate define or update operation. Upon finished the "Define" or the "Update" operation, a message is sent to activate the "Store" operation so that customer specification data can be stored in a specific data file by the user. The KAL implementation of the customer specification interface module and the two customers' data base objects are shown at Appendix 9.2, 2.1 and 2.2 respectively.

```
    /************************************
     **** CLASS: CustGenSpec
     ************************************/
MakeClass( CustGenSpec, Root );

/************** METHOD: Define  **************/
MakeMethod( CustGenSpec, Define, [],
  {
  InterpretFile( "c:\kapfile\blancust.kap" );
  EnumSubClasses( CustGenSpec, dummyname,
     {
     GetSlotList( dummyname, Global:TempList );
     EnumList( Global:TempList, SlotName, ResetValue( dummyname:SlotName ) );      } );
  PostMessage( "Initialization of Customer General Specification data base is completed.  Input data!!" );
  EnumSubClasses( CustGenSpec, dummyname,
     {
     GetSlotList( dummyname, Global:TempList );
     EnumList( Global:TempList, SlotName, AskValue( dummyname:SlotName ) );      } );
  } );

/************** METHOD: Update  **************/
MakeMethod( CustGenSpec, Update, [],
  {
  EnumSubClasses( CustGenSpec, dummyname,
     {
     GetSlotList( dummyname, Global:TempList );
     EnumList( Global:TempList, SlotName, AskValue( dummyname:SlotName ) );      } );
  } );

/************** METHOD: StoreData  **************/
MakeMethod( CustGenSpec, StoreData, [],
  {
  PostInputForm( "Drive and File Name to Store Customer General Specification Data",
     Global, DriveDir, "Drive (e.g. b:)", Global, FileNameListCusGenSpec,
     "Gen. Spec. File Name (e.g. custg)" );
  BuildUpKnowFileName( Global:FileNameListCusGenSpec );
  StoreKnowledge( CustGenSpec );
  EnumSubClasses( CustAppSurFin, dumname, DeleteClass( dumname ) );
  EnumSubClasses( CustGenSpec, dumname, DeleteClass( dumname ) );
  } );

/************** METHOD: LoadData  **************/
MakeMethod( CustGenSpec, LoadData, [],
  {
  PostInputForm( "Drive and File Name of Customer General Specification",
     Global, DriveDir, "Drive (e.g. b:)", Global, FileNameListCusGenSpec,
     "Gen. Spec. File Name (e.g. custg)" );
  BuildUpKnowFileName( Global:FileNameListCusGenSpec );
  InterpretFile( Global:TempFileName );
  } );
```

Figure 5.11 The Four Operations in the "*CustGenSpec*" Object

5.5 Modelling and Implementation of the Product Requirements Interface Module

The object model of the product requirements interface module is developed and presented in Figure 5.12. There are five object classes in this model and six operations in the interface object.

5.5.1 Object Classes of the Product Requirements Data Base

The object classes of the product requirements data base are described below:

Customer Product Requirements Document Information Object Class ("*ReqDocInf*")
The customer product requirements document information object class is used to store the detailed information of the customer's product document. This object class consists of five attributes: the customer name ("*CustName*"), the product name ("*ProdName*"), the product in-house part number ("*PNInHouse*"), the customer part number ("*PNCust*") and the customer purchase order number ("*PurOrdNo*").

Product Documents Object Class (Abbreviated as "*ProDocument*")
The product documents object class is used to store all the reference numbers and names of the product documents. Attributes in this object class include the product film number ("*FilmNo*"), the product drawing number ("*DrawNo*"), the specification document number ("*SpecNo*") and the drill file number ("*DrillFileNo*").

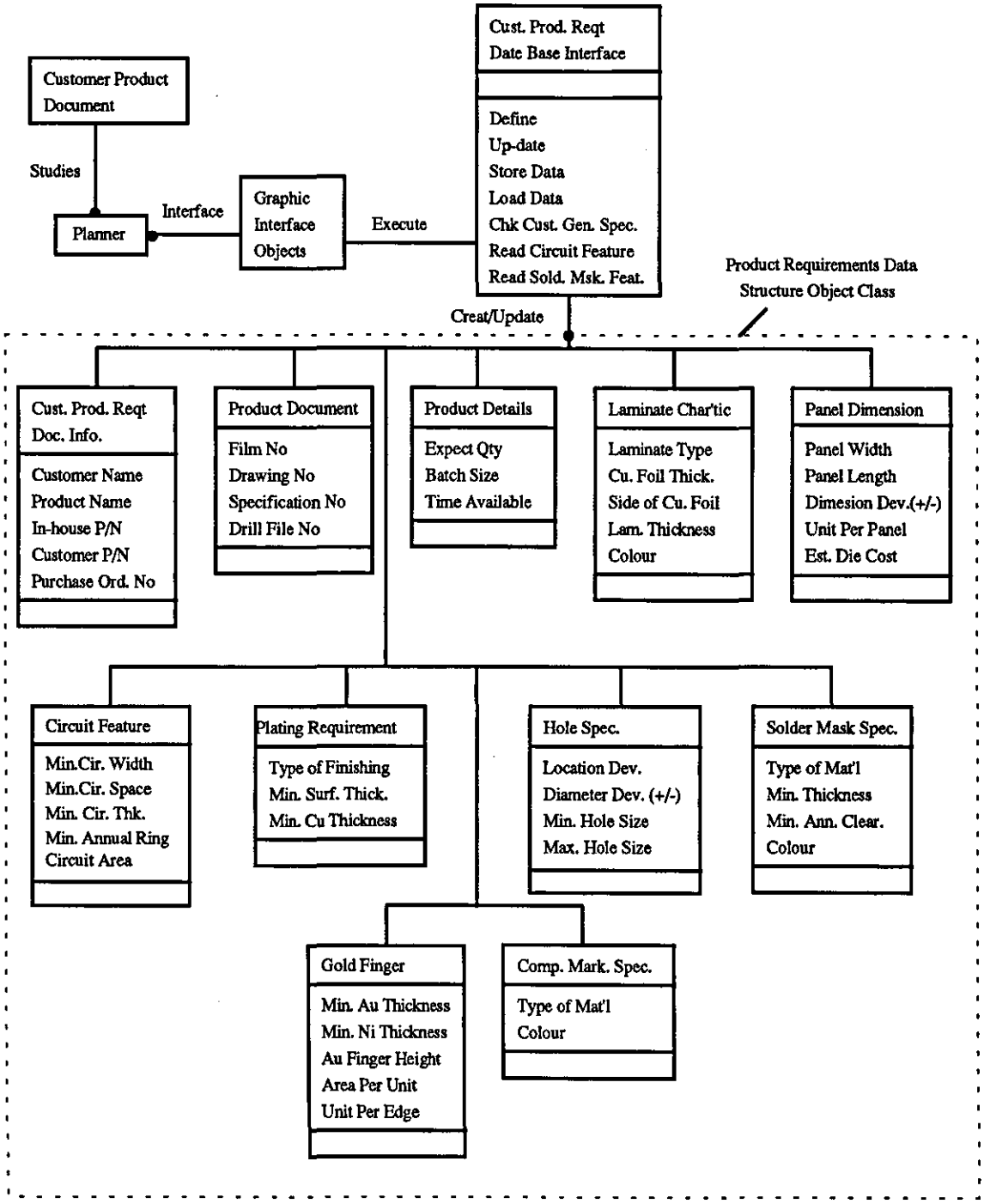


Figure 5.12 Object Model of the Customer Product Requirements Interface Module

Production Details Object Class (Abbreviated as "*ProProdDetails*")

The production details object class stores up the production characteristics of the product. It includes the total expected order quantity of the product ("*ExpQty*"), the batch size in this order ("*BatchSize*") and the time (in working day) available to manufacture this batch of product ("*WkDateAvail*").

Laminate Characteristic Object Class (Abbreviated as "*ProLamChar*")

The laminate characteristic object class records all the characteristics of the laminate requirements of this product. It includes the type of laminate to be used ("*LaminateType*"), the thickness of the copper foil ("*CuFoilThickness*"), the number of copper foil layer ("*CuFoilSide*"), the laminate thickness ("*LamThickness*") and the colour of the laminate ("*Colour*").

Panel Dimension Object Class (Abbreviated as "*ProPanelDimen*")

The attributes of this object class include: the width ("*Width*") and the length ("*Length*") of the panel, the allowable physical dimension deviation of the board ("*DimenDev*"), the number of unit(s) in each panel ("*UnitPerPanel*") and the estimated blanking die cost ("*EstDieCost*") if blanking is used to blank out the board.

Plating Requirements Object Class (Abbreviated as "*ProPlatingReq*")

This object class is used to store the values of surface finish ("*SurfaceFinType*") required for the board, the minimum plated metal thickness allowed ("*SurfaceFinThick*") and the minimum plated copper thickness required ("*CuThickMin*").

Circuit Feature Object Class (Abbreviated as "*ProFeatCircuit*")

This object class has four attributes: the minimum allowable circuit width ("*CirWidth*"), the minimum allowable circuit spacing ("*SpaceWidth*"), the minimum allowable circuit thickness ("*CirThickness*") and the minimum allowable circuit annular ring ("*AnnularRing*").

Hole Size Specifications Object Class (Abbreviated as "*ProHoleSpec*")

This object class stores up the specifications such as the allowable deviation in hole location position against the true position ("*LocDev*"), the deviation of the hole diameter ("*DiaDev*"), the minimum hole size ("*MinHoleSize*") and the maximum hole size ("*MaxHoleSize*").

Solder Mask Specifications Object Class (Abbreviated as "*ProSoldMask*")

The attributes of this object class include: the type of solder mask material used for this product ("*MaterialType*"), the minimum allowable masking thickness ("*MinThick*"), the minimum allowable solder mask opening ("*MinClear*") and the allowable solder mask material colour ("*Colour*").

Gold Finger Specifications Object Class (Abbreviated as "*ProAuFinger*")

This object class is used to store all the specifications of the board gold finger. It includes the height of the gold finger ("*Height*"), the required minimum gold thickness ("*AuThick*"), the minimum plated nickel thickness under the gold ("*NiThick*"), the number of unit per plating edge ("*UnitPerEdge*") and the plating area of gold finger per unit circuit ("*PlatAreaPerUnit*").

Component Marking Specifications Object Class (Abbreviated as "*ProCompMark*")

The component marking specifications object class is used to store the component marking material ("*CompMaterial*") and the colour of the marking material ("*CompColour*") to be used.

5.5.2 Implementation of the Customer Product Requirements Interface Module

The interface module "*CustProdReqt*" and the product requirements data base object classes (Figure 5.13) are implemented using the Kappa "KAL" language. All the product data objects can be created or modified by the "*CustProdReqt*" interface object. In addition to the four operations "Define", "UpDate", "StoreData" and "LoadData" described in Section 5.4.2, three more operations: "CheckCustGenSpecMethod", "ReadCircuitFeature" and "ReadSolderMaskFeature" are created in the "*CustProdReqt*" interface object.

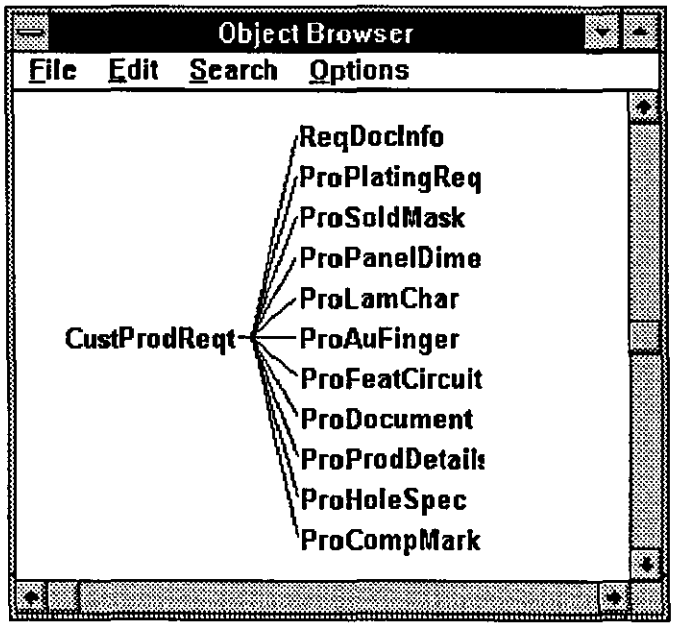


Figure 5.13 Hierarchy of the Subclasses under the Class "*CustProdReqt*"

The "CheckCustGenSpecMethod" operations are used to manipulate the rules to check the specification and will be discussed in detail in the next chapter. The "ReadCircuitFeature" (Figure 5.14) and the "ReadSolderMaskFeature" operations (Figure 5.15) are used to read the circuit and solder masking features directly from the data file. The "SetValue" command in the two operations will be used to set the respective circuit or solder masking object attribute values.

```
/************** METHOD: ReadCircuitFeature  **************/
MakeMethod( CustProdReqt, ReadCircuitFeature, [],
 {
   SetValue( Global, DriveDir, "b:" );
   SetValue( Global, dumstring, cirfea.fil );
   PostInputForm( "Input Drive and File Name of the Circuit Feature File",
      Global, DriveDir, "Drive Name (e.g. b:)", Global, dumstring,
      "File Name (e.g. cirfea.fil)" );
   OpenReadFile( Global:DriveDir # Global:dumstring );
   SetValue( Global:CircuitArea, ReadLine( ) );
   SetValue( Global:CirSpec1, ReadLine( ) );
   SetValue( Global:CirSpec1No, ReadLine( ) );
   SetValue( Global:CirSpec2, ReadLine( ) );
   SetValue( Global:CirSpec2No, ReadLine( ) );
   SetValue( Global:CirSpec3, ReadLine( ) );
   SetValue( Global:CirSpec3No, ReadLine( ) );
   ReadLine( );
   CloseReadFile( );
   SetValue( ProFeatCircuit:CircuitArea, Global:CircuitArea );
   If ( Global:CirSpec1No == 0 )
    Then SetValue( ProFeatCircuit:SpaceWidth, Global:CirSpec1 );
   If ( Global:CirSpec2No == 0 )
    Then SetValue( ProFeatCircuit:SpaceWidth, Global:CirSpec2 );
   If ( Global:CirSpec3No == 0 )
    Then SetValue( ProFeatCircuit:SpaceWidth, Global:CirSpec3 );
} );
```

Figure 5.14 The "ReadCircuitFeature" Operation

```
/************** METHOD: ReadSolderMaskFeature **************/
MakeMethod( CustProdReqt, ReadSolderMaskFeature, [],
{
  SetValue( Global, DriveDir, "b:" );
  SetValue( Global, dumstring, solmas.fil );
  PostInputForm( "Input Drive and File Name of the Solder Mask Feature File",
    Global, DriveDir, "Drive Name (e.g. b:)", Global, dumstring, "File Name (e.g. sold.fil)" );
  OpenReadFile( Global:DriveDir # Global:dumstring );
  SetValue( Global:SoldSpec1, ReadLine( ) );
  SetValue( Global:SoldSpec1No, ReadLine( ) );
  SetValue( Global:SoldSpec2, ReadLine( ) );
  SetValue( Global:SoldSpec2No, ReadLine( ) );
  SetValue( Global:SoldSpec3, ReadLine( ) );
  SetValue( Global:SoldSpec3No, ReadLine( ) );
  CloseReadFile( );
  If ( Global:SoldSpec1No == 0 )
   Then SetValue( ProSoldMask:MinClear, Global:SoldSpec1 );
  If ( Global:SoldSpec2No == 0 )
   Then SetValue( ProSoldMask:MinClear, Global:SoldSpec2 );
  If ( Global:SoldSpec3No == 0 )
   Then SetValue( ProSoldMask:MinClear, Global:SoldSpec3 );
} );
```

Figure 5.15 The "ReadSolderMaskFeature" Operation

During the user interactive input, the "Input Board Characteristic" window (Figure 5.16) will be displayed to allow the selection of different product requirements objects.

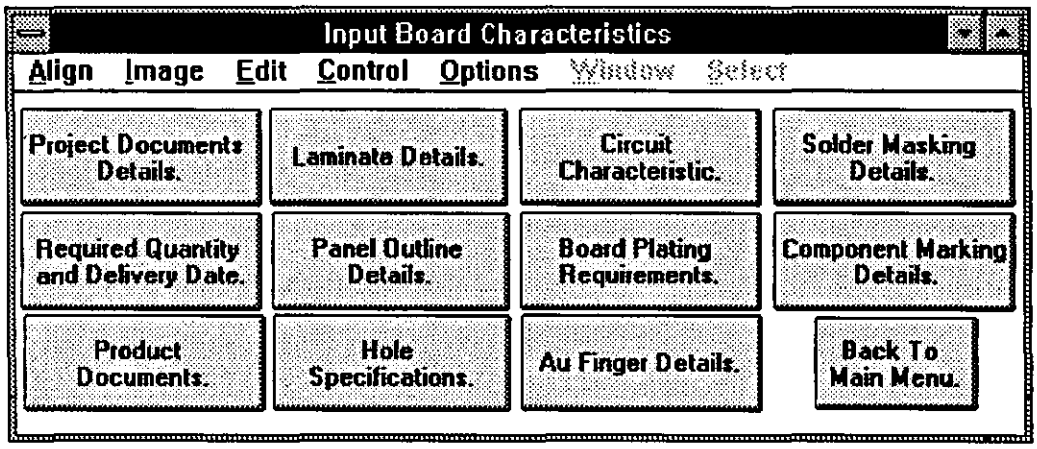


Figure 5.16 The "Input Board Characteristics" Window

For example, if the user selects the "Au Finger Details" option, the "Input/Update Gold Finger Characteristics" window (Figure 5.17) will be displayed for input/up-date of gold finger requirements details. When the user has finished the input/up-date, the colour of the "Au Finger Details" button in the "Input Board Characteristics" window will change indicating that the gold finger details have been input/up-dated. After the user has finished the input/up-date of all the requirements objects, the user can select the "Back To Main Menu" option. The "StoreData" operation will then store the product requirements in a data file. The KAL representation of the board requirements interface module object is at Appendix 9.3.

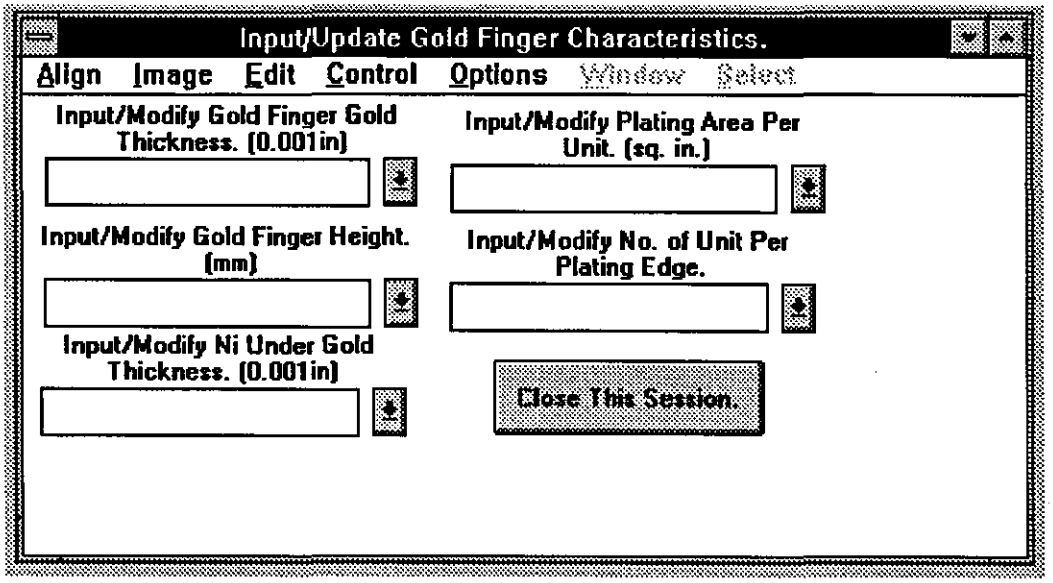


Figure 5.17 The "Input/Update Gold Finger Characteristics" Window

5.6 Modelling and Implementation of the Features Extraction Modules

Features are defined in this project as prominent manufacturing characteristics of the circuit board which are meaningful in the computer aided process planning system. If the electronic CAD databases can provide the required planning features information, a ready link between the design and the process planning will be established. However, as the circuit and solder masking artwork data files do not hold the required planning features information, an interface program is needed to be developed and is termed as the feature extraction module in this project.

The features considered in this project are geometric in nature and can be classified into three groups: drilling features, circuit features and solder masking features. The drilling features comprise hole diameters, hole locations and the total number of different sized holes. The circuit features include unit pattern area, minimum pad size, minimum line width, minimum circuit spacing, gold finger area and height of gold finger. The solder masking features are size of solder pad opening and solder pad annular ring clearance.

5.6.1 Modelling of the Circuit Feature Extraction Module

The object model of the circuit feature extraction module is shown in Figure 5.18. The "Gerber" format circuit files, the program, circuit feature data files and program operators are the four object classes of this module. The attributes of the circuit feature data files object class includes circuit area, specifications for checking and the number of position under the specification.

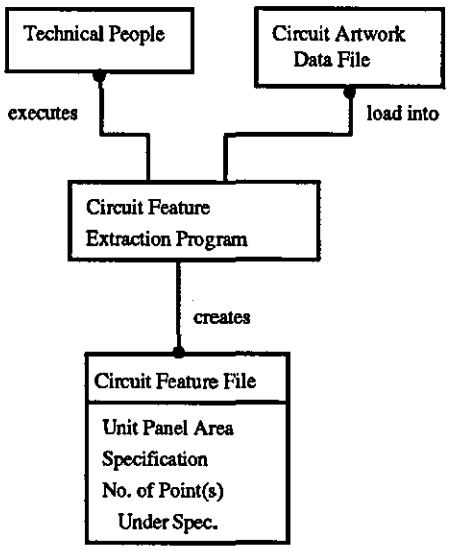


Figure 5.18 Object Modelling of the Circuit Feature Extraction Module

The functional model (in data flow diagram form) of the circuit feature extraction program is shown in Figure 5.19. The circuit image file and the aperture file are the two actor objects which provide data for the subsequent extraction process. The circuit manipulation process, the spacing calculation process and the check specification process are all implemented using C++ programming language. The pad and trace data are processed and stored as an intermediate file for use by the solder masking feature extraction program. Appendix 4.1 is the program listing of the circuit feature extraction program. The results of the circuit features of circuitA are included at Appendix 5.1.

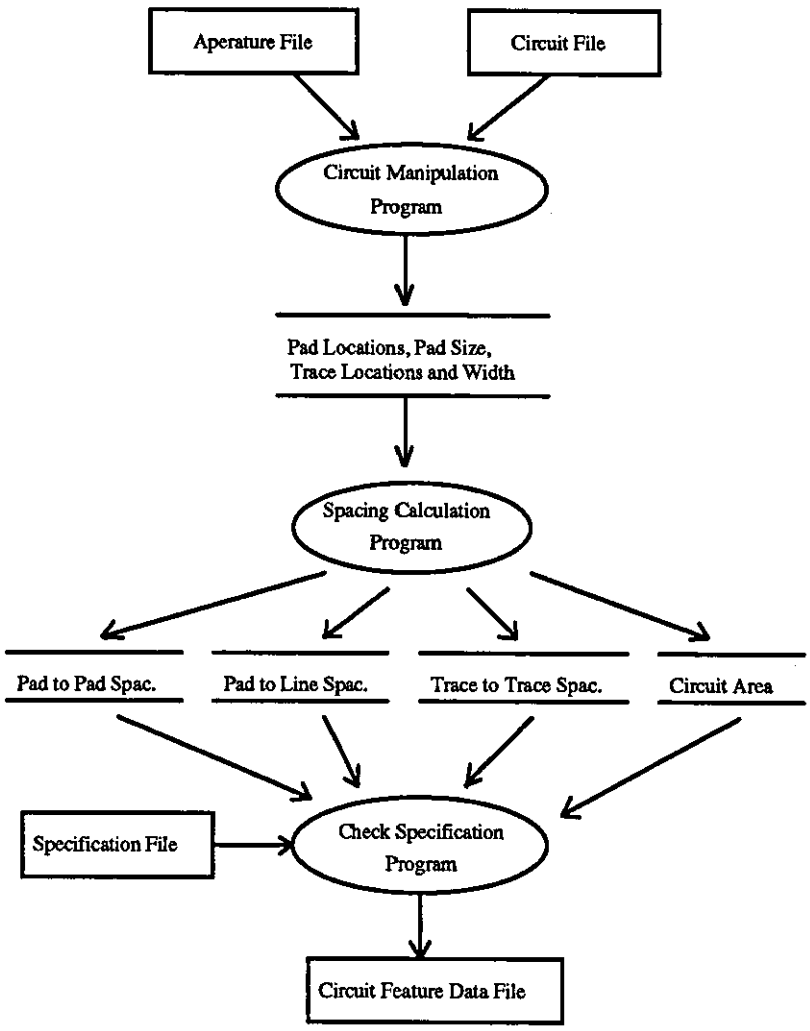


Figure 5.19 Functional Model (in Data Flow Diagram) of the Circuit Feature Extraction Module

The circuit feature data file contains the total unit pattern circuit plating area and the number of point(s) by which the spacing(s) is(are) smaller than the distance as specified in the spacing specification file. This data file can be accessed by the "ReadCircuitFeature" operation as discussed in Section 5.5.2. For the time being, the gold finger features cannot be automatically extracted by this program and planners are required to extract or calculate these features by themselves and input the results into the system interactively using the product requirements interface module.

5.6.2 Modelling of the Solder Masking Feature Extraction Module

The object model of the solder masking feature extraction module is shown in Figure 5.20. The "Gerber" format solder masking files, pad location data file, the solder masking extraction program, the solder masking data files and the program operators are the five object classes of this module. The attributes of the solder masking feature object class include specification standards for the checking and the number of position under the specification.

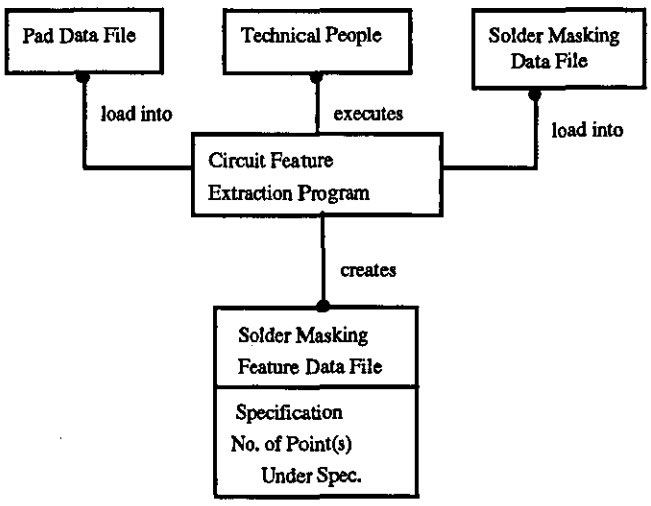


Figure 5.20 The Object Model of the Solder Masking Feature Extraction Module

The functional model (using data flow diagram representation method) of the solder masking feature extraction module is shown in Figure 5.21. The pad location file and the solder masking image file are the actor objects which provide the required data for the subsequent processing. Size and location of the solder mask pads will be extracted from the solder masking image file and the aperture data file. When the location of circuit pads and solder masking pads have been matched, their sizes are used to calculate the actual

solder pad clearances. The number of the pad clearances which is smaller than the clearance specification will be counted and output to the feature data file. A program in C++ is implemented using this logic. The programming results are presented at Appendix 4.2; the results of the solder masking features of circuitA are included at Appendix 5.2

.

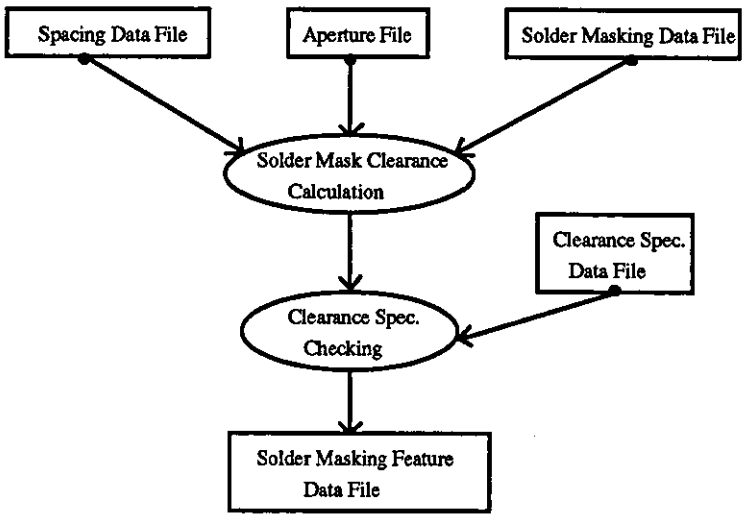


Figure 5.21 Functional Modelling of the Solder Masking Feature Extraction Module

5.7 Summary

This chapter has discussed the modelling and implementation of the two supporting modules: interface module and feature extraction module. The three high level models, namely, the object model, the dynamic model and the functional model of the interface module, were first discussed. Then the detail object models of the three sub-level interface modules:- the in-house process capability interface module, the customer general specification interface module and the customer product requirements interface module were presented. The respective object structures were defined and explained in detail.

The implementation of the models using the Kappa software shell were illustrated and supported by examples. The modelling and implementation of the circuit and the solder masking feature extraction modules were addressed at the end of the chapter.

CHAPTER 6
PLANNING KNOWLEDGE USED IN THE KB PCB CAPP SYSTEM

6.1 Introduction

This chapter begins presenting the organisation structure of the knowledge base in the system. The used knowledge represented in the system is categorised into five groups: product requirements checking knowledge, process selection knowledge, process sequencing knowledge, machine selection knowledge and process recipe generation knowledge. The representation method of each category of knowledge is discussed together with description and explanation of examples of how the knowledge is used in each category.

6.2 Knowledge Representation in the PCB CAPP System

The amount of knowledge required in the planning of PCB manufacture is very extensive and diverse. The knowledge has been extracted from board manufacturing experts, handbooks and reference books [Bosshart, 1988], [Coombs, 1988], [Coombs, 1990], [Ginsberg, 1991] standards [IPC-D-275] and instruction sheets from suppliers. The extracted knowledge was, then, classified and represented systematically. Two kinds of knowledge representation method are used in the system: declarative facts and procedural rules. Declarative facts are represented as slots in the object structure using the Kappa KAL syntax. The in-house process capability knowledge, customer general specifications and product requirement are represented as declarative facts and their structures have

already been described in Chapter 5. Other planning knowledge of the system is discussed in the following sections.

The knowledge structure in this process planning system is organized using the object-oriented concept. Because of the inheritance characteristic in the object-oriented approach, a new knowledge object added under the same parent object class is executed like the existing object class does. The encapsulation property of objects also separates one knowledge object from another and thus eliminates propagation effects, if one knowledge object has to be changed. There are five groups of knowledge objects in the system: the product requirements checking knowledge object, the process selection knowledge object, the process sequencing knowledge object, the machine selection knowledge object and the process recipe generation knowledge object. The algorithms to execute the knowledge are implemented in the operations of the objects. Declaratives facts are stored as attribute values in the knowledge objects while other knowledge is represented as rules. The object modelling of the knowledge in the planning system is presented in Figure 6.1.

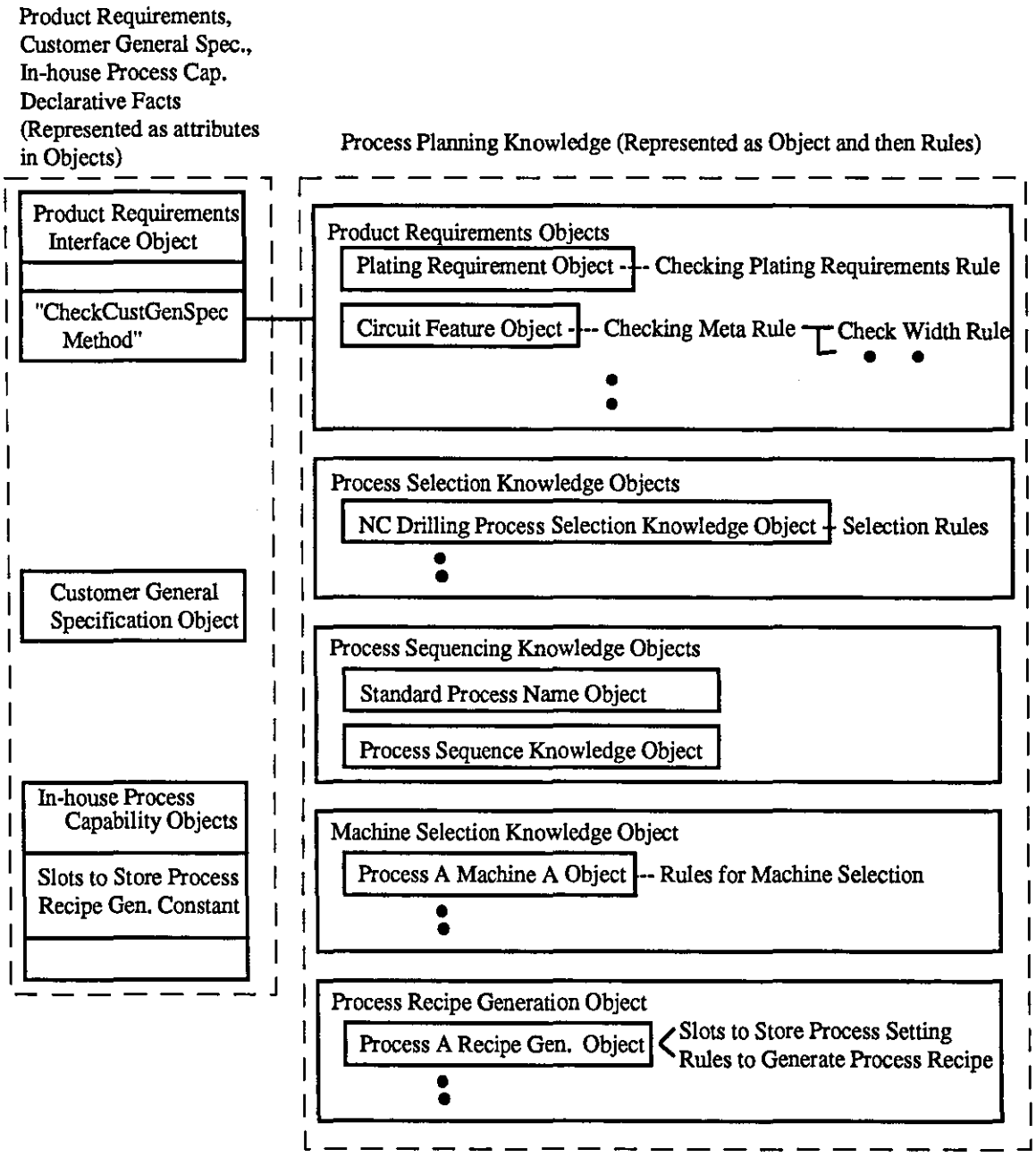


Figure 6.1 Object Modelling of the Knowledge in the PCB CAPP System

6.3 Rules to Check Product Requirements against Customer General Specifications

A set of rules have been developed and represented to check product requirements against customer general specifications. The checking will ensure that the required circuit board is within the customer general specifications. The following "CheckCustGenSpecMethod" operation in the product requirements interface object class will activate the "CheckCustGenSpec" methods in all the product requirements sub-classes. Other commands in this operation is responsible for the storage of the out of specification message(s) in text file.

```
MakeMethod( CustProdReqt, CheckCustGenSpecMethod, [],
  {
  ClearList( MessageFile:OutGenSpecMessage );
  EnumSubClasses( CustProdReqt, dummyclassname,
    SendMessage ( dummyclassname, CheckCustGenSpec ));
    SetValue( Global, DriveDir, "b:" );
  SetValue( Global, dumstring, Global:FileNameListProd # .msg );
  OpenWriteFile( Global:DriveDir # Global:dumstring );
  CloseWriteFile( );
  If ( LengthList( MessageFile:OutGenSpecMessage )
       > 0 )
   Then {
      SetValue( Global, DriveDir, "b:" );
      PostInputForm( "Out of Specification Message Found !!! Input Drive Name for the Message File",
        Global, DriveDir, "Drive Name (e.g. b:)", Global, FileNameListProd,
        "Out of Spec. Message File Name (e.g. p8001)" );
      SetValue( Global, dumstring, Global:FileNameListProd  # .msg );
      OpenWriteFile( Global:DriveDir # Global:dumstring );
      WriteLine( "/* Prod. Reqt. Out of Cust. Gen. Spec. */" );
      WriteLine( "/* Type of Reqt. --- Prod.--Spec. (Value)*/" );
      EnumList( MessageFile:OutGenSpecMessage, message, WriteLine( message ) );
      CloseWriteFile( );
      PostMessage( "Out of Specification Message Stored in File "
                 # Global:DriveDir # Global:dumstring );
      }
    Else {PostMessage( "Product Requirements Within Customer General Specification.")};
  } );
```

At present, nine checking rules have been developed and will be executed by the "CheckCustGenSpec" operations of the product requirements sub-objects. The rules are organised in a hierarchical way for effective program execution and easy subsequent enhancement (Figure 6.2).

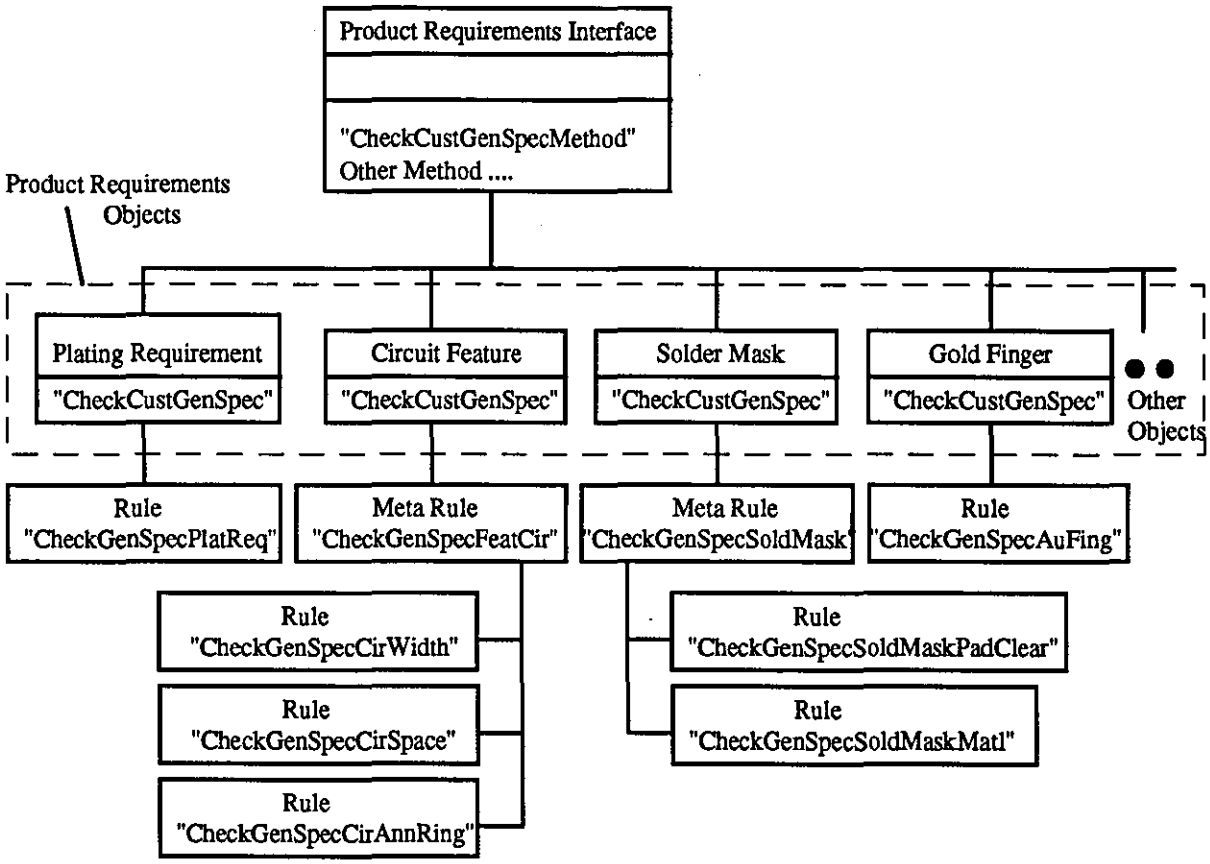


Figure 6.2 Hierarchy of the Product Requirements Objects and the Rules under Various Sub-objects

6.3.1 Rules to Check Circuit Features

The "CheckGenSpecFeatCir" meta rule activated by the "CheckCustGenSpec" operation of the "*circuit feature*" object is to control the operation of three sub-level checking rules.

```
MakeRule( CheckGenSpecFeatCir, [], TRUE,
{
ForwardChain( [ NOASSERT ], NULL, CheckGenSpecFeatCirWidth );
ForwardChain( [ NOASSERT ], NULL, CheckGenSpecFeatCirSpace );
ForwardChain( [ NOASSERT ], NULL, CheckGenSpecFeatCirAnnRing );
} );
```

The following three sub-level checking rules are used to check whether the circuit requirements are within the customer general specification.

Rule to Check Circuit Width Specification

```
MakeRule( CheckGenSpecFeatCirWidth, [],
  ProFeatCircuit:CirWidth < CustCirStd:MinLnWid,
  {
  PostMessage( "Circuit width " # ProFeatCircuit:CirWidth # " is SMALLER than customer
general accepted circuit width "  # CustCirStd:MinLnWid );
  AppendToList( MessageFile:OutGenSpecMessage, CircuitWidth  # " -- "
            # ProFeatCircuit:CirWidth  # < # CustCirStd:MinLnWid );
  } );
```

Rule to Check Circuit Spacing Specification

```
MakeRule( CheckGenSpecFeatCirSpace, [],
  ProFeatCircuit:SpaceWidth < CustCirStd:MinLnSpac,
  {
  PostMessage( "Board circuit spacing " # ProFeatCircuit:SpaceWidth # " is SMALLER than
            customer general accepted circuit spacing " # CustCirStd:MinLnSpac );
  AppendToList( MessageFile:OutGenSpecMessage, CircuitSpacing # " -- "
            # ProFeatCircuit:SpaceWidth # < # CustCirStd:MinLnSpac );
  } );
```

Rule to Check Annular Ring Specification

```
  MakeRule( CheckGenSpecFeatCirAnnRing, [],
  ProFeatCircuit:AnnularRing < CustCirStd:MinAnnRing,
  {
  PostMessage( "Board circuit annular ring " # ProFeatCircuit:AnnularRing
            # " is SMALLER than customer general accepted annular ring "
            # CustCirStd:MinAnnRing );
  AppendToList( MessageFile:OutGenSpecMessage, CirAnnularRing # " -- "
            # ProFeatCircuit:AnnularRing # < # CustCirStd:MinAnnRing );
  } );
```

Each rule will check one type of the circuit board's physical characteristics against the customer general specification. If a violation is found, a pop-up window (the "Post Message" command) with a specific warning message will be posted to the user. In addition, a warning message will be appended (the "AppendToList" command) into the "out of specification message" slot of the "message file" object. Users can look at the message file later or print it out for reference.

6.3.2 Rule to Check Copper Plating Requirement

This rule is activated by the "CheckCustGenSpec" operation of the "*plating requirement*" object. The format of this rule is similar to that of the sub-level checking rules discussed in the previous section and is as follows:

```
MakeRule( CheckGenSpecPlatReq, [],
  ProPlatingReq:CuThickMin < CustPlatCu:MinThick,
  {
  PostMessage( "Board required Cu thickness " # ProPlatingReq:CuThickMin
             # " is LESS than customer general required min. thickness "
             # CustPlatCu:MinThick );
  AppendToList( MessageFile:OutGenSpecMessage, PlatedMinCuThk  # " -- "
             # ProPlatingReq:CuThickMin  # < # CustPlatCu:MinThick );
  } );
```

6.3.3 Rule to Check Solder Masking Requirements

The "CheckGenSpecSoldMask" meta rule will be activated by the "CheckCustGenSpec" operation of the "solder mask" object. Its main function is to control the execution of the two sub-level solder masking checking rules.

```
MakeRule( CheckGenSpecSoldMask, [],
Not( Null?( ProSoldMask:MaterialType ) );
{
ForwardChain( [ NOASSERT ], NULL, CheckGenSpecSoldMaskPadClear );
ForwardChain( [ NOASSERT ], NULL, CheckGenSpecSoldMaskMatl );
});
```

The following sub-level rule is used to check whether the minimum solder masking clearance of the board is within the customer general specification.

```
MakeRule( CheckGenSpecSoldMaskPadClear, [],
ProSoldMask:MinClear < CustSMStd:PadClear,
{
PostMessage( "Board required solder mask pad clearance " # ProSoldMask:MinClear
            # " is LESS than customer general required min. clearance "
            # CustSMStd:PadClear );
AppendToList( MessageFile:OutGenSpecMessage, SolderPadClear # " -- " #
            ProSoldMask:MinClear  # < # CustSMStd:PadClear );
} );
```

The following sub-level rule is used to check whether the product required solder mask material is accepted by the customer general specification.

```
MakeRule( CheckGenSpecSoldMaskMatl, [],
  Not( CustSMMatl:SMMaterial #= ProSoldMask:MaterialType ),
  {
  PostMessage( "Board required solder mask material " # ProSoldMask:MaterialType
            # " is NOT customer general required " # CustSMMatl:SMMaterial );
  AppendToList( MessageFile:OutGenSpecMessage, SolderMaskMatl # " -- " #
              ProSoldMask:MaterialType  # "Not accepted" );
  } );
```

6.3.4 Rule to Check Gold Finger Requirement

The following rule is used to check the board gold finger requirement against the customer's gold finger general specification.

```
MakeRule( CheckGenSpecAuFing, [],
  ProAuFinger:AuThick < CustAuFing:MinAuThick,
  {
  PostMessage( "Board required gold finger thickness " # ProAuFinger:AuThick
            # " is LESS than customer general required min. thickness "
            # CustAuFing:MinAuThick );
  AppendToList( MessageFile:OutGenSpecMessage, AuFingerAuThk  # " -- "
            # ProAuFinger:AuThick  # < # CustAuFing:MinAuThick );
  } );
```

6.3.5 Adding New Specification Checking Rules

Similar rules to check the board requirements against the customer general specifications can be created using the Kappa software rule edit tool. The name of the created rule should be included in the "CheckCustGenSpec" operation of respective product requirements object. However, the programming logic need not to be modified as the "EnumSubClasses (CustProdReqt, dummyclassname, SendMessage (dummyclassname, CheckCustGenSpec))" command in the "CheckCustGenSpec" methods will automatically activate all operations under the "*CustProdReqt*" object classes. This is one of the advantages of using knowledge-based object oriented Kappa software shell which supports both object-oriented programming approach and rule base reasoning approach.

6.4 Process Selection Knowledge

The process selection knowledge of each process is represented by rules. The execution of these rules is implemented as operation on each process class (Figure 6.3). The function "FindInHouProcess" will execute the process selection operations of all process instances under the object "SugProInHou". The appropriate process selection rules will be used to check whether that process should be used to manufacture that board. If that process is required and selected, the next level of rules under the process selection rules to check the in-house process capability will be executed. If the in-house process is found to be not capable of manufacturing the board, a warning message will be posted to the planner interactively. Also, a warning message will be put into the slot "OutGenSpecMessage" of the "MessageFile" object for reference or printing later. 19 rules have been developed for process selection and ten rules have been developed for checking with in-house process capability. Rules to represent the knowledge of NC drilling process selection, electroless copper plating process selection, circuit image transfer dry film process selection and bare copper process selection will be discussed in the following sections.

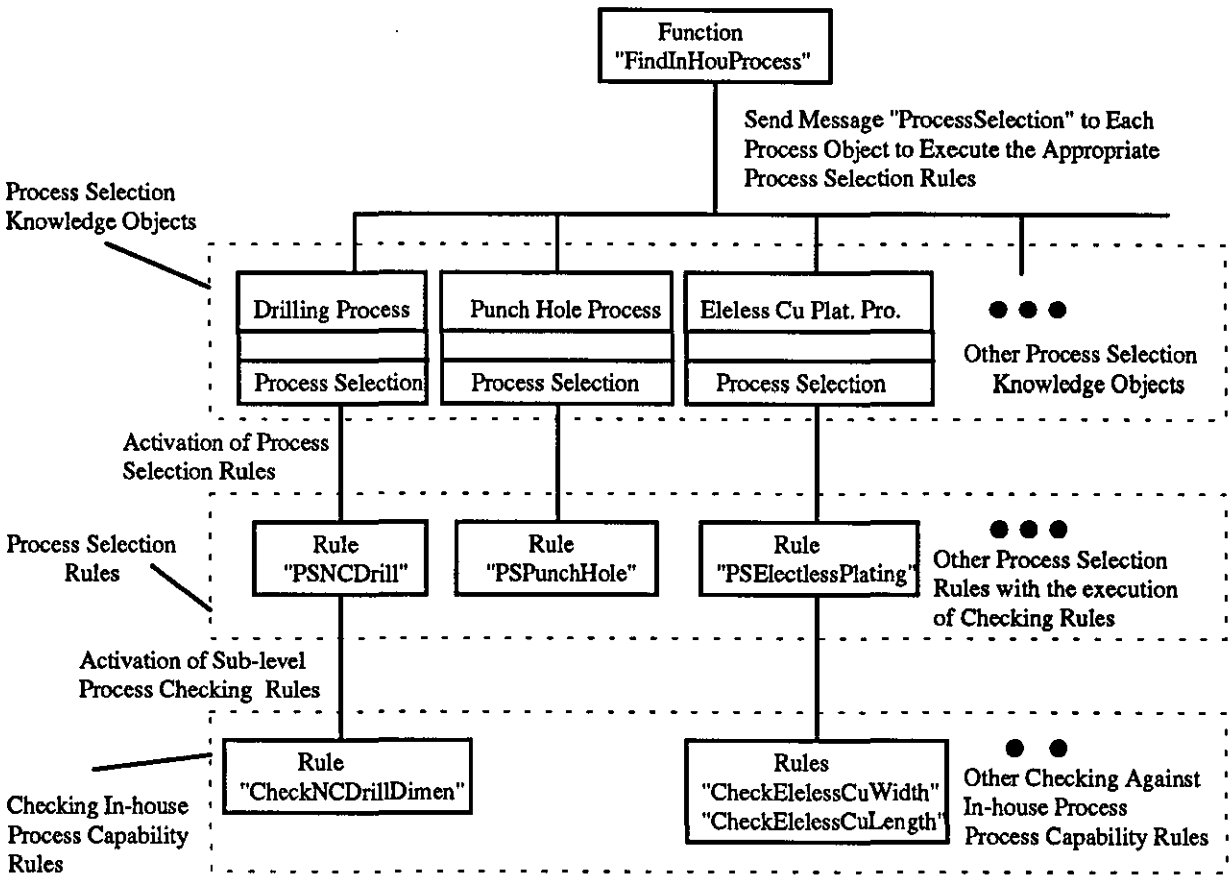


Figure 6.3 Hierarchy of the Process Selection Knowledge Objects, Process Selection Rules and the Process Capability Checking Rules

6.4.1 Rules to Select NC Drilling Process

If the hole location deviation requirement can be fulfilled by the in-house drilling machine and there is a plating requirement for the hole wall, the NC drilling process will be selected. The KAL implementation of the rule is shown on next page.

```
MakeRule( PSNCDrill, [],
  ProHoleSpec:LocDev >= InNCDrill:MachinedAcc And ProPlatingReq:CuThickMin
    > 0 And Not( Member?( SuggProInHou:ProcessName, NCDrilling ) ),
  {
  AppendToList( SuggProInHou:ProcessName, NCDrilling );
  Assert( InNCDrill, NCLength );   Assert( InNCDrill, NCWidth );
  ForwardChain( NULL, CheckNCDrillDimen );
  } );
```

Then, the panel size will be checked against the largest in-house NC drilling machine size by the "CheckNCDrillDimen" rule as shown.

```
MakeRule( CheckNCDrillDimen, [],
   InNCDrill:NCLength < ProPanelDimen:Length Or
              InNCDrill:NCWidth< ProPanelDimen:Width,
    PostMessage( "One side of product panel size longer than NC drilling machine size.  Product
cannot be produced in-house" ) );
```

6.4.2 Rules to Select Electroless Copper Plating Process

When there is a plating requirement for the hole wall, the electroless copper plating process and the pattern copper plating process will be selected. Then, the panel sizes of the board will also be checked against the tank sizes of the two processes.

```
MakeRule( PSElectlessPlating, [],
  ProPlatingReq:CuThickMin > 0,
  {
  AppendToList( SuggProInHou:ProcessName, ElectrolessCuPlat );
  Assert( InElelessCu, Width);   Assert( InElelessCu, Length );
  ForwardChain( NULL, CheckElelessCuLength, CheckElelessCuWidth );
  AppendToList( SuggProInHou:ProcessName, PatternCuPlat );
  Assert( InCu, Width);   Assert( InCu, Length );
  ForwardChain( NULL, CheckElecPlatProcessWidth, CheckElecPlatProcessLength );
  } );
```

6.4.3 Rules to Select Circuit Image Transfer Dry Film Process

If the circuit width is larger than the dry film process resolution, the circuit spacing larger than the dry film process resolution, the wet film process is not selected and the silk screen process is not selected, the dry film process will be selected for image transfer the board. The board size will then be checked against the dry film process maximum machine size.

```
MakeRule( PSImageTransDFCir, [],
  ProFeatCircuit:CirWidth >= InDryFilm:Resolution And ProFeatCircuit:SpaceWidth
    >= InDryFilm:Resolution And Not( Member?( SuggProInHou:ProcessName,
                                ImTranCirSilkScreen ) )
    And Not( Member?( SuggProInHou:ProcessName, ImTranCirWetFilm ) )
    And Not( Member?( SuggProInHou:ProcessName, ImTranCirDryFilm ) ),
  {
  AppendToList( SuggProInHou:ProcessName, ImTranCirDryFilm );
  Assert( InDryFilm, Width );
  Assert( InDryFilm, Length );
  ForwardChain( NULL, CheckImageTranDimen );
  } );
```

6.4.4 Rules to Select Bare Copper Plating Process

If a bare copper surface finish is required by the customer, the pattern tin-lead plating process, the tin-lead stripping process and the solder levelling process will be selected. The Tin/Lead plating tank dimension and the levelling process dimension will also be checked. The process selection rule is shown on the next page.

```
MakeRule( PSPatternBareCuPlat, [],
  ProPlatingReq:SurfaceFinType #= BareCopper And Not( Member?(
        SuggProInHou:ProcessName, PatternSnPbPlat ) ), {
                AppendToList( SuggProInHou:ProcessName, PatternSnPbPlat );
                AppendToList( SuggProInHou:ProcessName, SnPbStripping );
                AppendToList( SuggProInHou:ProcessName, SolderLevelling );
                Assert( InSnPb, Width);
                  Assert( InSnPb, Length );
                ForwardChain( NULL, CheckElecPlatProcessWidth,
                                                    CheckEecPlatProcessLength );
                   Assert( InSnPbLevelling, Length );
                   Assert( InSnPbLevelling, Width );
                   ForwardChain( NULL, CheckSnPbLevDimen );
  } );
```

6.4.5 Other Rules to Select Process

In addition to the above four process selection rules, a total of 16 rules have been developed for the selection of process and their names are shown below.

```
PSAuFinger
PSBlank
PSCompMarking
PSHolePunch
PSImageTransDFSM
PSImageTransSSCir
PSImageTransSSSM
PSImageTransWFCir
PSImageTranWFSM
PSImStripCuEtching
PSPanPatCuPlat
PSPatternAuPlat
PSPatternNiPlat
PSPatternSnPbPlat
PSRouting
PSSnPbReflow
```

A list of all these 16 process selection rules is at Appendix 7.2; the process capability checking rules are at Appendix 7.4.

6.5 Process Sequencing Knowledge

After the identification of the feasible processes for the board, the next function in the planning system is to sequence all the feasible processes into an appropriate manufacture sequence. The sequencing of the processes is deduced from the process sequencing knowledge. Circuit board manufacturing sequence knowledge is represented as declarative facts in this system. Based on data collected from the board manufacture experts and handbooks, the precedence and procedence processes of each manufacturing process are identified and a unique manufacturing sequence number is assigned to each of the manufacturing processes. Such a process sequence is represented by a sequence graph (Figure 6.4). Based on such process sequence knowledge and sequencing rules, the manufacturing sequence of all feasible processes is determined. When a new process is introduced later, an appropriate manufacturing sequence number between the precedence and the procedence processes' sequence number can be assigned. Using this method, sequencing rules and sequence numbers of the existing processes need not be modified when a new process is introduced into the system.

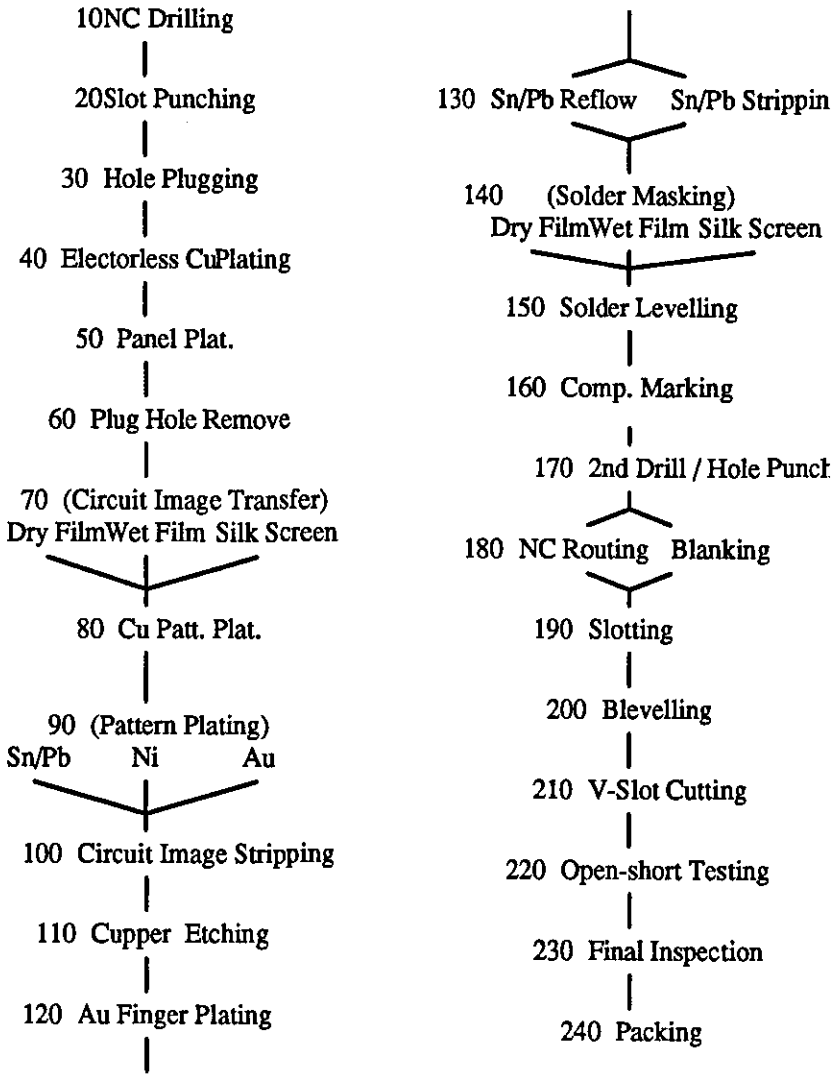


Figure 6.4 Sequence graph of the PCB Manufacturing Processes Considered in this Work

The sequence knowledge is represented as slot values in the two sequence knowledge objects. The slot called "Name" in the standard process name object called "*ProcessName*" is used to store the standard process name to be used in the system. The slot called "SeqNo" in the process sequence knowledge object ("*MfgSeqNo*") is used to store the manufacture sequence number of the corresponding process. The KAL representation of these two knowledge objects is shown on the next page. If a certain

process is not available in-house, that manufacturing sequence number will be replaced by "0" (such as the "VSlotCutting" process).

```
/************************************
 **** CLASS: ProcessName
 ************************************/
MakeClass( ProcessName, ProNameSeqNo );
MakeSlot( ProcessName:Name );
SetSlotOption( ProcessName:Name, MULTIPLE );
SetValue( ProcessName:Name, NCDrilling, SlotPunching, HolePlugging, ElectrolessCuPlat,
PanelCuPlat, PlugHoleRemove, ImTranCirDryFilm, ImTranCirWetFilm,
ImTranCirSilkScreen, PatternCuPlat, PatternNiPlat, PatternAuPlat, PatternSnPbPlat,
CirImageStrip, CuEtching, AuFinger, SnPbReflow, SnPbStripping, ImTranSMDryFilm,
ImTranSMWetFilm, ImTranSMSilkScreen, SolderLevelling, CompMarkPrint, 2ndDrill,
OutLineRout, OutLineBlank, Slotting, Blevelling, VSlotCutting, OpenShortTesting,
FinalInspection, Packing );


/************************************
 **** CLASS: MfgSeqNo
 ************************************/
MakeClass( MfgSeqNo, ProNameSeqNo );
MakeSlot( MfgSeqNo:SeqNo );
SetSlotOption( MfgSeqNo:SeqNo, MULTIPLE );
SetSlotOption( MfgSeqNo:SeqNo, VALUE_TYPE, NUMBER );
SetValue( MfgSeqNo:SeqNo, 10, 20, 30, 40, 50, 60, 70, 70, 70, 80, 90, 90, 90, 100, 110, 120,
130, 130, 140, 140, 140, 150, 160, 170, 180, 180, 190, 200, 0, 220, 230, 240 );
```

The sequence rules used to sequence all feasible processes are shown on the next page. These rules are implemented in the "SeqProcess" function and will be executed when the "SeqProcess" button instance is activated.

```
/**** FUNCTION: SeqProcess ******/
ResetValue(SuggProInHou:ProcessNameSeqNo);
    EnumList(SuggProInHou:ProcessName, name, AppendToList( SuggProInHou:
      ProcessNameSeqNo,
      {GetNthElem(MfgSeqNo:SeqNo, GetElemPos(ProcessName:Name, name))})
    );
  ClearList(SuggProInHou:ProcessNameInSeq);
  For numb From 0 To 300 By 10
    Do { Global:dumno = GetElemPos(SuggProInHou:ProcessNameSeqNo, numb);
         If Global:dumno > 0
           Then {AppendToList(SuggProInHou:ProcessNameInSeq,
                    GetNthElem(SuggProInHou:ProcessName, Global:dumno)) ;
    };
```

At the start of system operation, the process names in the "ProcessNameSeqNo" slot of the object "*SuggProInHou*" are reset. Then the manufacturing sequence of each feasible process is put in the slot "ProcessNameSeqNo" of the "*SuggProInHou*" object. The manufacturing sequence number in the list is then compared with a number set by the "For" loop recurrently, each feasible process name will then be put in the "ProcessNameInSeq" slot of the "*SuggProInHou*" object according to the manufacturing sequence. The sequenced process names in the "ProcessNameInSeq" will be output to the user or included in the operation instruction sheet.

6.6 Machine Selection Knowledge

Machine selection knowledge is used to select the appropriate machine(s) to manufacture the board. The NC drilling process, blanking process and silk screen process are assumed to have more than one machine available. Their respective machine selection rules are implemented in the "MachineSelection" operation of the process object (Figure 6.5). For all the selected processes with more than one machine, a message will be sent to each

process object and the appropriate "SelectMachine" rule will be executed when the "*MachineSelection*" button instance is activated.

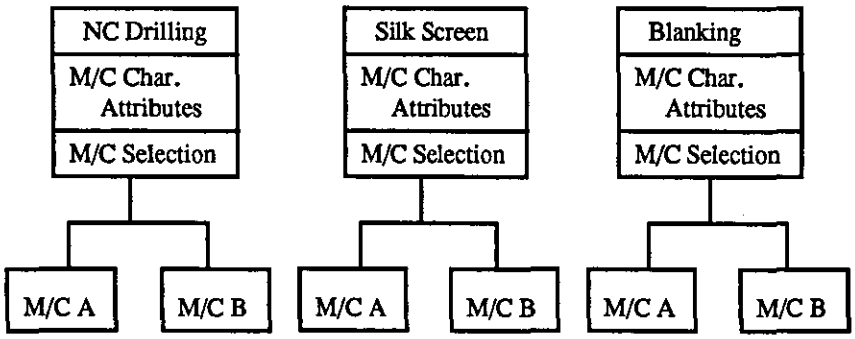


Figure 6.5 Object Model of Machine Selection Process

The drilling machine selection rules are represented as follows:

```
ForAll[mc!DrillMC]
    {If mc:MachinedAcc < ProHoleSpec:LocDev And
        mc:NCLength > ProPanelDimen:Length And
        mc:NCWidth > ProPanelDimen:Width And
        mc:MaxHoleSize > ProHoleSpec:MaxHoleSize And
        mc:MinHoleSize < ProHoleSpec:MinHoleSize
      Then (AppendToList(NCDrilling:MCListing, mc:MachineName));
    };
```

It means that the slot values of the drilling machine object will be compared with the appropriate product requirements. If a drilling machine is capable of manufacturing the board, that machine will be selected and its name will be put in the "MCListing" slot. The "ForAll[mc!DrillMC]" command will loop over all the NC drilling machine objects in the

drilling process to identify all the machines which are capable of manufacturing the board. In future, when a new drilling machine is purchased, the characteristic of that drilling machine should be represented using the drilling machine object class structure and added under the "*NCDrilling*" object class. Then, the new drilling machine will be included into consideration by the "ForAll[mclDrillMC]" command during the machine selection process. This is the "inheritance-data-structure" advantage by using an object oriented programming approach.

Similarly, the following rules are used in the selection of machine for silk screen process, the solder masking process and the component marking process. The resolution, length and width of the machine will be checked against the product requirements to test if that machine can be selected.

```
ForAll[mclSilkScreenMC]
    {If mc:Resolution < ACC And
        mc:Length > ProPanelDimen:Length And
        mc:Width > ProPanelDimen:Width
    Then (AppendToList(MCLIST:MCListing, mc:MachineName));};
```

Similarly, the blanking machine selection rule is also developed:

```
ForAll [ mclBlankMC ]
        {
        If ( mc:Accuracy < ProPanelDimen:DimenDev And mc:Length
              > ProPanelDimen:Length And mc:Width > ProPanelDimen:Width )
         Then AppendToList( BlankingMC:MCListing, mc:MachineName );
        };
```

6.7 Process Recipe Generation Knowledge

Process recipe generation knowledge of panel copper plating, pattern copper plating, pattern tin-lead plating, pattern nickel plating, pattern gold plating and gold finger plating are developed and represented by rules as listed at Appendix 7.6.

For example, the nickel plating process recipe generation knowledge is represented by the following rule:

```
MakeRule( RecipeGenPatternNi, [],
  TRUE,
  {
  RecPatPlatNi:Current = CalPatternArea( ProPanelDimen:Width, ProPanelDimen:Length,
     ProPanelDimen:UnitPerPanel, ProPanelDimen:UnitWidth, ProPanelDimen:UnitLength,
     ProFeatCircuit:CircuitArea ) * InNi:CurrentDensity;
  RecPatPlatNi:PlatingTime=( InNi:TimeToDeposit * ProPlatingReq:SurfaceFinThick * 10 );
  } );
```

The plating current and the plating time of the nickel plating process are calculated. Plating current is determined by the plating area of the panel and the current density of the process. Plating time depends on the required thickness of surface finishing and the time needed to deposit certain thickness.

The calculation of the plating area is implemented by the function "CalPatternArea". Data on width of the panel, length of the panel, number of unit in each panel, width of the units, length of the units and circuit area of each unit will be used by this function to calculate the plating area of the panel. All other pattern plating recipe rules will also use this function to calculate the pattern plating area. The "CalPatternArea" function is shown on next page.

```
/*************************************
    **** FUNCTION: CalPatternArea
    *************************************/
MakeFunction( CalPatternArea, [panwidth panlength noofunit unitwidth unitlength unitcirarea],
{
  Global:dumno = ((panwidth * panlength) - (unitwidth * unitlength * noofunit)) * .6/25.4/25.4/144;
  If ( ProLamChar:CuFoilSide #= Double)
        Then ( Global:dumno = Global:dumno * 2 );
  Global:dumno + ( unitcirarea / 144 * noofunit );
} );
```

The current density constant and the "time to deposit" constant of the various plating processes are all stored as slot values in the various plating process objects under the in-house process capability object class and as shown below:

```
InCu:CurrentDensity = 40;
InCu:TimeToDeposit = 3;

InAu:CurrentDensity = 5;
InAu:TimeToDeposit = 38;

InSnPb:CurrentDensity = 20;
InSnPb:TimeToDeposit = 2.5;

InNi:CurrentDensity = 30;
InNi:TimeToDeposit = 3.86;
```

Similar rules are used to represent other plating process recipe generation knowledge.

6.8 Summary

Knowledge representation is one of the key issues in a knowledge-based system. This chapter has addressed this key issue. The knowledge representation structure adopted in

the system was discussed. The five categories of knowledge:- product requirements checking knowledge, process selection knowledge, process sequencing knowledge, machine selection knowledge and process recipe generation knowledge were examined. The inference mechanism of each knowledge structure was illustrated. Justifications for using the object orientation approach to implement the inference mechanism were given. The advantage of adopting the approach has been pointed out that new rules could be incorporated into the knowledge base whenever required without the requirement of modifying the inference structure.

CHAPTER 7
IMPLEMENTATION AND OPERATION OF THE KB PCB CAPP SYSTEM

7.1 Introduction

This chapter summarises the implementation and operation of the KB PCB CAPP system using the Kappa object oriented, knowledge-based software. The implementation of the data loading session is first discussed. The object model and the dynamic model of the process selection session and process sequence session are then introduced, with a description of their implementation. A discussion on the modelling of the machine selection session and its operation follows. The implementation of the process recipe and instruction generation session using object model and the dynamic model is explained at the close.

7.2 Implementation of the Process Selection and Sequencing Session

There are two parts in this session: loading of data objects and process selection. The implementation of the latter part is explained using the object model and the dynamic model.

7.2.1 Loading of Data Objects

When the user activates the "Load Cust. Gen. Spec. & Board Reqt." option ("Button23") in the main session window (Figure 7.1), the "LdCustSpBdReqt" function (Figure 7.2) is

executed, loading the customer general specification data objects and the product requirements data objects onto the system.

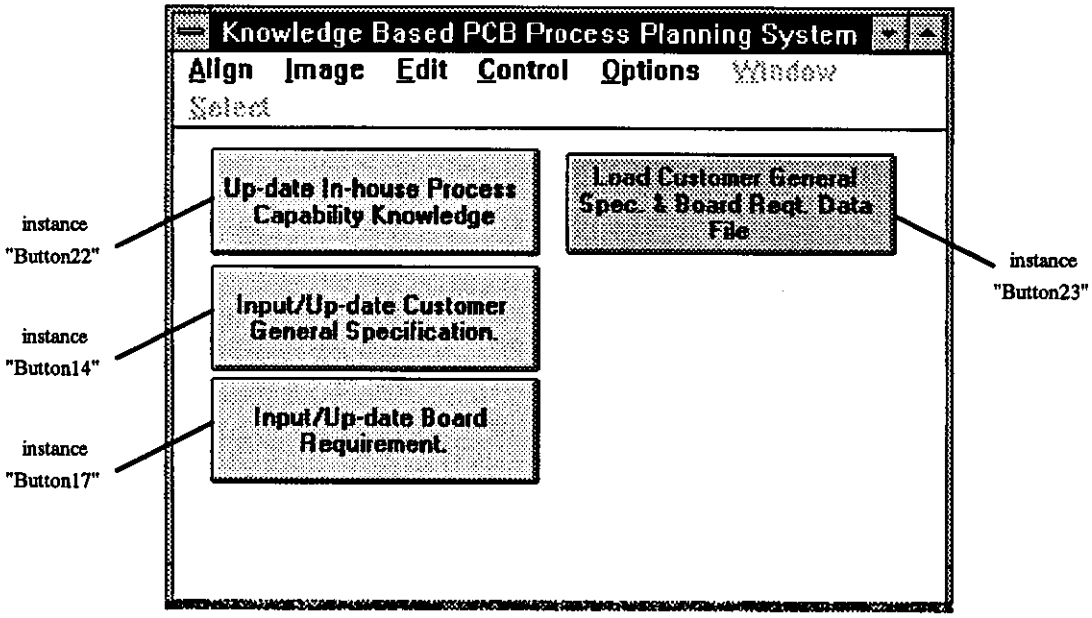


Figure 7.1 The Main Session Window

```
MakeFunction( LdCustSpBdReqt, [],
  {
  If Class?( ReqDocInfo )
   Then PostMessage( "ERROR!! Customer and/or Product Data Base Already Loaded!!!" )
   Else {
      SendMessage( CustGenSpec, LoadData );
      SendMessage( CustProdReqt, LoadData );
      SendMessage( CustProdReqt, CheckCustGenSpecMethod );
      PostMessage( "Customer General Specification Data and Product Requirements Data
Successfully Loaded. ");
      ShowImage( Button18 );
      HideImage( Button23 );
      HideImage( Button22 );
      HideImage( Button14 );
      HideImage( Button17 );
      };
  } );
```

Figure 7.2 The "LdCustSpBdReqt" Function

First, the existence of customer product requirements object in the planning system is checked. If the customer product requirements object does not exist, the function continues by hiding the three data base interface buttons ("Button22", "Button14" and "Button17") from the session window to prevent their accidental selection. The user is, then, allowed to load the customer general specification data file and the product requirements data file onto the system by inputting the respective file names. Then, the "SendMessage" command will send messages to activate the operation "LoadData" of the "*CustomGenSpec*" objects and the operation "LoadData" of the "*CustProdReqt*" objects (Please refer to Figure 5.11 in Chapter 5 for the operation of the "LoadData" in the "*CustGenSpec*" object). After the customer general specification data objects and the product requirements data objects are loaded, the checking rules as described in Section 6.3 are then activated by the "SendMessage(CustProdReqt, CheckCustGenSpecMethod)" command; the product requirements are therefore checked against the customer specifications. If out of specification parameters are found, the user will get a notification message interactively. Then, the successful loading of the data objects will then be indicated. The button to activate the process selection session ("Button18") then appears and the other four buttons will be hidden from the session window.

7.2.2 Process Selection and Sequencing

After the successful loading of the customer general specification and product requirements data base, a "Process Selection and Sequencing Session" button ("Button18") will appear for the user to start the process selection activities. The developed object model and the dynamic model (in state diagram form) of the process selection and sequencing session are presented respectively in Figure 7.3 and Figure 7.4.

