

**A methodology for developing resilient distributed control systems.**

TAHOLAKIAN, Aram M.

Available from Sheffield Hallam University Research Archive (SHURA) at:

<http://shura.shu.ac.uk/20418/>

---

This document is the author deposited version. You are advised to consult the publisher's version if you wish to cite from it.

**Published version**

TAHOLAKIAN, Aram M. (1997). A methodology for developing resilient distributed control systems. Doctoral, Sheffield Hallam University (United Kingdom)..

---

**Copyright and re-use policy**

See <http://shura.shu.ac.uk/information.html>

Sheffield Hallam University

**REFERENCE ONLY**

ProQuest Number: 10701064

All rights reserved

INFORMATION TO ALL USERS

The quality of this reproduction is dependent upon the quality of the copy submitted.

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

**uest**

ProQuest 10701064

Published by ProQuest LLC(2017). Copyright of the Dissertation is held by the Author.

All rights reserved.

This work is protected against unauthorized copying under Title 17, United States Code  
Microform Edition © ProQuest LLC.

ProQuest LLC.  
789 East Eisenhower Parkway  
P.O. Box 1346  
Ann Arbor, MI 48106- 1346

**A Methodology for Developing  
Resilient Distributed Control Systems**

**Aram Meguerditch Taholakian BEng (Hons)**

**A thesis submitted in partial fulfilment of the  
requirements of Sheffield Hallam University for the degree of  
Doctor of Philosophy**

**April 1997**

## Abstract

Manufacturing industries rely on automated manufacturing systems to improve the efficiency, quality and flexibility of production. Such systems typically consist of a variety of manufacturing machinery and control hardware, e.g. CNC machine tools, robots, PCs, Programmable Logic Controllers (PLCs) etc., which operate concurrently. The cost of developing and implementing an automated manufacturing system is high, and is particularly so if the control system is found to be unreliable or unsafe during operation. Distributed Control Systems are generally used to control complex concurrent systems.

At present the methods used to develop DCSs tend to follow a sequence of steps, viz. a statement of the requirements of the DCS, a functional specification of the DCS, the design of the DCS, generation of the software code for the DCS, implementation of the software. This step approach is inadequate because of the dissimilarity of techniques used to represent each step, which leads to difficulties in ensuring equivalence between the final implementation of the DCS and the initial requirements, which in turn leads to errors in the final software. To overcome this, work has been conducted to unify the specification, design, and software coding phases of the DCS development procedure by ensuring formal equivalencies between them. One particular outcome of such previous work is a tool named Petri Net - Occam Methodology, developed by Dr. P. Gray, which produces dependable Occam code for DCSs. Gray's methodology produces readable designs, directly from the specification of systems, in a graphical but formal way, and results in a Petri Net graph which is equivalent to the final Occam code. However, his methodology is not for a complete DCS but only for one containing Transputers.

The PLC is widely used in industry and an integral part of DCSs for Automated Manufacture. This research has developed a methodology, named  $PN \leftrightarrow PLC$ , which produces dependable PLC control programs, in a graphical but formal way, directly from a system's specification. It uses the same tool, Petri Nets, for both designing and simulating the control system, and specifies rules which ensure the correct design, simulation and encoding of PLC programs. The PN designs are a one-to-one equivalent to PLC code and can be directly translated into Ladder Diagrams. Therefore if the simulation shows the design to be correct, the final software will be correct.

$PN \leftrightarrow PLC$  works as a stand alone tool for developing dependable PLC control programs, and also unifies with Gray's methodology to produce a complete tool for developing a resilient DCS containing Transputers and PLCs. The unification of the two methodologies is also reported in this thesis.

The research work presented in this thesis contributes to knowledge in the field of DCS development. Recommendations for further work regarding the applicability of the unified methodology on a wide scale industrial basis are also given.

# Table of Contents

<b>1. Introduction.....</b>	<b>2</b>
<b>1.1 Automated Manufacture.....</b>	<b>2</b>
1.1.1 Flexible Manufacturing .....	2
<b>1.2 Distributed Control Systems.....</b>	<b>3</b>
<b>1.3 Development of Distributed Control Software .....</b>	<b>5</b>
1.3.1 Informal Methods for Designing DCSs.....	6
1.3.2 Formal Methods for Designing DCSs .....	6
<b>1.4 The Need for a Methodology for Developing DCSs.....</b>	<b>8</b>
<b>2. Background .....</b>	<b>12</b>
<b>2.1 The FMC at the School of Engineering .....</b>	<b>12</b>
2.1.1 Levels of Control .....	14
2.1.2 Reasons for Distributing Control.....	16
<b>2.2 Gray's Petri Net - Occam Methodology .....</b>	<b>17</b>
2.2.1 Transputers and Occam .....	18
2.2.1.1 Occam: Language Definition.....	18
2.2.2 Introduction to Petri Nets.....	19
2.2.2.1 High Level Petri Nets .....	21
2.2.3 The Motivation and Objectives of Gray's Methodology.....	22
2.2.4 Gray's Methodology Applied to the FMC.....	24
2.2.5 Advantages, Disadvantages and Claims of Gray's Methodology .....	26
<b>2.3 Programmable Logic Controllers.....</b>	<b>26</b>
2.3.1 Available PLC Programming Tools and Techniques .....	27
2.3.1.1 Instruction List.....	27

2.3.1.2 Structured Text .....	28
2.3.1.3 Function Block Diagram.....	28
2.3.1.4 Sequential Function Chart .....	28
2.3.1.5 Ladder Diagram .....	29
2.3.1.5.1 Step Ladder Diagram .....	30
2.3.2 Review of Existing PLC Languages.....	30
<b>2.4 PNs and PLCs: Previous Work .....</b>	<b>32</b>
2.4.1 PNs for Designing and Modelling DCSs Containing PLCs .....	32
2.4.2 PNs for Designing PLC Programs .....	32
<b>2.5 The Aim and Objectives of the Ph.D. Research .....</b>	<b>34</b>
<b>3. The Application of Gray's Methodology to an Overall DCS.....</b>	<b>37</b>
<b>3.1 Gray's Methodology Applied to a Simple PLC Control Task.....</b>	<b>37</b>
3.1.1 Design and Simulation.....	38
3.1.2 Translation of PN into LDs.....	42
<b>3.2 Discussion .....</b>	<b>43</b>
<b>4. PN<math>\leftrightarrow</math>PLC: The Methodology .....</b>	<b>46</b>
<b>4.1 Design Process.....</b>	<b>47</b>
4.1.1 Design Rules.....	48
4.1.2 Terminology and Symbols.....	52
<b>4.2 Translation Process .....</b>	<b>54</b>
4.2.1 Translation Rules .....	54
<b>4.3 Simulation Process.....</b>	<b>56</b>
4.3.1 Simulation Steps .....	57
4.3.2 Simulation Steps Applied to Scenario 2 .....	61
4.3.3 Simulation Results .....	64

4.4 The Investigative Approach Taken to Develop PN $\leftrightarrow$ PLC .....	64
4.5 Designing the Control Algorithm of the Oil Tank Using SFCs.....	70
5. PN $\leftrightarrow$ PLC Applied to the FMC.....	73
5.1 PN $\leftrightarrow$ PLC Applied to the Puma Work Station .....	74
5.1.1 Enable Vice to Table PN Group .....	76
5.1.2 PN $\leftrightarrow$ PLC and Timers.....	82
5.1.3 Conveyor Controller .....	84
5.2 PN $\leftrightarrow$ PLC Applied to Miller Work Station .....	84
5.3 PN $\leftrightarrow$ PLC Applied to Lathe Work Station.....	87
5.3.1 Terminology and Symbols (Revised) .....	92
5.4 Error Handling .....	94
5.4.1 Reliability and Safety Achieved by PN $\leftrightarrow$ PLC.....	97
5.4.1.1 Reporting Errors .....	101
6. Unification of the Methodologies.....	105
6.1 The Graphical and Logical Unification of the Methodologies .....	105
6.2 The Unification of the Methodologies for Simulation .....	109
6.3 The Unification of the Methodologies for Translation.....	111
6.4 Fault Avoidance and Elimination Achieved by the Unified Methodology	112
6.5 Discussion .....	113
7. Conclusions and Recommendations for Future Work.....	117
7.1 Conclusions.....	117
7.2 Recommendations for Future Work.....	120
References.....	123
Appendices.....	131



## List of Figures

Figure 2-1 Layout of the FMC at the School of Engineering .....	13
Figure 2-2 Levels of Control of the FMC .....	14
Figure 2-3 The Labyrinth PN Graph of the FMC .....	inside back cover
Figure 2-4(a) A Simple Petri Net.....	19
Figure 2-4(b) A Marked Petri Net.....	20
Figure 2-4(c) Marking After the Firing of the Transition .....	20
Figure 2-4(d) Inhibitor Arc PN Graph Showing the Marking Before and After the Firing of the Transition.....	21
Figure 2-5 Gray's PN Graph for the Control of the FMC .....	inside back cover
Figure 3-1 Oil Tank Level.....	38
Figure 3-2 PN Graph of Oil Tank Level Control Produced By Gray's Methodology .....	38
Figure 3-2(a) Marking Achieved as a Result of Simulating Oil Sensor Switching on .....	39
Figure 3-2(b) Marking Achieved as a Result of Simulating Push Button Being Pressed .....	40
Figure 3-2(c) Marking Achieved as a Result of Simulating Push Button Being Released .....	40
Figure 3-2(d) Marking Achieved as a Result of Simulating Oil Sensor Switching off.....	41
Figure 4-1 Layout of PN $\leftrightarrow$ PLC Graph .....	47
Figure 4-2(a) First Design Rule Applied to Siren .....	48
Figure 4-2(b) Second Design Rule Applied to Siren.....	49
Figure 4-2(bl) LD Rung for Siren Obtained by Applying PN $\leftrightarrow$ PLC Translation Rules to Figure 4-2(b).....	50

<b>Figure 4-2(b1) A Realistic Marking Applied to Figure 4-2(b).....</b>	<b>50</b>
<b>Figure 4-2(b2) Evolution of the Marking After Applying Simulation Step 3 to Figure 4-2(b1).....</b>	<b>50</b>
<b>Figure 4-2(c) Design Rules Applied to Siren and Lamp .....</b>	<b>51</b>
<b>Figure 4-2(d) Final PN<math>\Leftrightarrow</math>PLC Graph of Oil Tank Level Control.....</b>	<b>52</b>
<b>Figure 4-3(a) LD Showing the Conditions for Switching the Siren on. First Translation Rule .....</b>	<b>54</b>
<b>Figure 4-3(b) LD Showing How the Siren is Latched on. Second Translation Rule .....</b>	<b>55</b>
<b>Figure 4-3(c) LD Showing the Conditions for Switching the Siren off. Third Translation Rule .....</b>	<b>55</b>
<b>Figure 4-3(d) Final LD. An Exact Behavioural Equivalent to the PN<math>\Leftrightarrow</math>PLC Graph Shown in Figure 4-2(d) .....</b>	<b>56</b>
<b>Figure 4-4(a) Initial Marking Representing the Oil Sensor Switching on, i.e. Scenario 1, Event (a). Simulation Step 2.....</b>	<b>58</b>
<b>Figure 4-4(b) Evolution of Marking After Applying Simulation Step 3 to Figure 4-4(a) .....</b>	<b>58</b>
<b>Figure 4-4(c) Marking which Represents Scenario 1, Event (b) .....</b>	<b>60</b>
<b>Figure 4-4(d) Evolution of Marking After Applying Simulation Steps to Figure 4- 4(c) .....</b>	<b>60</b>
<b>Figure 4-4(e) Marking which Represents Scenario 1, Event (c).....</b>	<b>60</b>
<b>Figure 4-4(f) Evolution of Marking After Applying Simulation Steps to Figure 4- 4(e) .....</b>	<b>60</b>
<b>Figure 4-4(g) Marking which Represents Scenario 1, Event (d) .....</b>	<b>61</b>
<b>Figure 4-4(h) Evolution of Marking After Applying Simulation Steps to Figure 4- 4(g) .....</b>	<b>61</b>
<b>Figure 4-5(a) Marking which Represents Scenario 2, Event (a) .....</b>	<b>62</b>

<b>Figure 4-5(b) Evolution of Marking After Applying Simulation Steps to Figure 4-5(a)</b> .....	<b>62</b>
<b>Figure 4-5(c) Marking which Represents Scenario 2, Event (b)</b> .....	<b>62</b>
<b>Figure 4-5(d) Evolution of Marking After Applying Simulation Steps to Figure 4-5(c)</b> .....	<b>62</b>
<b>Figure 4-5(e) Marking which Represents Scenario 2, Event (c)</b> .....	<b>63</b>
<b>Figure 4-5(f) Evolution of Marking After Applying Simulation Steps to Figure 4-5(e)</b> .....	<b>63</b>
<b>Figure 4-5(g) Marking which Represents Scenario 2, Event (d)</b> .....	<b>63</b>
<b>Figure 4-5(h) Evolution of Marking After Applying Simulation Steps to Figure 4-5(g)</b> .....	<b>63</b>
<b>Figure 4-6 Petri Net Graph of Oil Tank Level Control</b> .....	<b>65</b>
<b>Figure 4-6(a) Evolution of the Marking After Simulating Scenario 1, Event (a)</b>	<b>66</b>
<b>Figure 4-6(b) Evolution of the Marking After Simulating Scenario 1, Event (b)</b>	<b>66</b>
<b>Figure 4-6(c) Evolution of the Marking After Simulating Scenario 1, Event (c)</b>	<b>67</b>
<b>Figure 4-6(d) Evolution of the Marking After Simulating Scenario 1, Event (d)</b>	<b>67</b>
<b>Figure 4-7(a) &amp;Figure 4-7(b) Two Possible LD Interpretations of the PN Graph Shown in Figure 4-6</b> .....	<b>70</b>
<b>Figure 4-8 Design of Oil Tank Level Control Using SFCs</b> .....	<b>71</b>
<b>Figure 4-9 LD Translation of the SFC in Figure 4-8</b> .....	<b>71</b>
<b>Figure 5-1 Initial Error Checks Performed Using a ‘pulse’</b> .....	<b>77</b>
<b>Figure 5-2 PN Graph Showing the Use of a ‘group signal’</b> .....	<b>78</b>
<b>Figure 5-3 PN Group Containing Initial Error Checks</b> .....	<b>80</b>
<b>Figure 5-4 PN Group Showing an Alternative Error Check</b> .....	<b>81</b>
<b>Figure 5-8 A PN Group Showing the Use of a Timer</b> .....	<b>82</b>
<b>Figure 5-11 Enable Start Miller PN Group</b> .....	<b>86</b>

<b>Figure 5-13 Ladder Diagram Translation of ‘Milling PN Graph’ .....</b>	<b>87</b>
<b>Figure 5-14 A Ladder Diagram Equivalent to Figure 5-13 .....</b>	<b>87</b>
<b>Figure 5-15 Enable Vice to Table PN Step .....</b>	<b>89</b>
<b>Figure 5-15L Step Diagram Translation of Figure 5-15 .....</b>	<b>91</b>
<b>Figure 5-24 Puma Station Error Status PN Graph .....</b>	<b>99</b>
<b>Figure 5-23 Puma Station Complete Status PN Graph.....</b>	<b>101</b>
<b>Figure 6-1 A PN Graph Showing the Unification of PN<math>\leftrightarrow</math>PLC with Gray’s Methodology .....</b>	<b>inside back cover</b>

## **Dedication**

I wish to dedicate this thesis to all my family, especially my father and my mother for their moral and financial support throughout my studies.

I also wish to dedicate this thesis to my partner for her support and patience, particularly throughout the writing-up period.

## **Acknowledgments**

I would like to express my sincere gratitude to my director of studies, Dr W M M Hales, for his excellent supervision and support. I would also like to thank my supervisor, Prof F Poole, Mrs J Grove, Mr G Cockerham and Prof E Lo for their support and advice.

Thanks also to all technicians involved, colleagues and friends.

## **Declaration**

I declare while registered as a candidate for the University's research degree, I have not been a registered candidate or enrolled student for another award of the University or other academic or professional organisation. I further declare that no material contained in this thesis has been used in any other submission for an academic award.

Aram Taholakian

# Chapter 1

## Introduction

### Contents

<b>1. Introduction.....</b>	<b>2</b>
<b>1.1 Automated Manufacture.....</b>	<b>2</b>
1.1.1 Flexible Manufacturing .....	2
<b>1.2 Distributed Control Systems.....</b>	<b>3</b>
<b>1.3 Development of Distributed Control Software .....</b>	<b>5</b>
1.3.1 Informal Methods for Designing DCSs.....	6
1.3.2 Formal Methods for Designing DCSs .....	6
<b>1.4 The Need for a Methodology for Developing DCSs.....</b>	<b>8</b>

## **1. Introduction**

This chapter gives a brief introduction to Distributed Control Systems (DCSs), mainly Flexible Manufacturing Cells, and discusses the current approaches used to develop the various parts of a DCS. The introduction then concludes by identifying a need for a methodology for developing a dependable DCS, such that the methodology is applicable to all aspects of the DCS, and not just the higher levels or the lower levels, in a formal but readable way.

### **1.1 Automated Manufacture**

Manufacturing industries employ a wide range of technologies in their processes to improve their competitive strength in the market place. Recent advancements in information technology have produced fully automated, efficient but complex manufacturing systems. Programmable automation tools, such as Programmable Logic Controllers (PLCs), Robots and computer numerical control (CNC) machines, are used to increase the efficiency and quality of manufacturing plants. Automated Manufacturing Systems offer numerous advantages such as short lead-times, low work-in-progress, reduced labour and hence reduced unit costs, and quick change-over times. They also provide flexibility which enables rapid responsiveness to market changes.

#### **1.1.1 Flexible Manufacturing**

The ability to meet customer demands and remain in touch with an ever increasing competitive market place [Ford 1991] has resulted in manufacturing industries focusing their attention on flexible manufacturing. Flexible Manufacturing Cells

(FMCs) have been well established for a number of years [Narahari and Viswanadham 1986, Huang and Chang 1992, Lu and Huang 1992, Chao *et al* 1992] and are used in batch manufacturing industries to improve the efficiency of production. They consist of a number of CNC machine tools, automated work/tool handling equipment and a control system to synchronise the operation of the machine tools and handling equipment. The integration and coordination of a group of FMCs produces a larger flexible manufacturing environment often referred to as a Flexible Manufacturing System (FMS) [Simpson *et al* 1982, Ayers 1988]. FMSs can be used to produce complex products, where individual components are made in various FMCs and then assembled in a Flexible Assembly System (FAS) [Williams and Lill 1987].

The safe and reliable operation of Flexible Manufacturing Cells is clearly desirable, if not essential for their effective use. However FMCs are complex systems the elements of which operate concurrently, and interact at irregular times depending upon the components to be produced. Therefore the development of a control system is not a trivial affair [Slack 1988, Duan and Kumara 1993]. Control systems not only consist of computer algorithms, but also computer hardware, communications protocols and cell monitoring equipment, such as sensors and transducers.

## **1.2 Distributed Control Systems**

Some automation tasks are simple to achieve, whereas others, for example FMCs, are far from simple, because they consist of a number of autonomous devices which operate concurrently, but also in synchronisation with each other.

The cost of designing and implementing automated manufacturing systems is high,



and is particularly so if the design is found to be unreliable during or after implementation.

Controlling complex concurrent systems is only feasible if the task of control is distributed amongst a number of devices (section 2.1.2). Flexible manufacturing operational strategies such as Distributed Control are now established practices in industry. Real time control is achieved by distributing the various functions of a control system which operate concurrently. It is essential however that these functions or elements communicate with each other to achieve complete synchronisation of the whole system. Such communication could be achieved by networking individual PCs, which each run sequential programs. However as the number of processors increases, the management overhead increases [Das and Fay Freund 1983].

The programming strategies and languages available for conventional microprocessors do not model a parallel architecture [Naghdy and Strickland 1989]. Apart from developing the program for each processor separately from others, the programmer has to introduce techniques for synchronisation and communication of one processor with others. However the software is directly dependent on the hardware. By increasing the number of processors, the complexity of the software design, development and testing will multiply. Expansion or modification of the operation of the hardware requires major modification of the software.

In recent years, systems specific processors and software have been developed to overcome the above mentioned problems. For example, the combination of the Transputer (a microprocessor, see section 2.2.1) and its programming language Occam (section 2.2.1.1) were developed together [Inmos 1989b], and are now used in

various industries for embedded systems, numerical analysis, image processing and Real-Time Distributed Control Systems.

### 1.3 Development of Distributed Control Software

At present the methods used to develop Distributed Control Systems tend to follow a sequence of steps:

- Statement of requirements of the DCS
- Statement of the functional specification of the DCS
- Design of the DCS to meet the specification
- Generation of the software code for the DCS
- Implementation of the software

It is widely acknowledged that this method has its drawbacks [Shatz and Wang 1987, Boehm 1988, Jelly and Gorton 1994]. The requirements and specification are usually written in English, the design may be depicted graphically or textually, and the software usually written in a high level language. This leads to difficulties in ensuring equivalence between the final implementation of the DCS and the initial requirements, because of the dissimilarity of techniques used to represent the steps, and the differences in the expertise of those who develop each step. The development of one step is a translation of the previous step, thus there is no reliable isomorphism between the steps. Practitioners attempt to ensure equivalence between the various steps by checking the equivalence of each step with its preceding step. Some methods used to do this are formalised, and some are ad hoc; none are totally satisfactory [Bloomfield and Froome 1991, Jelly and Gorton 1994].

To improve on this method of developing DCSs, equivalence of representation is required at each step. The ideal way to achieve this would be to develop a methodology which uses the same language, one that all parties could understand, at each stage of the development, and could be verified against the system requirements prior to implementation.

### **1.3.1 Informal Methods for Designing DCSs**

Distributed Control Systems are often categorised into more than one level of control. For example the control of an FMC may be divided into several levels ranging from the Programmable Logic Controller (PLC) operated sensor and actuator control level, to a high decision making level, often referred to as the Cell Controller (CC) level.

A considerable amount of research into the different levels of DCSs has been recently carried out by various parties. For example, many have attempted to develop alternative PLC program representations (discussed further in Chapter 2, section 2.4). For the most part, these attempts have focused, not only on the lowest level of a DCS, but also on a particular aspect of the software development process, mainly modelling. There have also been attempts to use Petri Nets (PNs) to design PLC programs, but virtually all the resulting PN models pay scant regard to the readability [Taholokian and Hales 1995], which severely restricts their practical applications: a design which is unclear to read is difficult to check for correctness, difficult to maintain and difficult to update, i.e. it is unreliable.

### **1.3.2 Formal Methods for Designing DCSs**

Extensive research has also been carried out into the higher levels of DCSs. For over a decade, the System Integration Group at Loughborough University led by R. H.

Weston has researched into areas such as Computer Integrated Manufacture (CIM) and has produced computerised systems for developing CIM systems [Weston 1991, Weston 1993], namely CIM-BIOSYS, which is an infrastructure designed to integrate the various software applications needed to achieve highly autonomous and flexible CIM systems. The group has also investigated the behavioural modelling of CIM systems using techniques such as CIM-OSA and Stochastic Timed Petri Nets [ESPRIT/AMICE 1991, Aguiar and Weston 1993], and of late has investigated the use of PNs to design code for DCSs [Ariffin *et al* 1995].

A large amount of research has been carried out by various parties into the use of Petri Nets for modelling and performance evaluation of DCSs: [Kamath and Viswanadham 1986, Huang and Chang 1992, Viswanadham and Narahari 1992, Barkaoui and Ben Abdallah 1993, Hilal and Ladet 1993, Reddy *et al* 1993]. Numerous publications referring to the use of PNs for the detection of faults, specifically “deadlock”, are also widely accessible [Murata and Shenker 1989, Viswanadham *et al* 1990, Ezpeleta *et al* 1995].

A group of academics and researchers at Sheffield University has worked on automatically producing software from a system specification, for implementation on a parallel platform consisting of Transputers and C40s [Bass *et al* 1994]. Others have also investigated the implications of Transputers and Parallel Processing for DCSs [Draper and Holding 1989, Tudruj 1992, Lau and Seet 1993, Sundaram and Narahari 1993, Moore and O’Donoghue 1994].

At the School of Engineering (SOE), Sheffield Hallam University, there is ongoing research into the development of dependable DCSs for Flexible Manufacturing [Hales *et al* 1993]. One outcome of this research was a Ph.D. Thesis [Gray 1995], which

reports on a PN-Occam based methodology for developing DCSs (Chapter 2, section 2.2.3). Gray's methodology pays strict attention to the readability and comprehensibility of the design. Gray claims that the design translates, with exact equivalence, into the Occam programming language to run on Transputers. He also claims that it is expandable and avoids control errors such as "deadlock" from being introduced into the code. However Gray's methodology has only been applied to Transputers and Occam, which is only part of an overall DCS. More than often DCSs in manufacturing industries consist of more than one type of controller. The PLC is widely used in industry (section 2.3), in conjunction with other controllers such as PCs and Transputers, to achieve Distributed Control. There is no evidence whether his methodology is applicable to such an overall DCS.

#### **1.4 The Need for a Methodology for Developing DCSs**

From the preceding discussion it is apparent that the current methods for developing distributed control software do not ensure dependability of the system. Although work has been done on designing parts of DCSs, there is no methodology for ensuring that an overall DCS is dependable and meets the requirements.

A need has been identified for a methodology for designing DCSs such that:

- the methodology is easy to understand and of practical use in industry.
- the methodology is applicable to all aspects of a DCS and not just the higher levels or the lower levels.
- the design of the DCS fully meets the requirements of the system.