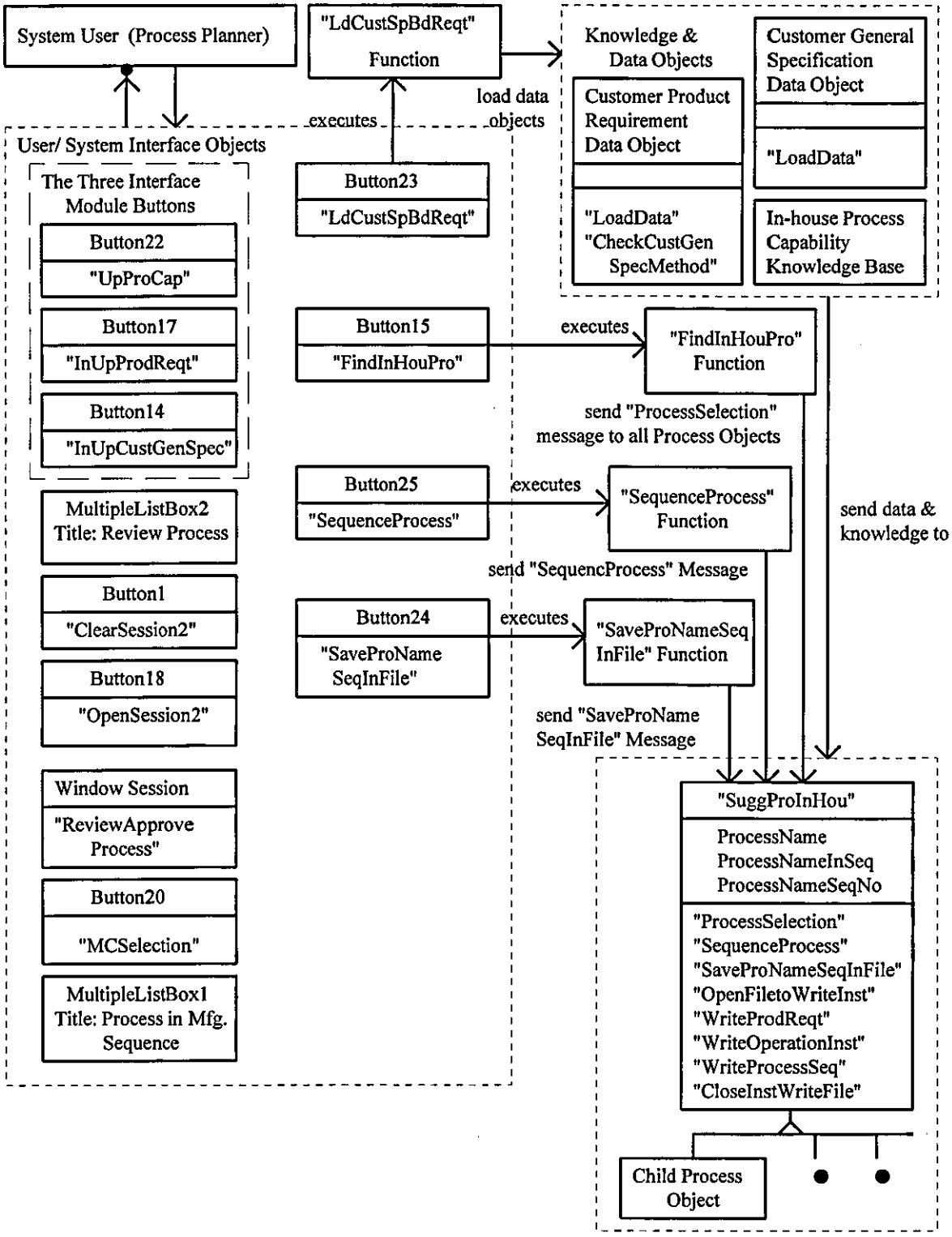


Figure 7.3 Object Model of the Process Selection and Sequencing Session

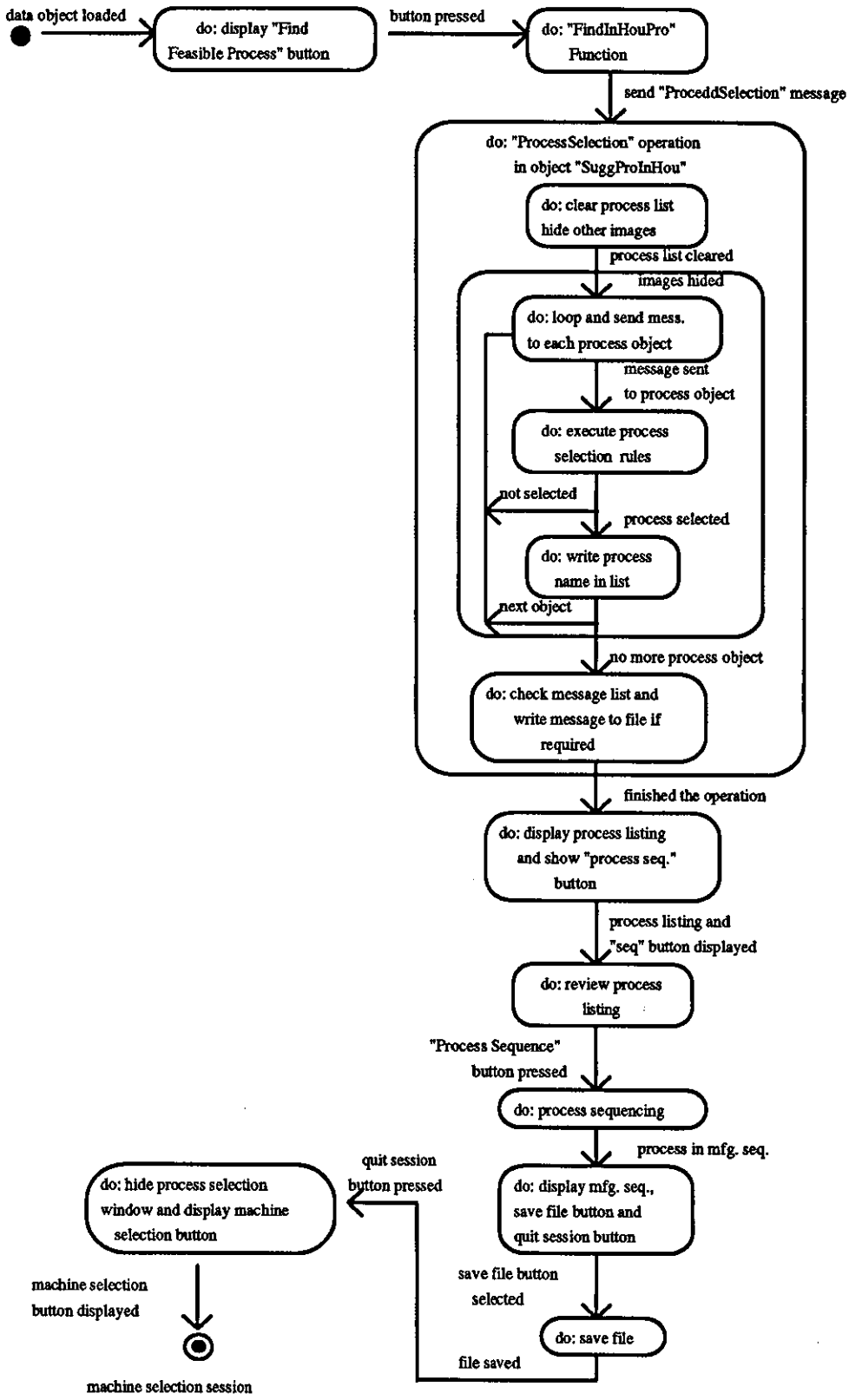


Figure 7.4 The Dynamic Model (in State Diagram form) of the Process Selection and Sequence Session

When the "Process Selection and Sequencing Session" button ("Button18) is activated, the "Review and Approved Process" session window (Figure 7.5) and the "Find Feasible Process" button ("Button15") will appear on the screen.

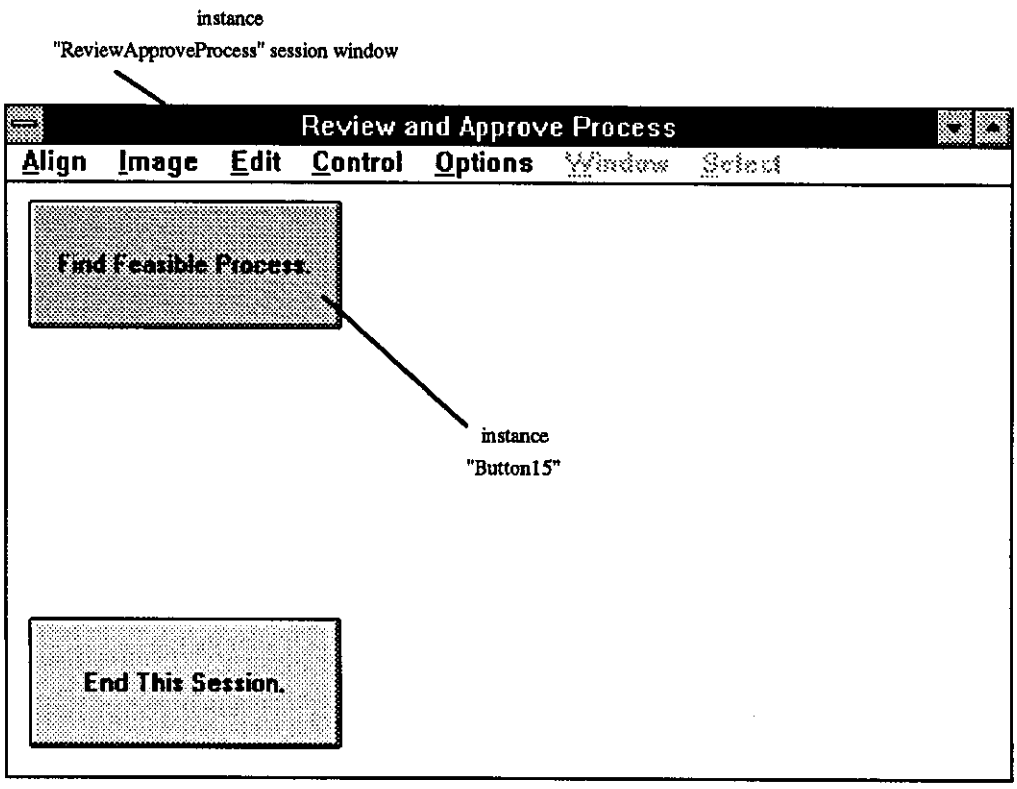


Figure 7.5 The "Review and Approved Process" Session Window

When the "Find Feasible Process" button is clicked, a message will be sent to activate the "ProcessSelection" method in the "*SuggProInHou*" object. The commands of the method is shown below.

```
  /************** METHOD: ProcessSelection  **************/
MakeMethod( SuggProInHou, ProcessSelection, [],
  {
  ClearList( SuggProInHou:ProcessName );
  ClearList(MessageFile:ProCanNotMfgInHouse);
   HideImage( MultipleListBox2 );
  HideImage( Button25 );
  HideImage( Button24 );
  ForAll [ process|SuggProInHou ]
      SendMessage( process, ProcessSelection );
  If ( LengthList( MessageFile:ProCanNotMfgInHouse )
       > 0 )
   Then {
      SetValue( Global, dumstring, Global:FileNameListProd#".msg");
      OpenWriteFile( Global:DriveDir # Global:dumstring, APPEND );
      WriteLine( "/* Process(es) Cannot Perform In-house*/" );
      WriteLine( "/* Process Name(s) -- Problem */" );
      EnumList (MessageFile:ProCanNotMfgInHouse, message, WriteLine(message));
      CloseWriteFile();
      PostMessage("Process(es) Cannot Perform In-house Message Stored in File"#
      Global:DriveDir#Global:dumstring);
    };
  ShowImage( MultipleListBox2 );
  ShowImage( Button25 );
  } );
```

The method starts by clearing the values in the "ProcessName" slot and the "ProCanNotMfgInHouse" slot and then hides all the unused buttons and images. Messages will, then, be sent to activate the "ProcessSelection" operations of all child process objects (Figure 7.6) of the "*SuggProInHou*" object. Each operation will execute the process selection rules (as discussed in Section 6.4 of Chapter 6) of each object to establish whether the process is required in the manufacture of the required circuit board. If the process is required, the standard name of the process will be written onto the "ProcessName" slot of the "*SuggProInHou*" object. When the execution all the

"ProcessSelection" operations of all the child process objects is completed, the "LengthList (MessageFile:ProCanNotMfgInHouse)" command is then executed to check if a message is found in the "ProCanNotMfgInHouse" slot of the "*MessageFile*" object. If a message is found, it will be written into a file for later reference later.

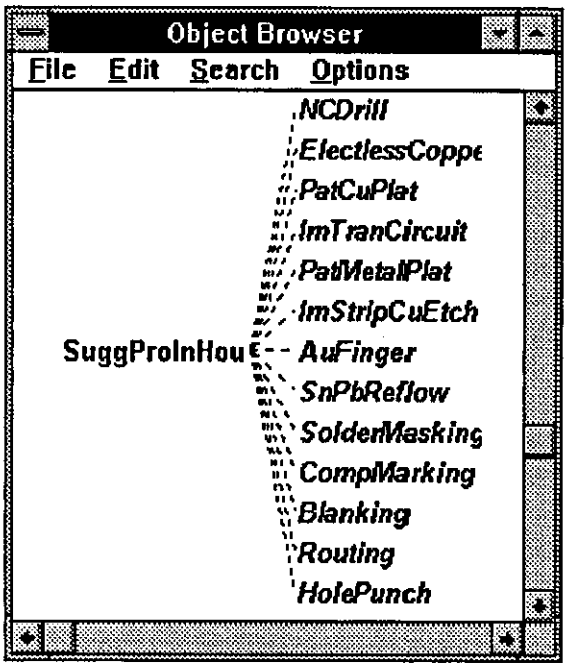


Figure 7.6 Process Objects Under the "*SuggProInHou*" Object

Then, the "Review System Suggested Feasible Process" window ("*MultipleListBox2*" image object) will display all the selected process names for the user to review and to approve (Figure 7.7). All feasible process names will be high-lighted and the user can delete any process names from or add new process names to the list. The user, then, can proceed to the process sequencing session by pressing the "Sequence Approved Process" button ("Button25"). The button will activate the process sequencing rules as described in Section 6.5 of Chapter 6. The process names will be arranged in the appropriate manufacturing sequence and written in the "ProcessNameInSeq" slot of the "SuggProInHou" object.

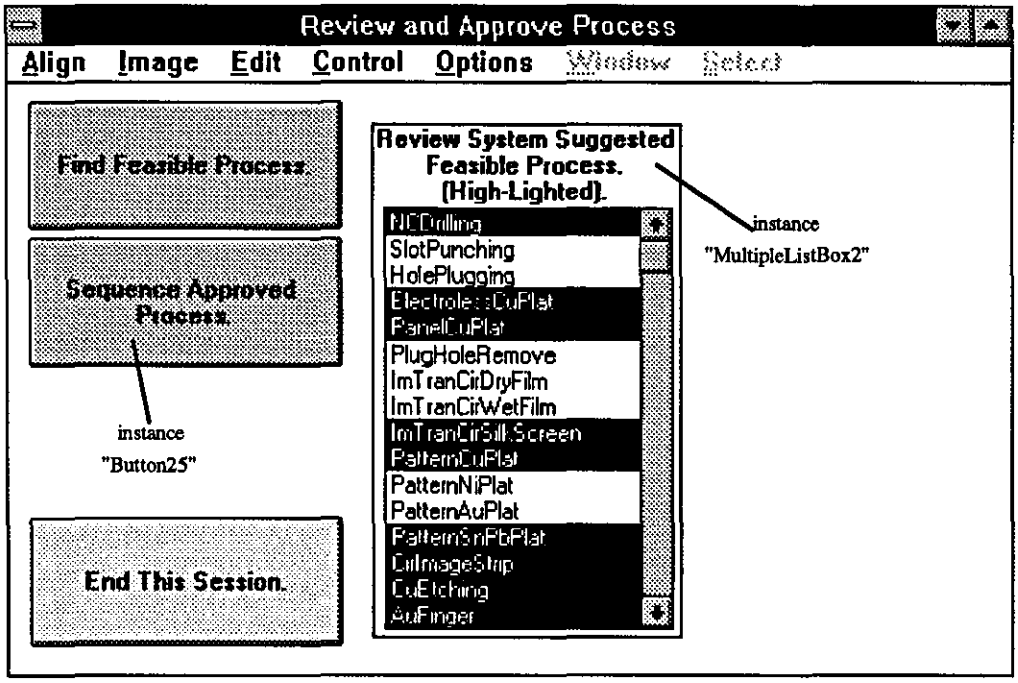


Figure 7.7 System Suggested Feasible Process

The sequenced process names will be displayed through the "Process in Mfg. Sequence" window ("MultipleListBox1" image) (Figure 7.8). The user is then allowed to save the sequenced process name in a text file by clicking the "Save Process Name" button ("Button24") or clicking the "End This Session" button ("Button1") to quit the "Review and Approve Process" session.

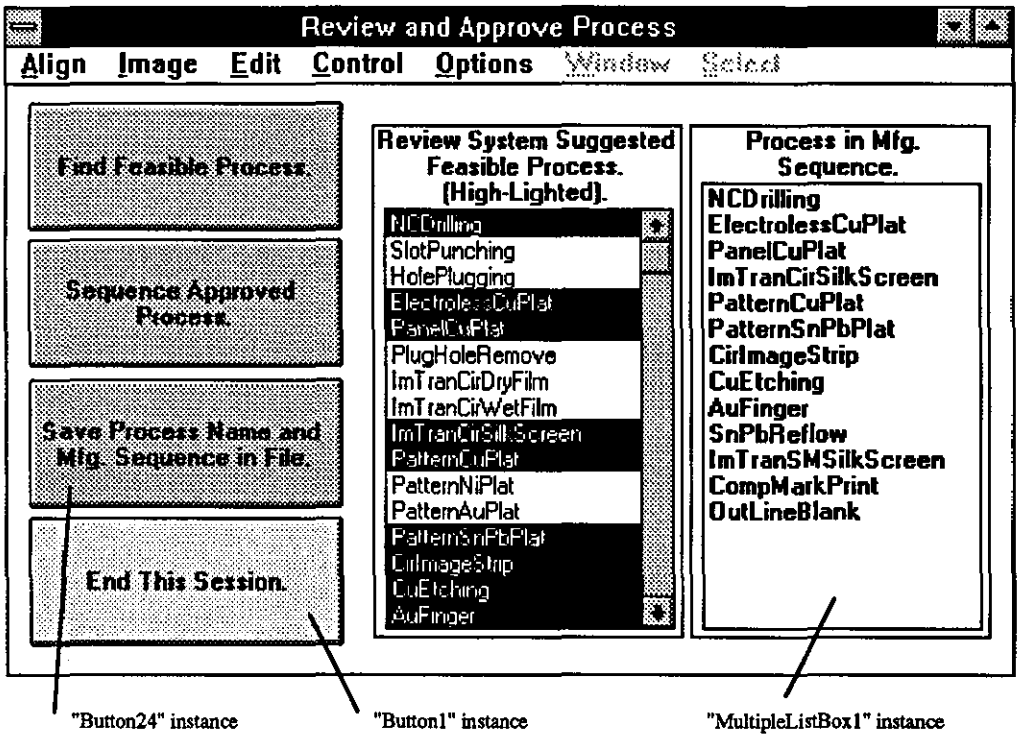


Figure 7.8 Process Names in the Manufacturing Sequence

7.3 Machine Selection Session

Figure 7.9 is the object model and Figure 7.10 is the dynamic model (in a state diagram form) of the machine selection session. When the user clicks the "Machine Selection" button ("Button20"), the "MCSelection" function will be activated. The "MCSelection" function will start checking the existence of the product data object in the system. If the product data object does not exist in the planning system, a message to show the problem will be posted to the user. Otherwise, a message will be sent to the "*MCNameList*" object and activate the "MCSelection" operation.

The KAL commands of the "MCSelection" operation are shown as follows:

```
/************** METHOD: MCSelection  **************/
MakeMethod( MCNameList, MCSelection, [],
{
  ClearList( MessageFile:MCSelectionNoMessage );
  ForwardChain( [ NOASSERT ], NULL, MCSelectionRules );
  If ( LengthList( MessageFile:MCSelectionNoMessage ) > 0 )
    Then {
      SetValue( Global, dumstring, Global:FileNameListProd#".msg");
      OpenWriteFile( Global:DriveDir # Global:dumstring, APPEND);
      WriteLine( "/* Process(es) WITHOUT Machine Assigned */" );
      WriteLine( "/* Process Name(s) */ ");
      EnumList (MessageFile:MCSelectionNoMessage, message, WriteLine(message));
      CloseWriteFile();
    };
  PostMessage( "Machine Selection Finished.  Go to Recipe & Instruction Sheet
                    Generation Session." );
} );
```

The "MCSelectionNoMessage" slot will be reset and the "MCSelectionRules" will be activated to select machines for the processes. The operation will then check for any message in the slot of the "*MessageFile*" object. If a message is found, it will be written onto the message file for later reference.

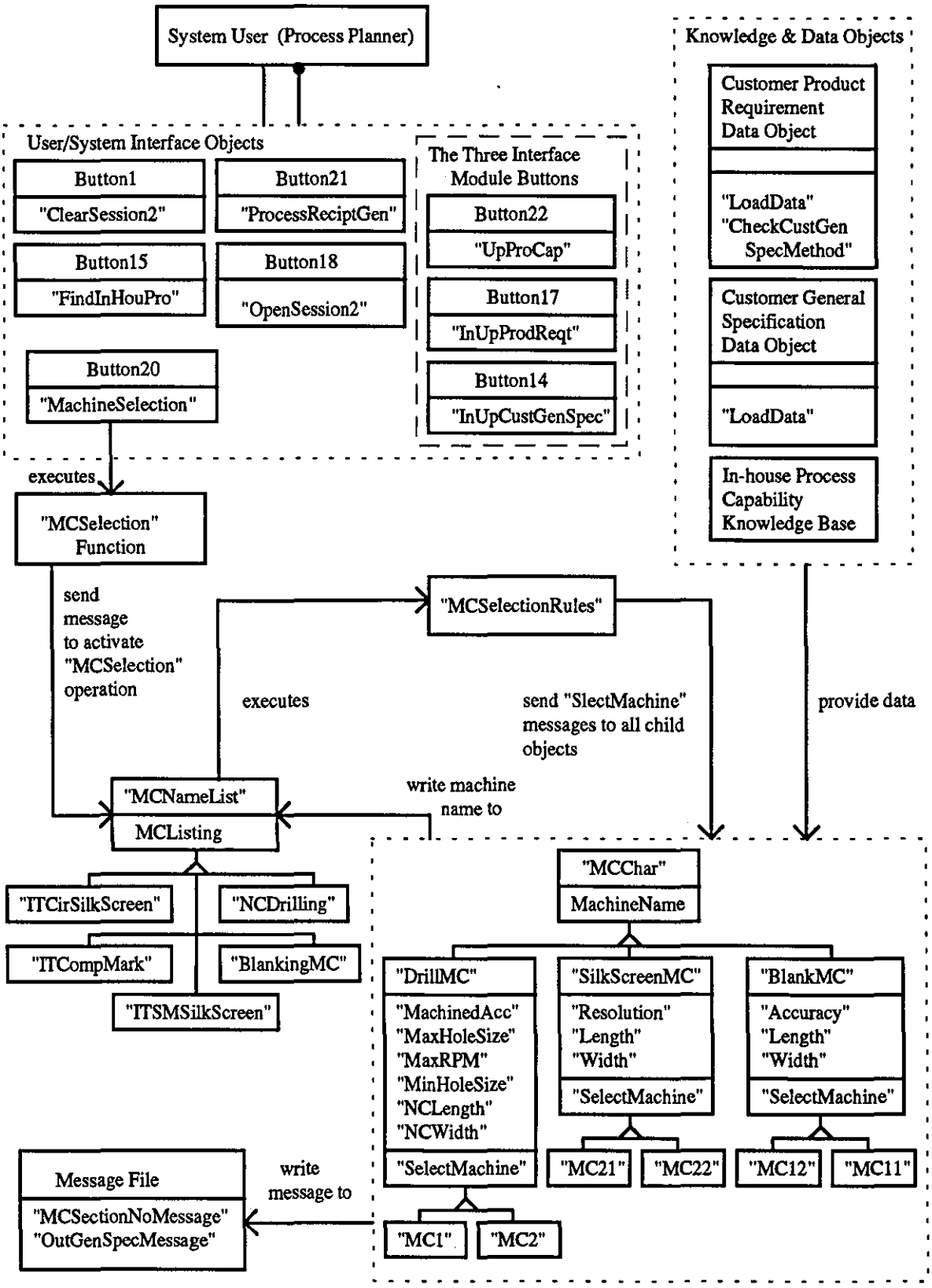


Figure 7.9 Object Model of the Machine Selection Module

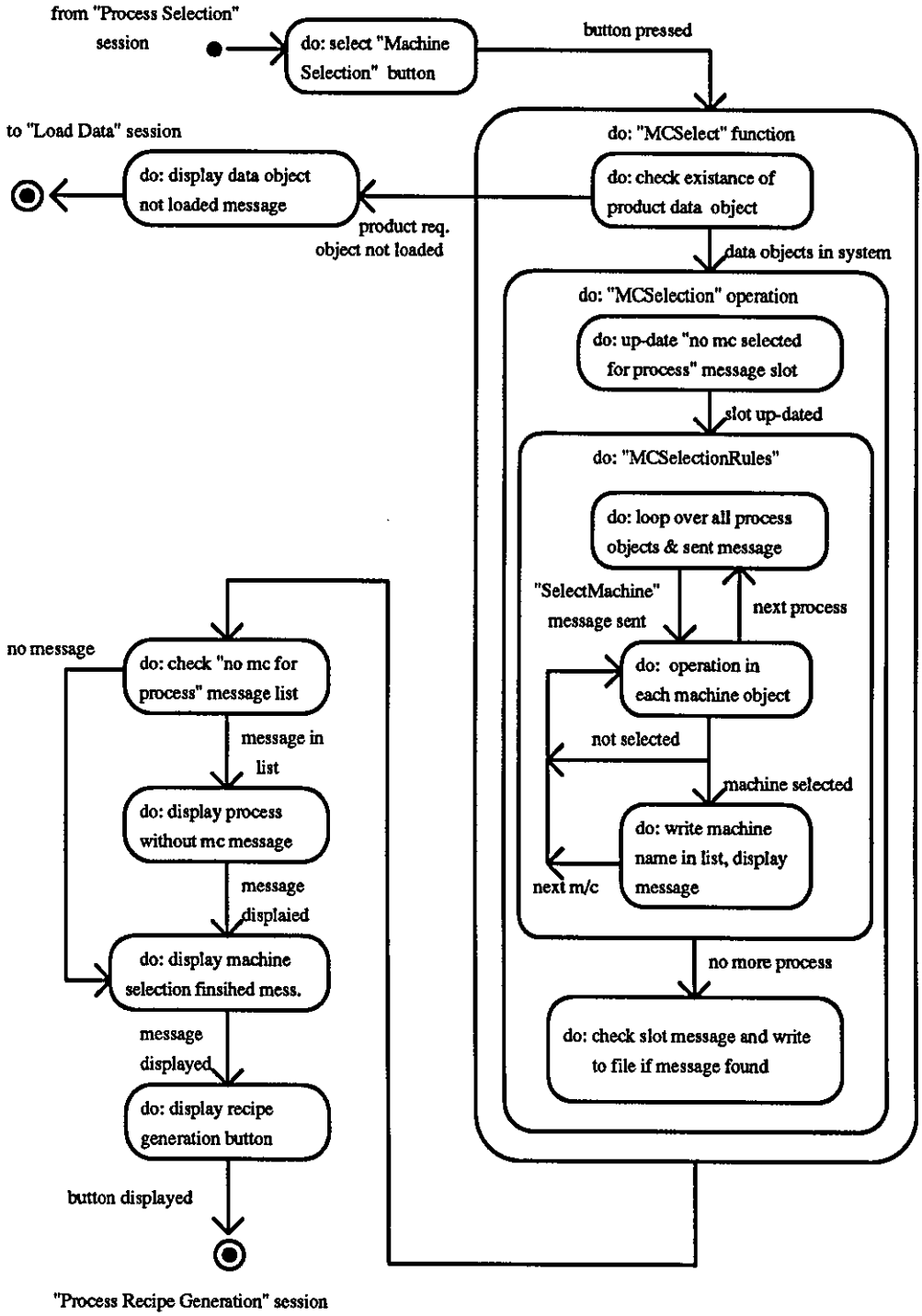


Figure 7.10 State Diagram of the Machine Selection Session

The "MCSelectionRules" are shown as follows:

```
MakeRule( MCSelectionRules, [],
  TRUE,
  {
  EnumSubClasses( MCNameList, x,
    {
    ClearList( x:MCListing );
    } );
  EnumList( SuggProInHou:ProcessName, x,
    {
    If ( x #= NCDrilling )
     Then SendMessage( DrillMC, SelectMachine );
    If ( x #= OutLineBlank )
     Then SendMessage( BlankMC, SelectMachine );
    If ( x #= ImTranCirSilkScreen )
     Then SendMessage( SilkScreenMC, SelectMachine, ProFeatCircuit:AnnularRing,
          ITCirSilkScreen );
    If ( x #= ImTranSMSilkScreen )
     Then SendMessage( SilkScreenMC, SelectMachine, ProSoldMask:MinClear,
          ITSMSilkScreen );
    If ( x #= CompMarkPrint )
     Then SendMessage( SilkScreenMC, SelectMachine, 0.3, ITCompMark );
    } );
  ShowImage( Button21 );
  } );
```

First, the "MCListing" slot values of all the child process objects of the "*MCNameList*" object are reset. Then, the "ProcessName"s will be compared with each name of those processes with more than one machine. If the process names match, a message will be sent to activate the "SelectMachine" operation of that process object. The machine selection rules as described in Section 6.6 will be executed to check which machine in that process object is capable of manufacturing the required circuit board. The name of the capable machine will be written onto the "MCListing" slot of the process object. After all the machine objects are checked, the "MCListing" slot of that process will then be checked. If no machine name is found in the "MCListing" slot, the user will be informed by the message: "NO Suitable Machine for This Process Find In-house". Afterwards, a

similar message will also be put into the empty "MCListing" slot. After all the process objects and machine objects are checked, the user is prompted that the machine selection session is ended. A "Recipe and Instruction Generation" button ("Button21") will appear for the user to proceed to the recipe generation session.

7.4 Process Recipe Generation Session

The object model and the dynamic model of the recipe and instruction generation session are developed and presented respectively in Figures 7.11 and Figure 7.12.

When the user selects the "Recipe and Instruction Generation" button ("Button21"), the "ProcessRecipe" function will be activated. The commands of this function are shown below.

```
MakeFunction( ProcessRecipeGenFunction, [],
  {
  If Class?( ReqDocInfo )
    Then {
        SendMessage( ProcessRecipe, GenerateProcessRecipe );
        SaveProdCharProcessPlan( );
        ClearTranscriptImage( Transcript1 );
        DisplayFile( Transcript1, Global:DriveDir # Global:dumstring );
        ShowWindow( ProcessPlanOutputSession );
        ShowImage( Button16 );
        ShowImage( Button26 );
        }
    Else {
        PostMessage( "ERROR!! Customer and/or Product Data Base NOT Loaded. Load
Data Base First." );
        };
  } );
```

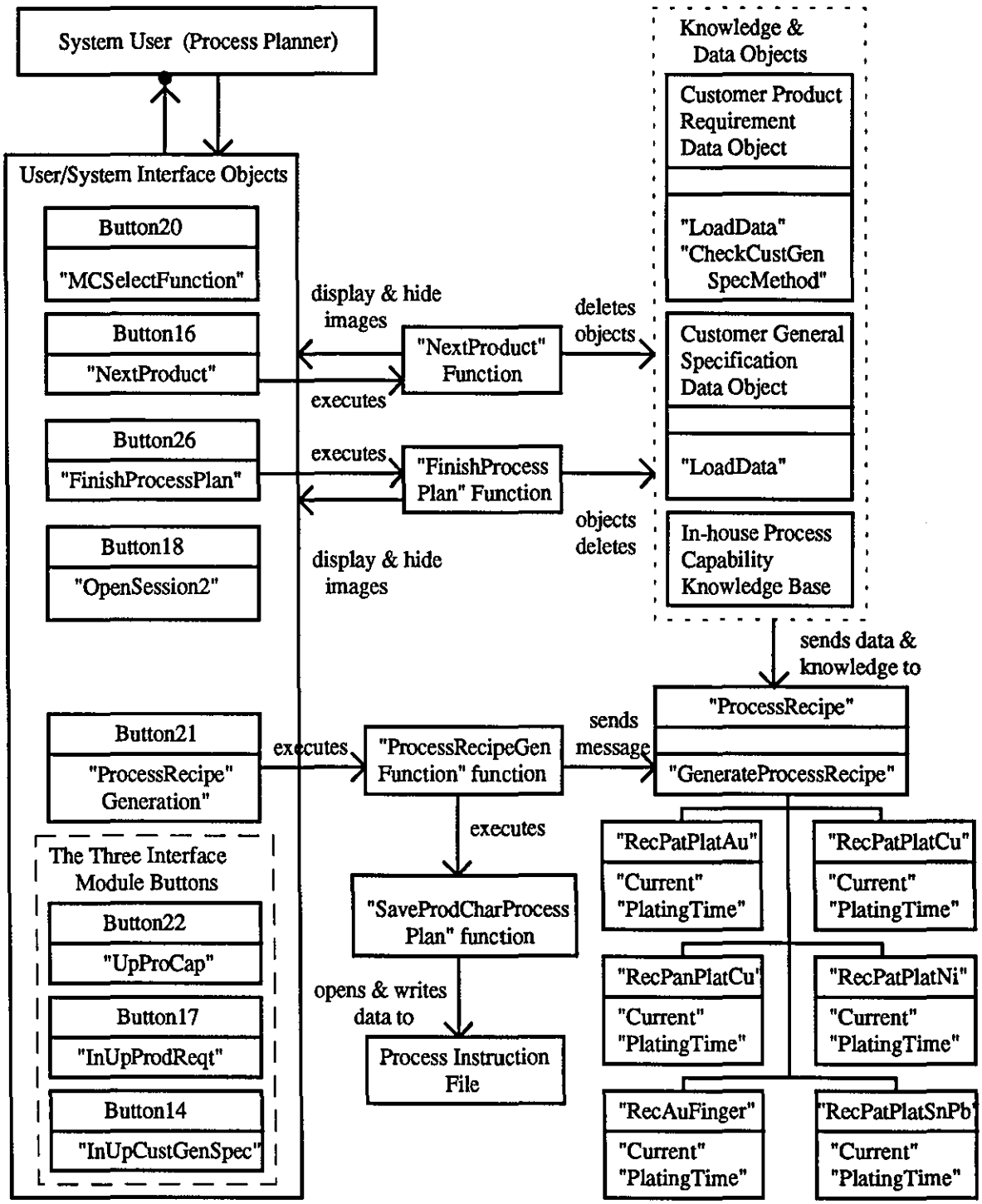


Figure 7.11 Object Model of the Process Recipe Generation Module

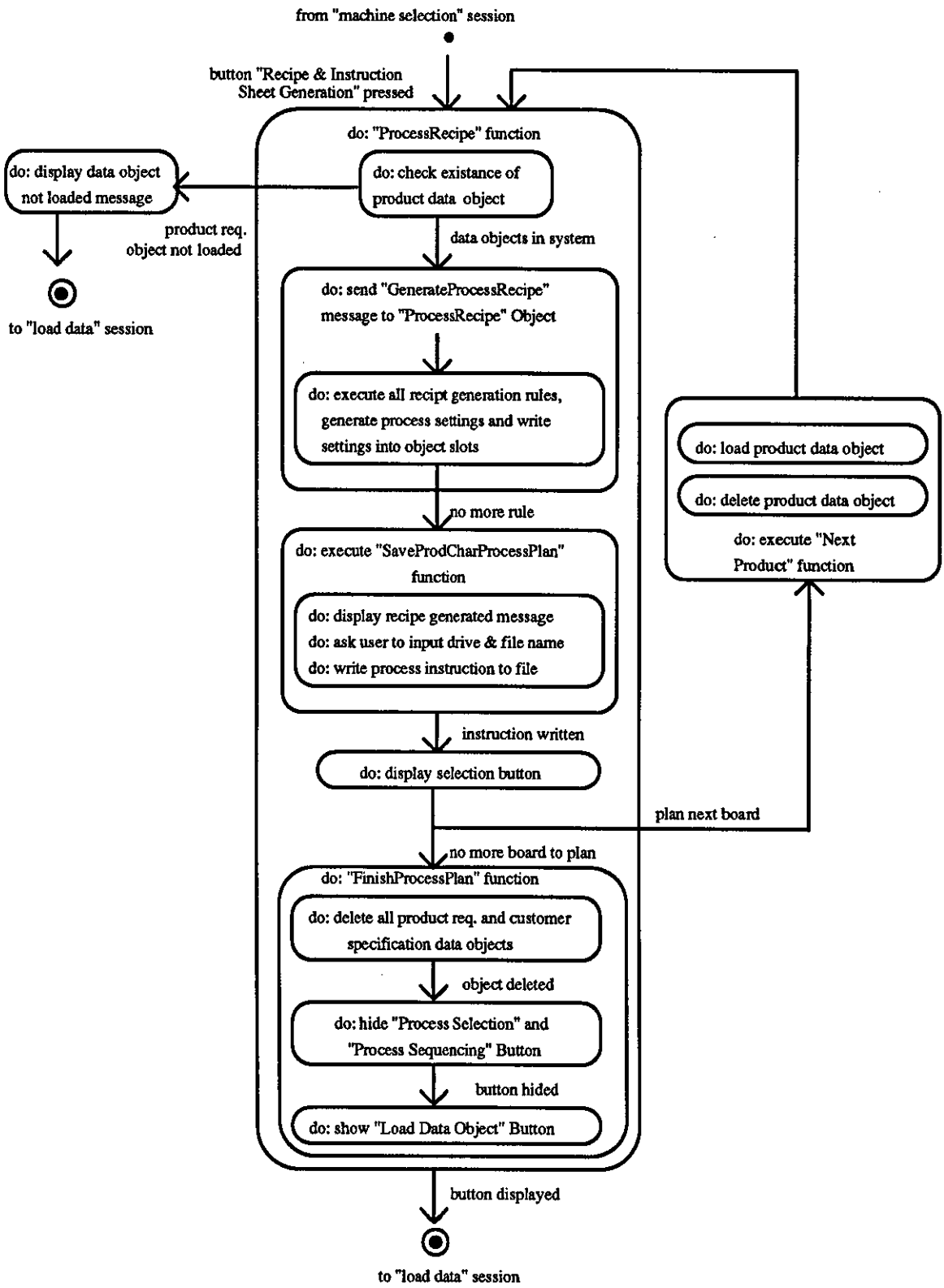


Figure 7.12 Dynamic Model (in form of State Diagram) of the Process Recipe and Instruction Generation Session

The "ProcessRecipeGenFunction" starts with checking the existence of the product data object in the system. If the data object is loaded, the "GenerateProcessRecipe" operation in the "*ProcessRecipe*" object will be activated. The commands in this operation are shown below.

```
/************** METHOD: GenerateProcessRecipe  **************/
MakeMethod( ProcessRecipe, GenerateProcessRecipe, [],
  {
  ForwardChain( [ NOASSERT ], NULL, CheckGenPanelCuPlatRecipe);
  ForwardChain( [ NOASSERT ], NULL, CheckGenPatternCuPlatRecipe);
  ForwardChain( [ NOASSERT ], NULL, CheckGenPatternNiPlatRecipe);
  ForwardChain( [ NOASSERT ], NULL, CheckGenPatternAuPlatRecipe);
  ForwardChain( [ NOASSERT ], NULL, CheckGenPatternSnPbPlatRecipe);
  ForwardChain( [ NOASSERT ], NULL, CheckGenAuFingerRecipe );
  } );
```

The rules will activate the "GenerateProcessRecipe" operation of all the child objects of the "*ProcessRecipe*" object. The process recipe object and its child objects are shown in Figure 7.13.

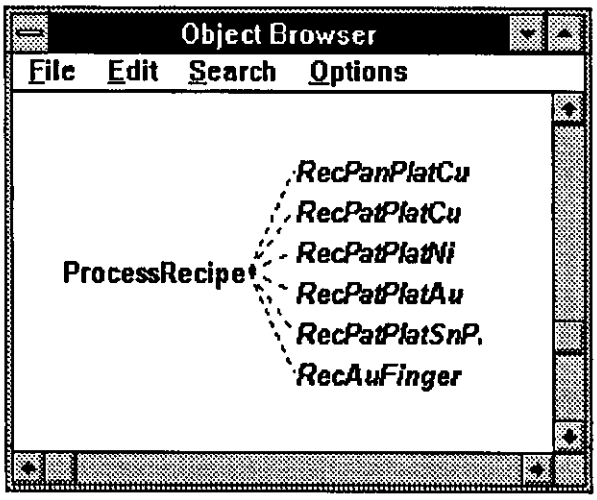


Figure 7.13 The Six Child Objects under the "*ProcessRecipe*" Object

The process recipe knowledge as described in Section 6.7 will be executed, generating a process recipe and the process setting data will be stored in the slots of each child object. When the looping of all process objects is completed, the "SaveProdCharProcessPlan" function will be executed. The user will be prompted to specify the drive and input a file name to save the product requirements, the sequenced process name, the process recipe and the operation instruction of the required circuit board. Then a window will pop up and the process plan data file will be displayed for the user to view the product requirements (Figure 7.14), the approved process name, the suggested process sequence, the out of specification message (if any), the suggested processes that cannot be performed in-house (if any) (Figure 7.15) and the generated process recipe (Figure 7.16).

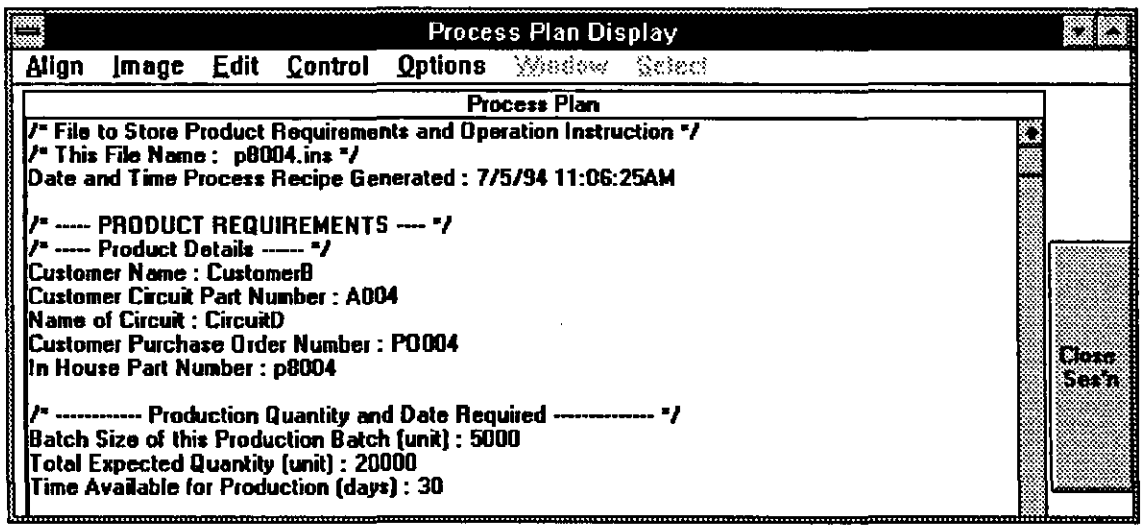


Figure 7.14 Process Plan Display Window Showing the Product Requirements

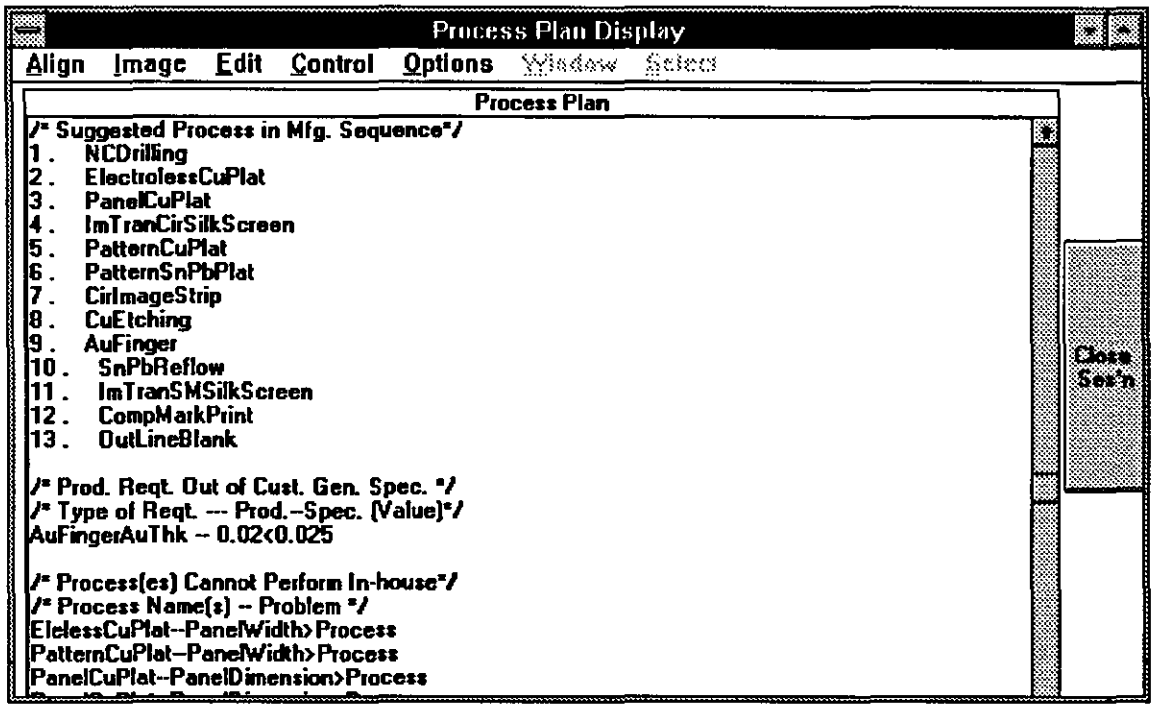


Figure 7.15 Process Plan Display Window Showing the Process Sequence and the Out Of Specification Message

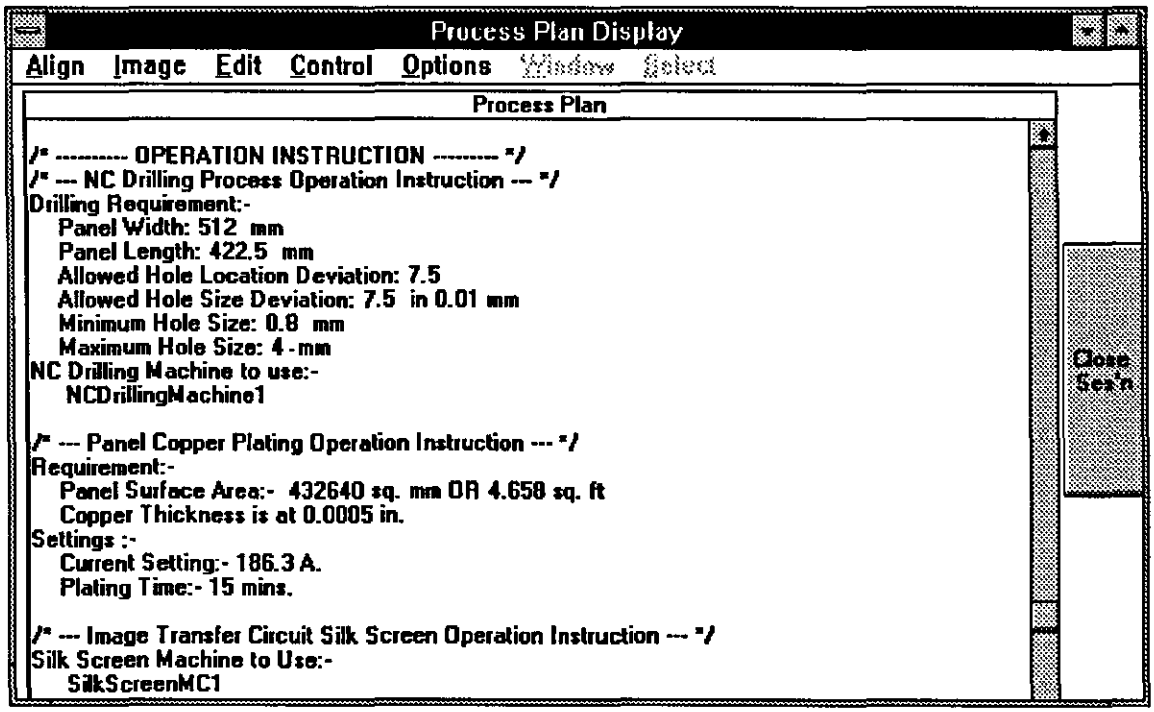


Figure 7.16 Process Plan Display Window Showing the Operation Instruction and the Process Recipe

After the user has viewed the process plan, buttons for the user to plan the next product or end the whole planning session (Figure 7.17) will appear. If the "Plan Next Product" button ("Button16") is selected, the existing product data objects will be deleted from the system. New product data objects will be loaded by prompting the user to input the file name of the next product and the next process planning session will begin. If the "No More Planing" button ("Button26") is clicked, the customer specification objects and product requirements objects will be deleted from the system. Then, the three interface buttons ("Button22", "Button14" and "Button17") and the "Load Data" button ("Button23") will appear.

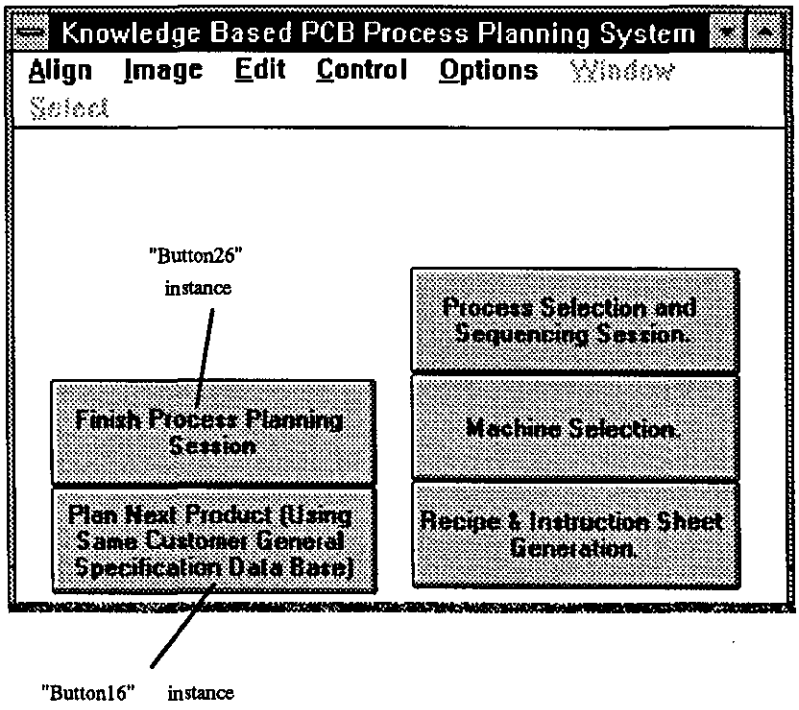


Figure 7.17 Options for "Planning Next Product" and "Finish Planning Session"

7.5 Summary

The object model and the dynamic model of the three sub-sessions of the system:- process selection and sequencing session, the machine selection session and the process recipe generation session were presented. The Kappa object oriented, knowledge-based software implementation of each session based on the models was illustrated. The successful implementation and operation of each session were supported by system graphical output.

CHAPTER 8
EVALUATION OF THE DEVELOPED KB PCB CAPP SYSTEM

8.1 Introduction

This chapter is concerned with the assessment of the system's ability to perform circuit board planning tasks. The method adopted for the evaluation of the system is described first. Then, the concerned in-house process capability, machine characteristics, customer general specifications and product requirements used in the evaluation are introduced. The results of the evaluation are presented and discussed.

8.2 Description of the Evaluation

An experiment was conducted to test the performance of the developed knowledge-based PCB CAPP system. Four people, including two experienced PCB process planners and two people without any PCB manufacture experience, were invited to be subjects. The planning results produced by the two experienced PCB planners were compared with that obtained by the other two people who using the KB PCB CAPP system. The dynamic diagram of the testing procedure is shown in Figure 8.1.

The PCB facilities in the PCB laboratory in the Department of Manufacturing Engineering at the City Polytechnic of Hong Kong were first analysed by the author. The information on facilities capabilities was given to the experienced PCB process planners. The same information was also input into the CAPP system by the author using the interface module.

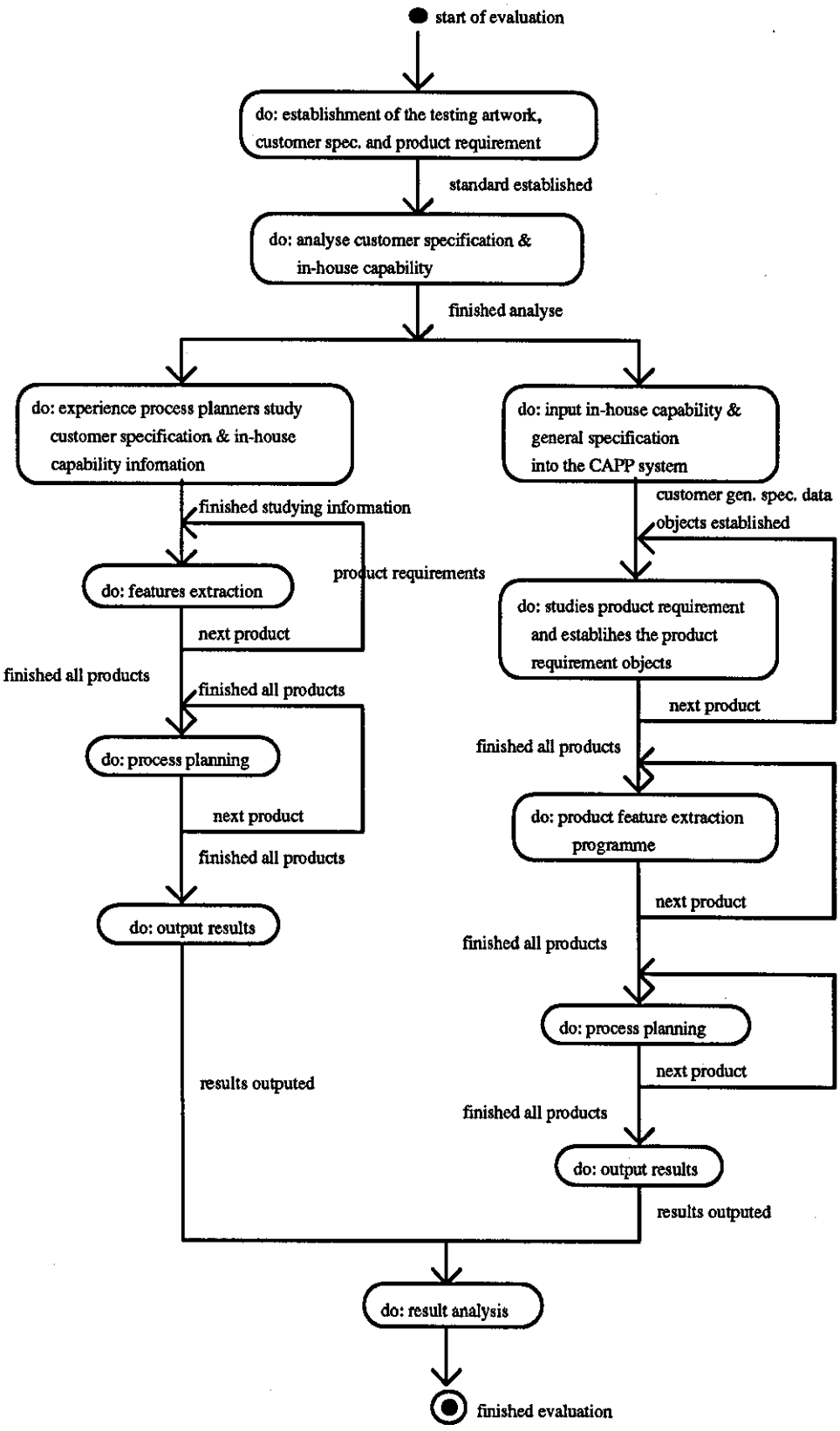


Figure 8.1 Dynamic Diagram of the Testing Procedure

General customer specifications of the two selected companies were presented to the two experienced PCB planners. The same specifications were also analysed by the author separately and then input into the system using the customer specifications interface module.

Five circuit boards:- circuitA, circuitB, circuitC circuitD and circuitE were created or selected as sample boards for the evaluation. The requirements of the first four circuit boards and their artworks were presented to the two process planners, who then completed a questionnaire (Figure 8.2) about their findings and the process plan of the circuits. The time they used to complete the planning of each circuit board was recorded. circuitE is used for the evaluation of the "checking customer general specification" ability of the experienced planners and the computer planning system. So no artwork is created.

The other two people were only provided with the product requirements and the artwork (in Gerber file or in film). After a brief training on the system operation, they were asked to input the product requirements into the system using the product requirements interface module, and do the process planning on the KB PCB CAPP system.

8.2.1 In-house Process Capability and In-house Machine Capability

22 object classes are used to represent the in-house capability facts. The attributes of each object have been described in Chapter 5. The KAL representation of the in-house capability objects is at Appendix 1.1. Figure 8.3 is a summary of the in-house process capability as presented to the experienced process planners. Figure 8.4 is the object representation of the in-house machine capability. A summary of the in-house machine capability representation is given in Figure 8.5.

Printed Circuit Board Process Planning Questionnaire

Information Provided: Circuit Requirement, Circuit Artwork (or Gerber Format File), Customer General Specifications, In-house Process Capability and In-house Machine Capability

Please Provide the Following Information:

Circuit Name:
Time to Start:
Time to Finish:

Section 1: Circuit Feature
1.1 Identify the unit circuit area.
1.2 Identify the number of location(s) with spacing smaller than 0.004 in.
1.3 Identify the number of location(s) with spacing smaller than 0.006 in.
1.4 Identify the number of location(s) with spacing smaller than 0.008 in.
1.5 Identify the number of solder pad location(s) with clearance smaller than 0.004 in.
1.6 Identify the number of solder pad location(s) with clearance smaller than 0.006 in.
1.7 Identify the number of solder pad location(s) with clearance smaller than 0.008 in.

Section 2: Out of Specifications
2.1 Are there any circuit requirements not meeting customer's general specifications?

Section 3: Process and Manufacture Sequence
3.1 Identify the process required in order to manufacture the required circuit board.
3.2 Arrange the process in the appropriate manufacture sequence.
3.3 Is/are there any suggested process(s) which is/are not available in-house.
3.4 Is/are there any suggested process(s) which is/are out of in-house process capability.

Section 4: Machine Selection
4.1 Suggest machine for process(s) with more than one machine in it.
4.2 Are there any process(s) without any suitable machine to manufacture the board?

Section 5: Process Recipe
5.1 Recommend the panel plating area in sq. ft.
5.2 Recommend the panel plating current in ampere and the plating time in minutes.
5.3 Recommend the pattern plating area of the panel in sq. ft.
5.4 Recommend the pattern copper copper plating current in ampere and plating time in minutes.
5.5 Recommend the metal finishing pattern plating current in ampere.
5.6 Recommend the metal finishing pattern plating time in minutes.
5.7 Recommend the gold finger plating area in sq. ft.
5.8 Recommend the gold finger nickel plating current and plating time.
5.9 Recommend the gold finger gold plating current and plating time.

Figure 8.2 Questionnaire for the Experienced Process Planner to Fill In

NC Drilling Machine Capability

InNCDrill:MaxRPM = 70000;
InNCDrill:NCWidth = 406;
InNCDrill:NCLength = 355;
InNCDrill:MachinedAcc = 0.05;

Electroless Copper Plating Capability

InElelessCu:Width = 406;
InElelessCu:Length = 508;

Copper Pattern Plating Capability

InCu:Width = 406;
InCu:Length = 508;
InCu:MaxCur =50;
InCu:CurrentDensity = 40;
InCu:TimeToDeposit = 3;

Tin/Lead Pattern Plating Capability

InSnPb:Width = 406;
InSnPb:Length = 508;
InSnPb:MaxCur = 50;
InSnPb:CurrentDensity = 20;
InSnPb:TimeToDeposit = 2.5;

Dry Film Image Transfer Capability

InDryFilm:Width = 508;
InDryFilm:Length = 508;
InDryFilm:Resolution = 0.05;
InDryFilm:FirstLeadTime = 0.5;
InDryFilm:CostPerSide = 2;
InDryFilm:ProdLeadTime = 0.08;

Silk Screen Image Transfer Capability

InSilkScreen:Width = 508;
InSilkScreen:Length = 558;
InSilkScreen:Resolution = 0.15;
InSilkScreen:FirstLeadTime = 2;
InSilkScreen:CostPerSide = 1;
InSilkScreen:ProdLeadTime = 0.5;

Solder Levelling Capability

InSnPbLevelling:Width = 304;
InSnPbLevelling:Length = 500;

Tin/Lead Reflow Capability

InReflow:Width = 355;

Panel Copper Plating Capability

InPanelPlat:Width = 406;
InPanelPlat:Length = 508;
InPanelPlat:MaxCur = 100;
InPanelPlat:CurrentDensity = 40;
InPanelPlat:TimeToDeposit = 3;

Copper Etching Capability

InCuEtching:UnCutRatio = 5;
InCuEtching:LineWidth = 0.1;

Gold Pattern Plating Capability

InAu:Width = 406;
InAu:Length = 508;
InAu:MaxCur = 20;
InAu:CurrentDensity = 5;
InAu:TimeToDeposit = 38;

Nickel Pattern Plating Capability

InNi:Width = 406;
InNi:Length = 508;
InNi:MaxCur = 50;
InNi:CurrentDensity = 30;
InNi:TimeToDeposit = 3.86;

Wet Film Image Transfer Capability

InWetFilm:Width = 508;
InWetFilm:Length = 508;
InWetFilm:Resolution = 0.1;
InWetFilm:FirstLeadTime = 1;
InWetFilm:CostPerSide = 1.5;
InWetFilm:ProdLeadTime = 0.3;

Gold Finger Plating Capability

InAuFing:Height = 50;
InAuFing:MaxCur = 20;

Blanking Machine Capability

InBlank:Width = 600;
InBlank:Length = 600;
InBlank:Accuracy = 0.15

NC Routing Capability

InNCRout:NCWidth = 406;
InNCRout:NCLength = 355;
InNCRout:MachinedAcc = 0.075;

Figure 8.3 The In-house Process Capability as Presented to the Experience Planner

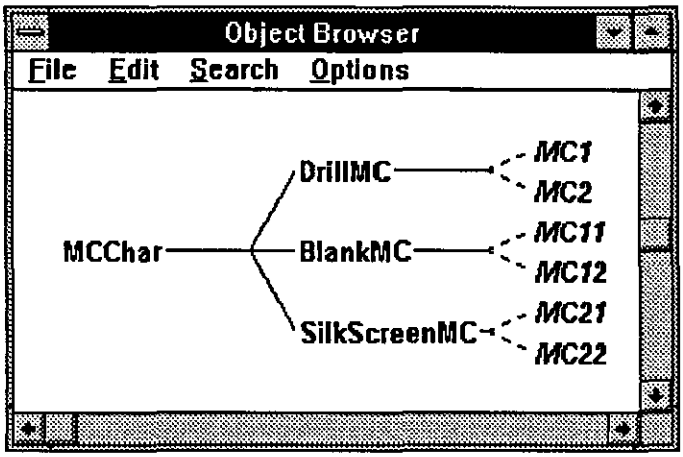


Figure 8.4 Object Classes and Hierarchy of the In-house Machine Capability

CNC Drilling Machine (MC1 and MC2)

MC1:NCWidth = 550; MC1:NCLength = 600; MC1:MachinedAcc = 0.1; MC1:MachineName = NCDrillingMachine1; MC1:MinHoleSize = 0.4; MC1:MaxHoleSize = 5;	MC2:NCWidth = 350; MC2:NCLength = 450; MC2:MachinedAcc = 0.125; MC1:MachineName = NCDrillingMachine2; MC2:MinHoleSize = 0.5; MC2:MaxHoleSize = 5.5;

Blanking Machine (MC11 and MC12)

MC11:Width = 250; MC11:Length = 250; MC11:Accuracy = 0.15; MC11:MachineName = BlankingMC1;	MC12:Width = 600; MC12:Length = 600; MC12:Accuracy = 0.15; MC12:MachineName = BlankingMC2;

Silk Screen Machine (MC21 and MC22)

MC21:Width = 600; MC21:Length = 600; MC21:Resolution = 0.2; MC21:MachineName = SilkScreenMC1;	MC22:Width = 300; MC22:Length = 300; MC22:Resolution = 0.19; MC22:MachineName = SilkScreenMC2;

Figure 8.5 The In-house Machine Capability as Presented to the Experience Planner

8.2.2 Customer General Specifications

Two customer general specifications, CustomerA and CustomerB, were selected. These were taken from industry. The object representation of the two customer general specifications are shown in Figure 8.6. The two customer general specifications were analysed by the author independently and input into the system using the interface module. It took the author about one hour to analyses one specification and input the data into the system using the interface module. The full KAL representation of the two customer specifications are at Appendix 2.1 and 2.2 respectively.

General Specifications of CustomerA	General Specifications of CustomerB
SpecInfo:CustName = CustomerA; SpecInfo:SpecNo = -; SpecInfo:RevDate = 11Jun84; SpecInfo:UpDatedBy = HangWai; CustLaminate:TypeOfLaminate1 = FR-4,1.6,1/1; CustLaminate:TypeOfLaminate2 = FR-4,1.6,1/0; CustLaminate:TypeOfLaminate3 = CAM3,1.6,1/1; CustCirStd:MinLnWid = 20 (in 0.01mm); CustCirStd:MinLnSpac = 15 (in 0.01mm); CustCirStd:Dev = 25 (%); CustCirStd:MinAnnRing = 15 (in 0.01mm); CustOutFin:DevDimen = 25 (in 0.01mm); CustOutFin:SurFinReq = Average; CustHoleStd:DevLoc = 13 (in 0.01mm); CustHoleStd:DevDia = 5 (+/- in 0.01mm); CustAuFing:MinNiThick = 0.3 (in 0.001in.); CustAuFing:MinAuThick = 0.045 (in 0.001in.); CustSMMatl:SMMaterial = PC501; CustPlatCu:MinThick = 1 (in 0.001in.); CustPlatCu:MaxThick = 3 (in 0.001in.); CustSMStd:MinThick = 0.8 (in 0.01mm); CustSMStd:PadClear = 10 (in 0.01mm); CustEtchStd:UCutRatio = 5; CustEtchStd:DevOrig = 25 (in %); SnPb:Min = 0.3 (in 0.001in); Ni:Min = 0.2 (in 0.001in); Au:Min = 0.02 (in 0.001in);	SpecInfo:CustName = CustomerB; SpecInfo:SpecNo = "93-2653"; SpecInfo:RevDate = 25Sept84; SpecInfo:UpDatedBy = "Hang Wai"; CustLaminate:TypeOfLaminate1 = FR-4,1.6,1/1; CustLaminate:TypeOfLaminate2 = FR-4,1.6,0.5/0.5; CustLaminate:TypeOfLaminate3 = FR-4,1.6,1/0; CustCirStd:MinLnWid = 12.5 (in 0.01mm); CustCirStd:MinLnSpac = 12.5 (in 0.01mm); CustCirStd:Dev = 15 (%); CustCirStd:MinAnnRing = 5 (in 0.01mm); CustOutFin:DevDimen = 20 (in 0.01mm); CustOutFin:SurFinReq = Normal; CustHoleStd:DevLoc = 7.5 (in 0.01mm); CustHoleStd:DevDia = 7.5 (+/- in 0.01mm); CustAuFing:MinNiThick = 0.15 (in 0.001in.); CustAuFing:MinAuThick = 0.025 (in 0.001in.); CustSMMatl:SMMaterial = PC501; CustPlatCu:MinThick = 1 (in 0.001in.); CustPlatCu:MaxThick = 3 (in 0.001in.); CustSMStd:MinThick = 1.78 (in 0.01mm); CustSMStd:PadClear = 5 (in 0.01mm.); CustEtchStd:UCutRatio = 2.5; CustEtchStd:DevOrig = 20 (in %); SnPb:Min = 0.3 (in 0.001in.); Ni:Min = 0.2 (in 0.001in.); Au:Min = 0.02 (in 0.001in.);

Figure 8.6 The Object Representation of the Two Customer General Specifications

8.2.3 Product Requirements

The specifications and requirements of circuitA are:

Project Documents Details:

Customer Name:	CustomerA
Customer Part Number:	A001
Name of Circuit:	CircuitA
Purchase Order No.:	PO001
In-House Part No.:	p8001

Required Quantity & Delivery Date:

Batch Size:	5,000
Total quantity:	20,000
Date available (in days):	30

Product Documents:

Drawing Number:	DN001
Drill File Number:	-
Film Number:	PCB8
Customer Spec. No:	1001

Laminate Details:

Laminate Material:	FR-4
Laminate Thickness:	1.6 mm
Copper Foil Thickness:	1 oz
Side of Copper Foil:	Single
Colour:	Green

Panel and Unit Dimension Details:

Unit Width (mm):	150
Unit Length (mm):	124
Panel Width (mm):	330
Panel Length (mm):	278
No. of Unit Per Panel	4
Outline Finishing Acc.:	0.15 mm
Est. Die Cost:	5,000

Hole Specifications:

Hole Diameter Deviation:	0.075 mm
Hole Location Deviation:	0.075 mm
Max. Hole Diameter:	2.5 mm
Min. Hole Diameter:	0.8 mm

Circuit Characteristic:

Min. Cir. Annular Ring:	0.25 mm
Total Unit Circuit Area:	1.32 sq. in
Min. Circuit Thickness:	0.04 mm
Min. Circuit Width:	0.7 mm
Min. Circuit Spacing:	>3.5 mm

Board Plating Requirements:

Min. Plat. Cu Thickness:	0
Type of Surface Finish:	SnPb
Min. Plated Metal Thick:	0.0005 in

Gold Finger Details:

Gold Fing. Gold Thick.:	0
Gold Fing. Height:	0
Gold Fing. Ni Thick.:	0
Fing. Area Per Unit:	0
No. of Unit Per Edge:	0

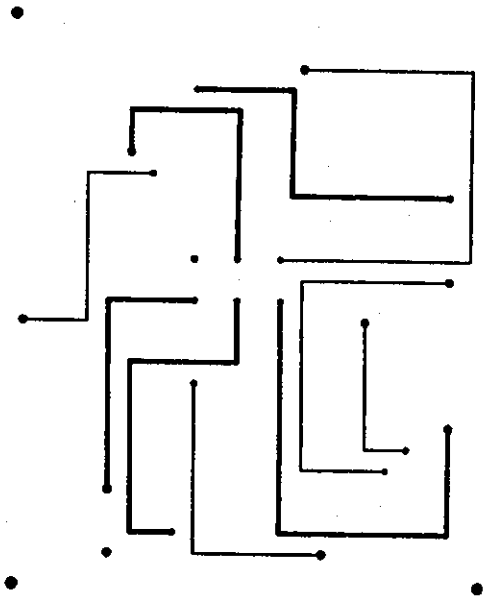
Solder Masking Details:

Solder Mask Material:	PC501
Solder Mask Colour:	green
Solder Mask Thickness:	0.01 mm
Solder Pad Clearance:	0.2 mm

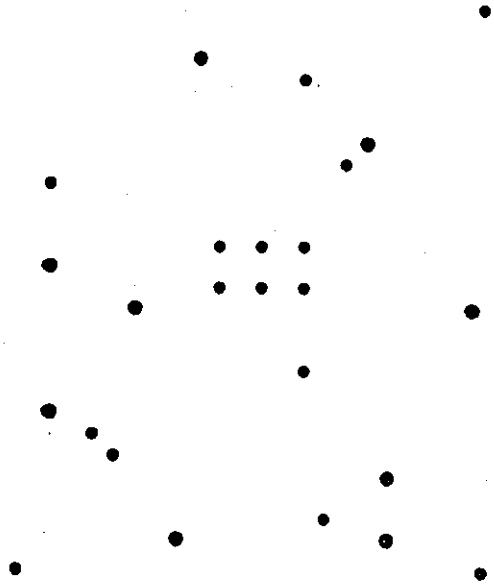
Component Marking Details:

Marking Material:	materialA
Marking Colour:	white

The Circuit Image of CircuitA:

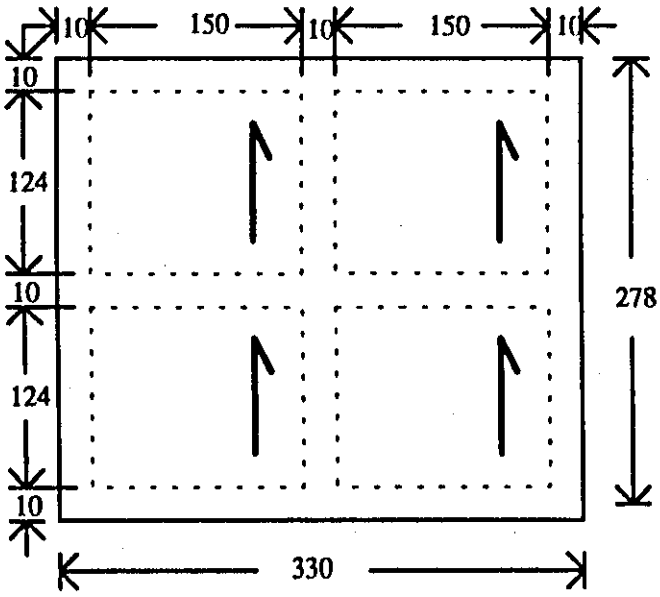


The Solder Masking Image of CircuitA:



(Not to scale)

The Panel Layout of CircuitA:



(Not to Scale, All Dimensions in mm)

The final and correct KAL representation of CircuitA requirements is shown at Appendix 3.1.

The specifications and requirements of circuitB are:

Project Documents Details:

Customer Name:	CustomerA
Customer Part Number:	A002
Name of Circuit:	CircuitB
Purchase Order No.:	PO002
In-House Part No.:	p8002

Required Quantity & Delivery Date:

Batch Size:	5,000
Total quantity:	20,000
Date available (in days):	30

Product Documents:

Drawing Number:	DN002
Drill File Number:	-
Film Number:	PCB
Customer Spec. No:	1001

Laminate Details:

Laminate Material:	FR-4
Laminate Thickness:	1.6 mm
Copper Foil Thickness:	1 oz
Side of Copper Foil:	Double
Colour:	Green

Panel and Unit Dimension Details:

Unit Width (mm):	106
Unit Length (mm):	100
Panel Width (mm):	242
Panel Length (mm):	230
No. of Unit Per Panel	4
Outline Finishing Acc.:	0.15 mm
Est. Die Cost:	9,000

Hole Specifications:

Hole Diameter Deviation:	0.075 mm
Hole Location Deviation:	0.075 mm
Max. Hole Diameter:	2.5 mm
Min. Hole Diameter:	0.8 mm

Circuit Characteristic:

Min. Cir. Annular Ring:	0.25 mm
Total Unit Circuit Area:	3.24 sq. in
Min. Circuit Thickness:	0.04 mm
Min. Circuit Width:	0.5 mm
Min. Circuit Spacing:	0.5 mm

Board Plating Requirements:

Min. Plat. Cu Thickness:	0.0015 in
Type of Surface Finish:	SnPb
Min. Plated Metal Thick:	0.0005 in

Gold Finger Details:

Gold Fing. Gold Thick.:	0
Gold Fing. Height:	0
Gold Fing. Ni Thick.:	0
Fing. Area Per Unit:	0
No. of Unit Per Edge:	0

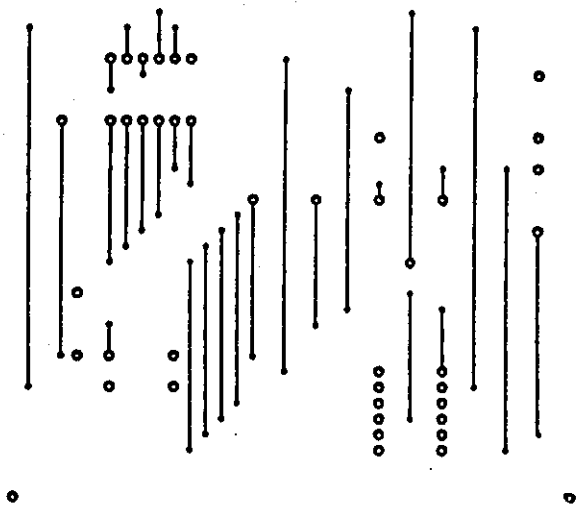
Solder Masking Details:

Solder Mask Material:	PC501
Solder Mask Colour:	green
Solder Mask Thickness:	0.01 mm
Solder Pad Clearance:	0.5 mm

Component Marking Details:

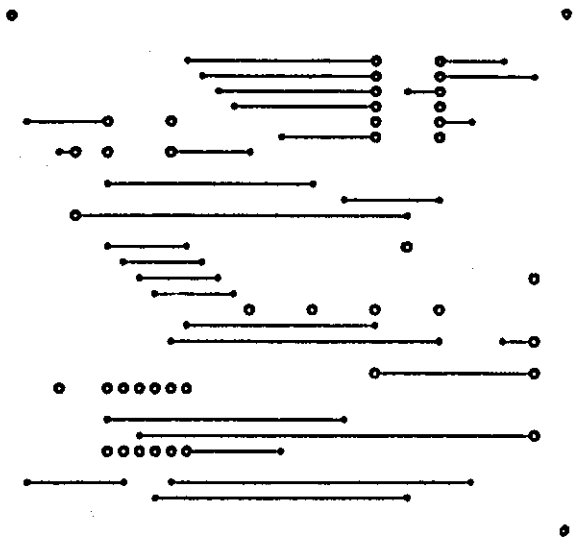
Marking Material:	materialA
Marking Colour:	white

The Component Side Circuit Image of CircuitB :



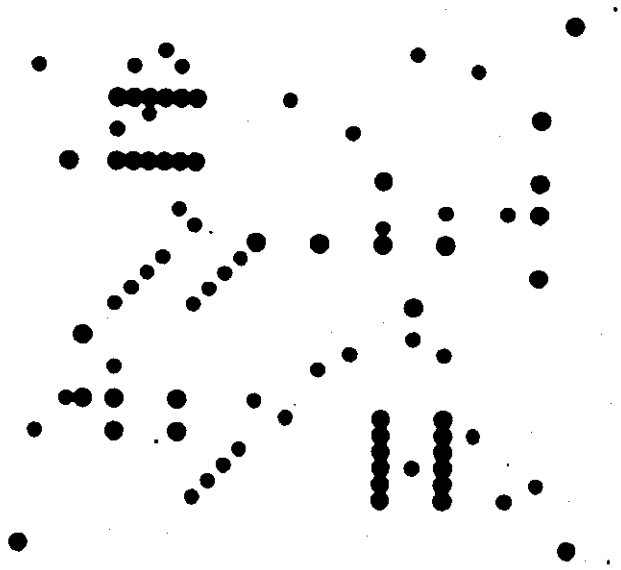
(Not to Scale)

The Solder Side Circuit Image of CircuitB :



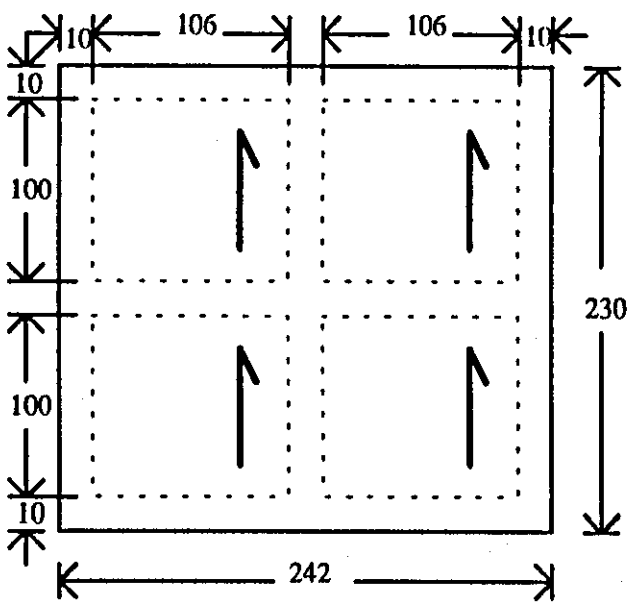
(Not to Scale)

The Solder Masking Image of CircuitB:



(Not To Scale)

The Panel Layout of CircuitB:



(Not to Scale, All Dimensions in mm)

The final and correct KAL representation of CircuitB requirements is shown at Appendix 3.2.

The specifications and requirements of circuitC are:

Project Documents Details:

Customer Name:	CustomerB
Customer Part Number:	B001
Name of Circuit:	CircuitC
Purchase Order No.:	PO008
In-House Part No.:	p8003

Required Quantity & Delivery Date:

Batch Size:	5,000
Total quantity:	20,000
Date available (in days):	30

Product Documents:

Drawing Number:	DN003
Drill File Number:	-
Film Number:	PCB8
Customer Spec. No:	1002

Laminate Details:

Laminate Material:	FR-4
Laminate Thickness:	1.6 mm
Copper Foil Thickness:	1 oz
Side of Copper Foil:	Double
Colour:	Green

Panel and Unit Dimension Details:

Unit Width (mm):	100
Unit Length (mm):	100
Panel Width (mm):	230
Panel Length (mm):	230
No. of Unit Per Panel	4
Outline Finishing Acc.:	0.15 mm
Est. Die Cost:	6,500

Hole Specifications:

Hole Diameter Deviation:	0.075 mm
Hole Location Deviation:	0.075 mm
Max. Hole Diameter:	2.5 mm
Min. Hole Diameter:	0.8 mm

Circuit Characteristic:

Min. Cir. Annular Ring:	0.25 mm
Total Unit Circuit Area:	3.36 sq. in
Min. Circuit Thickness:	0.04 mm
Min. Circuit Width:	0.5 mm
Min. Circuit Spacing:	0.5 mm

Board Plating Requirements:

Min. Plat. Cu Thickness:	0.0015 in
Type of Surface Finish:	Au
Min. Plated Metal Thick:	0.00002 in

Gold Finger Details:

Gold Fing. Gold Thick.:	0
Gold Fing. Height:	0
Gold Fing. Ni Thick.:	0
Fing. Area Per Unit:	0
No. of Unit Per Edge:	0

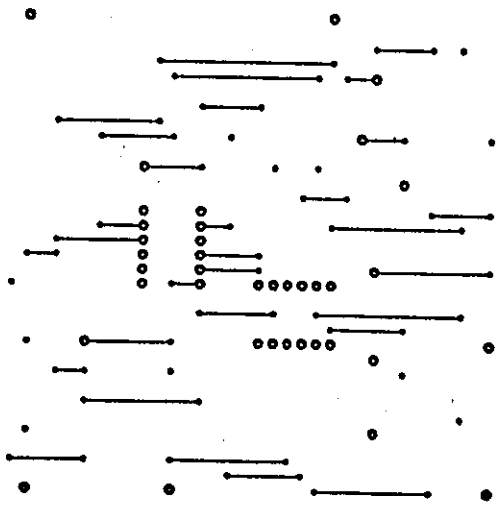
Solder Masking Details:

Solder Mask Material:	PC501
Solder Mask Colour:	green
Solder Mask Thickness:	0.01 mm
Solder Pad Clearance:	0.5 mm

Component Marking Details:

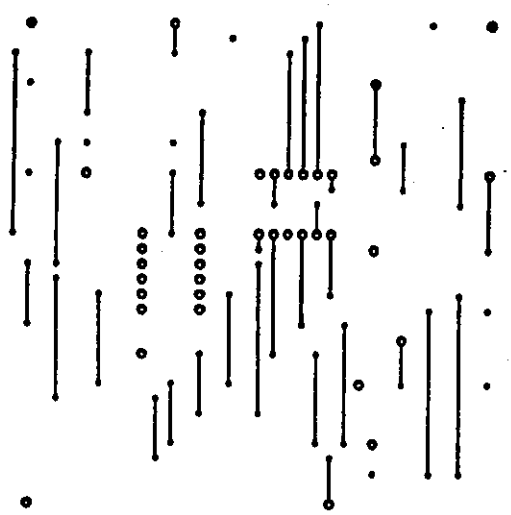
Marking Material:	materialB
Marking Colour:	yellow

The Component Side Circuit Image of CircuitC:



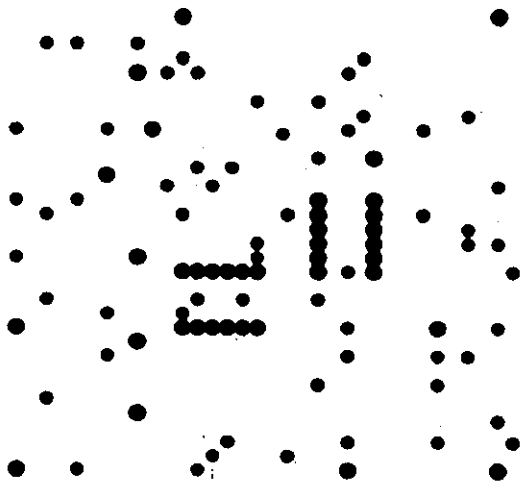
(Not to Scale)

The Solder Side Circuit Image of CircuitC:



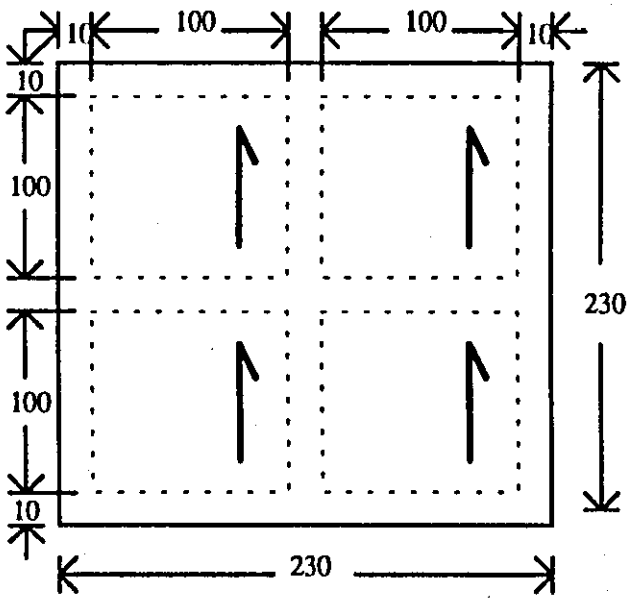
(Not to Scale)

The Solder Masking Image of CircuitC:



(Not To Scale)

The Panel Layout of CircuitC:



(Not to Scale, All Dimensions in mm)

The final and correct KAL representation of the CircuitC requirements is shown at Appendix 3.3.

The specifications and requirements of circuitD are:

Project Documents Details:

Customer Name:	CustomerB
Customer Part Number:	A004
Name of Circuit:	CircuitD
Purchase Order No.:	PO004
In-House Part No.:	p8004

Required Quantity & Delivery Date:

Batch Size:	5,000
Total quantity:	20,000
Date available (in days):	30

Product Documents:

Drawing Number:	DN004
Drill File Number:	-
Film Number:	PCB4
Customer Spec. No:	1002

Laminate Details:

Laminate Material:	FR-4
Laminate Thickness:	1.6 mm
Copper Foil Thickness:	1 oz
Side of Copper Foil:	Double
Colour:	Green

Panel and Unit Dimension Details:

Unit Width (mm):	241
Unit Length (mm):	127.5
Panel Width (mm):	512
Panel Length (mm):	422.5
No. of Unit Per Panel	6
Outline Finishing Acc.:	0.125 mm
Est. Die Cost:	8,000

Hole Specifications:

Hole Diameter Deviation:	0.075 mm
Hole Location Deviation:	0.075 mm
Max. Hole Diameter:	4.0 mm
Min. Hole Diameter:	0.8 mm

Circuit Characteristic:

Min. Cir. Annular Ring:	0.2 mm
Total Unit Circuit Area:	18.1 sq. in
Min. Circuit Thickness:	0.04 mm
Min. Circuit Width:	1.0 mm
Min. Circuit Spacing:	1.0 mm

Board Plating Requirements:

Min. Plat. Cu Thickness:	0.002 in
Type of Surface Finish:	SnPb
Min. Plated Metal Thick:	0.0003 in

Gold Finger Details:

Gold Fing. Gold Thick.:	0.00002 in
Gold Fing. Height:	13 mm
Gold Fing. Ni Thick.:	0.0002 in
Fing. Area Per Unit:	1.8 sq. in.
No. of Unit Per Edge:	3

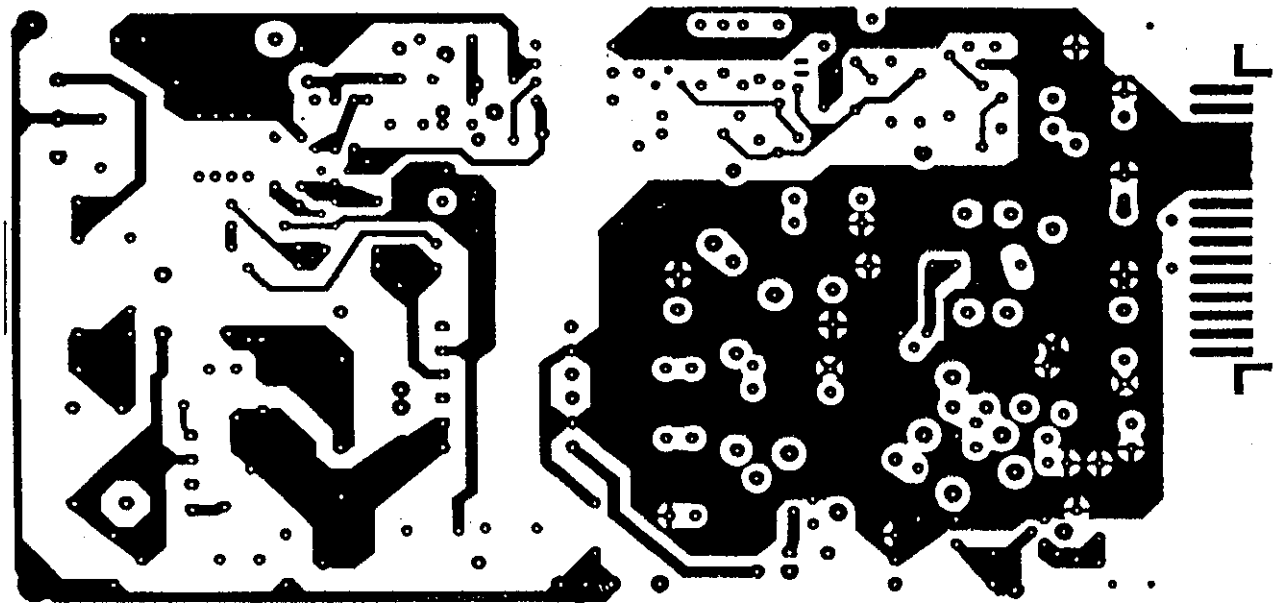
Solder Masking Details:

Solder Mask Material:	PC501
Solder Mask Colour:	Green
Solder Mask Thickness:	0.01 mm
Solder Pad Clearance:	0.5 mm

Component Marking Details:

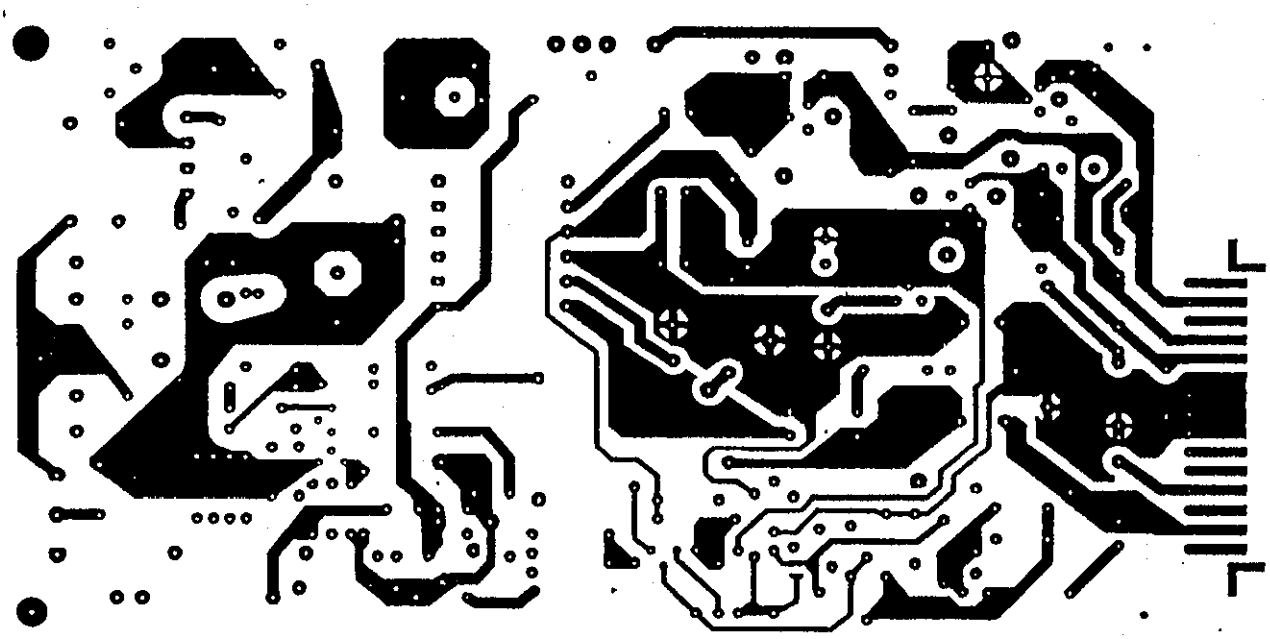
Marking Material:	materialA
Marking Colour:	white

The Component Side Circuit Image of CircuitD:



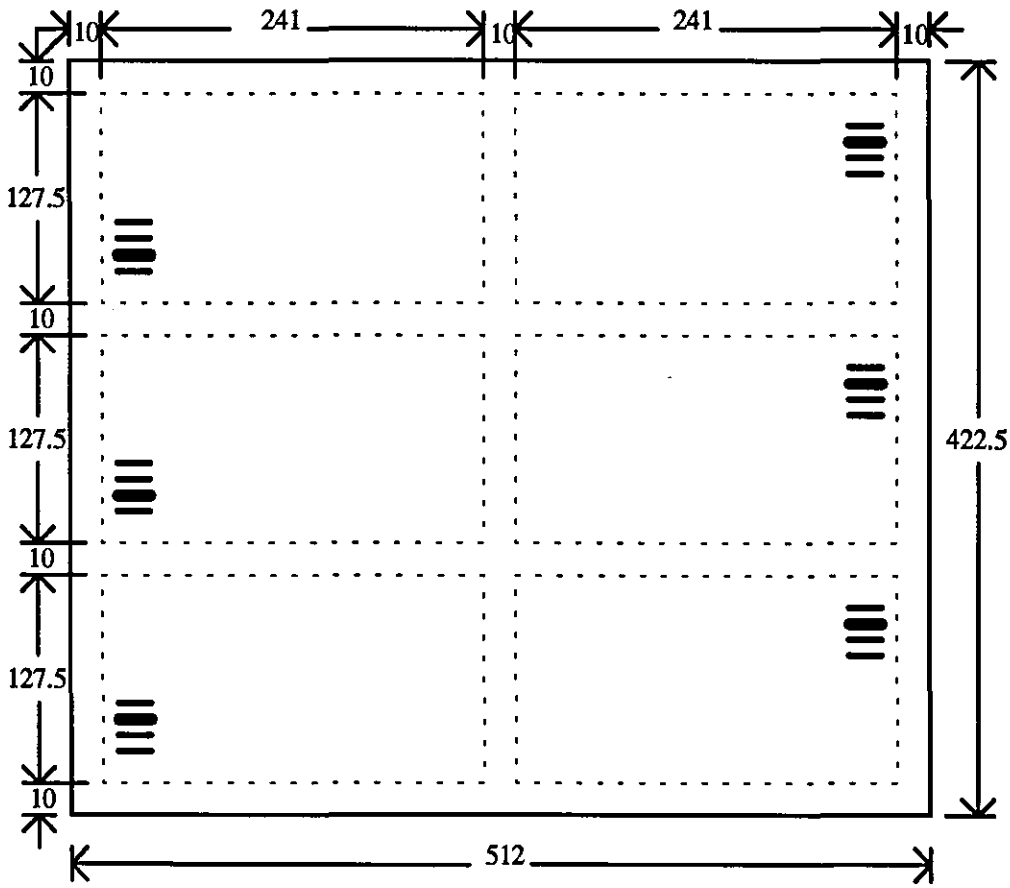
(Not to Scale)

The Solder Side Circuit Image of CircuitD:



(Not to Scale)

The Panel Layout of CircuitD:



Component Side Shown. (Not to Scale, All Dimensions in mm)

The final and correct KAL representation of the CircuitD requirements is shown at Appendix 3.4.

The specifications and requirements of circuitE are:

Project Documents Details:

Customer Name:	CustomerA
Customer Part Number:	A005
Name of Circuit:	CircuitE
Purchase Order No.:	PO005
In-House Part No.:	p8005

Required Quantity & Delivery Date:

Batch Size:	5,000
Total quantity:	20,000
Date available (in days):	30

Product Documents:

Drawing Number:	DN005
Drill File Number:	-
Film Number:	PCB10
Customer Spec. No:	1001

Laminate Details:

Laminate Material:	FR-4
Laminate Thickness:	1.6 mm
Copper Foil Thickness:	1 oz
Side of Copper Foil:	Double
Colour:	Green

Panel and Unit Dimension Details:

Unit Width (mm):	300
Unit Length (mm):	300
Panel Width (mm):	630
Panel Length (mm):	630
No. of Unit Per Panel	4
Outline Finishing Acc.:	0.10 mm
Est. Die Cost:	10,000

Hole Specifications:

Hole Diameter Deviation:	0.12 mm
Hole Location Deviation:	0.05 mm
Max. Hole Diameter:	5 mm
Min. Hole Diameter:	0.9 mm

Circuit Characteristic:

Min. Cir. Annular Ring:	0.02 mm
Total Unit Circuit Area:	6.2 sq. in
Min. Circuit Thickness:	0.04 mm
Min. Circuit Width:	0.1 mm
Min. Circuit Spacing:	0.1 mm

Board Plating Requirements:

Min. Plat. Cu Thickness:	0.002 in
Type of Surface Finish:	BareCopper
Min. Plated Metal Thick:	0.0003 in

Gold Finger Details:

Gold Fing. Gold Thick.:	0
Gold Fing. Ni Thick.:	0
Gold Fing. Height:	0
Fing. Area Per Unit:	0
No. of Unit Per Edge:	0

Solder Masking Details:

Solder Mask Material:	PC401
Solder Mask Colour:	green
Solder Mask Thickness:	0.01 mm
Solder Pad Clearance:	0.04 mm

Component Marking Details:

Marking Material:	materialB
Marking Colour:	yellow

The final and correct KAL representation of the CircuitE requirements is shown at Appendix 3.5.

8.3 Results of the Evaluation and Discussion

The detailed KB process planning results of CircuitA, CircuitB, CircuitC, CircuitD and CircuitE are at Appendices 10.1, 10.2, 10.3, 10.4 and 10.5 respectively.

8.3.1 Circuit Features

The circuit features of the evaluation are as follows:

CircuitA	Planner A	Planner B	Ext. Prog.
Total Unit Plating Area (sq. in.)	1.4	1.44	1.32
Min. Circuit Spacing (mm)	>3	>3	0.35
Time to Find Min. Spacing (minute)	5	6	3
Min. Sold. Pad Clear. (mm)	0.2	0.2	0.2
Time to Find Min. Clear (minute)	3	3	2

CircuitB	Planner A	Planner B	Ext. Prog.
Total Unit Plating Area (sq. in.)	3.2	5.2	3.24
Min. Circuit Spacing (mm)	3.6	3.6	3.5
Time to Find Min. Spacing (minute)	4	5	3
Min. Sold. Pad Clear. (mm)	0.5	0.5	0.5
Time to Find Min. Clear (minute)	2	3	2

CircuitC	Planner A	Planner B	Ext. Prog.
Total Unit Plating Area (sq. in.)	3.1	3.8	3.36
Min. Circuit Spacing (mm)	0.5	0.5	0.5
Time to Find Min. Spacing (minute)	4	4	3
Min. Sold. Pad Clear. (mm)	0.5	0.5	0.5
Time to Find Min. Clear. (minute)	2	2	2

	Experience Planner		Common People	
CircuitD (all done manually)	A	B	A	B
Total Unit Plating Area (sq. in.)	18	18	18	19
Min. Circuit Spacing (mm)	1	1	1	1
Time to Find Min. Spacing (minute)	4	5	16	18

The results of the feature extraction program, particularly in terms of accuracy, are better than that obtained from the experienced process planners. In the calculation of the unit circuit area, the two experienced process planners' estimations showed deviation. For example in circuitB, planner A's estimation was 3.2 while planner B's estimation was 5.2. In finding of the minimum spacing, the experienced planners took longer time than the feature extract program and showed a lack of confidence in their decisions. After the evaluation was finished, all planners agreed that the method used by the feature extraction program was more reliable. In finding the minimum solder masking pad clearance, the performance by the manual planners was also not as good as that by the feature extraction program. The manual process planners always required longer time than the program did. However, the feature extraction program also had its own limitations. The artwork should be in Gerber format and the existing system could only process relatively small circuits because the program is PC based. Also, inexperienced people took longer time than experience process planners in finding the circuit spacing.

8.3.2 Product Requirements Checking Against Customer General Specifications

Some of the requirements of circuitD and circuitE are out of their respective customer general specifications. The out of specification message file of circuitD and circuitE file generated by the planning system were as follows:

The Print Out of CircuitD Out of Specification Message File:

```
/* Prod. Reqt. Out of Cust. Gen. Spec. */
/* Type of Reqt. --- Prod.--Spec. (Value)*/
AuFingerAuThk -- 0.02<0.025
```

The Print Out of CircuitE Out of Specification Message File:

```
/* Prod. Reqt. Out of Cust. Gen. Spec. */
/* Type of Reqt. --- Prod.--Spec. (Value)*/
SolderPadClear -- 4<10
SolderMaskMatl -- PC401Not accepted
CircuitWidth -- 10<20
CircuitSpacing -- 10<15
CirAnnularRing -- 2<15
```

It shows that the planning system successfully found all the product requirements which were out of the customer general specifications.

However, the two experience process planners could not identify gold finger gold thickness requirement of circuitD (0.000020 in.) is out of customer general gold finger thickness specification (0.000025 in). Also, for circuitE, some of the product requirements violating the customer general specifications could not be identified by the two experienced process planners. The findings of the two experience planner were:

CircuitE	Experienced Planner	
Type of Requirement -- Product Req., General Spec.	A	B
Solder Pad Clearance -- 0.04 mm < 0.1 mm	found	found
Solder Mask Material -- PC401 Not accepted	not found	found
Circuit Width -- 0.1 mm < 0.2 mm	found	not found
Circuit Spacing -- 0.1 mm < 0.15 mm	found	not found
Circuit Annular Ring -- 0.02 mm < 0.15 mm	found	found

For example, planner A could not identify PC401 which was not the approved solder mask material to be used and planner B could not identify circuit width and circuit spacing out of the customer general specification. Besides, it always took the experienced planners a lot of time in referring back to the customer specification documentation before they had confidence in their decisions. This indicates that it is very difficult for the process planner

to remember all the values of the customer general specifications and consider them thoroughly when making their decisions.

8.3.3 Process Selection and Sequencing

The test results indicate that the process and the manufacture sequence recommended by the experienced planners were agreeable with those suggested by the planning system. With some basic system operation training and under the assistance of the KB PCB CAPP system, people without experience in circuit board manufacture could perform process selection and manufacture sequencing as well as the experienced circuit board process planners. However, the performance of the planning system depended on the user's understanding of the product requirements and the correctness in inputting the requirements into the system by using the interface module. There were several instances in which the users found it difficult to follow the input format of the product requirements interface module.

8.3.4 Machine Selection

The circuitA machine selection results generated by the planning system are as follows:

```
NC Drilling Machine to use in NC Drilling:-
   NCDrillingMachine1, NCDrillingMachine2
Silk Screen Machine to Use in Circuit Image Transfer:-
   SilkScreenMC1,   SilkScreenMC2
Silk Screen Machine to Use in Solder Masking Image Transfer:-
   SilkScreenMC1,   SilkScreenMC2
Silk Screen Machine to Use in Comp. Marking Image Transfer:-
   SilkScreenMC1,   SilkScreenMC2
Blanking Machine to Use in Blanking Operation:-
   BlankingMC1,   BlankingMC2
```

The machine selection results of circuitB and ciruitC are the same as circuitA.

The circuitD machine selection results generated by the planning system are as follows:

```
NC Drilling Machine to use in NC Drilling:-
   NCDrillingMachine1
Silk Screen Machine to Use in Circuit Image Transfer:-
   SilkScreenMC1
Silk Screen Machine to Use in Solder Masking Image Transfer:-
   SilkScreenMC1
Silk Screen Machine to Use in Comp. Marking Image Transfer:-
   SilkScreenMC1
Blanking Machine to Use in Blanking Operation:-
   BlankingMC2
```

The circuitE machine selection results generated by the planning system are as follows:

```
NC Drilling Machine to use in NC Drilling:-
*****NOSuitableNCDrillingMC
Silk Screen Machine to Use in Circuit Image Transfer:-
*****NOSuitableSilkScreenMC
Silk Screen Machine to Use in Solder Masking Image Transfer:-
*****NOSuitableSilkScreenMC
Silk Screen Machine to Use in Comp. Marking Image Transfer:-
*****NOSuitableSilkScreenMC
Blanking Machine to Use in Blanking Operation:-
*****NOSuitableBlankingMC
```

The results show that the planning system successfully identified machines for different processes for all five circuits. For circuitE, as all the selected processes are out of the in-house capability, no suitable machine can be identified.

In industry, machine selection is done by production personnel who base their decisions on their experience of using the machines. As each production people's experience about the machines is different, such method of machine selection is not always reliable. The results

of this evaluation show that even using an experienced process planer in machine selection, it is subject to error because a large amount of data needs to be considered. So, in this aspect, the developed process planning system performance is better than experienced process planners. Also, the machine selection procedure and machine capability are systematically represented and stored. As a result, the system performance is more reliable. Up-dating of the machine data in the system is also restricted to authorised process engineers; hence, the system data integrity is maintained.

8.3.5 Process Recipe

The plating area, plating current and plating time of CircuitA, CircuitB, CircuitC CircuitD and CircuitE are as shown:

	Exp. Planner		
CircuitA	A	B	CAPP Sysm.
Panel Plating Area (in sq. ft.)	*	*	*
Panel Plating Current (A)	*	*	*
Panel Plating Time (min.)	*	*	*
Pattern Plating Area (in sq. ft.)	0.165	0.132	0.149
Pattern Cu Plating Current (A)	*	*	*
Pattern Cu Plating Time (min.)	*	*	*
Finishing Plating Current (A)	3	2.5	2.98
Finishing Plating Time (min.)	15	13	12.5

* process not required, single sided board

	Exp. Planner		
CircuitB	A	B	CAPP Sysm.
Panel Plating Area (in sq. ft.)	1.2	1.2	1.2
Panel Plating Current (A)	48	46	48
Panel Plating Time (min.)	15	15	15
Pattern Plating Area (in sq. ft.)	0.24	0.186	0.262
Pattern Cu Plating Current (A)	9.6	6.882	10.46
Pattern Cu Plating Time (min.)	30	31	30
Finishing Plating Current (A)	4.8	3.5	5.23
Finishing Plating Time (min.)	12.5	13	12.5

CircuitC	Exp. Planner A	Exp. Planner B	CAPP Sysm.
Panel Plating Area (in sq. ft.)	1.1	1.1	1.14
Panel Plating Current (A)	44	44	45.6
Panel Plating Time (min.)	15	15	15
Pattern Plating Area (in sq. ft.)	0.24	0.166	0.26
Pattern Cu Plating Current (A)	9.6	6.14	10.4
Pattern Cu Plating Time (min.)	30	30	30
Finishing Plating Current (A)	1.2	0.8	1.3
Finishing Plating Time (min.)	7.6	8	7.6

CircuitD	Exp. Planner A	Exp. Planner B	CAPP Sysm.
Panel Plating Area (in sq. ft.)	4.7	4.7	4.658
Panel Plating Current (A)	188	188	186.3
Panel Plating Time (min.)	15	15	15
Pattern Plating Area (in sq. ft.)	1.23	2.2	1.167
Pattern Cu Plating Current (A)	49.2	81.4	46.68
Pattern Cu Plating Time (min.)	45	30	45
Finishing Plating Current (A)	24.6	40.9	23.34
Finishing Plating Time (min.)	7.5	8	7.5

CircuitE	Exp. Planner A	Exp. Planner B	CAPP Sysm.
Panel Plating Area (in sq. ft.)	8.5	8.5	8.55
Panel Plating Current (A)	340	340	341.8
Panel Plating Time (min.)	15	15	15
Pattern Plating Area (in sq. ft.)	0.73	0.76	0.649
Pattern Cu Plating Current (A)	29.2	28	25.96
Pattern Cu Plating Time (min.)	45	30	45
Finishing Plating Current (A)	14.6	18.6	12.98
Finishing Plating Time (min.)	7.5	8	7.5

From the test results, it is found that the two different experienced process planners always gave inconsistant process parameter settings; the deviation sometimes could be as high as 65% (pattern copper plating current of circuitD, 49.2 and 81.4). Such high deviation was mainly due to the deviation in their plating area estimations. In this aspect, the process settings suggested by the system were more consistant. In addition, the

accuracy of the system performance could be enhanced by improving the plating area estimation knowledge. The improved estimation knowledge could be implemented easily by modifying the area estimation function.

8.3.6 Planning Time

	Process Planner*			Using CAPP Sysm.#		
	A	B	**Ave.**	A	B	**Ave.**
CircuitA (in min.)	65	78	**71.5**	22	24	**23**
CircuitB (in min.)	58	67	**62.5**	21	22	**21.5**
CircuitC (in min.)	70	80	**75**	20	20	**20**
CircuitD (in min.)	53	79	**42**	18	20	**19**
CircuitE (in min.)	55	60	**57.5**	22	25	**23.5**
			61.7			**21.4**

Remarks:

* not including the time for planners to study the customer general specifications, but including the time for planners to fill in the questionnaire

\# including the time for the people to input the data into the product interface module and the computer program planning time

The average time required by the experienced process planners to plan each product manually was, 61.7 minutes. However, the average time required by the people using the PCB CAPP system to do the same plan was only 21.4 minutes. Furthermore, each process planner used about two hours to study the two customer general specifications and the in-house process capability data sheet before they started their planning. The people with no PCB manufacture experience saved this extra time but did the process planning job as competent as the experienced planners.

8.4 Overall Discussion

The evaluation results show that the feature extraction program provides a more systematic and reliable means for integrating the planning system with the electronic CAD data base. Pattern circuit area, circuit spacing and solder pad clearance are features that can be extracted directly from CAD data files. However, the existing feature extraction program has its limitations. For instance, the program is only applicable to small circuits and the artwork must be in Gerber format.

The system represents the in-house process capability data, the customer general specifications and the product requirements in an object oriented approach, giving a more structured representation which is subject to less bias during process planning. The information is accessible to all users. Both information up-dating and retrieval are more convenient because only one person is required for data acquisition and information input into the system. Hence, a better data integrity is achieved. The interpretation and inputting of the same piece of information by different process planners is not required.

Because the knowledge-based planner system always uses the same decision procedures and knowledge in the planning process, the system's performance is more consistent and free from the subjective inference of the human process planner. The knowledge and the decision logic can be up-dated and enhanced to further improve the system performance.

The evaluation also shows that the developed system can be successfully used in deskilling PCBs process planning activities. People with no circuit board manufacture experience can also use the system to do circuit board process planning as competently as the experienced circuit board process planners. In the aspect of circuit board feature

extraction, specification checking and machine selection, the performance of the system is even better than that of the experienced process planner.

The planning system also indicates that a structured approach would enhance system integration; the information within the system could be freely exchnaged among themselves because the developed KB PCB CAPP uses the standard data object to store information. When the KAL commands are properly modified, the information generated at different stages in the system can be accessible to other application programs such as production planning and costing.

8.5 Summary

The developed planning system has been evaluated in this chapter. Firstly, the evaluation method to be adopted was described with the help of dynamic diagrams. Secondly, the in-house process capability, machine characteristics, customer specifications and circuit requirements which the evaluation used were explained. Evaluation results were then presented. They indicated that compared with the performance of the experienced process planners, the developed circuit board computer aided process planning system had better functionalities, used shorter planning time and gave more consistent planning results. The data object and rule-based knowledge representation structure of the planning system also pointed to better data and knowledge integrity and more effective system performance.

CHAPTER 9
DISCUSSION AND CONCLUSION

9.1 Introduction

This chapter discusses the contributions of this research project and suggests areas for further research in object oriented knowledge-based printed circuit board process planning system.

9.2 Contributions of this Research Project

9.2.1 Pioneer CAPP Project in PCB Manufacture

This project is one of the pioneer research projects using knowledge-based computer system to do process planning in the context of bare printed circuit board manufacture. Evaluation results have shown that using a computer system to do PCB process planning has advantages over manual planning: shorter time in generating process plans, less dependence on the skill level of process planners, a better functionality system and higher quality process plans.

9.2.2 OMT Modelling Methodology in KB CAPP System

This research has demonstrated the pioneer effects of applying the OMT methodology to model a knowledge-based computer aided PCB process planning system. Very few knowledge-based computer aided process planning systems have been reported using such

approach. Most knowledge-based system developers usually adopt the ad hoc approach which ends up with an inflexible and unstructured system. It has been demonstrated in this research that using the OMT methodology could be a solution. The four development phases and the three models of the OMT methodology are followed in designing and implementing the planning system. Results has found that this methodology allows improvements of the planning system progressively and the system achieves its mature state gradually.

The object modelling provided by OMT methodology gives a clear picture on the object classes for application. The research has found the suitability of implementing the object model in an object-oriented shell or programming language; there is almost a direct mapping. For example, in this project, the object model of the system was directly implemented using the Kappa object-oriented knowledge-based software shell.

The standards of the OMT methodology also provide a systematic format for the representation of the process planning system. In addition, the message passing mechanism of an object-oriented system could effectively activate operations of all objects in the same class although the operations may take different forms in different classes. This has been illustrated fully during the check customer general specification, the process selection and the machine selection operations.

9.2.3 Systemic Structuring of the Knowledge in the System

The research has developed a solution to the problem of unstructured knowledge in most of the knowledge-based system. The process of building an object model gives system developers a means to structure the knowledge logically. The object model facilitates dividing the knowledge into logical groups and organising them in a hierarchical way.

Thus, a better understanding of the domain problem could be obtained. Such structure also provides a foundation for the knowledge engineer or the domain expert to update or review the knowledge contained in the system. Only one person is required for knowledge acquisition and up-dating, thus a better data integrity is achieved.

The research has also discovered that attributes of objects are a direct representation method for declarative type of PCB process planning knowledge. The in-house process capability knowledge, customer general specification facts and product requirements are successfully captured and represented using this method.

In addition, the hierarchical structure and inheritance property of an object-oriented schema provide an effective method to define knowledge structure for other similar processes. The defined attributes of a class can be inherited by lower classes in the hierarchy. When a new type of resource is added to the planning system, a new class can be easily created without redefining its structure. The inheritance property of an object-oriented schema has been found particularly useful in segregating the knowledge base according to the class of objects. The modularity of the knowledge has facilitated knowledge maintenance and helped to enhance the performance of the knowledge inference process.

The system also succeeds in formal capturing of human PCB planning knowledge other than declarative fact type which cannot be effectively represented by the object structure. Production rules are incorporated in such areas. The advantage of using production rules is its flexibility in modification whenever required. Also, the knowledge can be implemented easily using the rule representation function of the Kappa software shell. The inference mechanism of such type of knowledge can still be carried out under the operation structure of an object. Such an arrangement can improve the

performance of the inference process and facilitate the future system maintenance work. The results of the evaluation have indicated that the planning knowledge is effectively used by the knowledge-based computer aided process planning system in formulating the process plan.

The planning knowledge is also categorised into global knowledge (such as process selection rules and process sequencing facts) and local knowledge (such as in-house process capability facts and in-house machine capability facts). This will make the knowledge up-dating much easier. Another advantage of this arrangements is that it greatly improves the system portability. Only the local knowledge needs to be modified when the system is implemented in other factories.

9.2.4 Superiority of the KB PCB CAPP System

The research has found that the time required for in-experience planners to generate a process plan with the help of the KB PCB CAPP system is less. The demand on knowledge and the skill level of the process planner is reduced. This shows that the planning system successfully deskills the circuit board process planning activity. Training process planners on job is not longer required.

The functionality provided by the system is also much richer than that given manually by experienced circuit board process planners. For example, some circuit and solder mask features can be extracted directly from Gerber format data files. Moreover, the customer general specification checking and the machine selection function are implemented systematically by the planning system. As a result, both quality and stability of the process plan generated by the planning system is better than those produced manually by an experienced process planner.

The KB PCB CAPP system emphasizes engineering knowledge and reasoning procedures in the whole planning process. The engineering knowledge is stored in the system and the reasoning procedures are rigorously followed in the decision making process. Subjective inference of individual process planner is avoided. The system also allows the presence of both qualitative and quantitative approaches during the decision making whereas a human planner finds it difficulty to manage the two approaches at the some time. The system's capability to allow the presence of the dual approaches makes a robust planning system possible for process planning of PCB manufacture. The object structure means of storing information generated by the planning system also allows easy accessing by other application program (such as MRP production planning system and costing system) with minimal modification.

9.3 Suggestions for Future Work

A possible and logical extension of the research work is to integrate the CAPP system with a workstation type of PCB CAM software (such as "ECAM" [CAD Solutions, 1992]) so that artwork files of different formats can be transformed into the Gerber file format on which the feature extraction program is based.

Specification extraction knowledge should be added into the customer general specification interface module so that users need only to input the high level circuit board specification using American National Standard, Interconnections Packaging Circuitry Standard or British Standard. The interface module will then base on the selected specification standard and the system knowledge to find out the attribute values of the detail customer general specification objects.

The feature extraction program can also be up-graded so that it can be run on a workstation type of computer. As a result, more complex circuit images could be processed within reasonable time. The concept of partitioning in sub-dividing large artwork into smaller problem areas should be put to use so as to reduce the feature extraction problem size.

In addition, the functionality of the feature extraction program could be enhanced to enable more features be extracted automatically. For example, drilling feature and circuit feature of multi-layer circuit boards should be included. For production requirements data, an enhanced planning system is believed to extract data directly from file in the Electronic Data Interchange format.

The process planning knowledge could be further enhanced so that the planning system could be capable of handling multi-layer circuit boards. Rules used in the generation of plating current and plating time could also be enhanced to include the factor such as anode to cathode ratio and board circuit geometric feature.

An extensive amount of knowledge in the areas of economic (such as process and machine costing) and production capacity could be added to the CAPP system bringing the system more comprehensive.

9.4 Conclusion

This thesis has covered the development of a structured, computer aided printed circuit board process planning system with a capability for generating process plans for the manufacture of bare printed circuit boards. Such planning system is crucial in the successful operation of board manufacturers in today's highly competitive environment.

A knowledge-based framework for the development of a computer aided process planning was first identified. The Object Modelling Technique (OMT) was justified and adopted as the system modelling and the implementation methodology. C++ computer programming language and Kappa object oriented knowledge-based software shell were used to implement the planning system. The author's implementation experience further strengthened the justification of using OMT as the modelling and implementation methodology. Various planning knowledge was captured and represented using the model. An evaluation on the planning system has justified to say that this is a successful pioneer research project in using knowledge-based computer aided system to do bare printed circuit board process planning.

REFERENCES

1. Alting L. and Zhang H., Computer Aided Process Planning: the state-of-the-art survey, International Journal of Production Research, Vol. 27. No. 4. 1989.

2. Alting L., Manufacturing Engineering Processes, Marcel Dekker Inc., New York, 1982.

3. Amstead B.H. et. al., Manufacturing processes, Manufacturing Processes, Seventh Edition, John Wiley & Sons, New York, 1979.

4. Badiru A. B., Expert Systems Applications in Engineering and Manufacturing, Prentice Hall, 1992.

5. Berard E.V., Essays on Object-oriented Software Engineering, Volume 1, Prentice Hall, USA, 1993.

6. Booch G., Object Oriented Design with Applications, Benjamin Cummings, 1991.

7. Boothroyd G., Fundamentals of Metal Machining and Machine Tools, McGraw-Hill Book Company, 1975.

8. Bosshart, W.C., Printed Circuit Boards Design and Technology, Tata McGraw-Hill, 1988.

9. Brindley K., Newnes Electronics Assembly Handbook, Heinemann Newnes 1990.

10. Browne J. et.al., A two-stage assembly process planning tool fro robot-based flexible assembly systems, International Journal Production Research, Vol. 29, No. 2, 1991, pp. 247-266.

11. Budge t. et. al., PARSEC - Process Analysis with Recipe Support for Etcher Control, IEEE Transactions on Semiconductor Manufacturing, Vol. 3, No. 1, February, 1990.

12. Chandraker R. et.al., Intelligent control of adhesive dispensing, International Journal of Computer Integrated Manufacturing, Vol. 3, No. 2, 1990, pp. 24-34.

13. Chang T.C. and Wysk R.A., An Introduction to Automated Process Planning Systems, Prentice-hall International, 1985.

14. Chang T.C., Expert Process Planning for Manufacturing, Addison-Wesley Publishing Co., 1990.

15. Chang T.C. et. al., Computer-Aided Manufacturing, Prentice-Hall, 1991.

16. Chiu C. et.al., A collision-free sequencing algorithm for PWB assembly, Journal of Electronic Manufacturing, Vol. 1, No. 1, Sept. 1991.

17. Coad P. and Yourdon E., Object-Oriented Analysis, Second Edition, Yourdon Press, 1991.

18. Colding B. et. al., Delphei Forecast of Manufacturing Technology - Manufacturing Management, SME, USA, 1978.

19. Coombs, C.F., Printed Circuits Handbook, 2 rd ed., McGraw-Hill, 1988.

20. Coombs C.F., Printed Circuits Handbook, 3nd edition. McGraw-Hill, 1990.

21. Coombs, C.F., Printed Circuits Workbook Series, McGraw-Hill, 1990.

22. Denney R., Using Keys to Control Backtracking, AI Expert, The Magazine of Artificial Intelligence in Practice, Vol. 3, No. 6, June 1988, pp. 44-51.

23. Doyel L. E., Manufacturing Processes and Materials for Engineers, Prentice-Hall Inc., 1969.

24. Embley D. W. et. al., Object-Oriented Systems Analysis - A Model-Driven Approach, Yourdon Press,1992.

25. Follette D., Machining Fundamentals - A Basic Approach to Metal Cutting, Society of Manufacturing Engineers, Michigan, USA, 1980.

26. Fu C.Y. et. al., Expert systems add intelligence to IC manufacturing lines, Electronics Engineer, May 1988., pp. 52-55.

27. Fu C.Y. et. al., Smart Integrated Circuit Processing, IEEE Transactions on Semiconductor Manufacturing, Vol. 2., No. 4., Nov. 1989.

28. Funakoshi K. and Mizuno K., A Rule-based VLSI Process Flow Validation System with Macroscopic Process Simulation, IEEE Transactions on Semiconductor Manufacturing, Vol. 3, No. 4, Nov. 1990, pp. 239-246.

29. Ginsberg G.L., Printed Circuits Design: Featuring Computer-Aided Technologies, McGraw-Hill Inc., 1991.

30. Granville, Thinking CAPP, CIM Review, Spring 1990, pp. 50-56.

31. Guldi R.L. et. al., Process Optimization Tweaking Tool (POTT) and its Application in Controlling Oxidation Thickness, IEEE Transactions on Semiconductor Manufacturing, Vol. 2., No. 2, May 1989, pp. 54-59.

32. Ham I., Computer-Aided Process Planning: The Present and the Future, Annual of CIRP, 1988.

33. Intellicorp, Kappa-PC User's Guide, IntelliCorp. 1992.

34. IPC-D-275, Design Standard for Rigid Printed Boards and Rigid Printed Board Assemblies, Institute for Interconnection and Packing Electronic Circuits, USA, September 1990.

35. Iwata K. and Moriwaki T., The Direction of Future Development and the Role of Knowledge in Manufacturing Technology, Robotics & Computer-Integrated Manufacturing, Vol. 1, No. 3/4, 1984.

36. Jacobsen A., Object Oriented Development in an Industrial Environment, OOPSLA'87 as ACM SIGPLAN 22, 12 December 1987, 183-191.

37. Joneja A. and Chang T.C., Search Anatomy in feature-based automated process planning, Journal of Design and Manufacturing, Vol. 1, 1991, pp. 7-15.

38. Joseph A.T., Elicitation of Process-planning Knowledge in a Manufacturing Environment, International Journal of Advanced Manufacturing Technology, Vol. 6, No. 1, 1991, pp. 6.

39. Kanumury M. and Chang T.C.,Process Planning in an Automated Manufacturing Environment, Journal of Manufacturing Systems, Vol. 10, No. 1, 1991, pp. 67-78.

40. Kumara S.R.T. et. al., Expert Systems in industrial engineering, International Journal of Production Research, Vol. 24, No. 5, 1986, pp. 1107-1125.

41. Kusiak A., Expert systems for planning and scheduling manufacturing systems, European Journal of Operational Research, Vol. 34, 1988, pp. 113-130.

42. Kusiak A., Intelligent Manufacturing Systems, Prentice-Hall, 1990.

43. Kusiak A, Process Planning: A Knowledge-based & Optimization Perspective, IEEE Transactions on Robotics and Automation, Vol.7, No. 3, June 1991, pp. 257-266.

44. Lam M.M., Computer Aided SMT Assembly Planning System for Maxtor (H.K.) Ltd, Final Year Project Report, Supervised by Mr. H.W. Law, City Polytechnic of Hong Kong, Hong Kong, 1994.

45. Law H.W. et. al., The Optimization of Drill Path, Proceeding of the 1990 Pacific Conference on Manufacturing, Australia, December 1990.

46. Lee I.B.H. et. al., IKOOPPS: an intelligent knowledge-based object-oriented process planning system for the manufacture of progressive dies, Expert Systems, Vol. 8, No. 1, February 1991, pp. 19-33.

47. Liu Y.S. and Allen R., A Proposed Synthetic, Interactive Process Planning System, Computer-aided Production Engineering, 1987, pp. 201.

48. Lubars M. et.al., Object-Oriented Analysis for Evolving Systems, Proceedings on 14th International Conference on Software Engineering, Australia, 1992.

49. Merchant E.M., CAPP in CIM-integration and future trends, 19th CIRP International Seminar on Manufacturing Systems, Penn. State, U.S.A., 1-2 June. 1987.

50. Michalski, R. S. et.al., Machine learning, an Artificial Intelligence Approach, Tioga, Palo Alto, 1983.

51. Niebel B.W., "Mechanized Process Selection for Planning New Designs", ASME Paper No. 737, 1965.

52. Nolen J., Computer-automated Process Planning for World Class Manufacturing, Marcel Dekker Inc, 1989, pp. 121.

53. Pillai D.D., Developing a Decision Support System for Optimizing Automated Wafer Fabrication, IEEE Transactions on Components, Hybrids, and Manufacturing Technology, Vol. 13., No. 1, March 1990.

54. Powers Jr. J.H., Computer-automated Manufacturing, McGraw-Hill,1987, pp. 274.

55. Riley F., The Electronics Assembly Handbook, IFS Ltd, 1988.

56. Rondeau J. M., Robot programming and task planning, Manufacturing Review, Vol. 3, No. 4, December 1990, pp. 245-251.

57. Rumbaugh J. et.al., Object-Oriented Modeling and Design, Prentice Hall, 1991.

58. Sachs E. et. al., Process Control System for VLSI Fabrication, IEEE Transactions on Semiconductor Manufacturing, Vol. 4, No.2, May 1991, pp. 134-144.

59. Segre A.M., Machine Learning of Robot Assembly Plans, Kluwer Academic Publishers, 1988.

60. Shlaer S. and Mellor S. J., Object Oriented Systems Analysis: Modeling the World in Data, Yourdon Press, 1988.

61. Srihari K. and Greene T. J., Expert Process Planning for Flexible Manufacturing, CIM Review, Winter, 1990, pp. 43-50.

62. Srinivasan K. and Sanii E.T., AI-Based Process Planning for Electronic Assembly, IEE Transactions, Vol. 23, No. 2, June 1991, pp.127-137.

63. Sutton G.P., Survey of Process Planning Practices and Needs, Journal of Manufacturing Systems, Vol. 8, No.1, 1989, pp. 71.

64. Traister J.E., Design Guidelines for Surface Mount Technology, Academic Press, Inc. 1990.

65. Tsang J.P. and Lagoude Y., Process plan representation and manipulation in generic expert planning systems, Presented at the 1986 IEEE International Conference on Robotics and Automation, April 7-10, San Francisco, California. 1986.

66. Wang H.P. and Wysk R.A., A knowledge-based approach for automated process planning, International Journal of Production Research, Vol. 26, No. 6, 1988, pp. 999-1014.

67. White J.A., Production Handbook, 4th edition, Chapter 8.3, John Wiley, New York, 1987.

68. Wilson B. et.al., Delphi forecast of manufacturing technology - computer aided manufacturing, SME., Dearbornm, Michigan, 1980.

69. Wirfs-Brock R. et. al., Designing Object Oriented Software, Prentice Hall, 1990.

70. Wong H. and Leu M.C., Adaptive Genetic Algorithm for Optimal Printed Circuit Board Assembly Planning, Annals of the CIRP, Vol. 42, 1993.

71. Yankee H.W., Manufacturing Processes, Prentice-Hall Inc., USA, 1979.

APPENDIX

APPENDIX 1	KAL Representation of the In-house Process Capability	A-1
APPENDIX 2	Customer General Specification	
	2.1 KAL Representation of the CustomerA General Specification	A-11
	2.2 KAL Representation of the CustomerB General Specification	A-15
APPENDIX 3	Product Requirements	
	3.1 KAL Representation of the CircuitA Requirements	A-19
	3.2 KAL Representation of the CircuitB Requirements	A-25
	3.3 KAL Representation of the CircuitC Requirements	A-31
	3.4 KAL Representation of the CircuitD Requirements	A-37
	3.5 KAL Representation of the CircuitE Requirements	A-43
APPENDIX 4	Feature Extraction Program	
	4.1 Listing of the Circuit Feature Extraction Program	A-49
	4.1 Listing of the Solder Masking Feature Extraction Program	A-93
APPENDIX 5	Product Image Data File	
	5.1 CircuitA Circuit Image Data File	A-105
	5.2 CircuitA Solder Masking Image Data File	A-107
APPENDIX 6	Feature Data File	
	6.1 CircuitA Circuit Feature Data File	A-109
	6.2 CircuitA Solder Masking Feature Data File	A-110
APPENDIX 7	Knowledge Base	
	7.1 Customer General Specification Checking Rules	A-111
	7.2 Process Selection Rules	A-114
	7.3 Process Sequencing Rules (Methods)	A-120
	7.4 Process Capability Checking Rules	A-121
	7.5 Machine Selection Rules (Methods)	A-124
	7.6 Process Recipe Generation Rules	A-126

APPENDIX 8 Developed System Support Functions A-129

APPENDIX 9 System Support Objects
9.1 In-house Process Capability Interface Objects A-141
9.2 Customer General Specification Interface Objects A-142
9.3 Product Requirements Interface Objects A-144
9.4 "ProNameSeqNo" Objects A-148
9.5 "SuggProInHou" Objects A-149
9.6 "MCChar" Objects A-160
9.7 "MCNameList" Objects A-164
9.8 "ProcessRecipe" Objects A-166
9.9 "MessageFile" Objects A-169
9.10 "Session" Windows Objects A-170
9.11 "Button" Objects A-176

APPENDIX 10 Process Planning Results
10.1 CircuitA Process Planning Results A-190
10.2 CircuitB Process Planning Results A-193
10.3 CircuitC Process Planning Results A-197
10.4 CircuitD Process Planning Results A-201
10.5 CircuitE Process Planning Results A-205

Appendix 1 KAL Representation of the In-house Process Capability

```
/*************************************
 **** CLASS: InNCMachine
 *************************************/
MakeClass( InNCMachine, ProCapKnow );
MakeSlot( InNCMachine:NCWidth );
SetSlotOption( InNCMachine:NCWidth, VALUE_TYPE, NUMBER );
SetSlotOption( InNCMachine:NCWidth, PROMPT, "Enter/Modify Default NC Machine
   Maximum Width (in mm)." );
MakeSlot( InNCMachine:NCLength );
SetSlotOption( InNCMachine:NCLength, VALUE_TYPE, NUMBER );
SetSlotOption( InNCMachine:NCLength, PROMPT, "Enter/Modify Default NC Machine
   Longest Length (in mm)." );
MakeSlot( InNCMachine:MachinedAcc );
SetSlotOption( InNCMachine:MachinedAcc, VALUE_TYPE, NUMBER );
InNCMachine:MachinedAcc = 99;
SetSlotOption( InNCMachine:MachinedAcc, PROMPT, "Enter/Modify Default NC Machined Part
    Best Accuracy  (in mm)." );

/*************************************
 **** CLASS: InElecPlat
 *************************************/
MakeClass( InElecPlat, ProCapKnow );
MakeSlot( InElecPlat:Width );
SetSlotOption( InElecPlat:Width, VALUE_TYPE, NUMBER );
InElecPlat:Width = 0;
SetSlotOption( InElecPlat:Width, PROMPT, "Enter/Modify Elect. Plating Default
   Acceptable Maximum Panel Width (in mm)." );
MakeSlot( InElecPlat:Length );
SetSlotOption( InElecPlat:Length, VALUE_TYPE, NUMBER );
InElecPlat:Length = 0;
SetSlotOption( InElecPlat:Length, PROMPT, "Enter/Modify Elect. Plating Default
   Maximum Length (in mm)." );
MakeSlot( InElecPlat:MaxCur );
SetSlotOption( InElecPlat:MaxCur, VALUE_TYPE, NUMBER );
InElecPlat:MaxCur = 0;
SetSlotOption( InElecPlat:MaxCur, PROMPT, "Enter/Modify Elect. Plating Default
 Maximum Plating Current (in A.)." );

/*************************************
 **** CLASS: InImageTran
 *************************************/
MakeClass( InImageTran, ProCapKnow );
MakeSlot( InImageTran:Width );
SetSlotOption( InImageTran:Width, VALUE_TYPE, NUMBER );
InImageTran:Width = 0;
SetSlotOption( InImageTran:Width, PROMPT, "Enter/Modify Default Image Transfer
  Machine Maximum Width (in mm)." );
MakeSlot( InImageTran:Length );
```

```
SetSlotOption( InImageTran:Length, VALUE_TYPE, NUMBER );
InImageTran:Length = 0;
SetSlotOption( InImageTran:Length, PROMPT, "Enter/Modify Default Image Transfer Process
    Machine Maximum Length (in mm)." );
MakeSlot( InImageTran:Resolution );
SetSlotOption( InImageTran:Resolution, VALUE_TYPE, NUMBER );
InImageTran:Resolution = 0;
SetSlotOption( InImageTran:Resolution, PROMPT, "Enter/Modify Default Image Transfer
        Resultion (in mm)." );
MakeSlot( InImageTran:FirstLeadTime );
SetSlotOption( InImageTran:FirstLeadTime, VALUE_TYPE, NUMBER );
InImageTran:FirstLeadTime = 0;
SetSlotOption( InImageTran:FirstLeadTime, PROMPT, "Enter/Modify Default Image Transfer
 Pre-Production Lead Time (in day)." );
MakeSlot( InImageTran:CostPerSide );
SetSlotOption( InImageTran:CostPerSide, VALUE_TYPE, NUMBER );
InImageTran:CostPerSide = 99999;
SetSlotOption( InImageTran:CostPerSide, PROMPT, "Enter/Modify Default Image Transfer
Production Cost Per Side (in HK$)." );
MakeSlot( InImageTran:ProdLeadTime );
SetSlotOption( InImageTran:ProdLeadTime, VALUE_TYPE, NUMBER );
InImageTran:ProdLeadTime = 0.5;
SetSlotOption( InImageTran:ProdLeadTime, PROMPT, "Enter/Modify Default Image Transfer
    Production Lead Time (in day)." );

    /*************************************
     **** CLASS: InAuFing
     *************************************/
MakeClass( InAuFing, ProCapKnow );
MakeSlot( InAuFing:Height );
SetSlotOption( InAuFing:Height, VALUE_TYPE, NUMBER );
InAuFing:Height = 50;
SetSlotOption( InAuFing:Height, PROMPT, "Enter/Modify Gold Finger Process Acceptable
        Maximum Finger Height from Panel Edge (in mm)." );
MakeSlot( InAuFing:MaxCur );
SetSlotOption( InAuFing:MaxCur, VALUE_TYPE, NUMBER );
InAuFing:MaxCur = 20;
SetSlotOption( InAuFing:MaxCur, PROMPT, "Enter/Modify Gold Finger Process
    Maximum Current (in A.)." );

    /*************************************
     **** CLASS: InReflow
     *************************************/
MakeClass( InReflow, ProCapKnow );
MakeSlot( InReflow:Width );
SetSlotOption( InReflow:Width, VALUE_TYPE, NUMBER );
InReflow:Width = 355;
SetSlotOption( InReflow:Width, PROMPT, "Enter/Modify Reflow Process Acceptable
    Maximum Panel Width (in mm)." );

    /*************************************
```

```
    **** CLASS: InElelessCu
    ************************************/
MakeClass( InElelessCu, ProCapKnow );
MakeSlot( InElelessCu:Width );
SetSlotOption( InElelessCu:Width, VALUE_TYPE, NUMBER );
InElelessCu:Width = 406;
SetSlotOption( InElelessCu:Width, PROMPT, "Enter/Modify Electroless Copper Plating
    Tank Acceptable Maximum Panel Width (in mm)." );
MakeSlot( InElelessCu:Length );
SetSlotOption( InElelessCu:Length, VALUE_TYPE, NUMBER );
InElelessCu:Length = 508;
SetSlotOption( InElelessCu:Length, PROMPT, "Enter/Modify Electroless Copper Process Plating
        Tank Accetpable Panel Maximum Length (in mm)." );

   /************************************
    **** CLASS: InCuEtching
    ************************************/
MakeClass( InCuEtching, ProCapKnow );
MakeSlot( InCuEtching:UnCutRatio );
SetSlotOption( InCuEtching:UnCutRatio, VALUE_TYPE, NUMBER );
InCuEtching:UnCutRatio = 5;
SetSlotOption( InCuEtching:UnCutRatio, PROMPT, "Enter/Modify the Smallest Under Cut
of  Copper Etching Process in mm." );
MakeSlot( InCuEtching:LineWidth );
SetSlotOption( InCuEtching:LineWidth, VALUE_TYPE, NUMBER );
InCuEtching:LineWidth = 0.1;
SetSlotOption( InCuEtching:LineWidth, PROMPT, "Enter/Modify Minimum Line Width Copper
  Etching Process Can Handle (in mm)." );

   /************************************
    **** CLASS: InMechMC
    ************************************/
MakeClass( InMechMC, ProCapKnow );
MakeSlot( InMechMC:Width );
SetSlotOption( InMechMC:Width, VALUE_TYPE, NUMBER );
InMechMC:Width = 0;
SetSlotOption( InMechMC:Width, PROMPT, "Enter/Modify Default Mechanical Machine
        Acceptable Maximum Panel Width (in mm)." );
MakeSlot( InMechMC:Length );
SetSlotOption( InMechMC:Length, VALUE_TYPE, NUMBER );
InMechMC:Length = 0;
SetSlotOption( InMechMC:Length, PROMPT, "Enter/Modify Default Mechanical Machine
Acceptable Maximum Panel Length (in mm)." );
MakeSlot( InMechMC:Accuracy );
SetSlotOption( InMechMC:Accuracy, VALUE_TYPE, NUMBER );
InMechMC:Accuracy = 0;
SetSlotOption( InMechMC:Accuracy, PROMPT, "Enter/Modify Default Mechanical Machine
        Production Accruacy (in mm)." );

   /************************************
    **** CLASS: InSnPbLevelling
```

```
    ************************************/
MakeClass( InSnPbLevelling, ProCapKnow );
MakeSlot( InSnPbLevelling:Width );
SetSlotOption( InSnPbLevelling:Width, VALUE_TYPE, NUMBER );
InSnPbLevelling:Width = 304;
SetSlotOption( InSnPbLevelling:Width, PROMPT, "Enter/Modify Solder Levelling Machine Acceptable
        Maximum Panel Width (in mm)." );
MakeSlot( InSnPbLevelling:Length );
SetSlotOption( InSnPbLevelling:Length, VALUE_TYPE, NUMBER );
InSnPbLevelling:Length = 0;
SetSlotOption( InSnPbLevelling:Length, PROMPT, "Enter/Modify Solder Levelling Machine Acceptable
          Maximum Panel Length (in mm)." );

    /************************************
     **** CLASS: DocCont
     ************************************/
MakeClass( DocCont, ProCapKnow );
MakeSlot( DocCont:RevBy );
DocCont:RevBy = HangWai;
SetSlotOption( DocCont:RevBy, PROMPT, "Enter/Modify Name of Person Who Revise this
       In-house Knowledge Base." );
MakeSlot( DocCont:RevDate );
DocCont:RevDate = 20Feb94;
SetSlotOption( DocCont:RevDate, PROMPT, "Enter the Revision Date." );

    /************************************
     **** CLASS: InNCDrill
     ************************************/
MakeClass( InNCDrill, InNCMachine );
MakeSlot( InNCDrill:MaxRPM );
SetSlotOption( InNCDrill:MaxRPM, VALUE_TYPE, NUMBER );
InNCDrill:MaxRPM = 70000;
SetSlotOption( InNCDrill:MaxRPM, PROMPT, "Enter/Modify NC Drilling Machine
       Maximum RPM." );
InNCDrill:NCWidth = 550;
SetSlotOption( InNCDrill:NCWidth, PROMPT, "Enter/Modify NC Drilling Machine
  Maximum Panel Width (in mm)." );
InNCDrill:NCLength = 600;
SetSlotOption( InNCDrill:NCLength, PROMPT, "Enter/Modify NC Drilling Machine
  Maximum Panel Length (in mm)." );
InNCDrill:MachinedAcc = 0.05;
SetSlotOption( InNCDrill:MachinedAcc, PROMPT, "Enter/Modify NC Drilling Machine Drilled
   Hole Best Position Accuracy (in mm)." );

    /************************************
     **** CLASS: InNCRout
     ************************************/
MakeClass( InNCRout, InNCMachine );
InNCRout:NCWidth = 406;
SetSlotOption( InNCRout:NCWidth, PROMPT, "Enter/Modify NC Routing Machine
  Maximum Panel Width (in mm)." );
```

```
InNCRout:NCLength = 355;
SetSlotOption( InNCRout:NCLength, PROMPT, "Enter/Modify NC Routing Machine Longest
        Panel Length (in mm)." );
InNCRout:MachinedAcc = 0.075;
SetSlotOption( InNCRout:MachinedAcc, PROMPT, "Enter/Modify NC Routing Machine Machined
    Parts Best Outline Accuracy (in mm)." );

    /*************************************
     **** CLASS: InPatternPlat
     *************************************/
MakeClass( InPatternPlat, InElecPlat );
InPatternPlat:Width = 406;
SetSlotOption( InPatternPlat:Width, PROMPT, "Enter/Modify Elect. Pattern Plating Default Max.
Width." );
InPatternPlat:Length = 508;
SetSlotOption( InPatternPlat:Length, PROMPT, "Enter/Modify Elect. Pattern Plating Default Max.
Length." );
InPatternPlat:MaxCur = 0;
SetSlotOption( InPatternPlat:MaxCur, PROMPT, "Enter/Modify Elect. Pattern Plating Default Max.
Plating Current." );
MakeSlot( InPatternPlat:TimeToDeposit );
SetSlotOption( InPatternPlat:TimeToDeposit, VALUE_TYPE, NUMBER );
InPatternPlat:TimeToDeposit = 0;
SetSlotOption( InPatternPlat:TimeToDeposit, PROMPT, "Input/modify Pattern Plating Default Time to
Deposit. (Pre-setted to ZERO)" );

    /*************************************
     **** CLASS: InPanelPlat
     *************************************/
MakeClass( InPanelPlat, InElecPlat );
MakeSlot( InPanelPlat:Width );
SetSlotOption( InPanelPlat:Width, VALUE_TYPE, NUMBER );
InPanelPlat:Width = 406;
SetSlotOption( InPanelPlat:Width, PROMPT, "Enter/Modify Elect. Panel Plating
 Tank Maximum Width (in mm)." );
MakeSlot( InPanelPlat:Length );
SetSlotOption( InPanelPlat:Length, VALUE_TYPE, NUMBER );
InPanelPlat:Length = 508;
SetSlotOption( InPanelPlat:Length, PROMPT, "Enter/Modify Elect. Panel Plating
Tank Maximum Length (in mm). " );
MakeSlot( InPanelPlat:MaxCur );
SetSlotOption( InPanelPlat:MaxCur, VALUE_TYPE, NUMBER );
InPanelPlat:MaxCur = 0;
SetSlotOption( InPanelPlat:MaxCur, PROMPT, "Enter/Modify Elect. Panel Plating
Maximum Plating Current (in A.)." );
MakeSlot( InPanelPlat:CurrentDensity );
SetSlotOption( InPanelPlat:CurrentDensity, VALUE_TYPE, NUMBER );
InPanelPlat:CurrentDensity = 40;
SetSlotOption( InPanelPlat:CurrentDensity, PROMPT, "Input/Modify Panel Copper Plating
    Current Density in ASF." );
MakeSlot( InPanelPlat:TimeToDeposit );
```

```
SetSlotOption( InPanelPlat:TimeToDeposit, VALUE_TYPE, NUMBER );
InPanelPlat:TimeToDeposit = 3;
SetSlotOption( InPanelPlat:TimeToDeposit, PROMPT, "Input/Modify Panel Copper Plating
    Time (in min.) to Deposit 0.0001 in. Copper." );

    /*************************************
     **** CLASS: InCu
     *************************************/
MakeClass( InCu, InPatternPlat );
MakeSlot( InCu:Width );
SetSlotOption( InCu:Width, VALUE_TYPE, NUMBER );
InCu:Width = 406;
SetSlotOption( InCu:Width, PROMPT, "Enter/Modify Elect. Copper Pattern Plating
     Tank Maximum Width (in mm)." );
MakeSlot( InCu:Length );
SetSlotOption( InCu:Length, VALUE_TYPE, NUMBER );
InCu:Length = 508;
SetSlotOption( InCu:Length, PROMPT, "Enter/Modify Elect. Copper Pattern Plating
     Tank Maximum Length (in mm). " );
MakeSlot( InCu:MaxCur );
SetSlotOption( InCu:MaxCur, VALUE_TYPE, NUMBER );
InCu:MaxCur = 0;
SetSlotOption( InCu:MaxCur, PROMPT, "Enter/Modify Elect. Copper Pattern Plating
    Maximum Plating Current (in A)." );
MakeSlot( InCu:CurrentDensity );
SetSlotOption( InCu:CurrentDensity, VALUE_TYPE, NUMBER );
InCu:CurrentDensity = 40;
SetSlotOption( InCu:CurrentDensity, PROMPT, "Input/modify in-house Pattern Copper Plating Current
Density in ASF." );
MakeSlot( InCu:TimeToDeposit );
SetSlotOption( InCu:TimeToDeposit, VALUE_TYPE, NUMBER );
InCu:TimeToDeposit = 3;
SetSlotOption( InCu:TimeToDeposit, PROMPT, "Input/modify Pattern Copper Plating
     Time (in min.) to Deposit 0.0001 in Copper." );

    /*************************************
     **** CLASS: InAu
     *************************************/
MakeClass( InAu, InPatternPlat );
MakeSlot( InAu:Width );
SetSlotOption( InAu:Width, VALUE_TYPE, NUMBER );
InAu:Width = 406;
SetSlotOption( InAu:Width, PROMPT, "Enter/Modify Elect. Gold Pattern Plating
     Tank Maximum Width (in mm)." );
MakeSlot( InAu:Length );
SetSlotOption( InAu:Length, VALUE_TYPE, NUMBER );
InAu:Length = 508;
SetSlotOption( InAu:Length, PROMPT, "Enter/Modify Elect. Gold Pattern Plating
    Tank Maximum Length (in mm)." );
MakeSlot( InAu:MaxCur );
SetSlotOption( InAu:MaxCur, VALUE_TYPE, NUMBER );
```

```
InAu:MaxCur = 20;
SetSlotOption( InAu:MaxCur, PROMPT, "Enter/Modify Elect. Gold Pattern Plating
    Maximum Plating Current (in A)." );
MakeSlot( InAu:CurrentDensity );
SetSlotOption( InAu:CurrentDensity, VALUE_TYPE, NUMBER );
InAu:CurrentDensity = 5;
SetSlotOption( InAu:CurrentDensity, PROMPT, "Input/modify Pattern Gold Plating Current Density in
ASF." );
MakeSlot( InAu:TimeToDeposit );
SetSlotOption( InAu:TimeToDeposit, VALUE_TYPE, NUMBER );
InAu:TimeToDeposit = 38;
SetSlotOption( InAu:TimeToDeposit, PROMPT, "Input/modify Pattern Gold Plating Time (in min.) to
Deposit 0.0001 in Thickness." );

    /*************************************
     **** CLASS: InSnPb
     *************************************/
MakeClass( InSnPb, InPatternPlat );
MakeSlot( InSnPb:Width );
SetSlotOption( InSnPb:Width, VALUE_TYPE, NUMBER );
InSnPb:Width = 406;
SetSlotOption( InSnPb:Width, PROMPT, "Enter/Modify Elect. Sn/Pb Pattern
      Plating Tank Maximum Width (in mm)." );
MakeSlot( InSnPb:Length );
SetSlotOption( InSnPb:Length, VALUE_TYPE, NUMBER );
InSnPb:Length = 508;
SetSlotOption( InSnPb:Length, PROMPT, "Enter/Modify Elect. Sn/Pb Pattern Plating
        Tank Maximum Length (in mm). " );
MakeSlot( InSnPb:MaxCur );
SetSlotOption( InSnPb:MaxCur, VALUE_TYPE, NUMBER );
InSnPb:MaxCur = 0;
SetSlotOption( InSnPb:MaxCur, PROMPT, "Enter/Modify Elect. Sn/Pb Pattern Plating
            Maximum Plating Current." );
MakeSlot( InSnPb:CurrentDensity );
SetSlotOption( InSnPb:CurrentDensity, VALUE_TYPE, NUMBER );
InSnPb:CurrentDensity = 20;
SetSlotOption( InSnPb:CurrentDensity, PROMPT, "Input/modify Tin/lead Plating Current Density in
ASF." );
MakeSlot( InSnPb:TimeToDeposit );
SetSlotOption( InSnPb:TimeToDeposit, VALUE_TYPE, NUMBER );
InSnPb:TimeToDeposit = 2.5;
SetSlotOption( InSnPb:TimeToDeposit, PROMPT, "Input/modify Tin/lead Plating Time (in min.) to
Deposit 0.0001 in Thickness." );

    /*************************************
     **** CLASS: InNi
     *************************************/
MakeClass( InNi, InPatternPlat );
MakeSlot( InNi:Width );
SetSlotOption( InNi:Width, VALUE_TYPE, NUMBER );
InNi:Width = 0;
```

```
SetSlotOption( InNi:Width, PROMPT, "Enter/Modify Elect. Nickel Pattern
  Plating Tank Maximum Width (in mm)." );
MakeSlot( InNi:Length );
SetSlotOption( InNi:Length, VALUE_TYPE, NUMBER );
InNi:Length = 0;
SetSlotOption( InNi:Length, PROMPT, "Enter/Modify Elect. Nickel Pattern Plating
     Tank Maximum Length (in mm). " );
MakeSlot( InNi:MaxCur );
SetSlotOption( InNi:MaxCur, VALUE_TYPE, NUMBER );
InNi:MaxCur = 0;
SetSlotOption( InNi:MaxCur, PROMPT, "Enter/Modify Elect. Nickel Pattern Plating
      Maximum Plating Current (in A)." );
MakeSlot( InNi:CurrentDensity );
SetSlotOption( InNi:CurrentDensity, VALUE_TYPE, NUMBER );
InNi:CurrentDensity = 30;
SetSlotOption( InNi:CurrentDensity, PROMPT, "Input/modify Nickel plating current density (in ASF)." );
MakeSlot( InNi:TimeToDeposit );
SetSlotOption( InNi:TimeToDeposit, VALUE_TYPE, NUMBER );
InNi:TimeToDeposit = 3.86;
SetSlotOption( InNi:TimeToDeposit, PROMPT, "Input/modify Nickel Plating Time (in min.) to Deposit
0.0001 in Thickness." );

    /************************************
     **** CLASS: InSilkScreen
     ************************************/
MakeClass( InSilkScreen, InImageTran );
InSilkScreen:Width = 600;
SetSlotOption( InSilkScreen:Width, PROMPT, "Enter/Modify Silk Screen Machine
     Maximum Width (in mm)." );
InSilkScreen:Length = 600;
SetSlotOption( InSilkScreen:Length, PROMPT, "Enter/Modify Silk Screen Process Machine
    Acceptable Maximum Panel Length." );
InSilkScreen:Resolution = 0.19;
SetSlotOption( InSilkScreen:Resolution, PROMPT, "Enter/Modify Silk Screen Machine Best
        Resultion (in mm)." );
InSilkScreen:FirstLeadTime = 2;
SetSlotOption( InSilkScreen:FirstLeadTime, PROMPT, "Enter/Modify Silk Screen Process Pre-
Production Lead Time (in days)." );
InSilkScreen:CostPerSide = 1;
SetSlotOption( InSilkScreen:CostPerSide, PROMPT, "Enter/Modify Silk Screen Process Lowest
            Production Cost Per Side." );
InSilkScreen:ProdLeadTime = 0.5;
SetSlotOption( InSilkScreen:ProdLeadTime, PROMPT, "Enter/Modify Silk Screen Process
Production Lead Time (in day)." );

    /************************************
     **** CLASS: InDryFilm
     ************************************/
MakeClass( InDryFilm, InImageTran );
InDryFilm:Width = 508;
SetSlotOption( InDryFilm:Width, PROMPT, "Enter/Modify Dry Film Process Acceptable
```

```
      Maximum Panel Width (in mm)." );
InDryFilm:Length = 508;SetSlotOption( InDryFilm:Length, PROMPT, "Enter/Modify Dry Film Process
Acceptable
          Maximum Panel Length." );
InDryFilm:Resolution = 0.05;
SetSlotOption( InDryFilm:Resolution, PROMPT, "Enter/Modify Dry Film Process Best
   Production Resultion (in mm)." );
InDryFilm:FirstLeadTime = 0.5;
SetSlotOption( InDryFilm:FirstLeadTime, PROMPT, "Enter/Modify Dry Film Pre-Production
          Lead Time (in day)." );
InDryFilm:CostPerSide = 2;
SetSlotOption( InDryFilm:CostPerSide, PROMPT, "Enter/Modify Dry Film Production
     Cost Per Side (in HK$)." );
InDryFilm:ProdLeadTime = 0.08;
SetSlotOption( InDryFilm:ProdLeadTime, PROMPT, "Enter/Modify Dry Film Production
        Lead Time (in day)." );

    /*************************************
     **** CLASS: InWetFilm
     *************************************/
MakeClass( InWetFilm, InImageTran );
InWetFilm:Width = 508;
SetSlotOption( InWetFilm:Width, PROMPT, "Enter/Modify Wet Film Process Acceptable
      Maximum Panel Width (in mm)." );
InWetFilm:Length = 508;
SetSlotOption( InWetFilm:Length, PROMPT, "Enter/Modify Wet Film Process Acceptable
        Maximum Panel Length." );
InWetFilm:Resolution = 0.1;
SetSlotOption( InWetFilm:Resolution, PROMPT, "Enter/Modify Wet Film Process
    Best Resultion (in mm)." );
InWetFilm:FirstLeadTime = 1;
SetSlotOption( InWetFilm:FirstLeadTime, PROMPT, "Enter/Modify Wet Film Process Pre-Production
                Lead Time (in day)." );
InWetFilm:CostPerSide = 1.5;
SetSlotOption( InWetFilm:CostPerSide, PROMPT, "Enter/Modify Wet Film Production
      Cost Per Side (in HK$)." );
InWetFilm:ProdLeadTime = 0.3;
SetSlotOption( InWetFilm:ProdLeadTime, PROMPT, "Enter/Modify Wet Film Production
          Lead Time (in day)." );

    /*************************************
     **** CLASS: InBlank
     *************************************/
MakeClass( InBlank, InMechMC );
InBlank:Width = 600;
SetSlotOption( InBlank:Width, PROMPT, "Enter/Modify Blanking Machine
 Maximum UNIT Width (in mm)." );
InBlank:Length = 600;
SetSlotOption( InBlank:Length, PROMPT, "Enter/Modify Blanking Machine Maximum
        Board UNIT Length (in mm)." );
InBlank:Accuracy = 0.15;
```

```
SetSlotOption( InBlank:Accuracy, PROMPT, "Enter/Modify Blanking Machine Best
        Produced Board Outline Accruacy (in mm)." );
```