- the design is represented in a readable format which enables parties other than just the original designer to understand the design and thus reliably check for correctness, reliably maintain and expand the system.
- the designer is guided towards designing an error free control system prior to implementation.
- the design is modular to allow flexibility of the DCS.
- the design is graphical but formal and supports simulation thus eliminating the need to translate the design into an independent simulation software to analyse.
- the design is equivalent to the control algorithm, i.e. there are formal rules for translating the design into the final control code to ensure that the two operate in exactly the same way. Thus if the design is correct, then the control code will be correct.

# Chapter 2

## Background

### Contents

<b>2. Background .....</b>	<b>12</b>
<b>2.1 The FMC at the SOE.....</b>	<b>12</b>
2.1.1 Levels of Control .....	14
2.1.2 Reasons for Distributing Control.....	16
<b>2.2 Gray's Petri Net - Occam Methodology .....</b>	<b>17</b>
2.2.1 Transputers and Occam .....	18
2.2.1.1 Occam: Language Definition.....	18
2.2.2 Introduction to Petri Nets.....	19
2.2.2.1 High Level Petri Nets .....	21
2.2.3 The Motivation and Objectives of Gray's Methodology.....	22
2.2.4 Gray's Methodology Applied to the FMC.....	24
2.2.5 Advantages, Disadvantages and Claims of Gray's Methodology .....	26
<b>2.3 Programmable Logic Controllers.....</b>	<b>26</b>
2.3.1 Available PLC Programming Tools and Techniques .....	27
2.3.1.1 Instruction List.....	27
2.3.1.2 Structured Text .....	28
2.3.1.3 Function Block Diagram.....	28
2.3.1.4 Sequential Function Chart .....	28
2.3.1.5 Ladder Diagram .....	29
2.3.1.5.1 Step Ladder Diagram.....	30
2.3.2 Review of Existing PLC Languages.....	30

<b>2.4 PNs and PLCs: Previous Work .....</b>	<b>32</b>
2.4.1 PNs for Designing and Modelling DCSs Containing PLCs .....	32
2.4.2 PNs for Designing PLC Programs .....	32
<b>2.5 The Aim and Objectives of the Ph.D. Research .....</b>	<b>34</b>



## 2. Background

This chapter describes the Flexible Manufacturing Cell at the School of Engineering which was used as the test bed for the reported research project. It discusses the control and the communication between the various levels of the control. The chapter also introduces Petri Nets and a Petri Net - Occam methodology, which was the outcome of previous research carried out at Sheffield Hallam University's School of Engineering. The incompleteness of previous approaches, including the Petri Net - Occam methodology, are discussed, and in particular their inapplicability to low level PLC control. The PLC is an integral part of an FMC, and in this chapter the inadequacies of existing PLC programming strategies are also discussed. A need for a methodology for developing PLC based DCSs is identified. Finally the aims and the objectives of the PhD research are stated.

### 2.1 The FMC at the School of Engineering

Gray's Petri Net - Occam methodology was the initial motivation for this research project. The Flexible Manufacturing Cell (FMC) at the School of Engineering (SOE), Sheffield Hallam University, was chosen for the development and testing of both Gray's methodology and the reported Ph.D. research. It is therefore appropriate to describe the FMC prior to explaining Gray's methodology.

The FMC consists of a CNC lathe, CNC milling machine, a conveyor track, and three work stations (WS) as follows:

- Puma WS for loading and unloading raw material and finished parts.
- Lathe WS for loading and unloading the lathe.



### 2.1.1 Levels of Control

The equipment used to perform the logical control of all the elements within the FMC is distributed between Transputers and PLCs (Programmable Logic Controllers), and divided into three levels of control, as shown in Figure 2-2.

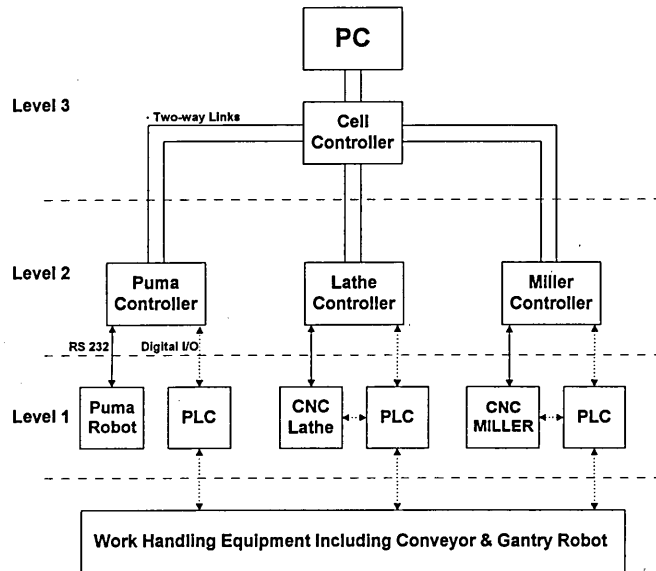


Figure 2-2 Levels of Control of the FMC

#### Level 1

The simple sequential control of primitive devices, such as actuators and sensors, is carried out in Level 1 using PLCs.

One PLC is devoted to the Puma WS. The opening of the dogs on the conveyor, closing them when the pallets have moved on, and the monitoring of the appropriate sensors at the Puma WS, to determine the arrival/departure of pallets at/from the station, is carried out through this PLC.

A second PLC is devoted to the Miller WS. It monitors the appropriate sensors at the Miller WS to determine the arrival of the pallets, activates the loading/unloading

cylinder, the milling fixture clamp and instructs the miller to start or stop machining.

The PLC also monitors the machining status of the Miller.

A third PLC is used to control and monitor the Lathe WS and the gantry robot, synchronising the opening and shutting of the lathe door with the entry and exit of the robot arm whilst loading and unloading workpieces. The execution of the “Start Machining” command to the Lathe, and the monitoring of the machining status is also carried out through this PLC.

### Level 2

The control of the machine tools, robots and the PLCs conducting Level 1 control is achieved in this level.

The Puma Transputer, for example, instructs the Puma robot to execute the program “Load Cell”, which is used to place a blank from the raw material stock in a vice on a pallet. It also instructs the PLC at the Puma WS to convey pallets between work stations.

The Miller Transputer, for example, instructs the PLC at the Miller WS to transfer the vice into the fixture on the milling table and clamp it in place.

The Lathe Transputer instructs the PLC at the Lathe WS to operate the gantry robot, to load or unload the lathe, and to monitor the arrival of pallets and the opening and closing of their vice accordingly.

### Level 3

Level 3 is conducted by the Cell Controller. The Cell Controller's task is to ensure that a given schedule of parts are produced by the cell. In order to do this, it passes

instructions to the Level 2 controllers to load and unload the machine tools, to start machining cycles and convey parts round the cell.

### **2.1.2 Reasons for Distributing Control**

Having established that the control of the School of Engineering's cell is achieved by distributed control, it is important to explain the reasons behind such a design. Why, for instance, is the whole cell not controlled by only one computer.

#### Two essential reasons:

1. To reduce the complexity of the control algorithms/computer programs.

Even with the small FMC at the SOE., the number of sensors that need monitoring, and the number of actuators which need operating is very large. Since the various parts of the cell operate concurrently, the control of such an environment by one sequential algorithm becomes virtually impossible, and certainly inefficient. Changes within the cell would mean redesigning the complete control algorithm, hence making future growth a very difficult task. An example of concurrency within the cell could be, when a part is to be unloaded from the conveyor by the Puma robot, also a part is to be loaded onto the miller and whilst this is being carried out, the lathe finishes machining and requires its workpiece to be removed.

2. To achieve the required communications necessary to operate the machine tools and work handling equipment.

There are several types of data that have to be communicated within the cell. For example, whether a sensor is on or off, which program the Puma is to run, what component is to arrive at the Miller. This data is communicated in different ways depending on the complexity of the data. For instance, a sensor's data may be on or

off (24 or 0 volts) and requires only two wires to transmit its data. However, more complex messages have to be passed between station controllers and the machine tools, e.g. to instruct the Puma to run a certain program requires the Puma station controller to transmit the command "EXEC LOADPART", and the Puma to reply "PART LOADED". RS 232C is used to handle this duplex communication.

Messages which pass between the Cell Controller and the station controllers may be transmitted simultaneously, therefore a Local Area Network (LAN) having a multiple access communication protocol is used. A single control algorithm to handle all these communications in Real-Time would be difficult or perhaps impossible to write.

## 2.2 Gray's Petri Net - Occam Methodology

Over the years, Petri Nets (section 2.2.2) have been used in modelling safety critical systems to reduce faults in the system by removing them before use [Sahraoui *et al* 1987, Leveson and Stolzy 1987, Viswanadham and Johnson 1988, Reddy *et al* 1993]. FMCs are a particular case of safety critical systems in that, although automated they work in conjunction with humans.

Gray conducted research to show that Occam and Transputers could considerably simplify the design of DCSs for FMCs. He chose to use Petri Nets as the formal method for developing DCSs, but soon found that Petri Net graphs can become very complex (Figure 2-3, inside back cover). The graph is not readable and difficult to follow.

For these reasons and inspired by the capabilities of Occam and Petri Nets, Gray developed a methodology for producing dependable distributed control software for flexible manufacturing.

It is important however to discuss Transputers, Occam and Petri Nets, in more detail, for a better understanding of the methodology.

### **2.2.1 Transputers and Occam**

The combination of the Inmos developed hardware and software called Transputer and Occam respectively has now been recognised as a solution to the problem of programming concurrent systems.

A Transputer is a microcomputer with its own local memory, and with communication links for connecting one Transputer to another Transputer [Inmos Ltd. 1990]. The protocols for communications between Transputers is built into the hardware of the Transputer chip. It can be used in a single processor system, or in networks to build high performance parallel architectures. By linking processors together, a linear increase in data processing capacity can be achieved, as opposed to the limited processing capacity of typical multi-processor control systems (MPCS). A MPCS can only be increased to a certain limit before experiencing a drop off in effective computing power [De Gaspari 1992].

#### **2.2.1.1 Occam: Language Definition**

Occam simplifies the writing of concurrent programs by taking most of the burden of synchronisation away from the programmer [Inmos Ltd. 1989]. Occam uses channels for communicating values and does not distinguish between two processes (programs) running concurrently on different computers, or concurrently on the same computer. However channels are one-way only, and therefore two are needed for a two-way communication.

Although Occam provides synchronised communication, the programmer is still left with the responsibility of avoiding “deadlock”, i.e. a process waiting for communication that will never arrive, for it is prepared to do so forever . For the non professional programmer, or rather the professional engineer and system designer, avoiding deadlock, in even a relatively simple FMC such as the one in the School of Engineering, is difficult.

### 2.2.2 Introduction to Petri Nets

Petri Nets are a tool for the study of systems, and a graphical representation of systems [Peterson 1981]. In 1962 Carl Adam Petri, a German mathematician defined Petri Nets as a mathematical modelling tool for describing relations between conditions and events, whether sequential or concurrent. The basics of Petri Net graphs are as follows:

#### Places, Transitions and Arcs

A **place** is represented by a circle and a **transition** is represented by a bar or a box. Places and transitions are connected by **arcs** (Figure 2-4(a), below). An arc is directed and connects either a place to a transition or a transition to a place. Arcs directed from a place to a transition define the place to be an input of the transition. Arcs directed from a transition to a place define the place to be an output of the transition. Thus in Figure 2-4(a) A, B and C are **input places** to transition **t1**, and D and E are **output places** of **t1**.

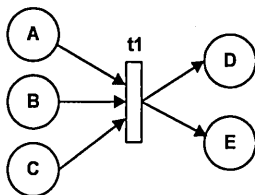


Figure 2-4(a) A Simple Petri Net



Marking

**Marking** is the assignment of **tokens** or **marks** to the places of a PN (Figure 2-4(b), below). A token is represented by a dot in a place. The marking at a certain moment defines the state of the system described by the PN. In Figure 2-4(b) the input places A, B and C are said to be active. Places can also be ‘bounded’, which means there is a limit on the maximum number of tokens a place can hold. e.g. a place bounded to one token can only contain one token at any one time.

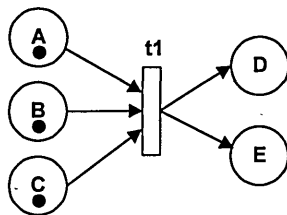


Figure 2-4(b) A Marked Petri Net

Firing of Transitions

A transition can only be **fired** if each of its input places has at least one token. The transition is then said to be **fireable** or **enabled** (Figure 2-4(b), above). Firing of a transition consists of withdrawing a token from each of its input places and adding a token to each of its output places (Figure 2-4(c), below). The firing of a transition is indivisible (has zero duration), except when considering Timed or Synchronised PNs.

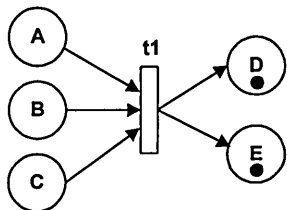
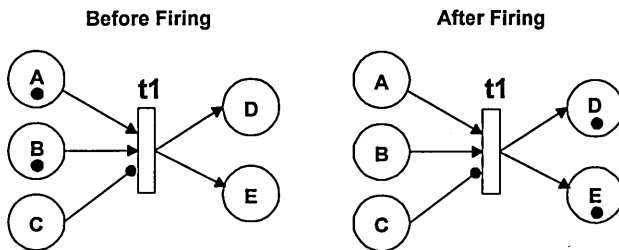


Figure 2-4(c) Marking After the Firing of the Transition

### Inhibitor Arcs

An **inhibitor** arc is a directed arc which joins a place to a transition. However its end is marked with a small circle (Figure 2-4(d), below). The inhibitor arc between the input place C and the transition means that the transition is only enabled if the place C does not contain a token, as shown in Figure 2-4(d).



**Figure 2-4(d) Inhibitor Arc PN Graph Showing the Marking Before and After the Firing of the Transition**

The above is not a definitive description of PN's, but is sufficient for understanding the methodology.

#### **2.2.2.1 High Level Petri Nets**

High level Petri Nets are folded versions of the general Petri Nets described above.

Unfolding a high level Petri Net produces a set of places, where one place was, and a set of transitions, where one transition was. Folding a general Petri Net into a high level one is the reverse of this. The way in which general and high level Petri Nets are read is different. In reading general Petri Nets more emphasis is given to places and transition rather than tokens and arcs; "WIZIWIG, what you see is what you get". In high level Petri Nets more emphasis is given to reading tokens and arcs rather than places and transitions, due to some, if not most, of the information being folded away.

The major types of high level Petri Nets are Predicate-Transition Nets [Genrich 1987], Coloured Petri Nets [Jensen 1991] and Hierarchical Petri Nets [Gracanin 1994].

### 2.2.3 The Motivation and Objectives of Gray's Methodology

Gray highlighted the need of a novel methodology in three stages as follows:

- a) The needs of a Transputer based FMC
- b) The needs of an Occam based methodology
- c) The use of Petri Nets with Occam

A verification of these needs and a full criticism of current use of Petri Nets and Occam is given in his thesis [Gray 1995]. Gray's conclusion of these needs are drawn from the suitability of Transputers and Occam as the hardware and software for the control of a DCS, the lack of sufficient guidance in the development of flexible and dependable Occam code, and the applicability of Petri Nets to DCSs and their popularity in the manufacturing field.

In addition to the references to the lack of adequate guidance provided by other existing Petri Net-Occam techniques, Gray highlights other significant disadvantages associated with current practices, mainly the following:

1. The most common approach of developing Petri Net models is informal, i.e. produce a Petri Net graph and analyse it. When the model is found to be incorrect, it is then modified and re-analysed. This may be repeated several times before an accurate model is achieved.
2. High level Petri Net graphs are often used because they are graphically more concise than their equivalent general Petri Net graphs, but the information held within them is often more difficult to absorb because it is folded away.

3. Even general Petri Net graphs are almost always found to be very difficult to read, and thus understand, because they are drawn in an unstructured fashion (Figure 2-3, inside back cover).
4. Occam code refinement is tedious. The conventional approach of developing Occam code is to draft its basic structure using DFDs or PNs, produce the Occam code and try and compile it. If and when it does not compile, it is then modified and re-compiled. This process is repeated until successful.
5. Using formal design techniques will help produce reliable and safe control systems. However, a tedious development process is often involved in using formal methods and mathematicians are required to design and verify the control programs. This is costly both in time and in wages, and therefore industry is reluctant to use such methods.

Having discussed the problems associated with existing DCS development methods and techniques, Gray concludes that there is a need for a methodology which makes better use of Petri Nets and Occam. The aims and objectives of his novel methodology can be summarised as follows:

- to produce the specification of the DCS directly from the manufacturing requirements.
- to produce the specification in a graphical but formal way, which will be comprehensible during all stages of the development life cycle of the FMC, i.e. specification, design, simulation, coding, implementation and maintenance.
- to obtain equivalence between the model and the code by exploiting the similarities of PNs and Occam.

- to incorporate “deadlock” prevention into the methodology by having a unidirectional communication protocol.
- to produce dependable Occam code to run on a network of Transputers for the control of an FMC.

#### **2.2.4 Gray’s Methodology Applied to the FMC**

To achieve the aims and objectives listed above in section 2.2.3, Gray’s methodology consists of four design steps and employs certain techniques which play a key role in the development of dependable DCSs. For example, “deadlock” is avoided by employing a technique that Gray calls “output-work-backwards” and a uni-directional flow of information between the various controllers of the FMC. Where two way communication is required between controllers, for example between the Cell Controller and the Status Handler (Figure 2-5, inside back cover), then a client-server relationship between the two is employed (discussed below).

The four steps of the methodology are:

*Step 1* Identify concurrent and sequential operations.

*Step 2* Produce a Petri Net graph for each controller.

*Step 3* Combine the controller Petri Net graphs.

*Step 4* Translate the Petri Net graphs into Occam code.

Each step consists of one or more tasks, as described below.

The task in step 1 is to analyse the overall concurrency of the FMC. Some operations, such as the lathe and the miller machining at the same time, are stated in the requirements of an FMC. Other concurrent operations are quite obvious because they

are independent, for example lathe machining and the conveyor indexing. Most sequential operations are also obvious, e.g. loading the lathe before machining, or loading a part into a pallet before indexing (transporting) the part to the lathe. The need for a Cell Controller (CC) and a Status Handler (SH) is also identified during this step. The hierarchical nature of the manufacturing control system and the master-slave method of communication often used in Transputer networks [Gray 1995] prompts the need for a cell supervisor or master to the slave work station controllers. The Status Handler is slave to the work station controllers but operates as a server to the client Cell Controller. The task of the Status Handler is to prevent “deadlock” by gathering feedback from the work station controllers, and acts as a buffer between the Cell Controller and the rest of the cell. The communication between the Status Handler and the Cell Controller is controlled, to avoid “deadlock”. The Status Handler will not communicate with the Cell Controller unless the Cell Controller requests an update for the status of the cell.

The tasks involved in step 2 are to examine the requirements of the controllers and design the logic of the individual controllers. Petri Net graphs for each controller are produced, which clearly list the inputs to the controllers, the logical operation of the controllers and the outputs from the controllers. Where the inputs come from and where the outputs go to, are also clearly stated during this step. Gray prescribes a specific format for drafting PN graphs, to ensure that they are readable.

Step 3 is concerned with integrating the individual controller PN graphs to produce an overall PN graph, which is the specification of the DCS. Managing the communication between controllers in a closed loop of control, to prevent “deadlock”, is also carried out during this step.

The task in step 4 is that of translating the Petri Net graph into Occam. The rules of the methodology ensure that the overall PN graph is equivalent to the Occam constructs, making the translation process reliable, efficient and relatively simple.

By following these steps, the overall PN graph for the control of the FMC described in section 2.1 was produced (Figure 2-5, inside back cover).

### **2.2.5 Advantages, Disadvantages and Claims of Gray's Methodology**

Gray's objectives for his methodology are worthwhile; he achieves a readable, modular and dependable design which can be checked for correctness, and translated reliably into control software. However, his methodology could not be used to design a complete DCS which consists of controllers other than Transputers. In particular, Gray does not consider PLCs which are very commonly used in industry worldwide. Indeed PLCs are an integral part of the FMC chosen by Gray for the development of his Petri Net - Occam methodology. Nevertheless, Gray claims that his methodology produces a design which can be both reliably updated and expanded. These claims were investigated as part of the research programme and are discussed in Chapter 6.

## **2.3 Programmable Logic Controllers**

The PLC is a microcomputer which evolved from the conventional computers of the late 1960s and early 1970s [Webb 1992]. Over the last twenty years the PLC has become an integral part of industrial control systems due to its rugged structure and high processing capabilities. The basic structure of the PLC consists of a Central Processing Unit (CPU) and input/output modules or terminals. Increased technology has made it possible to cram more functions such as numerous relays, timers and counters into smaller and relatively inexpensive PLCs. Input signals to the PLC and

output signal from the PLC (I/Os) can be either 24 volts or 0 volts (section 2.3.1.5), and are used to control a wide range of devices such as solenoids, actuators and robots.

### **2.3.1 Available PLC Programming Tools and Techniques**

One of the major restrictions to new PLC concepts being applied in industry is the difficulty to change the desire of some users to use Ladders Diagrams [Webb 1992]. In fact, the Ladder Diagram approach of PLC programming, based on conventional 'relay logic', still remains the most commonly used technique employed by the vast majority of industries in the UK and the USA [Jafari and Boucher 1994]. A decade ago S. M. Cotter [Cotter and Woodward 1986] wrote: "Like it or not, then, the ladder diagram is likely to be with us for some years yet." This still holds true today despite the definition of other PLC languages in the IEC 1131-3 standard [IEC 1992] by the International Electrotechnical Commission. The standard suggests five languages for programming PLCs; Instruction List (IL), Structured Text (ST), Function Block Diagram (FBD), Sequential Function Chart (SFC) and finally Ladder Diagram (LD). However none of the five suggested languages are design methodologies, nor are they widely accepted or tested, with the exception of Ladder Diagrams.

#### **2.3.1.1 Instruction List**

Instruction List is a textual low level language, similar to an assembler language, without support for structuring. An Instruction List consists of a sequence of instructions or operations, with each instruction beginning on a new line. It is useful for smaller applications or for optimising parts of an application.



### **2.3.1.2 Structured Text**

Structured Text, as its name implies, is a structured textual high level language, which has a similar syntax to the programming language PASCAL. Structured Text can be used to create application specific function blocks. Complex statements involving variables which represent a wide range of data types (including analogue and digital) can be expressed using ST. It also supports types of data specific to batch processing applications, such as time, date and duration. In addition, ST supports iteration loops such as REPEAT UNTIL, conditional execution using IF-THEN-ELSE constructs, and Math and Trig functions such as SQRT and SIN.

### **2.3.1.3 Function Block Diagram**

This is a graphical language with some limited support for hierarchical structures. FBDs allow program elements which appear as function blocks to be connected to one another in a manner similar to a circuit diagram. A function block is a program organisation unit, which, when executed, yields one or more values. The FBD is suitable for applications which involve the flow of information or data between control components.

### **2.3.1.4 Sequential Function Chart**

SFCs are a graphical high level language which can be used to structure PLC code for the purpose of performing sequential control functions. The basic elements of the SFC are a set of steps and transitions, interconnected by directed links. Associated with each step is a set of actions, and with each transition a transition condition. Since SFC elements require storage of state information, the only program organisation units which can be structured using these elements are function blocks and programs.

A program is defined in the standard as a “logical assembly of all the programming language elements and constructs necessary for the intended signal processing required for the control of a machine or process by a PLC”.

### **2.3.1.5 Ladder Diagram**

The LD is based on conventional 'relay logic'. The required actions of the program are represented sequentially by lines, or rungs, on the ladder diagram [Webb 1992]. The input signals to the PLC are marked on the left hand side of the diagram. These signals may be 'ON' or 'OFF' (24V or 0V) and form the conditions for the output signals (instructions from the PLC) which are marked on the right hand side of the ladder. The physical input and output contacts, I/Os, of the PLC are clearly marked on the ladder (X0, X1, X400, X401, etc., and Y30, Y31, Y430, Y431, etc.). The internal relays of the PLC are also marked as M numbers (M100, M103, etc.). A 'pulse' is an internal relay which, if switched on, remains on for only the first scan of the LD. The use of a 'pulse' is ideal for carrying out safety checks on the initial starting conditions of the system, e.g. checking if a robot is in the safe position before instructing it to perform an operation. In addition to internal relays and I/O contacts, the PLC is also equipped with timers. The most common timing function is the 'delay on' timer (section 5.1.2), which is the basic function. There are also several other derived timing functions, such as 'delay off', 'interval pulse' and 'multiple pulse'.

The order in which the program switches outputs on or off is dependent on the logic, and not on the order of the rungs on the ladder diagram (discussed in more detail in section 4.3). Therefore the input conditions must be carefully determined for each output instruction, to avoid complications later in the program. However, the risk of

programming errors is high and verifying whether the program meets the specification is difficult [Nagao 1993]. This is because Ladder Diagrams do not provide adequate simulation support or analytical capabilities. Furthermore, the maintaining or updating of the software, by anyone other than the original programmer, is both difficult and time consuming.

#### **2.3.1.5.1 Step Ladder Diagram**

As the name suggests, the use of Step Ladders [Wardman 1994] allows a complex PLC program to be divided into a number of steps, depending on the complexity of the operations carried out by the PLC.

Each operation is given a step number at the beginning of the Step ladder and an input address which forms the condition for that particular Step Ladder to be 'Set' (to start). The internal 'Step relay' assigned for that Step (e.g. S601) is 'Set' once at the beginning of the Step Ladder. The Step Ladder is 'Reset', by resetting the 'Step relay', only if the required operation is carried out successfully. The layout of the required actions in between the 'Set' and the 'Reset' lines is similar to that of a standard Ladder diagram. However repeated instructions carried out by the PLC do not cause programming complications as they can be grouped in separate Step Ladders. Therefore the programming of the PLC is simplified and the risk of errors reduced.

#### **2.3.2 Review of Existing PLC Languages**

The DTI/SERC sponsored collaborative project [SEMSPLC 1992-95] entitled Software Engineering Methods for Safe PLCs (SEMSPLC) has produced a Code of Practice for developing safe PLC application software [Clarke *et al* 1995]. In this

Code of Practice the IEC 1131-3 standard was criticised for suggesting a wide range of techniques with no support to the engineer as to how to choose from the wide range of techniques and to fit them into applications to achieve safety levels. SFCs, FBDs and LDs have also been criticised for their restrictions and lack of analytical capabilities [Jafari and Boucher 1994, Nagao 1993]. A detailed constructive criticism of the IEC 1131-3 standard was also published by W. A Halang [Halang 1989]. Some of the mentioned drawbacks of the languages were the poor timing control features available and the fact that the duration of the single process states is implementation dependent, and not under program control.

Sequential Function Charts are based on Grafset [Courvoisier *et al* 1983] which evolved from the work of several French research teams, working in areas such as flow charts, simulation of logic systems and state diagrams. Firstly a French standard, in 1987 it became an international standard, with slight adaptations [David and Alla 1992]. Although the structure of the SFC is tidy, with the majority of the control detail hidden within the action blocks, it tends towards redundancy in manipulating low level devices, such as sensors and actuators [Sato *et al* 1992].

It is claimed in the IEC 1131-3 standard that all five languages are equivalent.

However, no verification of this is given in the standard. Some proprietary software can translate Grafset diagrams, which are nominally equivalent to SFCs, into Ladder Diagrams. However, they produce very inefficient and lengthy Ladder Diagrams as shown in section 4.4. Furthermore, the translation rules from the SFC to LD are far from simple.

## **2.4 PNs and PLCs: Previous Work**

This section is included for the sake of completeness . It briefly reviews previous work where Petri Nets have been used in conjunction with PLCs. It shows the general incompleteness of previous approaches in meeting the objectives of designing dependable DCSs. This previous work can be categorised into two subject areas, as follows:

1. PNs for designing and modelling PLC based DCSs.
2. PNs for designing PLC programs.

### **2.4.1 PNs for Designing and Modelling DCSs Containing PLCs**

Various people have used PNs for designing and modelling DCSs that contain PLCs. For the most part, Petri Nets have been used to model the overall system, and not the control software. The various parts of systems, for example an assembly plant, have been represented by PN graphs to investigate the flow of control between controllers at a high level [Farrington and Billington 1996], but the lower level control software has not been included. The control of low level devices, often by PLCs, forms a critical part of manufacturing systems, and must be part of any design of a system. Thus previous work in this area [Viswanadham and Johnson 1988, Borusan 1993, Huang and Chang 1992, Gray 1995] is incomplete because they do not include the PLC control logic in the overall design model of the DCS.

### **2.4.2 PNs for Designing PLC Programs**

Research into the development of alternative PLC programming methods is still being carried out, despite the standardisation of five existing languages by the IEC.

The literature presented in this section refers to past research into the use of Petri Nets specifically for designing PLC programs. The motivation for this past work appears to be to overcome the problems encountered when using Ladder Diagrams to design and code systems (section 2.3.2), in an attempt to make the design process more reliable by using the modelling capabilities of PNs. However it is not apparent that any of the attempts are an improvement on Ladder Diagrams. The readability of the design is in no way an improvement from LDs, and in several cases a regression, due to the lack of structure of the PNs [Taholakian and Hales 1995].

Some have demonstrated their approach using only a simple sequential program and claim that it can be simply applied to any system [Henry and Webb 1988, Green 1990, Badri and Henry 1992].

Others have not considered translation of their PN into any of the languages of the IEC, and therefore have no proof if it works [Hasagawa *et al* 1990, Farrington and Billington 1996].

Readability of the design has been addressed by some who use high level PNs.

However, the detail of the design is hidden away and when unfolded, this detail leads to non-readability [Hasagawa *et al* 1990, Pardey *et al* 1994].

Most have applied the general principles of PNs to the PLC program by showing that when a transition fires, its input place loses its token and the output place gains a token simultaneously [Hasagawa *et al* 1990, Satoh *et al* 1992, Badri and Henry 1992, Jafari and Boucher 1994]. However, it is not shown how a PLC program can reliably perform the same task. This very important point is discussed in more detail in Chapter 4.

No-one has produced a methodology which enables a low level graphical PLC program to be produced in a structured manner, i.e. which clearly shows all the inputs to the PLC, the internal logic of the PLC, and the outputs from the PLC, stating exactly what operations are intended by those outputs by using readable names and not code, which is a significant advantage of Ladder Diagrams.

## 2.5 The Aim and Objectives of the Ph.D. Research

The aim of the research was to devise a methodology for developing dependable overall DCSs. In order to meet the needs identified and to alleviate many of the problems mentioned in Chapter 1, and also section 2.4, specific objectives were identified:-

- a) to devise a graphical and textual methodology for representing both the specification and design of the control in an integrated format such that:
  - the specification and design are identical.
  - the design can be transformed, with exact equivalence, into software.
  - the design can be modelled and proved to be correct: thus if the design is correct, the software will be correct.
  - the design can be readily understood by all, thus integrating the role and skills of the specifier, designer, software writer and user of the system, and also enabling reliable modifications, updates and maintenance to be made to the system.
- b) to devise rules which will ensure that:
  - the system is designed to be dependable. For example, rules will be formulated to prevent the designer from developing an uncodable design, and rule out the

possibility of “deadlock” occurring.

- errors within the system are either avoided or eliminated at the design stage.
- proving the correctness of the design, through modelling, yields reliable results.

Gray’s approach was considered to be the most favourable compared to the other approaches listed in the literature, due to its readability, reliability, modularity and his claims of expandability. However, Gray’s methodology was found to be incomplete because it only considers Transputers and Occam and not the overall DCS, which may include other controllers such as PLCs. The application of Gray’s methodology to PLC programming was conducted to test his claims of expandability and is discussed in Chapter 3.



## Chapter 3

### The Application of Gray's Methodology to an Overall DCS

#### Contents

<b>3. The Application of Gray's Methodology to an Overall DCS.....</b>	<b>37</b>
<b>3.1 Gray's Methodology Applied to a Simple PLC Control Task.....</b>	<b>37</b>
3.1.1 Design and Simulation.....	38
3.1.2 Translation of PN into LDs.....	42
<b>3.2 Discussion .....</b>	<b>43</b>

### **3. The Application of Gray's Methodology to an Overall DCS**

This chapter discusses Gray's claims that his Petri Net - Occam methodology can be expanded to accommodate all parts of a DCS. Gray's claims of the expandability of his methodology were investigated at the early stages of the research program, by attempting to apply the methodology to develop the control software for the low level PLCs. His methodology was found to be inapplicable to designing PLC programs.

The reasons for this are briefly explained and discussed in this chapter. The claims of modularity and expandability were also tested and are discussed in Chapter 6.

#### **3.1 Gray's Methodology Applied to a Simple PLC Control Task**

Gray applied his methodology to the School of Engineering's FMC, and developed an overall PN graph of the Distributed Control System (Figure 2-5, inside back cover).

Although this is readable, the concurrency within the system is clearly shown as are the control logic and communications, it is not a complete design of the overall DCS.

This is because the control carried out by the PLCs is not included in the overall design.

Gray's methodology was tested on concurrent, as well as sequential, PLC control systems. The example chosen in this section is not part of the School's DCS.

However, it clearly demonstrates that by applying Gray's methodology to PLC programming, the resulting PN graph does not model the correct logic.

##### **Example: Oil Tank Level**

The operational requirements for the system shown below in Figure 3-1 are; when the oil level drops, the oil sensor switches on and a siren is sounded. By pressing and

releasing a spring return push button the siren is switched off and a lamp is switched on. The lamp is switched off only if oil is added to the tank.

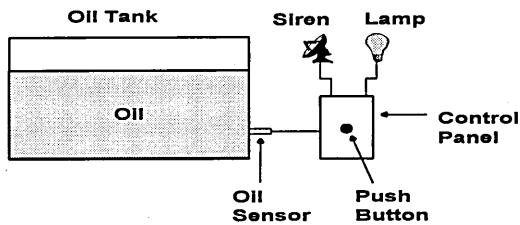


Figure 3-1 Oil Tank Level

### 3.1.1 Design and Simulation

Gray's methodology was applied to the control problem, Figure 3-1, and the PN graph shown in Figure 3-2 was developed, directly from the requirements.

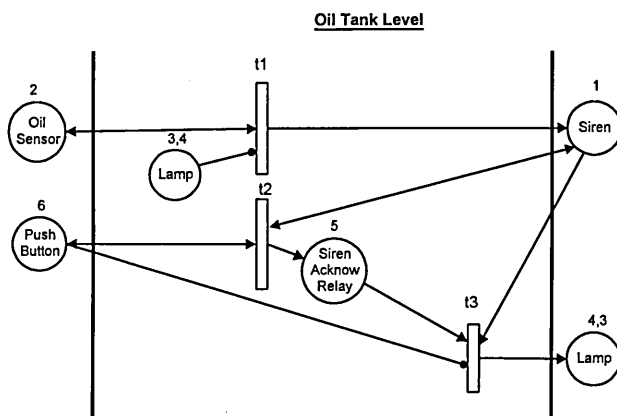


Figure 3-2 PN Graph of Oil Tank Level Control Produced By Gray's Methodology

As claimed, a readable design has been produced, in that the outputs are on the right-hand side of the graph boundaries, the inputs are on the left-hand side of the boundaries, and the internal logic is in between the boundary lines sequentially down the page. However, it is not a dependable design, and Gray's methodology does not give full guidance to design dependable PLC control systems. For example, inhibitor arcs had to be used but they are not part of his methodology. Instead, when representing inverse conditions, Gray uses separate Occam variables for each

condition, for example for the Miller Controller (Figure 2-5) he uses a 'Milling in Progress' state and 'Milling Finished' state, which are inverse conditions, i.e. 'Milling Finished' is logically the same as 'Milling in Progress' 'Not On'. This representation has no equivalence in relay logic or Ladder Diagrams when considering the state of sensors which can be either 'On' or 'Off', or output signals which can also be 'On' or 'Off'. Therefore an inhibitor arc is used to represent a signal which is 'Not On'. Note also that a PLC signal only has two states, 'On' or 'Off', and therefore places were declared to be bounded to one token.

To simulate the control system an initial marking is required. For example, to simulate the oil sensor switching on, a token is placed inside the place **Oil Sensor**, and the states of all transitions are considered sequentially down the page. Transition **t1** is enabled and thus fires. Transitions **t2** and **t3** do not fire. The resulting marking is shown in Figure 3-2(a), which shows that if the sensor is switched on, the siren is sounded, as stated by the requirements. Note that **Oil Sensor** regains its token via the return arc, which is needed to show that the sensor does not switch off as a result of the siren switching on. The siren retains its token, i.e. continues to sound, until the push button is pressed and then released.

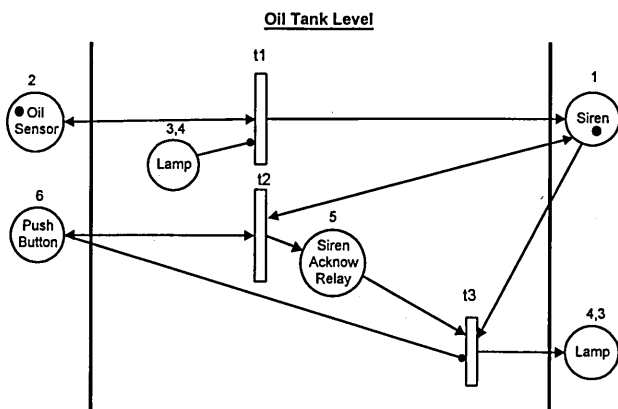
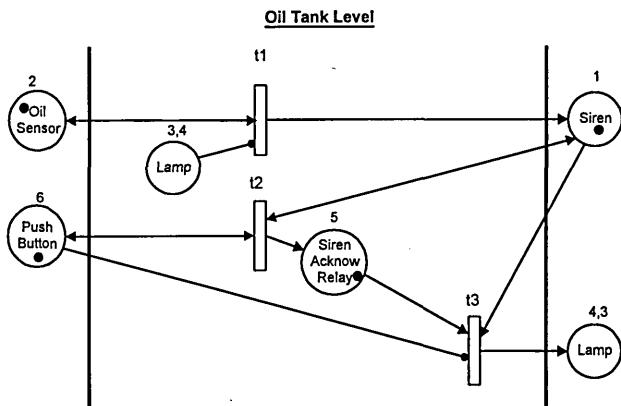


Figure 3-2(a) Marking Achieved as a Result of Simulating Oil Sensor Switching on

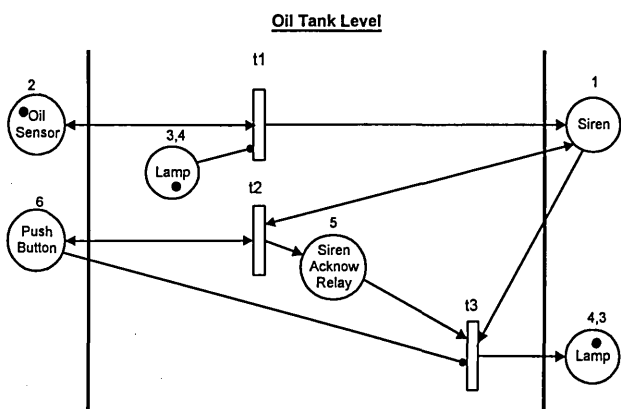
To simulate the pressing of the push button, a token is placed inside **Push Button**.

Figure 3-2(a) is simulated again and a final marking as shown in Figure 3-2(b) is obtained.



**Figure 3-2(b) Marking Achieved as a Result of Simulating Push Button Being Pressed**

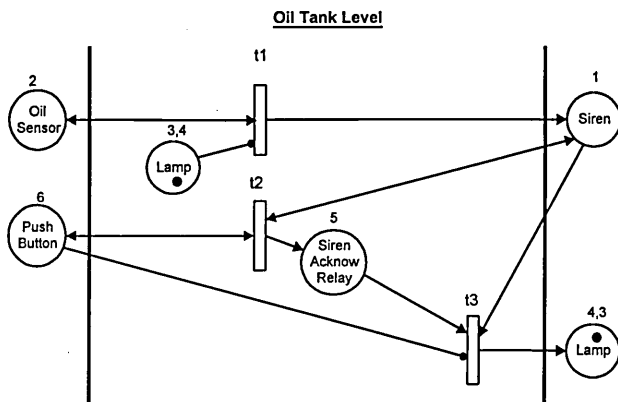
This also meets the requirements of the system. Figure 3-2(b) shows that if the siren is sounding and the push button is pressed, an internal relay is switched on, which represents the acknowledgment of the siren. This marking remains unchanged until the token is removed from **Push Button**, which simulates the push button being released, and a final marking as shown in Figure 3-2(c) is obtained.



**Figure 3-2(c) Marking Achieved as a Result of Simulating Push Button Being Released**

The place **Lamp** receives a token via transition **t3**. Also via **t3** the places **Siren Acknow Relay** and **Siren** lose their tokens. This again appears to correctly represent the requirements; when the button is pressed and subsequently released, the siren is switched off and the lamp is switched on.

The design rules of the methodology do not offer guidance for switching output signals off. Gray assumes that all outputs are communications over Occam channels and does not consider such outputs as the lamp. Although **Siren** is switched off via **t3** and **Siren Acknow Relay** is switched off via **t2**, the methodology does not have a design rule for switching **Lamp** off. This is shown in Figure 3-2(d). The token is removed from **Oil Sensor** to simulate that oil is added to the tank, but **Lamp** continues to retain its token.



**Figure 3-2(d) Marking Achieved as a Result of Simulating Oil Sensor Switching off**

The simulation of the PN graph shows that by using the design rules of Gray's methodology, unreliable PLC programs can be produced. This is because the methodology was developed for Occam to run on Transputers, resulting in rules which ensure the reliable design and simulation solely for such systems. To achieve the same level of reliability for PLCs and LDs, it is essential to develop rules (section 4.1) which ensure that the PN design is simulated in the same way as a LD translated

from the PN design would run on a PLC.

### 3.1.2 Translation of PN into LDs

Gray has only devised rules for translation into Occam, but not into any of the PLC languages. By applying his translation rules to the PN graph, the following Occam code could be produced, although Gray has not considered inputs from sensors but has only considered inputs such as communications from other Occam processes.

#### Occam Code

*WHILE TRUE*

*SEQ*

*IF*

*oil-sensor AND (NOT lamp)*

*siren := TRUE*

*SKIP*

*IF*

*push-button AND siren*

*siren-acknow-relay := TRUE*

*SKIP*

*IF*

*siren-acknow-relay AND NOT push-button AND siren*

*SEQ*

*lamp := TRUE*

*siren := FALSE*

*siren-acknow-relay := FALSE*

*SKIP*

This Occam code does not have an equivalent Ladder Diagram. This is because IF constructs do not resemble the rungs of a Ladder Diagram. Programmable Logic Controllers read and execute their control logic in a way that has not been considered by Gray's methodology. This is discussed in more detail in section 4.3.

Gray has not considered translation of the PN graph directly into PLC code. Also, the Occam translation of the control does not model PLC logic, and has no equivalent Ladder Diagram. It was decided, therefore, that rules for translating from a Petri Net design into a Ladder Diagram needed to be developed (section 4.2).

### 3.2 Discussion

When applying Gray's methodology to develop the low level control logic of the FMC, it was found that:

- insufficient guidance is given to designing control systems that have inputs or outputs other than Occam channels.
- the resulting PN graph does not model the PLC programming languages, because there is no equivalent to a transition in any of the PLC languages. For example, transition **t3** firing, Figure 3-2(b), simultaneously switches the lamp on, and the siren and the internal relay off. There is no equivalent to this simultaneous state change in Ladder Diagrams.
- no guidance is given to simulating the PN graph in the same way as a PLC executes its control logic.
- rules are not given for translating the PN graph into PLC code.



Having studied the higher level parts of the DCS, Gray developed a readable and dependable methodology for generating reliable Occam code to run on Transputers. His methodology was found to be incomplete for designing the control of the overall DCS due to its inapplicability to the lower level PLC control. However, the readability of Gray's PN graphs, achieved as a result of his design approach, was considered to be very promising, and therefore research into the development of design, simulation and translation rules to apply to PLC control was conducted, to achieve the objectives listed in section 2.5. The outcome of this work was the development of a methodology,  $PN \Leftrightarrow PLC$ , which will work as a stand alone tool for developing dependable PLC control programs, and will also unify with Gray's methodology to produce a complete methodology for developing an overall dependable Distributed Control System. This work is reported in Chapters 4, 5 and 6.

# Chapter 4

## PN $\leftrightarrow$ PLC: The Methodology

### Contents

<b>4. PN<math>\leftrightarrow</math>PLC: The Methodology .....</b>	<b>46</b>
<b>4.1 Design Process .....</b>	<b>47</b>
4.1.1 Design Rules.....	48
4.1.2 Terminology and Symbols.....	52
<b>4.2 Translation Process .....</b>	<b>54</b>
4.2.1 Translation Rules .....	54
<b>4.3 Simulation Process.....</b>	<b>56</b>
4.3.1 Simulation Steps .....	57
4.3.2 Simulation Steps Applied to Scenario 2 .....	61
4.3.3 Simulation Results.....	64
<b>4.4 The Investigative Approach Taken to Develop PN<math>\leftrightarrow</math>PLC .....</b>	<b>64</b>
<b>4.5 Designing the Control Algorithm of the Oil Tank Using SFCs.....</b>	<b>70</b>

## 4. PN $\leftrightarrow$ PLC: The Methodology

This chapter describes the novel methodology, developed during this research programme, for designing PLC programs. The methodology, PN $\leftrightarrow$ PLC, uses a similar format to LDs, but is readable by any party, be they the DCS designer, the software writer, the commissioner or user of the DCS, supports simulation and can be easily and exactly translated into an efficient Ladder Diagram [Taholakian and Hales 1996]. The methodology consists of three stages, Design, Simulation and Translation which are introduced in this chapter using a simple control example. The rules and terminology required to use the methodology are also explained. They are essential for producing reliable control systems, and a complete set of the rules is contained in Appendix A.

The investigative approach taken to develop PN $\leftrightarrow$ PLC is discussed at the end of the chapter, which also helps to explain the inadequacies of previous applications of Petri Nets to PLC programming and the comparative advantages of PN $\leftrightarrow$ PLC.

### **Control Example: Oil Tank Level**

In order to explain the PN $\leftrightarrow$ PLC methodology the previous example, Oil Tank Level (section 3.1.1), is used.

Consider Figure 3-1, duplicated below for convenience. When the oil level drops, the oil sensor switches on and a siren is sounded. By pressing and releasing a spring return push button the siren is switched off and a lamp is switched on. The lamp is switched off only if oil is added to the tank, i.e. the oil level sensor has switched off.

The siren must sound regardless of the push button being jammed in the pressed position, in an attempt by the operator to stop the siren from ever sounding.

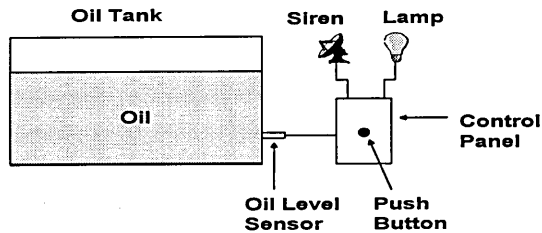


Figure 3-1 Oil Tank Level

## 4.1 Design Process

The general structure of the methodology is based on Gray's methodology, which in turn is very similar to Ladder Diagrams. Two vertical lines represent the boundaries of the PN. The logical sequence of events carried out by the PLC which includes the setting of internal relays is represented within these boundaries. The inputs to the PLC are represented by input places outside the graph boundaries on the left-hand side. The outputs from the PLC are represented by output places on the right hand side of the boundaries as shown in Figure 4-1.

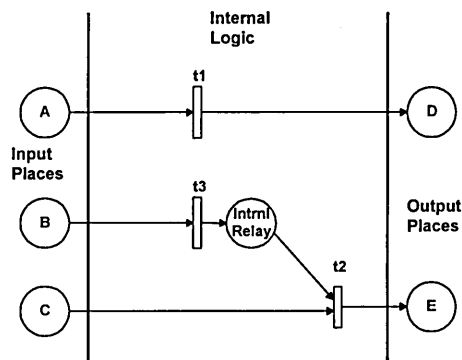
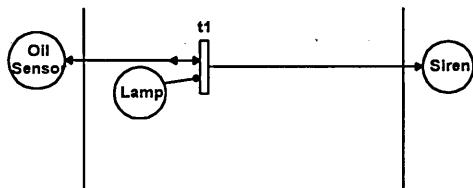


Figure 4-1 Layout of PN⇔PLC Graph

The design rules listed below are an integral part of the methodology. They ensure that the resulting PN graph of the control system is correct, readable, easily checked, simulated and can be exactly translated into a Ladder. Also the symbols and terminology used in Figures 4-2(a) through 4-2(d) are specified by the methodology and are explained in section 4.1.2.

### 4.1.1 Design Rules

**Rule 1.** *The drafting of the PN $\leftrightarrow$ PLC graph is carried out using a right-to-left and down-the-page approach i.e. starting with an output, work backwards and determine the conditions for switching the output on. This must be represented with one or more "switching-on transitions" within the boundaries of the PN graph, and their "switching-on places", as shown in Figure 4-2(a).*



**Figure 4-2(a)** First Design Rule Applied to Siren

The control specification states that the siren is the first output from the PLC and therefore the first rule is applied to the siren, as shown in Figure 4-2(a). Transition **t1** is the "switching-on transition" of the output place **Siren**. **Oil Sensor 'On' AND Lamp 'Not On'** are the conditions for switching the siren on, and are therefore connected to **t1** as shown. Although an output from the PLC, the place **Lamp** is duplicated closer to **t1** for readability (section 4.1.2). The place **Oil Sensor** is connected to **t1** with a "return arc" because the conditions for switching it on or off are not controlled by the PLC (section 4.1.2).

**Rule 2.** *Consider the conditions for switching the output off. This must be represented with one or more "switching-off transitions", to the right of the output place, and their "switching-off places", as shown in Figure 4-2(b).*

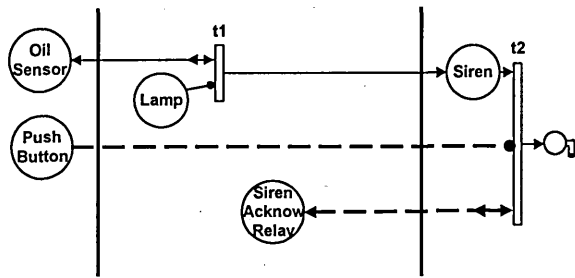
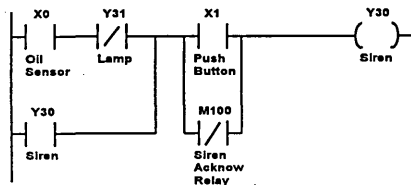


Figure 4-2(b) Second Design Rule Applied to Siren

Transition **t2** is the "switching-off transition" of the output place **Siren**. The place **Siren Acknow Relay** is an internal relay which represents that the sounding of the siren has been acknowledged. **Siren Acknow Relay 'On' AND Push Button 'Not On'** are the conditions for switching the siren off, and are therefore connected to **t2** with broken or "switching-off arcs". This rule, combined with the symbols used (section 4.1.2), is particularly important to ensure that the PN graph is readable. Both the switching-off state and switching-off instance are shown. A third party looking at the PN graph in Figure 4-2(b) knows how and when the designer intends to switch the siren off. Moreover, the rule is essential for the correct translation of the PN graph into PLC code.