Appendix 2.1 KAL Representation of the CustomerA General Specification

```
/*************************************
 **** CLASS: SpecInfo
 *************************************/
MakeClass( SpecInfo, CustGenSpec );
MakeSlot( SpecInfo:CustName );
SpecInfo:CustName = CustomerA;
SetSlotOption( SpecInfo:CustName, PROMPT, "Name of the Customer." );
MakeSlot( SpecInfo:SpecNo );
SpecInfo:SpecNo = -;
SetSlotOption( SpecInfo:SpecNo, PROMPT, "Specification Number." );
MakeSlot( SpecInfo:RevDate );
SpecInfo:RevDate = 11June84;
SetSlotOption( SpecInfo:RevDate, PROMPT, "Revision Date. );
MakeSlot( SpecInfo:UpDatedBy );
SpecInfo:UpDatedBy = "Hang Wai";
SetSlotOption( SpecInfo:UpDatedBy, PROMPT, "Input/Modify by." );

/*************************************
 **** CLASS: CustLaminate
 **** Type 1 laminate approved by the customer.
 ****
 *************************************/
MakeClass( CustLaminate, CustGenSpec );
SetClassComment( CustLaminate, "Type 1 laminate approved by the customer." );
MakeSlot( CustLaminate:TypeOfLaminate1 );
SetSlotComment( CustLaminate:TypeOfLaminate1, "Type one of laminate material approved." );
SetSlotOption( CustLaminate:TypeOfLaminate1, ALLOWABLE_VALUES, "CAM3,1.6,1/1", "CAM-
3,1.6,1/1", "FR-4,0.8,0.5/0", "FR-4,0.8,1/0", "FR-4,0.8,0.5/0.5", "FR-4,0.8,1/1", "FR-4,1.6,0.5/0", "FR-
4,1.6,1/0", "FR-4,1.6,0.5/0.5", "FR-4,1.6,1/1" );
CustLaminate:TypeOfLaminate1 = "FR-4,1.6,1/1";
SetSlotOption( CustLaminate:TypeOfLaminate1, PROMPT, "Approved Laminate Type 1." );
MakeSlot( CustLaminate:TypeOfLaminate2 );
SetSlotOption( CustLaminate:TypeOfLaminate2, ALLOWABLE_VALUES, "CAM3,1.6,1/1", "CAM-
3,1.6,1/1", "FR-4,0.8,0.5/0", "FR-4,0.8,1/0", "FR-4,0.8,0.5/0.5", "FR-4,0.8,1/1", "FR-4,1.6,0.5/0", "FR-
4,1.6,1/0", "FR-4,1.6,0.5/0.5", "FR-4,1.6,1/1" );
CustLaminate:TypeOfLaminate2 = "FR-4,1.6,1/0";
SetSlotOption( CustLaminate:TypeOfLaminate2, PROMPT, "Approved Laminate Type 2." );
MakeSlot( CustLaminate:TypeOfLaminate3 );
SetSlotOption( CustLaminate:TypeOfLaminate3, ALLOWABLE_VALUES, "CAM3,1.6,1/1", "CAM-
3,1.6,1/1", "FR-4,0.8,0.5/0", "FR-4,0.8,1/0", "FR-4,0.8,0.5/0.5", "FR-4,0.8,1/1", "FR-4,1.6,0.5/0", "FR-
4,1.6,1/0", "FR-4,1.6,0.5/0.5", "FR-4,1.6,1/1" );
CustLaminate:TypeOfLaminate3 = "CAM3,1.6,1/1";
SetSlotOption( CustLaminate:TypeOfLaminate3, PROMPT, "Approved Laminate Type 3." );

/*************************************
 **** CLASS: CustCirStd
 *************************************/
```

```
MakeClass( CustCirStd, CustGenSpec );
MakeSlot( CustCirStd:MinLnWid );
SetSlotOption( CustCirStd:MinLnWid, VALUE_TYPE, NUMBER );
CustCirStd:MinLnWid = 20;
SetSlotOption( CustCirStd:MinLnWid, PROMPT, "Acceptable Min. Circuit Line
      Width in 0.01mm." );
MakeSlot( CustCirStd:MinLnSpac );
SetSlotOption( CustCirStd:MinLnSpac, VALUE_TYPE, NUMBER );
CustCirStd:MinLnSpac = 15;
SetSlotOption( CustCirStd:MinLnSpac, PROMPT, "Acceptable Min. Line Spcaing in 0.01mm." );
MakeSlot( CustCirStd:Dev );
SetSlotOption( CustCirStd:Dev, VALUE_TYPE, NUMBER );
CustCirStd:Dev = 25;
SetSlotOption( CustCirStd:Dev, PROMPT, "Circuit Width Deviation from
   Original Artwork  (%)." );
MakeSlot( CustCirStd:MinAnnRing );
SetSlotOption( CustCirStd:MinAnnRing, VALUE_TYPE, NUMBER );
CustCirStd:MinAnnRing = 15;
SetSlotOption( CustCirStd:MinAnnRing, PROMPT, "Acceptable Min. Circuit Annular
         Ring in 0.01mm." );

    /*************************************
     **** CLASS: CustOutFin
     *************************************/
MakeClass( CustOutFin, CustGenSpec );
MakeSlot( CustOutFin:DevDimen );
SetSlotOption( CustOutFin:DevDimen, VALUE_TYPE, NUMBER );
CustOutFin:DevDimen = 25;
SetSlotOption( CustOutFin:DevDimen, PROMPT, "Acceptable Max. Deviation in Outline
      Dimension in 0.01mm." );
MakeSlot( CustOutFin:SurFinReq );
SetSlotOption( CustOutFin:SurFinReq, ALLOWABLE_VALUES, Best, Above, Average, Normal );
CustOutFin:SurFinReq = Average;
SetSlotOption( CustOutFin:SurFinReq, PROMPT, "Acceptable Outline Surface Finishing." );

    /*************************************
     **** CLASS: CustHoleStd
     *************************************/
MakeClass( CustHoleStd, CustGenSpec );
MakeSlot( CustHoleStd:DevLoc );
SetSlotOption( CustHoleStd:DevLoc, VALUE_TYPE, NUMBER );
CustHoleStd:DevLoc = 13;
SetSlotOption( CustHoleStd:DevLoc, PROMPT, "Acceptable Deviation of Drilled Hole
      Location in 0.01mm." );
MakeSlot( CustHoleStd:DevDia );
SetSlotOption( CustHoleStd:DevDia, VALUE_TYPE, NUMBER );
CustHoleStd:DevDia = 5;
SetSlotOption( CustHoleStd:DevDia, PROMPT, "Acceptable Deviation in Hole Diameter
         (+/- in 0.01mm)." );

    /*************************************
```

```
    **** CLASS: CustAppSurFin
    ************************************/
MakeClass( CustAppSurFin, CustGenSpec );
MakeSlot( CustAppSurFin:Min );
SetSlotOption( CustAppSurFin:Min, VALUE_TYPE, NUMBER );
CustAppSurFin:Min = 0.5;
SetSlotOption( CustAppSurFin:Min, PROMPT, "Acceptable Default Surface Finishing
   Metal Min. Plated Thickness  in 0.001in." );

    /************************************
    **** CLASS: CustAuFing
    ************************************/
MakeClass( CustAuFing, CustGenSpec );
MakeSlot( CustAuFing:MinNiThick );
SetSlotOption( CustAuFing:MinNiThick, VALUE_TYPE, NUMBER );
CustAuFing:MinNiThick = 0.3;
SetSlotOption( CustAuFing:MinNiThick, PROMPT, "Acceptable Min. Plated Nickel Thickness
      Under Gold in 0.001in." );
MakeSlot( CustAuFing:MinAuThick );
SetSlotOption( CustAuFing:MinAuThick, VALUE_TYPE, NUMBER );
CustAuFing:MinAuThick = 0.045;
SetSlotOption( CustAuFing:MinAuThick, PROMPT, "Acceptable Min. Gold Finger Gold
     Thickness in 0.001in." );

    /************************************
    **** CLASS: CustSMMatl
    ************************************/
MakeClass( CustSMMatl, CustGenSpec );
MakeSlot( CustSMMatl:SMMaterial );
CustSMMatl:SMMaterial = PC501;
SetSlotOption( CustSMMatl:SMMaterial, PROMPT, "    Acceptable Type of Solder
      Masking Material." );

    /************************************
    **** CLASS: CustPlatCu
    ************************************/
MakeClass( CustPlatCu, CustGenSpec );
MakeSlot( CustPlatCu:MinThick );
SetSlotOption( CustPlatCu:MinThick, VALUE_TYPE, NUMBER );
CustPlatCu:MinThick = 1;
SetSlotOption( CustPlatCu:MinThick, PROMPT, "Acceptable Min. Plated Hole Wall
  Copper Thickness in 0.001in" );
MakeSlot( CustPlatCu:MaxThick );
SetSlotOption( CustPlatCu:MaxThick, VALUE_TYPE, NUMBER );
CustPlatCu:MaxThick = 3;
SetSlotOption( CustPlatCu:MaxThick, PROMPT, "Plated Copper Max. Hole Wall
    Thickness in 0.001in." );

    /************************************
    **** CLASS: CustSMStd
    ************************************/
```

```
MakeClass( CustSMStd, CustGenSpec );
MakeSlot( CustSMStd:MinThick );
SetSlotOption( CustSMStd:MinThick, VALUE_TYPE, NUMBER );
CustSMStd:MinThick = 0.8;
SetSlotOption( CustSMStd:MinThick, PROMPT, "Acceptable Min. Solder Masking
   Thickness in 0.01mm." );
MakeSlot( CustSMStd:PadClear );
SetSlotOption( CustSMStd:PadClear, VALUE_TYPE, NUMBER );
CustSMStd:PadClear = 10;
SetSlotOption( CustSMStd:PadClear, PROMPT, "Acceptable Solder Masking Pad
Minimum Clearance in 0.01mm." );

    /************************************
     **** CLASS: CustEtchStd
     ************************************/
MakeClass( CustEtchStd, CustGenSpec );
MakeSlot( CustEtchStd:UCutRatio );
SetSlotOption( CustEtchStd:UCutRatio, VALUE_TYPE, NUMBER );
CustEtchStd:UCutRatio = 5;
SetSlotOption( CustEtchStd:UCutRatio, PROMPT, "Acceptable Maximum Under Cut in 0.01 mm." );
MakeSlot( CustEtchStd:DevOrig );
SetSlotOption( CustEtchStd:DevOrig, VALUE_TYPE, NUMBER );
CustEtchStd:DevOrig = 25;
SetSlotOption( CustEtchStd:DevOrig, PROMPT, "Width of Etched Circuit Deviation from
  the Original Artwork Width (in %)." );

    /************************************
     **** CLASS: SnPb
     ************************************/
MakeClass( SnPb, CustAppSurFin );
SnPb:Min = 0.3;
SetSlotOption( SnPb:Min, PROMPT, "Acceptable Min. Plated Sn/Pb Surface
  Finishing Thickness in 0.001in." );

    /************************************
     **** CLASS: Ni
     ************************************/
MakeClass( Ni, CustAppSurFin );
Ni:Min = 0.2;
SetSlotOption( Ni:Min, PROMPT, "Acceptable Min. Plated Ni Surface
Finishing Thickness in 0.001in." );

    /************************************
     **** CLASS: Au
     ************************************/
MakeClass( Au, CustAppSurFin );
Au:Min = 0.02;
SetSlotOption( Au:Min, PROMPT, "Acceptable Min. Plated Gold Surface
Finishing Thickness in 0.001in." );
```

Appendix 2.2 KAL Representation of the CustomerB General Specification

```
/*************************************
 **** CLASS: SpecInfo
 *************************************/
MakeClass( SpecInfo, CustGenSpec );
MakeSlot( SpecInfo:CustName );
SpecInfo:CustName = CustomerB;
SetSlotOption( SpecInfo:CustName, PROMPT, "Name of the Customer." );
MakeSlot( SpecInfo:SpecNo );
SpecInfo:SpecNo = "93-2653";
SetSlotOption( SpecInfo:SpecNo, PROMPT, "Specification Number." );
MakeSlot( SpecInfo:RevDate );
SpecInfo:RevDate = 25Sept84;
SetSlotOption( SpecInfo:RevDate, PROMPT, "Revision Date." );
MakeSlot( SpecInfo:UpDatedBy );
SpecInfo:UpDatedBy = "Hang Wai";
SetSlotOption( SpecInfo:UpDatedBy, PROMPT, "Input/Modify by." );

/*************************************
 **** CLASS: CustLaminate
 **** Type 1 laminate approved by the customer.
 ****
 *************************************/
MakeClass( CustLaminate, CustGenSpec );
SetClassComment( CustLaminate, "Type 1 laminate approved by the customer." );
MakeSlot( CustLaminate:TypeOfLaminate1 );
SetSlotComment( CustLaminate:TypeOfLaminate1, "Type one of laminate material approved." );
SetSlotOption( CustLaminate:TypeOfLaminate1, ALLOWABLE_VALUES, "CAM3,1.6,1/1", "CAM-
3,1.6,1/1", "FR-4,0.8,0.5/0", "FR-4,0.8,1/0", "FR-4,0.8,0.5/0.5", "FR-4,0.8,1/1", "FR-4,1.6,0.5/0", "FR-
4,1.6,1/0", "FR-4,1.6,0.5/0.5", "FR-4,1.6,1/1" );
CustLaminate:TypeOfLaminate1 = "FR-4,1.6,1/1";
SetSlotOption( CustLaminate:TypeOfLaminate1, PROMPT, "Approved Laminate Type 1." );
MakeSlot( CustLaminate:TypeOfLaminate2 );
SetSlotOption( CustLaminate:TypeOfLaminate2, ALLOWABLE_VALUES, "CAM3,1.6,1/1", "CAM-
3,1.6,1/1", "FR-4,0.8,0.5/0", "FR-4,0.8,1/0", "FR-4,0.8,0.5/0.5", "FR-4,0.8,1/1", "FR-4,1.6,0.5/0", "FR-
4,1.6,1/0", "FR-4,1.6,0.5/0.5", "FR-4,1.6,1/1" );
CustLaminate:TypeOfLaminate2 = "FR-4,1.6,0.5/0.5";
SetSlotOption( CustLaminate:TypeOfLaminate2, PROMPT, "Approved Laminate Type 2." );
MakeSlot( CustLaminate:TypeOfLaminate3 );
SetSlotOption( CustLaminate:TypeOfLaminate3, ALLOWABLE_VALUES, "CAM3,1.6,1/1", "CAM-
3,1.6,1/1", "FR-4,0.8,0.5/0", "FR-4,0.8,1/0", "FR-4,0.8,0.5/0.5", "FR-4,0.8,1/1", "FR-4,1.6,0.5/0", "FR-
4,1.6,1/0", "FR-4,1.6,0.5/0.5", "FR-4,1.6,1/1" );
CustLaminate:TypeOfLaminate3 = "FR-4,1.6,1/0";
SetSlotOption( CustLaminate:TypeOfLaminate3, PROMPT, "Approved Laminate Type 3." );

/*************************************
 **** CLASS: CustCirStd
 *************************************/
MakeClass( CustCirStd, CustGenSpec );
```

```
MakeSlot( CustCirStd:MinLnWid );
SetSlotOption( CustCirStd:MinLnWid, VALUE_TYPE, NUMBER );
CustCirStd:MinLnWid = 12.5;
SetSlotOption( CustCirStd:MinLnWid, PROMPT, "Acceptable Min. Circuit Line
      Width in 0.01mm." );
MakeSlot( CustCirStd:MinLnSpac );
SetSlotOption( CustCirStd:MinLnSpac, VALUE_TYPE, NUMBER );
CustCirStd:MinLnSpac = 12.5;
SetSlotOption( CustCirStd:MinLnSpac, PROMPT, "Acceptable Min. Line Spcaing  in 0.01mm." );
MakeSlot( CustCirStd:Dev );
SetSlotOption( CustCirStd:Dev, VALUE_TYPE, NUMBER );
CustCirStd:Dev = 15;
SetSlotOption( CustCirStd:Dev, PROMPT, "Circuit Width Deviation from
   Original Artwork  (%)." );
MakeSlot( CustCirStd:MinAnnRing );
SetSlotOption( CustCirStd:MinAnnRing, VALUE_TYPE, NUMBER );
CustCirStd:MinAnnRing = 5;
SetSlotOption( CustCirStd:MinAnnRing, PROMPT, "Acceptable Min. Circuit Annular
        Ring in 0.01mm." );

    /*************************************
     **** CLASS: CustOutFin
     *************************************/
MakeClass( CustOutFin, CustGenSpec );
MakeSlot( CustOutFin:DevDimen );
SetSlotOption( CustOutFin:DevDimen, VALUE_TYPE, NUMBER );
CustOutFin:DevDimen = 20;
SetSlotOption( CustOutFin:DevDimen, PROMPT, "Acceptable Max. Deviation in Outline
      Dimension in 0.01mm." );
MakeSlot( CustOutFin:SurFinReq );
SetSlotOption( CustOutFin:SurFinReq, ALLOWABLE_VALUES, Best, Above, Average, Normal );
CustOutFin:SurFinReq = Normal;
SetSlotOption( CustOutFin:SurFinReq, PROMPT, "Acceptable Outline Surface Finishing." );

    /*************************************
     **** CLASS: CustHoleStd
     *************************************/
MakeClass( CustHoleStd, CustGenSpec );
MakeSlot( CustHoleStd:DevLoc );
SetSlotOption( CustHoleStd:DevLoc, VALUE_TYPE, NUMBER );
CustHoleStd:DevLoc = 7.5;
SetSlotOption( CustHoleStd:DevLoc, PROMPT, "Acceptable Deviation of Drilled Hole
      Location in 0.01mm." );
MakeSlot( CustHoleStd:DevDia );
SetSlotOption( CustHoleStd:DevDia, VALUE_TYPE, NUMBER );
CustHoleStd:DevDia = 7.5;
SetSlotOption( CustHoleStd:DevDia, PROMPT, "Acceptable Deviation in Hole Diameter
          (+/- in 0.01mm)." );

    /*************************************
     **** CLASS: CustAppSurFin
```

```
    ************************************/
MakeClass( CustAppSurFin, CustGenSpec );
MakeSlot( CustAppSurFin:Min );
SetSlotOption( CustAppSurFin:Min, VALUE_TYPE, NUMBER );
CustAppSurFin:Min = .3;
SetSlotOption( CustAppSurFin:Min, PROMPT, "Acceptable Default Surface Finishing
   Metal Min. Plated Thickness in 0.001in." );

    /************************************
     **** CLASS: CustAuFing
     ************************************/
MakeClass( CustAuFing, CustGenSpec );
MakeSlot( CustAuFing:MinNiThick );
SetSlotOption( CustAuFing:MinNiThick, VALUE_TYPE, NUMBER );
CustAuFing:MinNiThick = 0.15;
SetSlotOption( CustAuFing:MinNiThick, PROMPT, "Acceptable Min. Plated Nickel Thickness
      Under Gold in 0.001in." );
MakeSlot( CustAuFing:MinAuThick );
SetSlotOption( CustAuFing:MinAuThick, VALUE_TYPE, NUMBER );
CustAuFing:MinAuThick = 0.025;
SetSlotOption( CustAuFing:MinAuThick, PROMPT, "Acceptable Min. Gold Finger Gold
     Thickness in 0.001in." );

    /************************************
     **** CLASS: CustSMMatl
     ************************************/
MakeClass( CustSMMatl, CustGenSpec );
MakeSlot( CustSMMatl:SMMaterial );
CustSMMatl:SMMaterial = PC501;
SetSlotOption( CustSMMatl:SMMaterial, PROMPT, "    Acceptable Type of Solder
      Masking Material." );

    /************************************
     **** CLASS: CustPlatCu
     ************************************/
MakeClass( CustPlatCu, CustGenSpec );
MakeSlot( CustPlatCu:MinThick );
SetSlotOption( CustPlatCu:MinThick, VALUE_TYPE, NUMBER );
CustPlatCu:MinThick = 1;
SetSlotOption( CustPlatCu:MinThick, PROMPT, "Acceptable Min. Plated Hole Wall
  Copper Thickness in 0.001in" );
MakeSlot( CustPlatCu:MaxThick );
SetSlotOption( CustPlatCu:MaxThick, VALUE_TYPE, NUMBER );
CustPlatCu:MaxThick = 3;
SetSlotOption( CustPlatCu:MaxThick, PROMPT, "Plated Copper Max. Hole Wall
    Thickness in 0.001in." );

    /************************************
     **** CLASS: CustSMStd
     ************************************/
MakeClass( CustSMStd, CustGenSpec );
```

```
MakeSlot( CustSMStd:MinThick );
SetSlotOption( CustSMStd:MinThick, VALUE_TYPE, NUMBER );
CustSMStd:MinThick = 1.78;
SetSlotOption( CustSMStd:MinThick, PROMPT, "Acceptable Min. Solder Masking
   Thickness in 0.01mm." );
MakeSlot( CustSMStd:PadClear );
SetSlotOption( CustSMStd:PadClear, VALUE_TYPE, NUMBER );
CustSMStd:PadClear = 5;
SetSlotOption( CustSMStd:PadClear, PROMPT, "Acceptable Solder Masking Pad
Minimum Clearance in 0.01mm." );

    /*************************************
     **** CLASS: CustEtchStd
     *************************************/
MakeClass( CustEtchStd, CustGenSpec );
MakeSlot( CustEtchStd:UCutRatio );
SetSlotOption( CustEtchStd:UCutRatio, VALUE_TYPE, NUMBER );
CustEtchStd:UCutRatio = 2.5;
SetSlotOption( CustEtchStd:UCutRatio, PROMPT, "Acceptable Maximum Under Cut in 0.01 mm." );
MakeSlot( CustEtchStd:DevOrig );
SetSlotOption( CustEtchStd:DevOrig, VALUE_TYPE, NUMBER );
CustEtchStd:DevOrig = 20;
SetSlotOption( CustEtchStd:DevOrig, PROMPT, "Width of Etched Circuit Deviation from
 the Original Artwork Width (in %)." );

    /*************************************
     **** CLASS: SnPb
     *************************************/
MakeClass( SnPb, CustAppSurFin );
SnPb:Min = 0.3;
SetSlotOption( SnPb:Min, PROMPT, "Acceptable Min. Plated Sn/Pb Surface
  Finishing Thickness in 0.001in." );

    /*************************************
     **** CLASS: Ni
     *************************************/
MakeClass( Ni, CustAppSurFin );
Ni:Min = 0.2;
SetSlotOption( Ni:Min, PROMPT, "Acceptable Min. Plated Ni Surface
Finishing Thickness in 0.001in." );

    /*************************************
     **** CLASS: Au
     *************************************/
MakeClass( Au, CustAppSurFin );
Au:Min = 0.02;
SetSlotOption( Au:Min, PROMPT, "Acceptable Min. Plated Gold Surface
Finishing Thickness in 0.001in." );
```

Appendix 3.1 KAL Representation of the CircuitA Requirements

```
    /*************************************
     **** CLASS: ReqDocInfo
     *************************************/
MakeClass( ReqDocInfo, CustProdReqt );

/************** METHOD: CheckCustGenSpec  **************/
MakeMethod( ReqDocInfo, CheckCustGenSpec, [],TRUE );
MakeSlot( ReqDocInfo:CustName );
SetSlotOption( ReqDocInfo:CustName, ALLOWABLE_VALUES, CustomerA, CustomerB, CustomerC );
ReqDocInfo:CustName = CustomerA;
SetSlotOption( ReqDocInfo:CustName, PROMPT, "Name of Customer." );
MakeSlot( ReqDocInfo:ProdName );
ReqDocInfo:ProdName = CircuitA;
SetSlotOption( ReqDocInfo:ProdName, PROMPT, "Name of the Product." );
MakeSlot( ReqDocInfo:PNInHouse );
ReqDocInfo:PNInHouse = p8001;
SetSlotOption( ReqDocInfo:PNInHouse, PROMPT, "Assigned In-House Part Number for this Product." );
MakeSlot( ReqDocInfo:PNCust );
ReqDocInfo:PNCust = A001;
SetSlotOption( ReqDocInfo:PNCust, PROMPT, "Customer Product Part Number." );
MakeSlot( ReqDocInfo:PurOrdNo );
ReqDocInfo:PurOrdNo = PO001;
SetSlotOption( ReqDocInfo:PurOrdNo, PROMPT, "Purchase Order Number." );

    /*************************************
     **** CLASS: ProPlatingReq
     *************************************/
MakeClass( ProPlatingReq, CustProdReqt );

/************** METHOD: CheckCustGenSpec  **************/
MakeMethod( ProPlatingReq, CheckCustGenSpec, [],
{
  ForwardChain( [NOASSERT], NULL, CheckGenSpecPlatReq );
} );
MakeSlot( ProPlatingReq:SurfaceFinType );
SetSlotOption( ProPlatingReq:SurfaceFinType, ALLOWABLE_VALUES, Ni, Au, BareCopper, SnPb );
ProPlatingReq:SurfaceFinType = SnPb;
SetSlotOption( ProPlatingReq:SurfaceFinType, PROMPT, "Type of Surface Finishing." );
MakeSlot( ProPlatingReq:SurfaceFinThick );
SetSlotOption( ProPlatingReq:SurfaceFinThick, VALUE_TYPE, NUMBER );
ProPlatingReq:SurfaceFinThick = 0.5;
SetSlotOption( ProPlatingReq:SurfaceFinThick, PROMPT, "Plated Metal Min. Surface Thickness (other
than copper).  (0.001in)" );
MakeSlot( ProPlatingReq:CuThickMin );
SetSlotOption( ProPlatingReq:CuThickMin, VALUE_TYPE, NUMBER );
ProPlatingReq:CuThickMin = 0;
SetSlotOption( ProPlatingReq:CuThickMin, PROMPT, "Plated Copper Min. Thickness.  (0.001in)" );
```

```
    /************************************
     **** CLASS: ProSoldMask
     ************************************/
MakeClass( ProSoldMask, CustProdReqt );

/************** METHOD: CheckCustGenSpec  **************/
MakeMethod( ProSoldMask, CheckCustGenSpec, [],
{
  ForwardChain( [NOASSERT], NULL, CheckGenSpecSoldMask );
} );
MakeSlot( ProSoldMask:MaterialType );
ProSoldMask:MaterialType = PC501;
SetSlotOption( ProSoldMask:MaterialType, PROMPT, "Type of Solder Masking Material." );
SetSlotOption( ProSoldMask:MaterialType, IMAGE, StateBox1 );
MakeSlot( ProSoldMask:MinThick );
SetSlotOption( ProSoldMask:MinThick, VALUE_TYPE, NUMBER );
ProSoldMask:MinThick = 1;
SetSlotOption( ProSoldMask:MinThick, PROMPT, "Acceptable Min. Solder Masking Thickness.
(0.01mm)" );
MakeSlot( ProSoldMask:MinClear );
SetSlotOption( ProSoldMask:MinClear, VALUE_TYPE, NUMBER );
ProSoldMask:MinClear = 20;
SetSlotOption( ProSoldMask:MinClear, PROMPT, "Acceptable Min. Solder Masking Pad Clearance.
(0.01mm)" );
MakeSlot( ProSoldMask:Colour );
ProSoldMask:Colour = Green;
SetSlotOption( ProSoldMask:Colour, PROMPT, "Colour of Solder Masking Material." );

    /************************************
     **** CLASS: ProPanelDimen
     ************************************/
MakeClass( ProPanelDimen, CustProdReqt );

/************** METHOD: CheckCustGenSpec  **************/
MakeMethod( ProPanelDimen, CheckCustGenSpec, [], TRUE );
MakeSlot( ProPanelDimen:Width );
SetSlotOption( ProPanelDimen:Width, VALUE_TYPE, NUMBER );
ProPanelDimen:Width = 330;
SetSlotOption( ProPanelDimen:Width, PROMPT, "Board Panel Width. (in mm)" );
MakeSlot( ProPanelDimen:Length );
SetSlotOption( ProPanelDimen:Length, VALUE_TYPE, NUMBER );
ProPanelDimen:Length = 278;
SetSlotOption( ProPanelDimen:Length, PROMPT, "Board Panel Length. (in mm)" );
MakeSlot( ProPanelDimen:DimenDev );
SetSlotOption( ProPanelDimen:DimenDev, VALUE_TYPE, NUMBER );
ProPanelDimen:DimenDev = 15;
SetSlotOption( ProPanelDimen:DimenDev, PROMPT, "Acceptable Deviation in Outline Dimension.
(0.01mm)" );
MakeSlot( ProPanelDimen:UnitLength );
SetSlotOption( ProPanelDimen:UnitLength, VALUE_TYPE, NUMBER );
ProPanelDimen:UnitLength = 124;
```

```
SetSlotOption( ProPanelDimen:UnitLength, PROMPT, "Unit Length (in mm)." );
MakeSlot( ProPanelDimen:UnitWidth );
ProPanelDimen:UnitWidth = 150;
SetSlotOption( ProPanelDimen:UnitWidth, PROMPT, "Unit Width (in mm)." );
MakeSlot( ProPanelDimen:UnitPerPanel );
SetSlotOption( ProPanelDimen:UnitPerPanel, VALUE_TYPE, NUMBER );
ProPanelDimen:UnitPerPanel = 4;
SetSlotOption( ProPanelDimen:UnitPerPanel, PROMPT, "No. of Unit Per Panel" );
MakeSlot( ProPanelDimen:EstDieCost );
SetSlotOption( ProPanelDimen:EstDieCost, VALUE_TYPE, NUMBER );
ProPanelDimen:EstDieCost = 5000;
SetSlotOption( ProPanelDimen:EstDieCost, PROMPT, "Estimated Blanking Die Cost. (HK$)" );

    /*************************************
     **** CLASS: ProLamChar
     *************************************/
MakeClass( ProLamChar, CustProdReqt );

/************** METHOD: CheckCustGenSpec  **************/
MakeMethod( ProLamChar, CheckCustGenSpec, [], TRUE );
MakeSlot( ProLamChar:LaminateType );
SetSlotOption( ProLamChar:LaminateType, ALLOWABLE_VALUES, CEM2, CEM3, FR3, FR4 );
ProLamChar:LaminateType = FR4;
SetSlotOption( ProLamChar:LaminateType, PROMPT, "Type of Laminate." );
MakeSlot( ProLamChar:CuFoilThickness );
SetSlotOption( ProLamChar:CuFoilThickness, ALLOWABLE_VALUES, 0.5, 1, 1.5, 2, 2.5 );
ProLamChar:CuFoilThickness = 1;
SetSlotOption( ProLamChar:CuFoilThickness, PROMPT, "Copper Foil Thickness. (oz.)" );
MakeSlot( ProLamChar:CuFoilSide );
SetSlotOption( ProLamChar:CuFoilSide, ALLOWABLE_VALUES, Single, Double );
ProLamChar:CuFoilSide = Single;
SetSlotOption( ProLamChar:CuFoilSide, PROMPT, "Side of Copper Foil." );
MakeSlot( ProLamChar:LamThickness );
SetSlotOption( ProLamChar:LamThickness, ALLOWABLE_VALUES, 0.8, 1.2, 1.6, 1.8, 3.2 );
ProLamChar:LamThickness = 1.6;
SetSlotOption( ProLamChar:LamThickness, PROMPT, "Laminate Thickness. (mm)" );
MakeSlot( ProLamChar:Colour );
SetSlotOption( ProLamChar:Colour, ALLOWABLE_VALUES, Green, Red, Blue );
ProLamChar:Colour = Green;
SetSlotOption( ProLamChar:Colour, PROMPT, "Colour of Laminate." );

    /*************************************
     **** CLASS: ProAuFinger
     *************************************/
MakeClass( ProAuFinger, CustProdReqt );

/************** METHOD: CheckCustGenSpec  **************/
MakeMethod( ProAuFinger, CheckCustGenSpec, [],
{
  ForwardChain( [NOASSERT], NULL, CheckGenSpecAuFing );
} );
```

```
MakeSlot( ProAuFinger:AuThick );
SetSlotOption( ProAuFinger:AuThick, VALUE_TYPE, NUMBER );
ProAuFinger:AuThick = 0;
SetSlotOption( ProAuFinger:AuThick, PROMPT, "Finger Gold Thickness. (0.001in)" );
MakeSlot( ProAuFinger:NiThick );
SetSlotOption( ProAuFinger:NiThick, VALUE_TYPE, NUMBER );
ProAuFinger:NiThick = 0;
SetSlotOption( ProAuFinger:NiThick, PROMPT, "Finger Nickel Thickness. (0.001in)" );
MakeSlot( ProAuFinger:Height );
SetSlotOption( ProAuFinger:Height, VALUE_TYPE, NUMBER );
ProAuFinger:Height = 0;
SetSlotOption( ProAuFinger:Height, PROMPT, "Height of Gole Finger. (mm)" );
MakeSlot( ProAuFinger:UnitPerEdge );
SetSlotOption( ProAuFinger:UnitPerEdge, VALUE_TYPE, NUMBER );
ProAuFinger:UnitPerEdge = 0;
SetSlotOption( ProAuFinger:UnitPerEdge, PROMPT, "No. of Unit Per Plating Edge" );
MakeSlot( ProAuFinger:PlatAreaPerUnit );
SetSlotOption( ProAuFinger:PlatAreaPerUnit, VALUE_TYPE, NUMBER );
ProAuFinger:PlatAreaPerUnit = 0;
SetSlotOption( ProAuFinger:PlatAreaPerUnit, PROMPT, "Gold Finger Plating Area Per Unit" );

    /*************************************
     **** CLASS: ProFeatCircuit
     *************************************/
MakeClass( ProFeatCircuit, CustProdReqt );

/************** METHOD: CheckCustGenSpec  **************/
MakeMethod( ProFeatCircuit, CheckCustGenSpec, [],
{
  ForwardChain( [NOASSERT], NULL, CheckGenSpecFeatCir );
} );
MakeSlot( ProFeatCircuit:CirWidth );
SetSlotOption( ProFeatCircuit:CirWidth, VALUE_TYPE, NUMBER );
SetSlotOption( ProFeatCircuit:CirWidth, MINIMUM_VALUE, 0 );
ProFeatCircuit:CirWidth = 70;
SetSlotOption( ProFeatCircuit:CirWidth, PROMPT, "Min. Circuit Width.  (0.01mm)" );
MakeSlot( ProFeatCircuit:SpaceWidth );
SetSlotOption( ProFeatCircuit:SpaceWidth, VALUE_TYPE, NUMBER );
SetSlotOption( ProFeatCircuit:SpaceWidth, MINIMUM_VALUE, 0 );
ProFeatCircuit:SpaceWidth = 350;
SetSlotOption( ProFeatCircuit:SpaceWidth, PROMPT, "Min. Circuit Spacing.  (0.01mm)" );
MakeSlot( ProFeatCircuit:CirThickness );
SetSlotOption( ProFeatCircuit:CirThickness, VALUE_TYPE, NUMBER );
ProFeatCircuit:CirThickness = 4;
SetSlotOption( ProFeatCircuit:CirThickness, PROMPT, "Min. Circuit Thickness.  (0.01mm)" );
MakeSlot( ProFeatCircuit:AnnularRing );
SetSlotOption( ProFeatCircuit:AnnularRing, VALUE_TYPE, NUMBER );
ProFeatCircuit:AnnularRing = 25;
SetSlotOption( ProFeatCircuit:AnnularRing, PROMPT, "Min. Annular Ring.  (0.01mm)" );
MakeSlot( ProFeatCircuit:CircuitArea );
SetSlotOption( ProFeatCircuit:CircuitArea, VALUE_TYPE, NUMBER );
```

```
ProFeatCircuit:CircuitArea = 1.32;
SetSlotOption( ProFeatCircuit:CircuitArea, PROMPT, "Unit Circuit area in sq. in." );

    /*************************************
     **** CLASS: ProDocument
     *************************************/
MakeClass( ProDocument, CustProdReqt );

/************** METHOD: CheckCustGenSpec  **************/
MakeMethod( ProDocument, CheckCustGenSpec, [], TRUE );
MakeSlot( ProDocument:FilmNo );
ProDocument:FilmNo = PCB8;
SetSlotOption( ProDocument:FilmNo, PROMPT, "Film File Number." );
MakeSlot( ProDocument:DrawNo );
ProDocument:DrawNo = DN001;
SetSlotOption( ProDocument:DrawNo, PROMPT, "Drawing Number." );
MakeSlot( ProDocument:SpecNo );
ProDocument:SpecNo = 1001;
SetSlotOption( ProDocument:SpecNo, PROMPT, "Specification Number." );
MakeSlot( ProDocument:DrillFileNo );
ProDocument:DrillFileNo = -;
SetSlotOption( ProDocument:DrillFileNo, PROMPT, "Drill File Number." );

    /*************************************
     **** CLASS: ProProdDetails
     *************************************/
MakeClass( ProProdDetails, CustProdReqt );

/************** METHOD: CheckCustGenSpec  **************/
MakeMethod( ProProdDetails, CheckCustGenSpec, [],
  TRUE );
MakeSlot( ProProdDetails:ExpQty );
SetSlotOption( ProProdDetails:ExpQty, VALUE_TYPE, NUMBER );
ProProdDetails:ExpQty = 20000;
SetSlotOption( ProProdDetails:ExpQty, PROMPT, "Total Expected Production Quantity." );
MakeSlot( ProProdDetails:BatchSize );
SetSlotOption( ProProdDetails:BatchSize, VALUE_TYPE, NUMBER );
ProProdDetails:BatchSize = 5000;
SetSlotOption( ProProdDetails:BatchSize, PROMPT, "This Production Batch Size." );
MakeSlot( ProProdDetails:WkDateAvail );
SetSlotOption( ProProdDetails:WkDateAvail, VALUE_TYPE, NUMBER );
ProProdDetails:WkDateAvail = 30;
SetSlotOption( ProProdDetails:WkDateAvail, PROMPT, "Working Day Available for this Batch." );

    /*************************************
     **** CLASS: ProHoleSpec
     *************************************/
MakeClass( ProHoleSpec, CustProdReqt );

/************** METHOD: CheckCustGenSpec  **************/
MakeMethod( ProHoleSpec, CheckCustGenSpec, [], TRUE );
```

```
MakeSlot( ProHoleSpec:LocDev );
SetSlotOption( ProHoleSpec:LocDev, VALUE_TYPE, NUMBER );
ProHoleSpec:LocDev = 7.5;
SetSlotOption( ProHoleSpec:LocDev, PROMPT, "Deviation in Hole Location. (0.01mm)" );
MakeSlot( ProHoleSpec:DiaDev );
SetSlotOption( ProHoleSpec:DiaDev, VALUE_TYPE, NUMBER );
ProHoleSpec:DiaDev = 7.5;
SetSlotOption( ProHoleSpec:DiaDev, PROMPT, "Deviation in Hole Diameter. (0.01mm)" );
MakeSlot( ProHoleSpec:MinHoleSize );
SetSlotOption( ProHoleSpec:MinHoleSize, VALUE_TYPE, NUMBER );
SetSlotOption( ProHoleSpec:MinHoleSize, MINIMUM_VALUE, 0.4 );
SetSlotOption( ProHoleSpec:MinHoleSize, MAXIMUM_VALUE, 5 );
ProHoleSpec:MinHoleSize = 0.8;
SetSlotOption( ProHoleSpec:MinHoleSize, PROMPT, "Min. Hole Size. (0.01mm)" );
MakeSlot( ProHoleSpec:MaxHoleSize );
SetSlotOption( ProHoleSpec:MaxHoleSize, ALLOWABLE_VALUES, 0.4, 5 );
ProHoleSpec:MaxHoleSize = 2.5;
SetSlotOption( ProHoleSpec:MaxHoleSize, PROMPT, "Max. Hole Size." );

    /*************************************
     **** CLASS: ProCompMark
     *************************************/
MakeClass( ProCompMark, CustProdReqt );

/************** METHOD: CheckCustGenSpec  **************/
MakeMethod( ProCompMark, CheckCustGenSpec, [], TRUE );
MakeSlot( ProCompMark:CompMaterial );
ProCompMark:CompMaterial = materialA;
SetSlotOption( ProCompMark:CompMaterial, PROMPT, "Marking Material." );
MakeSlot( ProCompMark:CompColour );
ProCompMark:CompColour = white;
SetSlotOption( ProCompMark:CompColour, PROMPT, "Marking Colour." );
```

Appendix 3.2 KAL Representation of the CircuitB Requirements

```
    /*************************************
     **** CLASS: ReqDocInfo
     *************************************/
MakeClass( ReqDocInfo, CustProdReqt );

/************** METHOD: CheckCustGenSpec  **************/
MakeMethod( ReqDocInfo, CheckCustGenSpec, [],
TRUE );
MakeSlot( ReqDocInfo:CustName );
SetSlotOption( ReqDocInfo:CustName, ALLOWABLE_VALUES, CustomerA, CustomerB, CustomerC );
ReqDocInfo:CustName = CustomerA;
SetSlotOption( ReqDocInfo:CustName, PROMPT, "Name of Customer." );
MakeSlot( ReqDocInfo:ProdName );
ReqDocInfo:ProdName = CircuitB;
SetSlotOption( ReqDocInfo:ProdName, PROMPT, "Name of the Product." );
MakeSlot( ReqDocInfo:PNInHouse );
ReqDocInfo:PNInHouse = p8002;
SetSlotOption( ReqDocInfo:PNInHouse, PROMPT, "Assigned In-House Part Number for this Product." );
MakeSlot( ReqDocInfo:PNCust );
ReqDocInfo:PNCust = A002;
SetSlotOption( ReqDocInfo:PNCust, PROMPT, "Customer Product Part Number." );
MakeSlot( ReqDocInfo:PurOrdNo );
ReqDocInfo:PurOrdNo = PO002;
SetSlotOption( ReqDocInfo:PurOrdNo, PROMPT, "Purchase Order Number." );

    /*************************************
     **** CLASS: ProPlatingReq
     *************************************/
MakeClass( ProPlatingReq, CustProdReqt );

/************** METHOD: CheckCustGenSpec  **************/
MakeMethod( ProPlatingReq, CheckCustGenSpec, [],
{
  ForwardChain( [NOASSERT], NULL, CheckGenSpecPlatReq );
} );
MakeSlot( ProPlatingReq:SurfaceFinType );
SetSlotOption( ProPlatingReq:SurfaceFinType, ALLOWABLE_VALUES, Ni, Au, BareCopper, SnPb );
ProPlatingReq:SurfaceFinType = SnPb;
SetSlotOption( ProPlatingReq:SurfaceFinType, PROMPT, "Type of Surface Finishing." );
MakeSlot( ProPlatingReq:SurfaceFinThick );
SetSlotOption( ProPlatingReq:SurfaceFinThick, VALUE_TYPE, NUMBER );
ProPlatingReq:SurfaceFinThick = 0.5;
SetSlotOption( ProPlatingReq:SurfaceFinThick, PROMPT, "Plated Metal Min. Surface Thickness (other
than copper).  (0.001in)" );
MakeSlot( ProPlatingReq:CuThickMin );
SetSlotOption( ProPlatingReq:CuThickMin, VALUE_TYPE, NUMBER );
ProPlatingReq:CuThickMin = 1.5;
SetSlotOption( ProPlatingReq:CuThickMin, PROMPT, "Plated Copper Min. Thickness.  (0.001in)" );
```

```
    /*************************************
     **** CLASS: ProSoldMask
     *************************************/
MakeClass( ProSoldMask, CustProdReqt );

/************** METHOD: CheckCustGenSpec  **************/
MakeMethod( ProSoldMask, CheckCustGenSpec, [],
{
  ForwardChain( [NOASSERT], NULL, CheckGenSpecSoldMask );
} );
MakeSlot( ProSoldMask:MaterialType );
ProSoldMask:MaterialType = PC501;
SetSlotOption( ProSoldMask:MaterialType, PROMPT, "Type of Solder Masking Material." );
SetSlotOption( ProSoldMask:MaterialType, IMAGE, StateBox1 );
MakeSlot( ProSoldMask:MinThick );
SetSlotOption( ProSoldMask:MinThick, VALUE_TYPE, NUMBER );
ProSoldMask:MinThick = 1;
SetSlotOption( ProSoldMask:MinThick, PROMPT, "Acceptable Min. Solder Masking Thickness.
(0.01mm)" );
MakeSlot( ProSoldMask:MinClear );
SetSlotOption( ProSoldMask:MinClear, VALUE_TYPE, NUMBER );
ProSoldMask:MinClear = 50;
SetSlotOption( ProSoldMask:MinClear, PROMPT, "Acceptable Min. Solder Masking Pad Clearance.
(0.01mm)" );
MakeSlot( ProSoldMask:Colour );
ProSoldMask:Colour = Green;
SetSlotOption( ProSoldMask:Colour, PROMPT, "Colour of Solder Masking Material." );

    /*************************************
     **** CLASS: ProPanelDimen
     *************************************/
MakeClass( ProPanelDimen, CustProdReqt );

/************** METHOD: CheckCustGenSpec  **************/
MakeMethod( ProPanelDimen, CheckCustGenSpec, [],   TRUE );
MakeSlot( ProPanelDimen:Width );
SetSlotOption( ProPanelDimen:Width, VALUE_TYPE, NUMBER );
ProPanelDimen:Width = 242;
SetSlotOption( ProPanelDimen:Width, PROMPT, "Board Panel Width. (in mm)" );
MakeSlot( ProPanelDimen:Length );
SetSlotOption( ProPanelDimen:Length, VALUE_TYPE, NUMBER );
ProPanelDimen:Length = 230;
SetSlotOption( ProPanelDimen:Length, PROMPT, "Board Panel Length. (in mm)" );
MakeSlot( ProPanelDimen:DimenDev );
SetSlotOption( ProPanelDimen:DimenDev, VALUE_TYPE, NUMBER );
ProPanelDimen:DimenDev = 15;
SetSlotOption( ProPanelDimen:DimenDev, PROMPT, "Acceptable Deviation in Outline Dimension.
(0.01mm)" );
MakeSlot( ProPanelDimen:UnitLength );
SetSlotOption( ProPanelDimen:UnitLength, VALUE_TYPE, NUMBER );
```

```
ProPanelDimen:UnitLength = 100;
SetSlotOption( ProPanelDimen:UnitLength, PROMPT, "Unit Length (in mm)." );
MakeSlot( ProPanelDimen:UnitWidth );
ProPanelDimen:UnitWidth = 106;
SetSlotOption( ProPanelDimen:UnitWidth, PROMPT, "Unit Width (in mm)." );
MakeSlot( ProPanelDimen:UnitPerPanel );
SetSlotOption( ProPanelDimen:UnitPerPanel, VALUE_TYPE, NUMBER );
ProPanelDimen:UnitPerPanel = 4;
SetSlotOption( ProPanelDimen:UnitPerPanel, PROMPT, "No. of Unit Per Panel" );
MakeSlot( ProPanelDimen:EstDieCost );
SetSlotOption( ProPanelDimen:EstDieCost, VALUE_TYPE, NUMBER );
ProPanelDimen:EstDieCost = 9000;
SetSlotOption( ProPanelDimen:EstDieCost, PROMPT, "Estimated Blanking Die Cost. (HK$)" );

    /************************************
     **** CLASS: ProLamChar
     ************************************/
MakeClass( ProLamChar, CustProdReqt );

/************** METHOD: CheckCustGenSpec  **************/
MakeMethod( ProLamChar, CheckCustGenSpec, [],   TRUE );
MakeSlot( ProLamChar:LaminateType );
SetSlotOption( ProLamChar:LaminateType, ALLOWABLE_VALUES, CEM2, CEM3, FR3, FR4 );
ProLamChar:LaminateType = FR4;
SetSlotOption( ProLamChar:LaminateType, PROMPT, "Type of Laminate." );
MakeSlot( ProLamChar:CuFoilThickness );
SetSlotOption( ProLamChar:CuFoilThickness, ALLOWABLE_VALUES, 0.5, 1, 1.5, 2, 2.5 );
ProLamChar:CuFoilThickness = 1;
SetSlotOption( ProLamChar:CuFoilThickness, PROMPT, "Copper Foil Thickness. (oz.)" );
MakeSlot( ProLamChar:CuFoilSide );
SetSlotOption( ProLamChar:CuFoilSide, ALLOWABLE_VALUES, Single, Double );
ProLamChar:CuFoilSide = Double;
SetSlotOption( ProLamChar:CuFoilSide, PROMPT, "Side of Copper Foil." );
MakeSlot( ProLamChar:LamThickness );
SetSlotOption( ProLamChar:LamThickness, ALLOWABLE_VALUES, 0.8, 1.2, 1.6, 1.8, 3.2 );
ProLamChar:LamThickness = 1.6;
SetSlotOption( ProLamChar:LamThickness, PROMPT, "Laminate Thickness." );
MakeSlot( ProLamChar:Colour );
SetSlotOption( ProLamChar:Colour, ALLOWABLE_VALUES, Green, Red, Blue );
ProLamChar:Colour = Green;
SetSlotOption( ProLamChar:Colour, PROMPT, "Colour of Laminate." );

    /************************************
     **** CLASS: ProAuFinger
     ************************************/
MakeClass( ProAuFinger, CustProdReqt );

/************** METHOD: CheckCustGenSpec  **************/
MakeMethod( ProAuFinger, CheckCustGenSpec, [],
{
  ForwardChain( [NOASSERT], NULL, CheckGenSpecAuFing );
```

```
} );
MakeSlot( ProAuFinger:AuThick );
SetSlotOption( ProAuFinger:AuThick, VALUE_TYPE, NUMBER );
ProAuFinger:AuThick = 0;
SetSlotOption( ProAuFinger:AuThick, PROMPT, "Finger Gold Thickness. (0.001in)" );
MakeSlot( ProAuFinger:NiThick );
SetSlotOption( ProAuFinger:NiThick, VALUE_TYPE, NUMBER );
ProAuFinger:NiThick = 0;
SetSlotOption( ProAuFinger:NiThick, PROMPT, "Finger Nickel Thickness. (0.001in)" );
MakeSlot( ProAuFinger:Height );
SetSlotOption( ProAuFinger:Height, VALUE_TYPE, NUMBER );
ProAuFinger:Height = 0;
SetSlotOption( ProAuFinger:Height, PROMPT, "Height of Gole Finger. (mm)" );
MakeSlot( ProAuFinger:UnitPerEdge );
SetSlotOption( ProAuFinger:UnitPerEdge, VALUE_TYPE, NUMBER );
ProAuFinger:UnitPerEdge = 0;
SetSlotOption( ProAuFinger:UnitPerEdge, PROMPT, "No. of Unit Per Plating Edge" );
MakeSlot( ProAuFinger:PlatAreaPerUnit );
SetSlotOption( ProAuFinger:PlatAreaPerUnit, VALUE_TYPE, NUMBER );
ProAuFinger:PlatAreaPerUnit = 0;
SetSlotOption( ProAuFinger:PlatAreaPerUnit, PROMPT, "Gold Finger Plating Area Per Unit" );

    /*************************************
     **** CLASS: ProFeatCircuit
     *************************************/
MakeClass( ProFeatCircuit, CustProdReqt );

/************** METHOD: CheckCustGenSpec  **************/
MakeMethod( ProFeatCircuit, CheckCustGenSpec, [],
{
  ForwardChain( [NOASSERT], NULL, CheckGenSpecFeatCir );
} );
MakeSlot( ProFeatCircuit:CirWidth );
SetSlotOption( ProFeatCircuit:CirWidth, VALUE_TYPE, NUMBER );
SetSlotOption( ProFeatCircuit:CirWidth, MINIMUM_VALUE, 0 );
ProFeatCircuit:CirWidth = 50;
SetSlotOption( ProFeatCircuit:CirWidth, PROMPT, "Min. Circuit Width.  (0.01mm)" );
MakeSlot( ProFeatCircuit:SpaceWidth );
SetSlotOption( ProFeatCircuit:SpaceWidth, VALUE_TYPE, NUMBER );
SetSlotOption( ProFeatCircuit:SpaceWidth, MINIMUM_VALUE, 0 );
ProFeatCircuit:SpaceWidth = 50;
SetSlotOption( ProFeatCircuit:SpaceWidth, PROMPT, "Min. Circuit Spacing.  (0.01mm)" );
MakeSlot( ProFeatCircuit:CirThickness );
SetSlotOption( ProFeatCircuit:CirThickness, VALUE_TYPE, NUMBER );
ProFeatCircuit:CirThickness = 4;
SetSlotOption( ProFeatCircuit:CirThickness, PROMPT, "Min. Circuit Thickness.  (0.01mm)" );
MakeSlot( ProFeatCircuit:AnnularRing );
SetSlotOption( ProFeatCircuit:AnnularRing, VALUE_TYPE, NUMBER );
ProFeatCircuit:AnnularRing = 25;
SetSlotOption( ProFeatCircuit:AnnularRing, PROMPT, "Min. Annular Ring.  (0.01mm)" );
MakeSlot( ProFeatCircuit:CircuitArea );
```

```
SetSlotOption( ProFeatCircuit:CircuitArea, VALUE_TYPE, NUMBER );
ProFeatCircuit:CircuitArea = 3.24;
SetSlotOption( ProFeatCircuit:CircuitArea, PROMPT, "Unit Circuit area in sq. in." );

    /*************************************
     **** CLASS: ProDocument
     *************************************/
MakeClass( ProDocument, CustProdReqt );

/************** METHOD: CheckCustGenSpec  **************/
MakeMethod( ProDocument, CheckCustGenSpec, [],   TRUE );
MakeSlot( ProDocument:FilmNo );
ProDocument:FilmNo = PCB;
SetSlotOption( ProDocument:FilmNo, PROMPT, "Film File Number." );
MakeSlot( ProDocument:DrawNo );
ProDocument:DrawNo = DN002;
SetSlotOption( ProDocument:DrawNo, PROMPT, "Drawing Number." );
MakeSlot( ProDocument:SpecNo );
ProDocument:SpecNo = 1001;
SetSlotOption( ProDocument:SpecNo, PROMPT, "Specification Number." );
MakeSlot( ProDocument:DrillFileNo );
ProDocument:DrillFileNo = -;
SetSlotOption( ProDocument:DrillFileNo, PROMPT, "Drill File Number." );

    /*************************************
     **** CLASS: ProProdDetails
     *************************************/
MakeClass( ProProdDetails, CustProdReqt );

/************** METHOD: CheckCustGenSpec  **************/
MakeMethod( ProProdDetails, CheckCustGenSpec, [],   TRUE );
MakeSlot( ProProdDetails:ExpQty );
SetSlotOption( ProProdDetails:ExpQty, VALUE_TYPE, NUMBER );
ProProdDetails:ExpQty = 20000;
SetSlotOption( ProProdDetails:ExpQty, PROMPT, "Total Expected Production Quantity." );
MakeSlot( ProProdDetails:BatchSize );
SetSlotOption( ProProdDetails:BatchSize, VALUE_TYPE, NUMBER );
ProProdDetails:BatchSize = 5000;
SetSlotOption( ProProdDetails:BatchSize, PROMPT, "This Production Batch Size." );
MakeSlot( ProProdDetails:WkDateAvail );
SetSlotOption( ProProdDetails:WkDateAvail, VALUE_TYPE, NUMBER );
ProProdDetails:WkDateAvail = 30;
SetSlotOption( ProProdDetails:WkDateAvail, PROMPT, "Working Day Available for this Batch." );

    /*************************************
     **** CLASS: ProHoleSpec
     *************************************/
MakeClass( ProHoleSpec, CustProdReqt );

/************** METHOD: CheckCustGenSpec  **************/
MakeMethod( ProHoleSpec, CheckCustGenSpec, [],   TRUE );
```

```
MakeSlot( ProHoleSpec:LocDev );
SetSlotOption( ProHoleSpec:LocDev, VALUE_TYPE, NUMBER );
ProHoleSpec:LocDev = 7.5;
SetSlotOption( ProHoleSpec:LocDev, PROMPT, "Deviation in Hole Location. (0.01mm)" );
MakeSlot( ProHoleSpec:DiaDev );
SetSlotOption( ProHoleSpec:DiaDev, VALUE_TYPE, NUMBER );
ProHoleSpec:DiaDev = 7.5;
SetSlotOption( ProHoleSpec:DiaDev, PROMPT, "Deviation in Hole Diameter. (0.01mm)" );
MakeSlot( ProHoleSpec:MinHoleSize );
SetSlotOption( ProHoleSpec:MinHoleSize, VALUE_TYPE, NUMBER );
SetSlotOption( ProHoleSpec:MinHoleSize, MINIMUM_VALUE, 0.4 );
SetSlotOption( ProHoleSpec:MinHoleSize, MAXIMUM_VALUE, 5 );
ProHoleSpec:MinHoleSize = 0.8;
SetSlotOption( ProHoleSpec:MinHoleSize, PROMPT, "Min. Hole Size. (0.01mm)" );
MakeSlot( ProHoleSpec:MaxHoleSize );
SetSlotOption( ProHoleSpec:MaxHoleSize, ALLOWABLE_VALUES, 0.4, 5 );
ProHoleSpec:MaxHoleSize = 2.5;
SetSlotOption( ProHoleSpec:MaxHoleSize, PROMPT, "Max. Hole Size." );

    /*************************************
     **** CLASS: ProCompMark
     *************************************/
MakeClass( ProCompMark, CustProdReqt );

/************** METHOD: CheckCustGenSpec  **************/
MakeMethod( ProCompMark, CheckCustGenSpec, [],   TRUE );
MakeSlot( ProCompMark:CompMaterial );
ProCompMark:CompMaterial = materialA;
SetSlotOption( ProCompMark:CompMaterial, PROMPT, "Marking Material." );
MakeSlot( ProCompMark:CompColour );
ProCompMark:CompColour = white;
SetSlotOption( ProCompMark:CompColour, PROMPT, "Marking Colour." );
```

Appendix 3.3 KAL Representation of the CircuitC Requirements

```
    /************************************
     **** CLASS: ReqDocInfo
     ************************************/
MakeClass( ReqDocInfo, CustProdReqt );

/************** METHOD: CheckCustGenSpec  **************/
MakeMethod( ReqDocInfo, CheckCustGenSpec, [],
TRUE );
MakeSlot( ReqDocInfo:CustName );
SetSlotOption( ReqDocInfo:CustName, ALLOWABLE_VALUES, CustomerA, CustomerB, CustomerC );
ReqDocInfo:CustName = CustomerB;
SetSlotOption( ReqDocInfo:CustName, PROMPT, "Name of Customer." );
MakeSlot( ReqDocInfo:ProdName );
ReqDocInfo:ProdName = CircuitC;
SetSlotOption( ReqDocInfo:ProdName, PROMPT, "Name of the Product." );
MakeSlot( ReqDocInfo:PNInHouse );
ReqDocInfo:PNInHouse = p8003;
SetSlotOption( ReqDocInfo:PNInHouse, PROMPT, "Assigned In-House Part Number for this Product." );
MakeSlot( ReqDocInfo:PNCust );
ReqDocInfo:PNCust = B001;
SetSlotOption( ReqDocInfo:PNCust, PROMPT, "Customer Product Part Number." );
MakeSlot( ReqDocInfo:PurOrdNo );
ReqDocInfo:PurOrdNo = PO008;
SetSlotOption( ReqDocInfo:PurOrdNo, PROMPT, "Purchase Order Number." );

    /************************************
     **** CLASS: ProPlatingReq
     ************************************/
MakeClass( ProPlatingReq, CustProdReqt );

/************** METHOD: CheckCustGenSpec  **************/
MakeMethod( ProPlatingReq, CheckCustGenSpec, [],
{
  ForwardChain( [NOASSERT], NULL, CheckGenSpecPlatReq );
} );
MakeSlot( ProPlatingReq:SurfaceFinType );
SetSlotOption( ProPlatingReq:SurfaceFinType, ALLOWABLE_VALUES, Ni, Au, BareCopper, SnPb );
ProPlatingReq:SurfaceFinType = Au;
SetSlotOption( ProPlatingReq:SurfaceFinType, PROMPT, "Type of Surface Finishing." );
MakeSlot( ProPlatingReq:SurfaceFinThick );
SetSlotOption( ProPlatingReq:SurfaceFinThick, VALUE_TYPE, NUMBER );
ProPlatingReq:SurfaceFinThick = 0.02;
SetSlotOption( ProPlatingReq:SurfaceFinThick, PROMPT, "Plated Metal Min. Surface Thickness (other
than copper). (0.001in)" );
MakeSlot( ProPlatingReq:CuThickMin );
SetSlotOption( ProPlatingReq:CuThickMin, VALUE_TYPE, NUMBER );
ProPlatingReq:CuThickMin = 1.5;
SetSlotOption( ProPlatingReq:CuThickMin, PROMPT, "Plated Copper Min. Thickness. (0.001in)" );
```

```
    /************************************
     **** CLASS: ProSoldMask
     ************************************/
MakeClass( ProSoldMask, CustProdReqt );

/************** METHOD: CheckCustGenSpec  **************/
MakeMethod( ProSoldMask, CheckCustGenSpec, [],
{
  ForwardChain( [NOASSERT], NULL, CheckGenSpecSoldMask );
} );
MakeSlot( ProSoldMask:MaterialType );
ProSoldMask:MaterialType = PC501;
SetSlotOption( ProSoldMask:MaterialType, PROMPT, "Type of Solder Masking Material." );
SetSlotOption( ProSoldMask:MaterialType, IMAGE, StateBox1 );
MakeSlot( ProSoldMask:MinThick );
SetSlotOption( ProSoldMask:MinThick, VALUE_TYPE, NUMBER );
ProSoldMask:MinThick = 1;
SetSlotOption( ProSoldMask:MinThick, PROMPT, "Acceptable Min. Solder Masking Thickness.
(0.01mm)" );
MakeSlot( ProSoldMask:MinClear );
SetSlotOption( ProSoldMask:MinClear, VALUE_TYPE, NUMBER );
ProSoldMask:MinClear = 5;
SetSlotOption( ProSoldMask:MinClear, PROMPT, "Acceptable Min. Solder Masking Pad Clearance.
(0.01mm)" );
MakeSlot( ProSoldMask:Colour );
ProSoldMask:Colour = Green;
SetSlotOption( ProSoldMask:Colour, PROMPT, "Colour of Solder Masking Material." );

    /************************************
     **** CLASS: ProPanelDimen
     ************************************/
MakeClass( ProPanelDimen, CustProdReqt );

/************** METHOD: CheckCustGenSpec  **************/
MakeMethod( ProPanelDimen, CheckCustGenSpec, [],
  TRUE );
MakeSlot( ProPanelDimen:Width );
SetSlotOption( ProPanelDimen:Width, VALUE_TYPE, NUMBER );
ProPanelDimen:Width = 230;
SetSlotOption( ProPanelDimen:Width, PROMPT, "Board Panel Width. (in mm)" );
MakeSlot( ProPanelDimen:Length );
SetSlotOption( ProPanelDimen:Length, VALUE_TYPE, NUMBER );
ProPanelDimen:Length = 230;
SetSlotOption( ProPanelDimen:Length, PROMPT, "Board Panel Length. (in mm)" );
MakeSlot( ProPanelDimen:DimenDev );
SetSlotOption( ProPanelDimen:DimenDev, VALUE_TYPE, NUMBER );
ProPanelDimen:DimenDev = 15;
SetSlotOption( ProPanelDimen:DimenDev, PROMPT, "Acceptable Deviation in Outline Dimension.
(0.01mm)" );
MakeSlot( ProPanelDimen:UnitLength );
```

```
SetSlotOption( ProPanelDimen:UnitLength, VALUE_TYPE, NUMBER );
ProPanelDimen:UnitLength = 100;
SetSlotOption( ProPanelDimen:UnitLength, PROMPT, "Unit Length (in mm)." );
MakeSlot( ProPanelDimen:UnitWidth );
ProPanelDimen:UnitWidth = 100;
SetSlotOption( ProPanelDimen:UnitWidth, PROMPT, "Unit Width (in mm)." );
MakeSlot( ProPanelDimen:UnitPerPanel );
SetSlotOption( ProPanelDimen:UnitPerPanel, VALUE_TYPE, NUMBER );
ProPanelDimen:UnitPerPanel = 4;
SetSlotOption( ProPanelDimen:UnitPerPanel, PROMPT, "No. of Unit Per Panel" );
MakeSlot( ProPanelDimen:EstDieCost );
SetSlotOption( ProPanelDimen:EstDieCost, VALUE_TYPE, NUMBER );
ProPanelDimen:EstDieCost = 6500;
SetSlotOption( ProPanelDimen:EstDieCost, PROMPT, "Estimated Blanking Die Cost. (HK$)" );

    /*************************************
     **** CLASS: ProLamChar
     *************************************/
MakeClass( ProLamChar, CustProdReqt );

/************** METHOD: CheckCustGenSpec  **************/
MakeMethod( ProLamChar, CheckCustGenSpec, [],
  TRUE );
MakeSlot( ProLamChar:LaminateType );
SetSlotOption( ProLamChar:LaminateType, ALLOWABLE_VALUES, CEM2, CEM3, FR3, FR4 );
ProLamChar:LaminateType = "FR-4";
SetSlotOption( ProLamChar:LaminateType, PROMPT, "Type of Laminate." );
MakeSlot( ProLamChar:CuFoilThickness );
SetSlotOption( ProLamChar:CuFoilThickness, ALLOWABLE_VALUES, 0.5, 1, 1.5, 2, 2.5 );
ProLamChar:CuFoilThickness = 1;
SetSlotOption( ProLamChar:CuFoilThickness, PROMPT, "Copper Foil Thickness. (oz.)" );
MakeSlot( ProLamChar:CuFoilSide );
SetSlotOption( ProLamChar:CuFoilSide, ALLOWABLE_VALUES, Single, Double );
ProLamChar:CuFoilSide = Double;
SetSlotOption( ProLamChar:CuFoilSide, PROMPT, "Side of Copper Foil." );
MakeSlot( ProLamChar:LamThickness );
SetSlotOption( ProLamChar:LamThickness, ALLOWABLE_VALUES, 0.8, 1.2, 1.6, 1.8, 3.2 );
ProLamChar:LamThickness = 1.6;
SetSlotOption( ProLamChar:LamThickness, PROMPT, "Laminate Thickness." );
MakeSlot( ProLamChar:Colour );
SetSlotOption( ProLamChar:Colour, ALLOWABLE_VALUES, Green, Red, Blue );
ProLamChar:Colour = Green;
SetSlotOption( ProLamChar:Colour, PROMPT, "Colour of Laminate." );

    /*************************************
     **** CLASS: ProAuFinger
     *************************************/
MakeClass( ProAuFinger, CustProdReqt );

/************** METHOD: CheckCustGenSpec  **************/
MakeMethod( ProAuFinger, CheckCustGenSpec, [],
```

```
{
  ForwardChain( [NOASSERT], NULL, CheckGenSpecAuFing );
} );
MakeSlot( ProAuFinger:AuThick );
SetSlotOption( ProAuFinger:AuThick, VALUE_TYPE, NUMBER );
ProAuFinger:AuThick = 0;
SetSlotOption( ProAuFinger:AuThick, PROMPT, "Finger Gold Thickness. (0.001in)" );
MakeSlot( ProAuFinger:NiThick );
SetSlotOption( ProAuFinger:NiThick, VALUE_TYPE, NUMBER );
ProAuFinger:NiThick = 0;
SetSlotOption( ProAuFinger:NiThick, PROMPT, "Finger Nickel Thickness. (0.001in)" );
MakeSlot( ProAuFinger:Height );
SetSlotOption( ProAuFinger:Height, VALUE_TYPE, NUMBER );
ProAuFinger:Height = 0;
SetSlotOption( ProAuFinger:Height, PROMPT, "Height of Gole Finger. (mm)" );
MakeSlot( ProAuFinger:UnitPerEdge );
SetSlotOption( ProAuFinger:UnitPerEdge, VALUE_TYPE, NUMBER );
ProAuFinger:UnitPerEdge = 0;
SetSlotOption( ProAuFinger:UnitPerEdge, PROMPT, "No. of Unit Per Plating Edge" );
MakeSlot( ProAuFinger:PlatAreaPerUnit );
SetSlotOption( ProAuFinger:PlatAreaPerUnit, VALUE_TYPE, NUMBER );
ProAuFinger:PlatAreaPerUnit = 0;
SetSlotOption( ProAuFinger:PlatAreaPerUnit, PROMPT, "Gold Finger Plating Area Per Unit" );

    /*************************************
     **** CLASS: ProFeatCircuit
     *************************************/
MakeClass( ProFeatCircuit, CustProdReqt );

/************** METHOD: CheckCustGenSpec  **************/
MakeMethod( ProFeatCircuit, CheckCustGenSpec, [],
{
  ForwardChain( [NOASSERT], NULL, CheckGenSpecFeatCir );
} );
MakeSlot( ProFeatCircuit:CirWidth );
SetSlotOption( ProFeatCircuit:CirWidth, VALUE_TYPE, NUMBER );
SetSlotOption( ProFeatCircuit:CirWidth, MINIMUM_VALUE, 0 );
ProFeatCircuit:CirWidth = 50;
SetSlotOption( ProFeatCircuit:CirWidth, PROMPT, "Min. Circuit Width.  (0.01mm)" );
MakeSlot( ProFeatCircuit:SpaceWidth );
SetSlotOption( ProFeatCircuit:SpaceWidth, VALUE_TYPE, NUMBER );
SetSlotOption( ProFeatCircuit:SpaceWidth, MINIMUM_VALUE, 0 );
ProFeatCircuit:SpaceWidth = 50;
SetSlotOption( ProFeatCircuit:SpaceWidth, PROMPT, "Min. Circuit Spacing.  (0.01mm)" );
MakeSlot( ProFeatCircuit:CirThickness );
SetSlotOption( ProFeatCircuit:CirThickness, VALUE_TYPE, NUMBER );
ProFeatCircuit:CirThickness = 4;
SetSlotOption( ProFeatCircuit:CirThickness, PROMPT, "Min. Circuit Thickness.  (0.01mm)" );
MakeSlot( ProFeatCircuit:AnnularRing );
SetSlotOption( ProFeatCircuit:AnnularRing, VALUE_TYPE, NUMBER );
ProFeatCircuit:AnnularRing = 25;
```

```
SetSlotOption( ProFeatCircuit:AnnularRing, PROMPT, "Min. Annular Ring.  (0.01mm)" );
MakeSlot( ProFeatCircuit:CircuitArea );
SetSlotOption( ProFeatCircuit:CircuitArea, VALUE_TYPE, NUMBER );
ProFeatCircuit:CircuitArea = 3.36;
SetSlotOption( ProFeatCircuit:CircuitArea, PROMPT, "Unit Circuit area in sq. in." );

    /*************************************
     **** CLASS: ProDocument
     *************************************/
MakeClass( ProDocument, CustProdReqt );

/************** METHOD: CheckCustGenSpec  **************/
MakeMethod( ProDocument, CheckCustGenSpec, [],
  TRUE );
MakeSlot( ProDocument:FilmNo );
ProDocument:FilmNo = PCB8;
SetSlotOption( ProDocument:FilmNo, PROMPT, "Film File Number." );
MakeSlot( ProDocument:DrawNo );
ProDocument:DrawNo = DN003;
SetSlotOption( ProDocument:DrawNo, PROMPT, "Drawing Number." );
MakeSlot( ProDocument:SpecNo );
ProDocument:SpecNo = 1002;
SetSlotOption( ProDocument:SpecNo, PROMPT, "Specification Number." );
MakeSlot( ProDocument:DrillFileNo );
ProDocument:DrillFileNo = -;
SetSlotOption( ProDocument:DrillFileNo, PROMPT, "Drill File Number." );

    /*************************************
     **** CLASS: ProProdDetails
     *************************************/
MakeClass( ProProdDetails, CustProdReqt );

/************** METHOD: CheckCustGenSpec  **************/
MakeMethod( ProProdDetails, CheckCustGenSpec, [],
  TRUE );
MakeSlot( ProProdDetails:ExpQty );
SetSlotOption( ProProdDetails:ExpQty, VALUE_TYPE, NUMBER );
ProProdDetails:ExpQty = 20000;
SetSlotOption( ProProdDetails:ExpQty, PROMPT, "Total Expected Production Quantity." );
MakeSlot( ProProdDetails:BatchSize );
SetSlotOption( ProProdDetails:BatchSize, VALUE_TYPE, NUMBER );
ProProdDetails:BatchSize = 5000;
SetSlotOption( ProProdDetails:BatchSize, PROMPT, "This Production Batch Size." );
MakeSlot( ProProdDetails:WkDateAvail );
SetSlotOption( ProProdDetails:WkDateAvail, VALUE_TYPE, NUMBER );
ProProdDetails:WkDateAvail = 30;
SetSlotOption( ProProdDetails:WkDateAvail, PROMPT, "Working Day Available for this Batch." );

    /*************************************
     **** CLASS: ProHoleSpec
     *************************************/
```

```
MakeClass( ProHoleSpec, CustProdReqt );

/************** METHOD: CheckCustGenSpec  **************/
MakeMethod( ProHoleSpec, CheckCustGenSpec, [],
  TRUE );
MakeSlot( ProHoleSpec:LocDev );
SetSlotOption( ProHoleSpec:LocDev, VALUE_TYPE, NUMBER );
ProHoleSpec:LocDev = 7.5;
SetSlotOption( ProHoleSpec:LocDev, PROMPT, "Deviation in Hole Location.  (0.01mm)" );
MakeSlot( ProHoleSpec:DiaDev );
SetSlotOption( ProHoleSpec:DiaDev, VALUE_TYPE, NUMBER );
ProHoleSpec:DiaDev = 7.5;
SetSlotOption( ProHoleSpec:DiaDev, PROMPT, "Deviation in Hole Diameter.  (0.01mm)" );
MakeSlot( ProHoleSpec:MinHoleSize );
SetSlotOption( ProHoleSpec:MinHoleSize, VALUE_TYPE, NUMBER );
SetSlotOption( ProHoleSpec:MinHoleSize, MINIMUM_VALUE, 0.4 );
SetSlotOption( ProHoleSpec:MinHoleSize, MAXIMUM_VALUE, 5 );
ProHoleSpec:MinHoleSize = 0.8;
SetSlotOption( ProHoleSpec:MinHoleSize, PROMPT, "Min. Hole Size.  (0.01mm)" );
MakeSlot( ProHoleSpec:MaxHoleSize );
SetSlotOption( ProHoleSpec:MaxHoleSize, ALLOWABLE_VALUES, 0.4, 5 );
ProHoleSpec:MaxHoleSize = 2.5;
SetSlotOption( ProHoleSpec:MaxHoleSize, PROMPT, "Max. Hole Size." );

    /*************************************
     ****  CLASS: ProCompMark
     *************************************/
MakeClass( ProCompMark, CustProdReqt );

/************** METHOD: CheckCustGenSpec  **************/
MakeMethod( ProCompMark, CheckCustGenSpec, [],
  TRUE );
MakeSlot( ProCompMark:CompMaterial );
ProCompMark:CompMaterial = meterialB;
SetSlotOption( ProCompMark:CompMaterial, PROMPT, "Marking Material." );
MakeSlot( ProCompMark:CompColour );
ProCompMark:CompColour = yellow;
SetSlotOption( ProCompMark:CompColour, PROMPT, "Marking Colour." );
```

Appendix 3.4 KAL Representation of the CircuitD Requirements

```
    /************************************
     **** CLASS: ReqDocInfo
     ************************************/
MakeClass( ReqDocInfo, CustProdReqt );

/************** METHOD: CheckCustGenSpec  **************/
MakeMethod( ReqDocInfo, CheckCustGenSpec, [],
TRUE );
MakeSlot( ReqDocInfo:CustName );
SetSlotOption( ReqDocInfo:CustName, ALLOWABLE_VALUES, CustomerA, CustomerB, CustomerC );
ReqDocInfo:CustName = CustomerB;
SetSlotOption( ReqDocInfo:CustName, PROMPT, "Name of Customer." );
MakeSlot( ReqDocInfo:ProdName );
ReqDocInfo:ProdName = CircuitD;
SetSlotOption( ReqDocInfo:ProdName, PROMPT, "Name of the Product." );
MakeSlot( ReqDocInfo:PNInHouse );
ReqDocInfo:PNInHouse = p8004;
SetSlotOption( ReqDocInfo:PNInHouse, PROMPT, "Assigned In-House Part Number for this Product." );
MakeSlot( ReqDocInfo:PNCust );
ReqDocInfo:PNCust = A004;
SetSlotOption( ReqDocInfo:PNCust, PROMPT, "Customer Product Part Number." );
MakeSlot( ReqDocInfo:PurOrdNo );
ReqDocInfo:PurOrdNo = PO004;
SetSlotOption( ReqDocInfo:PurOrdNo, PROMPT, "Purchase Order Number." );

    /************************************
     **** CLASS: ProPlatingReq
     ************************************/
MakeClass( ProPlatingReq, CustProdReqt );

/************** METHOD: CheckCustGenSpec  **************/
MakeMethod( ProPlatingReq, CheckCustGenSpec, [],
{
  ForwardChain( [NOASSERT], NULL, CheckGenSpecPlatReq );
} );
MakeSlot( ProPlatingReq:SurfaceFinType );
SetSlotOption( ProPlatingReq:SurfaceFinType, ALLOWABLE_VALUES, Ni, Au, BareCopper, SnPb );
ProPlatingReq:SurfaceFinType = SnPb;
SetSlotOption( ProPlatingReq:SurfaceFinType, PROMPT, "Type of Surface Finishing." );
MakeSlot( ProPlatingReq:SurfaceFinThick );
SetSlotOption( ProPlatingReq:SurfaceFinThick, VALUE_TYPE, NUMBER );
ProPlatingReq:SurfaceFinThick = 0.3;
SetSlotOption( ProPlatingReq:SurfaceFinThick, PROMPT, "Plated Metal Min. Surface Thickness (other
than copper).  (0.001in)" );
MakeSlot( ProPlatingReq:CuThickMin );
SetSlotOption( ProPlatingReq:CuThickMin, VALUE_TYPE, NUMBER );
ProPlatingReq:CuThickMin = 2;
SetSlotOption( ProPlatingReq:CuThickMin, PROMPT, "Plated Copper Min. Thickness.  (0.001in)" );
```

```
    /*************************************
     **** CLASS: ProSoldMask
     *************************************/
MakeClass( ProSoldMask, CustProdReqt );

/************** METHOD: CheckCustGenSpec  **************/
MakeMethod( ProSoldMask, CheckCustGenSpec, [],
{
  ForwardChain( [NOASSERT], NULL, CheckGenSpecSoldMask );
} );
MakeSlot( ProSoldMask:MaterialType );
ProSoldMask:MaterialType = PC501;
SetSlotOption( ProSoldMask:MaterialType, PROMPT, "Type of Solder Masking Material." );
SetSlotOption( ProSoldMask:MaterialType, IMAGE, StateBox1 );
MakeSlot( ProSoldMask:MinThick );
SetSlotOption( ProSoldMask:MinThick, VALUE_TYPE, NUMBER );
ProSoldMask:MinThick = 1;
SetSlotOption( ProSoldMask:MinThick, PROMPT, "Acceptable Min. Solder Masking Thickness.
(0.01mm)" );
MakeSlot( ProSoldMask:MinClear );
SetSlotOption( ProSoldMask:MinClear, VALUE_TYPE, NUMBER );
ProSoldMask:MinClear = 50;
SetSlotOption( ProSoldMask:MinClear, PROMPT, "Acceptable Min. Solder Masking Pad Clearance.
(0.01mm)" );
MakeSlot( ProSoldMask:Colour );
ProSoldMask:Colour = Green;
SetSlotOption( ProSoldMask:Colour, PROMPT, "Colour of Solder Masking Material." );

    /*************************************
     **** CLASS: ProPanelDimen
     *************************************/
MakeClass( ProPanelDimen, CustProdReqt );

/************** METHOD: CheckCustGenSpec  **************/
MakeMethod( ProPanelDimen, CheckCustGenSpec, [],
  TRUE );
MakeSlot( ProPanelDimen:Width );
SetSlotOption( ProPanelDimen:Width, VALUE_TYPE, NUMBER );
ProPanelDimen:Width = 512;
SetSlotOption( ProPanelDimen:Width, PROMPT, "Board Panel Width. (in mm)" );
MakeSlot( ProPanelDimen:Length );
SetSlotOption( ProPanelDimen:Length, VALUE_TYPE, NUMBER );
ProPanelDimen:Length = 422.5;
SetSlotOption( ProPanelDimen:Length, PROMPT, "Board Panel Length. (in mm)" );
MakeSlot( ProPanelDimen:DimenDev );
SetSlotOption( ProPanelDimen:DimenDev, VALUE_TYPE, NUMBER );
ProPanelDimen:DimenDev = 12.5;
SetSlotOption( ProPanelDimen:DimenDev, PROMPT, "Acceptable Deviation in Outline Dimension.
(0.01mm)" );
MakeSlot( ProPanelDimen:UnitLength );
```

```
SetSlotOption( ProPanelDimen:UnitLength, VALUE_TYPE, NUMBER );
ProPanelDimen:UnitLength = 127.5;
SetSlotOption( ProPanelDimen:UnitLength, PROMPT, "Unit Length (in mm)." );
MakeSlot( ProPanelDimen:UnitWidth );
ProPanelDimen:UnitWidth = 241;
SetSlotOption( ProPanelDimen:UnitWidth, PROMPT, "Unit Width (in mm)." );
MakeSlot( ProPanelDimen:UnitPerPanel );
SetSlotOption( ProPanelDimen:UnitPerPanel, VALUE_TYPE, NUMBER );
ProPanelDimen:UnitPerPanel = 6;
SetSlotOption( ProPanelDimen:UnitPerPanel, PROMPT, "No. of Unit Per Panel" );
MakeSlot( ProPanelDimen:EstDieCost );
SetSlotOption( ProPanelDimen:EstDieCost, VALUE_TYPE, NUMBER );
ProPanelDimen:EstDieCost = 8000;
SetSlotOption( ProPanelDimen:EstDieCost, PROMPT, "Estimated Blanking Die Cost. (HK$)" );

    /*************************************
     **** CLASS: ProLamChar
     *************************************/
MakeClass( ProLamChar, CustProdReqt );

/************** METHOD: CheckCustGenSpec  **************/
MakeMethod( ProLamChar, CheckCustGenSpec, [],
  TRUE );
MakeSlot( ProLamChar:LaminateType );
SetSlotOption( ProLamChar:LaminateType, ALLOWABLE_VALUES, CEM2, CEM3, FR3, FR4 );
ProLamChar:LaminateType = FR4;
SetSlotOption( ProLamChar:LaminateType, PROMPT, "Type of Laminate." );
MakeSlot( ProLamChar:CuFoilThickness );
SetSlotOption( ProLamChar:CuFoilThickness, ALLOWABLE_VALUES, 0.5, 1, 1.5, 2, 2.5 );
ProLamChar:CuFoilThickness = 1;
SetSlotOption( ProLamChar:CuFoilThickness, PROMPT, "Copper Foil Thickness. (oz.)" );
MakeSlot( ProLamChar:CuFoilSide );
SetSlotOption( ProLamChar:CuFoilSide, ALLOWABLE_VALUES, Single, Double );
ProLamChar:CuFoilSide = Double;
SetSlotOption( ProLamChar:CuFoilSide, PROMPT, "Side of Copper Foil." );
MakeSlot( ProLamChar:LamThickness );
SetSlotOption( ProLamChar:LamThickness, ALLOWABLE_VALUES, 0.8, 1.2, 1.6, 1.8, 3.2 );
ProLamChar:LamThickness = 1.6;
SetSlotOption( ProLamChar:LamThickness, PROMPT, "Laminate Thickness." );
MakeSlot( ProLamChar:Colour );
SetSlotOption( ProLamChar:Colour, ALLOWABLE_VALUES, Green, Red, Blue );
ProLamChar:Colour = Green;
SetSlotOption( ProLamChar:Colour, PROMPT, "Colour of Laminate." );

    /*************************************
     **** CLASS: ProAuFinger
     *************************************/
MakeClass( ProAuFinger, CustProdReqt );

/************** METHOD: CheckCustGenSpec  **************/
MakeMethod( ProAuFinger, CheckCustGenSpec, [],
```

```
{
  ForwardChain( [NOASSERT], NULL, CheckGenSpecAuFing );
} );
MakeSlot( ProAuFinger:AuThick );
SetSlotOption( ProAuFinger:AuThick, VALUE_TYPE, NUMBER );
ProAuFinger:AuThick = 0.02;
SetSlotOption( ProAuFinger:AuThick, PROMPT, "Finger Gold Thickness. (0.001in)" );
MakeSlot( ProAuFinger:NiThick );
SetSlotOption( ProAuFinger:NiThick, VALUE_TYPE, NUMBER );
ProAuFinger:NiThick = 0.2;
SetSlotOption( ProAuFinger:NiThick, PROMPT, "Finger Nickel Thickness. (0.001in)" );
MakeSlot( ProAuFinger:Height );
SetSlotOption( ProAuFinger:Height, VALUE_TYPE, NUMBER );
ProAuFinger:Height = 13;
SetSlotOption( ProAuFinger:Height, PROMPT, "Height of Gole Finger. (mm)" );
MakeSlot( ProAuFinger:UnitPerEdge );
SetSlotOption( ProAuFinger:UnitPerEdge, VALUE_TYPE, NUMBER );
ProAuFinger:UnitPerEdge = 3;
SetSlotOption( ProAuFinger:UnitPerEdge, PROMPT, "No. of Unit Per Plating Edge" );
MakeSlot( ProAuFinger:PlatAreaPerUnit );
SetSlotOption( ProAuFinger:PlatAreaPerUnit, VALUE_TYPE, NUMBER );
ProAuFinger:PlatAreaPerUnit = 1.8;
SetSlotOption( ProAuFinger:PlatAreaPerUnit, PROMPT, "Gold Finger Plating Area Per Unit" );

    /*************************************
     **** CLASS: ProFeatCircuit
     *************************************/
MakeClass( ProFeatCircuit, CustProdReqt );

/************** METHOD: CheckCustGenSpec  **************/
MakeMethod( ProFeatCircuit, CheckCustGenSpec, [],
{
  ForwardChain( [NOASSERT], NULL, CheckGenSpecFeatCir );
} );
MakeSlot( ProFeatCircuit:CirWidth );
SetSlotOption( ProFeatCircuit:CirWidth, VALUE_TYPE, NUMBER );
SetSlotOption( ProFeatCircuit:CirWidth, MINIMUM_VALUE, 0 );
ProFeatCircuit:CirWidth = 100;
SetSlotOption( ProFeatCircuit:CirWidth, PROMPT, "Min. Circuit Width.  (0.01mm)" );
MakeSlot( ProFeatCircuit:SpaceWidth );
SetSlotOption( ProFeatCircuit:SpaceWidth, VALUE_TYPE, NUMBER );
SetSlotOption( ProFeatCircuit:SpaceWidth, MINIMUM_VALUE, 0 );
ProFeatCircuit:SpaceWidth = 100;
SetSlotOption( ProFeatCircuit:SpaceWidth, PROMPT, "Min. Circuit Spacing.  (0.01mm)" );
MakeSlot( ProFeatCircuit:CirThickness );
SetSlotOption( ProFeatCircuit:CirThickness, VALUE_TYPE, NUMBER );
ProFeatCircuit:CirThickness = 4;
SetSlotOption( ProFeatCircuit:CirThickness, PROMPT, "Min. Circuit Thickness.  (0.01mm)" );
MakeSlot( ProFeatCircuit:AnnularRing );
SetSlotOption( ProFeatCircuit:AnnularRing, VALUE_TYPE, NUMBER );
ProFeatCircuit:AnnularRing = 20;
```

```
SetSlotOption( ProFeatCircuit:AnnularRing, PROMPT, "Min. Annular Ring.  (0.01mm)" );
MakeSlot( ProFeatCircuit:CircuitArea );
SetSlotOption( ProFeatCircuit:CircuitArea, VALUE_TYPE, NUMBER );
ProFeatCircuit:CircuitArea = 18.1;
SetSlotOption( ProFeatCircuit:CircuitArea, PROMPT, "Unit Circuit area in sq. in." );

    /*************************************
     **** CLASS: ProDocument
     *************************************/
MakeClass( ProDocument, CustProdReqt );

/************** METHOD: CheckCustGenSpec  **************/
MakeMethod( ProDocument, CheckCustGenSpec, [],
  TRUE );
MakeSlot( ProDocument:FilmNo );
ProDocument:FilmNo = PCB4;
SetSlotOption( ProDocument:FilmNo, PROMPT, "Film File Number." );
MakeSlot( ProDocument:DrawNo );
ProDocument:DrawNo = DN004;
SetSlotOption( ProDocument:DrawNo, PROMPT, "Drawing Number." );
MakeSlot( ProDocument:SpecNo );
ProDocument:SpecNo = 1002;
SetSlotOption( ProDocument:SpecNo, PROMPT, "Specification Number." );
MakeSlot( ProDocument:DrillFileNo );
ProDocument:DrillFileNo = -;
SetSlotOption( ProDocument:DrillFileNo, PROMPT, "Drill File Number." );

    /*************************************
     **** CLASS: ProProdDetails
     *************************************/
MakeClass( ProProdDetails, CustProdReqt );

/************** METHOD: CheckCustGenSpec  **************/
MakeMethod( ProProdDetails, CheckCustGenSpec, [],
  TRUE );
MakeSlot( ProProdDetails:ExpQty );
SetSlotOption( ProProdDetails:ExpQty, VALUE_TYPE, NUMBER );
ProProdDetails:ExpQty = 20000;
SetSlotOption( ProProdDetails:ExpQty, PROMPT, "Total Expected Production Quantity." );
MakeSlot( ProProdDetails:BatchSize );
SetSlotOption( ProProdDetails:BatchSize, VALUE_TYPE, NUMBER );
ProProdDetails:BatchSize = 5000;
SetSlotOption( ProProdDetails:BatchSize, PROMPT, "This Production Batch Size." );
MakeSlot( ProProdDetails:WkDateAvail );
SetSlotOption( ProProdDetails:WkDateAvail, VALUE_TYPE, NUMBER );
ProProdDetails:WkDateAvail = 30;
SetSlotOption( ProProdDetails:WkDateAvail, PROMPT, "Working Day Available for this Batch." );

    /*************************************
     **** CLASS: ProHoleSpec
     *************************************/
```

```
MakeClass( ProHoleSpec, CustProdReqt );

/************** METHOD: CheckCustGenSpec  **************/
MakeMethod( ProHoleSpec, CheckCustGenSpec, [],
  TRUE );
MakeSlot( ProHoleSpec:LocDev );
SetSlotOption( ProHoleSpec:LocDev, VALUE_TYPE, NUMBER );
ProHoleSpec:LocDev = 7.5;
SetSlotOption( ProHoleSpec:LocDev, PROMPT, "Deviation in Hole Location.  (0.01mm)" );
MakeSlot( ProHoleSpec:DiaDev );
SetSlotOption( ProHoleSpec:DiaDev, VALUE_TYPE, NUMBER );
ProHoleSpec:DiaDev = 7.5;
SetSlotOption( ProHoleSpec:DiaDev, PROMPT, "Deviation in Hole Diameter.  (0.01mm)" );
MakeSlot( ProHoleSpec:MinHoleSize );
SetSlotOption( ProHoleSpec:MinHoleSize, VALUE_TYPE, NUMBER );
SetSlotOption( ProHoleSpec:MinHoleSize, MINIMUM_VALUE, 0.4 );
SetSlotOption( ProHoleSpec:MinHoleSize, MAXIMUM_VALUE, 5 );
ProHoleSpec:MinHoleSize = 0.8;
SetSlotOption( ProHoleSpec:MinHoleSize, PROMPT, "Min. Hole Size.  (0.01mm)" );
MakeSlot( ProHoleSpec:MaxHoleSize );
SetSlotOption( ProHoleSpec:MaxHoleSize, ALLOWABLE_VALUES, 0.4, 5 );
ProHoleSpec:MaxHoleSize = 4;
SetSlotOption( ProHoleSpec:MaxHoleSize, PROMPT, "Max. Hole Size." );

    /************************************
     ****  CLASS: ProCompMark
     ************************************/
MakeClass( ProCompMark, CustProdReqt );

/************** METHOD: CheckCustGenSpec  **************/
MakeMethod( ProCompMark, CheckCustGenSpec, [],
  TRUE );
MakeSlot( ProCompMark:CompMaterial );
ProCompMark:CompMaterial = materialA;
SetSlotOption( ProCompMark:CompMaterial, PROMPT, "Marking Material." );
MakeSlot( ProCompMark:CompColour );
ProCompMark:CompColour = white;
SetSlotOption( ProCompMark:CompColour, PROMPT, "Marking Colour." );
```

Appendix 3.5 KAL Representation of the CircuitE Requirements

```
    /************************************
     **** CLASS: ReqDocInfo
     ************************************/
MakeClass( ReqDocInfo, CustProdReqt );

/************** METHOD: CheckCustGenSpec  **************/
MakeMethod( ReqDocInfo, CheckCustGenSpec, [],
TRUE );
MakeSlot( ReqDocInfo:CustName );
SetSlotOption( ReqDocInfo:CustName, ALLOWABLE_VALUES, CustomerA, CustomerB, CustomerC );
ReqDocInfo:CustName = CustomerA;
SetSlotOption( ReqDocInfo:CustName, PROMPT, "Name of Customer." );
MakeSlot( ReqDocInfo:ProdName );
ReqDocInfo:ProdName = CircuitE;
SetSlotOption( ReqDocInfo:ProdName, PROMPT, "Name of the Product." );
MakeSlot( ReqDocInfo:PNInHouse );
ReqDocInfo:PNInHouse = p8005;
SetSlotOption( ReqDocInfo:PNInHouse, PROMPT, "Assigned In-House Part Number for this Product." );
MakeSlot( ReqDocInfo:PNCust );
ReqDocInfo:PNCust = A005;
SetSlotOption( ReqDocInfo:PNCust, PROMPT, "Customer Product Part Number." );
MakeSlot( ReqDocInfo:PurOrdNo );
ReqDocInfo:PurOrdNo = PO005;
SetSlotOption( ReqDocInfo:PurOrdNo, PROMPT, "Purchase Order Number." );

    /************************************
     **** CLASS: ProPlatingReq
     ************************************/
MakeClass( ProPlatingReq, CustProdReqt );

/************** METHOD: CheckCustGenSpec  **************/
MakeMethod( ProPlatingReq, CheckCustGenSpec, [],
{
  ForwardChain( [NOASSERT], NULL, CheckGenSpecPlatReq );
} );
MakeSlot( ProPlatingReq:SurfaceFinType );
SetSlotOption( ProPlatingReq:SurfaceFinType, ALLOWABLE_VALUES, Ni, Au, BareCopper, SnPb );
ProPlatingReq:SurfaceFinType = BareCopper;
SetSlotOption( ProPlatingReq:SurfaceFinType, PROMPT, "Type of Surface Finishing." );
SetSlotOption( ProPlatingReq:SurfaceFinType, IMAGE, ComboBox32 );
MakeSlot( ProPlatingReq:SurfaceFinThick );
SetSlotOption( ProPlatingReq:SurfaceFinThick, VALUE_TYPE, NUMBER );
ProPlatingReq:SurfaceFinThick = 0.3;
SetSlotOption( ProPlatingReq:SurfaceFinThick, PROMPT, "Plated Metal Min. Surface Thickness (other
than copper). (0.001in)" );
MakeSlot( ProPlatingReq:CuThickMin );
SetSlotOption( ProPlatingReq:CuThickMin, VALUE_TYPE, NUMBER );
ProPlatingReq:CuThickMin = 2;
```

```
SetSlotOption( ProPlatingReq:CuThickMin, PROMPT, "Plated Copper Min. Thickness.  (0.001in)" );

    /*************************************
     **** CLASS: ProSoldMask
     *************************************/
MakeClass( ProSoldMask, CustProdReqt );

/************** METHOD: CheckCustGenSpec  **************/
MakeMethod( ProSoldMask, CheckCustGenSpec, [],
{
  ForwardChain( [NOASSERT], NULL, CheckGenSpecSoldMask );
} );
MakeSlot( ProSoldMask:MaterialType );
ProSoldMask:MaterialType = PC401;
SetSlotOption( ProSoldMask:MaterialType, PROMPT, "Type of Solder Masking Material." );
SetSlotOption( ProSoldMask:MaterialType, IMAGE, StateBox1 );
MakeSlot( ProSoldMask:MinThick );
SetSlotOption( ProSoldMask:MinThick, VALUE_TYPE, NUMBER );
ProSoldMask:MinThick = 1;
SetSlotOption( ProSoldMask:MinThick, PROMPT, "Acceptable Min. Solder Masking Thickness.
(0.01mm)" );
MakeSlot( ProSoldMask:MinClear );
SetSlotOption( ProSoldMask:MinClear, VALUE_TYPE, NUMBER );
ProSoldMask:MinClear = 4;
SetSlotOption( ProSoldMask:MinClear, PROMPT, "Acceptable Min. Solder Masking Pad Clearance.
(0.01mm)" );
MakeSlot( ProSoldMask:Colour );
ProSoldMask:Colour = Green;
SetSlotOption( ProSoldMask:Colour, PROMPT, "Colour of Solder Masking Material." );

    /*************************************
     **** CLASS: ProPanelDimen
     *************************************/
MakeClass( ProPanelDimen, CustProdReqt );

/************** METHOD: CheckCustGenSpec  **************/
MakeMethod( ProPanelDimen, CheckCustGenSpec, [],
  TRUE );
MakeSlot( ProPanelDimen:Width );
SetSlotOption( ProPanelDimen:Width, VALUE_TYPE, NUMBER );
ProPanelDimen:Width = 630;
SetSlotOption( ProPanelDimen:Width, PROMPT, "Board Panel Width. (in mm)" );
MakeSlot( ProPanelDimen:Length );
SetSlotOption( ProPanelDimen:Length, VALUE_TYPE, NUMBER );
ProPanelDimen:Length = 630;
SetSlotOption( ProPanelDimen:Length, PROMPT, "Board Panel Length. (in mm)" );
MakeSlot( ProPanelDimen:DimenDev );
SetSlotOption( ProPanelDimen:DimenDev, VALUE_TYPE, NUMBER );
ProPanelDimen:DimenDev = 10;
SetSlotOption( ProPanelDimen:DimenDev, PROMPT, "Acceptable Deviation in Outline Dimension.
(0.01mm)" );
```

```
MakeSlot( ProPanelDimen:UnitLength );
SetSlotOption( ProPanelDimen:UnitLength, VALUE_TYPE, NUMBER );
ProPanelDimen:UnitLength = 300;
SetSlotOption( ProPanelDimen:UnitLength, PROMPT, "Unit Length (in mm)." );
MakeSlot( ProPanelDimen:UnitWidth );
ProPanelDimen:UnitWidth = 300;
SetSlotOption( ProPanelDimen:UnitWidth, PROMPT, "Unit Width (in mm)." );
MakeSlot( ProPanelDimen:UnitPerPanel );
SetSlotOption( ProPanelDimen:UnitPerPanel, VALUE_TYPE, NUMBER );
ProPanelDimen:UnitPerPanel = 4;
SetSlotOption( ProPanelDimen:UnitPerPanel, PROMPT, "No. of Unit Per Panel" );
MakeSlot( ProPanelDimen:EstDieCost );
SetSlotOption( ProPanelDimen:EstDieCost, VALUE_TYPE, NUMBER );
ProPanelDimen:EstDieCost = 10000;
SetSlotOption( ProPanelDimen:EstDieCost, PROMPT, "Estimated Blanking Die Cost. (HK$)" );

    /************************************
     **** CLASS: ProLamChar
     ************************************/
MakeClass( ProLamChar, CustProdReqt );

/************** METHOD: CheckCustGenSpec  **************/
MakeMethod( ProLamChar, CheckCustGenSpec, [],
  TRUE );
MakeSlot( ProLamChar:LaminateType );
SetSlotOption( ProLamChar:LaminateType, ALLOWABLE_VALUES, CEM2, CEM3, FR3, FR4 );
ProLamChar:LaminateType = FR4;
SetSlotOption( ProLamChar:LaminateType, PROMPT, "Type of Laminate." );
MakeSlot( ProLamChar:CuFoilThickness );
SetSlotOption( ProLamChar:CuFoilThickness, ALLOWABLE_VALUES, 0.5, 1, 1.5, 2, 2.5 );
ProLamChar:CuFoilThickness = 1;
SetSlotOption( ProLamChar:CuFoilThickness, PROMPT, "Copper Foil Thickness. (oz.)" );
MakeSlot( ProLamChar:CuFoilSide );
SetSlotOption( ProLamChar:CuFoilSide, ALLOWABLE_VALUES, Single, Double );
ProLamChar:CuFoilSide = Double;
SetSlotOption( ProLamChar:CuFoilSide, PROMPT, "Side of Copper Foil." );
MakeSlot( ProLamChar:LamThickness );
SetSlotOption( ProLamChar:LamThickness, ALLOWABLE_VALUES, 0.8, 1.2, 1.6, 1.8, 3.2 );
ProLamChar:LamThickness = 1.6;
SetSlotOption( ProLamChar:LamThickness, PROMPT, "Laminate Thickness." );
MakeSlot( ProLamChar:Colour );
SetSlotOption( ProLamChar:Colour, ALLOWABLE_VALUES, Green, Red, Blue );
ProLamChar:Colour = Green;
SetSlotOption( ProLamChar:Colour, PROMPT, "Colour of Laminate." );

    /************************************
     **** CLASS: ProAuFinger
     ************************************/
MakeClass( ProAuFinger, CustProdReqt );

/************** METHOD: CheckCustGenSpec  **************/
```

```
MakeMethod( ProAuFinger, CheckCustGenSpec, [],
{
  ForwardChain( [NOASSERT], NULL, CheckGenSpecAuFing );
} );
MakeSlot( ProAuFinger:AuThick );
SetSlotOption( ProAuFinger:AuThick, VALUE_TYPE, NUMBER );
ProAuFinger:AuThick = 0;
SetSlotOption( ProAuFinger:AuThick, PROMPT, "Finger Gold Thickness. (0.001in)" );
MakeSlot( ProAuFinger:NiThick );
SetSlotOption( ProAuFinger:NiThick, VALUE_TYPE, NUMBER );
ProAuFinger:NiThick = 00;
SetSlotOption( ProAuFinger:NiThick, PROMPT, "Finger Nickel Thickness. (0.001in)" );
MakeSlot( ProAuFinger:Height );
SetSlotOption( ProAuFinger:Height, VALUE_TYPE, NUMBER );
ProAuFinger:Height = 0;
SetSlotOption( ProAuFinger:Height, PROMPT, "Height of Gole Finger. (mm)" );
MakeSlot( ProAuFinger:UnitPerEdge );
SetSlotOption( ProAuFinger:UnitPerEdge, VALUE_TYPE, NUMBER );
ProAuFinger:UnitPerEdge = 0;
SetSlotOption( ProAuFinger:UnitPerEdge, PROMPT, "No. of Unit Per Plating Edge" );
MakeSlot( ProAuFinger:PlatAreaPerUnit );
SetSlotOption( ProAuFinger:PlatAreaPerUnit, VALUE_TYPE, NUMBER );
ProAuFinger:PlatAreaPerUnit = 0;
SetSlotOption( ProAuFinger:PlatAreaPerUnit, PROMPT, "Gold Finger Plating Area Per Unit" );

    /*************************************
     **** CLASS: ProFeatCircuit
     *************************************/
MakeClass( ProFeatCircuit, CustProdReqt );

/************** METHOD: CheckCustGenSpec  **************/
MakeMethod( ProFeatCircuit, CheckCustGenSpec, [],
{
  ForwardChain( [NOASSERT], NULL, CheckGenSpecFeatCir );
} );
MakeSlot( ProFeatCircuit:CirWidth );
SetSlotOption( ProFeatCircuit:CirWidth, VALUE_TYPE, NUMBER );
SetSlotOption( ProFeatCircuit:CirWidth, MINIMUM_VALUE, 0 );
ProFeatCircuit:CirWidth = 10;
SetSlotOption( ProFeatCircuit:CirWidth, PROMPT, "Min. Circuit Width.  (0.01mm)" );
MakeSlot( ProFeatCircuit:SpaceWidth );
SetSlotOption( ProFeatCircuit:SpaceWidth, VALUE_TYPE, NUMBER );
SetSlotOption( ProFeatCircuit:SpaceWidth, MINIMUM_VALUE, 0 );
ProFeatCircuit:SpaceWidth = 10;
SetSlotOption( ProFeatCircuit:SpaceWidth, PROMPT, "Min. Circuit Spacing.  (0.01mm)" );
MakeSlot( ProFeatCircuit:CirThickness );
SetSlotOption( ProFeatCircuit:CirThickness, VALUE_TYPE, NUMBER );
ProFeatCircuit:CirThickness = 4;
SetSlotOption( ProFeatCircuit:CirThickness, PROMPT, "Min. Circuit Thickness.  (0.01mm)" );
MakeSlot( ProFeatCircuit:AnnularRing );
SetSlotOption( ProFeatCircuit:AnnularRing, VALUE_TYPE, NUMBER );
```

```
ProFeatCircuit:AnnularRing = 2;
SetSlotOption( ProFeatCircuit:AnnularRing, PROMPT, "Min. Annular Ring.  (0.01mm)" );
MakeSlot( ProFeatCircuit:CircuitArea );
SetSlotOption( ProFeatCircuit:CircuitArea, VALUE_TYPE, NUMBER );
ProFeatCircuit:CircuitArea = 6.2;
SetSlotOption( ProFeatCircuit:CircuitArea, PROMPT, "Unit Circuit area in sq. in." );

    /************************************
     **** CLASS: ProDocument
     ************************************/
MakeClass( ProDocument, CustProdReqt );

/************** METHOD: CheckCustGenSpec  **************/
MakeMethod( ProDocument, CheckCustGenSpec, [],
  TRUE );
MakeSlot( ProDocument:FilmNo );
ProDocument:FilmNo = PCB10;
SetSlotOption( ProDocument:FilmNo, PROMPT, "Film File Number." );
MakeSlot( ProDocument:DrawNo );
ProDocument:DrawNo = DN005;
SetSlotOption( ProDocument:DrawNo, PROMPT, "Drawing Number." );
MakeSlot( ProDocument:SpecNo );
ProDocument:SpecNo = 1001;
SetSlotOption( ProDocument:SpecNo, PROMPT, "Specification Number." );
MakeSlot( ProDocument:DrillFileNo );
ProDocument:DrillFileNo = -;
SetSlotOption( ProDocument:DrillFileNo, PROMPT, "Drill File Number." );

    /************************************
     **** CLASS: ProProdDetails
     ************************************/
MakeClass( ProProdDetails, CustProdReqt );

/************** METHOD: CheckCustGenSpec  **************/
MakeMethod( ProProdDetails, CheckCustGenSpec, [],
  TRUE );
MakeSlot( ProProdDetails:ExpQty );
SetSlotOption( ProProdDetails:ExpQty, VALUE_TYPE, NUMBER );
ProProdDetails:ExpQty = 20000;
SetSlotOption( ProProdDetails:ExpQty, PROMPT, "Total Expected Production Quantity." );
MakeSlot( ProProdDetails:BatchSize );
SetSlotOption( ProProdDetails:BatchSize, VALUE_TYPE, NUMBER );
ProProdDetails:BatchSize = 5000;
SetSlotOption( ProProdDetails:BatchSize, PROMPT, "This Production Batch Size." );
MakeSlot( ProProdDetails:WkDateAvail );
SetSlotOption( ProProdDetails:WkDateAvail, VALUE_TYPE, NUMBER );
ProProdDetails:WkDateAvail = 30;
SetSlotOption( ProProdDetails:WkDateAvail, PROMPT, "Working Day Available for this Batch." );

    /************************************
     **** CLASS: ProHoleSpec
```

```
    ************************************/
MakeClass( ProHoleSpec, CustProdReqt );

/************** METHOD: CheckCustGenSpec  **************/
MakeMethod( ProHoleSpec, CheckCustGenSpec, [],
  TRUE );
MakeSlot( ProHoleSpec:LocDev );
SetSlotOption( ProHoleSpec:LocDev, VALUE_TYPE, NUMBER );
ProHoleSpec:LocDev = 5;
SetSlotOption( ProHoleSpec:LocDev, PROMPT, "Deviation in Hole Location.  (0.01mm)" );
MakeSlot( ProHoleSpec:DiaDev );
SetSlotOption( ProHoleSpec:DiaDev, VALUE_TYPE, NUMBER );
ProHoleSpec:DiaDev = 12;
SetSlotOption( ProHoleSpec:DiaDev, PROMPT, "Deviation in Hole Diameter.  (0.01mm)" );
MakeSlot( ProHoleSpec:MinHoleSize );
SetSlotOption( ProHoleSpec:MinHoleSize, VALUE_TYPE, NUMBER );
SetSlotOption( ProHoleSpec:MinHoleSize, MINIMUM_VALUE, 0.4 );
SetSlotOption( ProHoleSpec:MinHoleSize, MAXIMUM_VALUE, 5 );
ProHoleSpec:MinHoleSize = 0.9;
SetSlotOption( ProHoleSpec:MinHoleSize, PROMPT, "Min. Hole Size.  (0.01mm)" );
MakeSlot( ProHoleSpec:MaxHoleSize );
SetSlotOption( ProHoleSpec:MaxHoleSize, ALLOWABLE_VALUES, 0.4, 5 );
ProHoleSpec:MaxHoleSize = 5;
SetSlotOption( ProHoleSpec:MaxHoleSize, PROMPT, "Max. Hole Size." );

    /************************************
     **** CLASS: ProCompMark
     ************************************/
MakeClass( ProCompMark, CustProdReqt );

/************** METHOD: CheckCustGenSpec  **************/
MakeMethod( ProCompMark, CheckCustGenSpec, [],
  TRUE );
MakeSlot( ProCompMark:CompMaterial );
ProCompMark:CompMaterial = materialA;
SetSlotOption( ProCompMark:CompMaterial, PROMPT, "Marking Material." );
MakeSlot( ProCompMark:CompColour );
ProCompMark:CompColour = white;
SetSlotOption( ProCompMark:CompColour, PROMPT, "Marking Colour." );
```

Appendix 4.1 Listing of the Circuit Feature Extraction Program

```
/*

        (a):\ger\gerfile.cpp                    6July93 02:00

        This program will read in Gerber format file and separate the file data
        into pad data and line data. Each different type of pad and line data
        will be storied into different file.

        The pad file name format will be called pad(1).f02 while the line file
        name format will be line(1).f02. After that, the line files will be
        further separated into whether it is parallel to x axis or parallel to
        y axis or whether it is not parallel to x or y axis.

        This program assume a max of three different line and three different pads.

*/

#include <stdio.h>
#include <stdlib.h>  // for exit function
#include <iostream.h>// for cout use
#include <string.h>
#include <math.h>   // for math function use
#include <delfile.h>        // self developed function to delete file
#include <handfile.h>       // self developed function to handle file
#include <bld_f_n.h>                // self developed function to build up file name

#define Max_line_in_ger 450  // max number of line in gerber file
#define Dec_pt 1000                             // dec pt

static char ger_file_name[]="a:\\test.gbr";
                                                        // drive must be change for
different computer

static int amp_look_up_table[]={122, 133, 167};
int no_of_pad_file=3, no_of_line_file=3;

class ger_f_for
{
        char pad_file_name[30];
        char line_file_name[30];

        public:
        void ger_f_for::ger_to_f01(char *ger_file_name);
        void master_pad_line_file(void);
        void sort_line_file(char *line_file_name);
        void sort_pad_file(char *pad_file_name);
        void sort_small_par_x(char *line_file_name);
  void sort_small_par_y(char *line_file_name);
```

```
}       circuit_film[3], circuit_filmf02[6];

void main(void)
{
        ger_f_for read_ger_file;
        read_ger_file.ger_to_f01(ger_file_name);
        read_ger_file.master_pad_line_file();
}

void ger_f_for::sort_small_par_x(char *line_file_name)
{
        int i, no_line, file_no;
        float x1,y1,x2,y2,dum_x;
        char new_file_name[30];

        file_no=atoi(&line_file_name[15]);
        fstream1=file_handling(line_file_name, "r+t", fstream1,
                " for reading line location.");
        while(fgetc(fstream1)!='\n');
        fscanf(fstream1, "%i", &no_line);
        strcpy(new_file_name, build_file_name("c:\\feature\\line", file_no,".f03"));
        fstream2=file_handling(new_file_name, "w+t", fstream2, " for writing line location.");
  fprintf(fstream2, "%s\n%i\n", new_file_name, no_line);
        for (i=0; i<no_line; ++i)
        {
                fscanf(fstream1, "%f%f%f%f", &x1, &y1, &x2, &y2);
                if (x1>x2)
                {
                        dum_x=x1;
                        x1=x2;
                        x2=dum_x;
                }
                fprintf(fstream2, "%8.3f %8.3f %8.3f %8.3f\n", x1,y1,x2,y2);
        }
        fclose(fstream1), fclose(fstream2);
}

void ger_f_for::sort_small_par_y(char *line_file_name)
{
        int i, no_line, file_no;
        float x1,y1,x2,y2,dum_y;
        char new_file_name[30];

        file_no=atoi(&line_file_name[15]);
        fstream1=file_handling(line_file_name, "r+t", fstream1,
                " for reading line location.");
        while(fgetc(fstream1)!='\n');
        fscanf(fstream1, "%i", &no_line);
        strcpy(new_file_name, build_file_name("c:\\feature\\line", file_no,".f03"));
        fstream2=file_handling(new_file_name, "w+t", fstream2, " for writing line location.");
  fprintf(fstream2, "%s\n%i\n", new_file_name, no_line);
```

```
        for (i=0; i<no_line; ++i)
        {
                fscanf(fstream1, "%f%f%f%f", &x1, &y1, &x2, &y2);
                if (y1>y2)
                {
                        dum_y=y1;
                        y1=y2;
                        y2=dum_y;
                }
                fprintf(fstream2, "%8.3f %8.3f %8.3f %8.3f\n", x1,y1,x2,y2);
        }
        fclose(fstream1), fclose(fstream2);
}

void ger_f_for::sort_pad_file(char *pad_file_name)
{
        int file_no, no_line=0, i, j;
  int pt_posit[Max_line_in_ger-200];
        float x[Max_line_in_ger-200], y[Max_line_in_ger-200], dum_x, dum_y;
  char dum_c=' ';

        file_no=atoi(&pad_file_name[14]);
        fstream1=file_handling(pad_file_name, "r+t", fstream1,
                " for reading pad location.");
        while(fgetc(fstream1)!='\n');
        while(dum_c!='*'){
                dum_c=fgetc(fstream1);
                if(dum_c=='*')break;
                else
                {
                        while(fgetc(fstream1)!='\n');
     ++no_line;
                }
  }
        rewind(fstream1);
        while(fgetc(fstream1)!='\n');

        for (i=0; i<no_line; ++i)
                fscanf(fstream1, "%f%f", &x[i], &y[i]);
        for (i=0; i<no_line; ++i)
        {
                for (j=i+1; j<no_line; ++j)
                {
                        if (x[i]>x[j])
                        {
                                dum_x=x[i];
                                dum_y=y[i];
                                x[i]=x[j];
                                y[i]=y[j];
                                x[j]=dum_x;
                                y[j]=dum_y;
```

```
                        }
                }
        }
        fclose(fstream1);

        // open file for output sorted for smallest x pads
        fstream1=file_handling(circuit_filmf02[file_no-1].pad_file_name,
                "w+t", fstream1, " for type .f02 pad location.");
        fprintf(fstream1, "%s\n%i\n", circuit_filmf02[file_no-1].pad_file_name,
                no_line);
        for (i=0; i<no_line; ++i)
                fprintf(fstream1, "%8.3f %8.3f\n", x[i]/Dec_pt, y[i]/Dec_pt);

        // sort pad based on smallest y
        for (i=0; i<no_line; ++i)   // store up the position
        pt_posit[i]=i;
        for (i=0; i<no_line; ++i)
        {
                for (j=i+1; j<no_line; ++j)
                {
                        if (y[i]>y[j])
                        {
                                dum_x = pt_posit[i];
                                dum_y = y[i];
                                pt_posit[i] = pt_posit[j];
                                y[i] = y[j];
                                pt_posit[j] = dum_x;
                                y[j] = dum_y;
                        }
                }
  }
        for (i=0; i<no_line; ++i)
                fprintf(fstream1,"%i ", pt_posit[i]);
        fclose(fstream1);
}

void ger_f_for::sort_line_file(char *line_file_name)
{
        int file_no, no_line=0, line1=0, line2=0, i;
        float x1, y1, x2, y2;
        char dum_c=' ';

        file_no=atoi(&line_file_name[15]);
  fstream1=file_handling(line_file_name, "r+t", fstream1,
                " for reading line location.");
        while(fgetc(fstream1)!='\n');
        while(dum_c!='*'){
                dum_c=fgetc(fstream1);
                if(dum_c=='*')break;
                else
                {
```

```
                        while(fgetc(fstream1)!='\n');
    ++no_line;
                }
}
        rewind(fstream1);
        while(fgetc(fstream1)!='\n');
        for (i=0; i<no_line; ++i)
        {
                fscanf(fstream1, "%f%f%f%f", &x1, &y1, &x2, &y2);
                if (y1==y2)
                        ++line1;
                else if (x1==x2)
                        ++line2;
                else
                        cout<<"ERROR in line file. Line not parallel. Data ignored.";
        }
        rewind(fstream1);

        fstream2=file_handling(circuit_filmf02[(file_no-1)*2].line_file_name,
                "w+t", fstream2, " for type X1.f02 line location.");
        fstream3=file_handling(circuit_filmf02[(file_no-1)*2+1].line_file_name,
                "w+t", fstream3, " for type X1.f02 line location.");
        fprintf(fstream2, "%s\n%i\n", circuit_filmf02[(file_no-1)*2].line_file_name,
                line1);
        fprintf(fstream3, "%s\n%i\n", circuit_filmf02[(file_no-1)*2+1].line_file_name,
                line2);
 while(fgetc(fstream1)!='\n');
        for (i=0; i<no_line; ++i)
        {
                fscanf(fstream1, "%f%f%f%f", &x1, &y1, &x2, &y2);
                if (y1==y2)
                        fprintf(fstream2,"%8.3f %8.3f %8.3f %8.3f\n",x1/Dec_pt,y1/Dec_pt,
                                x2/Dec_pt,y2/Dec_pt);
                else if (x1==x2)
                        fprintf(fstream3,"%8.3f %8.3f %8.3f %8.3f\n",x1/Dec_pt,y1/Dec_pt,
                                x2/Dec_pt,y2/Dec_pt);
                else
                        cout<<"ERROR in line file. Line not parallel. Data ignored.";
        }

        fclose(fstream1), fclose(fstream2), fclose(fstream3);
}

void ger_f_for::master_pad_line_file(void)
{
        int i, j, k;
 char dum_str[30];

        for (j=0; j<no_of_pad_file; ++j)
        {
                strcpy(circuit_filmf02[j].pad_file_name, "c:\\feature\\pad");
```

```
                strcat(circuit_filmf02[j].pad_file_name, itoa(j+1,dum_str,10));
                strcat(circuit_filmf02[j].pad_file_name, ".f02");
        }

        for (i=0; i<no_of_line_file; ++i)
        {

                for (j=0; j<2; ++j)
                {
                        strcpy(circuit_filmf02[i*2+j].line_file_name, "c:\\feature\\line");
                        strcat(circuit_filmf02[i*2+j].line_file_name, itoa(i*10+j+11,dum_str,10));
                        strcat(circuit_filmf02[i*2+j].line_file_name, ".f02");
                }
        }

  for (i=0; i<no_of_line_file; ++i)
                sort_line_file(circuit_film[i].line_file_name);

        for (i=0; i<no_of_pad_file; ++i)
                sort_pad_file(circuit_film[i].pad_file_name);

        for (i=0; i<no_of_line_file; ++i)
        {
                sort_small_par_x(circuit_filmf02[i*2+0].line_file_name);
                sort_small_par_y(circuit_filmf02[i*2+1].line_file_name);
        }

}

void ger_f_for::ger_to_f01(char *ger_file_name)
{
        int i=0, j, amp_code, d_code;
        int x, y, pre_x, pre_y;
        char dum_c, str1[50], pre_str[50], dum_str[20];
  char *str_y, *str_d;
        fstream1=file_handling(ger_file_name, "r+t", fstream1,
                " for reading Gerber file.");
        fstream2=file_handling("c:\\feature\\test.g01", "w+t", fstream2,
        " for storying temp. gerber data.");
        while (dum_c != 'M')
        {
                fscanf(fstream1, "%s", str1);
                strncpy(&dum_c, str1, 1);
                if (dum_c=='M')break;
                if (str1[0]=='D')
    {
                        i=atoi(&str1[1]);
                        if(i>100) fprintf(fstream2, "%s\n", str1);
                        else if(i==3)
                        {
        j=0;
```

```
                        do{
                        fputc(pre_str[j], fstream2);
                        ++j;
                        }while(pre_str[j]!='D');
                        fprintf(fstream2, "D03*\n");
                }
        }
        else
        {
                fprintf(fstream2, "%s\n", str1);
}
strcpy(pre_str, str1);
    }
    fprintf(fstream2, "M02*\n");

    fclose(fstream1);
    rewind(fstream2);
    fstream3=file_handling("c:\\feature\\testline.g01", "w+t", fstream3,
            " for storying line data.");
    fprintf(fstream3, "c:\\feature\\testline.g01\n");
    fstream4=file_handling("c:\\feature\\testpad.g01", "w+t", fstream4,
    " for storying pad data.");
    fprintf(fstream4, "c:\\feature\\testpad.g01\n");
    do{
            fscanf(fstream2, "%s", str1);
            strncpy(&dum_c, str1, 1);
            if (dum_c=='M')break;
            if ((dum_c=='X')||(dum_c=='Y'))
            {
                    if (dum_c=='X')
                    {
                            x=atoi(&str1[1]);
                            if((str_y=strchr(str1,'Y'))==NULL)
                            {
                                    y=pre_y;
        }
    else y=atoi(&str_y[1]);
                    }
                    else if (dum_c=='Y')
                    {
                            x=pre_x;
                            y=atoi(&str1[1]);
                    }
                    str_d=strchr(str1,'D');
                    d_code=atoi(&str_d[1]);
                    if(d_code==1)
                    {
                            fprintf(fstream3, "%i %i %i %i\n", pre_x,pre_y,x,y);
                    }
                    if(d_code==2);
                    if(d_code==3)
```

```
                        {
                                fprintf(fstream4, "%i %i\n", x, y);
    }
                        pre_x=x;
   pre_y=y;
  }

                if (dum_c=='D')
                {
                        i=atoi(&str1[1]);
                        if(i>100)
                        {
                                amp_code=i;
                                fprintf(fstream3, "D%i\n", i);
     fprintf(fstream4, "D%i\n", i);
                        }
                }
        } while(dum_c!='M');
        fprintf(fstream3, "M02");
fprintf(fstream4, "M02");
        fclose(fstream2), fclose(fstream3), fclose(fstream4);

        for (j=0; j<3; ++j)          // a max of THREE pad and THREE line file
        {
                strcpy(circuit_film[j].pad_file_name, "c:\\feature\\pad");
                strcat(circuit_film[j].pad_file_name, itoa(j+1,dum_str,10));
                strcat(circuit_film[j].pad_file_name, ".f01");

                strcpy(circuit_film[j].line_file_name, "c:\\feature\\line");
                strcat(circuit_film[j].line_file_name, itoa(j+1,dum_str,10));
                strcat(circuit_film[j].line_file_name, ".f01");
        }

        fstream1=file_handling(circuit_film[0].pad_file_name, "w+t", fstream1,
                " for type 1 pad location.");
        fprintf(fstream1, "%s\n", circuit_film[0].pad_file_name);
        fstream2=file_handling(circuit_film[1].pad_file_name, "w+t", fstream2,
                " for type 2 pad location.");
        fprintf(fstream2, "%s\n", circuit_film[1].pad_file_name);
        fstream3=file_handling(circuit_film[2].pad_file_name, "w+t", fstream3,
                " for type 3 pad location.");
        fprintf(fstream3, "%s\n", circuit_film[2].pad_file_name);
        fstream4=file_handling("c:\\feature\\testpad.g01", "r+t", fstream4,
                " for reading pad data.");
        while(fgetc(fstream4)!='\n');
dum_c=fgetc(fstream4);
        fscanf(fstream4, "%i", &amp_code);
        fscanf(fstream4, "%i%i", &x, &y);
while(fgetc(fstream4)!='\n');
        while(dum_c!='M')
        {
```

```
                if (amp_code == amp_look_up_table[0])
    fprintf(fstream1, "%i %i\n", x, y);
                else if (amp_code == amp_look_up_table[1])
                        fprintf(fstream2, "%i %i\n", x, y);
                else if (amp_code == amp_look_up_table[2])
                        fprintf(fstream3, "%i %i\n", x, y);
                else
                        cout<<"ERROR in amp_code. Location ignored!!!";

                fscanf(fstream4, "%s", str1);
                strncpy(&dum_c, str1, 1);
                if(dum_c=='D')
                {
                        amp_code=atoi(&str1[1]);
                        fscanf(fstream4, "%i%i", &x, &y);
  }
                else if(dum_c=='M')break;
                else
                {
      x=atoi(&str1[0]);
                        fscanf(fstream4, "%i", &y);
    fgetc(fstream4);
  }
        }
        fprintf(fstream1, "*"),      fprintf(fstream2, "*"), fprintf(fstream3, "*");
        fclose(fstream1), fclose(fstream2), fclose(fstream3), fclose(fstream4);

        fstream1=file_handling(circuit_film[0].line_file_name, "w+t", fstream1,
                " for type 1 line location.");
        fprintf(fstream1, "%s\n", circuit_film[0].line_file_name);
        fstream2=file_handling(circuit_film[1].line_file_name, "w+t", fstream2,
                " for type 2 line location.");
        fprintf(fstream2, "%s\n", circuit_film[1].line_file_name);
        fstream3=file_handling(circuit_film[2].line_file_name, "w+t", fstream3,
                " for type 3 line location.");
        fprintf(fstream3, "%s\n", circuit_film[2].line_file_name);

        fstream4=file_handling("c:\\feature\\testline.g01", "r+t", fstream4,
        " for reading line data.");
        while(fgetc(fstream4)!='\n');
dum_c=fgetc(fstream4);
        fscanf(fstream4, "%i", &amp_code);
        fscanf(fstream4, "%i%i%i%i", &pre_x, &pre_y, &x, &y);
while(fgetc(fstream4)!='\n');
        while(dum_c!='M')
        {
                if (amp_code == amp_look_up_table[0])
    fprintf(fstream1, "%i %i %i %i\n", pre_x, pre_y, x, y);
                else if (amp_code == amp_look_up_table[1])
                        fprintf(fstream2, "%i %i %i %i\n", pre_x, pre_y, x, y);
                else if (amp_code == amp_look_up_table[2])
```

```
                        fprintf(fstream3, "%i %i %i %i\n", pre_x, pre_y, x, y);
                else
                        cout<<"ERROR in amp_code. Location ignored!!!";

                fscanf(fstream4, "%s", str1);
                strncpy(&dum_c, str1, 1);
                if(dum_c=='D')
                {
                        amp_code=atoi(&str1[1]);
                        fscanf(fstream4, "%i%i%i%i", &pre_x, &pre_y, &x, &y);
    }
                else if(dum_c=='M')break;
                else
                {
        pre_x=atoi(&str1[0]);
                        fscanf(fstream4, "%i%i%i", &pre_y, &x, &y);
                        fgetc(fstream4);
    }
        }
  fprintf(fstream1, "*"),   fprintf(fstream2, "*"), fprintf(fstream3, "*");
        fclose(fstream1), fclose(fstream2), fclose(fstream3), fclose(fstream4);
}
```

```
/* :\ger\sort_line.cpp        date: 19July93 11:00am
        this file read line format file and then
                1.      sort different line into three files (format in integer)
                                c:\\feature\\line11.f02
                                c:\\feature\\line12.f02
                                c:\\feature\\line13.f02
                2.      the files will then sort according to
                                the starting pt of the line w.r.t. smallest x
                                the starting pt of the line w.r.t. smallest y
                                the end pt of the line w.r.t. smallest x
                                the end pt of the lines w.r.t. smallest y
                                and then output back into the three files called
                                c:\\feature\\line11.f03                 (format will be in float pt)
                                c:\\feature\\line12.f03
                                c:\\feature\\line13.f03
*/

#include <stdio.h>
#include <stdlib.h>         // for exit function
#include <iostream.h>       // for cout use
#include <delfil.h>                 // self developed function to delete file

#define Max_line_in_line_file 150
#define No_of_line_file no_of_line_file=2   // set number of pad file
#define Dec_pt 10                   // decimal pt of the file 10 means 1 point
FILE *fstream1, *fstream2, *fstream3;
static char fname_line1f02[] = "c:\\feature\\line1.f02";

static char fname_line11f02[] = "c:\\feature\\line11.f02";
static char fname_line12f02[] = "c:\\feature\\line12.f02";
static char fname_line13f02[] = "c:\\feature\\line13.f02";

static char fname_line11f03[] = "c:\\feature\\line11.f03";
static char fname_line12f03[] = "c:\\feature\\line12.f03";
static char fname_line13f03[] = "c:\\feature\\line13.f03";

int no_line1;

class line_f_for01
{
        float x1cor;
        float y1cor;
        float x2cor;
        float y2cor;
  public:
        void sort_line(void);
};
class line_f_for02
{
        float x1cor;
        float y1cor;
```

```
        float x2cor;
        float y2cor;
  public:
        void sort_line(void);
};

void main(void)
{
        line_f_for01 dummy_name1;
        dummy_name1.sort_line();
        line_f_for02 dummy_name2;
  dummy_name2.sort_line();
}

void line_f_for02::sort_line()
{
        line_f_for02 xy_pt[Max_line_in_line_file];

        cout<<"Default open file "<<fname_line11f02<<
                " with // coord x to find smallest y \n";
        if((fstream1=fopen(fname_line11f02,"r+t"))==NULL)
        {
                cout << "** ERROR: Can't open line file called " <<
                        fname_line11f02 << " for input.";
                exit (1);
        }
  int i=0, i1, i2;
  do{
                fscanf (fstream1, "%f%f%f%f", &xy_pt[i].x1cor,
                        &xy_pt[i].y1cor, &xy_pt[i].x2cor, &xy_pt[i].y2cor);
                ++i;            // count no of line in the file
        } while (fgetc(fstream1) != EOF);
        fclose(fstream1);
        i = i-1;
        // sort starting point of the line w.r.t. x
        float x1, y1, x2, y2;
        for (i1=0; i1<i; ++i1)
        {
                // check whether starting pt of co-ordinate x,
                // if larger than finishing pt, revise the co-ordinate
                if (xy_pt[i1].x1cor>xy_pt[i1].x2cor)
    {
                        x1 = xy_pt[i1].x1cor;
                        xy_pt[i1].x1cor = xy_pt[i1].x2cor;
                        xy_pt[i1].x2cor = x1;
                        // do not have to change y coord as they are the same // to y
                }
        }
        for(i1=0; i1<i; ++i1)
        {
    for(i2=i1+1; i2<i; ++i2)
```

```
                {
                        if(xy_pt[i1].y1cor > xy_pt[i2].y1cor)
                        {
                                x1 = xy_pt[i1].x1cor;       //?? do not have to change y coord??
                                y1 = xy_pt[i1].y1cor;
                                x2 = xy_pt[i1].x2cor;
                                y2 = xy_pt[i1].y2cor;
                                xy_pt[i1].x1cor = xy_pt[i2].x1cor;
        xy_pt[i1].y1cor = xy_pt[i2].y1cor;
                                xy_pt[i1].x2cor = xy_pt[i2].x2cor;
        xy_pt[i1].y2cor = xy_pt[i2].y2cor;
                                xy_pt[i2].x1cor = x1;
                                xy_pt[i2].y1cor = y1;
                                xy_pt[i2].x2cor = x2;
                                xy_pt[i2].y2cor = y2;
                        }
                }
        }

// store up the data into a file
        cout<<"Default open file "<<fname_line11f03<<" to store up data.\n";
        if((fstream1=fopen(fname_line11f03,"w+t"))==NULL)
        {
                cout << "** ERROR: Can't open line file called " <<
                        fname_line11f03 << " for output.";
                exit (1);
        }
fprintf(fstream1, "%i\n", i);       // number of line in file
        for (i1=0; i1<i; ++i1)
        {
                fprintf(fstream1, "%7.3f %7.3f %7.3f %7.3f\n",
                        xy_pt[i1].x1cor,  xy_pt[i1].y1cor,
                        xy_pt[i1].x2cor,  xy_pt[i1].y2cor);
        }
        fclose(fstream1);
        file_delete(fname_line11f02);

        // for line // to x sort according to smallest y
        cout<<"Default open file "<<fname_line12f02<<
                " with // coord y to find smallest x \n";
        if((fstream1=fopen(fname_line12f02,"r+t"))==NULL)
        {
                cout << "** ERROR: Can't open line file called " <<
                        fname_line12f02 << " for input.";
                exit (1);
        }
i=0;            // initialized line number to zero
do{
                fscanf (fstream1, "%f%f%f%f", &xy_pt[i].x1cor,
                        &xy_pt[i].y1cor, &xy_pt[i].x2cor, &xy_pt[i].y2cor);
                ++i;            // count no of line in the file
```

```
        } while (fgetc(fstream1) != EOF);
        fclose(fstream1);
        i = i-1;
        for (i1=0; i1<i; ++i1)
        {
                // check whether starting pt of co-ordinate y,
                // if larger than finishing pt, revise the co-ordinate
                if (xy_pt[i1].y1cor>xy_pt[i1].y2cor)
  {
                        y1 = xy_pt[i1].y1cor;
                        xy_pt[i1].y1cor = xy_pt[i1].y2cor;
                        xy_pt[i1].y2cor = y1;
                        // do not have to change x coord as they are the same // to x
                }
        }
        for(i1=0; i1<i; ++i1)
        {
  for(i2=i1+1; i2<i; ++i2)
                {
                        if(xy_pt[i1].x1cor > xy_pt[i2].x1cor)
                        {
                                x1 = xy_pt[i1].x1cor;
                                y1 = xy_pt[i1].y1cor;
                                x2 = xy_pt[i1].x2cor;
                                y2 = xy_pt[i1].y2cor;
                                xy_pt[i1].x1cor = xy_pt[i2].x1cor;
      xy_pt[i1].y1cor = xy_pt[i2].y1cor;
                                xy_pt[i1].x2cor = xy_pt[i2].x2cor;
      xy_pt[i1].y2cor = xy_pt[i2].y2cor;
                                xy_pt[i2].x1cor = x1;
                                xy_pt[i2].y1cor = y1;
                                xy_pt[i2].x2cor = x2;
                                xy_pt[i2].y2cor = y2;
                        }
                }
        }

// store up the data into a file
        cout<<"Default open file "<<fname_line12f03<<" to store up data.\n";
        if((fstream1=fopen(fname_line12f03,"w+t"))==NULL)
        {
                cout << "** ERROR: Can't open line file called " <<
                        fname_line12f03 << " for output.";
                exit (1);
        }
fprintf(fstream1, "%i\n", i);        // number of line in file
        for (i1=0; i1<i; ++i1)
        {
                fprintf(fstream1, "%7.3f %7.3f %7.3f %7.3f\n",
                        xy_pt[i1].x1cor,  xy_pt[i1].y1cor,
                        xy_pt[i1].x2cor,  xy_pt[i1].y2cor);
```

```
        }
        fclose(fstream1);
        file_delete(fname_line12f02);

        // for line not // to x or y
        cout<<"Default open file "<<fname_line13f02<<
                " with coord not // to x or y to change format\n";
        if((fstream1=fopen(fname_line13f02,"r+t"))==NULL)
        {
                cout << "** ERROR: Can't open line file called " <<
                        fname_line13f02 << " for input.";
                exit (1);
        }
 i=0;           // initialized line number to zero
 do{
                fscanf (fstream1, "%f%f%f%f", &xy_pt[i].x1cor,
                        &xy_pt[i].y1cor, &xy_pt[i].x2cor, &xy_pt[i].y2cor);
                ++i;            // count no of line in the file
        } while (fgetc(fstream1) != EOF);
        fclose(fstream1);
        i = i-1;
        // store up the data into a file
        cout<<"Default open file "<<fname_line13f03<<" to store up data.\n";
        if((fstream1=fopen(fname_line13f03,"w+t"))==NULL)
        {
                cout << "** ERROR: Can't open line file called " <<
                        fname_line13f03 << " for output.";
                exit (1);
        }
 fprintf(fstream1, "%i\n", i);      // number of line in file
        for (i1=0; i1<i; ++i1)
        {
                fprintf(fstream1, "%7.3f %7.3f %7.3f %7.3f\n",
                        xy_pt[i1].x1cor,  xy_pt[i1].y1cor,
                        xy_pt[i1].x2cor,  xy_pt[i1].y2cor);
        }
        fclose(fstream1);
        file_delete(fname_line13f02);
}

void line_f_for01::sort_line()
{
        line_f_for01 xy_pt[Max_line_in_line_file];
        int i = 0, i1, i2;
        char  dum_char1, dum_char2;
        cout << "Default open file "<<fname_line1f02<<
                " to sort line.\n";
        if ((fstream1 = fopen(fname_line1f02, "r+t")) == NULL)
        {
                cout<<"** ERROR: Can't open pad file called "<<
                        fname_line1f02<<" for input.";
```

```
                exit (1);
        }
        fscanf(fstream1, "%i", &no_line1);
        cout <<" no_line1 ="<<no_line1<<"\n";
do{
                fscanf (fstream1, "%f%f%f%f", &xy_pt[i].x1cor,
                        &xy_pt[i].y1cor, &xy_pt[i].x2cor, &xy_pt[i].y2cor);
                ++i;
        } while (fgetc (fstream1) != EOF);
        fclose(fstream1);
        cout << "Default open file "<<fname_line11f02<<
                " to store line parallel to x.\n";
        if((fstream1=fopen(fname_line11f02,"w+t"))==NULL)
        {
                cout << "** ERROR: Can't open line file called " <<
                        fname_line11f02 << " for output.";
                exit (1);
        }
        cout<<"Default open file "<<fname_line12f02<<
                " to store line parallel to y.\n";
        if((fstream2=fopen(fname_line12f02,"w+t"))==NULL)
        {
                cout << "** ERROR: Can't open line file called " <<
                        fname_line12f02 << " for output.";
                exit (1);
        }
        cout << "Default open file "<<fname_line13f02<<
                " to store line not parallel to x or y.\n";
if((fstream3=fopen(fname_line13f02,"w+t"))==NULL)
        {
                cout << "** ERROR: Can't open line file called " <<
                        fname_line13f02 << " for output.";
                exit (1);
        }

        for (i1=0; i1<no_line1; ++i1)
        {
                if(xy_pt[i1].x1cor==xy_pt[i1].x2cor)          // line parallel to y axis
                        fprintf(fstream2, "%f %f %f %f\n", xy_pt[i1].x1cor,
                                xy_pt[i1].y1cor, xy_pt[i1].x2cor, xy_pt[i1].y2cor);
                else
                {
                        if(xy_pt[i1].y1cor==xy_pt[i1].y2cor)          // line parallel to x axis
                                fprintf (fstream1, "%f %f %f %f\n", xy_pt[i1].x1cor,
                                        xy_pt[i1].y1cor, xy_pt[i1].x2cor, xy_pt[i1].y2cor);
                        //line not parallel to x or y axis
                        else
                                fprintf (fstream3, "%f %f %f %f\n", xy_pt[i1].x1cor,
                                xy_pt[i1].y1cor, xy_pt[i1].x2cor, xy_pt[i1].y2cor);
  }
        }
```

```
        fclose(fstream1);
        fclose(fstream2);
        fclose(fstream3);
}
```

```
/*

        b:\ger\pad_dis.cpp                      date:26uly93 16:00
        program to
        1.      find the distance between pads of the same file and
        2.      the distance between pads of two different file
                files input are called
                        f_name_pad1f02[] = "c:\\feature\\pad1.f02"
                        f_name_pad2f02[] = "c:\\feature\\pad2.f02"
                        f_name_pad3f02[] = "c:\\feature\\pad3.f02"
        Files output are called
                        f_name_pad123dis[] ="c:\\feature\\pad123.dis"
                        f_name_pad12dis[] = "c:\\feature\\pad12.dis"
                        f_name_pad13dis[] = "c:\\feature\\pad13.dis"
                        f_name_pad23dis[] = "c:\\feature\\pad23.dis"
        This program can only accept a max. of three pad files.
*/

#include <stdio.h>
#include <stdlib.h>                     // for exit function
#include <iostream.h>   // for cout use
#include <math.h>       // for math function use
#include <string.h>                     // for string handling
#include <io.h>                         // for checking file
#include <delfile.h>    // self developed function to delete file
#include <finddis.h>    // self developed function to find distance
#include <handfile.h>// self developed function to handle file

#define Max_line_in_pad_file 100        // max no of line in pad file

static int no_of_pad_file=3;            // THIS NUMBER IS THE NO OF PAD FILE

static char f_name_pad1f02[] = "c:\\feature\\pad1.f02";
static char f_name_pad2f02[] = "c:\\feature\\pad2.f02";
static char f_name_pad3f02[] = "c:\\feature\\pad3.f02";

static char f_name_pad123dis[] ="c:\\feature\\pad123.dis";
static char f_name_pad12dis[] = "c:\\feature\\pad12.dis";
static char f_name_pad13dis[] = "c:\\feature\\pad13.dis";
static char f_name_pad23dis[] = "c:\\feature\\pad23.dis";

int pad_no1=0, pad_no2=0, pad_no3=0;

class pad_dis
{
        float x;
        float y;
        float loc_dis;
        public:
        void search_all_pad_file(int no_of_pad_file);
        void find_pad_dis(char *f_name1, char *f_name2, char *ind);
```

```
        void find_pad_dis(char *f_name1, char *f_name2, char *f_name3, char *ind);
};

void main(void)
{
        pad_dis dummy1;
        dummy1.search_all_pad_file(no_of_pad_file);
}

void pad_dis::search_all_pad_file(int no_of_pad_file)
{
        if(access(f_name_pad123dis, 0)==0)file_delete(f_name_pad123dis);
        if(access(f_name_pad12dis, 0)==0)file_delete(f_name_pad12dis);
        if(access(f_name_pad13dis, 0)==0)file_delete(f_name_pad13dis);
        if(access(f_name_pad23dis, 0)==0)file_delete(f_name_pad23dis);
        switch(no_of_pad_file)
        {
                case 3:
                        find_pad_dis(f_name_pad3f02, f_name_pad123dis, "pad3");
                        find_pad_dis(f_name_pad1f02, f_name_pad3f02, f_name_pad13dis, "pad13");
    find_pad_dis(f_name_pad2f02, f_name_pad3f02, f_name_pad23dis, "pad23");
                case 2:
                        find_pad_dis(f_name_pad2f02, f_name_pad123dis, "pad2");
    find_pad_dis(f_name_pad1f02, f_name_pad2f02, f_name_pad12dis, "pad12");
                case 1:
                        find_pad_dis(f_name_pad1f02, f_name_pad123dis, "pad1");
    break;
                default:
                        cout<<"**Error Number of pad file not correct.\n";
                        break;
        }
}

void pad_dis::find_pad_dis(char *f_name1, char *f_name2, char *ind)
{
        pad_dis pad_cor[Max_line_in_pad_file];
        int i,i1=0,j,k;
        float dis;

        fstream1=file_handling(f_name1, "r+t", fstream1, " to find dis. between pads.");
  while(fgetc(fstream1)!='\n');
        fscanf(fstream1, "%i", &i);
        for(j=0; j<i; ++j)
  {
                fscanf(fstream1, "%f %f ", &pad_cor[j].x,   &pad_cor[j].y);
        }
        fclose(fstream1);

        fstream1=file_handling(f_name2, "a+t", fstream1, " to store dis. between pads.");
  fprintf(fstream1, "%s\n", f_name2);
        for (j=i-1; j>0; --j)
```

```
                i1+=j;
        fprintf(fstream1, "%s %i %i\n", ind, i, i1);
        for(j=0; j<i; ++j)
        {
                for(k=j+1; k<i; ++k)
                {
                        dis = find_dis(pad_cor[j].x,pad_cor[j].y,pad_cor[k].x,pad_cor[k].y);
    fprintf(fstream1, "%7.3f \n", dis);
                }
        }
        fclose(fstream1);
}

void pad_dis::find_pad_dis(char *f_name1, char *f_name2,
        char *f_name3,   char *ind)
{
        pad_dis pad_cor1[Max_line_in_pad_file];
        pad_dis pad_cor2[Max_line_in_pad_file];
        int p1,p2,i1=0,j,k;
        float dis;

        fstream1=file_handling(f_name1, "r+t", fstream1, " to find dis. between pads.");
  while(fgetc(fstream1)!='\n');
        fscanf(fstream1, "%i", &p1);
        for(j=0; j<p1; ++j)
                fscanf(fstream1, "%f %f ", &pad_cor1[j].x, &pad_cor1[j].y);
        fclose(fstream1);

        fstream1=file_handling(f_name2, "r+t", fstream1, " to find dis. between pads.");
        while(fgetc(fstream1)!='\n');
  fscanf(fstream1, "%i", &p2);
        for(j=0; j<p2; ++j)
                fscanf(fstream1, "%f %f ", &pad_cor2[j].x, &pad_cor2[j].y);
        fclose(fstream1);

        fstream1=file_handling(f_name3, "a+t", fstream1, " to find dis. between pads.");
        fprintf(fstream1, "%s\n", f_name3);
  i1=p1*p2;
        fprintf(fstream1, "%s %i %i %i\n", ind, p1, p2, i1);
        for(j=0; j<p1; ++j)
        {
                for(k=0; k<p2; ++k)
                {
                        dis = find_dis(pad_cor1[j].x, pad_cor1[j].y, pad_cor2[k].x,
                                pad_cor2[k].y);
    fprintf(fstream1, "%7.3f \n", dis);
                }
        }
        fclose(fstream1);
}
```

```
/*

        (b):\ger\padlidis.cpp                   date:30ly93 22:00
        program to find the distance between pads and lines of files
        pad files input are called
                        f_name_pad1f02[] = "c:\\feature\\pad1.f02"
                        f_name_pad2f02[] = "c:\\feature\\pad2.f02"
                        f_name_pad3f02[] = "c:\\feature\\pad3.f02"
        line files input are called (included in header file: l1133f03.h)
                        f_name_line11f03[] = "c:\\feature\\line11.f02"
                        f_name_line12f03[] = "c:\\feature\\line12.f02"
                        f_name_line21f03[] = "c:\\feature\\line21.f02"
                        f_name_line22f03[] = "c:\\feature\\line22.f02"
                        f_name_line31f03[] = "c:\\feature\\line31.f02"
                        f_name_line32f03[] = "c:\\feature\\line32.f02"
        The distance between the pads and line are storied into file called
                        f_name_pad_line1dis[] = "c:\\feature\\pad_line1.dis"
                        f_name_pad_line2dis[] = "c:\\feature\\pad_line2.dis"
                        f_name_pad_line3dis[] = "c:\\feature\\pad_line3.dis"
        This program can only accept a max of THREE pads files and THREE line files.

*/

#include <stdio.h>
#include <stdlib.h>             // for exit function
#include <iostream.h>   // for cout use
#include <math.h>       // for math function use
#include <string.h>             // for string handling
#include <io.h>                 // for checking file
#include <delfile.h>    // self developed function to delete file
#include <finddis.h>    // self developed function to find distance
#include <handfile.h>   // self developed function to handle file
#include <l1132f02.h>   // include line file names

#define Max_line_in_pad_file 130        // max no of line in pad file
#define Max_line_in_line_file 130 // max no of line in line file

static int no_of_pad_file=3;        // THIS NUMBER IS THE NO OF PAD FILE
static int no_of_line_file=3;       // THIS NUMBER IS THE NO OF LINE FILE

static float width_of_line[3]={1.0,1.2,1.4};
static float dia_of_pad[3]={0.8, 1.0, 1.2};

static char f_name_pad1f02[] = "c:\\feature\\pad1.f02";
static char f_name_pad2f02[] = "c:\\feature\\pad2.f02";
static char f_name_pad3f02[] = "c:\\feature\\pad3.f02";
static char f_name_pad[]    = "c:\\feature\\pad";

static char f_name_line[]    = "c:\\feature\\line";

static char         f_name_pad_line1dis[] = "c:\\feature\\padline1.dis";
```

```
static char        f_name_pad_line2dis[] = "c:\\feature\\padline2.dis";
static char        f_name_pad_line3dis[] = "c:\\feature\\padline3.dis";

char f_name1[28], f_name2[28];

void pad_line_files(int no_of_pad_file, int no_of_line_file);
void pad_line_dis(char *f_name1, char *f_name2, char *f_name3,
        char *f_name5, int i, int k);

void main(void)
{
        pad_line_files(no_of_pad_file, no_of_line_file);
}

void pad_line_files(int no_of_pad_file, int no_of_line_file)
{
        int i, k;
        if(access(f_name_pad_line1dis,0)==0)file_delete(f_name_pad_line1dis);
        if(access(f_name_pad_line2dis,0)==0)file_delete(f_name_pad_line2dis);
        if(access(f_name_pad_line3dis,0)==0)file_delete(f_name_pad_line3dis);

        for (i=1; i<no_of_pad_file+1; ++i)
        {
                if (i==1)strcpy(f_name1, f_name_pad1f02);
                if (i==2)strcpy(f_name1, f_name_pad2f02);
    if (i==3)strcpy(f_name1, f_name_pad3f02);
                switch (no_of_line_file)
                {
                        case 3:
                                pad_line_dis(f_name1, f_name_line31f02, f_name_line32f02,
                                        f_name_pad_line3dis, i-1, k-1);    // neet to - 1 as array

        // starts from o!!!!!!!!
                        case 2:
                                pad_line_dis(f_name1, f_name_line21f02, f_name_line22f02,
                                        f_name_pad_line2dis, i-1, k-1);
                        case 1:
                                pad_line_dis(f_name1, f_name_line11f02, f_name_line12f02,
                                        f_name_pad_line1dis, i-1, k-1);
                                break;
                        default:
                                cout<<"ERROR No of line file is not correct.\n";
                                break;
                }
        }
}

void pad_line_dis(char *f_name1, char *f_name2, char *f_name3,
        char *f_name5, int pad_file_no, int line_file_no)
{
        int no_of_pad, no_of_line;
```

```
        int i, j;
        float pad_no[Max_line_in_pad_file][2];
        float line_no[Max_line_in_line_file][4];

        fstream1=file_handling(f_name1,"r+t",fstream1,
                " for reading pad location.");
  while(fgetc(fstream1)!='\n');
        fscanf(fstream1, "%i", &no_of_pad);
        for (i=0; i<no_of_pad; ++i)
                fscanf(fstream1,"%f %f", &pad_no[i][0], &pad_no[i][1]);
        fclose(fstream1);

        fstream1=file_handling(f_name2,"r+t",fstream1,
                " for reading line location.");
  while(fgetc(fstream1)!='\n');
        fscanf(fstream1, "%i", &no_of_line);
        for(j=0; j<no_of_line; ++j)
        fscanf(fstream1, "%f%f%f%f", &line_no[j][0],
                &line_no[j][1], &line_no[j][2], &line_no[j][3]);
        fclose(fstream1);

        fstream2=file_handling(f_name5, "a+t", fstream2,
                " for storying pad and line distance.");
        fprintf(fstream2, "%s\n", f_name5);
  fprintf(fstream2, "%s %i %s %i\n", f_name1, no_of_pad, f_name2, no_of_line);        float dis = 0.0;

  float xl, yl;       // for storage of line co-ordinates
        for (i=0; i<no_of_pad; ++i)
  {
                for(j=0; j<no_of_line; ++j)
                {
                        if ((pad_no[i][0]>=line_no[j][0])&&(pad_no[i][0]<=line_no[j][2]))
                        {
/*                              cout<<"pad x = "<<pad_no[i][0]<<" pad y = "<<pad_no[i][1]<<"\n";
                                cout<<"line x1 = "<<line_no[j][0]<<" line y1 =
"<<line_no[j][1]<<"\n";
        cout<<"line x2 = "<<line_no[j][2]<<" line y2 = "<<line_no[j][3]<<"\n";
*/                              dis=fabs(pad_no[i][1]-line_no[j][1])-dia_of_pad[pad_file_no]/2-
                                        width_of_line[line_file_no]/2;
//      cout<<" dis1 = "<<dis<<"\n";
      }
                        else if (pad_no[i][0]<line_no[j][0])
                        {
                                xl = line_no[j][0];
                                yl = line_no[j][1];
/*                              cout<<"pad x = "<<pad_no[i][0]<<" pad y = "<<pad_no[i][1]<<"\n";
                                cout<<"line x1 = "<<line_no[j][0]<<" line y1 =
"<<line_no[j][1]<<"\n";
        cout<<"line x2 = "<<line_no[j][2]<<" line y2 = "<<line_no[j][3]<<"\n";
*/                              dis = sqrt(fabs((pad_no[i][0]-xl)*(pad_no[i][0]-xl))+
```

```
                                        fabs((pad_no[i][1]-yl)*(pad_no[i][1]-yl)))-
dia_of_pad[pad_file_no]/2-
                                        width_of_line[line_file_no]/2;
//                              cout<<" dis2 = "<<dis<<"\n";
                        }
                        else if (pad_no[i][0]>line_no[j][2])
                        {
                                float dis1, dis2;   // for debugging purpose
                                xl = line_no[j][2];
                                yl = line_no[j][3];
                                dis1 = sqrt(fabs((pad_no[i][0]-xl)*(pad_no[i][0]-xl))+
                                        fabs((pad_no[i][1]-yl)*(pad_no[i][1]-yl)));
                                dis2= dia_of_pad[pad_file_no]/2+width_of_line[line_file_no]/2;
        dis = dis1-dis2;
/*                              cout<<"pad x = "<<pad_no[i][0]<<" pad y = "<<pad_no[i][1]<<"\n";
                                cout<<"line x1 = "<<line_no[j][0]<<" line y1 =
"<<line_no[j][1]<<"\n";
                cout<<"line x2 = "<<line_no[j][2]<<" line y2 = "<<line_no[j][3]<<"\n";
                                cout<<" dis1 = "<<dis1<<" dis2 = "<<dis2<<" dis3 = "<<dis<<"\n";
*/                      }
                        else
                        {
                                cout<<"ERROR in finding pad and line distance.\n";
                                dis = 99999;
                        }
                        fprintf(fstream2, "%8.3f\n", dis);
    }
        }
        // for line file // to y

        fstream1=file_handling(f_name3, "r+t", fstream1,
                " for reading line location.");
  while(fgetc(fstream1)!='\n');
        fscanf(fstream1, "%i", &no_of_line);
        for(j=0; j<no_of_line; ++j)
                fscanf(fstream1, "%f%f%f%f", &line_no[j][0],
                        &line_no[j][1], &line_no[j][2], &line_no[j][3]);
        fclose(fstream1);

        fprintf(fstream2, "%s\n", f_name5);
        fprintf(fstream2, "%s %i %s %i\n", f_name1, no_of_pad, f_name3, no_of_line);

        for (i=0; i<no_of_pad; ++i)
  {
                for(j=0; j<no_of_line; ++j)
                {
                        if ((pad_no[i][1]>=line_no[j][1])&&(pad_no[i][1]<=line_no[j][3]))
     {
                                dis=fabs(pad_no[i][0]-line_no[j][0])-dia_of_pad[pad_file_no]/2-
                                        width_of_line[line_file_no]/2;
/*                              cout<<"pad x = "<<pad_no[i][0]<<" pad y = "<<pad_no[i][1]<<"\n";
```

```
                                        cout<<"line x1 = "<<line_no[j][0]<<" line y1 =
"<<line_no[j][1]<<"\n";
        cout<<"line x2 = "<<line_no[j][2]<<" line y2 = "<<line_no[j][3]<<"\n";
                                        cout<<" dis4 = "<<dis<<"\n";
*/                              }
      else if (pad_no[i][1]<line_no[j][1])
      {
                                        x1 = line_no[j][0];
                                        y1 = line_no[j][1];
                                        dis = sqrt(fabs((pad_no[i][0]-x1)*(pad_no[i][0]-x1))+
                                                fabs((pad_no[i][1]-y1)*(pad_no[i][1]-y1)))-
dia_of_pad[pad_file_no]/2-
                                                width_of_line[line_file_no]/2;
/*                                      cout<<"pad x = "<<pad_no[i][0]<<" pad y = "<<pad_no[i][1]<<"\n";
                                        cout<<"line x1 = "<<line_no[j][0]<<" line y1 =
"<<line_no[j][1]<<"\n";
        cout<<"line x2 = "<<line_no[j][2]<<" line y2 = "<<line_no[j][3]<<"\n";
                                        cout<<" dis5 = "<<dis<<"\n";
*/      }
      else if (pad_no[i][1]>line_no[j][3])
                                {
                                        x1 = line_no[j][2];
                                        y1 = line_no[j][3];
                                        dis = sqrt(fabs((pad_no[i][0]-x1)*(pad_no[i][0]-x1))+
                                                fabs((pad_no[i][1]-y1)*(pad_no[i][1]-y1)))-
dia_of_pad[pad_file_no]/2-
                                                width_of_line[line_file_no]/2;
/*                                      cout<<"pad x = "<<pad_no[i][0]<<" pad y = "<<pad_no[i][1]<<"\n";
                                        cout<<"line x1 = "<<line_no[j][0]<<" line y1 =
"<<line_no[j][1]<<"\n";
        cout<<"line x2 = "<<line_no[j][2]<<" line y2 = "<<line_no[j][3]<<"\n";
                                        cout<<" dis6 = "<<dis<<"\n";
*/                              }
                                else
      {
                                        cout<<"ERROR in finding pad and line distance.\n";
                                        dis = 99999;
                                }
                                fprintf(fstream2, "%8.3f\n", dis);
                        }
          }
          fclose(fstream2);
}
```

```
/*

        (b):\ger\linlidis.cpp                    date:29July93 22:00
        program to find the distance between lines of same file and lines of two
        different line files.  Line files input are called (included in header file
        called 11133f03.h)
                        f_name_line11f03[] = "c:\\feature\\line11.f03"
                        f_name_line12f03[] = "c:\\feature\\line12.f03"
                        f_name_line21f03[] = "c:\\feature\\line21.f03"
                        f_name_line22f03[] = "c:\\feature\\line22.f03"
                        f_name_line31f03[] = "c:\\feature\\line31.f03"
                        f_name_line32f03[] = "c:\\feature\\line32.f03"
        The distance between the lines are storied into file called
                        f_name_line123dis[] = "c:\\feature\\line123.dis"
                        f_name_line1dis[]   = "c:\\feature\\line1.dis"
                        f_name_line2dis[]   = "c:\\feature\\line2.dis"
                        f_name_line3dis[]   = "c:\\feature\\line3.dis"
                        f_name_line12dis[]  = "c:\\feature\\line12.dis"
                        f_name_line13dis[]  = "c:\\feature\\line13.dis"
                        f_name_line23dis[]  = "c:\\feature\\line23.dis"
        This program can only accept a max of THREE different width line files.

*/

#include <stdio.h>
#include <stdlib.h>             // for exit function
#include <iostream.h>   // for cout use
#include <math.h>       // for math function use
#include <string.h>             // for string handling
#include <io.h>                 // for checking file
#include <delfile.h>    // self developed function to delete file
#include <handfile.h>   // self developed function to handle file
#include <bld_f_n.h>            // self developed function to build up file name
#include <finddis.h>    // self developed function to find distance
#include <minmax.h>             // self dveloped function to find min and max

#define Max_line_in_line_file 130 // max no of line in line file


static char f_name_line[]               = "c:\\feature\\line";
static char         f_name_line123dis[] = "c:\\feature\\line123.dis";
static char f_name_line1dis[]   = "c:\\feature\\line1.dis";
static char f_name_line2dis[]   = "c:\\feature\\line2.dis";
static char f_name_line3dis[]   = "c:\\feature\\line3.dis";
static char f_name_line12dis[]  = "c:\\feature\\line12.dis";
static char f_name_line13dis[]  = "c:\\feature\\line13.dis";
static char f_name_line23dis[]  = "c:\\feature\\line23.dis";
static char char_dis[]                          = ".dis";
static float width_of_line[3]={1.0, 1.2, 1.4};

char f_name1[28], f_name2[28], f_name3[28];
```

```
int no_of_line_file=3;                    // set number of line file = 3
float min_of_4_dis(float, float, float, float, float, float, float, float);
float line_width_file_name(char *fname);

class line_formatf03
{
        float x1;
        float y1;
        float x2;
        float y2;
        public:
  void master_find_line_file_dis(int no_of_line_file);
        void line_dis_same_file(char *f_name1, char *f_name2);
        void line_dis_dif_file(char *f_name1, char *f_name2, char *f_name3);
        float line_dis(float, float, float, float, float, float, float, float);
};

void main(void)
{
        float dis;
        line_formatf03 dum_name;
        dum_name.master_find_line_file_dis(no_of_line_file);
}

float line_formatf03::line_dis(float l1x1, float l1y1, float l1x2, float l1y2,
        float l2x1, float l2y1, float l2x2, float l2y2)
{
        float dis=99999, d1, d2;
//      cout<<l1x1<<" "<<l1y1<<" "<<l1x2<<" "<<l1y2<<" , "<<
//              l2x1<<" "<<l2y1<<" "<<l2x2<<" "<<l2y2<<" ";
        // perpendicular to each other
        if ((l1y1==l1y2)&&(l2x1==l2x2))
        {
                if((l1y1>=l2y1)&&(l1y1<=l2y2))
                {
                        d1=abs(l2x1-l1x1);
                        d2=abs(l2x1-l1x2);
                        dis=min(d1,d2);
//                      cout<<" case11 ";
                }
                else
                {
                        dis=min_of_4_dis(l1x1, l1y1, l1x2, l1y2, l2x1, l2y1, l2x2, l2y2);
//                      cout<<" case12 ";
                }
        }
        if ((l1x1==l1x2)&&(l2y1==l2y2))
        {
                if((l1y1<=l2y1)&&(l1y2>=l2y1))
                {
```

```
                        d1=abs(l1x1-l2x1);
                        d2=abs(l1x1-l2x2);
                        dis=min(d1,d2);
//                      cout<<" case13 ";
                }
                else
                {
                        dis=min_of_4_dis(l1x1, l1y1, l1x2, l1y2, l2x1, l2y1, l2x2, l2y2);
//                      cout<<" case14 ";
                }
        }
        // parallel to each other
        if ((l1y1==l1y2)&&(l2y1==l2y2))
        {
                if(((l1x1<=l2x1)&&(l1x2>=l2x1))||((l1x1<=l2x2)&&(l1x2>=l2x2)))
                {
                        dis=abs(l1y1-l2y1);
//                      cout<<" case21 ";
                }
                else
                {
                        dis=min_of_4_dis(l1x1, l1y1, l1x2, l1y2, l2x1, l2y1, l2x2, l2y2);
//                      cout<<" case22 ";
                }
        }
        if ((l1x1==l1x2)&&(l2x1==l2x2))
        {
        if(((l1y1<=l2y1)&&(l1y2>=l2y1))||((l1y1<=l1y2)&&(l1y2>=l2y2)))
                {
                        dis=abs(l1x1-l2x1);
//                      cout<<" case23 ";
                }
                else
                {
                        dis=min_of_4_dis(l1x1, l1y1, l1x2, l1y2, l2x1, l2y1, l2x2, l2y2);
//                      cout<<" case24 ";
                }
        }
  return(dis);
}

void line_formatf03::master_find_line_file_dis(int no_of_line_file)
{
        int i, k, n;
        int line_file_no1, line_file_no2;
        char dummy_char1[5], dummy_char2[5], dummy_char3[5];
  if(access(f_name_line123dis,0)==0)file_delete(f_name_line123dis);
        if(access(f_name_line1dis,0)==0)file_delete(f_name_line1dis);
        if(access(f_name_line2dis,0)==0)file_delete(f_name_line2dis);
        if(access(f_name_line3dis,0)==0)file_delete(f_name_line3dis);
        if(access(f_name_line12dis,0)==0)file_delete(f_name_line12dis);
```

```
        if(access(f_name_line13dis,0)==0)file_delete(f_name_line13dis);
if(access(f_name_line23dis,0)==0)file_delete(f_name_line23dis);

        // find line distance from its own file
// store in file called line123.dis
        strcpy(f_name2, f_name_line);
        strcat(f_name2, "123.dis");
        switch (no_of_line_file)
        {
                case 3:         //file number 3s
                        for (i=31; i<33; ++i)
                        {
                                strcpy(f_name1, build_file_name(f_name_line, i, ".f03"));
                                line_dis_same_file(f_name1, f_name2);
    }
                case 2:         //file number 2s
                        for (i=21; i<23; ++i)
                        {
                                strcpy(f_name1, build_file_name(f_name_line, i, ".f03"));
                                line_dis_same_file(f_name1, f_name2);
                        }
                case 1:         //file number 1s
        for (i=11; i<13; ++i)
                        {
                                strcpy(f_name1, build_file_name(f_name_line, i, ".f03"));
                                line_dis_same_file(f_name1, f_name2);
                        }
        break;
                default:
                        cout<<"ERROR in no of line file!!!\n";
    cout<<"Can't find distance between lines!!!\n";
        }
        // find distance of different line files
        switch (no_of_line_file)
        {
                case 3:         // for files 1s and 3s
                        n=13;
                        strcpy(f_name3, build_file_name(f_name_line, n, ".dis"));
                        for (i=11; i<13; ++i)
                        {
                                strcpy(f_name1, build_file_name(f_name_line, i, ".f03"));
                                for (k=31; k<33; ++k)
                                {
                                        strcpy(f_name2, build_file_name(f_name_line, k, ".f03"));
                                        line_dis_dif_file(f_name1, f_name2, f_name3);
                                }
                        }
                        // for files 2s and 3s
                        n=23;
                        strcpy(f_name3, build_file_name(f_name_line, n, ".dis"));
                        for (i=21; i<23; ++i)
```

```
                {
                        strcpy(f_name1, build_file_name(f_name_line, i, ".f03"));
                        for (k=31; k<33; ++k)
                        {
                                strcpy(f_name2, build_file_name(f_name_line, k, ".f03"));
                                line_dis_dif_file(f_name1, f_name2, f_name3);
                        }
                }
// for line files 3s to find distance
// store in lind distance file called line2.dis
                n=3;
                strcpy(f_name3, build_file_name(f_name_line, n, ".dis"));
i=31;
                strcpy(f_name1, build_file_name(f_name_line, i, ".f03"));
                k=32;
                strcpy(f_name2, build_file_name(f_name_line, k, ".f03"));
                line_dis_dif_file(f_name1, f_name2, f_name3);
        case 2:         // for files 1s and 2s
                // store in line distance file called line12.dis
                n = 12;
                strcpy(f_name3, build_file_name(f_name_line, n, ".dis"));
                for (i=11; i<13; ++i)
                {
                        strcpy(f_name1, build_file_name(f_name_line, i, ".f03"));
                        for (k=21; k<23; ++k)
                        {
                                strcpy(f_name2, build_file_name(f_name_line, k, ".f03"));
                                line_dis_dif_file(f_name1, f_name2, f_name3);
                        }
                }
                // for file 2s to find distance
                // store in lind distance file called line2.dis
                n=2;
                strcpy(f_name3, build_file_name(f_name_line, n, ".dis"));
                i=21;
                strcpy(f_name1, build_file_name(f_name_line, i, ".f03"));
                k=22;
                strcpy(f_name2, build_file_name(f_name_line, k, ".f03"));
                line_dis_dif_file(f_name1, f_name2, f_name3);

        case 1:         // for file 1s to find distance
// store in line distance file called line1.dis
n = 1;
                strcpy(f_name3, build_file_name(f_name_line, n, ".dis"));
                i=11;
                strcpy(f_name1, build_file_name(f_name_line, i, ".f03"));
                k=12;
                strcpy(f_name2, build_file_name(f_name_line, k, ".f03"));
                line_dis_dif_file(f_name1, f_name2, f_name3);
                break;
        default:
```

```
                        cout<<"ERROR in no of line file!!!!!\n";
  }
}

void line_formatf03::line_dis_same_file(char *f_name1, char *f_name2)
{
        int no, i, j;
  float line_width, dis;
        line_formatf03 file1[Max_line_in_line_file];
        // read in line from file
        line_width=line_width_file_name(f_name1);
        // no message output!!!
        fstream1 = file_handling(f_name1, "r+t", fstream1, " line file for input", 0);
  while(fgetc(fstream1)!='\n');
        fscanf(fstream1, "%i", &no);
        for (i=0; i<no; ++i)
                fscanf(fstream1, "%f%f%f%f", &file1[i].x1, &file1[i].y1, &file1[i].x2,
                        &file1[i].y2);
        fclose(fstream1);
        // output line distance to distance file
  // no message output!!!!
        fstream1 = file_handling(f_name2, "a+t", fstream1,
                " line distance for output",0);

        fprintf(fstream1, "%s\n", f_name2);
  fprintf(fstream1, "%s %i\n", f_name1, no);
        for (i=0; i<no; ++i)
        {
                for(j=i+1; j<no; ++j)
                { dis=line_dis(file1[i].x1, file1[i].y1, file1[i].x2, file1[i].y2,
                                file1[j].x1, file1[j].y1, file1[j].x2, file1[j].y2)-line_width;
                        fprintf(fstream1,"%8.3f\n", dis);
                }
        }
        fclose(fstream1);
}

void line_formatf03::line_dis_dif_file(char *f_name1, char *f_name2,
        char *f_name3)
{
        int no1, no2;
        int i, j;
  float dis, line_width1, line_width2;
        line_formatf03 file1[Max_line_in_line_file], file2[Max_line_in_line_file];
        line_width1=line_width_file_name(f_name1);
        line_width2=line_width_file_name(f_name2);
  // read in line from first file
        fstream1 = file_handling(f_name1, "r+t", fstream1, " line file for input");
  while(fgetc(fstream1)!='\n');
        fscanf(fstream1, "%i", &no1);
        for (i=0; i<no1; ++i)
```

```
                fscanf(fstream1, "%f%f%f%f", &file1[i].x1, &file1[i].y1, &file1[i].x2,
                        &file1[i].y2);
        fclose(fstream1);
        // read in line from second file
        fstream1 = file_handling(f_name2, "r+t", fstream1, " line file for input");
  while(fgetc(fstream1)!='\n');
        fscanf(fstream1, "%i", &no2);
        for (i=0; i<no2; ++i)
                fscanf(fstream1, "%f%f%f%f", &file2[i].x1, &file2[i].y1, &file2[i].x2,
                        &file2[i].y2);
        fclose(fstream1);

        // output line distance to distance file
        fstream1 = file_handling(f_name3, "a+t", fstream1,
                " line distance for output");

/*      fprintf(fstream1, "line%i.dis\n", atoi(&f_name3[15]));
        fprintf(fstream1, "line%i.f03 %i line%i.f03 %i\n", atoi(&f_name1[15]), no1,
                atoi(&f_name2[15]), no2);
*/
        fprintf(fstream1, "%s\n", f_name3);
  fprintf(fstream1, "%s %i %s %i\n", f_name1, no1, f_name2, no2);
        for (i=0; i<no1; ++i)
        {
                for (j=0; j<no2; ++j)
                {
                        dis=line_dis(file1[i].x1, file1[i].y1, file1[i].x2, file1[i].y2,
                                file2[j].x1, file2[j].y1, file2[j].x2, file2[j].y2)-
                                ((line_width1+line_width2)/2);
                        fprintf(fstream1,"%8.3f\n", dis);
                }
        }
        fclose(fstream1);
}

float min_of_4_dis(float x11, float y11, float x12, float y12,
        float x21, float y21, float x22, float y22)
{
        float d1,d2;
        d1=sqrt((x11-x21)*(x11-x21)+(y11-y21)*(y11-y21));
        d2=sqrt((x11-x22)*(x11-x22)+(y11-y22)*(y11-y22));
        if(d2<d1)d1=d2;
        d2=sqrt((x12-x21)*(x12-x21)+(y12-y21)*(y12-y21));
        if(d2<d1)d1=d2;
        d2=sqrt((x12-x22)*(x12-x22)+(y12-y22)*(y12-y22));
        if(d2<d1)d1=d2;
        return(d1);
}

float line_width_file_name(char *fname)
{       int i,line_file_no;
```

```
        line_file_no=atoi(&fname[15])/10;
        return(width_of_line[line_file_no-1]);
}
```

```
/*

        (b):\ger\findgap.cpp                    date:31July93     24:00
        program to search pad( ).dis, pad_line().dis and line( ).dis files
        to find the number of different kinds of gaps.

*/

#include <stdio.h>
#include <stdlib.h>                     // for exit function
#include <iostream.h>       // for cout use
#include <math.h>           // for math function use
#include <string.h>                     // for string handling
#include <io.h>                         // for checking file
#include <delfile.h>                    // self developed function to delete file
#include <handfile.h>       // self developed function to handle file

static float gap_range[5]={0.2, 0.4, 0.6, 0.8, 1.0};
static char f_name_pad123dis[]="c:\\feature\\pad123.dis";

static int no_of_pad_file=3;
static int no_of_line_file=3;

class find_gap
{
        public:
        int find_no_of_gap_pad(char *f_name, int no_of_block, int gap_range_no);
        int find_no_of_gap_pdln(char *f_name, int gap_range_no);
        int find_no_of_gap_lnln123(char *f_name, int no_of_block, int gap_range_no);
        int find_no_of_gap_lnlntwo(char *f_name, int no_of_block, int gap_range_no);
        int master_find_gap(int);
};

void main(void)
{
        int gap_range_no = 3, no_of_gap;
        find_gap dum_name;
        no_of_gap = dum_name.master_find_gap(gap_range_no);
        printf("Total no of gap smaller than gap_range[%i](%5.3fmm) = %i",
                gap_range_no, gap_range[gap_range_no], no_of_gap);

}

int find_gap::master_find_gap(int gap_range_no)
{

        int tot_no_of_gap_pad = 0, tot_no_of_gap_pdln = 0;
        int tot_no_of_gap_lnln = 0;
        if(access("c:\\feature\\gap.gap",0)==0)file_delete("c:\\feature\\gap.gap");
        // for all the pad.dis files
        switch (no_of_pad_file)
```

```
        {
                case 3:
                        tot_no_of_gap_pad += find_no_of_gap_pad("c:\\feature\\pad13.dis",
                                1, gap_range_no);
                        tot_no_of_gap_pad += find_no_of_gap_pad("c:\\feature\\pad23.dis",
                                1, gap_range_no);
                case 2:
                        tot_no_of_gap_pad += find_no_of_gap_pad("c:\\feature\\pad12.dis",
                                1, gap_range_no);
                case 1:
                        tot_no_of_gap_pad += find_no_of_gap_pad("c:\\feature\\pad123.dis",
                                no_of_pad_file, gap_range_no);
                        break;
                default:
                        cout<<"ERROR in no of pad file!!!!\n";
                        cout<<"Can't find no of gap from pad.dis file!!!\n";
                        break;
        }

//      cout<<"tot_no_of_gap_pad = "<<tot_no_of_gap_pad<<"\n";

  // for all the padline.dis files
        switch (no_of_line_file)
        {
                case 3:
                        tot_no_of_gap_pdln += find_no_of_gap_pdln("c:\\feature\\padline3.dis",
                                gap_range_no);
                case 2:
                        tot_no_of_gap_pdln += find_no_of_gap_pdln("c:\\feature\\padline2.dis",
                                gap_range_no);
                case 1:
                        tot_no_of_gap_pdln += find_no_of_gap_pdln("c:\\feature\\padline1.dis",
                                gap_range_no);
                        break;
                default:
                        cout<<"ERROR in no of pad file!!!!\n";
                        cout<<"Can't find no of gap from padline.dis file!!!\n";
                        break;
        }

//  cout<<"tot_no_of_gap_pdln = "<<tot_no_of_gap_pdln<<"\n";

        // for all the line.dis files
  // for line123.dis files
        tot_no_of_gap_lnln += find_no_of_gap_lnln123("c:\\feature\\line123.dis",
                no_of_line_file*2, gap_range_no); // assume two type of lines

        // for line(1).dis files
        switch  (no_of_line_file)
        {
                case 3:
```

```
                tot_no_of_gap_lnln += find_no_of_gap_lnlntwo("c:\\feature\\line3.dis",
                        1, gap_range_no);
                case 2:
                tot_no_of_gap_lnln += find_no_of_gap_lnlntwo("c:\\feature\\line2.dis",
                        1, gap_range_no);
                case 1:
                tot_no_of_gap_lnln += find_no_of_gap_lnlntwo("c:\\feature\\line1.dis",
                        1, gap_range_no);
    break;
                default: .
                        cout<<"ERROR in no of line file!!!!\n";
                        cout<<"Can't find no of gap from line.dis file!!!\n";
                        break;
        }
        // for line(12).dis files
        switch (no_of_line_file)
        {
                case 3:
                tot_no_of_gap_lnln += find_no_of_gap_lnlntwo("c:\\feature\\line23.dis",
                        4, gap_range_no);
                tot_no_of_gap_lnln += find_no_of_gap_lnlntwo("c:\\feature\\line13.dis",
                        4, gap_range_no);
                case 2:
                tot_no_of_gap_lnln += find_no_of_gap_lnlntwo("c:\\feature\\line12.dis",
                        4, gap_range_no);
                        break;
                default:
                        cout<<"ERROR in no of line file!!!\n";
                        cout<<"Can'f find no of gap from line(12).dis file!!!\n";
                        break;
        }
        return(tot_no_of_gap_pad+tot_no_of_gap_pdln+tot_no_of_gap_lnln);
}

int find_gap::find_no_of_gap_pad(char *f_name,int no_of_block,int gap_range_no)
{
        int tot_no_of_gap=0, no_of_gap=0, file_no, no_of_line1;
        int no_of_line2, no_of_dis, i, j, k;
  float dis;
        char dum_c;
        fstream2=file_handling("c:\\feature\\gap.gap", "a+t", fstream2,
                " to store gap position.");
        fstream1=file_handling(f_name, "r+t", fstream1, " to find no of gap.");
        int control_block=0;
        for (j=0; j<no_of_block; ++j)
        {
                no_of_gap=0;
                if(control_block>0)
                        while(fgetc(fstream1)!='\n');          //scan out the return char after 1 round
    control_block=1;
    while(fgetc(fstream1)!='\n');
```

```
                while(fgetc(fstream1) != 'd');          // scan while dum_c not equal to d

                fscanf(fstream1, "%i", &file_no);
//    cout<<"file_no"<<file_no<<"\n";
   if(file_no>10)fscanf(fstream1, "%i", &no_of_line2);
                fscanf(fstream1, "%i%i", &no_of_line1, &no_of_dis);
                        fprintf(fstream2, "c:\\feature\\gap.gap\n");

                if(file_no<10)     fprintf(fstream2, "pad%i.f02\n", file_no);
                else if(file_no>10)
                {
                        k=file_no/10;
                        k=k*10;
     k=file_no-k;
                        fprintf(fstream2, "pad%i.f02 pad%i.f02\n", file_no/10,k);
                }
        for (i=0; i<no_of_dis; ++i)
                {
                        fscanf(fstream1, "%f", &dis);
//                      cout<<"dis "<<dis;
                        if((dis>=0)&&(dis<=gap_range[gap_range_no]))
            {
                                no_of_gap+=1;
//          cout<<"dis="<<dis;
            fprintf(fstream2,"%i ", i);
                        }
        }
                if(no_of_gap==0)fprintf(fstream2, "-1");     // no gap
                fprintf(fstream2, "\n");
                tot_no_of_gap+=no_of_gap;
        }
        fclose(fstream1);
  fclose(fstream2);
        return(tot_no_of_gap);
}

int find_gap::find_no_of_gap_pdln(char *f_name, int gap_range_no)
{
        int tot_no_of_gap=0, no_of_gap;
        int pad_file_no, no_of_line1, line_file_no, no_of_line2,
                no_of_dis, i, j;
  float dis;

        fstream1=file_handling(f_name, "r+t", fstream1, " to find no of gap.");
        fstream2=file_handling("c:\\feature\\gap.gap", "a+t", fstream2,
                " to store gap position.");
        for (i=0; i<(no_of_pad_file*2); ++i)
  {
                no_of_gap=0;
                while(fgetc(fstream1) != '\n');
    while(fgetc(fstream1) != 'd');    // not equal to char d
```

```
                fscanf(fstream1, "%i", &pad_file_no);
                while(fgetc(fstream1) != ' ');                  // not equal to space
                fscanf(fstream1, "%i", &no_of_line1);
                while(fgetc(fstream1) != 'n');          // not equal to char n
                fgetc(fstream1);                // get char e
                fscanf(fstream1, "%i", &line_file_no);
                while(fgetc(fstream1) != ' ');          // not equal to space
                fscanf(fstream1, "%i", &no_of_line2);
                fprintf(fstream2, "c:\\feature\\gap.gap\n");
    fprintf(fstream2, "pad%i.f02 line%i.f03\n", pad_file_no, line_file_no);
    for (j=0; j<(no_of_line1*no_of_line2); ++j)
                {
                        fscanf(fstream1, "%f", &dis);
                        if((dis>=0)&&(dis<=gap_range[gap_range_no]))
                  {
                                no_of_gap+=1;
                                fprintf(fstream2,"%i ", j);
//                              cout<<dis<<" ";
                        }
                }
                if(no_of_gap == 0)fprintf(fstream2, "-1");
    fprintf(fstream2, "\n");
//          cout <<" no_of_gap = "<<no_of_gap;
                tot_no_of_gap += no_of_gap;
                while(fgetc(fstream1)!= '\n');
          }

          fclose(fstream1), fclose(fstream2);
  return(tot_no_of_gap);
}

int find_gap::find_no_of_gap_lnln123(char *f_name, int no_of_block, int gap_range_no)
{
          int tot_no_of_gap=0, no_of_gap;
          int no_of_line1, line_file_no, no_of_dis, i, j, k;
  float dis;

          fstream1=file_handling(f_name, "r+t", fstream1, " to find no of gap.");
          fstream2=file_handling("c:\\feature\\gap.gap", "a+t", fstream2,
                  " to store gap position.");
//        int control_no=0;
          for (k=0; k<no_of_block; ++k)
  {
                no_of_gap = 0;
//              if(control_no==1)fgetc(fstream1);  //remove the carrage return character
                        // in the second round
//    control_no = 1;
                while(fgetc(fstream1) != '\n');
                while(fgetc(fstream1) != 'n');          // not equal to char n in line
                while(fgetc(fstream1) != 'e');          // not equal to char e in line
                fscanf(fstream1, "%i", &line_file_no);
```

```
//      cout<<"line_file_no="<<line_file_no;
                while(fgetc(fstream1) != ' ');                  // not equal to space
                fscanf(fstream1, "%i", &no_of_line1);
//      cout<<"no_of_line1="<<no_of_line1;
            no_of_dis = 0;
                for (i=0; i<no_of_line1; ++i)
                        no_of_dis +=i;                          // generate no of line
//              cout<<"line_file_no = "<<line_file_no<<"no_of_line1 = "<<no_of_line1<<
//              " no_of_dis = "<<no_of_dis<<"\n";
                fprintf(fstream2, "c:\\feature\\gap.gap\n");
                fprintf(fstream2, "line%i.f03 line%i.f03\n", line_file_no, line_file_no);
                for (j=0; j<no_of_dis; ++j)
                {
                        fscanf(fstream1, "%f", &dis);
//        cout<<"dis="<<dis;
                        if((dis>=0)&&(dis<=gap_range[gap_range_no]))
                        {
                                no_of_gap+=1;
                                fprintf(fstream2,"%i ", j);
                        }
                }


                fgetc(fstream1);
                if(no_of_gap == 0)fprintf(fstream2, "-1");
                fprintf(fstream2, "\n");
                tot_no_of_gap +=no_of_gap;
//              cout<<"no_of_gap="<<no_of_gap<<"\n";
        }
  fclose(fstream1), fclose(fstream2);
        return(tot_no_of_gap);
}

int find_gap::find_no_of_gap_lnlntwo(char *f_name, int no_of_block, int gap_range_no)
{
        int tot_no_of_gap=0, no_of_gap;
        int line_file_no1, no_of_line1, line_file_no2, no_of_line2,
                no_of_dis, i, j, k;
  float dis;

        fstream1=file_handling(f_name, "r+t", fstream1, " to find no of gap.");
        fstream2=file_handling("c:\\feature\\gap.gap", "a+t", fstream2,
                " to store gap position.");
        for (k=0; k<no_of_block; ++k)
  {
        no_of_gap = 0;
                while(fgetc(fstream1) != '\n');
                while(fgetc(fstream1) != 'n');
                while(fgetc(fstream1) != 'e');          // not equal to char e
                fscanf(fstream1, "%i", &line_file_no1);
                while(fgetc(fstream1) != ' ');                  // not equal to space
```

```
                fscanf(fstream1, "%i", &no_of_line1);
                while(fgetc(fstream1) != 'n');
                while(fgetc(fstream1) != 'e');          // not equal to char e
                fscanf(fstream1, "%i", &line_file_no2);
                while(fgetc(fstream1) != ' ');                  // not equal to space
                fscanf(fstream1, "%i", &no_of_line2);
                no_of_dis = no_of_line1*no_of_line2;

                fprintf(fstream2, "c:\\feature\\gap.gap\n");
                fprintf(fstream2, "line%i.f03 line%i.f03\n", line_file_no1, line_file_no2);

                for (i=0; i<no_of_dis; ++i)
                {
                        fscanf(fstream1, "%f", &dis);
                        if((dis>=0)&&(dis<=gap_range[gap_range_no]))
                        {
                                no_of_gap+=1;
                                fprintf(fstream2,"%i ", i);
//                              cout<<dis<<" ";
                        }
                }
                if(no_of_gap == 0)fprintf(fstream2, "-1");
                fprintf(fstream2, "\n");
                tot_no_of_gap +=no_of_gap;
    fgetc(fstream1);        // remove the carrage return character
  }
        fclose(fstream1), fclose(fstream2);
//      cout<<tot_no_of_gap;
        return(tot_no_of_gap);
}
```

```
/*

        program sur_area.cpp                date: 30July93 17:20

        program to find surface area from pad file and line file
        read in pad file and then line file
                pad file read in: c:\\feature\\pad1.f02
                line file read in: c:\\feature\\line1.f01

*/

#include <stdio.h>
#include <stdlib.h>  // for exit function
#include <iostream.h>// for cout use
#include <math.h>    // for math function use
#include <finddis.h>        // self develop function to find distance between two pts
#include <handfile.h>// self develop function to handle file open
#define Pie 3.1415927
#define Dec_pt 1000

static float width_of_line[3]={0.1, 0.2, 0.3};
static float dia_of_pad[3]={0.8, 1.0, 1.2};
static int no_of_pad_file=3;
static int no_of_line_file=3;          // no of pad file and line file

static char f_name_pad1f02[] = "c:\\feature\\pad1.f02";
static char f_name_pad2f02[] = "c:\\feature\\pad2.f02";
static char f_name_pad3f02[] = "c:\\feature\\pad3.f02";

static char f_name_line1f01[] = "c:\\feature\\line1.f01";
static char f_name_line2f01[] = "c:\\feature\\line2.f01";
static char f_name_line3f01[] = "c:\\feature\\line3.f01";

float total_surface_area(int, int);

class pad_area
{
        int no_of_pad;
        public:
        int find_no_pad(char *char_pt);
        float total_pad_area(int no_of_pad_file);
} pad_file_no[3];

class line_area
{
        float ind_line_area;
        public:
        float line_file_unit_length(char *f_name);
        float total_line_area(int no_of_line_file);
} line_file_no[3];
```

```
void main(void)
{
        float total_area=0.0;
  total_area=total_surface_area(no_of_line_file, no_of_pad_file);
        cout<<"Total surface area of the board = "<<total_area<<"\n";
        fstream1=file_handling("c:\\feature\\area.001","w+t",
                fstream1," to store board area.");
        fprintf(fstream1, "c:\\feature\\area.001\n%9.3f\n", total_area);
        fclose(fstream1);
}

float total_surface_area (int no_of_line_file, int no_of_pad_file)
{
        line_area dummy_name1;
        float tot_line_area, tot_pad_area;
        tot_line_area=dummy_name1.total_line_area(no_of_line_file);
        cout<<"Total line area = "<<tot_line_area<<"\n";
        pad_area dummy_name2;
        tot_pad_area=dummy_name2.total_pad_area(no_of_pad_file);
        cout<<"Total pad area = "<<tot_pad_area<<"\n";
        return(tot_line_area+tot_pad_area);
}

float pad_area::total_pad_area(int no_of_pad_file)
{
        float total_pad_area=0;
        switch (no_of_pad_file)
        {
    case 3:
                        total_pad_area=Pie/4*dia_of_pad[3-1]*dia_of_pad[3-1]*
                                find_no_pad(f_name_pad3f02);
                case 2:
                        total_pad_area+=Pie/4*dia_of_pad[2-1]*dia_of_pad[2-1]*
                                find_no_pad(f_name_pad2f02);
                case 1:
                        total_pad_area+=Pie/4*dia_of_pad[1-1]*dia_of_pad[1-1]*
                                find_no_pad(f_name_pad1f02);
      break;
                default:
                        cout<<"Error Number of pad file is not correct!!";
                        break;
        }
  return total_pad_area;
}

int pad_area::find_no_pad(char *f_name)
{
        int i = 0;
  fstream1=file_handling(f_name,"r+t",fstream1," for counting no of pad.");
  while(fgetc(fstream1)!='\n');
        fscanf(fstream1, "%i", &i);
```

```
  fclose(fstream1);
  return(i);
}

float line_area::total_line_area(int i)
{
        float total_line_area=0;
  int line_file_no;
        switch(i)
  {
                case 3:
        line_file_no=3;
                        total_line_area+=line_file_unit_length(f_name_line3f01)*
                                width_of_line[line_file_no-1];
                        cout<<" total line area = "<<total_line_area<<"\n";
                case 2:
        line_file_no=2;
                        total_line_area+=line_file_unit_length(f_name_line2f01)*
                                width_of_line[line_file_no-1];
                        cout<<" total line area = "<<total_line_area<<"\n";
                case 1:
        line_file_no=1;
                        total_line_area+=line_file_unit_length(f_name_line1f01)*
                                width_of_line[line_file_no-1];
                        cout<<" total line area = "<<total_line_area<<"\n";
                break;
                default:
                cout<<"Error in number of line file.!!!!\n";
                break;
        }
        return(total_line_area);
}

float line_area::line_file_unit_length(char *f_name)
// function to find total unit length from line file
{
        int i = 0, no_of_line;                  // number of line in line file
  float x1, x2, y1, y2;
  float length=0.0;
        char char1;
        fstream1=file_handling(f_name,"r+t", fstream1," to find length of line.");
        while(fgetc(fstream1)!='\n');
        fscanf(fstream1, "%i", &no_of_line);
//      cout<<"no of line = "<<no_of_line;
        for (i=0; i<no_of_line; ++i)
  {
                fscanf(fstream1, "%f %f %f %f ", &x1, &y1, &x2, &y2);
//              printf("x1=%8.3f y1=%8.3f x2=%8.3f y2=%8.3f ", x1, y1, x2, y2);
                x1/=Dec_pt;
                y1/=Dec_pt;
                x2/=Dec_pt;
```

```
    y2/=Dec_pt;
//                  printf("x1=%8.3f y1=%8.3f x2=%8.3f y2=%8.3f ", x1, y1, x2, y2);
                    length += find_dis(x1, y1, x2, y2);
//                  printf(" length = %8.3f\n", length);
          }
  fclose (fstream1);
          return (length);
}
```