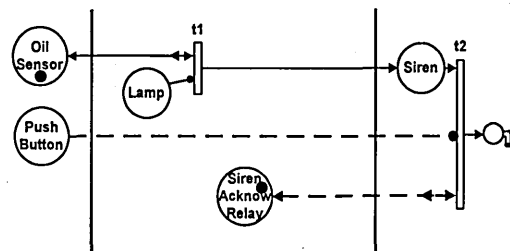
### Firing of Transitions in a PN $\Leftrightarrow$ PLC Graph

In order to understand how the PN $\Leftrightarrow$ PLC graph in Figure 4-2(b) correctly represents the operation of a PLC program, it is necessary to understand the movement of tokens and the firing of transitions. A translation of Figure 4-2(b) is shown in Figure 4-2(bl), without any explanation of how the translation was made. The complete translation rules of PN $\Leftrightarrow$ PLC are given in sections 4.2.1.

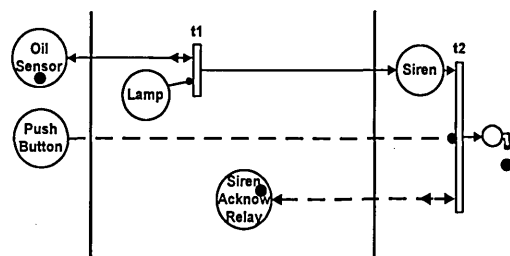


**Figure 4-2(b1) LD Rung for Siren Obtained by Applying  
PN $\leftrightarrow$ PLC Translation Rules to Figure 4-2(b)**

A realistic marking is shown in Figure 4-2(b1), which represents **Oil Sensor** ‘On’ and **Siren Acknow Relay** ‘On’ to be true. The result of this marking is shown in Figure 4-2(b2). In Figure 4-2(b1), both the “switching-on transition”, **t1**, and the “switching-off transition”, **t2**, are enabled and fired. Therefore the place **Siren** does not gain a token, as shown in Figure 4-2(b2).



**Figure 4-2(b1) A Realistic Marking Applied to Figure 4-2(b)**



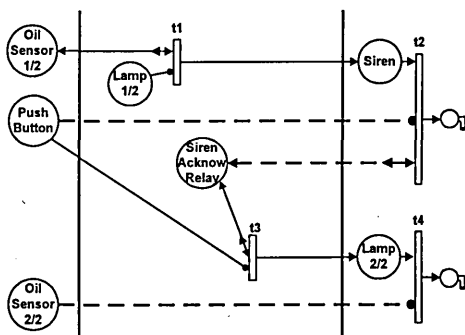
**Figure 4-2(b2) Evolution of the Marking After Applying  
Simulation Step 3 to Figure 4-2(b1)**

Simulating the oil sensor and the internal relay being ‘On’ in the Ladder Diagram, Figure 4-2(b1), it can be seen that the siren does not switch on. Therefore it is important to note that when simulating a PN $\leftrightarrow$ PLC graph, an output is considered not to have switched on if both its “switching-on

transition" is enabled and the switching-off conditions are true. This very important simulation rule is listed in section 4.3.1 as rule 3, and is essential to ensure that a  $PN \Leftrightarrow PLC$  graph correctly simulates the Ladder Diagram into which it is translated.

**Rule 3.** *Rules 1 and 2 are repeated for all output places, and also for any internal relays used.*

By applying the design rules to the lamp the  $PN \Leftrightarrow PLC$  graph shown in Figure 4-2(c) is achieved. The specification states that the conditions for switching the siren off are also the conditions for switching the lamp on, therefore **Siren Acknow Relay 'On'** **AND Push Button 'Not On'** are the "switching-on places" and are connected to the "switching-on transition", **t3**, of the output place **Lamp**. The condition for switching the lamp off is **Oil Sensor 'Not On'** as indicated by the "switching-off transition" **t4**.



**Figure 4-2(c) Design Rules Applied to Siren and Lamp**

Finally the design rules for switching the place **Siren Acknow Relay** on and off are applied. It is by pressing the push button that the sounding of the siren is acknowledged. Therefore **t5**, as shown in Figure 4-2(d) below, is the "switching-on transition" for **Siren Acknow Relay**. The correct conditions for switching **Siren Acknow Relay** off are **Oil Sensor 'Not On'** **AND Siren 'Not On'**. There is



potentially the risk of excluding the former of the two switching-off conditions at the design stage, however the simulation process, discussed in section 4.3, highlights the need for both conditions to switch **Siren Acknow Relay** off.

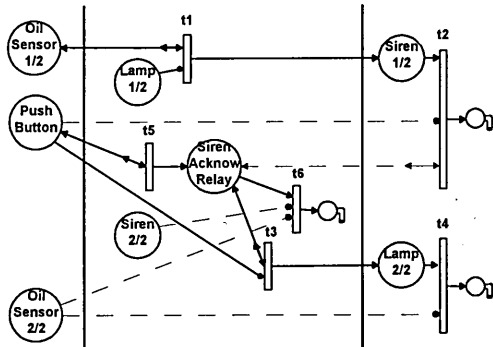


Figure 4-2(d) Final PN to PLC Graph of Oil Tank Level Control

### 4.1.2 Terminology and Symbols

In order to make the PN graph readable, certain symbols have been adopted as follows:

←→ is a "return arc" which shows that an input place will receive its token back after the transition has fired. For example this applies to input signals over which the PLC has no control. In Figure 4-2(d), the oil sensor or the push button can switch on or off at any time, and not as a result of the transitions (t1 or t5) firing. Also, the place **Siren Acknow Relay** is connected to t3 with a "return arc" because it does not switch off as a result of the lamp switching-on.

⋯→/⋯● are "switching-off arcs" which clearly show what is being switched off and what is doing the switching off. Consider transition t4 in Figure 4-2(c); **Oil Sensor 2/2** 'Not On' is the condition to switch **Lamp 2/2** off, and not vice versa. These arcs are essential for the design of correct PLC programs. For instance in Figure 2-4(a), duplicated below for convenience, there is an ambiguity. It is not clear

which of the inputs (A, B or C) is the condition for switching-off the others, or indeed whether it is a combination of two of the inputs that switches the third off and therefore this can not be translated reliably into Ladder logic. Combined with design rule 2, the "switching-off arcs" show exactly what the designer has in mind.

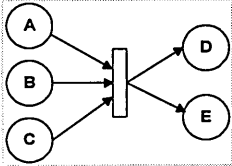


Figure 2-4(a)



are duplicated places. Where a place is the input to several transitions, it can be duplicated in order to reduce the number of arcs that cross. This simplifies the overall PN graph and improves readability. In Figure 4-2(d), places **Siren**, **Lamp** and **Oil Sensor** are duplicated for this reason. It is, however, essential that a tally is kept of the number of duplicate places (1/2, 2/2) so that they are not overlooked by the designer and simulator.



is a "dummy" or "drain place", which is used to demonstrate that a token is drained from a particular place as a result of its transition firing. This symbol clearly shows the "switching-off transitions", and proves that the designer has considered how each output and internal relay will be switched off. For instance in Figure 4-2(d), it is clear that transition  $t_6$  is the "switching-off transition" of **Siren Acknow Relay**.

## 4.2 Translation Process

Simulation, not translation, is actually the next stage in the methodology after design.

However the translation rules need to be explained before the simulation can be fully comprehended.

The PN $\leftrightarrow$ PLC has only a few simple rules for directly translating the PN $\leftrightarrow$ PLC graph into an exactly equivalent Ladder Diagram: if the design is correct, the code is correct.

Similar to design rule number 1, the translation process is carried out using the right-to-left and down-the-page approach.

### 4.2.1 Translation Rules

*Rule 1. Consider an output place and situate it as an output on the right-hand side of the Ladder Diagram. Consider the “switching-on transition” to the left of that output place. The input places to that transition are the conditions for switching the output place on, and are therefore represented as inputs on the left-hand side of the Ladder Diagram.*

Figure 4-3(a) represents the rung for switching on the output Siren.

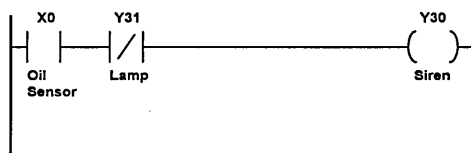


Figure 4-3(a) LD Showing the Conditions for Switching the Siren on.  
First Translation Rule

*Rule 2. Always latch the output with itself.*

This is done to ensure that the output remains 'On' until the conditions to switch it off become true, as shown below in Figure 4-3(b).

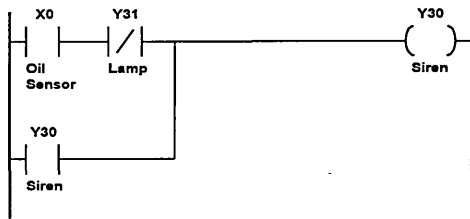


Figure 4-3(b) LD Showing How the Siren is Latched on. Second Translation Rule

**Rule 3.** Consider the “switching-off transition” to the right of the output place. The “switching-off arcs” joining places to this transition identify these places as the switching-off conditions of that output place, and are therefore represented on the LD as shown in Figure 4-3(c).

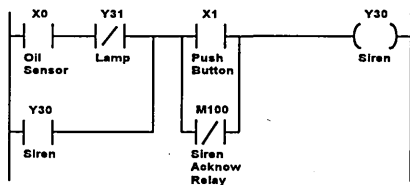
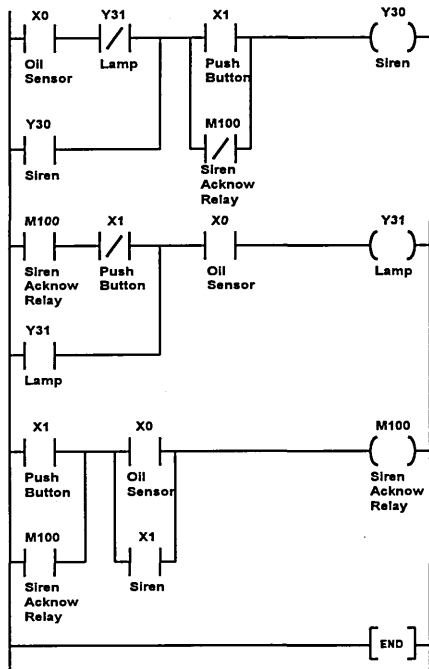


Figure 4-3(c) LD Showing the Conditions for Switching the Siren off. Third Translation Rule

In Figure 4-2(d) Siren Acknow Relay 'On' AND Push Button 'Not On' are the conditions for switching-off the siren. However, it is the inverse of these states which keep the siren 'On', thus using De Morgan's law, they are represented as Push Button 'On' OR Siren Acknow Relay 'Not On' on the Ladder Diagram, as shown in Figure 4-3(c).

**Rule 4.** Repeat rules 1 - 3 for all the places.

Rules 1 to 3 are also applied to the places **Lamp** and **Siren Acknow Relay**, and a final LD of the  $PN \Leftrightarrow PLC$  graph in Figure 4-2(d) is achieved as shown in Figure 4-3(d).



**Figure 4-3(d) Final LD. An Exact behavioural Equivalent to the  $PN \Leftrightarrow PLC$  Graph Shown in Figure 4-2(d)**

### 4.3 Simulation Process

The simulation of the  $PN \Leftrightarrow PLC$  graph is essential to check the correctness of its logic. To achieve true simulation of the control algorithm, it is therefore necessary that the mode of operation of the PLC is simulated by the  $PN \Leftrightarrow PLC$  graph (This very important point is discussed in greater depth in Chapter 6, sections 6.3 and 6.5). Most PLCs scan through a program in a cyclic fashion, and only update their outputs at the end of each scan, not during a scan. Therefore if the "switching-on transition" and the "switching-off transition" of an output place are fired consecutively within the same scan, that output will not be switched on at the end of that scan. Also, a PLC signal

only has two states, 'On' or 'Off', and therefore any place is bounded to one token, i.e. if it has a token it can not receive another.

Simulation is performed by marking places with tokens and determining their flow through the transitions. For most applications there will be a large number of marking permutations, i.e. simulation scenarios, most of which will be unrealistic, and therefore the designer will only model a few of them. 'Scenario 1' in section 4.3.1 represents a realistic simulation scenario, in which it is assumed that events occur in sequence as stated by the specification.

### 4.3.1 Simulation Steps

The following scenario of events is used to explain the simulation steps.

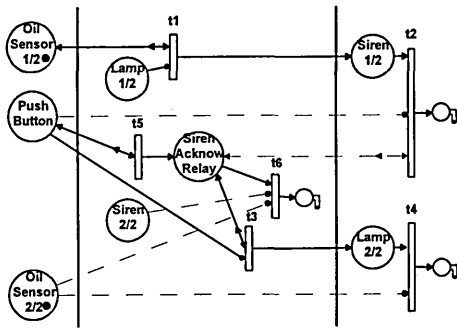
#### Scenario 1

- a) the oil sensor switches on.
- b) if the siren is on, the push button is pressed.
- c) the push button is then released.
- d) finally the oil sensor is switched off.

*Step 1. The first step of the simulation process is, thus, to determine one or more realistic scenarios of events.*

*Step 2. Consider the first event of the chosen scenario and mark the PN $\leftrightarrow$ PLC graph accordingly.*

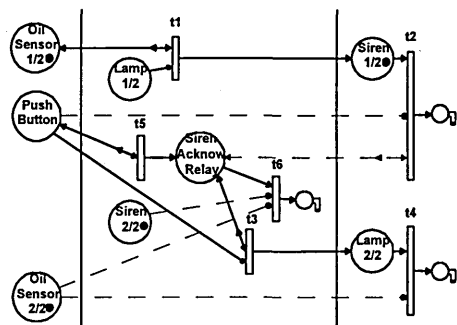
As stated in 'Scenario 1', the first event is that the oil sensor switches on and therefore a token is put inside the place **Oil Sensor** 1/2 and 2/2, as shown below in Figure 4-4(a).



**Figure 4-4(a) Initial Marking Representing the Oil Sensor Switching on, i.e. Scenario 1, Event (a). Simulation Step 2**

**Step 3.** *Given the initial marking, determine the state of each output place and internal relay. This is done by considering the “switching-on transition” and then the “switching-off transition” of each place consecutively. Note, if both transitions are fired then the output place does not receive a token. The final marking of the  $PN \Leftrightarrow PLC$  graph represents the state of the system after one scan of the program.*

Applying simulation step 3 to Figure 4-4(a), the state of the place **Siren** is affected by this marking. Transition **t1** is enabled and fired. Transition **t2** is not enabled and therefore **Siren 1/2** and **2/2** retain their tokens. Likewise transitions **t3** to **t6** are not enabled and therefore **Siren Acknow Relay** and **Lamp** are not affected by the oil sensor and siren being ‘On’ and thus do not gain tokens. Figure 4-4(b) below shows the final marking of the  $PN \Leftrightarrow PLC$  graph after applying this rule and represents the state of the system at the end of the first scan.



**Figure 4-4(b) Evolution of Marking After Applying Simulation Step 3 to Figure 4- 4(a)**

*Step 4. The resulting marking from one scan is the initial marking for the next scan which is simulated as in step 3. A number of scans are made until the  $PN \Leftrightarrow PLC$  graph reaches a steady state, i.e. the marking does not change from one scan to the next.*

For example when simulated for a further scan, the marking shown in Figure 4-4(b) does not change and therefore indicates that the  $PN \Leftrightarrow PLC$  graph is in a steady state, i.e. when the oil sensor switches on, the siren is sounded, as stated by the requirements.

It is important to note that the same final marking is obtained regardless of the order in which the places are considered, provided they are considered as stated in Steps 3 and 4<sup>1</sup>. This is essential in order to ensure that the design is deterministic, i.e. the same outcome is always obtained for a given set of input conditions, irrespective of the order in which the outputs occur in the graph; given the initial marking in Figure 4-4(a), the final marking in Figure 4-4(b) is achieved whether the state of **Siren** is considered first, second or last. This exactly simulates a PLC program.

*Step 5. Once a steady state has been achieved then the markings representing the next event in the scenario are included in the  $PN \Leftrightarrow PLC$  graph.*

Continuing with scenario 1, event (b) is considered and following from Figure 4-4(b), a token is placed inside the place **Push Button** as shown below in Figure 4-4(c).

---

<sup>1</sup> In some cases, the control requirements may prescribe that a specific sequence of outputs occurs. This can be achieved by using “priority” transitions as described in Chapter 5.



Steps 1 - 4 are applied to Figure 4-4(c). The result of this event is shown in Figure 4-4(d).

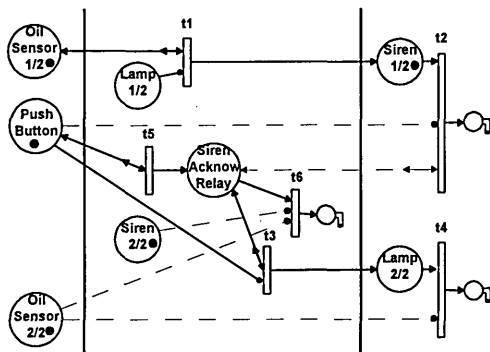


Figure 4-4(c) Marking which Represents Scenario 1, Event (b)

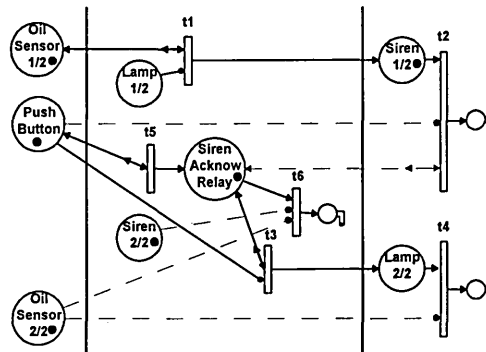


Figure 4-4(d) Evolution of Marking After Applying Simulation Steps to Figure 4-4(c)

Figures 4-4(c) & 4-4(d) show that if the siren is sounding and the push button is pressed, the internal relay **Siren Acknow Relay** is switched on but the siren continues to sound, as stated by the specification.

Event (c) is considered next, and following from Figure 4-4(d), the token is removed from the place **Push Button** as shown below in Figure 4-4(e). Steps 1 - 4 are applied to Figure 4-4(e). The result of this event is shown in Figure 4-4(f).

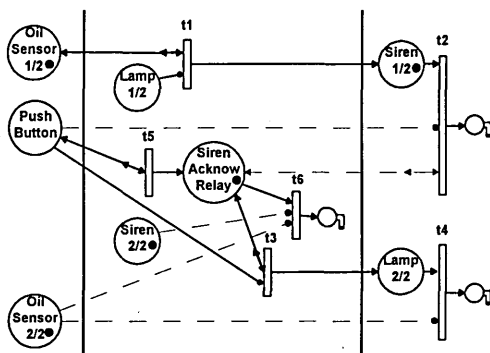


Figure 4-4(e) Marking which Represents Scenario 1, Event (c)

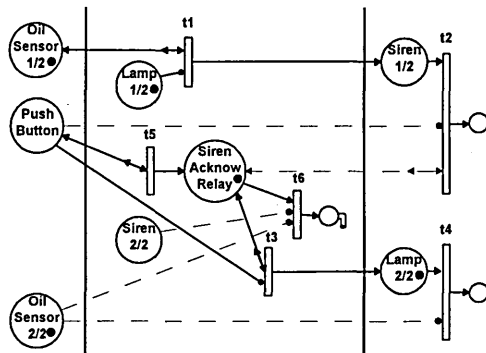
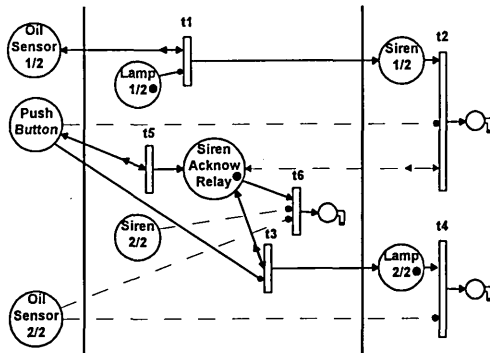


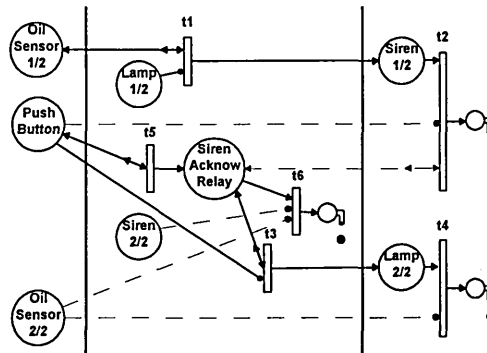
Figure 4-4(f) Evolution of Marking After Applying Simulation Steps to Figure 4-4(e)

Figures 4-4(e) & 4-4(f) show that if the push button is released, the siren is switched off and the lamp is switched on, as stated by the specification.

Finally, event (d) is considered, and following from Figure 4-4(f), the token is removed from the place **Oil Sensor**, as shown below in Figure 4-4(g). Steps 1 - 4 are applied to Figure 4-4(g). The result of this event is shown in Figure 4-4(h).



**Figure 4-4(g) Marking which Represents Scenario 1, Event (d)**



**Figure 4-4(h) Evolution of Marking After Applying Simulation Steps to Figure 4-4(g)**

Figures 4-4(g) and 4-4(h) show that when oil is added to the tank, i.e. the oil sensor switches off, the lamp switches off, as stated by the specification. The internal relay also switches off and therefore the system resumes its normal state.

### 4.3.2 Simulation Steps Applied to Scenario 2

Another possible realistic simulation scenario is described in Scenario 2 as follows:

#### Scenario 2

- the push button is jammed in the pressed position, in an attempt by the operator to prevent the siren from ever sounding.
- the oil sensor switches on.
- the oil sensor switches off due to the oil being low but unsettled.
- the push button is released.

In Scenario 2, the events do not follow the sequence described by the specification.

The listed sequence of events is however a realistic possibility and a good safety

check, to ensure that the control system is correct.

Figure 4-5(a) represents event (a). Applying the simulation steps to Figure 4-5(a), a final marking as shown in Figure 4-5(b) is achieved. Although transition **t5** fires and **Siren Acknow Relay** receives a token, transition **t6** also fires within the same scan, and therefore the internal relay will not switch on given the condition in event (a).

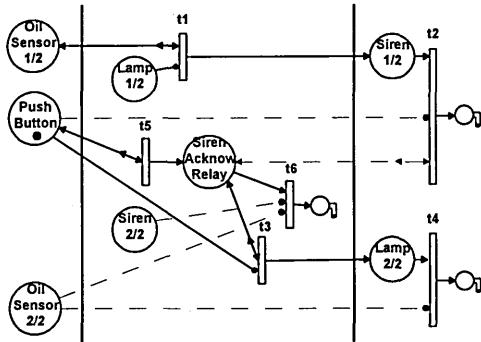


Figure 4-5(a) Marking which Represents Scenario 2, Event (a)

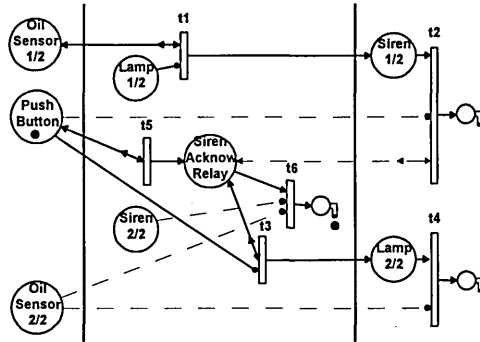


Figure 4-5(b) Evolution of Marking After Applying Simulation Steps to Figure 4-5(a)

Event (b) is represented in Figure 4-5(c). The resulting marking is shown in Figure 4-5(d). The design is shown to be correct in that the siren is sounded regardless of jamming the push button.

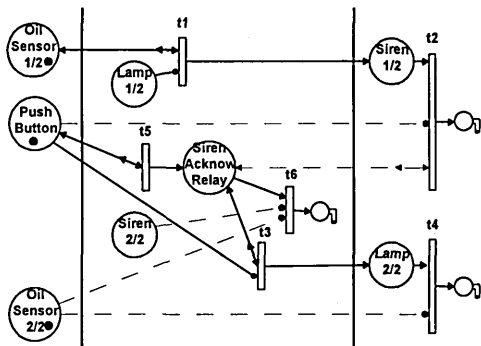


Figure 4-5(c) Marking which Represents Scenario 2, Event (b)

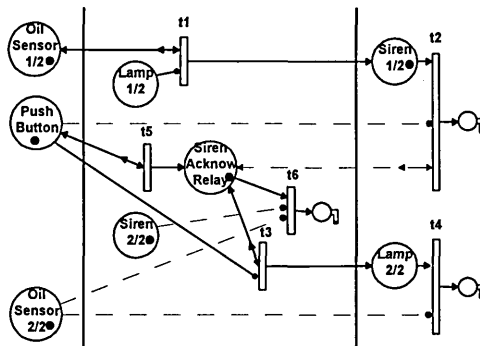


Figure 4-5(d) Evolution of Marking After Applying Simulation Steps to Figure 4-5(c)

In event (c) the possibility of a flickering sensor is simulated, as shown in Figure 4-5(e). Figure 4-5(f) shows that there is no change in the state of the system: the siren

continues to sound regardless of the oil sensor being off, as prescribed by the specification.

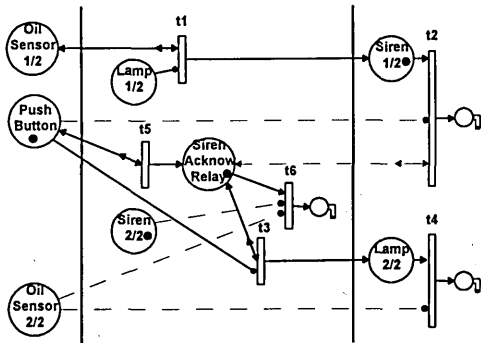


Figure 4-5(e) Marking which Represents Scenario 2, Event (c)

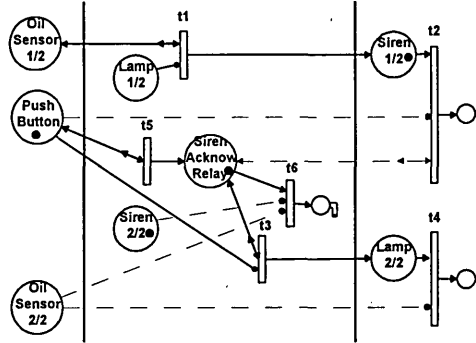


Figure 4-5(f) Evolution of Marking After Applying Simulation Steps to Figure 4-5(e)

Event (d) is represented in Figure 4-5(g). The push button is released and therefore the token is taken out of the place **Push Button**. The resulting marking is shown in Figure 4-5(h). The places **Siren** and **Siren Acknow Relay** lose their tokens indicating that the system returns to the normal state. This verifies the correctness of the design: the siren switches off and the lamp does not switch on because the oil sensor is no longer on.

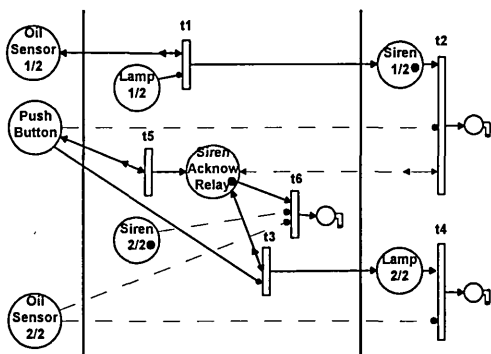


Figure 4-5(g) Marking which Represents Scenario 2, Event (d)

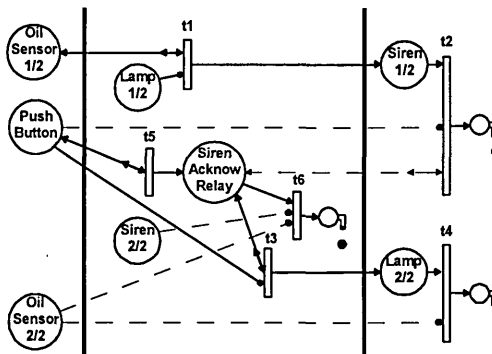


Figure 4-5(h) Evolution of Marking After Applying Simulation Steps to Figure 4-5(g)

Other scenarios can be simulated in this way, for example inputs which change during a scan can be included as events, but are not included here.

When considering ‘Scenario 2’ the design, simulation and translation rules of  $PN \Leftrightarrow PLC$  can be fully appreciated. The design rules enforce a  $PN \Leftrightarrow PLC$  graph which exactly represents and simulates a PLC program. This eliminates the need for verification of the PLC code against the design.

### 4.3.3 Simulation Results

The results of the simulation can also be obtained using the following **State or Boolean Equations**, instead of marking the  $PN \Leftrightarrow PLC$  graph:

$$\mathbf{Siren} = ((\overline{\mathbf{Lamp}} \bullet \mathbf{Oil\ Sensor}) + \mathbf{Siren}) \bullet (\mathbf{Push\ Button} + \overline{\mathbf{Siren\ Acknow}})$$

$$\mathbf{Lamp} = ((\mathbf{Siren\ Acknow} \bullet \overline{\mathbf{Push\ Button}}) + \mathbf{Lamp}) \bullet \mathbf{Oil\ Sensor}$$

$$\mathbf{Siren\ Acknow} = (\mathbf{Push\ Button} + \mathbf{Siren\ Acknow}) \bullet (\mathbf{Oil\ Sensor} + \mathbf{Siren})$$

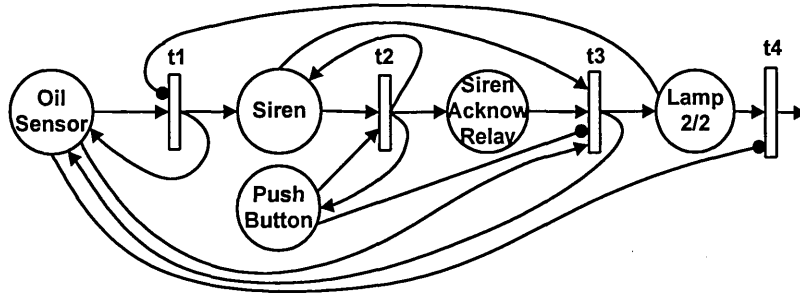
These equations are derived from the  $PN \Leftrightarrow PLC$  graph, Figure 4-2(d), and are of use when computerising the graphical simulation process. They are exact equivalents of the rungs of the Ladder Diagram.

This representation of the control logic underpins the formalism of the methodology. However the understanding of this mathematical representation is not required in order to use  $PN \Leftrightarrow PLC$ .

## 4.4 The investigative approach taken to develop $PN \Leftrightarrow PLC$

The reported research programme underwent a trial and error process, which led to the development of the reliable design, simulation and translation rules of  $PN \Leftrightarrow PLC$ . The approach followed in other work was initially trialled, but was found to be inapplicable. The discussion given below gives an indication of the errors which arose using these approaches and reinforces the comparative advantages of  $PN \Leftrightarrow PLC$ .

The previous oil tank level example is used and, by following the requirements of the system (section 4), the design of the system can be represented as a basic PN graph, as shown in Figure 4-6.



**Figure 4-6 Petri Net Graph of Oil Tank Level Control**

This PN representation was the most frequently encountered during the literature survey phase (sections 1.3.1 and 2.4.2) of the reported research programme.

Simulation ‘Scenario 1’, duplicated below for convenience, is used in order to verify the PN design in Figure 4-6 against the requirements of the system.

#### Scenario 1

- a) the oil sensor switches on.
- b) if the siren is on, the push button is pressed.
- c) the push button is then released.
- d) finally the oil sensor is switched off.

As stated above, the first event is that the oil sensor switches on and therefore a token is placed inside **Oil Sensor**. Transition **t1** is enabled and fired. Transition **t3** is not enabled therefore **Siren** retains its token. Likewise transitions **t2** and **t4** are not enabled and therefore **Siren Acknow Relay** and **Lamp** are not affected by the oil sensor and siren being ‘On’, and thus do not gain tokens. Figure 4-6(a) below shows the final marking of the PN graph after simulating event (a).

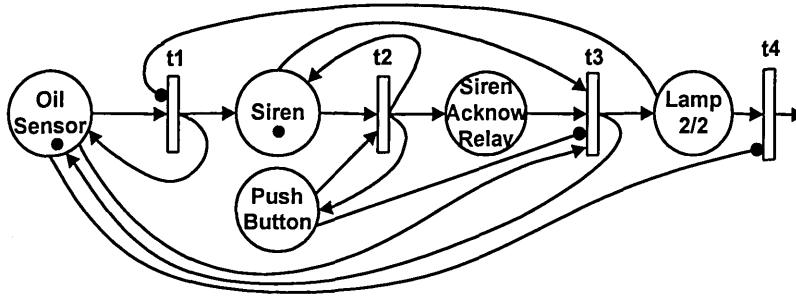


Figure 4-6(a) Evolution of the Marking After Simulating Scenario 1, Event (a)

Continuing with scenario 1, event (b) is considered and following from Figure 4-6(a), a token is placed inside **Push Button** and the PN graph is simulated. The result of this event is shown below in Figure 4-6(b).

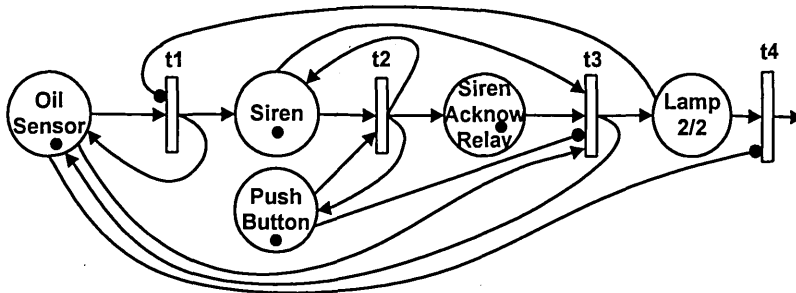


Figure 4-6(b) Evolution of the Marking After Simulating Scenario 1, Event (b)

Figure 4-6(b) shows that if the siren is sounding and the push button is pressed, the internal relay **Siren Acknow Relay** is switched on, but the siren continues to sound, as stated by the requirements.

Event (c) is considered next, and following from Figure 4-6(b), the token is removed from the place **Push Button** and the graph is simulated, the result of which is shown in Figure 4-6(c) below. Figure 4-6(c) shows that if the push button is released, the siren is switched off and the lamp is switched on, as stated by the requirements.

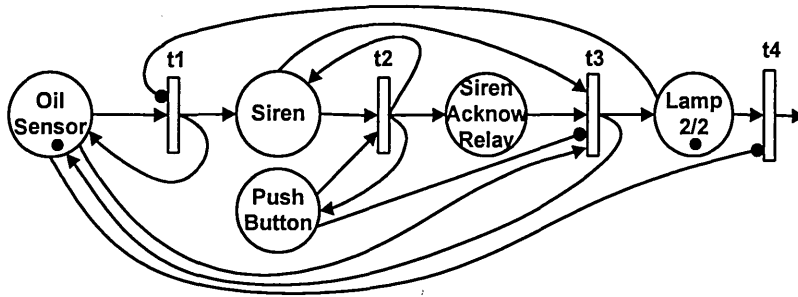


Figure 4-6(c) Evolution of the Marking After Simulating Scenario 1, Event (c)

Finally, event (d) is considered, and following from Figure 4-6(c), the token is removed from the place **Oil Sensor**, and as a result the lamp switches off, as stated by the requirements, and the system resumes its normal state. The result of event (d) is shown in Figure 4-6(d).

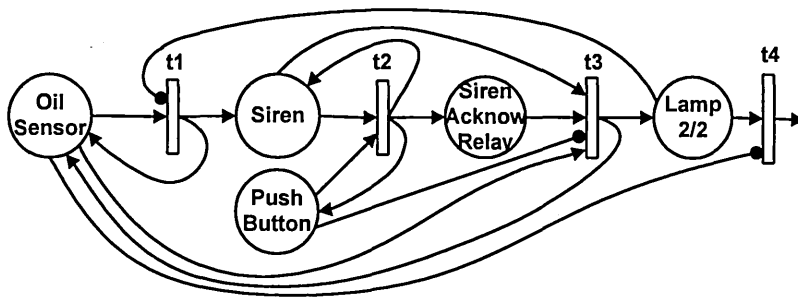


Figure 4-6(d) Evolution of the Marking After Simulating Scenario 1, Event (d)

The simulation of the PN graph shows that the design correctly models the requirements of the system. Most such Petri Net representations which were encountered in the literature (section 2.4.2) do not consider translation of the PN graphs into PLC code. Some have included Ladder Diagram translations of the PN graphs but not included any rules for carrying out the translation process. Furthermore, there is no consistency in the representation of the LD rungs [Green 1989]. Jafari and Boucher [Jafari and Boucher 1994] have considered translation into Ladder Diagrams. However, their approach undergoes a series of transformation



processes, whereby Integrated Computer Aided Manufacturing Definition 0 (IDEF0) is used for the top-down hierarchical decomposition of the system, until sufficient detail is obtained from which a model can be built. The IDEF0 representation is then transformed into a Petri Net model which in turn is analysed for reachability, modified to include the inputs and outputs of the controller, from which a state table is generated. A set of boolean equations is generated from the state table which are then transformed into a Ladder Diagram. Although the rules for the various transformation processes have been included it is not apparent that the reliability of this approach has been tested on PLCs. Moreover, the approach is complex due to the tedious development process which draws into question its usability in industry.

One feasible Ladder Diagram interpretation of the PN design of Figure 4-6 is shown in Figure 4-7(a). In Figure 4-6, when transition  $t_3$  is enabled and fired, the lamp switches on and also the internal relay and the siren switch off, therefore **Lamp 'On'** is the condition for switching off both **Siren** and **Siren Acknow Relay**. It is the inverse of **Lamp 'On'**, **Lamp 'Not On'**, which keeps both the siren and the internal relay 'On' (De Morgan's Law, section 4.2.1) as shown in Figure 4-7(a), rungs 1 and 2. The result of simulating events (a) and (b) in the LD of Figure 4-7(a) shows that the program meets the requirements of the system. The siren is switched on when the oil sensor switches on. The internal relay switches on when the siren is on and the push button is pressed. However, when simulating event (c) the LD does not model the system correctly. When the push button is released **Siren Acknow Relay** switches off and as a result the lamp does not switch on and the siren continues to sound.

Another feasible Ladder Diagram translation of the PN design is shown in Figure 4-7(b). In this figure a Ladder Diagram is generated from the PN design with the

understanding that the siren is switched off via transition **t3** with the following conditions: **Siren Acknow Relay** 'On', **Push Button** 'Not On' and **Oil Sensor** 'On'. **Siren Acknow Relay** is also switched off via **t3** with the following conditions: **Siren** 'On', **Push Button** 'Not On' and **Oil Sensor** 'On'. These are represented on the LD as the conditions to keep the siren and the internal relay 'On' using De Morgan's law, as shown in Figure 4-7(b) on rungs 1 and 2 respectively. Event (a) is simulated using Figure 4-7(b). The siren switches on due to **Oil Sensor** 'On' and **Lamp** 'On' being true and **Siren Acknow Relay** 'Not On' being true. Event (b) is simulated next and **Push Button** 'On' becomes true. The result of this event switches on **Siren Acknow Relay**. When considering event (c) the Ladder Diagram is again proved to be incorrect. Similar to Figure 4-7(a), when **Push Button** 'On' becomes false the internal relay switches off instantly, the siren continues to sound and the lamp does not switch on.

It can be seen that neither Ladder Diagram translation correctly represents the control algorithm for the oil tank level, yet the Petri Net design appears to be correct. To overcome this, research was conducted to develop rules such that the PN design correctly models the operation of the PLC code. This resulted in the rules of  $PN \Leftrightarrow PLC$  which ensure that PLC programs are designed and simulated in the same way as the Ladder Diagrams would run on a PLC.

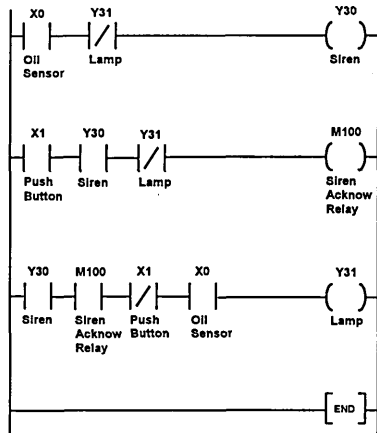


Figure 4-7(a)

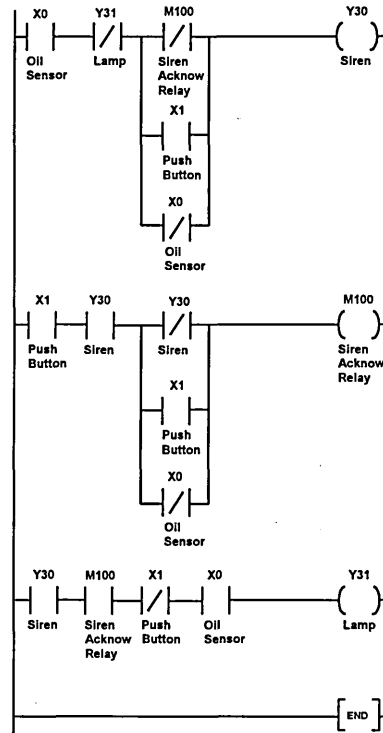


Figure 4-7(b)

Two Possible LD Interpretations of the PN Graph Shown in Figure 4-6

## 4.5 Designing the Control Algorithm of the Oil Tank Using SFCs

For the sake of comparison a Sequential Function Chart for the Oil Tank level control programme is shown below in Figure 4-8. Like the  $PN \leftrightarrow PLC$  graph it is easy to read.

The states of the system are denoted by the rectangular boxes (1 to 4). The bars labelled 1 to 5 are transitions. The program moves from one state into another via a transition. For instance, if in state 1 (normal) and the oil level sensor is switched on, the program moves into state 2 (siren), and is no longer in state 1. The *OR* branch following state 2 indicates that the program will next move into state 1 or state 3, depending on which transition is fired first. For example, if the conditions Oil Level Sensor 'Not On' and Push Button 'On' are true, transition 2 fires and the program moves into state 1. The logical sequence of events is clearly shown in the SFC, however the control logic, into which it is compiled, is lengthy and complex. To

illustrate this the SFC, Figure 4-8, has been translated into a Ladder Diagram using a proprietary SFC software package, Figure 4-9, which is considerably longer than that produced for the same task by PN $\leftrightarrow$ PLC, Figure 4-3(d).

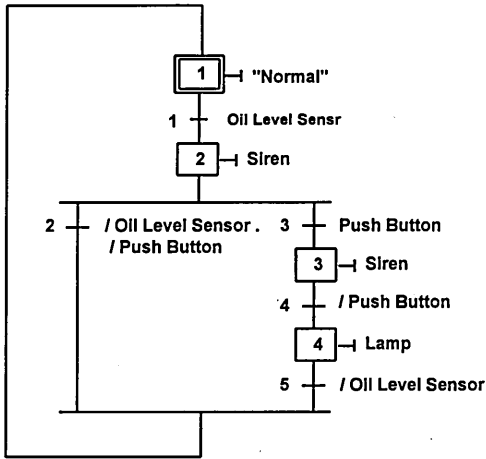


Figure 4-8 Design of Oil Tank Level Control Using SFCs

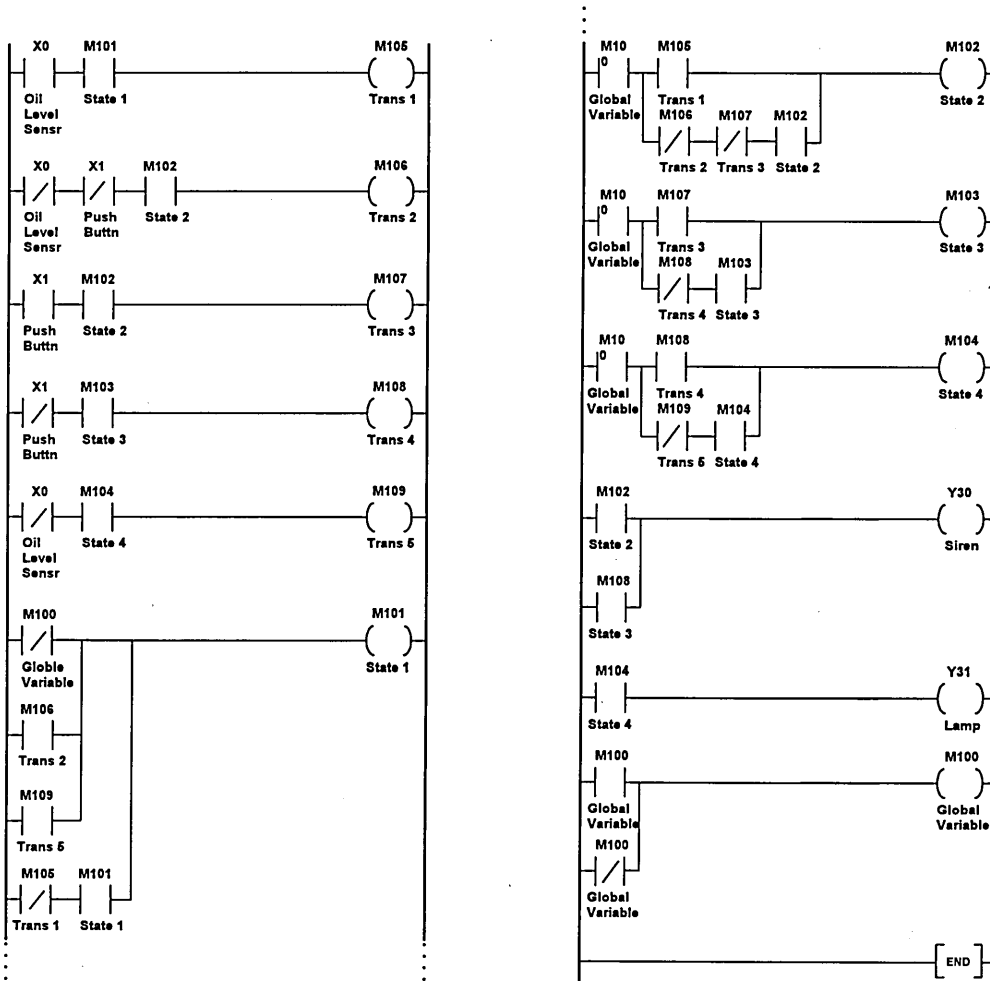


Figure 4-9 LD Translation of the SFC in Figure 4-8

# Chapter 5

## PN↔PLC Applied to the FMC

### Contents

<b>5. PN↔PLC Applied to the FMC.....</b>	<b>73</b>
<b>5.1 PN↔PLC Applied to the Puma Work Station .....</b>	<b>74</b>
5.1.1 Enable Vice to Table PN Group .....	76
5.1.2 PN↔PLC and Timers.....	82
5.1.3 Conveyor Controller .....	84
<b>5.2 PN↔PLC Applied to Miller Work Station .....</b>	<b>84</b>
<b>5.3 PN↔PLC Applied to Lathe Work Station.....</b>	<b>87</b>
5.3.1 Terminology and Symbols (Revised) .....	92
<b>5.4 Error Handling .....</b>	<b>94</b>
5.4.1 Reliability and Safety Achieved by PN↔PLC.....	97
5.4.1.1 Reporting Errors .....	101

## 5. PN $\leftrightarrow$ PLC Applied to the FMC

Having explained the methodology using a simple control problem, in this chapter the application of PN $\leftrightarrow$ PLC to a more complex control system is discussed. PN $\leftrightarrow$ PLC was used to develop the control programs for the three work stations of the FMC. The design rules of the methodology were applied to each work station, and PN designs were developed as shown in Appendices B, C, D and E. The PN designs were verified against the requirements of the FMC using the PN $\leftrightarrow$ PLC simulation rules. Using the translation rules, the PN designs were translated into Ladder Diagrams which were in turn inputted into the PLCs using a proprietary software. Also in this chapter, suggestions are proposed which help the designer to avoid introducing errors in complex sequential Ladder Diagrams.

The overall control algorithm for the level 1 PLCs has been successfully implemented and the three work stations are currently in full operation. As mentioned in Chapter 2, the Flexible Manufacturing Cell has three PLC controlled work stations which are in turn controlled by three work station controllers or Transputers (Figure 2-1, duplicated below for convenience). Each PLC executes its logic programs upon receiving the relevant signals from its controller.

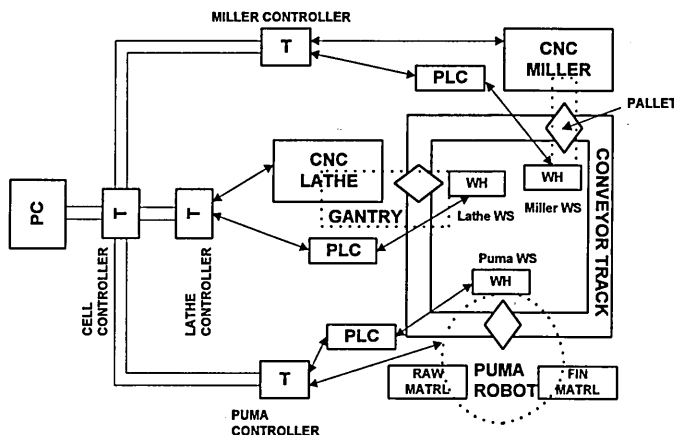


Figure 2-1 Layout of the FMC at the School of Engineering

The PN graph shown in Figure 2-5, inside back cover, produced by Gray's methodology represents the overall control program for the upper level Transputers. The PN graph shows how the various controllers concurrently communicate with one another. However, the graph does not show how these controllers reliably communicate with the lower level PLCs, nor does it give any regard to handling any possible errors during the control of the work stations. PN $\leftrightarrow$ PLC has produced correct and reliable PLC programs for the work stations of the FMC (sections 5.1-5.3), accounted for possible errors during the communication between the upper level controllers and the PLCs (discussed in section 5.4), and, due to the readability of its PN graphs, has produced a unified PN graph of the overall control program of the FMC (Chapter 6).

## 5.1 PN $\leftrightarrow$ PLC Applied to the Puma Work Station

The Puma PLC relies on a signal from the Puma Controller instructing it to perform a certain operation. Referring back to Gray's PN graph (Figure 2-5) it can be seen that only two output signals (instructions) are intended for the Puma PLC, 'Load Robot' and 'Unload Robot'<sup>1</sup>. However, there are several operations involved in loading and unloading parts to and from the Puma WS. For example, there are three operations involved in loading a part:

1. Ram the vice onto the table
2. Close the vice (after a part has been placed by the Puma robot)

---

<sup>1</sup> Gray's 'Load Robot' and 'Unload Robot' terminology is misleading. A clearer terminology would be 'Load Part' and 'Unload Part'

3. Pull the vice onto the pallet (ready to be indexed by the conveyor)

Note that the vice is air operated and will not function unless it is on the table.