Appendix 4.2 Listing of the Solder Masking Feature Extraction Program

```
/*

        (a):\ger\soldfile.cpp                    16Aug93 16:00

        This program will read in Gerber format SOLDER PAD file and transform the
        file into different files according to the amperature of the pad. The
        format of the will be comparatable for processing during solder pad
        clearance analysis program.

        This program assume a MAX of THREE different pads.

*/

#include <stdio.h>
#include <stdlib.h>  // for exit function
#include <iostream.h>// for cout use
#include <string.h>
#include <math.h>   // for math function use
#include <delfile.h>        // self developed function to delete file
#include <handfile.h>       // self developed function to handle file
#include <bld_f_n.h>                // self developed function to build up file name

#define Max_line_in_ger 450  // max number of line in gerber file
#define Dec_pt 1000                                  // dec pt

static char solder_file_name[]="a:\\solder.gbr";
                                                                    // drive must be change for
different computer

static int amp_look_up_table[]={122, 133, 167};
int no_of_solder_pad_file=3;

class ger_f_for
{
        char solder_file_name[30];

        public:
        void ger_to_f01(char *solder_file_name);
        void sort_solder_file(void);
}       solder_film[3], solder_filmf02[3];

void main(void)
{
        ger_f_for read_ger_file;
        read_ger_file.ger_to_f01(solder_file_name);
        read_ger_file.sort_solder_file();
}
```

```
void ger_f_for::sort_solder_file(void)
{
        int i, j, k, no_line, pt_posit[Max_line_in_ger-200];
        float x[Max_line_in_ger-200], y[Max_line_in_ger-200];
        float dum_x, dum_y;
  char dum_c=' ', dum_str[30];

        for (k=0; k<no_of_solder_pad_file; ++k)
        {
                strcpy(solder_filmf02[k].solder_file_name, "c:\\feature\\solder");
                strcat(solder_filmf02[k].solder_file_name, itoa(k+1,dum_str,10));
                strcat(solder_filmf02[k].solder_file_name, ".f02");
        }

        for (k=0; k<no_of_solder_pad_file; ++k)
  {
                no_line=0, strcpy(&dum_c, " ");
                fstream1=file_handling(solder_film[k].solder_file_name, "r+t", fstream1,
                        " for reading solder pad location.");
                while(fgetc(fstream1)!='\n');
                while(dum_c!='*')
                {
                        dum_c=fgetc(fstream1);
                        if(dum_c=='*')break;
                        else
                        {
                                while(fgetc(fstream1)!='\n');
            ++no_line;
                        }
                }
    cout<<"no_line"<<no_line;
                rewind(fstream1);
    while(fgetc(fstream1)!='\n');
    for (i=0; i<no_line; ++i)
                fscanf(fstream1, "%f%f", &x[i], &y[i]);
                for (i=0; i<no_line; ++i)
                {
                        for (j=i+1; j<no_line; ++j)
                        {
                                if (x[i]>x[j])
                                {
                                        dum_x=x[i];
                                        dum_y=y[i];
                                        x[i]=x[j];
                                        y[i]=y[j];
                                        x[j]=dum_x;
                                        y[j]=dum_y;
                                }
                        }
                }
                fclose(fstream1);
```

```
                // open file for output sorted for smallest x pads
                fstream1=file_handling(solder_filmf02[k].solder_file_name,
                        "w+t", fstream1, " to store type .f02 solder pad location.");
                fprintf(fstream1, "%s\n%i\n", solder_filmf02[k].solder_file_name,
                        no_line);
                for (i=0; i<no_line; ++i)
                        fprintf(fstream1, "%8.3f %8.3f\n", x[i]/Dec_pt, y[i]/Dec_pt);

                // sort solder pad based on smallest y
                for (i=0; i<no_line; ++i)   // store up the position
                pt_posit[i]=i;
                for (i=0; i<no_line; ++i)
                {
                        for (j=i+1; j<no_line; ++j)
                        {
                                if (y[i]>y[j])
                                {
                                        dum_x = pt_posit[i];
                                        dum_y = y[i];
                                        pt_posit[i] = pt_posit[j];
                                        y[i] = y[j];
                                        pt_posit[j] = dum_x;
                                        y[j] = dum_y;
                                }
                        }
            }
                for (i=0; i<no_line; ++i)
                        fprintf(fstream1,"%i ", pt_posit[i]);
                fclose(fstream1);
        }
}

void ger_f_for::ger_to_f01(char *solder_file_name)
{
        int i=0, j, amp_code, d_code;
        int x, y, pre_x, pre_y;
        char dum_c, str1[50], pre_str[50], dum_str[20];
  char *str_y, *str_d;
        fstream1=file_handling(solder_file_name, "r+t", fstream1,
                " for reading solder file.");
        fstream2=file_handling("c:\\feature\\temp.g01", "w+t", fstream2,
        " for storying temp. gerber data.");
        while (dum_c != 'M')
        {
                fscanf(fstream1, "%s", str1);
                strncpy(&dum_c, str1, 1);
                if (dum_c=='M')break;
                if (str1[0]=='D')
    {
                        i=atoi(&str1[1]);
```

```
                    if(i>100) fprintf(fstream2, "%s\n", str1);
                    else if(i==3)
                    {
    j=0;
                        do{
                        fputc(pre_str[j], fstream2);
                        ++j;
                        }while(pre_str[j]!='D');
                        fprintf(fstream2, "D03*\n");
                    }
            }
            else
            {
                    fprintf(fstream2, "%s\n", str1);
}
strcpy(pre_str, str1);
    }
    fprintf(fstream2, "M02*\n");

    fclose(fstream1);
    rewind(fstream2);
    fstream3=file_handling("c:\\feature\\tempsold.g01", "w+t", fstream3,
    " for storying solder pad data.");
    fprintf(fstream3, "c:\\feature\\tempsold.g01\n");
    do{
            fscanf(fstream2, "%s", str1);
            strncpy(&dum_c, str1, 1);
            if (dum_c=='M')break;
            if ((dum_c=='X')||(dum_c=='Y'))
            {
                    if (dum_c=='X')
                    {
                        x=atoi(&str1[1]);
                        if((str_y=strchr(str1,'Y'))==NULL)
                        {
                            y=pre_y;
        }
    else y=atoi(&str_y[1]);
                    }
                    else if (dum_c=='Y')
                    {
                        x=pre_x;
                        y=atoi(&str1[1]);
                    }
                    str_d=strchr(str1,'D');
                    d_code=atoi(&str_d[1]);
                    if(d_code==1)
                    {
                        cout<<"ERROR: Line code in solder pad file. Data ignored!!!";
                    }
                    if(d_code==2);
```

```
                        if(d_code==3)
                        {
                                fprintf(fstream3, "%i %i\n", x, y);
    }
                        pre_x=x;
    pre_y=y;
  }

                if (dum_c=='D')
                {
                        i=atoi(&str1[1]);
                        if(i>100)
                        {
                                amp_code=i;
                                fprintf(fstream3, "D%i\n", i);
                        }
                }
        } while(dum_c!='M');
        fprintf(fstream3, "M02");
        fclose(fstream2), fclose(fstream3);

        for (j=0; j<no_of_solder_pad_file; ++j)        // a max of THREE pad files
        {
                strcpy(solder_film[j].solder_file_name, "c:\\feature\\solder");
                strcat(solder_film[j].solder_file_name, itoa(j+1,dum_str,10));
                strcat(solder_film[j].solder_file_name, ".f01");
        }

        fstream1=file_handling(solder_film[0].solder_file_name, "w+t", fstream1,
                " for type 1 solder pad location.");
        fprintf(fstream1, "%s\n", solder_film[0].solder_file_name);
fstream2=file_handling(solder_film[1].solder_file_name, "w+t", fstream2,
                " for type 2 solder pad location.");
        fprintf(fstream2, "%s\n", solder_film[1].solder_file_name);
fstream3=file_handling(solder_film[2].solder_file_name, "w+t", fstream3,
                " for type 3 solder pad location.");
        fprintf(fstream3, "%s\n", solder_film[2].solder_file_name);

        fstream4=file_handling("c:\\feature\\tempsold.g01", "r+t", fstream4,
                " for reading pad data.");

        while(fgetc(fstream4)!='\n');
dum_c=fgetc(fstream4);
        fscanf(fstream4, "%i", &amp_code);
        fscanf(fstream4, "%i%i", &x, &y);
while(fgetc(fstream4)!='\n');
        while(dum_c!='M')
        {
                if (amp_code == amp_look_up_table[0])
    fprintf(fstream1, "%i %i\n", x, y);
                else if (amp_code == amp_look_up_table[1])
```

```
                fprintf(fstream2, "%i %i\n", x, y);
            else if (amp_code == amp_look_up_table[2])
                fprintf(fstream3, "%i %i\n", x, y);
            else
                cout<<"ERROR in amp_code. Solder pad location ignored!!!";

            fscanf(fstream4, "%s", str1);
            strncpy(&dum_c, str1, 1);
            if(dum_c=='D')
            {
                amp_code=atoi(&str1[1]);
                fscanf(fstream4, "%i%i", &x, &y);
    }
            else if(dum_c=='M')break;
            else
            {
        x=atoi(&str1[0]);
                fscanf(fstream4, "%i", &y);
     fgetc(fstream4);
    }
        }
        fprintf(fstream1, "*"),     fprintf(fstream2, "*"), fprintf(fstream3, "*");
        fclose(fstream1), fclose(fstream2), fclose(fstream3);
}
```

```
/*

        (a):\ger\soldcler.cpp        4Aug93 16:00

  program to find the clearance of solder mask pad
        This program will compare all the solder mask pad files with the pad files.
        The clearance will be output back to solder file with extension .f12. Then
        all the .f12 solder files will be searched. The number of solder mask on pad,
        the number of place with not enough clearance and the number of solder masking
        opening without pads will be found. All the outstanding solder pad
        information will be output and storied in a file called soldcler.out.

        This program can automatically generated all the required file names.

        This program assume a max of THREE solder pad files and a max of THREE pad
        files only. You can adjust the actual number of solder file and pad file in
        in the global valiables. The allowable opening for the solder mask should be
        input as well.

*/

#include <stdio.h>
#include <stdlib.h>                 // for exit function
#include <iostream.h>     // for cout use
#include <math.h>         // for math function use
#include <string.h>                 // for string handling
#include <io.h>                     // for checking file
#include <bld_f_n.h>                // self developed function to build up file name
#include <delfile.h>                // self developed function to delete file
#include <handfile.h>     // self developed function to handle file
#include <minmax.h>                 // self developed function to find min and max
#include <finddis.h>                // self developed function to find distance between pts

#define Max_line_in_pad_file 150          // max no of line in pad file
#define Max_line_in_sold_file 150         // max no of line in sold file

static char fname_cler_out[] = "c:\\feature\\soldcler.out";

int no_of_pad_file = 3;             //THIS IS THE NO OF PAD FILE
int no_of_solder_file = 3; //THIS IS THE NO OF SOLDER MASKER OPENING FILE

static float dia_of_pad[3] = {0.8, 1.0, 1.2};
static float dia_of_sold_pad[3] = {0.9, 1.2, 1.4};

class solder_mask_cler
{
        public:
        long int master_program(int no_of_solder_file, int no_of_pad_file,
                float cler_limit);
        void file_to_search(char *sold_file_name, char *sold_cler_file_name,
                char *pad_file_name,      int sold_dia_no, int pad_dia_no);
```

```
        float clearance(float sold_x, float sold_y, int sold_dia_no,
                float pad_x, float pad_y, int pad_dia_no);
        long int outstanding_cler(char *sold_fname, char *sold_cler_fname,
                float cler_limit);
};

void main(void)
{
        float cler_limit = 0.125;                       // set the solder mask clearance limit here
        solder_mask_cler solder_mask;
        cout << solder_mask.master_program(no_of_solder_file, no_of_pad_file,
                cler_limit);
}

long int solder_mask_cler::master_program(int no_of_solder_file,
        int no_of_pad_file, float cler_limit)
{
        char dum_str[30];
        char sold_file_name[3][30], sold_cler_file_name[3][30],
                pad_file_name[3][30];
        int i, j, k, no_of_solder_line;
        long int outstd_no=0;

        if(access(fname_cler_out,0)==0) file_delete(fname_cler_out);

        //build file name and delete sold clearance file if existance
  for (i=0; i<3; ++i)                   // assume a max of THREE solder mask file
        {
        strcpy(sold_file_name[i], "c:\\feature\\sold");
                strcat(sold_file_name[i], itoa(i+1,dum_str,10));
                strcat(sold_file_name[i], ".f02");

                strcpy(sold_cler_file_name[i], "c:\\feature\\sold");
                strcat(sold_cler_file_name[i], itoa(i+1,dum_str,10));
                strcat(sold_cler_file_name[i], ".f12");

                strcpy(pad_file_name[i], "c:\\feature\\pad");
                strcat(pad_file_name[i], itoa(i+1,dum_str,10));
                strcat(pad_file_name[i], ".f02");

                if(access(sold_cler_file_name[i],0)==0)
                        file_delete(sold_cler_file_name[i]);
        }

        for (i=0; i<no_of_solder_file; ++i)
        {
                fstream1=file_handling(sold_file_name[i], "r+t", fstream1,
                        "for reading solder pad location.");
    while(fgetc(fstream1)!='\n');
    fscanf(fstream1, "%i", &no_of_solder_line);
                fclose(fstream1);
```

```
                fstream1=file_handling(sold_cler_file_name[i], "w+t", fstream1,
                        "for writing solder clearance data.");
                fprintf(fstream1, "%s\n", sold_cler_file_name[i]);
    fprintf(fstream1, "%i\n", no_of_solder_line);
                for (j=0; j<no_of_solder_line; ++j)
                        fprintf(fstream1, "000000 999.999\n");
                fclose(fstream1);
        }

        for (i=0; i<no_of_solder_file; ++i)
        {
                for (j=0; j<no_of_pad_file; ++j)
                {
                        file_to_search(sold_file_name[i], sold_cler_file_name[i],
                                pad_file_name[j], i, j);
                }
        }
  // output the clearance limit to the clearance file first
        fstream1=file_handling(fname_cler_out, "a+t", fstream1,
                " for recording  the clearance limit first.");
        fprintf(fstream1, "%s\n", fname_cler_out);
  fprintf(fstream1, "clearance limit = %6.3f\n", cler_limit);
  fclose(fstream1);
        for (i= 0; i<no_of_solder_file; ++i)
                outstd_no+=outstanding_cler(sold_file_name[i], sold_cler_file_name[i],
                        cler_limit);

  return(outstd_no);

}

void solder_mask_cler::file_to_search(char *sold_name,
        char *cler_name, char *pad_name, int sold_dia_no, int pad_dia_no)
{
        int i, j, k, no_of_sold, no_of_pad, position[Max_line_in_sold_file];
  int pad_file_no;
        float sold_x[Max_line_in_sold_file], sold_y[Max_line_in_sold_file];
        float gap[Max_line_in_sold_file];
        float pad_x[Max_line_in_pad_file], pad_y[Max_line_in_pad_file];
        float rad_sold = (dia_of_sold_pad[sold_dia_no]/2);
//      float rad_pad = (dia_of_pad[pad_dia_no]/2);

        pad_file_no=atoi(&pad_name[14]);
        fstream1=file_handling(sold_name, "r+t", fstream1,
                " for reading solder pad location.");

  while(fgetc(fstream1)!='\n');
        fscanf(fstream1, "%i", &no_of_sold);
        for (i=0; i<no_of_sold; ++i)
                fscanf(fstream1,"%f %f", &sold_x[i], &sold_y[i]);
```

```
        fclose(fstream1);

        fstream1=file_handling(cler_name, "r+t", fstream1,
                " for reading clearance data.");
        while (fgetc(fstream1)!='\n');
  fscanf(fstream1, "%i", &no_of_sold);
  for (i=0; i<no_of_sold; ++i)
                fscanf(fstream1,"%i %f", &position[i], &gap[i]);
        fclose(fstream1);

        fstream1=file_handling(pad_name, "r+t", fstream1,
                " for reading pad location.");
  while(fgetc(fstream1)!='\n');
        fscanf(fstream1, "%i", &no_of_pad);
        for (i=0; i<no_of_pad; ++i)
                fscanf(fstream1, "%f %f", &pad_x[i], &pad_y[i]);
        fclose(fstream1);

        // start to do the checking
  for (i=0; i<no_of_sold; ++i)
        {
//              if (position[i]<1000)       // do the checking
//    {
                        for (j=0; j<no_of_pad; ++j)
                        {
        if(((sold_x[i]-rad_sold)<pad_x[j])&&((sold_x[i]+rad_sold)>pad_x[j])
                                        &&((sold_y[i]-
rad_sold)<pad_y[j])&&((sold_y[i]+rad_sold)>pad_y[j]))
                                {
                                        gap[i]=clearance(sold_x[i],sold_y[i],sold_dia_no, pad_x[j],
                                                pad_y[j], pad_dia_no);
                                        position[i] = pad_file_no*1000+j;
                                }
       // else default 000000 999.999
                        }
//    }
        }

        fstream1=file_handling(cler_name, "w+t", fstream1,
                " for writing clearance data.");
        fprintf(fstream1, "%s\n", cler_name);
        fprintf(fstream1, "%i\n", no_of_sold);
        for (i=0; i<no_of_sold; ++i)
                fprintf(fstream1,"%6i %6.3f\n", position[i], gap[i]);
        fclose(fstream1);

}

long int solder_mask_cler::outstanding_cler(char *sold_fname,
        char *sold_cler_fname, float cler_limit)
{
```

```
        float x, y, gap;
        int sold_line_no, sold_file_no, position, i, j, k;
        long int outstd_no=0;
        int no_cler_no=0, not_enough_all_no=0, sold_no_pad_no = 0;

        sold_file_no=atoi(&sold_fname[15]);

        fstream1=file_handling(sold_fname, "r+t", fstream1,
                " for reading solder loaction data.");
        while(fgetc(fstream1)!='\n');
        fscanf(fstream1, "%i", &sold_line_no);

        fstream2=file_handling(sold_cler_fname, "r+t", fstream2,
                " for reading clearance data.");
        while(fgetc(fstream2)!='\n');
        fscanf(fstream2, "%i", &sold_line_no);

        fstream3=file_handling(fname_cler_out, "a+t", fstream3,
                " for recording outstanding solder mask pad.");
 fprintf(fstream3, "%s\n", fname_cler_out);

        for (i=0; i<sold_line_no; ++i)
        {
                fscanf(fstream1, "%f %f", &x, &y);
                fscanf(fstream2, "%i %f", &position, &gap);

                if (gap<0)
        // solder on pad already
                {
                        no_cler_no += 1;
                        fprintf(fstream3, "%i %i %6i %8.3f %8.3f %6.3f\n",
                                sold_file_no, i, position, x, y, gap);
                }
                if((gap>=cler_limit)&&(gap<999))          // not enough gap
                {
                        not_enough_all_no += 1;
                        fprintf(fstream3, "%i %i %6i %8.3f %8.3f %6.3f\n",
                                sold_file_no, i, position, x, y, gap);
                }
                if(gap>999)
        // solder opening without pad
                {
                        sold_no_pad_no += 1;
                        fprintf(fstream3, "%i %i %6i %8.3f %8.3f %6.3f\n",
                                sold_file_no, i, position, x, y, gap);
    }
        }
 fclose(fstream1), fclose(fstream2), fclose(fstream3);
        outstd_no=no_cler_no*1000000+not_enough_all_no*1000+sold_no_pad_no;
 return(outstd_no);
}
```

```
float solder_mask_cler::clearance(float sold_x, float sold_y, int sold_dia_no,
        float pad_x, float pad_y, int pad_dia_no)
{
        float clearance;
        clearance = dia_of_sold_pad[sold_dia_no]/2 - dia_of_pad[pad_dia_no]/2 -
                find_dis(sold_x, sold_y, pad_x, pad_y);
        return(clearance);
}
```

Appendix 5.1 CircuitA Circuit Image Data File

```
D162*
X0Y0D02*
Y5000D01*
X6000D01*
Y0D01*
X0D01*
X300Y4700D02*
D03*
X5700D02*
D03*
Y300D02*
D03*
D111*
X800Y2000D02*
Y400D01*
X2600D01*
Y2200D01*
X2800Y600D02*
Y2000D01*
X4600D01*
Y1200D01*
X3200Y1400D02*
X4400D01*
Y1000D01*
X5400Y1800D02*
Y3000D01*
X3800D01*
X3200Y4600D02*
Y4000D01*
X1800D01*
Y3400D01*
D135*
X2600Y2600D02*
X1200D01*
Y3600D01*
X1600D01*
X1000Y3000D02*
Y2100D01*
X2000D01*
Y600D01*
X3000Y2200D02*
X5200D01*
Y600D01*
X4200D01*
X5200Y3200D02*
Y3600D01*
X3600D01*
Y2600D01*
```

```
X3000D01*
Y3000D02*
Y3800D01*
X4800D01*
D150*
X2000Y600D02*
D03*
X1000Y3000D02*
D03*
X2600D02*
D03*
Y2600D02*
D03*
Y2200D02*
D03*
X3000D02*
D03*
Y2600D02*
D03*
Y3000D02*
D03*
X3800D02*
D03*
X4400Y1000D02*
D03*
X5200Y3200D02*
D03*
X4600Y1200D02*
D03*
X1800Y3400D02*
D03*
D159*
X800Y2000D02*
D03*
X1600Y3600D02*
D03*
X2800Y600D02*
D03*
X3200Y1400D02*
D03*
X4200Y600D02*
D03*
X5400Y1800D02*
D03*
Y3800D02*
D03*
X4800D02*
D03*
X3200Y4600D02*
D03*
M02*
```

Appendix 5.2 CircuitA Solder Masking Image Data File

```
D162*
X0Y0D02*
Y5000D01*
X6000D01*
Y0D01*
X0D01*
X300Y4700D02*
D03*
X5700D02*
D03*
Y300D02*
D03*
D162*
X2000Y600D02*
D03*
X1000Y3000D02*
D03*
X2600D02*
D03*
Y2600D02*
D03*
Y2200D02*
D03*
X3000D02*
D03*
Y2600D02*
D03*
Y3000D02*
D03*
X3800D02*
D03*
X4400Y1000D02*
D03*
X5200Y3200D02*
D03*
X4600Y1200D02*
D03*
X1800Y3400D02*
D03*
D166*
X800Y2000D02*
D03*
X1600Y3600D02*
D03*
X2800Y600D02*
D03*
```

```
X3200Y1400D02*
D03*
X4200Y600D02*
D03*
X5400Y1800D02*
D03*
Y3800D02*
D03*
X4800D02*
D03*
X3200Y4600D02*
D03*
M02*
```

Appendix 6.1 CircuitA Circuit Feature Data File

1.32
0.008
0
0.016
0
0.14
0

Appendix 6.1 CircuitA Circuit Feature Data File

0.004
0
0.006
0
0.008
0

Appendix 7.1 Specification Checking Rules

```
    /*************************************
     **** RULE: CheckGenSpecPlatReq
     *************************************/
MakeRule( CheckGenSpecPlatReq, [],
  ProPlatingReq:CuThickMin < CustPlatCu:MinThick,
  {
  If ( ProPlatingReq:CuThickMin != 0 )
   Then {
      PostMessage( "Board required Cu thickness " # ProPlatingReq:CuThickMin
                 # " is LESS than customer general required min. thickness "
                 # CustPlatCu:MinThick );
      AppendToList( MessageFile:OutGenSpecMessage,
         PlatedMinCuThk # " -- " # ProPlatingReq:CuThickMin
            # < # CustPlatCu:MinThick );
      }
   Else { };
  } );

    /*************************************
     **** RULE: CheckGenSpecSoldMaskPadClear
     *************************************/
MakeRule( CheckGenSpecSoldMaskPadClear, [],
  ProSoldMask:MinClear < CustSMStd:PadClear,
  {
  PostMessage( "Board required solder mask pad clearance " # ProSoldMask:MinClear
             # "
 is LESS than customer general required min. clearance "
             # CustSMStd:PadClear );
  AppendToList( MessageFile:OutGenSpecMessage, SolderPadClear
                                  # " -- " #
                                  ProSoldMask:MinClear
                                  # < # CustSMStd:PadClear );
  } );

    /*************************************
     **** RULE: CheckGenSpecAuFing
     *************************************/
MakeRule( CheckGenSpecAuFing, [],
  ProAuFinger:AuThick < CustAuFing:MinAuThick,
  {
  If ( ProAuFinger:AuThick != 0 )
   Then {
      PostMessage( "Board required gold finger thickness "
                 # ProAuFinger:AuThick # " is LESS than customer general required min. thickness "
                 # CustAuFing:MinAuThick );
      AppendToList( MessageFile:OutGenSpecMessage,
         AuFingerAuThk # " -- " # ProAuFinger:AuThick # <
```

```
            # CustAuFing:MinAuThick );
        }
    Else { };
  } );

   /*************************************
    **** RULE: CheckGenSpecFeatCir
    *************************************/
MakeRule( CheckGenSpecFeatCir, [],
  TRUE,
  {
  ForwardChain( [ NOASSERT ], NULL, CheckGenSpecFeatCirWidth );
  ForwardChain( [ NOASSERT ], NULL, CheckGenSpecFeatCirSpace );
  ForwardChain( [ NOASSERT ], NULL, CheckGenSpecFeatCirAnnRing );
  } );

   /*************************************
    **** RULE: CheckGenSpecFeatCirWidth
    *************************************/
MakeRule( CheckGenSpecFeatCirWidth, [],
  ProFeatCircuit:CirWidth < CustCirStd:MinLnWid,
  {
  PostMessage( "Circuit width " # ProFeatCircuit:CirWidth # " is SMALLER than customer general
accepted circuit width "
            # CustCirStd:MinLnWid );
  AppendToList( MessageFile:OutGenSpecMessage, CircuitWidth
                                # " -- " #
                                ProFeatCircuit:CirWidth
                                # < # CustCirStd:MinLnWid );
  } );

   /*************************************
    **** RULE: CheckGenSpecFeatCirSpace
    *************************************/
MakeRule( CheckGenSpecFeatCirSpace, [],
  ProFeatCircuit:SpaceWidth < CustCirStd:MinLnSpac,
  {
  PostMessage( "Board circuit spacing " # ProFeatCircuit:SpaceWidth
            # " is SMALLER than customer general accepted circuit spacing "
            # CustCirStd:MinLnSpac );
  AppendToList( MessageFile:OutGenSpecMessage, CircuitSpacing
                                # " -- " #
                                ProFeatCircuit:SpaceWidth
                                # < # CustCirStd:MinLnSpac );
  } );

   /*************************************
    **** RULE: CheckGenSpecFeatCirAnnRing
    *************************************/
MakeRule( CheckGenSpecFeatCirAnnRing, [],
  ProFeatCircuit:AnnularRing < CustCirStd:MinAnnRing,
```

```
  {
  PostMessage( "Board circuit annular ring " # ProFeatCircuit:AnnularRing
            # " is SMALLER than customer general accepted annular ring "
            # CustCirStd:MinAnnRing );
  AppendToList( MessageFile:OutGenSpecMessage, CirAnnularRing
                                # " -- " #
                                ProFeatCircuit:AnnularRing
                                # < # CustCirStd:MinAnnRing );
  } );

/*************************************
    **** RULE: CheckGenSpecSoldMaskMatl
    *************************************/
MakeRule( CheckGenSpecSoldMaskMatl, [],
  Not( CustSMMatl:SMMaterial #= ProSoldMask:MaterialType ),
  {
  PostMessage( "Board required solder mask material " # ProSoldMask:MaterialType
            # " is NOT customer general required " # CustSMMatl:SMMaterial );
  AppendToList( MessageFile:OutGenSpecMessage, SolderMaskMatl
                                # " -- " #
                                ProSoldMask:MaterialType
                                # "Not accepted" );
  } );

   /*************************************
    **** RULE: CheckGenSpecSoldMask
    *************************************/
MakeRule( CheckGenSpecSoldMask, [],
  Not( Null?( ProSoldMask:MaterialType ) ),
  {
  ForwardChain( [ NOASSERT ], NULL, CheckGenSpecSoldMaskPadClear );
  ForwardChain( [ NOASSERT ], NULL, CheckGenSpecSoldMaskMatl );
  } );
```

Appendix 7.2 Process Selection Rules

```
    /*************************************
     **** RULE: PSElectlessPlating
     *************************************/
MakeRule( PSElectlessPlating, [],
  ProPlatingReq:CuThickMin > 0,
  {
  AppendToList( SuggProInHou:ProcessName, ElectrolessCuPlat );
  Assert( InElelessCu, Width );
  Assert( InElelessCu, Length );
  ForwardChain( NULL, CheckElelessCuLength, CheckElelessCuWidth );
  AppendToList( SuggProInHou:ProcessName, PatternCuPlat );
  Assert( InCu, Width );
  Assert( InCu, Length );
  ForwardChain( NULL, CheckElecPlatProcessWidth, CheckElecPlatProcessLength );
  } );

    /*************************************
     **** RULE: PSPatternNiPlat
     *************************************/
MakeRule( PSPatternNiPlat, [],
  ProPlatingReq:SurfaceFinType #= Ni And Not( Member?( SuggProInHou:ProcessName,
                                  PatternNiPlat ) ),
  {
  AppendToList( SuggProInHou:ProcessName, PatternNiPlat );
  Assert( InNi, Width );
  Assert( InNi, Length );
  ForwardChain( NULL, CheckElecPlatProcessWidth, CheckElecPlatProcessLength );
  } );

    /*************************************
     **** RULE: PSPatternSnPbPlat
     *************************************/
MakeRule( PSPatternSnPbPlat, [],
  ProPlatingReq:SurfaceFinType #= SnPb And Not( Member?
        ( SuggProInHou:ProcessName, PatternSnPbPlat ) ),
  {
  AppendToList( SuggProInHou:ProcessName, PatternSnPbPlat );
  Assert( InSnPb, Width );
  Assert( InSnPb, Length );
  ForwardChain( NULL, CheckElecPlatProcessWidth, CheckElecPlatProcessLength );
  } );
```

```
    /*************************************
     **** RULE: PSPatternAuPlat
     *************************************/
MakeRule( PSPatternAuPlat, [],
  ProPlatingReq:SurfaceFinType #= Au And Not( Member?( SuggProInHou:ProcessName,
                                  PatternAuPlat ) ),
  {
  AppendToList( SuggProInHou:ProcessName, PatternAuPlat );
  Assert( InAu, Width );
  Assert( InAu, Length );
  ForwardChain( NULL, CheckElecPlatProcessWidth, CheckElecPlatProcessLength );
  } );

    /*************************************
     **** RULE: PSAuFinger
     *************************************/
MakeRule( PSAuFinger, [],
  ProAuFinger:AuThick > 0 And Not( Member?( SuggProInHou:ProcessName,
                              AuFinger ) ),
  {
  AppendToList( SuggProInHou:ProcessName, AuFinger );
  Assert( InAuFing, Height );
  ForwardChain( NULL, CheckAuFingDimen );
  } );

    /*************************************
     **** RULE: PSImageTransSSCir
     *************************************/
MakeRule( PSImageTransSSCir, [],
  ProFeatCircuit:CirWidth >= InSilkScreen:Resolution And ProFeatCircuit:SpaceWidth
    >= InSilkScreen:Resolution And Not( Member?( SuggProInHou:ProcessName,
                                ImTranCirSilkScreen ) ),
  {
  AppendToList( SuggProInHou:ProcessName, ImTranCirSilkScreen );
  Assert( InSilkScreen, Width );
  Assert( InSilkScreen, Length );
  ForwardChain( NULL, CheckImageTranDimen );
  } );

    /*************************************
     **** RULE: PSImageTransDFCir
     *************************************/
MakeRule( PSImageTransDFCir, [],
  ProFeatCircuit:CirWidth >= InDryFilm:Resolution And ProFeatCircuit:SpaceWidth
    >= InDryFilm:Resolution And Not( Member?( SuggProInHou:ProcessName,
                              ImTranCirSilkScreen ) )
    And Not( Member?( SuggProInHou:ProcessName, ImTranCirWetFilm ) )
    And Not( Member?( SuggProInHou:ProcessName, ImTranCirDryFilm ) ),
  {
  AppendToList( SuggProInHou:ProcessName, ImTranCirDryFilm );
  Assert( InDryFilm, Width );
```

```
  Assert( InDryFilm, Length );
  ForwardChain( NULL, CheckImageTranDimen );
  } );

   /*************************************
    **** RULE: PSImageTransWFCir
    *************************************/
MakeRule( PSImageTransWFCir, [],
  ProFeatCircuit:CirWidth >= InWetFilm:Resolution And ProFeatCircuit:SpaceWidth
    >= InWetFilm:Resolution And Not( Member?( SuggProInHou:ProcessName,
                                ImTranCirSilkScreen ) )
    And Not( Member?( SuggProInHou:ProcessName, ImTranCirWetFilm ) ),
  {
  AppendToList( SuggProInHou:ProcessName, ImTranCirWetFilm );
  Assert( InWetFilm, Width );
  Assert( InWetFilm, Length );
  ForwardChain( NULL, CheckImageTranDimen );
  } );

   /*************************************
    **** RULE: PSImStripCuEtching
    *************************************/
MakeRule( PSImStripCuEtching, [],
  ( Member?( SuggProInHou:ProcessName, ImTranCirDryFilm )
      Or Member?( SuggProInHou:ProcessName, ImTranCirWetFilm )
      Or Member?( SuggProInHou:ProcessName, ImTranCirSilkScreen ) )
    And Not( Member?( SuggProInHou:ProcessName, CuEtching ) ),
  {
  AppendToList( SuggProInHou:ProcessName, CuEtching );
  AppendToList( SuggProInHou:ProcessName, CirImageStrip );
  } );
SetRulePriority( PSImStripCuEtching, -10 );

   /*************************************
    **** RULE: PSNCDrill
    *************************************/
MakeRule( PSNCDrill, [],
  ProHoleSpec:LocDev >= InNCDrill:MachinedAcc And ProPlatingReq:CuThickMin
    > 0 And Not( Member?( SuggProInHou:ProcessName, NCDrilling ) ),
  {
  AppendToList( SuggProInHou:ProcessName, NCDrilling );
  Assert( InNCDrill, NCLength );
  Assert( InNCDrill, NCWidth );
  ForwardChain( NULL, CheckNCDrillDimen );
  } );
```

```
    /*************************************
     **** RULE: PSImageTransWFSM
     *************************************/
MakeRule( PSImageTransWFSM, [],
  ProSoldMask:MinClear >= InWetFilm:Resolution And ProProdDetails:BatchSize
    <= 799 And ProProdDetails:WkDateAvail <= 3 And Not( Member?
        ( SuggProInHou:ProcessName, ImTranSMSilkScreen ) )
    And Not( Member?( SuggProInHou:ProcessName, ImTranSMWetFilm ) )
    And Not( Null?( ProSoldMask:MaterialType ) ),
  {
  AppendToList( SuggProInHou:ProcessName, ImTranSMWetFilm );
  Assert( InWetFilm, Width );
  Assert( InWetFilm, Length );
  ForwardChain( NULL, CheckImageTranDimen );
  } );

    /*************************************
     **** RULE: PSImageTransDFSM
     *************************************/
MakeRule( PSImageTransDFSM, [],
  ProSoldMask:MinClear >= InDryFilm:Resolution And ProProdDetails:BatchSize
    <= 499 And ProProdDetails:WkDateAvail >= 0 And Not( Member?
        ( SuggProInHou:ProcessName, ImTranSMSilkScreen ) )
    And Not( Member?( SuggProInHou:ProcessName, ImTranSMWetFilm ) )
    And Not( Member?( SuggProInHou:ProcessName, ImTranSMDryFilm ) )
    And Not( Null?( ProSoldMask:MaterialType ) ),
  {
  AppendToList( SuggProInHou:ProcessName, ImTranSMDryFilm );
  Assert( InDryFilm, Width );
  Assert( InDryFilm, Length );
  ForwardChain( NULL, CheckImageTranDimen );
  } );

    /*************************************
     **** RULE: PSImageTransSSSM
     *************************************/
MakeRule( PSImageTransSSSM, [],
  ProSoldMask:MinClear >= InSilkScreen:Resolution And ProProdDetails:BatchSize
    > 800 And ProProdDetails:WkDateAvail >= 4 And Not( Member?
        ( SuggProInHou:ProcessName, ImTranSMSilkScreen ) )
        And Not( Null?( ProSoldMask:MaterialType ) ),
  {
  AppendToList( SuggProInHou:ProcessName, ImTranSMSilkScreen );
  Assert( InSilkScreen, Width );
  Assert( InSilkScreen, Length );
  ForwardChain( NULL, CheckImageTranDimen );
  } );
```

```
    /*************************************
     **** RULE: PSSnPbReflow
     *************************************/
MakeRule( PSSnPbReflow, [],
  Member?( SuggProInHou:ProcessName, PatternSnPbPlat )
    And Not( Member?( SuggProInHou:ProcessName, SnPbReflow ) )
    And Not( ProPlatingReq:SurfaceFinType #= BareCopper ),
  {
  AppendToList( SuggProInHou:ProcessName, SnPbReflow );
  Assert( InReflow, Width );
  ForwardChain( NULL, CheckSnPbReflowDimen );
  } );

    /*************************************
     **** RULE: PSCompMarking
     *************************************/
MakeRule( PSCompMarking, [],
  Not( Null?( ProCompMark:CompMaterial ) ),
  AppendToList( SuggProInHou:ProcessName, CompMarkPrint ) );

    /*************************************
     **** RULE: PSPanPatCuPlat
     *************************************/
MakeRule( PSPanPatCuPlat, [],
  ProPlatingReq:CuThickMin > 0,
  {
  AppendToList( SuggProInHou:ProcessName, PanelCuPlat );
  Assert( InPanelPlat, Width );
  Assert( InPanelPlat, Length );
  ForwardChain( NULL, CheckPanCuDimen );
  } );

    /*************************************
     **** RULE: PSBlank
     *************************************/
MakeRule( PSBlank, [],
  ProPanelDimen:DimenDev >= InBlank:Accuracy And Not( Member?
        ( SuggProInHou:ProcessName, OutLineBlank ) ),
  {
  AppendToList( SuggProInHou:ProcessName, OutLineBlank );
  } );
```

```
    /*************************************
     **** RULE: PSPatternBareCuPlat
     *************************************/
MakeRule( PSPatternBareCuPlat, [],
  ProPlatingReq:SurfaceFinType #= BareCopper And Not( Member?
        ( SuggProInHou:ProcessName, PatternSnPbPlat ) ),
  {
  AppendToList( SuggProInHou:ProcessName, PatternSnPbPlat );
  AppendToList( SuggProInHou:ProcessName, SnPbStripping );
  AppendToList( SuggProInHou:ProcessName, SolderLevelling );
  Assert( InSnPb, Width );
  Assert( InSnPb, Length );
  ForwardChain( NULL, CheckElecPlatProcessWidth, CheckElecPlatProcessLength );
  Assert( InSnPbLevelling, Length );
  Assert( InSnPbLevelling, Width );
  ForwardChain( NULL, CheckSnPbLevDimen );
  } );

    /*************************************
     **** RULE: PSRouting
     *************************************/
MakeRule( PSRouting, [],
  TRUE,
  {
  If Not( Member?( SuggProInHou:ProcessName, OutLineBlank ) )
   Then AppendToList( SuggProInHou:ProcessName, OutLineRout );
  } );

/*************************************
     **** RULE: PSHolePunch
     *************************************/
MakeRule( PSHolePunch, [],
  ProPlatingReq:CuThickMin #= 0 And Not( Member?( SuggProInHou:ProcessName,
                              2ndDrillOrHolePunch ) ),
  AppendToList( SuggProInHou:ProcessName, 2ndDrillOrHolePunch ) );
```

Appendix 7.3 Process Sequencing Rules (Methods)

```
/************** METHOD: SequenceProcess  **************/
MakeMethod( SuggProInHou, SequenceProcess, [],
{
  ResetValue( SuggProInHou:ProcessNameSeqNo );
  EnumList( SuggProInHou:ProcessName, name, AppendToList( SuggProInHou:ProcessNameSeqNo,
     {  GetNthElem( MfgSeqNo:SeqNo, GetElemPos( ProcessName:Name, name ) );    }
     ) );
  ClearList( SuggProInHou:ProcessNameInSeq );
  For numb From 0 To 300 By 10
       Do {
      Global:dumno = GetElemPos( SuggProInHou:ProcessNameSeqNo,
                    numb );
      If ( Global:dumno > 0 )
        Then {
           AppendToList( SuggProInHou:ProcessNameInSeq,
              GetNthElem( SuggProInHou:ProcessName, Global:dumno ) );
           };
       };
} );
```