Similarly there are two operations involved in unloading a part:

1. Ram the vice onto the table
2. Open the vice (after the part has been gripped by the Puma robot)

It can be seen that there are four unique operations in total and that 'ram vice to table' is needed during both loading and unloading. Therefore for readability, reliability and safety reasons (discussed further in section 5.4.1) the output signal 'Load Robot', i.e. 'Load Part', from the Puma Controller (Figure 2-5) was replaced by three output signals and 'Unload Robot', i.e. 'Unload Part' was replaced by two output signals as follows:

#### 'Load Robot'

1. 'Enable Vice to Table'
2. 'Enable Close Vice'
3. 'Enable Vice to Pallet'

#### 'Unload Robot'

1. 'Enable Vice to Table'
2. 'Enable Open Vice'

These output signals become 'group signals' because they allow the distribution of the overall control program into four individual program groups or PN Groups which are then executed by the PLC in the sequences described above for loading and unloading. The four PN Groups are **Enable Vice to Table PN Group, Enable Vice**



to Pallet PN Group, Enable Close Vice PN Group and Enable Open Vice PN Group.

### 5.1.1 Enable Vice to Table PN Group

PN $\leftrightarrow$ PLC was used to design the PLC program for ramming the vice onto the table.

The approach used to design a Vice to Table PN graph is identical to that of Oil Tank

Level PN graph in Chapter 4 with only a few additional symbols and terminology

used throughout this Chapter. The simpler Oil Tank Level PN graph does not use

these additions, hence, to simplify the introduction of PN $\leftrightarrow$ PLC, they were not

included in section 4.1.2. These additions are not beyond the principals of Petri Nets

or PLCs and are an integral part of PN $\leftrightarrow$ PLC. Therefore the complete set of

terminology and symbols has been included in section 5.3.1, and also in Appendix A.

For safety reasons (section 5.4.1) the designer chooses to perform a few error checks

at the beginning of the PLC program. These error checks are as follows:

1. Make sure that the vice is not jammed part way between the pallet and the table .

There are two sensors for this purpose, one on the pallet and one on the table. If the vice is jammed in-between, neither sensor will be 'On'.

2. Make sure sensors are not faulty. If both sensors are 'On' an error must be reported.

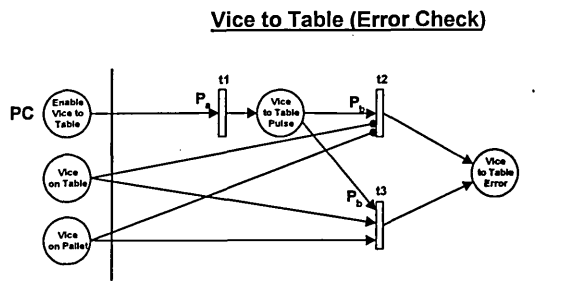
These error checks are only needed at the very beginning of the control sequence and

for the first scan of the PLC program. With Ladder Diagrams this is done by

switching on a 'pulse' (section 2.3.1.5). Similarly with PN $\leftrightarrow$ PLC, the first place

drawn on the PN graph (inside the boundaries) is the 'pulse'. Following the design

rules listed in the Appendix A, the error checking part of the PN graph is achieved as shown below in Figure 5-1.



**Figure 5-1 Initial Error Checks Performed Using a ‘pulse’**

**Vice to Table Pulse** is switched on when the signal from the Puma Controller ‘Enable Vice to Table’ is ‘On’. **Vice to Table Error** is switched on if either of its transitions, **t2** or **t3** which represent the two error conditions described above, are enabled. The additional symbols in the above Figure are the priority labels **P<sub>a</sub>** and **P<sub>b</sub>** and the shading of the place **Vice to Table Error**, which are explained below and are also included in the revised terminology and symbols table in section 5.3.1.

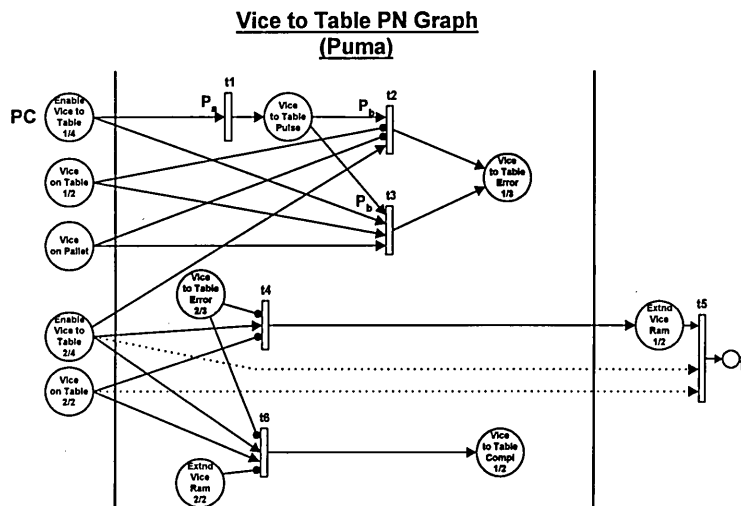
If a transition is labelled with **P<sub>a</sub>** it means it has priority over any other transition, i.e. its firing is considered before that of any other transition. A transition labelled with **P<sub>b</sub>** has priority over all transitions except for a transition labelled with **P<sub>a</sub>**. The illustration in Figure 5-1 correctly represents the setting of a ‘pulse’ and the use of the ‘pulse’ for performing initial error checks, as done by Ladder Diagrams, and is essential to ensure the correct simulation and translation of the PN graph.

The shading of **Vice to Table Error** indicates that this place occurs in, and is switched off in another PN graph. This is done to simplify the PN graph and make it more readable. It can be seen from Figure 5-1 that no ‘switching-off transition’ has been included for **Vice to Table Error**. As suggested by the shading, **Vice to Table**

**Error** has been linked to another PN graph (Figure 5-24, Appendix E) where the switching off transition is situated (discussed in detail in section 5.4.1).

Note that no switching off transition is required for **Vice to Table Pulse** as it is automatically switched off at the end of the first scan, as done by LDs (section 2.3.1.5).

Having achieved the initial error checking part of **Vice to Table PN**,  $PN \leftrightarrow PLC$  is used to complete the control program for ramming the vice onto the table, as shown in Figure 5-2 below. **Extend Vice Ram** is the only output signal needed from the PLC to complete the program.



**Figure 5-2 PN Graph Showing the Use of a 'group signal'**

Using  $PN \leftrightarrow PLC$  an Error Status PN graph was designed to report errors to the Puma Controller (section 5.4.1). However if no errors are detected, the PLC must also output a 'complete' signal informing the controller when it has successfully carried out its task. As with **Vice to Table Error**, the shaded **Vice to Table Compl** indicates that it is linked to another PN graph (Figure 5-23, Appendix E) in which the "switching-off transition" has been considered (also discussed in section 5.4.1). At this stage of the design process it is adequate to note that a **Vice to Table Compl**

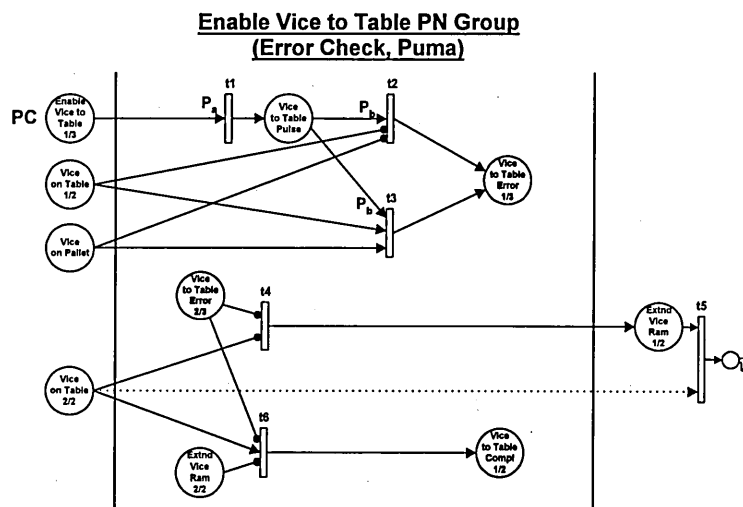
internal relay is switched on if **Vice to Table Error** is 'Not On', **Enable Vice to Table** is 'On', **Vice on Table** is 'On' and the output signal **Extend Vice Ram** is switched off.

It is also important to note that having **Extend Vice Ram** 'Not On' as one of the switching on conditions to **Vice to Table Compl** rules out non-determinism at the design stage. Consider Figure 5-2; without **Extend Vice Ram 2/2** it would not be clear whether the designer intends to switch off **Extend Vice Ram** before switching on **Vice to Table Compl** or vice versa. Moreover, it would present the risk of exiting the program prior to switching off the signal to **Extend Vice Ram** solenoid and would result in unreliable programming. This is a guideline to reliable sequential programming and is further appreciated when designing more complex programs (section 5.3).

As mentioned above, the operations of loading and unloading the Puma WS were divided into program groups to simplify the design process and, in conjunction with PN $\leftrightarrow$ PLC, to improve safety and reliability. Therefore the 'group signal' 'Enable Vice to Table' from the Puma controller is an input condition to every "switching-on transition" in the program **Vice to Table PN Graph** shown above in Figure 5-2. It can also be seen that **Enable Vice to Table** 'On' is one of the conditions for switching **Extend Vice Ram** off. This is essential to ensure that the place is switched off only with the conditions shown in this particular PN graph. For instance, if **Extend Vice Ram** is used by another part of the overall PLC program, then **Extend Vice Ram** would switch off with those other conditions if they happened to be valid. Therefore, in order to avoid such non-determinism, it is important to include the

'group signal' 'Enable Vice to Table' 'On' as the condition to all "switching-off transitions".

Connecting **Enable Vice to Table** to all transitions transforms the program **Vice to Table PN Graph** in Figure 5-2 to an independent program group. It does however increase the number of arcs crossing, and hence reduces the readability of the PN graph. Therefore the PN graph is simply labelled as a PN Group and all arcs from **Enable Vice to Table** are omitted as shown below in Figure 5-3. This also simplifies the simulation process, knowing that no condition in any other group will affect the PN Group considered. When translating, care must be taken to include **Enable Vice to Table** on every rung of the Ladder diagram.

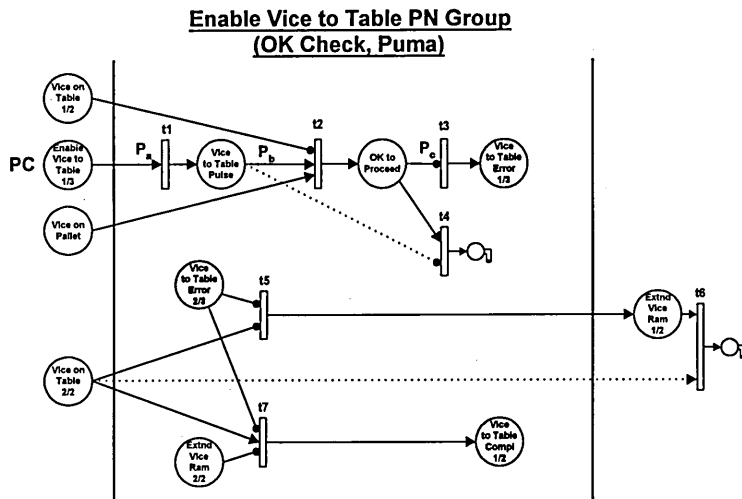


**Figure 5-3** PN Group Containing Initial Error Checks

Note that **Enable Vice to Table** to transition **t1** has not been omitted. This is done to show that **Enable Vice to Table** is the 'group signal' that initiates the PN Group.

PN Grouping is much the same as using Step Ladders, which are supported by some PLCs, without the need to use an internal step relay. However, PN $\leftrightarrow$ PLC can also be used to produce PN Steps, which are directly equivalent to Step Ladders (section 5.3).

The final consideration given to the simplification of the PN Group in Figure 5-3 is that of the initial error checking part. Consider Figure 5.4 below:



**Figure 5-4 PN Group Showing an Alternative Error Check**

The two error conditions denoted by **t2** and **t3** in Figure 5-3 are replaced by one OK condition denoted by transition **t2** and the internal relay **OK to Proceed**, as shown in Figure 5-4. It is safe to say that if initially the vice is on the pallet and not on the table, it is OK to proceed with the program. If these OK conditions are not satisfied, **OK to Proceed** will not be switched on and hence **Vice to Table Error** will switch on. Since **OK to Proceed** is only needed for the first scan of the program, it is simply switched off with the condition **Vice to Table Pulse** 'Not On', as shown by **t4** in Figure 5-4 above. The advantages and disadvantages of both representations are discussed in section 5.4.1.1.

Using  $PN \leftrightarrow PLC$ , **Enable Close Vice PN Group**, **Enable Vice to Pallet PN Group** and **Enable Open Vice PN Group** were also designed as shown in the Appendix B by Figures 5-6, 5-7 and 5-8 respectively. The Ladder Diagram translations of the Puma Work Station PN Groups are shown in Appendix F.

### 5.1.2 PN⇌PLC and Timers

PN⇌PLC can also incorporate PLC timers as shown in Figure 5-8, duplicated below for convenience. The vice uses an over-centre or toggle clamp and a pressure switch which senses that the vice is in the closed position. However, this pressure switch does not give a positive vice open signal. The interpretation of the input place **Vice Closed** 'Not On' as **Vice Open** is considered unsafe, as it can also mean that the vice is opening, or is closing. Therefore, the use of **Vice Closed** 'Not On' as the switching off condition for **Open Vice** (transition **t9**) would present the risk of switching this output off before the vice has fully opened and result in the unsafe operation of the vice. Due to the lack of **Vice Open** sensor, the designer has introduced **Vice Open Timer**, which is of type 'delay on'. A 'delay on' timer is represented in the same way as an internal relay.

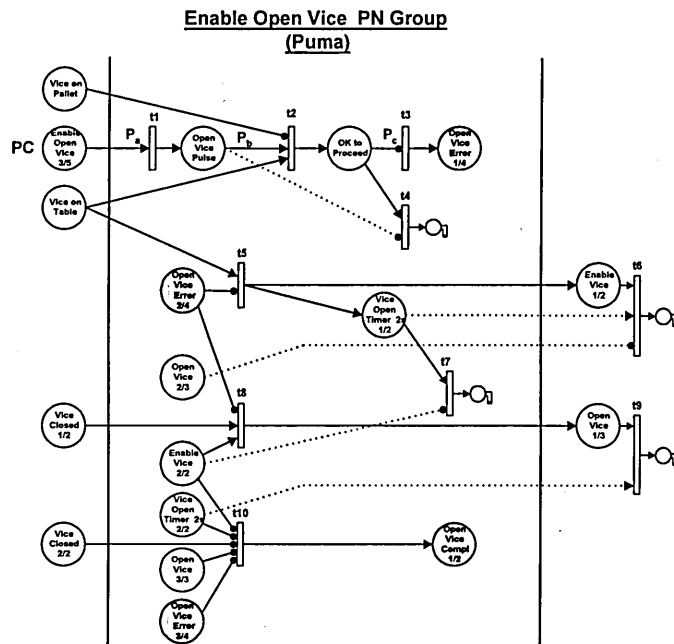


Figure 5-8 A PN Group Showing the Use of a Timer

Using the design rules of PN⇌PLC (Appendix A), **Vice Open Timer 2s** is switched on via the “switching-on transition” **t5**, where **Vice on Table** ‘On’ *AND* **Open Vice**

**Error** 'Not On' are its "switching-on places", and is switched off via the "switching-off transition"  $t_7$ , where **Enable Vice** 'Not On' is its "switching-off place". However, in order to correctly use the timer to perform the intended logic, the manner in which the timer operates must be understood:

A 'delay on' timer will switch on only after the specified delay time has elapsed, provided the conditions for switching it on remain unchanged for that duration. For example in Figure 5-8, **Vice Open Timer 2s** has a two second delay and therefore will switch on after two seconds, provided its switching-on conditions remain true for two seconds. In simulation terms, if **Vice on Table** has a token and **Open Vice Error** does not have a token, and if these conditions remain unchanged for two seconds, **Vice Open Timer 2s** will receive a token after the two seconds have elapsed<sup>1</sup>.

However, once the timer is switched on it will remain 'On', regardless of its switching-on conditions, until it is switched off via the "switching-off transition".

This further highlights the claims of the methodology to be readable, reliable and safe. This differentiation of the "switching-on" and "switching-off transitions", together with the remaining rules and symbols of  $PN \Leftrightarrow PLC$  provides the designer with a methodical approach to developing correct PLC programs. Furthermore, it promotes the overall readability of the PN graph and allows the simulation of the program correctly, i.e. as executed by the PLC.

A 'Delay off' timer has the inverse timing function of a 'delay on' timer, in that if the conditions for switching it on are true, the timer will switch on instantly, i.e. it will receive a token instantly. Using  $PN \Leftrightarrow PLC$  a 'delay off' timer is represented in much



the same way as a 'pulse'. Similar to a 'pulse', there is no need to include a "switching-off transition" for such timers. Whereas a 'pulse' is automatically switched-off at the end of the first scan, a 'delay off' timer switches off automatically after the specified delay time has elapsed. For example, if it is labelled '2s', the timer will automatically switch off, or lose its token, after two seconds.

### 5.1.3 Conveyor Controller

Referring back to Gray's PN graph (Figure 2-5, inside back cover), it can be seen that an independent controller has been allocated to the conveyor (CONVC) in parallel with the Cell Controller, Lathe, Miller and Puma Controllers and Status Handler. However, the Puma robot cannot load and unload parts from the Puma WS while the conveyor is being indexed and vice versa. Since the Conveyor Controller cannot function concurrently with the Puma Controller, there is no need to allocate an independent controller for the conveyor and therefore the task of indexing the conveyor has been given to the Puma Controller. The output signal 'Start Conveyor Index' from the CONVC was replaced by 'Enable Conveyor Index' and **Enable Conveyor Index PN Group** was achieved as shown in Figure 5-9, Appendix B.

## 5.2 PN $\leftrightarrow$ PLC Applied to Miller Work Station

The approach used to design the control program of the Miller WS is identical to that of the Puma WS. As can be seen from Gray's PN graph (Figure 2-5), three instructions are intended for the Miller WS PLC, 'Load Miller', 'Start Miller' and 'Unload Miller'. The operations involved in performing these instructions are as

---

<sup>1</sup> The output place **Enable Vice** receives a token instantly.

follows:

'Load Miller'

- 'Enable Vice to Table'

This involves pushing the vice, in which a part is held, onto the Miller table with a pneumatic cylinder. A fixture has been connected to the miller table to allow the vice to be guided to a fixed position, using a long pneumatic cylinder situated at the Miller WS. A pneumatically operated toggle clamp is then used to secure the vice once it is in this position .

'Start Miller'

- 'Enable Start Miller'

This is an instruction to the Miller to start its milling program.

'Unload Miller'

- 'Enable Vice to Pallet'

This involves pulling the vice back onto the pallet using the long pneumatic cylinder. Note that when the vice is on the Miller table there will be a redundant pallet on the conveyor, and it is this same pallet which must be indexed back to the Miller WS to accept the vice.

It can be seen that the correct number of output signals have been allocated by Gray's PN graph to control the Miller WS. However, it was the designer's choice to change the description of these output signals for the purpose of readability. Therefore the three output signals from the Miller Controller in Figure 2-5 were changed to the 'group signals' 'Enable Vice to Table', 'Enable Start Miller' and 'Enable Vice to

Pallet' and hence the three PN Groups 'Enable Vice to Table PN Group', 'Enable Start Miller PN Group' and 'Enable Vice to Pallet PN Group' were designed as shown in Appendix C by Figures 5-10, 5-11 and 5-12.

Figure 5-11 is duplicated below for convenience. It can be seen that a small PN Graph entitled 'Milling PN Graph' is also included. This is not part of 'Enable Start Miller PN Group', nor does it rely on an instruction signal from the Miller Controller. It is however part of the overall program for the Miller WS, and a necessary signal to the Miller Controller. For instance if **Enable Start Miller PN Group** has been successfully executed and the signal **Milling in Progress** is switched on, the Miller Controller understands not to send the signal 'Unload Miller' or 'Enable Vice to Pallet' until **Milling in Progress** is switched off. Although independent from **Enable Start Miller PN Group**, it is appropriate to show **Milling PN Graph** in Figure 5-11.

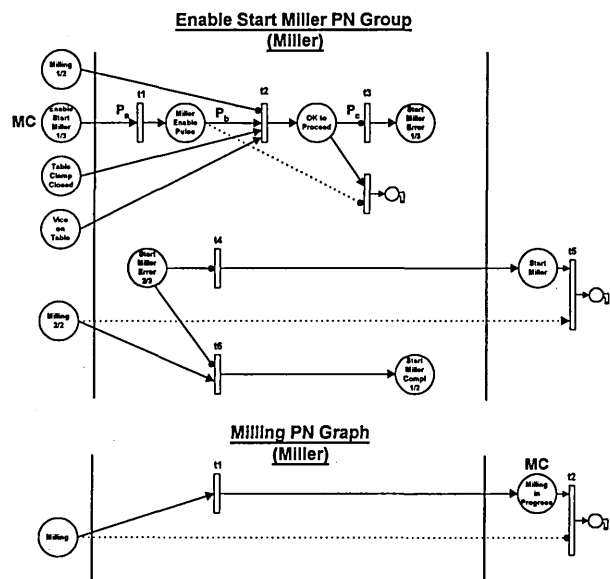
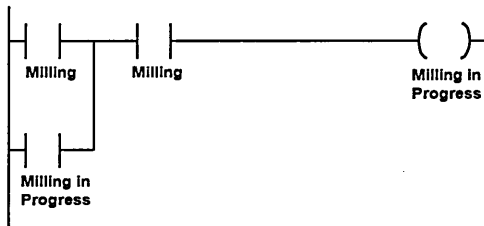


Figure 5-11 Enable Start Miller PN Group

**Milling PN Graph** is also used to emphasise the readability and reliability of PN $\leftrightarrow$ PLC. The graph simply reads as follows:

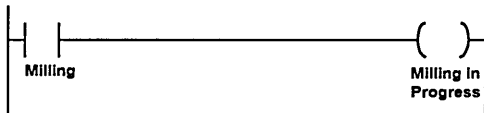
The signal **Milling in Progress**, to the Miller Controller, will be switched on if the input signal **Milling**, from the Miller, is 'On'. **Milling in Progress** will remain 'On' until **Milling** has switched off.

The Ladder Diagram translation of **Milling PN Graph** is shown below in Figure 5-13.



**Figure 5-13 Ladder Diagram Translation of 'Milling PN Graph'**

This representation may be considered to be unnecessary as a similar logic can be achieved by the basic LD shown below in Figure 5-14.



**Figure 5-14 A Ladder Diagram Equivalent to Figure 5-13**

However, the formalism of the methodology dictates that the PN Graph in Figure 5-11 translates into the LD in Figure 5-13. Keeping to the same format for rungs makes the LD more readable and gives confidence that the translation has been conducted correctly.

### 5.3 PN $\leftrightarrow$ PLC Applied to Lathe Work Station

Similar to the Miller WS, Gray has allocated three instructions for the Lathe PLC, 'Load Lathe', 'Start Lathe' and 'Unload Lathe', as shown by Figure 2-5. 'Load Lathe' and 'Unload Lathe' are intended for the Gantry Robot, while 'Start Lathe' is for the CNC Lathe instructing it to execute its CNC code. Figures 5-15 through 5-22,

Appendix D, show that the control program for the Lathe WS is lengthy and complex, therefore for safety and reliability reasons, the output signals from the Lathe Controller were divided into the following:

'Load Lathe'

1. 'Enable Vice to Table'
2. 'Enable Grip Workpiece'
3. 'Enable Workpiece to Chuck'
4. 'Enable Robot to Safe'
5. 'Enable Vice to Pallet'

'Start Lathe'

1. 'Enable Start Lathe'

'Unload Lathe'

1. 'Enable Vice to Table'
2. 'Enable Workpiece to Conveyor'
3. 'Enable Workpiece to Vice'
4. 'Enable Robot to Safe'
5. 'Enable Vice to Pallet'

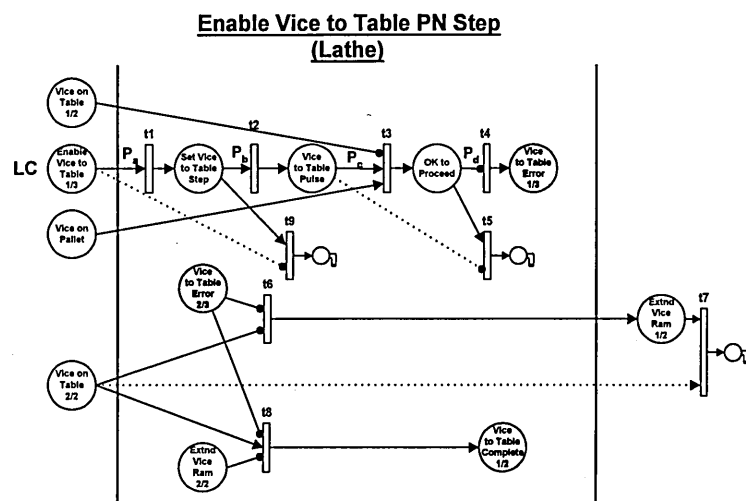
It can be seen that there are eleven operations in total. However, 'Enable Vice to Table', 'Enable Robot to Safe' and 'Enable Vice to Pallet' occur during both 'Load Lathe' and 'Unload Lathe', and therefore the overall control of the Lathe WS has eight independent PN graphs. This also simplifies the overall design process and promotes safety because the independent PN graphs can be reliably checked.