Appendix 7.4 Process Capability Checking Rules

```
/*************************************
    **** RULE: CheckElecPlatProcessWidth
    *************************************/
MakeRule( CheckElecPlatProcessWidth, [process|InElecPlat],
  process:Width < ProPanelDimen:Width,
  {
  PostMessage( "Note:- " # ReturnProcessName( process )
            # " process width is smaller than product panel width.  "
            # ReturnProcessName( process )
            # " process cannot perform in-house." );
  AppendToList( MessageFile:ProCanNotMfgInHouse,
    ReturnProcessName( process ) # "--PanelWidth>Process" );
  } );

   /*************************************
    **** RULE: CheckElecPlatProcessLength
    *************************************/
MakeRule( CheckElecPlatProcessLength, [process|InElecPlat],
  process:Length < ProPanelDimen:Length,
  {
  PostMessage( "Note:- " # ReturnProcessName( process )
            # " process length is smaller than product panel length.  "
            # ReturnProcessName( process )
            # " process cannot perform in-house." );
  AppendToList( MessageFile:ProCanNotMfgInHouse,
    ReturnProcessName( process ) # "--PanelLength>Process" );
  } );

   /*************************************
    **** RULE: CheckElelessCuLength
    *************************************/
MakeRule( CheckElelessCuLength, [],
  InElelessCu:Length < ProPanelDimen:Length,
  {
  PostMessage( "Note:- In-house Electroless Copper Plating process length is smaller than product panel
length.  Process cannot perform in-house." );
  AppendToList( MessageFile:ProCanNotMfgInHouse, "ElelessCuPlat--PanelLength>Process" );
  } );
```

```
    /*************************************
     **** RULE: CheckElelessCuWidth
     *************************************/
MakeRule( CheckElelessCuWidth, [],
  InElelessCu:Width < ProPanelDimen:Width,
  {
  PostMessage( "Note:- In-house Electroless Copper plating process width is smaller than product panel
width.  Process cannot perform in-house." );
  AppendToList( MessageFile:ProCanNotMfgInHouse, "ElelessCuPlat--PanelWidth>Process" );
  } );

    /*************************************
     **** RULE: CheckNCDrillDimen
     *************************************/
MakeRule( CheckNCDrillDimen, [],
  InNCDrill:NCLength < ProPanelDimen:Length Or InNCDrill:NCWidth
    < ProPanelDimen:Width,
  {
  PostMessage( "One side of product panel size longer than NC drilling machine size.  Product cannot be
produced in-house." );
  AppendToList( MessageFile:ProCanNotMfgInHouse, "NCDrilling--PanelSize>MCSize" );
  } );

    /*************************************
     **** RULE: CheckImageTranDimen
     *************************************/
MakeRule( CheckImageTranDimen, [process|InImageTran],
  process:Length < ProPanelDimen:Length Or process:Width < ProPanelDimen:Width,
  {
  PostMessage( "Note:- One dimension of  " # process # " process is smaller than panel dimension.
Product cannot be manufactured in-house." );
  AppendToList( MessageFile:ProCanNotMfgInHouse,
    process # "--PanelSize>Process" );
  } );

    /*************************************
     **** RULE: CheckAuFingDimen
     *************************************/
MakeRule( CheckAuFingDimen, [],
  InAuFing:Height < ProAuFinger:Height,
  {
  PostMessage( "Note:- Product gold finger height is longer than in-house gold process.  Product cannot
be manufactured in-house." );
  AppendToList( MessageFile:ProCanNotMfgInHouse, "AuFinger--FingerHeight>Process" );
  } );
```

```
    /*************************************
     **** RULE: CheckSnPbLevDimen
     *************************************/
MakeRule( CheckSnPbLevDimen, [],
  InSnPbLevelling:Width < ProPanelDimen:Width Or InSnPbLevelling:Length
     < ProPanelDimen:Length,
  {
  PostMessage( "Note:- One dimension of panel is longer than levelling machine dimension.  Product
cannot be manufactured in-house." );
  AppendToList( MessageFile:ProCanNotMfgInHouse, "SnPbLevelling--PanelSize>Process" );
  } );

    /*************************************
     **** RULE: CheckSnPbReflowDimen
     *************************************/
MakeRule( CheckSnPbReflowDimen, [],
  InReflow:Width < ProPanelDimen:Width,
  {
  PostMessage( "Note:- Product panel width is longer than in-house reflow machine width.  Product
cannot be manufactured in-house." );
  AppendToList( MessageFile:ProCanNotMfgInHouse, "SnPbReflow--PanelWidth>Process" );
  } );

    /*************************************
     **** RULE: CheckPanCuDimen
     *************************************/
MakeRule( CheckPanCuDimen, [],
  Max( InPanelPlat:Width, InPanelPlat:Length ) < Max( ProPanelDimen:Length,
                                        ProPanelDimen:Width ),
  {
  PostMessage( "Note:- In-house Panel Copper Plating process dimension smaller than product
dimension.  Panel copper process cannot perform in-house." );
  AppendToList( MessageFile:ProCanNotMfgInHouse, "PanelCuPlat--PanelDimension>Process" );
  } );
```

Appendix 7.5 Machine Selection Rules (Methods)

```
    /*************************************
     **** RULE: MCSelectionRules
     *************************************/
MakeRule( MCSelectionRules, [],
  TRUE,
  {
  EnumSubClasses( MCNameList, x,
     {
     ClearList( x:MCListing );
     } );
  EnumList( SuggProInHou:ProcessName, x,
     {
     If ( x #= NCDrilling )
      Then SendMessage( DrillMC, SelectMachine );
     If ( x #= OutLineBlank )
      Then SendMessage( BlankMC, SelectMachine );
     If ( x #= ImTranCirSilkScreen )
      Then SendMessage( SilkScreenMC, SelectMachine, ProFeatCircuit:AnnularRing,
            ITCirSilkScreen );
     If ( x #= ImTranSMSilkScreen )
      Then SendMessage( SilkScreenMC, SelectMachine, ProSoldMask:MinClear,
            ITSMSilkScreen );
     If ( x #= CompMarkPrint )
      Then SendMessage( SilkScreenMC, SelectMachine, 0.3, ITCompMark );
     } );
  ShowImage( Button21 );
  } );

    /*************************************
     **** CLASS: DrillMC
     *************************************/
MakeClass( DrillMC, MCChar );

/************** METHOD: SelectMachine  **************/
MakeMethod( DrillMC, SelectMachine, [],
  {
  ClearList( NCDrilling:MCListing );
  ForAll [ mc|DrillMC ]
       {
       If ( mc:MachinedAcc < ProHoleSpec:LocDev And mc:NCLength
            > ProPanelDimen:Length And mc:NCWidth > ProPanelDimen:Width
            And mc:MaxHoleSize > ProHoleSpec:MaxHoleSize
            And mc:MinHoleSize < ProHoleSpec:MinHoleSize )
        Then AppendToList( NCDrilling:MCListing, mc:MachineName );
       };
  If ( LengthList( NCDrilling:MCListing ) == 0 )
    Then {
       AppendToList( NCDrilling:MCListing, "*****NOSuitableNCDrillingMC" );
```

```
        AppendToList( MessageFile:MCSelectionNoMessage, NCDrilling );
        PostMessage( "NO Suitable NC Drilling Machine In-house for this Product." );
        };
    } );

    /*************************************
     **** CLASS: BlankMC
     *************************************/
MakeClass( BlankMC, MCChar );

/************** METHOD: SelectMachine  **************/
MakeMethod( BlankMC, SelectMachine, [],
    {
    ClearList( BlankingMC:MCListing );
    ForAll [ mclBlankMC ]
        {
        If ( mc:Accuracy < ProPanelDimen:DimenDev And mc:Length
              > ProPanelDimen:Length And mc:Width > ProPanelDimen:Width )
         Then AppendToList( BlankingMC:MCListing, mc:MachineName );
        };
    If ( LengthList( BlankingMC:MCListing ) == 0 )
     Then {
        AppendToList( BlankingMC:MCListing, "*****NOSuitableBlankingMC" );
        AppendToList( MessageFile:MCSelectionNoMessage, Blanking );
        PostMessage( "NO Suitable Blanking Machine Find In-house for this Product." );
        };
    } );

    /*************************************
     **** CLASS: SilkScreenMC
     *************************************/
MakeClass( SilkScreenMC, MCChar );

/************** METHOD: SelectMachine  **************/
MakeMethod( SilkScreenMC, SelectMachine, [ACC MCLIST ],
    {
    ClearList( MCLIST:MCListing );
    ForAll [ mclSilkScreenMC ]
        {
        If ( mc:Resolution < ACC And mc:Length > ProPanelDimen:Length
              And mc:Width > ProPanelDimen:Width )
         Then AppendToList( MCLIST:MCListing, mc:MachineName );
        };
    If ( LengthList( MCLIST:MCListing ) == 0 )
     Then {
        AppendToList( MCLIST:MCListing, "*****NOSuitableSilkScreenMC" );
        AppendToList( MessageFile:MCSelectionNoMessage, MCLIST );
        PostMessage( MCLIST # " NO Suitable Silk Screen Machine Find In-house for this Process." );
        };
    } );
```

Appendix 7.6 Process Recipe Generation Rules

```
/*************************************
    **** RULE: RecipeGenPatternNi
    *************************************/
MakeRule( RecipeGenPatternNi, [],
  TRUE,
  {
  RecPatPlatNi:Current = CalPatternArea( ProPanelDimen:Width,
                    ProPanelDimen:Length, ProPanelDimen:UnitPerPanel,
                    ProPanelDimen:UnitWidth, ProPanelDimen:UnitLength,
                    ProFeatCircuit:CircuitArea ) * InNi:CurrentDensity;
  RecPatPlatNi:PlatingTime = ( InNi:TimeToDeposit * ProPlatingReq:SurfaceFinThick * 10 );
  } );

   /*************************************
    **** RULE: RecipeGenPanelCu
    *************************************/
MakeRule( RecipeGenPanelCu, [],
  TRUE,
  {
  RecPanPlatCu:Current = ( ProPanelDimen:Length * ProPanelDimen:Width
                    / 25.4 / 25.4 / 144 * InPanelPlat:CurrentDensity );
  If ( ProLamChar:CuFoilSide #= Double )
   Then RecPanPlatCu:Current = RecPanPlatCu:Current * 2;
  RecPanPlatCu:PlatingTime = ( InPanelPlat:TimeToDeposit * 5 );
  } );

   /*************************************
    **** RULE: RecipeGenPatternCu
    *************************************/
MakeRule( RecipeGenPatternCu, [],
  TRUE,
  {
  RecPatPlatCu:Current = CalPatternArea( ProPanelDimen:Width,
                    ProPanelDimen:Length, ProPanelDimen:UnitPerPanel,
                    ProPanelDimen:UnitWidth, ProPanelDimen:UnitLength,
                    ProFeatCircuit:CircuitArea ) * InCu:CurrentDensity;
  RecPatPlatCu:PlatingTime = ( InPanelPlat:TimeToDeposit * ( ( ProPlatingReq:CuThickMin
                    * 10 ) - 5 ) );
  } );
```

```
    /*************************************
     **** RULE: RecipeGenPatternAu
     *************************************/
MakeRule( RecipeGenPatternAu, [],
  TRUE,
  {
  RecPatPlatAu:Current = CalPatternArea( ProPanelDimen:Width,
                    ProPanelDimen:Length, ProPanelDimen:UnitPerPanel,
                    ProPanelDimen:UnitWidth, ProPanelDimen:UnitLength,
                    ProFeatCircuit:CircuitArea ) * InAu:CurrentDensity;
  RecPatPlatAu:PlatingTime = ( InAu:TimeToDeposit * ProPlatingReq:SurfaceFinThick  * 10 );
  } );

    /*************************************
     **** RULE: RecipeGenPatternSnPb
     *************************************/
MakeRule( RecipeGenPatternSnPb, [],
  TRUE,
  {
  RecPatPlatSnPb:Current = CalPatternArea( ProPanelDimen:Width,
                      ProPanelDimen:Length, ProPanelDimen:UnitPerPanel,
                      ProPanelDimen:UnitWidth, ProPanelDimen:UnitLength,
                      ProFeatCircuit:CircuitArea ) * InSnPb:CurrentDensity;
  RecPatPlatSnPb:PlatingTime = ( InSnPb:TimeToDeposit * ProPlatingReq:SurfaceFinThick * 10 );
  } );

    /*************************************
     **** RULE: CheckGenNCDrillRecipe
     *************************************/
MakeRule( CheckGenNCDrillRecipe, [],
  Member?( SuggProInHou:ProcessNameInSeq, NCDrilling ),
  SendMessage( RecNCDrill, GenerateProcessRecipe ) );

    /*************************************
     **** RULE: CheckGenElessCuRecipe
     *************************************/
MakeRule( CheckGenElessCuRecipe, [],
  Member?( SuggProInHou:ProcessNameInSeq, ElectrolessCuPlat ),
  SendMessage( RecElessCuPlat, GenerateProcessRecipe ) );

    /*************************************
     **** RULE: CheckGenPanelCuPlatRecipe
     *************************************/
MakeRule( CheckGenPanelCuPlatRecipe, [],
  Member?( SuggProInHou:ProcessNameInSeq, PanelCuPlat ),
  SendMessage( RecPanPlatCu, GenerateProcessRecipe ) );
```

```
    /*************************************
     **** RULE: CheckGenPatternCuPlatRecipe
     *************************************/
MakeRule( CheckGenPatternCuPlatRecipe, [],
   Member?( SuggProInHou:ProcessNameInSeq, PatternCuPlat ),
   SendMessage( RecPatPlatCu, GenerateProcessRecipe ) );

    /*************************************
     **** RULE: CheckGenPatternNiPlatRecipe
     *************************************/
MakeRule( CheckGenPatternNiPlatRecipe, [],
   Member?( SuggProInHou:ProcessNameInSeq, PatternNiPlat ),
   SendMessage( RecPatPlatNi, GenerateProcessRecipe ) );

    /*************************************
     **** RULE: CheckGenPatternAuPlatRecipe
     *************************************/
MakeRule( CheckGenPatternAuPlatRecipe, [],
   Member?( SuggProInHou:ProcessNameInSeq, PatternAuPlat ),
   SendMessage( RecPatPlatAu, GenerateProcessRecipe ) );

    /*************************************
     **** RULE: CheckGenPatternSnPbPlatRecipe
     *************************************/
MakeRule( CheckGenPatternSnPbPlatRecipe, [],
   Member?( SuggProInHou:ProcessNameInSeq, PatternSnPbPlat ),
   SendMessage( RecPatPlatSnPb, GenerateProcessRecipe ) );

    /*************************************
     **** RULE: RecipeGenAuFinger
     *************************************/
MakeRule( RecipeGenAuFinger, [],
   TRUE,
   {
   RecAuFinger:AuCurrent = ( ProAuFinger:PlatAreaPerUnit / 144
                    * ProAuFinger:UnitPerEdge * InAu:CurrentDensity );
   RecAuFinger:AuPlatingTime = ( InAu:TimeToDeposit * ProAuFinger:AuThick  * 10 );
   RecAuFinger:NiCurrent = ( ProAuFinger:PlatAreaPerUnit / 144
                    * ProAuFinger:UnitPerEdge *  InNi:CurrentDensity );
   RecAuFinger:NiPlatingTime = ( InNi:TimeToDeposit * ProAuFinger:NiThick * 10 );
   } );

    /*************************************
     **** RULE: CheckGenAuFingerRecipe
     *************************************/
MakeRule( CheckGenAuFingerRecipe, [],
   Member?( SuggProInHou:ProcessNameInSeq, AuFinger ),
   SendMessage( RecAuFinger, GenerateProcessRecipe ) );
```

Appendix 8 Developed System Support Functions

```
    /*************************************
     **** FUNCTION: ClearSession2
     *************************************/
MakeFunction( ClearSession2, [],
  {
  HideImage( MultipleListBox2 );
  HideWindow( ReviewApproveProcess );
  ShowWindow( SESSION );
  } );

    /*************************************
     **** FUNCTION: ClearSession1
     *************************************/
MakeFunction( ClearSession1, [],
  {
  HideWindow( InPutCustProd );
  SendMessage( CustProdReqt, StoreData );
  ShowWindow( SESSION );
  } );

    /*************************************
     **** FUNCTION: FindInHouPro
     *************************************/
MakeFunction( FindInHouPro, [],
  {
  SendMessage( SuggProInHou, ProcessSelection );
  } );

    /*************************************
     **** FUNCTION: OpenSession1
     *************************************/
MakeFunction( OpenSession1, [],
  {
  ShowWindow( InPutCustProd );
  HideImage( MultipleListBox2 );
  } );

    /*************************************
     **** FUNCTION: OpenSession2
     *************************************/
MakeFunction( OpenSession2, [],
  {
  If Class?( ReqDocInfo )
   Then {
      HideImage( MultipleListBox2 );
      HideImage( MultipleListBox1 );
      HideImage( Button24 );
```

```
        HideImage( Button25 );
        ShowWindow( ReviewApproveProcess );
        }
    Else {
        PostMessage( "ERROR!! Customer and/or Product Data Base NOT Loaded.  Load Data Base
First." );
        };
  } );

    /*************************************
     **** FUNCTION: SaveProdCharProcessPlan
     *************************************/
MakeFunction( SaveProdCharProcessPlan, [],
  {
  SendMessage( SuggProInHou, OpenFileToWriteInst );
  SendMessage( SuggProInHou, WriteProdReqt );
  SendMessage( SuggProInHou, WriteProcessSeq );
  SendMessage( SuggProInHou, WriteOperationInst );
  SendMessage( SuggProInHou, CloseInstWriteFile );
  } );

    /*************************************
     **** FUNCTION: MCSelection
     *************************************/
MakeFunction( MCSelection, [],
  {
  If Class?( ReqDocInfo )
    Then {
          SendMessage(MCNameList, MCSelection);
        }
    Else {
        PostMessage( "ERROR!! Customer and/or Product Data Base NOT Loaded.  Load Data Base
First." );
        };
  } );

    /*************************************
     **** FUNCTION: ProcessRecipeGenFunction
     *************************************/
MakeFunction( ProcessRecipeGenFunction, [],
  {
  If Class?( ReqDocInfo )
    Then {
        SendMessage( ProcessRecipe, GenerateProcessRecipe );
        SaveProdCharProcessPlan( );
        ClearTranscriptImage( Transcript1 );
        DisplayFile( Transcript1, Global:DriveDir # Global:dumstring );
        ShowWindow( ProcessPlanOutputSession );
        ShowImage( Button16 );
        ShowImage( Button26 );
        }
```

```
    Else {
        PostMessage( "ERROR!! Customer and/or Product Data Base NOT Loaded.  Load Data Base
First." );
        };
  } );

    /*************************************
     **** FUNCTION: StoreKnowledge
     *************************************/
MakeFunction( StoreKnowledge, [ClassName],
  {
  OpenWriteFile( Global:TempFileName );
  EnumSubClasses( ClassName, dummy, WriteClass( dummy ) );
  CloseWriteFile( );
  } );

    /*************************************
     **** FUNCTION: BuildUpKnowFileName
     *************************************/
MakeFunction( BuildUpKnowFileName, [TypeOfKnow],
  {
  Global:TempFileName = Global:DriveDir # TypeOfKnow # .kap;
  } );

    /*************************************
     **** FUNCTION: ReadKnowledge
     *************************************/
MakeFunction( ReadKnowledge, [],
  {
  OpenReadFile( Global:TempFileName );
  InterpretFile( Global:TempFileName );
  CloseReadFile( );
  } );

    /*************************************
     **** FUNCTION: LdCustSpBdReqt
     *************************************/
MakeFunction( LdCustSpBdReqt, [],
  {
  If Class?( ReqDocInfo )
    Then PostMessage( "ERROR!! Customer and/or Product Data Base Already Loaded!!!" )
    Else {
        SendMessage( CustGenSpec, LoadData );
        SendMessage( CustProdReqt, LoadData );
        SendMessage( CustProdReqt, CheckCustGenSpecMethod );
        PostMessage( "Customer General Specification Data and Product Requirement Data Successfully
Loaded. ");
        ShowImage( Button18 );
        HideImage( Button23 );
        HideImage( Button22 );
        HideImage( Button14 );
```

```
        HideImage( Button17 );
        };
    } );

    /*************************************
     **** FUNCTION: UpProCap
     *************************************/
MakeFunction( UpProCap, [],
   SendMessage( ProCapKnow, Update ) );

    /*************************************
     **** FUNCTION: InUpCustGenSpec
     *************************************/
MakeFunction( InUpCustGenSpec, [],
   {
   PostInputForm( "Please Input '1' for input new cust. spec.
'2' for up-dating cust. spec.",
      Global, TempString, "Input number 1 or 2" );
   If ( Global:TempString #= 1 )
    Then {
        SendMessage( CustGenSpec, Define );
        SendMessage( CustGenSpec, StoreData );
        };
   If ( Global:TempString #= 2 )
    Then {
        SendMessage( CustGenSpec, LoadData );
        SendMessage( CustGenSpec, Update );
        SendMessage( CustGenSpec, StoreData );
        };
   } );

    /*************************************
     **** FUNCTION: InUpProdReqt
     *************************************/
MakeFunction( InUpProdReqt, [],
   {
   If Class?( ReqDocInfo )
      Then PostMessage( "ERROR!! Customer and/or Product Data Base Already Loaded!!!" )
      Else {
        SendMessage( CustGenSpec, LoadData );
        PostInputForm( "Please Input '1' for new prod. reqt. '2' for up-dating existing reqt.",
            Global, TempString, "Input number 1 or 2 " );
        If ( Global:TempString #= 1 )
            Then {
              SendMessage( CustProdReqt, Define );
             };
        If ( Global:TempString #= 2 )
            Then {
              SendMessage( CustProdReqt, LoadData );
              SendMessage( CustProdReqt, Update );
              };
```

```
    };
  } );

    /*************************************
     **** FUNCTION: SaveProNameSeqInFile
     *************************************/
MakeFunction( SaveProNameSeqInFile, [],
{
  SendMessage(SuggProInHou, SaveProNameSeqInFile);
  PostMessage( "Process Name and Sequence Saved in File " # Global:dumstring );
} );

    /*************************************
     **** FUNCTION: SequenceProcess
     *************************************/
MakeFunction( SequenceProcess, [],
{
  SendMessage(SuggProInHou, SequenceProcess);
  ShowImage( MultipleListBox1 );
  ShowImage( Button24 );
  ShowImage( Button20 );
} );

    /*************************************
     **** FUNCTION: NextProduct
     *************************************/
MakeFunction( NextProduct, [],
  {
  EnumSubClasses( CustProdReqt, dumname, DeleteClass( dumname ) );
  SendMessage( CustProdReqt, LoadData );
  SendMessage( CustProdReqt, CheckCustGenSpecMethod );
  PostMessage( "New Product Requirement Data Successfully Loaded.
Go to Process Selection Session." );
  ShowImage( Button18 );
  HideImage( Button20 );
  HideImage( Button21 );
  HideImage( Button23 );
  HideImage( Button22 );
  HideImage( Button14 );
  HideImage( Button17 );
  HideImage( Button26 );
  HideImage( Button16 );
  } );

    /*************************************
     **** FUNCTION: FinishProcessPlan
     *************************************/
MakeFunction( FinishProcessPlan, [],
  {
  EnumSubClasses( CustAppSurFin, dumname, DeleteClass( dumname ) );
  EnumSubClasses( CustGenSpec, dumname, DeleteClass( dumname ) );
```

```
  EnumSubClasses( CustProdReqt, dumname, DeleteClass( dumname ) );
  HideImage( Button18 );
  HideImage( Button20 );
  HideImage( Button21 );
  ShowImage( Button23 );
  ShowImage( Button22 );
  ShowImage( Button14 );
  ShowImage( Button17 );
  HideImage( Button16 );
  HideImage( Button26 );
  } );

   /*************************************
    **** FUNCTION: ClosePlanDisplay
    *************************************/
MakeFunction( ClosePlanDisplay, [],
  {
  HideWindow( ProcessPlanOutputSession );
  } );

   /*************************************
    **** FUNCTION: CalPatternArea
    *************************************/
MakeFunction( CalPatternArea, [panwidth panlength noofunit unitwidth unitlength unitcirarea],
  {
  Global:dumno = ((panwidth * panlength) - (unitwidth * unitlength * noofunit))
     * 0.6/25.4/25.4/144;
  If ( ProLamChar:CuFoilSide #= Double)
     Then ( Global:dumno = Global:dumno * 2 );
  Global:dumno + ( unitcirarea / 144 * noofunit );
  } );

   /*************************************
    **** FUNCTION: PostDoc
    *************************************/
MakeFunction( PostDoc, [],
  {
  ShowWindow( ProdDocWindow );
  SetValue(ComboBox1:Value, ReqDocInfo:CustName);
  UpdateImage(ComboBox1);
  SetValue(ComboBox2:Value, ReqDocInfo:PNCust);
  UpdateImage(ComboBox2);
  SetValue(ComboBox3:Value, ReqDocInfo:ProdName);
  UpdateImage(ComboBox3);
  SetValue(ComboBox4:Value , ReqDocInfo:PurOrdNo);
  UpdateImage(ComboBox4);
  SetValue(ComboBox5:Value , ReqDocInfo:PNInHouse);
  UpdateImage(ComboBox5);
  } );

   /*************************************
```

```
    **** FUNCTION: FinProDocWin
    ************************************/
MakeFunction( FinProDocWin, [],
  {
  HideWindow( ProdDocWindow );
  SetValue( Button10:BackgroundColor, 192, 192, 192 );
  } );

    /************************************
    **** FUNCTION: FinQuanDelWin
    ************************************/
MakeFunction( FinQuanDelWin, [],
  {
  HideWindow( ProProdWindow );
  SetValue( Button19:BackgroundColor, 192, 192, 192 );
  ShowImage( Button19 );
  } );

    /************************************
    **** FUNCTION: PostQuantity
    ************************************/
MakeFunction( PostQuantity, [],
  {
  ShowWindow( ProProdWindow );
  SetValue(ComboBox6:Value, ProProdDetails:BatchSize);
  UpdateImage(ComboBox6);
  SetValue(ComboBox7:Value, ProProdDetails:ExpQty);
  UpdateImage(ComboBox7);
  SetValue(ComboBox8:Value, ProProdDetails:WkDateAvail);
  UpdateImage(ComboBox8);
  } );

    /************************************
    **** FUNCTION: FinProLaminWin
    ************************************/
MakeFunction( FinProLaminWin, [],
  {
  HideWindow( ProLaminWindow );
  SetValue( Button8:BackgroundColor, 192, 192, 192 );
  } );

    /************************************
    **** FUNCTION: PostLamChar
    ************************************/
MakeFunction( PostLamChar, [],
  {
  ShowWindow( ProLaminWindow );
  SetValue(ComboBox9:Value, ProLamChar:LaminateType);
  UpdateImage(ComboBox9);
  SetValue(ComboBox10:Value, ProLamChar:LamThickness);
  UpdateImage(ComboBox10);
```

```
  SetValue(ComboBox11:Value, ProLamChar:CuFoilThickness);
  UpdateImage(ComboBox11);
  SetValue(ComboBox12:Value, ProLamChar:CuFoilSide);
  UpdateImage(ComboBox12);
  SetValue(ComboBox13:Value, ProLamChar:Colour);
  UpdateImage(ComboBox13);
  } );

   /*************************************
    **** FUNCTION: FinProPanWin
    *************************************/
MakeFunction( FinProPanWin, [],
  {
  HideWindow( ProPanDimenWindow );
  SetValue( Button7:BackgroundColor, 192, 192, 192 );
  } );

   /*************************************
    **** FUNCTION: PostPanDimen
    *************************************/
MakeFunction( PostPanDimen, [],
  {
  ShowWindow( ProPanDimenWindow );
  SetValue(ComboBox19:Value, ProPanelDimen:UnitWidth);
  UpdateImage(ComboBox19);
  SetValue(ComboBox16:Value, ProPanelDimen:UnitLength);
  UpdateImage(ComboBox16);
  SetValue(ComboBox20:Value, ProPanelDimen:Width);
  UpdateImage(ComboBox20);
  SetValue(ComboBox17:Value, ProPanelDimen:Length);
  UpdateImage(ComboBox17);
  SetValue(ComboBox18:Value, ProPanelDimen:UnitPerPanel);
  UpdateImage(ComboBox18);
  SetValue(ComboBox14:Value, ProPanelDimen:DimenDev);
  UpdateImage(ComboBox14);
  SetValue(ComboBox15:Value, ProPanelDimen:EstDieCost);
  UpdateImage(ComboBox15);

  } );

   /*************************************
    **** FUNCTION: FinProHoleWin
    *************************************/
MakeFunction( FinProHoleWin, [],
  {
  HideWindow( ProHoleWindow );
  SetValue( Button4:BackgroundColor, 192, 192, 192 );
  } );

   /*************************************
    **** FUNCTION: PostHoleSpec
```

```
    ************************************/
MakeFunction( PostHoleSpec, [],
  {
  ShowWindow( ProHoleWindow );
  SetValue(ComboBox21:Value, ProHoleSpec:DiaDev);
  UpdateImage(ComboBox21);
  SetValue(ComboBox22:Value, ProHoleSpec:LocDev);
  UpdateImage(ComboBox22);
  SetValue(ComboBox23:Value, ProHoleSpec:MaxHoleSize);
  UpdateImage(ComboBox23);
  SetValue(ComboBox24:Value, ProHoleSpec:MinHoleSize);
  UpdateImage(ComboBox24);
  } );

    /************************************
     **** FUNCTION: PostFeatCir
     ************************************/
MakeFunction( PostFeatCir, [],
  {
  ShowWindow( ProCircuitWindow );
  SetValue(ComboBox25:Value, ProFeatCircuit:AnnularRing);
  UpdateImage(ComboBox25);
  SetValue(ComboBox26:Value, ProFeatCircuit:CircuitArea);
  UpdateImage(ComboBox26);
  SetValue(ComboBox27:Value, ProFeatCircuit:CirThickness);
  UpdateImage(ComboBox27);
  SetValue(ComboBox28:Value, ProFeatCircuit:CirWidth);
  UpdateImage(ComboBox28);
  SetValue(ComboBox29:Value, ProFeatCircuit:SpaceWidth);
  UpdateImage(ComboBox29);
  } );

    /************************************
     **** FUNCTION: FinProCirWin
     ************************************/
MakeFunction( FinProCirWin, [],
  {
  HideWindow( ProCircuitWindow );
  SetValue( Button12:BackgroundColor, 192, 192, 192 );
  } );

    /************************************
     **** FUNCTION: PostPlatReq
     ************************************/
MakeFunction( PostPlatReq, [],
  {
  ShowWindow( ProPlatWindow );
  SetValue(ComboBox30:Value, ProPlatingReq:CuThickMin);
  UpdateImage(ComboBox30);
  SetValue(ComboBox32:Value, ProPlatingReq:SurfaceFinType);
  UpdateImage(ComboBox32);
```

```
  SetValue(ComboBox31:Value, ProPlatingReq:SurfaceFinThick);
  UpdateImage(ComboBox31);
  } );

   /*************************************
    **** FUNCTION: FinProPlatWin
    *************************************/
MakeFunction( FinProPlatWin, [],
  {
  HideWindow( ProPlatWindow );
  SetValue( Button3:BackgroundColor, 192, 192, 192 );
  } );

   /*************************************
    **** FUNCTION: PostAuFing
    *************************************/
MakeFunction( PostAuFing, [],
  {
  ShowWindow( ProAuFingerWindow );
  SetValue(ComboBox33:Value, ProAuFinger:AuThick);
  UpdateImage(ComboBox33);
  SetValue(ComboBox34:Value, ProAuFinger:Height);
  UpdateImage(ComboBox34);
  SetValue(ComboBox35:Value, ProAuFinger:NiThick);
  UpdateImage(ComboBox35);
  SetValue(ComboBox36:Value, ProAuFinger:PlatAreaPerUnit);
  UpdateImage(ComboBox36);
  SetValue(ComboBox37:Value, ProAuFinger:UnitPerEdge);
  UpdateImage(ComboBox37);
  } );

   /*************************************
    **** FUNCTION: FinAuFingerWin
    *************************************/
MakeFunction( FinAuFingerWin, [],
  {
  HideWindow( ProAuFingerWindow );
  SetValue( Button9:BackgroundColor, 192, 192, 192 );
  } );

   /*************************************
    **** FUNCTION: PostSolderMask
    *************************************/
MakeFunction( PostSolderMask, [],
  {
  ShowWindow( ProSolderMaskWindow );
  SetValue(ComboBox39:Value, ProSoldMask:MaterialType);
  UpdateImage(ComboBox39);
  SetValue(ComboBox38:Value, ProSoldMask:Colour);
  UpdateImage(ComboBox38);
  SetValue(ComboBox41:Value, ProSoldMask:MinThick);
```

```
  UpdateImage(ComboBox41);
  SetValue(ComboBox40:Value, ProSoldMask:MinClear);
  UpdateImage(ComboBox40);
  } );

   /*************************************
    **** FUNCTION: FinProSoldWin
    *************************************/
MakeFunction( FinProSoldWin, [],
  {
  HideWindow( ProSolderMaskWindow );
  SetValue( Button5:BackgroundColor, 192, 192, 192 );
  } );

   /*************************************
    **** FUNCTION: FinProCompWin
    *************************************/
MakeFunction( FinProCompWin, [],
  {
  HideWindow( ProCompWindow );
  SetValue( Button36:BackgroundColor, 192, 192, 192 );
  } );

   /*************************************
    **** FUNCTION: PostComp
    *************************************/
MakeFunction( PostComp, [],
  {
  ShowWindow( ProCompWindow );
  SetValue(ComboBox43:Value, ProCompMark:CompMaterial);
  UpdateImage(ComboBox43);
  SetValue(ComboBox42:Value, ProCompMark:CompColour);
  UpdateImage(ComboBox42);

  } );

   /*************************************
    **** FUNCTION: FinProProdWin
    *************************************/
MakeFunction( FinProProdWin, [],
  {
  HideWindow( ProProdDocWindow );
  SetValue( Button2:BackgroundColor, 192, 192, 192 );
  ShowImage( Button2 );
  } );
```

```
    /*************************************
     **** FUNCTION: PostProdDoc
     *************************************/
MakeFunction( PostProdDoc, [],
  {
  ShowWindow( ProProdDocWindow );
  SetValue(ComboBox44:Value, ProDocument:DrawNo);
  UpdateImage(ComboBox44);
  SetValue(ComboBox45:Value, ProDocument:DrillFileNo);
  UpdateImage(ComboBox45);
  SetValue(ComboBox46:Value, ProDocument:FilmNo);
  UpdateImage(ComboBox46);
  SetValue(ComboBox47:Value, ProDocument:SpecNo);
  UpdateImage(ComboBox47);

  } );

    /*************************************
     **** FUNCTION: SetProdButtonToYellow
     *************************************/
MakeFunction( SetProdButtonToYellow, [],
  {
  SetValue( Button9:BackgroundColor, 255, 255, 0 );
  SetValue( Button12:BackgroundColor, 255, 255, 0 );
  SetValue( Button36:BackgroundColor, 255, 255, 0 );
  SetValue( Button36:BackgroundColor, 255, 255, 0 );
  SetValue( Button10:BackgroundColor, 255, 255, 0 );
  SetValue( Button4:BackgroundColor, 255, 255, 0 );
  SetValue( Button8:BackgroundColor, 255, 255, 0 );
  SetValue( Button7:BackgroundColor, 255, 255, 0 );
  SetValue( Button3:BackgroundColor, 255, 255, 0 );
  SetValue( Button2:BackgroundColor, 255, 255, 0 );
  SetValue( Button5:BackgroundColor, 255, 255, 0 );
  SetValue( Button19:BackgroundColor, 255, 255, 0 );
  } );

    /*************************************
     **** FUNCTION: ReturnProcessName
     *************************************/
MakeFunction( ReturnProcessName, [processname],
{
If processname #= InCu Then PatternCuPlat
  Else {If processname #= InAu Then PatternGoldPlat
    Else {If processname #= InSnPb Then PatternSnPbPlat
      Else { If processname #= InNi Then PatternNiPlat };
       };
     };
} );
```

Appendix 9.1 In-house Process Capability Interface Objects

```
    /*************************************
    **** CLASS: ProCapKnow
    *************************************/
MakeClass( ProCapKnow, Root );

/************** METHOD: Define **************/
MakeMethod( ProCapKnow, Define, [],
  {
  EnumSubClasses( ProCapKnow, dummyname,
    {
    GetSlotList( dummyname, Global:TempList );
    EnumList( Global:TempList, SlotName, ResetValue( dummyname:SlotName ) );
    } );
  PostMessage( "Initialization of Process Capability Knowledge Base is completed." );
  EnumSubClasses( ProCapKnow, dummyname,
    {
    GetSlotList( dummyname, Global:TempList );
    EnumList( Global:TempList, SlotName, AskValue( dummyname:SlotName ) );
    } );
  } );

/************** METHOD: Update **************/
MakeMethod( ProCapKnow, Update, [],
  {
  EnumSubClasses( ProCapKnow, dummyname,
    {
    GetSlotList( dummyname, Global:TempList );
    EnumList( Global:TempList, SlotName, AskValue( dummyname:SlotName ) );
    } );
  } );
```

Appendix 9.2 Customer General Specification Interface Objects

```
    /************************************
     **** CLASS: CustGenSpec
     ************************************/
MakeClass( CustGenSpec, Root );

/************** METHOD: Define  **************/
MakeMethod( CustGenSpec, Define, [],
  {
  InterpretFile( "c:\kapfile\blancust.kap" );
  EnumSubClasses( CustGenSpec, dummyname,
     {
     GetSlotList( dummyname, Global:TempList );
     EnumList( Global:TempList, SlotName, ResetValue( dummyname:SlotName ) );
     } );
  PostMessage( "Initialization of Customer General Specification data base is completed.  Input data!!" );
  EnumSubClasses( CustGenSpec, dummyname,
     {
     GetSlotList( dummyname, Global:TempList );
     EnumList( Global:TempList, SlotName, AskValue( dummyname:SlotName ) );
     } );
  } );

/************** METHOD: Update  **************/
MakeMethod( CustGenSpec, Update, [],
  {
  EnumSubClasses( CustGenSpec, dummyname,
     {
     GetSlotList( dummyname, Global:TempList );
     EnumList( Global:TempList, SlotName, AskValue( dummyname:SlotName ) );
     } );
  } );

/************** METHOD: StoreData  **************/
MakeMethod( CustGenSpec, StoreData, [],
  {
  PostInputForm( "Drive and File Name to Store Customer General Specification Data",
     Global, DriveDir, "Drive (e.g. b:)", Global, FileNameListCusGenSpec,
     "Gen. Spec. File Name (e.g. custg)" );
  BuildUpKnowFileName( Global:FileNameListCusGenSpec );
  StoreKnowledge( CustGenSpec );
  EnumSubClasses( CustAppSurFin, dumname, DeleteClass( dumname ) );
  EnumSubClasses( CustGenSpec, dumname, DeleteClass( dumname ) );
  } );

/************** METHOD: LoadData  **************/
MakeMethod( CustGenSpec, LoadData, [],
  {
  PostInputForm( "Drive and File Name of Customer General Specification",
```

```
  Global, DriveDir, "Drive (e.g. b:)", Global, FileNameListCusGenSpec,
  "Gen. Spec. File Name (e.g. custg)" );
BuildUpKnowFileName( Global:FileNameListCusGenSpec );
InterpretFile( Global:TempFileName );
} );
```

Appendix 9.3 Product Requirements Interface Objects

```
    /************************************
    **** CLASS: CustProdReqt
    ************************************/
MakeClass( CustProdReqt, Root );

/************** METHOD: Define  **************/
MakeMethod( CustProdReqt, Define, [],
  {
  InterpretFile( "c:\kapfile\blanprod.kap" );
  EnumSubClasses( CustProdReqt, dummyname,
    {
    GetSlotList( dummyname, Global:TempList );
    EnumList( Global:TempList, SlotName, ResetValue( dummyname:SlotName ) );
    } );
  PostMessage( "Initialization of Customer Product Requirement data base is completed." );
  SetValue( Global, dumstring, NO );
  PostInputForm( "Read Circuit Feature From File?", Global, dumstring,
    "1 for YES, other for NO" );
  If ( Global:dumstring #= 1 )
   Then SendMessage(CustProdReqt, ReadCircuitFeature);
  SetValue( Global, dumstring, NO );
  PostInputForm( "Read Solder Mask Feature From File?", Global,
    dumstring, "1 for YES, other for NO" );
  If ( Global:dumstring #= 1 )
   Then SendMessage(CustProdReqt, ReadSolderMaskFeature);
  SetProdButtonToYellow( );
  ShowWindow( InPutCustProd );
  } );

/************** METHOD: Update  **************/
MakeMethod( CustProdReqt, Update, [],
  {
  SetValue( Global, dumstring, NO );
  PostInputForm( "Read Circuit Feature From File?", Global, dumstring,
    "1 for YES, other for NO" );
  If ( Global:dumstring #= 1 )
   Then SendMessage(CustProdReqt, ReadCircuitFeature);
  SetValue( Global, dumstring, NO );
  PostInputForm( "Read Solder Mask Feature From File?", Global,
    dumstring, "1 for YES, other for NO" );
  If ( Global:dumstring #= 1 )
   Then SendMessage(CustProdReqt, ReadSolderMaskFeature);
  SetProdButtonToYellow( );
  ShowWindow( InPutCustProd );
  } );
```

```
/************** METHOD: LoadData  **************/
MakeMethod( CustProdReqt, LoadData, [],
  {
  PostInputForm( "Drive and File Name of Product Requirement Data File",
    Global, DriveDir, "Drive (e.g. b:)", Global, FileNameListProd,
    "Gen. Spec. File Name (e.g. p8008)" );
  BuildUpKnowFileName( Global:FileNameListProd );
  OpenReadFile( Global:TempFileName );
  InterpretFile( Global:TempFileName );
  CloseReadFile( );
  } );

/************** METHOD: StoreData  **************/
MakeMethod( CustProdReqt, StoreData, [],
  {
  SendMessage( CustProdReqt, CheckCustGenSpecMethod );
  PostInputForm( "Drive and File Name to Store Product Requirements Data",
    Global, DriveDir, "Drive (e.g. b:)", Global, FileNameListProd,
    "Gen. Spec. File Name (e.g. p8008)" );
  BuildUpKnowFileName( Global:FileNameListProd );
  StoreKnowledge( CustProdReqt );
  EnumSubClasses( CustProdReqt, dumname, DeleteClass( dumname ) );
  EnumSubClasses( CustAppSurFin, dumname, DeleteClass( dumname ) );
  EnumSubClasses( CustGenSpec, dumname, DeleteClass( dumname ) );
  } );

/************** METHOD: CheckCustGenSpecMethod  **************/
MakeMethod( CustProdReqt, CheckCustGenSpecMethod, [],
  {
  ClearList( MessageFile:OutGenSpecMessage );
  EnumSubClasses( CustProdReqt, dummyclassname,
    SendMessage ( dummyclassname, CheckCustGenSpec ));
  SetValue( Global, DriveDir, "b:" );
  SetValue( Global, dumstring, Global:FileNameListProd # .msg );
  OpenWriteFile( Global:DriveDir # Global:dumstring );
  CloseWriteFile( );
  If ( LengthList( MessageFile:OutGenSpecMessage )
       > 0 )
   Then {
     SetValue( Global, DriveDir, "b:" );
     PostInputForm( "Out of Specification Message Found !!! Input Drive Name for the Message File",
       Global, DriveDir, "Drive Name (e.g. b:)", Global, FileNameListProd,
       "Out of Spec. Message File Name (e.g. p8001)" );
     SetValue( Global, dumstring, Global:FileNameListProd
                          # .msg );
     OpenWriteFile( Global:DriveDir # Global:dumstring );
     WriteLine( "/* Prod. Reqt. Out of Cust. Gen. Spec. */" );
     WriteLine( "/* Type of Reqt. --- Prod.--Spec. (Value)*/" );
     EnumList( MessageFile:OutGenSpecMessage, message, WriteLine( message ) );
     CloseWriteFile( );
     PostMessage( "Out of Specification Message Stored in File "
```

```
                    # Global:DriveDir # Global:dumstring );
        }
      Else {PostMessage( "Product Requirements Within Customer General Specification.")};
   } );

/************** METHOD: ReadSolderMaskFeature  **************/
MakeMethod( CustProdReqt, ReadSolderMaskFeature, [],
{
   SetValue( Global, DriveDir, "b:" );
   SetValue( Global, dumstring, solmas.fil );
   PostInputForm( "Input Drive and File Name of the Solder Mask Feature File",
      Global, DriveDir, "Drive Name (e.g. b:)", Global, dumstring,
      "File Name (e.g. sold.fil)" );
   OpenReadFile( Global:DriveDir # Global:dumstring );
   SetValue( Global:SoldSpec1, ReadLine( ) );
   SetValue( Global:SoldSpec1No, ReadLine( ) );
   SetValue( Global:SoldSpec2, ReadLine( ) );
   SetValue( Global:SoldSpec2No, ReadLine( ) );
   SetValue( Global:SoldSpec3, ReadLine( ) );
   SetValue( Global:SoldSpec3No, ReadLine( ) );
   CloseReadFile( );
   If ( Global:SoldSpec1No == 0 )
    Then SetValue( ProSoldMask:MinClear, Global:SoldSpec1 );
   If ( Global:SoldSpec2No == 0 )
    Then SetValue( ProSoldMask:MinClear, Global:SoldSpec2 );
   If ( Global:SoldSpec3No == 0 )
    Then SetValue( ProSoldMask:MinClear, Global:SoldSpec3 );
} );

/************** METHOD: ReadCircuitFeature  **************/
MakeMethod( CustProdReqt, ReadCircuitFeature, [],
 {
   SetValue( Global, DriveDir, "b:" );
   SetValue( Global, dumstring, cirfea.fil );
   PostInputForm( "Input Drive and File Name of the Circuit Feature File",
      Global, DriveDir, "Drive Name (e.g. b:)", Global, dumstring,
      "File Name (e.g. cirfea.fil)" );
   OpenReadFile( Global:DriveDir # Global:dumstring );
   SetValue( Global:CircuitArea, ReadLine( ) );
   SetValue( Global:CirSpec1, ReadLine( ) );
   SetValue( Global:CirSpec1No, ReadLine( ) );
   SetValue( Global:CirSpec2, ReadLine( ) );
   SetValue( Global:CirSpec2No, ReadLine( ) );
   SetValue( Global:CirSpec3, ReadLine( ) );
   SetValue( Global:CirSpec3No, ReadLine( ) );
   ReadLine( );
   CloseReadFile( );
   SetValue( ProFeatCircuit:CircuitArea, Global:CircuitArea );
   If ( Global:CirSpec1No == 0 )
    Then SetValue( ProFeatCircuit:SpaceWidth, Global:CirSpec1 );
   If ( Global:CirSpec2No == 0 )
```

```
    Then SetValue( ProFeatCircuit:SpaceWidth, Global:CirSpec2 );
  If ( Global:CirSpec3No == 0 )
    Then SetValue( ProFeatCircuit:SpaceWidth, Global:CirSpec3 );
} );
```

Appendix 9.4 "ProNameSeqNo" Objects

```
/*************************************
 **** CLASS: ProNameSeqNo
 *************************************/
MakeClass( ProNameSeqNo, Root );

/*************************************
 **** CLASS: ProcessName
 *************************************/
MakeClass( ProcessName, ProNameSeqNo );
MakeSlot( ProcessName:Name );
SetSlotOption( ProcessName:Name, MULTIPLE );
SetValue( ProcessName:Name, NCDrilling, SlotPunching, HolePlugging, ElectrolessCuPlat, PanelCuPlat,
PlugHoleRemove, ImTranCirDryFilm, ImTranCirWetFilm, ImTranCirSilkScreen, PatternCuPlat,
PatternNiPlat, PatternAuPlat, PatternSnPbPlat, CirImageStrip, CuEtching, AuFinger, SnPbReflow,
SnPbStripping, ImTranSMDryFilm, ImTranSMWetFilm, ImTranSMSilkScreen, SolderLevelling,
CompMarkPrint, 2ndDrillOrHolePunch, OutLineRout, OutLineBlank, Slotting, Blevelling, VSlotCutting,
OpenShortTesting, FinalInspection, Packing );

/*************************************
 **** CLASS: MfgSeqNo
 *************************************/
MakeClass( MfgSeqNo, ProNameSeqNo );
MakeSlot( MfgSeqNo:SeqNo );
SetSlotOption( MfgSeqNo:SeqNo, MULTIPLE );
SetSlotOption( MfgSeqNo:SeqNo, VALUE_TYPE, NUMBER );
SetValue( MfgSeqNo:SeqNo, 10, 20, 30, 40, 50, 60, 70, 70, 70, 80, 90, 90, 90, 100, 110, 120, 130, 130,
140, 140, 140, 150, 160, 170, 180, 180, 190, 200, 0, 220, 230, 240 );
```

Appendix 9.5 "SuggProInHou" Objects

```
    /*************************************
     **** CLASS: SuggProInHou
     *************************************/
MakeClass( SuggProInHou, Root );

/************** METHOD: ProcessSelection  **************/
MakeMethod( SuggProInHou, ProcessSelection, [],
   {
   ClearList( SuggProInHou:ProcessName );
   ClearList(MessageFile:ProCanNotMfgInHouse);
    HideImage( MultipleListBox2 );
   HideImage( Button25 );
   HideImage( Button24 );
   ForAll [ process|SuggProInHou ]
        SendMessage( process, ProcessSelection );
   If ( LengthList( MessageFile:ProCanNotMfgInHouse )
         > 0 )
     Then {
         SetValue( Global, dumstring, Global:FileNameListProd#".msg");
         OpenWriteFile( Global:DriveDir # Global:dumstring, APPEND );
         WriteLine( "/* Process(es) Cannot Perform In-house*/" );
         WriteLine( "/* Process Name(s) -- Problem */" );
         EnumList (MessageFile:ProCanNotMfgInHouse, message, WriteLine(message));
         CloseWriteFile();
         PostMessage("Process(es) Cannot Perform In-house Message Stored in File
"#Global:DriveDir#Global:dumstring);
      };
   ShowImage( MultipleListBox2 );
   ShowImage( Button25 );
   } );

/************** METHOD: WriteProdReqt  **************/
MakeMethod( SuggProInHou, WriteProdReqt, [],
   {
   WriteLine( "/* File to Store Product Requirements and Operation Instruction */" );
   WriteLine( "/* This File Name : ", Global:dumstring, "*/" );
   WriteLine( "Date and Time Process Recipe Generated :",
      Date( ), Time( ) );
   WriteLine( " " );
   WriteLine( "/* ----- PRODUCT REQUIREMENTS ---- */" );
   WriteLine( "/* ----- Product Details ------ */" );
   WriteLine( "Customer Name :", ReqDocInfo:CustName );
   WriteLine( "Customer Circuit Part Number :", ReqDocInfo:PNCust );
   WriteLine( "Name of Circuit :", ReqDocInfo:ProdName );
   WriteLine( "Customer Purchase Order Number :", ReqDocInfo:PurOrdNo );
   WriteLine( "In House Part Number :", ReqDocInfo:PNInHouse );
   WriteLine( " " );
   WriteLine( "/* ------------ Production Quantity and Date Required ---------------- */" );
```

```
WriteLine( "Batch Size of this Production Batch (unit) :", ProProdDetails:BatchSize );
WriteLine( "Total Expected Quantity (unit) :", ProProdDetails:ExpQty );
WriteLine( "Time Available for Production (days) :", ProProdDetails:WkDateAvail );
WriteLine( " " );
WriteLine( "/* --------------Product Documentation Details----------------*/" );
WriteLine( "Drawing Number :", ProDocument:DrawNo );
WriteLine( "Drilling File Number :", ProDocument:DrillFileNo );
WriteLine( "Film Number :", ProDocument:FilmNo );
WriteLine( "Specification Number :", ProDocument:SpecNo );
WriteLine( " " );
WriteLine( "/* -------------- Laminate Type ------------------------ */" );
WriteLine( "Type of Laminate Material :", ProLamChar:LaminateType );
WriteLine( "Thickness of Laminate (mm) :", ProLamChar:LamThickness );
WriteLine( "Copper Foil Thickness :", ProLamChar:CuFoilThickness );
WriteLine( "Side of Copper :", ProLamChar:CuFoilSide );
WriteLine( "Colour of Laminate :", ProLamChar:Colour );
WriteLine( " " );
WriteLine( "/* --------------- Panel and Unit Dimension Details -----------------*/" );
WriteLine( "Circuit Unit Width (mm) :", ProPanelDimen:UnitWidth);
WriteLine( "Circuit Unit Length (mm) :", ProPanelDimen:UnitLength);
WriteLine( "Panel Width (mm) :", ProPanelDimen:Width );
WriteLine( "Panel Length (mm) :", ProPanelDimen:Length );
WriteLine( "No. of Unit Per Panel :", ProPanelDimen:UnitPerPanel);
WriteLine( "Outline Dimenstion Deviation (0.01mm) :", ProPanelDimen:DimenDev );
WriteLine( "Estimated Die Cost:", ProPanelDimen:EstDieCost );
WriteLine( " " );
WriteLine( "/* --------------- Holes Characteristic -----------------*/" );
WriteLine( "Allowed Hole Diameter Deviation (0.01mm) :", ProHoleSpec:DiaDev );
WriteLine( "Allowed Hole Location Deviation (0.01mm) :", ProHoleSpec:LocDev );
WriteLine( "Maximum Hole Size (mm) :", ProHoleSpec:MaxHoleSize );
WriteLine( "Minimum Hole Size (mm) :", ProHoleSpec:MinHoleSize );
WriteLine( " " );
WriteLine( "/* ---------------- Circuit Feature ----------------------*/" );
WriteLine( "Minimum Annual Ring (0.01mm) :", ProFeatCircuit:AnnularRing );
WriteLine( "Total Unit Circuit Area (sq.in.) :", ProFeatCircuit:CircuitArea );
WriteLine( "Minimum Circuit Thickness (0.01mm) :", ProFeatCircuit:CirThickness );
WriteLine( "Minimum Circuit Width (0.01mm) :", ProFeatCircuit:CirWidth );
WriteLine( "Minimum Circuit Spacing (0.01mm) :", ProFeatCircuit:SpaceWidth );
WriteLine( " " );
WriteLine( "/* --------------- Plating Requirement -----------------*/" );
WriteLine( "Minimum Plated Copper Thickness (0.001in) :", ProPlatingReq:CuThickMin );
WriteLine( "Type of Surface Finishing Rquired:", ProPlatingReq:SurfaceFinType );
WriteLine( "Minimum Plated Surface Finishing Thickness (0.001in) :",
   ProPlatingReq:SurfaceFinThick );
WriteLine( " " );
WriteLine( "/* -------------- Gold Finger Requirements ---------------- */" );
WriteLine( "Gold Thickness (0.001in) :", ProAuFinger:AuThick );
WriteLine( "Height of the Gold Finger (mm) :", ProAuFinger:Height );
WriteLine( "Ni Under Gold Thickness (0.001in) :", ProAuFinger:NiThick );
WriteLine( "Gold Finger Unit Area :", ProAuFinger:PlatAreaPerUnit );
WriteLine( "No. of Unit Per Edge :", ProAuFinger:UnitPerEdge);
```

```
  WriteLine( " " );
  WriteLine( "/* ---------------- Solder Masking Requirements --------------- */" );
  WriteLine( "Type of Solder Masking Material:", ProSoldMask:MaterialType );
  WriteLine( "Solder Masking Colour:", ProSoldMask:Colour );
  WriteLine( "Minimum Thickness (0.01mm) :", ProSoldMask:MinThick );
  WriteLine( "Minimum Allowed Ring Clearance (0.01mm) :", ProSoldMask:MinClear );
  WriteLine( " " );
  WriteLine( "/* ----------------- Component Marking Requirements ---------- */" );
  WriteLine( "Component Marking Material Allowed :", ProCompMark:CompMaterial );
  WriteLine( "Marking Colour :", ProCompMark:CompColour );
  WriteLine( " " );
  WriteLine( " " );
  } );

/************** METHOD: WriteProcessSeq  **************/
MakeMethod( SuggProInHou, WriteProcessSeq, [],
  {
  WriteLine( "/* Suggested Process in Mfg. Sequence*/" );
  Global:dumno = 1;
  EnumList( SuggProInHou:ProcessNameInSeq, dummyname,
    {
    WriteLine( Global:dumno, ., "  ", dummyname );
    Global:dumno = ( Global:dumno + 1 );
    } );
  If ( LengthList( MessageFile:OutGenSpecMessage )
        > 0 )
   Then {
      WriteLine( " " );
      WriteLine( "/* Prod. Reqt. Out of Cust. Gen. Spec. */" );
      WriteLine( "/* Type of Reqt. --- Prod.--Spec. (Value)*/" );
      EnumList( MessageFile:OutGenSpecMessage, message, WriteLine( message ) );
      };
  If ( LengthList( MessageFile:ProCanNotMfgInHouse )
        > 0 )
   Then {
      WriteLine( " ");
      WriteLine( "/* Process(es) Cannot Perform In-house*/" );
      WriteLine( "/* Process Name(s) -- Problem */" );
      EnumList (MessageFile:ProCanNotMfgInHouse, message, WriteLine(message));
    };
  } );

/************** METHOD: WriteOperationInst  **************/
MakeMethod( SuggProInHou, WriteOperationInst, [],
  {
  WriteLine( " " );
  WriteLine( " " );
  WriteLine( "/* ---------- OPERATION INSTRUCTION --------- */" );
  If Member?( SuggProInHou:ProcessNameInSeq, NCDrilling )
   Then {
      WriteLine( "/* --- NC Drilling Process Operation Instruction --- */" );
```

```
    WriteLine( "Drilling Requirement:-" );
    WriteLine( "    Panel Width:", ProPanelDimen:Width, " mm" );
    WriteLine( "    Panel Length:", ProPanelDimen:Length, " mm" );
    WriteLine( "    Allowed Hole Location Deviation:", ProHoleSpec:LocDev );
    WriteLine( "    Allowed Hole Size Deviation:", ProHoleSpec:DiaDev, " in 0.01 mm" );
    WriteLine( "    Minimum Hole Size:", ProHoleSpec:MinHoleSize, " mm" );
    WriteLine( "    Maximum Hole Size:", ProHoleSpec:MaxHoleSize, " mm" );
    WriteLine( "NC Drilling Machine to use:- " );
    EnumList( NCDrilling:MCListing, dummyname,
      WriteLine( "    ", dummyname ) );
    };
If Member?( SuggProInHou:ProcessNameInSeq, PanelCuPlat )
 Then {
    WriteLine( " " );
    WriteLine( "/* --- Panel Copper Plating Operation Instruction --- */" );
    WriteLine( "Requirement:- " );
    If ( ProLamChar:CuFoilSide #= Double)
       Then (WriteLine( "    Panel Surface Area:-  " # ProPanelDimen:Width
              * ProPanelDimen:Length * 2 # " sq. mm OR "
              # Ceil( ProPanelDimen:Width * ProPanelDimen:Length
                     / 25.4 / 25.4 / 144 * 1000 ) / 1000 * 2 # " sq. ft " ))
       Else  (WriteLine( "    Panel Surface Area:-  " # ProPanelDimen:Width
              * ProPanelDimen:Length # " sq. mm OR "
              # Ceil( ProPanelDimen:Width * ProPanelDimen:Length
                     / 25.4 / 25.4 / 144 * 1000 ) / 1000 # " sq. ft " ));
    WriteLine( "    Copper Thickness is at 0.0005 in." );
    WriteLine( "Settings :-" );
    WriteLine( "    Current Setting:- " # Ceil( RecPanPlatCu:Current * 10 )  / 10 # " A." );
    WriteLine( "    Plating Time:- " # RecPanPlatCu:PlatingTime # " mins." );
    };
If Member?( SuggProInHou:ProcessNameInSeq, ImTranCirSilkScreen )
 Then {
    WriteLine( " " );
    WriteLine( "/* --- Image Transfer Circuit Silk Screen Operation Instruction --- */" );
    WriteLine( "Silk Screen Machine to Use:- " );
    EnumList( ITCirSilkScreen:MCListing, dummyname,
      WriteLine( "    ", dummyname ) );
    };
If Member?( SuggProInHou:ProcessNameInSeq, PatternCuPlat )
 Then {
    WriteLine( " " );
    WriteLine( "/* --- Pattern Copper Plating Operation Instruction ---*/" );
    WriteLine( "Requirement :-" );
    WriteLine( "    Plating Area:- ", Ceil( RecPatPlatCu:Current / InPanelPlat:CurrentDensity
         * 1000 ) / 1000, " sq. ft." );
    WriteLine( "    Minimum Plated Copper Thickness:- ", ProPlatingReq:CuThickMin,
      " in 0.001 in. unit.(0.0005 in. panel plated)" );
    WriteLine( "Settings:-" );
    WriteLine( "    Current Setting:- " # Ceil( RecPatPlatCu:Current * 100 ) / 100 # " A." );
    WriteLine( "    Plating Time:- " # RecPatPlatCu:PlatingTime # " mins." );
    };
```

```
If Member?( SuggProInHou:ProcessNameInSeq, PatternNiPlat )
 Then {
    WriteLine( " " );
    WriteLine( "/* --- Pattern Ni Plating Operation Instruction ---*/" );
    WriteLine( "Requirement:- " );
    WriteLine( "   Plating Area:- ", Ceil( RecPatPlatCu:Current / InPanelPlat:CurrentDensity
          * 1000 ) / 1000, " sq. ft." );
    WriteLine( "   Minimum Surface Finishing Thickness:- ",
      ProPlatingReq:SurfaceFinThick, " in 0.001 in. unit," );
    WriteLine( "Settings:-" );
    WriteLine( "   Current Setting:- " # Ceil( RecPatPlatNi:Current * 100 ) / 100 # " A." );
    WriteLine( "   Plating Time:- " # RecPatPlatNi:PlatingTime # " mins." );
    };
If Member?( SuggProInHou:ProcessNameInSeq, PatternAuPlat )
 Then {
    WriteLine( " " );
    WriteLine( "/* --- Pattern Gold Plating Operation Instruction ---*/" );
    WriteLine( "Requirement:- " );
    WriteLine( "   Plating Area:- ", Ceil( RecPatPlatCu:Current / InPanelPlat:CurrentDensity
          * 1000 ) / 1000, " sq. ft." );
    WriteLine( "   Minimum Surface Finishing Thickness:- ",
      ProPlatingReq:SurfaceFinThick, " in 0.001 in. unit." );
    WriteLine( "Settings:-" );
    WriteLine( "   Current Setting:- " # Ceil( RecPatPlatAu:Current * 100 )  / 100 # " A." );
    WriteLine( "   Plating Time:- " # RecPatPlatAu:PlatingTime # " mins." );
    };
If Member?( SuggProInHou:ProcessNameInSeq, PatternSnPbPlat )
 Then {
    WriteLine( " " );
    WriteLine( "/* --- Pattern SnPb Plating Operation Instruction ---*/" );
    WriteLine( "Requirement:- " );
    WriteLine( "   Plating Area:- ", Ceil( RecPatPlatCu:Current / InPanelPlat:CurrentDensity
          * 1000 ) / 1000, " sq. ft." );
    WriteLine( "   Minimum Surface Finishing Thickness:- ",
      ProPlatingReq:SurfaceFinThick, " in 0.001 in. unit." );
    WriteLine( "Settings:-" );
    WriteLine( "   Current Setting:- " # Ceil( RecPatPlatSnPb:Current * 100 ) / 100 # " A." );
    WriteLine( "   Plating Time:- " # RecPatPlatSnPb:PlatingTime # " mins." );
    };
If Member?( SuggProInHou:ProcessNameInSeq, AuFinger )
 Then {
    WriteLine( " " );
    WriteLine( "/* --- Gold Finger Plating Operation Instruction ---*/" );
    WriteLine( "Requirement:- " );
    WriteLine( "   Min. Ni under Gold Thickness:- ", ProAuFinger:NiThick, " in 0.001 in. unit." );
    WriteLine( "   Min. Gold Thickness:- ", ProAuFinger:AuThick,  " in 0.001 in. unit" );
    WriteLine( "Settings:-" );
    WriteLine( "   Ni Current Setting:- " # Ceil( RecAuFinger:NiCurrent * 100 ) / 100 # " A." );
    WriteLine( "   Ni Plating Time:- ", RecAuFinger:NiPlatingTime, " mins." );
    WriteLine( "   Gold Current Setting:- ",
      Ceil( RecAuFinger:AuCurrent * 100 ) / 100, " A." );
```

```
      WriteLine( "    Gold Plating Time:- ", RecAuFinger:AuPlatingTime, " mins." );
      };
  If Member?( SuggProInHou:ProcessNameInSeq, ImTranSMSilkScreen )
   Then {
      WriteLine( " " );
      WriteLine( "/* --- Image Transfer Solder Masking Operation Instruction ---*/" );
      WriteLine( "Requirement:-" );
      WriteLine( "    Type of Solder Masking Material:- ", ProSoldMask:MaterialType );
      WriteLine( "    Minimum Thickness:- ", ProSoldMask:MinThick );
      WriteLine( "    Minimum Allowed Ring Clearance:- ", ProSoldMask:MinClear, " in 0.01 mm." );
      WriteLine( "    Solder Masking Material Colour:- ", ProSoldMask:Colour );
      WriteLine( "Silk Screen Machine to Use:- " );
      EnumList( ITSMSilkScreen:MCListing, dummyname,
         WriteLine( "    ", dummyname ) );
      };
  If Member?( SuggProInHou:ProcessNameInSeq, CompMarkPrint )
   Then {
      WriteLine( " " );
      WriteLine( "/* --- Image Transfer Comp. Marking Operation Instruction ---*/" );
      WriteLine( "Requirement:- " );
      WriteLine( "    Component Marking Material Allowed:- ",
         ProCompMark:CompMaterial );
      WriteLine( "    Marking Colour:- ", ProCompMark:CompColour );
      WriteLine( "Silk Screen Machine to Use:- " );
      EnumList( ITCompMark:MCListing, dummyname,
         WriteLine( "    ", dummyname ) );
      };
  If Member?( SuggProInHou:ProcessNameInSeq, OutLineBlank )
   Then {
      WriteLine( " " );
      WriteLine( "/* --- Blanking Machine Operation Instruction ---*/" );
      WriteLine( "Blanking Machine to Use:- " );
      EnumList( BlankingMC:MCListing, dummyname,
         WriteLine( "    ", dummyname ) );
      };
  } );

/************** METHOD: OpenFileToWriteInst  **************/
MakeMethod( SuggProInHou, OpenFileToWriteInst, [],
  {
  SetValue( Global, DriveDir, "b:" );
  SetValue( Global, dumstring, Global:FileNameListProd#".ins" );
  PostMessage( "Process Recipe Successfully Generated!!" );
  PostInputForm( "Input Drive and File Name of the Operation Instruction File",
     Global, DriveDir, "Drive Name (e.g. b:)", Global, dumstring,
     "File Name (e.g. p8001.ins)" );
  OpenWriteFile( Global:DriveDir # Global:dumstring );
  } );
```

```
/************** METHOD: CloseInstWriteFile  **************/
MakeMethod( SuggProInHou, CloseInstWriteFile, [],
  {
  CloseWriteFile( );
  PostMessage( "Operation Instruction Sheet Stored in File ",Global:DriveDir,Global:dumstring );
  } );

/************** METHOD: SequenceProcess  **************/
MakeMethod( SuggProInHou, SequenceProcess, [],
{
  ResetValue( SuggProInHou:ProcessNameSeqNo );
  EnumList( SuggProInHou:ProcessName, name, AppendToList( SuggProInHou:ProcessNameSeqNo,
     { GetNthElem( MfgSeqNo:SeqNo, GetElemPos( ProcessName:Name, name ) );   }
     ) );
  ClearList( SuggProInHou:ProcessNameInSeq );
  For numb From 0 To 300 By 10
       Do {
      Global:dumno = GetElemPos( SuggProInHou:ProcessNameSeqNo,  numb );
      If ( Global:dumno > 0 )
        Then {
           AppendToList( SuggProInHou:ProcessNameInSeq,
             GetNthElem( SuggProInHou:ProcessName, Global:dumno ) );
           };
      };
} );

/************** METHOD: SaveProNameSeqInFile  **************/
MakeMethod( SuggProInHou, SaveProNameSeqInFile, [],
{
  SetValue( Global, DriveDir, "b:" );
  SetValue( Global, dumstring, proname.fil );
  PostInputForm( "Input Drive and File Name of the Process Name File",
     Global, DriveDir, "Drive Name (e.g. b:)", Global, dumstring,
     "File Name (e.g. operinst.fil)" );
  OpenWriteFile( Global:DriveDir # Global:dumstring );
  WriteLine( "/* File Store Up the Suggested Process in Mfg. Sequence*/" );
  WriteLine( "/* File Name ", Global:dumstring, " */" );
  WriteLine( "/* Product Name ", ReqDocInfo:ProdName, " */" );
  WriteLine( "/* Customer Name ", ReqDocInfo:CustName, " */" );
  EnumList( SuggProInHou:ProcessNameInSeq, dummyname,
     WriteLine( dummyname ) );
  CloseWriteFile( );
} );
MakeSlot( SuggProInHou:ProcessName );
SetSlotOption( SuggProInHou:ProcessName, MULTIPLE );
SetSlotOption( SuggProInHou:ProcessName, ALLOWABLE_VALUES, NCDrilling, SlotPunching,
HolePlugging, ElectrolessCuPlat, PanelCuPlat, PlugHoleRemove, ImTranCirDryFilm,
ImTranCirWetFilm, ImTranCirSilkScreen, PatternCuPlat, PatternNiPlat, PatternAuPlat, PatternSnPbPlat,
CirImageStrip, CuEtching, AuFinger, SnPbReflow, SnPbStripping, ImTranSMDryFilm,
ImTranSMWetFilm, ImTranSMSilkScreen, SolderLevelling, CompMarkPrint, 2ndDrillOrHolePunch,
```

```
OutLineRout, OutLineBlank, Slotting, Blevelling, VSlotCutting, OpenShortTesting, FinalInspection,
Packing );
SetValue( SuggProInHou:ProcessName, NCDrilling, ElectrolessCuPlat, PatternCuPlat, PanelCuPlat,
ImTranCirSilkScreen, PatternSnPbPlat, CuEtching, CirImageStrip, AuFinger, SnPbReflow,
ImTranSMSilkScreen, CompMarkPrint, OutLineBlank );
SetSlotOption( SuggProInHou:ProcessName, PROMPT, "Find feasible process." );
SetSlotOption( SuggProInHou:ProcessName, IMAGE, MultipleListBox1, CheckBoxGroup1,
MultipleListBox2, MultipleListBox3, MultipleListBox4 );
MakeSlot( SuggProInHou:ProcessNameSeqNo );
SetSlotOption( SuggProInHou:ProcessNameSeqNo, MULTIPLE );
SetValue( SuggProInHou:ProcessNameSeqNo, 10, 40, 80, 50, 70, 90, 110, 100, 120, 130, 140, 160, 180 );
SetSlotOption( SuggProInHou:ProcessNameSeqNo, IMAGE, MultipleListBox1 );
MakeSlot( SuggProInHou:ProcessNameInSeq );
SetSlotOption( SuggProInHou:ProcessNameInSeq, MULTIPLE );
SetValue( SuggProInHou:ProcessNameInSeq, NCDrilling, ElectrolessCuPlat, PanelCuPlat,
ImTranCirSilkScreen, PatternCuPlat, PatternSnPbPlat, CirImageStrip, CuEtching, AuFinger,
SnPbReflow, ImTranSMSilkScreen, CompMarkPrint, OutLineBlank );
SetSlotOption( SuggProInHou:ProcessNameInSeq, IMAGE, MultipleListBox1 );

    /*************************************
     **** INSTANCE: NCDrill
     *************************************/
MakeInstance( NCDrill, SuggProInHou );

/************** METHOD: ProcessSelection  **************/
MakeMethod( NCDrill, ProcessSelection, [],
  {
  Assert( ProHoleSpec, LocDev );
  Assert( ProPlatingReq, CuThickMin );
  ForwardChain( NULL, PSNCDrill );
  } );

    /*************************************
     **** INSTANCE: ElectlessCopper
     *************************************/
MakeInstance( ElectlessCopper, SuggProInHou );

/************** METHOD: ProcessSelection  **************/
MakeMethod( ElectlessCopper, ProcessSelection, [],
  {
  Assert( ProPlatingReq, CuThickMin );
  ForwardChain( NULL, PSElectlessPlating );
  } );

    /*************************************
     **** INSTANCE: PatCuPlat
     *************************************/
MakeInstance( PatCuPlat, SuggProInHou );

/************** METHOD: ProcessSelection  **************/
MakeMethod( PatCuPlat, ProcessSelection, [],
```

```
  {
  Assert( ProPlatingReq, CuThickMin );
  ForwardChain( NULL, PSPanPatCuPlat );
  } );

    /*************************************
     **** INSTANCE: ImTranCircuit
     *************************************/
MakeInstance( ImTranCircuit, SuggProInHou );

/************** METHOD: ProcessSelection  **************/
MakeMethod( ImTranCircuit, ProcessSelection, [],
  {
  Assert( ProFeatCircuit, CirWidth );
  ForwardChain( NULL, PSImageTransSSCir, PSImageTransWFCir, PSImageTransDFCir );
  } );

    /*************************************
     **** INSTANCE: PatMetalPlat
     *************************************/
MakeInstance( PatMetalPlat, SuggProInHou );

/************** METHOD: ProcessSelection  **************/
MakeMethod( PatMetalPlat, ProcessSelection, [],
  {
  Assert( ProPlatingReq, SurfaceFinType );
  ForwardChain( NULL, PSPatternNiPlat, PSPatternSnPbPlat, PSPatternAuPlat,
    PSPatternBareCuPlat );
  } );

    /*************************************
     **** INSTANCE: ImStripCuEtch
     *************************************/
MakeInstance( ImStripCuEtch, SuggProInHou );

/************** METHOD: ProcessSelection  **************/
MakeMethod( ImStripCuEtch, ProcessSelection, [],
  {
  Assert( SuggProInHou, ProcessName );
  ForwardChain( NULL, PSImStripCuEtching );
  } );

    /*************************************
     **** INSTANCE: AuFinger
     *************************************/
MakeInstance( AuFinger, SuggProInHou );

/************** METHOD: ProcessSelection  **************/
MakeMethod( AuFinger, ProcessSelection, [],
  {
  Assert( ProAuFinger, AuThick );
```

```
  ForwardChain( NULL, PSAuFinger );
  } );

    /*************************************
     **** INSTANCE: SnPbReflow
     *************************************/
MakeInstance( SnPbReflow, SuggProInHou );

/************** METHOD: ProcessSelection  **************/
MakeMethod( SnPbReflow, ProcessSelection, [],
  {
  Assert( SuggProInHou, ProcessName );
  ForwardChain( NULL, PSSnPbReflow );
  } );

    /*************************************
     **** INSTANCE: SolderMasking
     *************************************/
MakeInstance( SolderMasking, SuggProInHou );

/************** METHOD: ProcessSelection  **************/
MakeMethod( SolderMasking, ProcessSelection, [],
  {
  Assert( ProSoldMask, MinClear );
  ForwardChain( NULL, PSImageTransSSSM, PSImageTransWFSM, PSImageTransDFSM );
  } );

    /*************************************
     **** INSTANCE: CompMarking
     *************************************/
MakeInstance( CompMarking, SuggProInHou );

/************** METHOD: ProcessSelection  **************/
MakeMethod( CompMarking, ProcessSelection, [],
  {
  Assert( ProCompMark, CompMaterial );
  ForwardChain( NULL, PSCompMarking );
  } );

    /*************************************
     **** INSTANCE: Blanking
     *************************************/
MakeInstance( Blanking, SuggProInHou );

/************** METHOD: ProcessSelection  **************/
MakeMethod( Blanking, ProcessSelection, [],
  {
  Assert( ProPanelDimen, DimenDev );
  Assert( ProProdDetails, ExpQty );
  ForwardChain( NULL, PSBlank );
  } );
```

```
    /************************************
     **** INSTANCE: Routing
     ************************************/
MakeInstance( Routing, SuggProInHou );

/************** METHOD: ProcessSelection  **************/
MakeMethod( Routing, ProcessSelection, [],
  {
  ForwardChain( [ NOASSERT ], NULL, PSRouting );
  } );

    /************************************
     **** INSTANCE: HolePunch
     ************************************/
MakeInstance( HolePunch, SuggProInHou );

/************** METHOD: ProcessSelection  **************/
MakeMethod( HolePunch, ProcessSelection, [],
{
  Assert( ProPlatingReq, CuThickMin);
  ForwardChain (NULL, PSHolePunch);
} );
```

Appendix 9.6 "MCChar" Objects

```
    /************************************
     **** CLASS: MCChar
     ************************************/
MakeClass( MCChar, Root );
MakeSlot( MCChar:MachineName );

    /************************************
     **** CLASS: DrillMC
     ************************************/
MakeClass( DrillMC, MCChar );

/************** METHOD: SelectMachine  **************/
MakeMethod( DrillMC, SelectMachine, [],
  {
  ClearList( NCDrilling:MCListing );
  ForAll [ mclDrillMC ]
      {
      If ( mc:MachinedAcc < ProHoleSpec:LocDev And mc:NCLength
            > ProPanelDimen:Length And mc:NCWidth > ProPanelDimen:Width
            And mc:MaxHoleSize > ProHoleSpec:MaxHoleSize
            And mc:MinHoleSize < ProHoleSpec:MinHoleSize )
       Then AppendToList( NCDrilling:MCListing, mc:MachineName );
      };
  If ( LengthList( NCDrilling:MCListing ) == 0 )
    Then {
      AppendToList( NCDrilling:MCListing, "*****NOSuitableNCDrillingMC" );
      AppendToList( MessageFile:MCSelectionNoMessage, NCDrilling );
      PostMessage( "NO Suitable NC Drilling Machine In-house for this Product." );
      };
  } );
MakeSlot( DrillMC:MaxRPM );
MakeSlot( DrillMC:NCWidth );
SetSlotOption( DrillMC:NCWidth, VALUE_TYPE, NUMBER );
MakeSlot( DrillMC:NCLength );
SetSlotOption( DrillMC:NCLength, VALUE_TYPE, NUMBER );
MakeSlot( DrillMC:MachinedAcc );
MakeSlot( DrillMC:MinHoleSize );
SetSlotOption( DrillMC:MinHoleSize, VALUE_TYPE, NUMBER );
MakeSlot( DrillMC:MaxHoleSize );
SetSlotOption( DrillMC:MaxHoleSize, VALUE_TYPE, NUMBER );

    /************************************
     **** CLASS: BlankMC
     ************************************/
MakeClass( BlankMC, MCChar );
```

```
/************** METHOD: SelectMachine  **************/
MakeMethod( BlankMC, SelectMachine, [],
  {
  ClearList( BlankingMC:MCListing );
  ForAll [ mclBlankMC ]
      {
      If ( mc:Accuracy < ProPanelDimen:DimenDev And mc:Length
            > ProPanelDimen:Length And mc:Width > ProPanelDimen:Width )
       Then AppendToList( BlankingMC:MCListing, mc:MachineName );
      };
  If ( LengthList( BlankingMC:MCListing ) == 0 )
   Then {
      AppendToList( BlankingMC:MCListing, "*****NOSuitableBlankingMC" );
      AppendToList( MessageFile:MCSelectionNoMessage, Blanking );
      PostMessage( "NO Suitable Blanking Machine Find In-house for this Product." );
      };
  } );
MakeSlot( BlankMC:Width );
SetSlotOption( BlankMC:Width, VALUE_TYPE, NUMBER );
MakeSlot( BlankMC:Length );
SetSlotOption( BlankMC:Length, VALUE_TYPE, NUMBER );
MakeSlot( BlankMC:Accuracy );
SetSlotOption( BlankMC:Accuracy, VALUE_TYPE, NUMBER );

    /************************************
     **** CLASS: SilkScreenMC
     ************************************/
MakeClass( SilkScreenMC, MCChar );

/************** METHOD: SelectMachine  **************/
MakeMethod( SilkScreenMC, SelectMachine, [ACC MCLIST ],
  {
  ClearList( MCLIST:MCListing );
  ForAll [ mclSilkScreenMC ]
      {
      If ( mc:Resolution < ACC And mc:Length > ProPanelDimen:Length
            And mc:Width > ProPanelDimen:Width )
       Then AppendToList( MCLIST:MCListing, mc:MachineName );
      };
  If ( LengthList( MCLIST:MCListing ) == 0 )
   Then {
      AppendToList( MCLIST:MCListing, "*****NOSuitableSilkScreenMC" );
      AppendToList( MessageFile:MCSelectionNoMessage, MCLIST );
      PostMessage( MCLIST # " NO Suitable Silk Screen Machine Find In-house for this Process." );
      };
  } );
MakeSlot( SilkScreenMC:Width );
SetSlotOption( SilkScreenMC:Width, VALUE_TYPE, NUMBER );
MakeSlot( SilkScreenMC:Length );
SetSlotOption( SilkScreenMC:Length, VALUE_TYPE, NUMBER );
```

```
MakeSlot( SilkScreenMC:Resolution );
SetSlotOption( SilkScreenMC:Resolution, VALUE_TYPE, NUMBER );

    /*************************************
     **** INSTANCE: MC1
     *************************************/
MakeInstance( MC1, DrillMC );
MC1:NCWidth = 550;
MC1:NCLength = 600;
MC1:MachinedAcc = 0.1;
MC1:MachineName = NCDrillingMachine1;
MC1:MinHoleSize = 0.4;
MC1:MaxHoleSize = 5;

    /*************************************
     **** INSTANCE: MC2
     *************************************/
MakeInstance( MC2, DrillMC );
MC2:NCWidth = 350;
MC2:NCLength = 450;
MC2:MachinedAcc = 0.125;
MC2:MachineName = NCDrillingMachine2;
MC2:MinHoleSize = 0.5;
SetSlotOption( MC2:MaxHoleSize, VALUE_TYPE, NUMBER );
MC2:MaxHoleSize = 5.5;

    /*************************************
     **** INSTANCE: MC11
     *************************************/
MakeInstance( MC11, BlankMC );
MC11:Width = 250;
MC11:Length = 250;
MC11:Accuracy = 0.15;
MC11:MachineName = BlankingMC1;

    /*************************************
     **** INSTANCE: MC12
     *************************************/
MakeInstance( MC12, BlankMC );
MC12:Width = 600;
MC12:Length = 600;
MC12:Accuracy = 0.15;
MC12:MachineName = BlankingMC2;

    /*************************************
     **** INSTANCE: MC21
     *************************************/
MakeInstance( MC21, SilkScreenMC );
SetSlotOption( MC21:Width, VALUE_TYPE, NUMBER );
MC21:Width = 600;
SetSlotOption( MC21:Length, VALUE_TYPE, NUMBER );
```

```
MC21:Length = 600;
SetSlotOption( MC21:Resolution, VALUE_TYPE, NUMBER );
MC21:Resolution = 0.2;
MC21:MachineName = SilkScreenMC1;

    /*************************************
     **** INSTANCE: MC22
     *************************************/
MakeInstance( MC22, SilkScreenMC );
SetSlotOption( MC22:Width, VALUE_TYPE, NUMBER );
MC22:Width = 300;
SetSlotOption( MC22:Length, VALUE_TYPE, NUMBER );
MC22:Length = 300;
SetSlotOption( MC22:Resolution, VALUE_TYPE, NUMBER );
MC22:Resolution = 0.19;
MC22:MachineName = SilkScreenMC2;
```