As mentioned above in section 5.1,  $PN \leftrightarrow PLC$  can also be used to design PN Steps, which are directly equivalent to Step Ladder Diagrams (section 2.3.1.5.1).

The approach used to design PN Steps is similar to that of PN Groups, with only two simple adaptations, as follows:

1. Setting and resetting a Step Relay.
2. Simplification of the PN Step.

Figure 5-15 is duplicated below and is used to explain this.



**Figure 5-15 Enable Vice to Table PN Step**

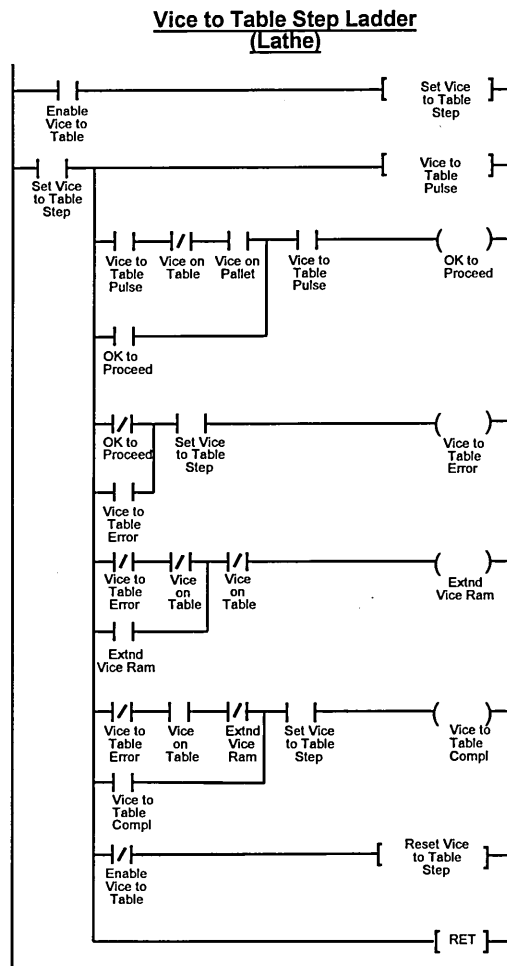
Comparing **Enable Vice to Table PN Step**, Figure 5-15, with **Enable Vice to Table PN Group** shown in Figure 5-5 (Appendix B), it can be seen that the place **Set Vice to Table Step** and its “switching-off transition”, **t9**, are the only additions to Figure 5-5.

As suggested by **P<sub>a</sub>**, it is clear that the designer intends to switch on an internal ‘Step relay’ on the very first rung of the LD. It can also be seen that the designer intends to reset the Step when the signal **Enable Vice to Table**, from the Lathe Controller, has been switched off.

By having **Set Vice to Table Step 'On'** as one of the conditions for switching all outputs on ensures that the outputs will only switch on by the conditions stated in this PN graph, as done by Step LDs. However, similar to PN Groups, the labelling of the PN graph as a PN Step eliminates multiple crossing arcs, and enhances the readability of the program.

Note that it is not necessary to include the Step Relay as one of the switching off conditions of all the outputs. This is because once the program is in a step, it will remain in that particular step until it is reset, and therefore only the switching off conditions stated in that same step will affect the outputs, as done by Step Ladders.

When translating PN Steps, care must be taken to include the Step Relay on every switching on rung of the Step LD. The PN Step in Figure 5-15 was translated using the PN $\leftrightarrow$ PLC rules and is shown below in Figure 5-15L.



**Figure 5-15L Step Diagram Translation of Figure 5-15**

**Enable Vice to Table** 'On' switches **Set Vice to Table Step** on, as shown on the first rung. **Enable Vice to Table** 'Off' resets **Set Vice to Table Step**, as shown on the last rung. This setting and resetting of the Step Relay is the most commonly practised representation in Step Ladder Programming.

Note that the switching off conditions for both **Vice to Table Complete** and **Vice to Table Error** have also been included. These conditions are shown in Figures 5-27 and 5-28 in Appendix E.


The control program for the Lathe WS can also be developed using PN↔PLC PN Groups. PN Steps were used to demonstrate the versatility of PN↔PLC, and are included in Appendix D.



### 5.3.1 Terminology and Symbols (Revised)


The Terminology and Symbols table shown in Chapter 4 (section 4.1.2) has been revised, due to the additional symbols and terminology used in this Chapter.


In order to make the PN graph readable, certain symbols have been adopted as follows:

 is a "**return arc**" which shows that an input place will receive its token back after the transition has fired. For example this applies to input signals over which the PLC has no control, such as signals from sensors. However, this symbol can be ignored once the following two points are appreciated:

1. Tokens are removed from an output **place** or an internal **place** only via the "switching-off transition" of that **place**.
2. Input signals, such as those from sensors, will not lose their tokens as a result of their output **transitions** firing.

Consider Figure 5-5 in Appendix B. **Extend Vice Ram** will lose its token as a result of **t6** firing. However, **Vice on Table** will not lose its token via **t6**, nor will it do so via **t7**. During simulation the token will be manually removed from **Vice on Table** only if the vice is no longer on the table.

 are "**switching-off arcs**" which clearly show what is being switched off and what is doing the switching off. Combined with design rule 2, the "switching-off arcs" show exactly what the designer has in mind.



 are duplicated places. Where a place is the input to several transitions, it can be duplicated in order to reduce the number of arcs that cross. This simplifies the

overall PN graph and improves readability. It is, however, essential that a tally is kept of the number of duplicate places (1/2, 2/2) so that they are not overlooked by the designer and simulator.



is a “**dummy**” or “**drain place**”, which is used to demonstrate that a token is drained from a particular place as a result of its transition firing. This symbol clearly shows the “switching-off transitions”, and proves that the designer has considered how each output and internal relay will be switched off.



is a **transition** which has priority over any other **transition**,  has priority over any **transition** except for , and so on.



is a shaded place and indicates that it occurs in and is switched off in another PN graph.

**PN Grouping** is used to simplify the PN graph and make it more readable. When an input signal is connected to every “switching-on” and “switching-off transition” it transforms the PN graph to a PN Group. Therefore PN graph is entitled PN Group and arcs from the input signal are not drawn to each transition.

**PN Steps** are used when designing Step Ladders. In a Step LD the Step Relay is connected to all “switching-on transitions” to ensure that outputs are only switched on and off by the conditions shown in that particular Step. Therefore to simplify the PN

graph and make it more readable, it is entitled PN Step and arcs from the Step Relay are not drawn to each “switching-on transition”.

## 5.4 Error Handling

The research carried out by Gray has produced a methodology which aims to produce dependable software for a Distributed Control System. The FMC at the SOE forms the basis of Gray's research but no consideration has been given by Gray to any errors which are likely to occur within the cell. However Gray claims that the methodology simplifies the modular growth of the control software, in an effort to build resilience into the complete system.

As well as producing PN $\leftrightarrow$ PLC and unifying it with Gray's methodology (Chapter 6) the reported work also gives some consideration to the resilience of the overall control system. The identification of possible errors within the system and the handling of such errors forms a major part in the production of a resilient system.

The following possible errors, which may occur during the operation of the cell, have been identified and categorised as below:

### Work Handling Errors

- Failure of sensors, solenoids, valves, etc.
- Manipulating errors (positioning, speed, etc.)
- Material errors (size, shape, alignment, position, etc.)
- Robot failure
- Power failure (air, electricity)

### **Machine Tool Errors**

- Catastrophic failure
- Gradual failure (wear)
- Work handling (fixturing) failure
- Tool failure
- CNC program errors

### **Control Errors (Hardware and Software)**

- Correctness, reliability
- Communication errors (timing of messages, message corruption, “deadlock”)
- Initial data errors
- Logic of control
- Speed of software
- Failure of hardware (PLCs, Transputers, trams, digital I/O)
- Power failure

### **Human Errors**

- Misuse, inappropriate use
- Mistakes (incorrect choice of material, tools etc.)

Having identified the possible errors which are likely to occur during the control of the Cell, it is important to discuss the various methods by which these errors are handled.

Ideally the elimination of the chance of errors occurring is most favourable. In reality this is not always possible due to the unpredictability of some errors. For instance, in the case of gradual failure of machine tools, efforts such as constant preventative maintenance and statistical process control will help eliminate errors. However a work handling device, such as an Infra-Red sensor, could fail suddenly and therefore needs to be accounted for by the PLC program to avoid catastrophic consequences.

It is also essential to establish a good understanding of the terms **successful** and **error** during the control of a resilient FMC. This is best explained with an example. The Cell Controller instructs the Lathe Controller to load the lathe which in turn instructs the Lathe PLC. A fault with the material results in the diameter of the bar, for instance, in the vice being smaller than specified. Although failing to grip the work piece in reality, the Gantry robot will carry out the operations “Grip Workpiece” and “Workpiece to Chuck” successfully, hence transporting “fresh air” which in turn is machined by the Lathe. Such control may be dependable, and even safe for that matter, it is not however in any way resilient.

Other considerations such as deciding a safe position for the Gantry robot is also important. It was decided not to leave the arm of the robot directly above the Lathe after transporting a work piece to the chuck. Tests showed that in the event of air supply failure, the robot arm will only remain in the “Up” position for a limited amount of time before rapidly falling. This would result in the arm breaking the glass door at the top of the Lathe and colliding with the rotating chuck. Therefore an additional PN Step was designed in the programming of the Gantry robot, taking the arm of the robot to a much safer position, above the conveyor.

However not all errors can be dealt with in Level 1. Communication errors such as message corruption and “deadlock” , and even failure of hardware such as PLCs and Transputers are classed as control errors and should be accounted for by the software in the top levels of the control. Human rectification, or rectification by the operator is also an important part of the error handling process and needs to be carefully dealt with by the Cell Controller at the decision making stage.

#### **5.4.1 Reliability and Safety Achieved by PN $\leftrightarrow$ PLC**

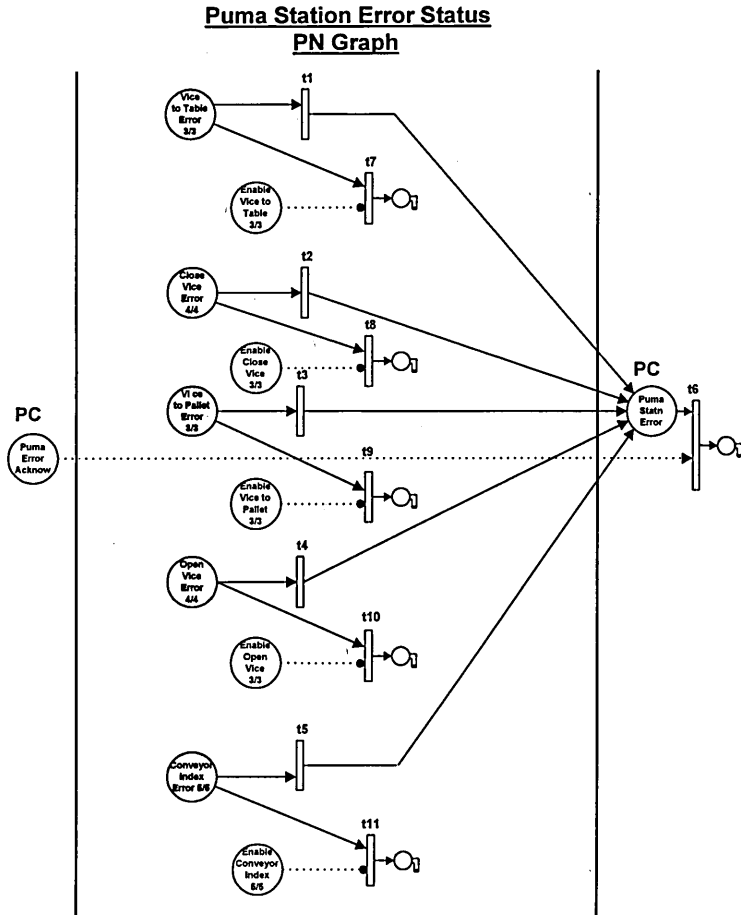
As mentioned above, the correctness and reliability of control software plays a big roll in the overall resilience of DCSs such as in an FMC. To achieve this, the correct understanding of the control problem is vital and the use of the appropriate tools and techniques essential. This very important point is appreciated when considering Gray’s PN-Occam methodology. Although successful at producing dependable Occam based control programs for the higher level Transputers, Gray’s methodology is not applicable to the lower level PLCs (Chapter 3). Furthermore, the reported research has shown that no consideration has been given to the reliable communications between the work station controllers and their relevant work handling (WH) equipment.

Achieving readability and simplicity of design is a major step towards producing resilient control programs. Referring back to Figure 2-5, and as mentioned in sections 5.1-5.3, signals such as ‘Load Robot’, ‘Load Miller’ and ‘Load Lathe’ from the Level 2 Transputers were divided into various ‘group signals’ relating to the unique operations involved in loading the machine tools and the WH devices. Subsequently, these ‘group signals’ were used to develop the PN Groups and PN Steps which, when

executed in the correct order, will perform the overall loading operations, as intended by Figure 2-5. This approach, together with PN $\leftrightarrow$ PLC, simplifies the design process, and produces readable programs. It allows a lengthy and complex program to be divided into smaller programs which are simple to design and subsequently read. Moreover, it promotes reliability and safety. The smaller PN graphs can be reliably checked, through simulation, prior to execution and can be reliably upgraded to achieve flexibility. This is also safer as errors can be reported and traced with greater ease and speed. Consider the instruction, from the Lathe Controller, 'Load Lathe'. This is a lengthy and complex process during which five individual operations are involved, as shown in Appendix D by Figures 5-15 through 5-19. The design of these individual PN Steps is far simpler, and their readability greater, than that of an overall 'Load Lathe' PN graph. They can be more reliably checked and updated than an overall 'Load Lathe' PN graph. Furthermore, errors which occur during the control of the WH devices can be accounted for during each independent operation, and hence traced back quickly and with great ease.

As mentioned in section 5.1.1, PN $\leftrightarrow$ PLC is also capable of performing WH error checks, such as sensor failure and manipulating errors. In designing the control programs for the three work stations, certain critical checks were carried out to ensure safety, as shown in Figures 5-5 through 5-22. For example in Figure 5-5, if the vice is jammed part way between the table and the pallet, it is considered unsafe to proceed with the operation **Enable Vice to Table**. In such an instance, an internal relay **Vice to Table Error** is switched on thus preventing the program from proceeding further. However, as suggested by the shading (section 5.3.1), such relays are dealt with in a separate PN for the purpose of readability. Therefore **Error Status PN Graphs** were

generated for each work station (Appendix E), in which it is shown how the errors are reported and switched off, as shown below in Figure 5-24.



**Figure 5-24 Puma Station Error Status PN Graph**

Figure 5-24 represents the Error Status PN Graph for the Puma WS. Initial safety checks are performed within each PN Group, Appendix B, and are included in Figure 5-24, as indicated by the shaded places. An error signal ‘Puma Statn Error’ is outputted to the Puma Controller in the case of any of the internal error relays switching on, hence the five “switching-on transitions” indicated by **t1**, **t2**, **t3**, **t4**, and **t5** shown in Figure 5-24. Note that non-determinism is not present in this representation as only one of the five “switching-on transitions” can be enabled at any one time. If an error occurs during any of the five PN Groups and ‘Puma Statn Error’ is outputted, the Puma Controller (Figure 6-1, inside back cover) then switches off the

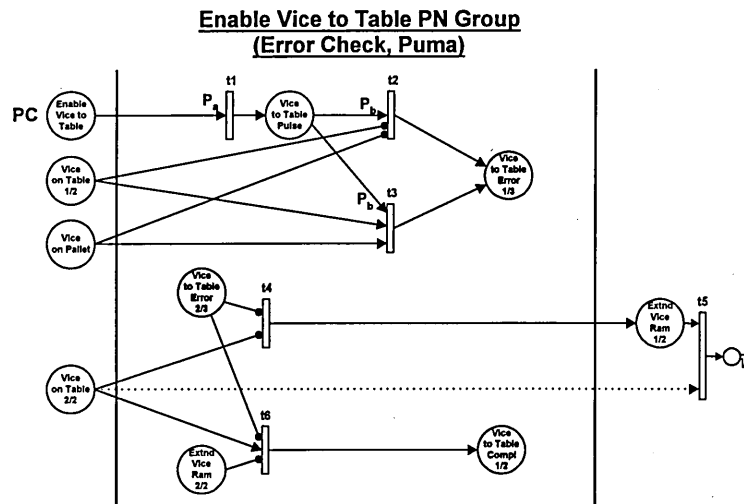


'group signal' to that particular PN Group as part of its error handling process, which in turn switches off the internal error relay of that PN Group (discussed in more detail in Chapter 6). For example, if **Vice to Table Error** is 'On', 'Puma Statn Error' is switched on via transition **t1** (Figure 5-24). On receiving this signal, the Puma Controller then switches off 'Enable Vice to Table' which in turn switches off **Vice to Table Error** (**t7**, Figure 5-24). To switch off 'Puma Statn Error' the Puma Controller then outputs the signal 'Puma Error Acknow', acknowledging that the error signal has been received, as shown in Figure 5-24. The Error Status PN graph ensures that critical errors during the control of the Puma WS are reliably reported to the Puma Controller and subsequently dealt with. These errors can be efficiently traced back to the relevant PN Groups and hence corrected quickly. It also promotes safety as PN Groups cease to continue with their control logic until the errors are rectified. The Error Status PN Graphs designed for the Miller and Lathe WSs are shown in the Appendix E.

As well as Error Status PN Graphs, Complete Status PN Graphs have also been designed. Similar to internal error relays, internal complete relays are also incorporated into PN $\leftrightarrow$ PLC. Continuing with the Puma WS, an internal complete relay is switched on at the end of each PN Group, provided they successfully carry out their control function. For example, if **Vice to Table Compl** is 'On', 'Puma Statn Compl' is switched on and is outputted to the Puma Controller, as shown below in Figure 5-23. This signal is then used by the Puma Controller to switch off the 'group signal' (discussed in more detail in Chapter 6), which in turn switches off **Vice to Table Compl** (**t7**, Figure 5-23). 'Puma Statn Compl' is switched off as a result of receiving the output signal 'Puma Compl Acknow', from the Puma Controller.



5.1.1,  $PN \Leftrightarrow PLC$  can also be used to carry out multiple error checks and report their exact nature. Consider Figure 5-3, duplicated below for convenience. This representation is capable of reporting an error caused by neither sensor being 'On', or both being 'On'.



**Figure 5-3 PN Group containing initial error checks**

It is essential to assess the extent in which errors are dealt with when aiming for resilient DCSs. This varies from one system to another, depending on the consequences of the errors occurring. As it stands, the PLC programs for the FMC report errors reliably, but the operator must then look at the OK conditions and try and work out which one caused the error. This is relatively simpler to produce at the design stage, at the expense of taking longer to fix the problem when it happens. Using the other method may need more effort at the design stage (for example Figure 5-16), but can pin point the exact error, perhaps a faulty sensor, or incorrectly positioned pallet, and thus allow problems to be addressed more efficiently.

However, the physical constraints on the number of output contacts on the PLC prescribed that only one error signal could be outputted to the controllers. Gray claims that the modularity of his methodology allows an Error Handler to be added,

concurrently with the current controllers, with relative ease, but does not give an example. This is certainly an area which would be worth investigating further.

## Chapter 6

### Unification of the Methodologies

#### Contents

<b>6. Unification of the Methodologies.....</b>	<b>105</b>
<b>6.1 The Graphical and Logical Unification of the Methodologies .....</b>	<b>105</b>
<b>6.2 The Unification of the Methodologies for Simulation .....</b>	<b>109</b>
<b>6.3 The Unification of the Methodologies for Translation.....</b>	<b>111</b>
<b>6.4 Fault Avoidance and Elimination Achieved by the Unified Methodology</b>	<b>112</b>
<b>6.5 Discussion .....</b>	<b>113</b>

## 6. Unification of the Methodologies

This chapter describes how Gray's Petri Net - Occam methodology and the reported  $PN \Leftrightarrow PLC$  methodology can be unified to produce an overall methodology for developing resilient DCSs.

Gray's Petri Net - Occam based methodology for developing dependable DCSs was studied and his claims of expandability were investigated. Gray's methodology was found to be incomplete, because it does not consider the lower levels of DCSs (Chapter 3). The readability of his methodology motivated the research into the applicability of his Petri Net approach to PLC programming, which resulted in a  $PN \Leftrightarrow PLC$  methodology (Chapters 4 & 5) which is also readable and allows reliable PLC programs to be produced, directly from the specifications. The readability and modularity of both methodologies results in an overall unified methodology for developing a complete DCS and is discussed in the following sections.

### 6.1 The Graphical and Logical Unification of the Methodologies

Using  $PN \Leftrightarrow PLC$  the overall PLC programs for the three low level Work Stations were produced (Chapter 5). In order to link the  $PN \Leftrightarrow PLC$  graphs to Gray's high level PN graph (Figure 2-5, inside back cover), some modifications had to be made to the communication signals from the Level 2 Controllers. This was because Gray had not considered in detail all the communications that would be required with the PLCs for reliable control. The Puma Controller is used (Figure 6-1, inside back cover) as an example to demonstrate that Gray's PN graph can be updated, to accommodate the communications required by  $PN \Leftrightarrow PLC$ , i.e. group signalling and reporting error and complete messages. The Robot Controller, part of Figure 2-5, was revised using

Gray's methodology, as shown in Figure 6-1. Note that the terminology used by Gray was also changed for the purpose of readability and clarity. For example the title of the controller was changed to Puma Controller, because it controls both the Puma Robot and the Puma Work Station (WS) and includes the conveyor (section 5.1.3). Also for readability, the word Robot inside places 'Load Robot', 'Unload Robot', 'Robot Loaded' and 'Robot Unloaded' was changed to the word Part (section 5.1). The PLC programs for the Puma WS, designed using PN $\leftrightarrow$ PLC, are also included in Figure 6-1 to show the readability and modularity of the unified methodology. The individual parts of the PLC program are represented concurrently with the Puma Controller, thus producing a readable software design for the overall system.

The graphical and logical unification of the two methodologies is reliable and efficient. Outputs from the level 2 controller (Puma Controller) are the inputs to the level 1 controller (PLC), and the outputs from the level 1 controller are the inputs to the level 2 controller. Also as a result of the readability of the methodologies, the overall design is expanded efficiently, without the need to re-design the whole control system. For example, in Figure 6-1 the first input to the Puma Controller 'Load Part' is from the Cell Controller and therefore remains unchanged. The output signal 'Part Loaded' is to the Status Handler and is also unchanged. However, the control logic in between these two signals was upgraded by PN $\leftrightarrow$ PLC, and therefore only this part of the control was updated.

Figure 6-1 shows that the first output from the Puma Controller, 'Load Robot' (Figure 2-5, meaning 'Load Part'), has been replaced with the signals **'Enable Vice to Table'**, **'Load Vice'**, **'Enable Close Vice'**, **'Robot to Safe'** and **'Enable Vice to Pallet'** respectively. These are outputs from the Puma Controller to the Work Handling PLC

and the Puma robot, and their successive execution corresponds to 'Load Part'. The 'group signals' (section 5.1) to the PLC are clearly labelled with meaningful names inside the places, i.e. '**Enable Vice to Table**', and the communication is also clearly identified by the letters **PLC** outside the places. Signals to and from the Puma robot are represented by double circles, for example '**Load Vice**'. The double circle is part of Gray's methodology which means that '**Load Vice**' is an executable program, written in a language specific to the Puma robot. Communications with the robot are also shown, by the letter **R** (for robot).

For reasons mentioned in section 5.1.3, the indexing of the conveyor is also included in Figure 6-1. 'Start Conveyor Index' becomes the second input signal from the Cell Controller to the Puma Controller, and its control logic is updated to accommodate the communications required by  $PN \Leftrightarrow PLC$ , as indicated by transitions **t10** to **t12**. The PN Group '**Enable Conveyor Index**' is also shown in Figure 6-1.

The third input from the Cell Controller to the Puma Controller is 'Unload Part'. Its control logic is updated for the same reasons and in exactly the same fashion as 'Load Part', and is shown in Figure 6-1 by transitions **t14** to **t20**.

Transitions **t13** and **t21** are unaffected by  $PN \Leftrightarrow PLC$  and thus their representation is unchanged.