Appendix 9.7 "MCNameList" Objects

```
    /************************************
     **** CLASS: MCNameList
     ************************************/
MakeClass( MCNameList, Root );

/************** METHOD: MCSelection  **************/
MakeMethod( MCNameList, MCSelection, [],
{
  ClearList( MessageFile:MCSelectionNoMessage );
  ForwardChain( [ NOASSERT ], NULL, MCSelectionRules );
  If ( LengthList( MessageFile:MCSelectionNoMessage ) > 0 )
     Then {
        SetValue( Global, dumstring, Global:FileNameListProd#".msg");
        OpenWriteFile( Global:DriveDir # Global:dumstring, APPEND);
        WriteLine( "/* Process(es) WITHOUT Machine Assigned */" );
        WriteLine( "/* Process Name(s) */ ");
        EnumList (MessageFile:MCSelectionNoMessage, message, WriteLine(message));
        CloseWriteFile();
     };
  PostMessage( "          Machine Selection Finished.
Go to Recipe & Instruction Sheet Generation Session." );
} );
MakeSlot( MCNameList:MCListing );
SetSlotOption( MCNameList:MCListing, MULTIPLE );
ClearList( MCNameList:MCListing );

    /************************************
     **** CLASS: NCDrilling
     ************************************/
MakeClass( NCDrilling, MCNameList );
MakeSlot( NCDrilling:MCListing );
SetSlotOption( NCDrilling:MCListing, MULTIPLE );
SetValue( NCDrillingMachine1 );

    /************************************
     **** CLASS: ITCirSilkScreen
     ************************************/
MakeClass( ITCirSilkScreen, MCNameList );
MakeSlot( ITCirSilkScreen:MCListing );
SetSlotOption( ITCirSilkScreen:MCListing, MULTIPLE );
SetValue( ITCirSilkScreen:MCListing, SilkScreenMC1 );

    /************************************
     **** CLASS: ITSMSilkScreen
     ************************************/
MakeClass( ITSMSilkScreen, MCNameList );
MakeSlot( ITSMSilkScreen:MCListing );
SetSlotOption( ITSMSilkScreen:MCListing, MULTIPLE );
```

```
SetValue( ITSMSilkScreen:MCListing, SilkScreenMC1 );

    /*************************************
     **** CLASS: ITCompMark
     *************************************/
MakeClass( ITCompMark, MCNameList );
MakeSlot( ITCompMark:MCListing );
SetSlotOption( ITCompMark:MCListing, MULTIPLE );
SetValue( ITCompMark:MCListing, SilkScreenMC1 );

    /*************************************
     **** CLASS: BlankingMC
     *************************************/
MakeClass( BlankingMC, MCNameList );
MakeSlot( BlankingMC:MCListing );
SetSlotOption( BlankingMC:MCListing, MULTIPLE );
SetValue( BlankingMC:MCListing, BlankingMC2 );
```

Appendix 9.8 "ProcessRecipe" Objects

```
    /*************************************
     **** CLASS: ProcessRecipe
     *************************************/
MakeClass( ProcessRecipe, Root );

/************** METHOD: GenerateProcessRecipe  **************/
MakeMethod( ProcessRecipe, GenerateProcessRecipe, [],
  {
  ForwardChain( [ NOASSERT ], NULL, CheckGenPanelCuPlatRecipe);
  ForwardChain( [ NOASSERT ], NULL, CheckGenPatternCuPlatRecipe);
  ForwardChain( [ NOASSERT ], NULL, CheckGenPatternNiPlatRecipe);
  ForwardChain( [ NOASSERT ], NULL, CheckGenPatternAuPlatRecipe);
  ForwardChain( [ NOASSERT ], NULL, CheckGenPatternSnPbPlatRecipe);
  ForwardChain( [ NOASSERT ], NULL, CheckGenAuFingerRecipe );
  } );

    /*************************************
     **** INSTANCE: RecPanPlatCu
     *************************************/
MakeInstance( RecPanPlatCu, ProcessRecipe );

/************** METHOD: GenerateProcessRecipe  **************/
MakeMethod( RecPanPlatCu, GenerateProcessRecipe, [],
  ForwardChain( [ NOASSERT ], NULL, RecipeGenPanelCu ) );
MakeSlot( RecPanPlatCu:Current );
SetSlotOption( RecPanPlatCu:Current, VALUE_TYPE, NUMBER );
RecPanPlatCu:Current = 186.275928107412;
MakeSlot( RecPanPlatCu:PlatingTime );
SetSlotOption( RecPanPlatCu:PlatingTime, VALUE_TYPE, NUMBER );
RecPanPlatCu:PlatingTime = 15;

    /*************************************
     **** INSTANCE: RecPatPlatCu
     *************************************/
MakeInstance( RecPatPlatCu, ProcessRecipe );

/************** METHOD: GenerateProcessRecipe  **************/
MakeMethod( RecPatPlatCu, GenerateProcessRecipe, [],
  ForwardChain( [ NOASSERT ], NULL, RecipeGenPatternCu ) );
MakeSlot( RecPatPlatCu:PlatingTime );
SetSlotOption( RecPatPlatCu:PlatingTime, VALUE_TYPE, NUMBER );
RecPatPlatCu:PlatingTime = 45;
MakeSlot( RecPatPlatCu:Current );
SetSlotOption( RecPatPlatCu:Current, VALUE_TYPE, NUMBER );
RecPatPlatCu:Current = 46.6767830202327;

    /*************************************
     **** INSTANCE: RecPatPlatNi
```

```
    ************************************/
MakeInstance( RecPatPlatNi, ProcessRecipe );

/************** METHOD: GenerateProcessRecipe  **************/
MakeMethod( RecPatPlatNi, GenerateProcessRecipe, [],
  {
  ForwardChain( [ NOASSERT ], NULL, RecipeGenPatternNi );
  } );
MakeSlot( RecPatPlatNi:Current );
SetSlotOption( RecPatPlatNi:Current, VALUE_TYPE, NUMBER );
RecPatPlatNi:Current = 19.4654452642239;
MakeSlot( RecPatPlatNi:PlatingTime );
SetSlotOption( RecPatPlatNi:PlatingTime, VALUE_TYPE, NUMBER );
RecPatPlatNi:PlatingTime = 11.58;

    /************************************
     **** INSTANCE: RecPatPlatAu
     ************************************/
MakeInstance( RecPatPlatAu, ProcessRecipe );

/************** METHOD: GenerateProcessRecipe  **************/
MakeMethod( RecPatPlatAu, GenerateProcessRecipe, [],
  {
  ForwardChain( [ NOASSERT ], NULL, RecipeGenPatternAu );
  } );
MakeSlot( RecPatPlatAu:Current );
SetSlotOption( RecPatPlatAu:Current, VALUE_TYPE, NUMBER );
RecPatPlatAu:Current = 1.29979333292;
MakeSlot( RecPatPlatAu:PlatingTime );
SetSlotOption( RecPatPlatAu:PlatingTime, VALUE_TYPE, NUMBER );
RecPatPlatAu:PlatingTime = 7.6;

    /************************************
     **** INSTANCE: RecPatPlatSnPb
     ************************************/
MakeInstance( RecPatPlatSnPb, ProcessRecipe );

/************** METHOD: GenerateProcessRecipe  **************/
MakeMethod( RecPatPlatSnPb, GenerateProcessRecipe, [],
  {
  ForwardChain( [ NOASSERT ], NULL, RecipeGenPatternSnPb );
  } );
MakeSlot( RecPatPlatSnPb:Current );
SetSlotOption( RecPatPlatSnPb:Current, VALUE_TYPE, NUMBER );
RecPatPlatSnPb:Current = 23.3383915101164;
MakeSlot( RecPatPlatSnPb:PlatingTime );
SetSlotOption( RecPatPlatSnPb:PlatingTime, VALUE_TYPE, NUMBER );
RecPatPlatSnPb:PlatingTime = 7.5;

    /************************************
     **** INSTANCE: RecAuFinger
```

```
    ************************************/
MakeInstance( RecAuFinger, ProcessRecipe );

/************** METHOD: GenerateProcessRecipe  **************/
MakeMethod( RecAuFinger, GenerateProcessRecipe, [],
  {
  ForwardChain( [ NOASSERT ], NULL, RecipeGenAuFinger );
  } );
MakeSlot( RecAuFinger:AuCurrent );
SetSlotOption( RecAuFinger:AuCurrent, VALUE_TYPE, NUMBER );
RecAuFinger:AuCurrent = 0.1875;
MakeSlot( RecAuFinger:NiCurrent );
SetSlotOption( RecAuFinger:NiCurrent, VALUE_TYPE, NUMBER );
RecAuFinger:NiCurrent = 1.125;
MakeSlot( RecAuFinger:AuPlatingTime );
SetSlotOption( RecAuFinger:AuPlatingTime, VALUE_TYPE, NUMBER );
RecAuFinger:AuPlatingTime = 7.6;
MakeSlot( RecAuFinger:NiPlatingTime );
SetSlotOption( RecAuFinger:NiPlatingTime, VALUE_TYPE, NUMBER );
RecAuFinger:NiPlatingTime = 7.72;
```

Appendix 9.9 "MessageFile" Objects

```
/************************************
 **** CLASS: MessageFile
 ************************************/
MakeClass( MessageFile, Root );
MakeSlot( MessageFile:OutGenSpecMessage );
SetSlotOption( MessageFile:OutGenSpecMessage, MULTIPLE );
SetValue( MessageFile:OutGenSpecMessage, "AuFingerAuThk -- 0.02<0.025" );
MakeSlot( MessageFile:MCSelectionNoMessage );
SetSlotOption( MessageFile:MCSelectionNoMessage, MULTIPLE );
ClearList( MessageFile:MCSelectionNoMessage );
MakeSlot( MessageFile:ProCanNotMfgInHouse );
SetSlotOption( MessageFile:ProCanNotMfgInHouse, MULTIPLE );
SetValue( MessageFile:ProCanNotMfgInHouse, "ElelessCuPlat--PanelWidth>Process", "PatternCuPlat--
PanelWidth>Process", "PanelCuPlat--PanelDimension>Process", "PanelCuPlat--
PanelDimension>Process", "PatternSnPbPlat--PanelWidth>Process", "SnPbReflow--PanelWidth>Process"
);
```

Appendix 9.10 "SESSION" Window Objects

```
/*************************************
 **** INSTANCE: SESSION
 *************************************/
SESSION:X = 125;
SESSION:Y = 24;
SESSION:Title = "Knowledge Based PCB Process Planning System";
SESSION:SessionNumber = 0;
SESSION:Width = 390;
SESSION:Height = 310;
SESSION:Visible = TRUE;
SESSION:State = NORM;
SESSION:Menu = TRUE;
SESSION:Titlebar = TRUE;
SESSION:Sizebox = TRUE;
ResetWindow ( SESSION );

/*************************************
 **** INSTANCE: InPutCustProd
 *************************************/
MakeInstance( InPutCustProd, KSession );
InPutCustProd:X = 43;
InPutCustProd:Y = 27;
InPutCustProd:Title = "Input Board Characteristics";
InPutCustProd:SessionNumber = 1;
InPutCustProd:Width = 547;
InPutCustProd:Height = 286;
InPutCustProd:Visible = FALSE;
InPutCustProd:State = HIDDEN;
InPutCustProd:Menu = TRUE;
InPutCustProd:Titlebar = TRUE;
InPutCustProd:Sizebox = TRUE;
ResetWindow ( InPutCustProd );

/*************************************
 **** INSTANCE: ReviewApproveProcess
 *************************************/
MakeInstance( ReviewApproveProcess, KSession );
ReviewApproveProcess:X = 57;
ReviewApproveProcess:Y = 26;
ReviewApproveProcess:Title = "Review and Approve Process";
ReviewApproveProcess:SessionNumber = 2;
ReviewApproveProcess:Width = 506;
ReviewApproveProcess:Height = 334;
ReviewApproveProcess:Visible = FALSE;
ReviewApproveProcess:State = HIDDEN;
ReviewApproveProcess:Menu = TRUE;
ReviewApproveProcess:Titlebar = TRUE;
ReviewApproveProcess:Sizebox = FALSE;
```

```
ResetWindow ( ReviewApproveProcess );

    /*************************************
     **** INSTANCE: ProcessPlanOutputSession
     *************************************/
MakeInstance( ProcessPlanOutputSession, KSession );
ProcessPlanOutputSession:X = 3;
ProcessPlanOutputSession:Y = 4;
ProcessPlanOutputSession:Title = "Process Plan Display";
ProcessPlanOutputSession:SessionNumber = 3;
ProcessPlanOutputSession:Width = 630;
ProcessPlanOutputSession:Height = 397;
ProcessPlanOutputSession:Visible = FALSE;
ProcessPlanOutputSession:State = HIDDEN;
SetValue( ProcessPlanOutputSession:BackgroundColor, 255, 255, 255 );
ProcessPlanOutputSession:Menu = TRUE;
ProcessPlanOutputSession:Titlebar = TRUE;
ProcessPlanOutputSession:Sizebox = TRUE;
ResetWindow ( ProcessPlanOutputSession );

    /*************************************
     **** INSTANCE: ProdDocWindow
     *************************************/
MakeInstance( ProdDocWindow, KSession );
ProdDocWindow:X = 33;
ProdDocWindow:Y = 12;
ProdDocWindow:Title = "Input/Update Product Document Details";
ProdDocWindow:SessionNumber = 4;
ProdDocWindow:Width = 557;
ProdDocWindow:Height = 345;
ProdDocWindow:Visible = FALSE;
ProdDocWindow:State = HIDDEN;
ProdDocWindow:Menu = TRUE;
ProdDocWindow:Titlebar = TRUE;
ProdDocWindow:Sizebox = TRUE;
ResetWindow ( ProdDocWindow );

    /*************************************
     **** INSTANCE: ProProdWindow
     *************************************/
MakeInstance( ProProdWindow, KSession );
ProProdWindow:X = 40;
ProProdWindow:Y = 12;
ProProdWindow:Title = "Input/Update Req. Qty. and Delivery Details";
ProProdWindow:SessionNumber = 5;
ProProdWindow:Width = 547;
ProProdWindow:Height = 212;
ProProdWindow:Visible = FALSE;
ProProdWindow:State = HIDDEN;
ProProdWindow:Menu = TRUE;
ProProdWindow:Titlebar = TRUE;
```

```
ProProdWindow:Sizebox = TRUE;
 ResetWindow ( ProProdWindow );

    /*************************************
     **** INSTANCE: ProLaminWindow
     *************************************/
MakeInstance( ProLaminWindow, KSession );
ProLaminWindow:X = 42;
ProLaminWindow:Y = 5;
ProLaminWindow:Title = "Input/Update Laminate Characteristics.";
ProLaminWindow:SessionNumber = 6;
ProLaminWindow:Width = 547;
ProLaminWindow:Height = 238;
ProLaminWindow:Visible = FALSE;
ProLaminWindow:State = HIDDEN;
ProLaminWindow:Menu = TRUE;
ProLaminWindow:Titlebar = TRUE;
ProLaminWindow:Sizebox = TRUE;
 ResetWindow ( ProLaminWindow );

    /*************************************
     **** INSTANCE: ProPanDimenWindow
     *************************************/
MakeInstance( ProPanDimenWindow, KSession );
ProPanDimenWindow:X = 42;
ProPanDimenWindow:Y = 12;
ProPanDimenWindow:Title = "Input/Update Panel Dimension Data";
ProPanDimenWindow:SessionNumber = 7;
ProPanDimenWindow:Width = 548;
ProPanDimenWindow:Height = 296;
ProPanDimenWindow:Visible = FALSE;
ProPanDimenWindow:State = HIDDEN;
ProPanDimenWindow:Menu = TRUE;
ProPanDimenWindow:Titlebar = TRUE;
ProPanDimenWindow:Sizebox = TRUE;
 ResetWindow ( ProPanDimenWindow );

    /*************************************
     **** INSTANCE: ProHoleWindow
     *************************************/
MakeInstance( ProHoleWindow, KSession );
ProHoleWindow:X = 44;
ProHoleWindow:Y = 11;
ProHoleWindow:Title = "Input/Update Hole Characteristics.";
ProHoleWindow:SessionNumber = 8;
ProHoleWindow:Width = 544;
ProHoleWindow:Height = 278;
ProHoleWindow:Visible = FALSE;
ProHoleWindow:State = HIDDEN;
ProHoleWindow:Menu = TRUE;
ProHoleWindow:Titlebar = TRUE;
```

```
ProHoleWindow:Sizebox = TRUE;
 ResetWindow ( ProHoleWindow );

    /*************************************
     **** INSTANCE: ProCircuitWindow
     *************************************/
MakeInstance( ProCircuitWindow, KSession );
ProCircuitWindow:X = 39;
ProCircuitWindow:Y = 10;
ProCircuitWindow:Title = "Input/Update Circuit Characteristic.";
ProCircuitWindow:SessionNumber = 9;
ProCircuitWindow:Width = 549;
ProCircuitWindow:Height = 278;
ProCircuitWindow:Visible = FALSE;
ProCircuitWindow:State = HIDDEN;
ProCircuitWindow:Menu = TRUE;
ProCircuitWindow:Titlebar = TRUE;
ProCircuitWindow:Sizebox = TRUE;
 ResetWindow ( ProCircuitWindow );

    /*************************************
     **** INSTANCE: ProPlatWindow
     *************************************/
MakeInstance( ProPlatWindow, KSession );
ProPlatWindow:X = 45;
ProPlatWindow:Y = 25;
ProPlatWindow:Title = "Input/Update Plating Characteristics.";
ProPlatWindow:SessionNumber = 10;
ProPlatWindow:Width = 545;
ProPlatWindow:Height = 278;
ProPlatWindow:Visible = FALSE;
ProPlatWindow:State = HIDDEN;
ProPlatWindow:Menu = TRUE;
ProPlatWindow:Titlebar = TRUE;
ProPlatWindow:Sizebox = TRUE;
 ResetWindow ( ProPlatWindow );

    /*************************************
     **** INSTANCE: ProAuFingerWindow
     *************************************/
MakeInstance( ProAuFingerWindow, KSession );
ProAuFingerWindow:X = 45;
ProAuFingerWindow:Y = 28;
ProAuFingerWindow:Title = "Input/Update Gold Finger Characteristics.";
ProAuFingerWindow:SessionNumber = 11;
ProAuFingerWindow:Width = 544;
ProAuFingerWindow:Height = 274;
ProAuFingerWindow:Visible = FALSE;
ProAuFingerWindow:State = HIDDEN;
ProAuFingerWindow:Menu = TRUE;
ProAuFingerWindow:Titlebar = TRUE;
```

```
ProAuFingerWindow:Sizebox = TRUE;
 ResetWindow ( ProAuFingerWindow );

    /*************************************
     **** INSTANCE: ProSolderMaskWindow
     *************************************/
MakeInstance( ProSolderMaskWindow, KSession );
ProSolderMaskWindow:X = 42;
ProSolderMaskWindow:Y = 26;
ProSolderMaskWindow:Title = "Input/Update Solder Masking Details.";
ProSolderMaskWindow:SessionNumber = 12;
ProSolderMaskWindow:Width = 545;
ProSolderMaskWindow:Height = 282;
ProSolderMaskWindow:Visible = FALSE;
ProSolderMaskWindow:State = HIDDEN;
ProSolderMaskWindow:Menu = TRUE;
ProSolderMaskWindow:Titlebar = TRUE;
ProSolderMaskWindow:Sizebox = TRUE;
 ResetWindow ( ProSolderMaskWindow );

    /*************************************
     **** INSTANCE: ProCompWindow
     *************************************/
MakeInstance( ProCompWindow, KSession );
ProCompWindow:X = 46;
ProCompWindow:Y = 28;
ProCompWindow:Title = "Input/Update Component Marking Characteristics.";
ProCompWindow:SessionNumber = 13;
ProCompWindow:Width = 544;
ProCompWindow:Height = 278;
ProCompWindow:Visible = FALSE;
ProCompWindow:State = HIDDEN;
ProCompWindow:Menu = TRUE;
ProCompWindow:Titlebar = TRUE;
ProCompWindow:Sizebox = TRUE;
 ResetWindow ( ProCompWindow );

    /*************************************
     **** INSTANCE: ProProdDocWindow
     *************************************/
MakeInstance( ProProdDocWindow, KSession );
ProProdDocWindow:X = 42;
ProProdDocWindow:Y = 6;
ProProdDocWindow:Title = "Input/Update Board Doc.  Details.";
ProProdDocWindow:SessionNumber = 14;
ProProdDocWindow:Width = 543;
ProProdDocWindow:Height = 229;
ProProdDocWindow:Visible = FALSE;
ProProdDocWindow:State = HIDDEN;
ProProdDocWindow:Menu = TRUE;
ProProdDocWindow:Titlebar = TRUE;
```

```
ProProdDocWindow:Sizebox = TRUE;
 ResetWindow ( ProProdDocWindow );
```

Appendix 9.11 "Button" Objects

```
    /************************************
     **** INSTANCE: Button1
     ************************************/
MakeInstance( Button1, Button );
Button1:SessionNumber = 2;
Button1:Title = "End This Session.";
SetValue( Button1:ForegroundColor, 0, 0, 0 );
SetValue( Button1:BackgroundColor, 255, 255, 0 );
SetValue( Button1:ForegroundColor2, 0, 0, 0 );
SetValue( Button1:BackgroundColor2, 255, 255, 255 );
Button1:Visible = TRUE;
Button1:X = 11;
Button1:Y = 216;
Button1:Width = 157;
Button1:Height = 64;
Button1:Action = ClearSession2;
 ResetImage ( Button1 );

    /************************************
     **** INSTANCE: Button2
     ************************************/
MakeInstance( Button2, Button );
Button2:SessionNumber = 1;
Button2:Title = "Product Documents.";
SetValue( Button2:ForegroundColor, 0, 0, 0 );
SetValue( Button2:BackgroundColor, 192, 192, 192 );
SetValue( Button2:ForegroundColor2, 0, 0, 0 );
SetValue( Button2:BackgroundColor2, 255, 255, 255 );
Button2:Visible = TRUE;
Button2:X = 3;
Button2:Y = 109;
Button2:Action = PostProdDoc;
Button2:Width = 121;
Button2:Height = 48;
 ResetImage ( Button2 );

    /************************************
     **** INSTANCE: Button3
     ************************************/
MakeInstance( Button3, Button );
Button3:SessionNumber = 1;
Button3:Title = "Board Plating Requirements.";
SetValue( Button3:ForegroundColor, 0, 0, 0 );
SetValue( Button3:BackgroundColor, 192, 192, 192 );
SetValue( Button3:ForegroundColor2, 0, 0, 0 );
SetValue( Button3:BackgroundColor2, 255, 255, 255 );
Button3:Visible = TRUE;
Button3:X = 255;
```

```
Button3:Y = 59;
Button3:Action = PostPlatReq;
Button3:Width = 121;
Button3:Height = 48;
 ResetImage ( Button3 );

    /*************************************
     **** INSTANCE: Button4
     *************************************/
MakeInstance( Button4, Button );
Button4:SessionNumber = 1;
Button4:Title = "Hole Specifications.";
SetValue( Button4:ForegroundColor, 0, 0, 0 );
SetValue( Button4:BackgroundColor, 192, 192, 192 );
SetValue( Button4:ForegroundColor2, 0, 0, 0 );
SetValue( Button4:BackgroundColor2, 255, 255, 255 );
Button4:Visible = TRUE;
Button4:X = 129;
Button4:Y = 109;
Button4:Action = PostHoleSpec;
Button4:Width = 121;
Button4:Height = 48;
 ResetImage ( Button4 );

    /*************************************
     **** INSTANCE: Button5
     *************************************/
MakeInstance( Button5, Button );
Button5:SessionNumber = 1;
Button5:Title = "Solder Masking Details.";
SetValue( Button5:ForegroundColor, 0, 0, 0 );
SetValue( Button5:BackgroundColor, 192, 192, 192 );
SetValue( Button5:ForegroundColor2, 0, 0, 0 );
SetValue( Button5:BackgroundColor2, 255, 255, 255 );
Button5:Visible = TRUE;
Button5:X = 382;
Button5:Y = 8;
Button5:Action = PostSolderMask;
Button5:Width = 121;
Button5:Height = 48;
 ResetImage ( Button5 );

    /*************************************
     **** INSTANCE: Button6
     *************************************/
MakeInstance( Button6, Button );
Button6:SessionNumber = 1;
Button6:Title = Button6;
SetValue( Button6:ForegroundColor, 0, 0, 0 );
SetValue( Button6:BackgroundColor, 192, 192, 192 );
SetValue( Button6:BackgroundColor2, 255, 255, 255 );
```

```
Button6:Visible = TRUE;
Button6:X = 100;
Button6:Y = 444;
Button6:Height = 48;
Button6:Width = 115;
SetValue( Button6:ForegroundColor2, 0, 0, 0 );
 ResetImage ( Button6 );

    /*************************************
     **** INSTANCE: Button7
     *************************************/
MakeInstance( Button7, Button );
Button7:SessionNumber = 1;
Button7:Title = "Panel and Unit Dimension Details.";
SetValue( Button7:ForegroundColor, 0, 0, 0 );
SetValue( Button7:BackgroundColor, 192, 192, 192 );
SetValue( Button7:ForegroundColor2, 0, 0, 0 );
SetValue( Button7:BackgroundColor2, 255, 255, 255 );
Button7:Visible = TRUE;
Button7:X = 128;
Button7:Y = 58;
Button7:Height = 48;
Button7:Width = 121;
Button7:Action = PostPanDimen;
 ResetImage ( Button7 );

    /*************************************
     **** INSTANCE: Button8
     *************************************/
MakeInstance( Button8, Button );
Button8:SessionNumber = 1;
Button8:Title = "Laminate Details.";
SetValue( Button8:ForegroundColor, 0, 0, 0 );
SetValue( Button8:BackgroundColor, 192, 192, 192 );
SetValue( Button8:ForegroundColor2, 0, 0, 0 );
SetValue( Button8:BackgroundColor2, 255, 255, 255 );
Button8:Visible = TRUE;
Button8:X = 128;
Button8:Y = 8;
Button8:Height = 48;
Button8:Width = 121;
Button8:Action = PostLamChar;
Button8:Font = "MS Sans Serif";
Button8:TextSize = 8;
Button8:Underline = FALSE;
Button8:Bold = TRUE;
Button8:Italic = FALSE;
Button8:StrikeOut = FALSE;
 ResetImage ( Button8 );
```

```
    /*************************************
     **** INSTANCE: Button9
     *************************************/
MakeInstance( Button9, Button );
Button9:SessionNumber = 1;
Button9:Title = "Gold Finger Details.";
SetValue( Button9:ForegroundColor, 0, 0, 0 );
SetValue( Button9:BackgroundColor, 192, 192, 192 );
SetValue( Button9:ForegroundColor2, 0, 0, 0 );
SetValue( Button9:BackgroundColor2, 255, 255, 255 );
Button9:Visible = TRUE;
Button9:X = 255;
Button9:Y = 109;
Button9:Height = 48;
Button9:Width = 121;
Button9:Action = PostAuFing;
 ResetImage ( Button9 );

    /*************************************
     **** INSTANCE: Button10
     *************************************/
MakeInstance( Button10, Button );
Button10:SessionNumber = 1;
Button10:Title = "Project Documents Details.";
SetValue( Button10:ForegroundColor, 0, 0, 0 );
SetValue( Button10:BackgroundColor, 192, 192, 192 );
SetValue( Button10:ForegroundColor2, 0, 0, 0 );
SetValue( Button10:BackgroundColor2, 255, 255, 255 );
Button10:Visible = TRUE;
Button10:X = 3;
Button10:Y = 8;
Button10:Height = 48;
Button10:Width = 121;
Button10:Action = PostDoc;
 ResetImage ( Button10 );

    /*************************************
     **** INSTANCE: Button12
     *************************************/
MakeInstance( Button12, Button );
Button12:SessionNumber = 1;
Button12:Title = "Circuit Characteristic.";
SetValue( Button12:ForegroundColor, 0, 0, 0 );
SetValue( Button12:BackgroundColor, 192, 192, 192 );
SetValue( Button12:ForegroundColor2, 0, 0, 0 );
SetValue( Button12:BackgroundColor2, 255, 255, 255 );
Button12:Visible = TRUE;
Button12:X = 255;
Button12:Y = 8;
Button12:Action = PostFeatCir;
Button12:Height = 48;
```

```
Button12:Width = 121;
Button12:Font = "MS Sans Serif";
Button12:TextSize = 8;
Button12:Bold = TRUE;
Button12:Underline = FALSE;
Button12:Italic = FALSE;
Button12:StrikeOut = FALSE;
 ResetImage ( Button12 );

    /*************************************
     **** INSTANCE: Button13
     *************************************/
MakeInstance( Button13, Button );
Button13:SessionNumber = 1;
Button13:Title = "Back To Main Menu.";
SetValue( Button13:ForegroundColor, 0, 0, 0 );
SetValue( Button13:BackgroundColor, 255, 255, 0 );
SetValue( Button13:ForegroundColor2, 0, 0, 0 );
SetValue( Button13:BackgroundColor2, 255, 255, 255 );
Button13:Visible = TRUE;
Button13:X = 402;
Button13:Y = 110;
Button13:Height = 46;
Button13:Width = 82;
Button13:Action = ClearSession1;
 ResetImage ( Button13 );

    /*************************************
     **** INSTANCE: Button15
     *************************************/
MakeInstance( Button15, Button );
Button15:SessionNumber = 2;
Button15:Title = "Find Feasible Process.";
SetValue( Button15:ForegroundColor, 0, 0, 0 );
SetValue( Button15:BackgroundColor, 192, 192, 192 );
SetValue( Button15:ForegroundColor2, 0, 0, 0 );
SetValue( Button15:BackgroundColor2, 255, 255, 255 );
Button15:Visible = TRUE;
Button15:X = 11;
Button15:Y = 9;
Button15:Height = 64;
Button15:Width = 157;
Button15:Action = FindInHouPro;
 ResetImage ( Button15 );

    /*************************************
     **** INSTANCE: Button18
     *************************************/
MakeInstance( Button18, Button );
Button18:SessionNumber = 0;
Button18:Title = "Process Selection and Sequencing Session.";
```

```
SetValue( Button18:ForegroundColor, 0, 0, 0 );
SetValue( Button18:BackgroundColor, 192, 192, 192 );
SetValue( Button18:ForegroundColor2, 0, 0, 0 );
SetValue( Button18:BackgroundColor2, 255, 255, 255 );
Button18:Visible = FALSE;
Button18:X = 200;
Button18:Y = 66;
Button18:Action = OpenSession2;
Button18:Height = 53;
Button18:Width = 165;
Button18:TabStop = 0;
ResetImage ( Button18 );

    /************************************
     **** INSTANCE: Button19
     ************************************/
MakeInstance( Button19, Button );
Button19:SessionNumber = 1;
Button19:Title = "Required Quantity and Delivery Date.";
SetValue( Button19:ForegroundColor, 0, 0, 0 );
SetValue( Button19:BackgroundColor, 192, 192, 192 );
SetValue( Button19:ForegroundColor2, 0, 0, 0 );
SetValue( Button19:BackgroundColor2, 255, 255, 255 );
Button19:Visible = TRUE;
Button19:X = 3;
Button19:Y = 59;
Button19:Height = 48;
Button19:Width = 121;
Button19:Action = PostQuantity;
ResetImage ( Button19 );

    /************************************
     **** INSTANCE: Button14
     ************************************/
MakeInstance( Button14, Button );
Button14:SessionNumber = 0;
Button14:Title = "Input/Up-date Customer General Specification.";
SetValue( Button14:ForegroundColor, 0, 0, 0 );
SetValue( Button14:BackgroundColor, 255, 255, 0 );
SetValue( Button14:ForegroundColor2, 0, 0, 0 );
SetValue( Button14:BackgroundColor2, 255, 255, 255 );
Button14:Visible = TRUE;
Button14:X = 12;
Button14:Y = 67;
Button14:Width = 165;
Button14:Height = 53;
Button14:Action = InUpCustGenSpec;
Button14:TabStop = 3;
ResetImage ( Button14 );

    /************************************
```

```
    **** INSTANCE: Button20
    ************************************/
MakeInstance( Button20, Button );
Button20:SessionNumber = 0;
Button20:Title = "Machine Selection.";
SetValue( Button20:ForegroundColor, 0, 0, 0 );
SetValue( Button20:BackgroundColor, 192, 192, 192 );
SetValue( Button20:ForegroundColor2, 0, 0, 0 );
SetValue( Button20:BackgroundColor2, 255, 255, 255 );
Button20:Visible = FALSE;
Button20:X = 200;
Button20:Y = 120;
Button20:Action = MCSelection;
Button20:Width = 165;
Button20:Height = 53;
Button20:TabStop = 0;
 ResetImage ( Button20 );

    /************************************
    **** INSTANCE: Button21
    ************************************/
MakeInstance( Button21, Button );
Button21:SessionNumber = 0;
Button21:Title = "Recipe & Instruction Sheet Generation.";
SetValue( Button21:ForegroundColor, 0, 0, 0 );
SetValue( Button21:BackgroundColor, 192, 192, 192 );
SetValue( Button21:ForegroundColor2, 0, 0, 0 );
SetValue( Button21:BackgroundColor2, 255, 255, 255 );
Button21:Visible = FALSE;
Button21:X = 200;
Button21:Y = 175;
Button21:Width = 165;
Button21:Height = 53;
Button21:Action = ProcessRecipeGenFunction;
Button21:TabStop = 0;
 ResetImage ( Button21 );

    /************************************
    **** INSTANCE: Button22
    ************************************/
MakeInstance( Button22, Button );
Button22:SessionNumber = 0;
Button22:Title = "Up-date In-house Process Capability Knowledge";
SetValue( Button22:ForegroundColor, 0, 0, 0 );
SetValue( Button22:BackgroundColor, 255, 255, 0 );
SetValue( Button22:ForegroundColor2, 0, 0, 0 );
SetValue( Button22:BackgroundColor2, 255, 255, 255 );
Button22:Visible = TRUE;
Button22:X = 11;
Button22:Y = 7;
Button22:Width = 165;
```

```
Button22:Height = 53;
Button22:Action = UpProCap;
Button22:TabStop = 1;
 ResetImage ( Button22 );

    /************************************
     **** INSTANCE: Button23
     ************************************/
MakeInstance( Button23, Button );
Button23:SessionNumber = 0;
Button23:Title = "Load Customer General Spec. & Board Reqt. Data File";
SetValue( Button23:ForegroundColor, 0, 0, 0 );
SetValue( Button23:BackgroundColor, 255, 255, 0 );
SetValue( Button23:ForegroundColor2, 0, 0, 0 );
SetValue( Button23:BackgroundColor2, 255, 255, 255 );
Button23:Visible = TRUE;
Button23:X = 188;
Button23:Y = 7;
Button23:Width = 165;
Button23:Height = 53;
Button23:Action = LdCustSpBdReqt;
Button23:TabStop = 2;
 ResetImage ( Button23 );

    /************************************
     **** INSTANCE: Button24
     ************************************/
MakeInstance( Button24, Button );
Button24:SessionNumber = 2;
Button24:Title = "Save Process Name and Mfg. Sequence in File.";
SetValue( Button24:ForegroundColor, 0, 0, 0 );
SetValue( Button24:BackgroundColor, 192, 192, 192 );
SetValue( Button24:ForegroundColor2, 0, 0, 0 );
SetValue( Button24:BackgroundColor2, 255, 255, 255 );
Button24:Visible = TRUE;
Button24:X = 11;
Button24:Y = 147;
Button24:Action = SaveProNameSeqInFile;
Button24:Width = 157;
Button24:Height = 64;
 ResetImage ( Button24 );

    /************************************
     **** INSTANCE: Button25
     ************************************/
MakeInstance( Button25, Button );
Button25:SessionNumber = 2;
Button25:Title = "Sequence Approved Process.";
SetValue( Button25:ForegroundColor, 0, 0, 0 );
SetValue( Button25:BackgroundColor, 192, 192, 192 );
SetValue( Button25:ForegroundColor2, 0, 0, 0 );
```

```
SetValue( Button25:BackgroundColor2, 255, 255, 255 );
Button25:Visible = TRUE;
Button25:X = 11;
Button25:Y = 77;
Button25:Action = SequenceProcess;
Button25:Width = 157;
Button25:Height = 64;
 ResetImage ( Button25 );

    /*************************************
     **** INSTANCE: Button16
     *************************************/
MakeInstance( Button16, Button );
Button16:SessionNumber = 0;
Button16:Title = "Plan Next Product (Using Same Customer General Specification Data Base)";
Button16:Visible = FALSE;
Button16:X = 18;
Button16:Y = 174;
SetValue( Button16:ForegroundColor, 0, 0, 0 );
SetValue( Button16:BackgroundColor, 192, 192, 192 );
SetValue( Button16:ForegroundColor2, 0, 0, 0 );
SetValue( Button16:BackgroundColor2, 255, 255, 255 );
Button16:Width = 165;
Button16:Height = 53;
Button16:Action = NextProduct;
 ResetImage ( Button16 );

    /*************************************
     **** INSTANCE: Button26
     *************************************/
MakeInstance( Button26, Button );
Button26:SessionNumber = 0;
Button26:Title = "Finish Process Planning Session";
Button26:Visible = FALSE;
Button26:X = 18;
Button26:Y = 121;
SetValue( Button26:ForegroundColor, 0, 0, 0 );
SetValue( Button26:BackgroundColor, 192, 192, 192 );
SetValue( Button26:ForegroundColor2, 0, 0, 0 );
SetValue( Button26:BackgroundColor2, 255, 255, 255 );
Button26:Action = FinishProcessPlan;
Button26:Width = 165;
Button26:Height = 53;
 ResetImage ( Button26 );

    /*************************************
     **** INSTANCE: Button27
     *************************************/
MakeInstance( Button27, Button );
Button27:SessionNumber = 3;
Button27:Title = "Close Ses'n";
```

```
Button27:Visible = TRUE;
Button27:X = 579;
Button27:Y = 86;
SetValue( Button27:ForegroundColor, 0, 0, 0 );
SetValue( Button27:BackgroundColor, 192, 192, 192 );
SetValue( Button27:ForegroundColor2, 0, 0, 0 );
SetValue( Button27:BackgroundColor2, 255, 255, 255 );
Button27:Width = 47;
Button27:Height = 138;
Button27:Action = ClosePlanDisplay;
 ResetImage ( Button27 );

    /************************************
     **** INSTANCE: Button11
     ************************************/
MakeInstance( Button11, Button );
Button11:SessionNumber = 4;
Button11:Title = "Close This Session.";
Button11:Visible = TRUE;
Button11:X = 234;
Button11:Y = 122;
SetValue( Button11:ForegroundColor, 0, 0, 0 );
SetValue( Button11:BackgroundColor, 192, 192, 192 );
SetValue( Button11:ForegroundColor2, 0, 0, 0 );
SetValue( Button11:BackgroundColor2, 255, 255, 255 );
Button11:Action = FinProDocWin;
Button11:Width = 152;
Button11:Height = 41;
 ResetImage ( Button11 );

    /************************************
     **** INSTANCE: Button28
     ************************************/
MakeInstance( Button28, Button );
Button28:SessionNumber = 5;
Button28:Title = "Close This Session.";
Button28:Visible = TRUE;
Button28:X = 226;
Button28:Y = 87;
SetValue( Button28:ForegroundColor, 0, 0, 0 );
SetValue( Button28:BackgroundColor, 192, 192, 192 );
SetValue( Button28:ForegroundColor2, 0, 0, 0 );
SetValue( Button28:BackgroundColor2, 255, 255, 255 );
Button28:Action = FinQuanDelWin;
Button28:Width = 143;
Button28:Height = 42;
 ResetImage ( Button28 );
```

```
    /************************************
     **** INSTANCE: Button29
     ************************************/
MakeInstance( Button29, Button );
Button29:SessionNumber = 6;
Button29:Title = "Close This Session.";
Button29:Visible = TRUE;
Button29:X = 228;
Button29:Y = 137;
SetValue( Button29:ForegroundColor, 0, 0, 0 );
SetValue( Button29:BackgroundColor, 192, 192, 192 );
SetValue( Button29:ForegroundColor2, 0, 0, 0 );
SetValue( Button29:BackgroundColor2, 255, 255, 255 );
Button29:Width = 154;
Button29:Height = 38;
Button29:Action = FinProLaminWin;
 ResetImage ( Button29 );

    /************************************
     **** INSTANCE: Button30
     ************************************/
MakeInstance( Button30, Button );
Button30:SessionNumber = 7;
Button30:Title = "Close This Session.";
Button30:Visible = TRUE;
Button30:X = 248;
Button30:Y = 197;
SetValue( Button30:ForegroundColor, 0, 0, 0 );
SetValue( Button30:BackgroundColor, 192, 192, 192 );
SetValue( Button30:ForegroundColor2, 0, 0, 0 );
SetValue( Button30:BackgroundColor2, 255, 255, 255 );
Button30:Action = FinProPanWin;
Button30:Width = 140;
Button30:Height = 37;
 ResetImage ( Button30 );

    /************************************
     **** INSTANCE: Button31
     ************************************/
MakeInstance( Button31, Button );
Button31:SessionNumber = 8;
Button31:Title = "Close This Session.";
Button31:Visible = TRUE;
Button31:X = 237;
Button31:Y = 94;
SetValue( Button31:ForegroundColor, 0, 0, 0 );
SetValue( Button31:BackgroundColor, 192, 192, 192 );
SetValue( Button31:ForegroundColor2, 0, 0, 0 );
SetValue( Button31:BackgroundColor2, 255, 255, 255 );
Button31:Width = 156;
Button31:Height = 53;
```

```
Button31:Action = FinProHoleWin;
 ResetImage ( Button31 );

    /*************************************
     **** INSTANCE: Button32
     *************************************/
MakeInstance( Button32, Button );
Button32:SessionNumber = 9;
Button32:Title = "Close This Session.";
Button32:Visible = TRUE;
Button32:X = 222;
Button32:Y = 142;
SetValue( Button32:ForegroundColor, 0, 0, 0 );
SetValue( Button32:BackgroundColor, 192, 192, 192 );
SetValue( Button32:ForegroundColor2, 0, 0, 0 );
SetValue( Button32:BackgroundColor2, 255, 255, 255 );
Button32:Width = 151;
Button32:Height = 39;
Button32:Action = FinProCirWin;
 ResetImage ( Button32 );

    /*************************************
     **** INSTANCE: Button33
     *************************************/
MakeInstance( Button33, Button );
Button33:SessionNumber = 10;
Button33:Title = "Close This Session.";
Button33:Visible = TRUE;
Button33:X = 224;
Button33:Y = 81;
SetValue( Button33:ForegroundColor, 0, 0, 0 );
SetValue( Button33:BackgroundColor, 192, 192, 192 );
SetValue( Button33:ForegroundColor2, 0, 0, 0 );
SetValue( Button33:BackgroundColor2, 255, 255, 255 );
Button33:Width = 150;
Button33:Height = 42;
Button33:Action = FinProPlatWin;
 ResetImage ( Button33 );

    /*************************************
     **** INSTANCE: Button34
     *************************************/
MakeInstance( Button34, Button );
Button34:SessionNumber = 11;
Button34:Title = "Close This Session.";
Button34:Visible = TRUE;
Button34:X = 239;
Button34:Y = 139;
SetValue( Button34:ForegroundColor, 0, 0, 0 );
SetValue( Button34:BackgroundColor, 192, 192, 192 );
SetValue( Button34:ForegroundColor2, 0, 0, 0 );
```

```
SetValue( Button34:BackgroundColor2, 255, 255, 255 );
Button34:Width = 136;
Button34:Height = 37;
Button34:Action = FinAuFingerWin;
 ResetImage ( Button34 );

    /*************************************
     **** INSTANCE: Button35
     *************************************/
MakeInstance( Button35, Button );
Button35:SessionNumber = 12;
Button35:Title = "Close This Session.";
Button35:Visible = TRUE;
Button35:X = 214;
Button35:Y = 103;
SetValue( Button35:ForegroundColor, 0, 0, 0 );
SetValue( Button35:BackgroundColor, 192, 192, 192 );
SetValue( Button35:ForegroundColor2, 0, 0, 0 );
SetValue( Button35:BackgroundColor2, 255, 255, 255 );
Button35:Width = 134;
Button35:Height = 37;
Button35:Action = FinProSoldWin;
 ResetImage ( Button35 );

    /*************************************
     **** INSTANCE: Button36
     *************************************/
MakeInstance( Button36, Button );
Button36:SessionNumber = 1;
Button36:Title = "Component Marking Details.";
Button36:Visible = TRUE;
Button36:X = 382;
Button36:Y = 59;
SetValue( Button36:ForegroundColor, 0, 0, 0 );
SetValue( Button36:BackgroundColor, 192, 192, 192 );
SetValue( Button36:ForegroundColor2, 0, 0, 0 );
SetValue( Button36:BackgroundColor2, 255, 255, 255 );
Button36:Action = PostComp;
Button36:Width = 121;
Button36:Height = 48;
 ResetImage ( Button36 );

    /*************************************
     **** INSTANCE: Button37
     *************************************/
MakeInstance( Button37, Button );
Button37:SessionNumber = 13;
Button37:Title = "Close This Session.";
Button37:Visible = TRUE;
Button37:X = 206;
Button37:Y = 79;
```

```
SetValue( Button37:ForegroundColor, 0, 0, 0 );
SetValue( Button37:BackgroundColor, 192, 192, 192 );
SetValue( Button37:ForegroundColor2, 0, 0, 0 );
SetValue( Button37:BackgroundColor2, 255, 255, 255 );
Button37:Action = FinProCompWin;
Button37:Width = 129;
Button37:Height = 43;
 ResetImage ( Button37 );

    /*************************************
     **** INSTANCE: Button38
     *************************************/
MakeInstance( Button38, Button );
Button38:SessionNumber = 14;
Button38:Title = "Close This Session.";
Button38:Visible = TRUE;
Button38:X = 248;
Button38:Y = 86;
SetValue( Button38:ForegroundColor, 0, 0, 0 );
SetValue( Button38:BackgroundColor, 192, 192, 192 );
SetValue( Button38:ForegroundColor2, 0, 0, 0 );
SetValue( Button38:BackgroundColor2, 255, 255, 255 );
Button38:Action = FinProProdWin;
Button38:Width = 119;
Button38:Height = 43;
 ResetImage ( Button38 );

    /*************************************
     **** INSTANCE: Button17
     *************************************/
MakeInstance( Button17, Button );
Button17:SessionNumber = 0;
Button17:Title = "Input/Up-date Board Requirements.";
SetValue( Button17:ForegroundColor, 0, 0, 0 );
SetValue( Button17:BackgroundColor, 255, 255, 0 );
SetValue( Button17:ForegroundColor2, 0, 0, 0 );
SetValue( Button17:BackgroundColor2, 255, 255, 255 );
Button17:Visible = TRUE;
Button17:X = 12;
Button17:Y = 131;
Button17:Action = InUpProdReqt;
Button17:Width = 165;
Button17:Height = 53;
Button17:TabStop = 4;
 ResetImage ( Button17 );
```

Appendix 10.1 CircuitA Process Planning Results

/* File to Store Product Requirements and Operation Instruction */
/* This File Name : p8001.ins */
Date and Time Process Recipe Generated : 7/20/94 11:29:00AM

/* ----- PRODUCT REQUIREMENTS ---- */
/* ----- Product Details ------ */
Customer Name : CustomerA
Customer Circuit Part Number : A001
Name of Circuit : CircuitA
Customer Purchase Order Number : PO001
In House Part Number : p8001

/* ---------- Production Quantity and Date Required ---------- */
Batch Size of this Production Batch (unit) : 5000
Total Expected Quantity (unit) : 20000
Time Available for Production (days) : 30

/* ----------Product Documentation Details----------*/
Drawing Number : DN001
Drilling File Number : -
Film Number : PCB8
Specification Number : 1001

/* ---------- Laminate Type ---------- */
Type of Laminate Material : FR4
Thickness of Laminate (mm) : 1.6
Copper Foil Thickness : 1
Side of Copper : Single
Colour of Laminate : Green

/* ---------- Panel and Unit Dimension Details ----------*/
Circuit Unit Width (mm) : 150
Circuit Unit Length (mm) : 124
Panel Width (mm) : 330
Panel Length (mm) : 278
No. of Unit Per Panel : 4
Outline Dimenstion Deviation (0.01mm) : 15
Estimated Die Cost: 5000

/* ---------- Holes Characteristic ----------*/
Allowed Hole Diameter Deviation (0.01mm) : 7.5
Allowed Hole Location Deviation (0.01mm) : 7.5
Maximum Hole Size (mm) : 2.5
Minimum Hole Size (mm) : 0.8

/* ---------- Circuit Feature ----------*/
Minimum Annual Ring (0.01mm) : 25
Total Unit Circuit Area (sq.in.) : 1.32

Minimum Circuit Thickness (0.01mm) : 4
Minimum Circuit Width (0.01mm) : 70
Minimum Circuit Spacing (0.01mm) : 350

/* ---------- Plating Requirement ----------*/
Minimum Plated Copper Thickness (0.001in) : 0
Type of Surface Finishing Rquired: SnPb
Minimum Plated Surface Finishing Thickness (0.001in) : 0.5

/* ---------- Gold Finger Requirements ---------- */
Gold Thickness (0.001in) : 0
Height of the Gold Finger (mm) : 0
Ni Under Gold Thickness (0.001in) : 0
Gold Finger Unit Area : 0
No. of Unit Per Edge : 0

/* ---------- Solder Masking Requirements ---------- */
Type of Solder Masking Material: PC501
Solder Masking Colour: Green
Minimum Thickness (0.01mm) : 1
Minimum Allowed Ring Clearance (0.01mm) : 20

/* ---------- Component Marking Requirements ---------- */
Component Marking Material Allowed : materialA
Marking Colour : white

/* Suggested Process in Mfg. Sequence*/
1. ImTranCirSilkScreen
2. PatternSnPbPlat
3. CirImageStrip
4. CuEtching
5. SnPbReflow
6. ImTranSMSilkScreen
7. CompMarkPrint
8. 2ndDrillOrHolePunch
9. OutLineBlank

/* ---------- OPERATION INSTRUCTION ---------- */

/* --- Image Transfer Circuit Silk Screen Operation Instruction --- */
Silk Screen Machine to Use:-
SilkScreenMC1

/* --- Pattern SnPb Plating Operation Instruction ---*/
Requirement:-
Plating Area:- 0.149 sq. ft.
Minimum Surface Finishing Thickness:- 0.5 in 0.001 in. unit.
Settings:-
Current Setting:- 2.98 A.

Plating Time:- 12.5 mins.

/* --- Image Transfer Solder Masking Operation Instruction ---*/
Requirement:-
Type of Solder Masking Material:- PC501
Minimum Thickness:- 1
Minimum Allowed Ring Clearance:- 20 in 0.01 mm.
Solder Masking Material Colour:- Green
Silk Screen Machine to Use:-
SilkScreenMC1

/* --- Image Transfer Comp. Marking Operation Instruction ---*/
Requirement:-
Component Marking Material Allowed:- materialA
Marking Colour:- white
Silk Screen Machine to Use:-
SilkScreenMC1

/* --- Blanking Machine Operation Instruction ---*/
Blanking Machine to Use:-
BlankingMC2

Appendix 10.2 CircuitB Process Planning Results

/* File to Store Product Requirements and Operation Instruction */
/* This File Name : p8002.ins */
Date and Time Process Recipe Generated : 7/20/94 11:32:35AM

/* ----- PRODUCT REQUIREMENTS ---- */
/* ----- Product Details ------ */
Customer Name : CustomerA
Customer Circuit Part Number : A002
Name of Circuit : CircuitB
Customer Purchase Order Number : PO002
In House Part Number : p8002

/* ---------- Production Quantity and Date Required ---------- */
Batch Size of this Production Batch (unit) : 5000
Total Expected Quantity (unit) : 20000
Time Available for Production (days) : 30

/* ----------Product Documentation Details----------*/
Drawing Number : DN002
Drilling File Number : -
Film Number : PCB
Specification Number : 1001

/* ---------- Laminate Type ---------- */
Type of Laminate Material : FR4
Thickness of Laminate (mm) : 1.6
Copper Foil Thickness : 1
Side of Copper : Double
Colour of Laminate : Green

/* ---------- Panel and Unit Dimension Details ----------*/
Circuit Unit Width (mm) : 106
Circuit Unit Length (mm) : 100
Panel Width (mm) : 242
Panel Length (mm) : 230
No. of Unit Per Panel : 4
Outline Dimenstion Deviation (0.01mm) : 15
Estimated Die Cost: 9000

/* ---------- Holes Characteristic ----------*/
Allowed Hole Diameter Deviation (0.01mm) : 7.5
Allowed Hole Location Deviation (0.01mm) : 7.5
Maximum Hole Size (mm) : 2.5
Minimum Hole Size (mm) : 0.8

/* ---------- Circuit Feature ----------*/
Minimum Annual Ring (0.01mm) : 25
Total Unit Circuit Area (sq.in.) : 3.24

Minimum Circuit Thickness (0.01mm) : 4
Minimum Circuit Width (0.01mm) : 50
Minimum Circuit Spacing (0.01mm) : 50

/* ---------- Plating Requirement ----------*/
Minimum Plated Copper Thickness (0.001in) : 1.5
Type of Surface Finishing Rquired: SnPb
Minimum Plated Surface Finishing Thickness (0.001in) : 0.5

/* ---------- Gold Finger Requirements ---------- */
Gold Thickness (0.001in) : 0
Height of the Gold Finger (mm) : 0
Ni Under Gold Thickness (0.001in) : 0
Gold Finger Unit Area : 0
No. of Unit Per Edge : 0

/* ---------- Solder Masking Requirements ---------- */
Type of Solder Masking Material: PC501
Solder Masking Colour: Green
Minimum Thickness (0.01mm) : 1
Minimum Allowed Ring Clearance (0.01mm) : 50

/* ---------- Component Marking Requirements ---------- */
Component Marking Material Allowed : materialA
Marking Colour : white

/* Suggested Process in Mfg. Sequence*/
1. NCDrilling
2. ElectrolessCuPlat
3. PanelCuPlat
4. ImTranCirSilkScreen
5. PatternCuPlat
6. PatternSnPbPlat
7. CirImageStrip
8. CuEtching
9. SnPbReflow
10. ImTranSMSilkScreen
11. CompMarkPrint
12. OutLineBlank

/* ---------- OPERATION INSTRUCTION ---------- */
/* --- NC Drilling Process Operation Instruction --- */
Drilling Requirement:-
Panel Width: 242 mm
Panel Length: 230 mm
Allowed Hole Location Deviation: 7.5
Allowed Hole Size Deviation: 7.5 in 0.01 mm
Minimum Hole Size: 0.8 mm
Maximum Hole Size: 2.5 mm

NC Drilling Machine to use:-
NCDrillingMachine1
NCDrillingMachine2

/* --- Panel Copper Plating Operation Instruction --- */
Requirement:-
Panel Surface Area:- 111320 sq. mm OR 1.2 sq. ft
Copper Thickness is at 0.0005 in.
Settings :-
Current Setting:- 48 A.
Plating Time:- 15 mins.

/* --- Image Transfer Circuit Silk Screen Operation Instruction --- */
Silk Screen Machine to Use:-
SilkScreenMC1
SilkScreenMC2

/* --- Pattern Copper Plating Operation Instruction ---*/
Requirement :-
Plating Area:- 0.262 sq. ft.
Minimum Plated Copper Thickness:- 1.5 in 0.001 in. unit.(0.0005 in. panel plated)
Settings:-
Current Setting:- 10.46 A.
Plating Time:- 30 mins.

/* --- Pattern SnPb Plating Operation Instruction ---*/
Requirement:-
Plating Area:- 0.262 sq. ft.
Minimum Surface Finishing Thickness:- 0.5 in 0.001 in. unit.
Settings:-
Current Setting:- 5.23 A.
Plating Time:- 12.5 mins.

/* --- Image Transfer Solder Masking Operation Instruction ---*/
Requirement:-
Type of Solder Masking Material:- PC501
Minimum Thickness:- 1
Minimum Allowed Ring Clearance:- 50 in 0.01 mm.
Solder Masking Material Colour:- Green
Silk Screen Machine to Use:-
SilkScreenMC1
SilkScreenMC2

/* --- Image Transfer Comp. Marking Operation Instruction ---*/
Requirement:-
Component Marking Material Allowed:- materialA
Marking Colour:- white
Silk Screen Machine to Use:-
SilkScreenMC1
SilkScreenMC2

/* --- Blanking Machine Operation Instruction ---*/
Blanking Machine to Use:-
    BlankingMC1
    BlankingMC2

Appendix 10.3 CircuitC Process Planning Results

/* File to Store Product Requirements and Operation Instruction */
/* This File Name : p8003.ins */
Date and Time Process Recipe Generated : 6/30/94 6:06:05PM

/* ----- PRODUCT REQUIREMENTS ---- */
/* ----- Product Details ------ */
Customer Name : CustomerB
Customer Circuit Part Number : B001
Name of Circuit : CircuitC
Customer Purchase Order Number : PO008
In House Part Number : p8003

/* ---------- Production Quantity and Date Required ---------- */
Batch Size of this Production Batch (unit) : 5000
Total Expected Quantity (unit) : 20000
Time Available for Production (days) : 30

/* ----------Product Documentation Details----------*/
Drawing Number : DN003
Drilling File Number : -
Film Number : PCB8
Specification Number : 1002

/* ---------- Laminate Type ---------- */
Type of Laminate Material : FR-4
Thickness of Laminate (mm) : 1.6
Copper Foil Thickness : 1
Side of Copper : Double
Colour of Laminate : Green

/* ---------- Panel and Unit Dimension Details ----------*/
Circuit Unit Width (mm) : 100
Circuit Unit Length (mm) : 100
Panel Width (mm) : 230
Panel Length (mm) : 230
No. of Unit Per Panel : 4
Outline Dimenstion Deviation (0.01mm) : 15
Estimated Die Cost: 6500

/* ---------- Holes Characteristic ----------*/
Allowed Hole Diameter Deviation (0.01mm) : 7.5
Allowed Hole Location Deviation (0.01mm) : 7.5
Maximum Hole Size (mm) : 2.5
Minimum Hole Size (mm) : 0.8

/* ---------- Circuit Feature ----------*/
Minimum Annual Ring (0.01mm) : 25
Total Unit Circuit Area (sq.in.) : 3.36

Minimum Circuit Thickness (0.01mm) : 4
Minimum Circuit Width (0.01mm) : 50
Minimum Circuit Spacing (0.01mm) : 50

/* ---------- Plating Requirement ----------*/
Minimum Plated Copper Thickness (0.001in) : 1.5
Type of Surface Finishing Rquired: Au
Minimum Plated Surface Finishing Thickness (0.001in) : 0.02

/* ---------- Gold Finger Requirements ---------- */
Gold Thickness (0.001in) : 0
Height of the Gold Finger (mm) : 0
Ni Under Gold Thickness (0.001in) : 0
Gold Finger Unit Area : 0
No. of Unit Per Edge : 0

/* ---------- Solder Masking Requirements ---------- */
Type of Solder Masking Material: PC501
Solder Masking Colour: Green
Minimum Thickness (0.01mm) : 1
Minimum Allowed Ring Clearance (0.01mm) : 5

/* ---------- Component Marking Requirements ---------- */
Component Marking Material Allowed : meterialB
Marking Colour : yellow

/* Suggested Process in Mfg. Sequence*/
1. NCDrilling
2. ElectrolessCuPlat
3. PanelCuPlat
4. ImTranCirSilkScreen
5. PatternCuPlat
6. PatternAuPlat
7. CirImageStrip
8. CuEtching
9. ImTranSMSilkScreen
10. CompMarkPrint
11. OutLineBlank

/* ---------- OPERATION INSTRUCTION ---------- */
/* --- NC Drilling Process Operation Instruction --- */
Drilling Requirement:-
Panel Width: 230 mm
Panel Length: 230 mm
Allowed Hole Location Deviation: 7.5
Allowed Hole Size Deviation: 7.5 in 0.01 mm
Minimum Hole Size: 0.8 mm
Maximum Hole Size: 2.5 mm
NC Drilling Machine to use:-

NCDrillingMachine1
NCDrillingMachine2

/* --- Panel Copper Plating Operation Instruction --- */
Requirement:-
Panel Surface Area:- 105800 sq. mm OR 1.14 sq. ft
Copper Thickness is at 0.0005 in.
Settings :-
Current Setting:- 45.6 A.
Plating Time:- 15 mins.

/* --- Image Transfer Circuit Silk Screen Operation Instruction --- */
Silk Screen Machine to Use:-
SilkScreenMC1
SilkScreenMC2

/* --- Pattern Copper Plating Operation Instruction ---*/
Requirement :-
Plating Area:- 0.26 sq. ft.
Minimum Plated Copper Thickness:- 1.5 in 0.001 in. unit.(0.0005 in. panel plated)
Settings:-
Current Setting:- 10.4 A.
Plating Time:- 30 mins.

/* --- Pattern Gold Plating Operation Instruction ---*/
Requirement:-
Plating Area:- 0.26 sq. ft.
Minimum Surface Finishing Thickness:- 0.02 in 0.001 in. unit.
Settings:-
Current Setting:- 1.3 A.
Plating Time:- 7.6 mins.

/* --- Image Transfer Solder Masking Operation Instruction ---*/
Requirement:-
Type of Solder Masking Material:- PC501
Minimum Thickness:- 1
Minimum Allowed Ring Clearance:- 5 in 0.01 mm.
Solder Masking Material Colour:- Green
Silk Screen Machine to Use:-
SilkScreenMC1
SilkScreenMC2

/* --- Image Transfer Comp. Marking Operation Instruction ---*/
Requirement:-
Component Marking Material Allowed:- meterialB
Marking Colour:- yellow
Silk Screen Machine to Use:-
SilkScreenMC1
SilkScreenMC2

/* --- Blanking Machine Operation Instruction ---*/

Blanking Machine to Use:-
BlankingMC1
BlankingMC2

Appendix 10.4 CircuitD Process Planning Results

/* File to Store Product Requirements and Operation Instruction */
/* This File Name : p8004.ins */
Date and Time Process Recipe Generated : 7/5/94 11:06:25AM

/* ----- PRODUCT REQUIREMENTS ---- */
/* ----- Product Details ------ */
Customer Name : CustomerB
Customer Circuit Part Number : A004
Name of Circuit : CircuitD
Customer Purchase Order Number : PO004
In House Part Number : p8004

/* ---------- Production Quantity and Date Required ---------- */
Batch Size of this Production Batch (unit) : 5000
Total Expected Quantity (unit) : 20000
Time Available for Production (days) : 30

/* ----------Product Documentation Details----------*/
Drawing Number : DN004
Drilling File Number : -
Film Number : PCB4
Specification Number : 1002

/* ---------- Laminate Type ---------- */
Type of Laminate Material : FR4
Thickness of Laminate (mm) : 1.6
Copper Foil Thickness : 1
Side of Copper : Double
Colour of Laminate : Green

/* ---------- Panel and Unit Dimension Details ----------*/
Circuit Unit Width (mm) : 241
Circuit Unit Length (mm) : 127.5
Panel Width (mm) : 512
Panel Length (mm) : 422.5
No. of Unit Per Panel : 6
Outline Dimenstion Deviation (0.01mm) : 12.5
Estimated Die Cost: 8000

/* ---------- Holes Characteristic ----------*/
Allowed Hole Diameter Deviation (0.01mm) : 7.5
Allowed Hole Location Deviation (0.01mm) : 7.5
Maximum Hole Size (mm) : 4
Minimum Hole Size (mm) : 0.8

/* ---------- Circuit Feature ----------*/
Minimum Annual Ring (0.01mm) : 20
Total Unit Circuit Area (sq.in.) : 18.1

Minimum Circuit Thickness (0.01mm) : 4
Minimum Circuit Width (0.01mm) : 100
Minimum Circuit Spacing (0.01mm) : 100

/* ---------- Plating Requirement ----------*/
Minimum Plated Copper Thickness (0.001in) : 2
Type of Surface Finishing Rquired: SnPb
Minimum Plated Surface Finishing Thickness (0.001in) : 0.3

/* ---------- Gold Finger Requirements ---------- */
Gold Thickness (0.001in) : 0.02
Height of the Gold Finger (mm) : 13
Ni Under Gold Thickness (0.001in) : 0.2
Gold Finger Unit Area : 1.8
No. of Unit Per Edge : 3

/* ---------- Solder Masking Requirements ---------- */
Type of Solder Masking Material: PC501
Solder Masking Colour: Green
Minimum Thickness (0.01mm) : 1
Minimum Allowed Ring Clearance (0.01mm) : 50

/* ---------- Component Marking Requirements ---------- */
Component Marking Material Allowed : materialA
Marking Colour : white

/* Suggested Process in Mfg. Sequence*/
1. NCDrilling
2. ElectrolessCuPlat
3. PanelCuPlat
4. ImTranCirSilkScreen
5. PatternCuPlat
6. PatternSnPbPlat
7. CirImageStrip
8. CuEtching
9. AuFinger
10. SnPbReflow
11. ImTranSMSilkScreen
12. CompMarkPrint
13. OutLineBlank

/* Prod. Reqt. Out of Cust. Gen. Spec. */
/* Type of Reqt. --- Prod.--Spec. (Value)*/
AuFingerAuThk -- 0.02<0.025

/* Process(es) Cannot Perform In-house*/
/* Process Name(s) -- Problem */
ElelessCuPlat--PanelWidth>Process
PatternCuPlat--PanelWidth>Process
PanelCuPlat--PanelDimension>Process

PanelCuPlat--PanelDimension>Process
PatternSnPbPlat--PanelWidth>Process
SnPbReflow--PanelWidth>Process

/* ---------- OPERATION INSTRUCTION --------- */
/* --- NC Drilling Process Operation Instruction --- */
Drilling Requirement:-
  Panel Width: 512 mm
  Panel Length: 422.5 mm
  Allowed Hole Location Deviation: 7.5
  Allowed Hole Size Deviation: 7.5 in 0.01 mm
  Minimum Hole Size: 0.8 mm
  Maximum Hole Size: 4 mm
NC Drilling Machine to use:-
  NCDrillingMachine1

/* --- Panel Copper Plating Operation Instruction --- */
Requirement:-
  Panel Surface Area:- 432640 sq. mm OR 4.658 sq. ft
  Copper Thickness is at 0.0005 in.
Settings :-
  Current Setting:- 186.3 A.
  Plating Time:- 15 mins.

/* --- Image Transfer Circuit Silk Screen Operation Instruction --- */
Silk Screen Machine to Use:-
  SilkScreenMC1

/* --- Pattern Copper Plating Operation Instruction ---*/
Requirement :-
  Plating Area:- 1.167 sq. ft.
  Minimum Plated Copper Thickness:- 2 in 0.001 in. unit.(0.0005 in. panel plated)
Settings:-
  Current Setting:- 46.68 A.
  Plating Time:- 45 mins.

/* --- Pattern SnPb Plating Operation Instruction ---*/
Requirement:-
  Plating Area:- 1.167 sq. ft.
  Minimum Surface Finishing Thickness:- 0.3 in 0.001 in. unit.
Settings:-
  Current Setting:- 23.34 A.
  Plating Time:- 7.5 mins.

/* --- Gold Finger Plating Operation Instruction ---*/
Requirement:-
  Min. Ni under Gold Thickness:- 0.2 in 0.001 in. unit.
  Min. Gold Thickness:- 0.02 in 0.001 in. unit
Settings:-
  Ni Current Setting:- 1.13 A.

Ni Plating Time:- 7.72 mins.
Gold Current Setting:- 0.19 A.
Gold Plating Time:- 7.6 mins.

/* --- Image Transfer Solder Masking Operation Instruction ---*/
Requirement:-
Type of Solder Masking Material:- PC501
Minimum Thickness:- 1
Minimum Allowed Ring Clearance:- 50 in 0.01 mm.
Solder Masking Material Colour:- Green
Silk Screen Machine to Use:-
SilkScreenMC1

/* --- Image Transfer Comp. Marking Operation Instruction ---*/
Requirement:-
Component Marking Material Allowed:- materialA
Marking Colour:- white
Silk Screen Machine to Use:-
SilkScreenMC1

/* --- Blanking Machine Operation Instruction ---*/
Blanking Machine to Use:-
BlankingMC2

Appendix 10.5 CircuitE Process Planning Results

/* File to Store Product Requirements and Operation Instruction */
/* This File Name : p8005.ins */
Date and Time Process Recipe Generated : 6/30/94 6:03:55PM

/* ----- PRODUCT REQUIREMENTS ---- */
/* ----- Product Details ------ */
Customer Name : CustomerA
Customer Circuit Part Number : A005
Name of Circuit : CircuitE
Customer Purchase Order Number : PO005
In House Part Number : p8005

/* ---------- Production Quantity and Date Required ---------- */
Batch Size of this Production Batch (unit) : 5000
Total Expected Quantity (unit) : 20000
Time Available for Production (days) : 30

/* ----------Product Documentation Details----------*/
Drawing Number : DN005
Drilling File Number : -
Film Number : PCB10
Specification Number : 1001

/* ---------- Laminate Type ---------- */
Type of Laminate Material : FR4
Thickness of Laminate (mm) : 1.6
Copper Foil Thickness : 1
Side of Copper : Double
Colour of Laminate : Green

/* ---------- Panel and Unit Dimension Details ----------*/
Circuit Unit Width (mm) : 300
Circuit Unit Length (mm) : 300
Panel Width (mm) : 630
Panel Length (mm) : 630
No. of Unit Per Panel : 4
Outline Dimenstion Deviation (0.01mm) : 10
Estimated Die Cost: 10000

/* ---------- Holes Characteristic ----------*/
Allowed Hole Diameter Deviation (0.01mm) : 12
Allowed Hole Location Deviation (0.01mm) : 5
Maximum Hole Size (mm) : 5
Minimum Hole Size (mm) : 0.9

/* ---------- Circuit Feature ----------*/
Minimum Annual Ring (0.01mm) : 2
Total Unit Circuit Area (sq.in.) : 6.2

Minimum Circuit Thickness (0.01mm) : 4
Minimum Circuit Width (0.01mm) : 10
Minimum Circuit Spacing (0.01mm) : 10

/* ---------- Plating Requirement ----------*/
Minimum Plated Copper Thickness (0.001in) : 2
Type of Surface Finishing Rquired: BareCopper
Minimum Plated Surface Finishing Thickness (0.001in) : 0.3

/* ---------- Gold Finger Requirements ---------- */
Gold Thickness (0.001in) : 0
Height of the Gold Finger (mm) : 0
Ni Under Gold Thickness (0.001in) : 0
Gold Finger Unit Area : 0
No. of Unit Per Edge : 0

/* ---------- Solder Masking Requirements ---------- */
Type of Solder Masking Material: PC401
Solder Masking Colour: Green
Minimum Thickness (0.01mm) : 1
Minimum Allowed Ring Clearance (0.01mm) : 4

/* ---------- Component Marking Requirements ---------- */
Component Marking Material Allowed : meterialB
Marking Colour : yellow

/* Suggested Process in Mfg. Sequence*/
1. NCDrilling
2. ElectrolessCuPlat
3. PanelCuPlat
4. ImTranCirSilkScreen
5. PatternCuPlat
6. PatternSnPbPlat
7. CirImageStrip
8. CuEtching
9. SnPbStripping
10. ImTranSMSilkScreen
11. SolderLevelling
12. CompMarkPrint
13. OutLineBlank

/* Prod. Reqt. Out of Cust. Gen. Spec. */
/* Type of Reqt. --- Prod.--Spec. (Value)*/
SolderPadClear -- 4<10
SolderMaskMatl -- PC401Not accepted
CircuitWidth -- 10<20
CircuitSpacing -- 10<15
CirAnnularRing -- 2<15

/* Process(es) Cannot Perform In-house*/

/* Process Name(s) -- Problem */
NCDrilling--PanelSize>MCSize
NCDrilling--PanelSize>MCSize
ElelessCuPlat--PanelWidth>Process
ElelessCuPlat--PanelLength>Process
PatternCuPlat--PanelWidth>Process
PatternCuPlat--PanelLength>Process
PanelCuPlat--PanelDimension>Process
PanelCuPlat--PanelDimension>Process
InSilkScreen--PanelSize>Process
InSilkScreen--PanelSize>Process
PatternSnPbPlat--PanelWidth>Process
PatternSnPbPlat--PanelLength>Process
SnPbLevelling--PanelSize>Process
SnPbLevelling--PanelSize>Process
InSilkScreen--PanelSize>Process
InSilkScreen--PanelSize>Process

/* ---------- OPERATION INSTRUCTION --------- */
/* --- NC Drilling Process Operation Instruction --- */
Drilling Requirement:-
  Panel Width: 630 mm
  Panel Length: 630 mm
  Allowed Hole Location Deviation: 5
  Allowed Hole Size Deviation: 12 in 0.01 mm
  Minimum Hole Size: 0.9 mm
  Maximum Hole Size: 5 mm
NC Drilling Machine to use:-
  *****NOSuitableNCDrillingMC

/* --- Panel Copper Plating Operation Instruction --- */
Requirement:-
  Panel Surface Area:- 793800 sq. mm OR 8.546 sq. ft
  Copper Thickness is at 0.0005 in.
Settings :-
  Current Setting:- 341.8 A.
  Plating Time:- 15 mins.

/* --- Image Transfer Circuit Silk Screen Operation Instruction --- */
Silk Screen Machine to Use:-
  *****NOSuitableSilkScreenMC

/* --- Pattern Copper Plating Operation Instruction ---*/
Requirement :-
  Plating Area:- 0.649 sq. ft.
  Minimum Plated Copper Thickness:- 2 in 0.001 in. unit.(0.0005 in. panel plated)
Settings:-
  Current Setting:- 25.96 A.
  Plating Time:- 45 mins.

/* --- Pattern SnPb Plating Operation Instruction ---*/
Requirement:-
Plating Area:- 0.649 sq. ft.
Minimum Surface Finishing Thickness:- 0.3 in 0.001 in. unit.
Settings:-
Current Setting:- 12.98 A.
Plating Time:- 7.5 mins.

/* --- Image Transfer Solder Masking Operation Instruction ---*/
Requirement:-
Type of Solder Masking Material:- PC401
Minimum Thickness:- 1
Minimum Allowed Ring Clearance:- 4 in 0.01 mm.
Solder Masking Material Colour:- Green
Silk Screen Machine to Use:-
*****NOSuitableSilkScreenMC

/* --- Image Transfer Comp. Marking Operation Instruction ---*/
Requirement:-
Component Marking Material Allowed:- meterialB
Marking Colour:- yellow
Silk Screen Machine to Use:-
*****NOSuitableSilkScreenMC

/* --- Blanking Machine Operation Instruction ---*/
Blanking Machine to Use:-
*****NOSuitableBlankingMC