The two PN graphs entitled **Puma WS Complete Status PN Graph** and **Puma WS Error Status PN Graph** were designed using  $PN \Leftrightarrow PLC$  for safety and reliability purposes (section 5.4.1) and are shown in Figure 6-1. The task of **Puma WS Complete Status PN Graph** is to inform the Puma Controller that it is safe to proceed with its control logic, whereas the task of **Puma WS Error Status PN**



**Graph** is to terminate any ongoing instructions to the work handling and to report the errors to the Status Handler. These two PN graphs are also incorporated into the Puma Controller PN Graph as follows:

For example, when the Puma Controller receives the signal 'Load Part' from the Cell Controller, **Enable Vice to Table** switches on via transition **t1** (Figure 6-1, Puma Controller PN Graph), and is outputted to **Enable Vice to Table PN Group**. If **Enable Vice to Table PN Group** is successfully executed, the internal relay **Vice to Table Compl 1/2** switches on via transition **t7** (Figure 6-1, Enable Vice to Table PN Group). As a result, **Puma Statn Compl** switches on via transition **t1** (Figure 6-1, Puma WS Complete Status PN Graph) and is outputted to the Puma Controller. This is shown as the second input place to the Puma Controller. **Enable Vice to Table** switches off and **Puma Compl Acknow** switches on via transition **t2** (Puma Controller PN Graph). Note that **Puma Statn Compl** is connected to **t2** with a return arc. This is because it is a signal from the PLC and is therefore switched off in **Puma Station Complete PN Graph**. 'Puma Compl Acknow' is outputted to **Puma WS Complete Status PN Graph**, and is shown as its only input place. **Vice to Table Compl 2/2** switches off via transition **t7** and **Puma Statn Compl** switches off via transition **t6** (Figure 6-1, Puma WS Complete Status PN Graph). **Puma Compl Acknow** switches off via transition **t3** (in Puma Controller PN Graph), and the next instruction signal, 'Load Vice' to the robot, is outputted. When the Puma Controller receives the input signal 'Vice Loaded', from the robot, the second 'group signal' to the PLC, 'Enable Close Vice' is outputted. This procedure can be followed through until transition **t9** fires and the signal 'Part Loaded' is outputted to the Status Handler (from the Puma Controller PN Graph, Figure 6-1).

As well as **Puma WS Complete Status PN Graph**, **Puma WS Error Status PN Graph** is also incorporated into **Puma Controller PN Graph**, as shown by transitions **t22** to **t31**. As mentioned in section 5.4.1, if an error occurs during the execution of any of the PN Groups, the signal ‘Puma Statn Error’ is outputted to the Puma Controller. For example, if an error occurs during **Enable Vice to Table PN Group**, **Vice to Table Error 1/3** switches on via transition **t3** and the Group proceeds no further. As a result **Puma Statn Error** switches on via transition **t1** and is outputted to the Puma Controller (from the Puma WS Error Status PN Graph, Figure 6-1). This is shown as the input place **Puma Station Error** to the Puma Controller. Consequently **Puma Error Acknow** switches on and **Enable Vice to Table** switches off via transition **t22**. ‘Puma Error Acknow’ is outputted to **Puma WS Error Status PN Graph** which then switches off **Puma Statn Error** via transition **t6**. In **Puma Controller PN Graph**, **Puma Error Acknow** is switched off and **Error Vice to Table**, which is an output to the Status Handler, is switched on via transition **t23**. This ensures that the error is reported, the group signal is switched off, and subsequent ‘group signals’ are not outputted until the error is rectified and the PN Group is successfully executed. Errors generated from the other PN Groups are represented in the same way as shown by transitions **t24** to **t31**.

## 6.2 The Unification of the Methodologies for Simulation

The graphical and logical unification of the two methodologies allows the unified simulation of the overall control system. Gray’s simulation rules for the higher level PN graphs are different from the simulation rules of  $PN \leftrightarrow PLC$ . This is because the control algorithm is executed differently. When simulating the Petri Net Occam graphs of the controllers transitions are considered individually and sequentially down

the page, in the order they are represented in the graph. On the other hand, the simulation rules of  $PN \Leftrightarrow PLC$  ensure that the simulation process correctly models the scan cycles of a PLC. Therefore all transitions are considered in a cyclic fashion until a steady state is achieved (section 4.3, also Appendix A). However, the clarity of the design and the readability of the unified graph enables the simulation of the overall system. This is because the methodology employs low level Petri Nets in which information is not hidden or folded away, unlike high level PNs (section 2.2.2.1). All places have meaningful names and not coded labels. In Figure 6-1, the first output from the Puma Controller, **Enable Vice to Table** is an actual message outputted via an Occam channel and not some coded label, i.e. it appears as *Enable Vice to Table* in the Occam code. **Enable Vice to Table 1/3** shown as an input place to **Enable Vice to Table PN Group** is a physical input, a 24V signal, to the PLC and not a coded label, i.e. it will appear as *Enable Vice to Table* on the Ladder Diagram. Therefore when transition **t1** fires (Figure 6-1, Puma Controller PN Graph) and the place **Enable Vice to Table** receives a token, the graph indicates that a 24V signal is outputted to the PLC and thus a token is also placed inside the place **Enable Vice to Table 1/3** (Figure 6-1, Enable Vice to Table PN Group). Note that all duplicate places labelled **Enable Vice to Table** receive a token each (Appendix A, Terminology and Symbols). The numbering inside the duplicate places, the distinct input and output boundaries of the PNs, the marking of the communications and labels outside places (PLC, R, SH, etc.), play a major role in the reliable simulation of the overall control system.

**Enable Vice to Table PN Graph** is simulated next, using the  $PN \Leftrightarrow PLC$  simulation rules. If this is completed, the signal 'Puma Statn Compl' is outputted to the Puma Controller and transition **t2** is simulated using Gray's simulation rules. If on the other

hand an error occurs during **Enable Vice to Table PN Graph**, then the signal 'Puma Statn Error' is outputted to the Puma Controller, and therefore transition **t22** is simulated using Gray's rules.

### 6.3 The Unification of the Methodologies for Translation

The simulation of the unified methodology ensures that the overall Distributed Control System is correct. Having simulated the overall design, dependable algorithm is produced directly from the PN graph. An equivalent Ladder Diagram has been produced from the **Puma WS PLC Petri Net graphs** (Figure 6-1) by following the PN  $\leftrightarrow$  PLC translation rules, and are shown in Appendix F. An equivalent Occam code can not be produced from the **Puma Controller Petri Net Graph**, because Gray only gives translation rules for input/output places that represent communications between Occam processes. He does not give translation rules for communications with PLCs. Transputers communicate via links by passing synchronous data via channels. The PLCs used in this case receive and output electrical signals, 24V or 0V, and therefore direct communication can not be achieved between them. The use of digital I/O interface cards between the Transputers and the PLCs overcomes this problem. Moreover, it allows "deadlock" avoidance to be maintained throughout the unified methodology by enabling a client-server relationship between the PLCs and the Transputers, similar to the relationship between the Cell Controller and the Status Handler (section 2.2.4).

Each Transputer and PLC is interfaced using a digital I/O card which is connected to the Transputer by a link. To communicate with a PLC, an Occam process specifies a channel and writes a byte to the digital I/O card which interprets it as an instruction to

switch on 24V to one or more of its outputs. An Occam process reads a byte from the digital I/O card on another specified channel. This byte informs the Occam process which, if any, of the PLC outputs is on.

Gray's rules need only a slight addition for his PN outputs to the  $PN \leftrightarrow PLC$  graphs to be reliably translated into Occam. The channel name can be constructed similarly to that of a channel between two Occam processes. For example, the output channel from the Puma Controller to the PLC could be specified by `CHAN OF BYTE pc2plc`. The constant name for any output would still be the name inside the place. Taking the first output place in Figure 6-1 (Puma Controller PN Graph), the constant name would be specified by:

```
VAL BYTE enable.vice.to.table IS value :
```

where value is determined by the configuration of the digital I/O card terminals.

## **6.4 Fault Avoidance and Elimination Achieved by the Unified Methodology**

In addition to the clarity and readability, the design rules of the unified methodology ensure that faults are avoided at the design stage, for the following reasons:

The “output-work-backwards” technique employed by the methodology plays an important role in avoiding faults at the design stage. All the outputs from the controllers are taken directly from the specification, thus ensuring that the design meets the specification. Working backwards, both the logic and the inputs required to achieve each of the specified outputs is developed. This produces a more concise design and control algorithm for a DCS than other approaches, for example where a

designer may consider all possible states of the system and then determine the outputs given those states (section 4.4).

The “output-work-backwards” technique also leads to the unidirectional flow of information between the various controllers. Although this flow is broken between the Cell Controller and the Status Handler, “deadlock” is avoided by having a client-server relationship between the Cell Controller and the Status Handler (section 2.2.4).

There is also no chance of “deadlock” occurring between the Level 2 controllers and the Level 1 PLCs because the communications are made via a digital I/O card which again enables a client-server relationship between the Transputers and the PLCs (section 6.3).

The methodology uses Petri Nets as the design tool. This allows a DCS to be simulated at the design stage and thus eliminates design errors. By selecting realistic event scenarios the overall design can be simulated against the specification of the DCS. If the simulation results show that the design correctly achieves the specification of the system, the control software into which it is translated will be correct. This is because the methodology was developed on design and translation rules which ensure that the PN design exactly translates into the control algorithm, i.e. the PN graph and the final code are equivalent. The result of this equivalence leads to the elimination of errors at the design stage and ensures that no errors are introduced into the DCS between the design stage and the implementation of the final code.

## **6.5 Discussion**

Gray’s claims of modularity and expandability of the Petri Net-Occam methodology have been substantiated, given the above changes to the design of the Level 2

controller (Figure 6-1) and the slight addition to his translation rules (section 6.3).

The introduction of 'group signals' and PN Groups by PN $\leftrightarrow$ PLC promotes ease of Level 1 programming, readability and hence reliability. Gray's PN design has been reliably updated to accommodate PN $\leftrightarrow$ PLC, and therefore an overall unified methodology which is complete and dependable has been produced. The graphical readability of the unified methodology, its powerful simulation capabilities and its direct and exact equivalence to the control algorithm enables resilient DCSs to be developed.

However, the unified methodology is only directly applicable to DCSs which consist of Transputers and Occam, and PLCs and Ladder Diagrams. This is because the rules of the methodology were developed based on the specific hardware and software being used.

To develop the Petri Net-Occam methodology specific characteristics had to be recognised, e.g.:

- how Transputers communicate with one another
- the Occam language
- how Occam processes communicate

To develop PN $\leftrightarrow$ PLC specific characteristics had to be recognised, e.g.:

- how PLCs communicate
- Ladder Logic programming
- how PLCs execute their control programs

Similarly, to unify the two methodologies, it was essential to investigate the communication protocol between the PLCs and the Transputers.

Therefore it can be concluded that a generic methodology for developing DCSs consisting of any type of controller and language can not be achieved. To design and implement a dependable Distributed Control System consisting of other hardware and software, it is essential to follow the above mentioned approach in order to develop a methodology specifically for that DCS.



# Chapter 7

## Conclusions and Recommendations for Future Work

### Contents

<b>7. Conclusions and Recommendations for Future Work.....</b>	<b>117</b>
<b>7.1 Conclusions.....</b>	<b>117</b>
<b>7.2 Recommendations for Future Work.....</b>	<b>120</b>

## 7. Conclusions and Recommendations for Future Work

Chapters 1, 2 and 3 introduce Distributed Control Systems and discuss the advantages and disadvantages of current techniques used to develop DCSs. Chapter 4 reports a methodology, called PN $\leftrightarrow$ PLC, which was developed as a major part of this research project. The methodology provides a step by step guide for developing DCSs.

Chapter 5 presents the complete control programs for a Flexible Manufacturing Cell, which were designed and tested using PN $\leftrightarrow$ PLC. Chapter 6 demonstrates how PN $\leftrightarrow$ PLC can be unified with a methodology for developing Transputer based DCSs, resulting in an overall methodology for a complete FMC. This chapter is divided into two sections, Conclusions and Recommendations for Future Work and begins by summarising the drawbacks of previous DCS development processes and discusses how the unified methodology overcomes these drawbacks. It also highlights the limitations of the methodology and recommends future work to further enhance the resilience of the unified methodology.

### 7.1 Conclusions

A methodology for developing complete and dependable Distributed Control Systems does not exist. Of the many previous attempts made to develop design and modelling tools for DCSs, none offer a tool for designing a complete system for one or more of the following reasons:

- The tools use complex formal techniques which militate against their use by industry, because they are difficult to learn and use. Furthermore, the complexity of the representation obscures the specification of the system and renders it incomprehensible to the customer.

- The tools address only one or two steps of the control system development cycle, i.e. the specification, or the design, or the simulation.
- The tools are only applicable to certain aspects of the DCS, i.e. the higher level control or the lower level control.
- The tools are unusable, especially by industry, because they pay little attention to readability.
- The tools do not model the precise operation of the control algorithm, as implemented in the hardware.
- The tools do not guide the designer towards designing an error free control system prior to implementation or indeed modelling.
- The tools do not include formal rules for translating the design into an equivalent control software.

Gray's Petri Net - Occam methodology addresses all but one of the above mentioned deficiencies. His methodology is formal, but yet readable. It also allows the accurate simulation of the code, and the precise translation of the design into software.

However, it is not for a complete DCS but only for one containing Transputers.

A major element of DCSs is the PLC. A methodology, named  $PN \Leftrightarrow PLC$ , has been developed by the author, which allows PLC control programs to be developed in a formal, but readable way, directly from the specification of the system.  $PN \Leftrightarrow PLC$  was designed to be easy to understand, and used by those in industry who are not necessarily mathematicians or computer scientists. It uses the same tool, Petri Nets, for both designing and simulating the control system, and thus eliminates the need for a translation process between the design stage and the simulation stage. Dependable

PLC Ladder Diagrams can be designed, simulated and encoded using  $PN \leftrightarrow PLC$ . The design rules lead the designer towards a “right-first-time” solution. The simulation rules ensure that the design models the code exactly, and provides fault elimination and fault avoidance prior to implementation. The translation rules ensure a one-to-one equivalence between the  $PN \leftrightarrow PLC$  graph and the Ladder Diagram. Therefore if the simulation proves the design to be correct, the final control software will be correct.

Furthermore,  $PN \leftrightarrow PLC$  can be unified with Gray’s Petri Net - Occam methodology to produce a complete development tool for Distributed Control Systems. However, this unified methodology is only directly applicable to DCSs which consist of Transputers and Occam, and PLCs and Ladder Diagrams. This is because the rules of the methodology were developed based on the specific hardware and software being used. For example, simulation of the design can only be conducted reliably if the design is simulated as the control software is executed on the specific devices. Designs can only be developed reliably if the system constituents are known in detail. Generic designs model the specification of DCSs and not the control algorithm.

Although the specific objectives of the research programme (section 2.5) have been met, the methodology has certain limitations. Currently the DCS designs are simulated on paper hence making the simulation process laborious and slow and increasing the risk of user error. As a result of this, only a small number of possible simulation scenarios can be considered for a DCS, leaving the user with the critical task of determining the most important scenarios. The methodology does not give guidance in choosing simulation scenarios. However, investigation into this area was

not within the scope of this work and would require further research to establish a method of simulation which ensures that the designs can be robustly verified.

One other limitation of the methodology is the lack of documentation. In an industrial application of the methodology, more documentation would be required than is currently generated by the methodology. For example, one important part of the documentation would be to specify the types of equipment used and their specific operational characteristics. Such detail as whether a proximity switch is of type 'PNP' or 'NPN' must be documented for maintenance or modifications to the system.

There is still valuable work which could be conducted to assist developers of DCSs, and is discussed in section 7.2.

## **7.2 Recommendations for Future Work**

To date, the steps in the methodology have been performed using a variety of drafting packages and pen and paper. For example the Petri Net graphs are drawn in Visio, the translation is done manually, and the code written using Occam and Medoc. The simulation process is also performed manually, on paper, and is slow.

Computerisation would speed up the simulation process significantly, enabling more scenarios to be considered, thus improving the reliability of the overall design.

Continual broadening of the simulation and translation rules of the unified methodology will have to occur whenever additional hardware, other than Transputers and PLCs, and languages, other than Occam and Ladder Diagrams, are encountered in a DCS.

The unified methodology addresses the issues of fault avoidance and fault elimination effectively by removing them at the design stage, using design and simulation

guidelines. However in an environment such as a Flexible Manufacturing Cell the reliability of the constituent elements cannot be guaranteed. This does not imply that the achievement of resilience is an impossible task. A reliable system failing catastrophically is less resilient than a relatively unreliable one failing safely. Therefore the handling of such possible errors efficiently is very important.

The readability, modularity and expandability of the unified methodology enables the inclusion of one or more error handling elements in parallel with the existing controllers, without the complete alteration of the original design and code. The task of an Error Handler could be, for example, the constant monitoring of the errors occurring within the Cell. However, similar to the Status Handler, there would still be the potential for “deadlock” between the Cell Controller and the Error Handler.

Several non-trivial factors would need to be investigated to incorporate an Error Handler, such as the frequency at which the Cell Controller demands updates from the Status and Error Handlers, the configuration of the Error and Status processes, the communication between them and the error handling strategy are all areas in need of further research.

PN $\leftrightarrow$ PLC can be used to pin point errors during the control of the low level work handling equipment (Chapter 5). The unification of PN $\leftrightarrow$ PLC with Petri Net - Occam methodology enables the PLCs to notify the related Controllers, which in turn send the data to the Error Handler, which in turn notifies the Cell Controller. Errors could then be differentiated by the Cell Controller, labelled (source and type errors) and reported to the operator (PC screen), who could in turn track and rectify them.

The various possible configurations of processors and processes need also be considered to determine the best layout for the resilient control of the FMC. For

example, similar to the Status Handler, the Error Handler could be included as a process within the Cell Controller, or as a process within the Status Handler whereby the Status Handler requests error updates from it and in turn reports back to the Cell Controller, or even as a separate processor reporting back to its own station (PC) and with its own decision making process.

Further work is also needed in the area of the decision making process. For example, in the event of an error on behalf of the Gantry robot during the loading of the Lathe, and if the Miller is being loaded at the same time, is it safe for the Cell Controller to instruct the operator to attend to the Gantry or does it need to instruct the Miller Controller to stop loading the Miller first, or even temporarily ignore the error until the Miller is Loaded.

The work presented in this thesis contributes to knowledge in the field of DCS development, identifying a methodology for designing, simulating and coding Transputer and PLC based DCSs. However, it remains to be proved whether the unified methodology is of practical use to those involved in developing DCSs in industry. The claims of readability, fault avoidance and expandability can only be fully tested in real-life industrial environments.

## References

- Aguiar, M. W. C. and Weston R. H., 1993, CIM-OSA and Stochastic Time Petri Nets for Behavioural Modelling and Model Handling in CIM Systems Design and Building, Procs. of Inst. Mech. Engineers, vol. 207, pp. 147-158.
- Ariffin, S., Weston R. H. and Harrison, R., 1995, Modular Petri Net Approach to the Design of Distributed Machine Control Systems, Procs. of the 31<sup>st</sup> International Matador Conf., p 621.
- Ayers, R. V., 1988, Future Trends in Factory Automation, Manufacturing Review, vol. 1, no. 2, pp. 93-103.
- Badry, A. B. and Henry, R. M., 1992, An Approach in Developing Sequential Control System Using Petri Net, Procs. of International Conf. on Manufacturing Automation, pp. 46-51.
- Barkaoui, K. and Ben-Abdallah, I., 1993, Modelling and Performance Evaluation of Tool Sharing Management in FMS Using Stochastic Petri Nets, Procs. of the IEEE International Conf. on Systems, Man and Cybernetics, vol. 1, pp. 282-88.
- Bass, J. M., Browne, A. R., Hajji, M. S., Marriott, D. G., Croll, P. R. and Fleming, P. J., 1994, Automating the Development of Distributed Control Software, IEEE Parallel & Distributed Technology: Systems & Applications, vol. 2 ISS: 4 pp. 9-19.
- Bloomfield, R. E. and Froome, P. K. D., 1991, Formal Methods in the Production and Assessment of Safety Critical Software, Reliability Engineering and System Safety, vol. 32, pp. 51-66.



- Boehm, W. B., May 1988, A Spiral Model of Software Development and Enhancement, Computer, vol. 21, no. 5, pp. 61-72.
- Borusan, A., 1993, Coloured Petri Net Based Modelling of FMS, Procs. of the IEEE International Conf. on Systems, Man and Cybernetics, vol. 1, pp. 54-59.
- Chao, D. Y., Chen, T. H. and Wang, D. T., Oct. 1992, X Window Implementation of Petri Net Based Animation for FMS, Procs. of IEEE International Conf. on System, Man and Cybernetics, vol. 2, pp. 1651-6.
- Clarke, S., Faulkner, P., Hedley, D., Maisey, D. and Pegler, S., 1995, A Code of Practice for the Development of Safe PLC Software, Procs. of the Safety-critical Systems Symposium, Brighton 1995, Springer-Verlag, pp. 207-22.
- Cotter, S. M. and Woodward, A. T., 1986, Designing Better Programs for Controllers, Control and Instrumentation, vol. 18, pp. 75-83.
- Courvoisier, M., Valette, R., Bigou, J. M. and Esteban, P., 1983, A Programmable Logic Controller Based on a High Level Specification Tool, Procs. of IECON Conf. on Industrial Electronics, pp. 174-179.
- Das, P. K. and Fay Freund, E., 1983, Fast Non-linear Control With Arbitrary Pole Placement for Individual Robots and Manipulators, International Journal of Robotics Research, vol. 1, no. 1, pp. 65-78.
- David, R. and Alla, H., 1992, Petri Nets & Grafset, Prentice Hall International (UK) Ltd..
- De Gaspari, J., March 1992, High-Speed, Multi-Tasking Ability Claimed for New 'Transputer' Controls, Plastics Technology, pp. 13-15.

- Draper, C. M. and Holding, D. J., 1989, The Specification and Fast Prototyping of a Distributed Real-Time Computer Control System for a Modular Independently Driven High-Speed Machine, Procs. of International Conf. on Software Engineering for Real Time Systems, no. 309, pp. 199-203.
- Duan, N. and Kumara, R. T., Feb. 1993, A Distributed Hierarchical Control Model for Highly Autonomous Flexible Manufacturing Systems, Procs. of the International Conf. on Intelligent Autonomous Systems, pp. 532-41.
- Ezpeleta, J., Colom, J. M. and Martinez, J., 1995, Petri Net Based Deadlock Prevention Policy for Flexible Manufacturing Systems, IEEE Transactions on Robotics and Automation, vol. 11, no. 2, pp. 173-184.
- Farrington, M. and Billington, J., March 1996, A CP-net Approach to Control Logic Engineering, IFIP Working Conference on Software Engineering, Stugart.
- Ford, R. G., 1991, Integration of Equipment for a Flexible Manufacturing Laboratory, Procs. of Annual American Society for Engineering Education, Challenges for Change, vol. 2, pp. 1117-21.
- Gray, P., 1995, A Petri Net-Occam Based Methodology for the Development of Dependable Distributed Control Software, Ph.D. Thesis, Sheffield Hallam University.
- Green, J., 1989, Petri Net Design Methodology for Sequential Control, Measurement & Control, vol. 22, pp. 288-291.
- Halang, W. A., 1989, Languages and Tools for the Graphical and Textual System Independent Programming of Programmable Logic Controllers, Microprocessing and Microprogramming, vol. 27, ISS. 1-5, pp. 583-590.

- Hales, W. M. M., Gray, P. and Poole, F., 1993, The Development of a Dependable Distributed Control System for Flexible Manufacture, Procs. WTC'93, World Transputer Congress. pp. 152-68.
- Hasegawa, M., Takata, M., Temmyo, T. and Matsuka, H., 1990, Procs. of IEEE International Conf. on Robotics and Automation, vol. 1, pp. 514-19.
- Hilal, R. and Ladet, P., 1993, Modelling, Control, and Simulation of Flexible Manufacturing Systems Through the Use of Synchronous Petri Nets, ISBN: 0780308913, pp. 559-63.
- Huang, H. P. and Chang, P. C., 1992, Specification, Modelling and Control of a Flexible Manufacturing Cell, International Journal of Production Research, vol. 30, no. 11, pp. 2515-43.
- IEC, International Electrotechnical Commission, 1992, Programmable Controllers - Programming Languages, 1131 Part 3 Standard, , 65B Secretariat, Central Office, Geneva.
- Inmos Ltd., 1989, Occam 2 Toolset User Manual, "A Tutorial Introduction To Occam Programming".
- Inmos ltd., 1989b, Transputer Databook, Second Edition.
- Inmos Ltd., 1990, The Transputer Instruction Set; A Compiler Writer's Guide. "Transputer Architecture and Overview; Transputer Technical Specifications".
- Jafari, M. A. and Boucher, T. O., 1994, A Rule-Based System for Generating a Ladder Logic Control Program from a High Level Systems Model, Journal of Intelligent Manufacturing, vol. 5, ISS. 2, pp. 103-120.

- Jelly, I., and Gorton, I., 1994, Software Engineering for Parallel Systems, Information and Software Technology, vol. 36, no. 7, pp. 381-396.
- Kamath, M., and Viswanadham, N., 1986, Applications of Petri Net Based Models in the Modelling and Analysis of Flexible Manufacturing Systems, Procs. of the International Conf. on Robotics and Automation, pp. 312-17.
- Lau, M. W. S. and Seet, G., 1993, The Use of Petri Nets for Occam Programming for Transputers, Advances in Engineering Software, vol. 17, pp. 155-63.
- Leveson, N. G. and Stolzy, J. L., Mar. 1987, Safety Analysis Using Petri Nets, IEEE Trans. Software Engineering, vol. SE-13, no. 3, pp. 386-97.
- Lu, S. S. and Huang, H. P., Aug. 1992, Modularisation and Properties of Flexible Manufacturing Systems, Procs. of the 8<sup>th</sup> International Conf. on CAD/CAM, Robotics and Factories of the Future, vol. 2, pp. 1768-82.
- Moore, P. and O'Donoghue, P., 1994, Developing Transputer-Based Systems Using HOOD and Parallel C, Information and Software Technology, vol. 36, no. 6, pp. 353-60.
- Murata, T., Shenker, B. and Shatz, S. M., 1989, Detection of Ada Static Deadlocks Using Petri Net Invariants, IEEE Transactions on Software Engineering, vol. 15, no. 3, pp. 314-25.
- Nagao, Y., 1993, Application of Petri Nets to Sequence Control, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, vol. E76-A, ISS: 10, pp. 1598-606.

- Pardy, J., Amroun, A., Bolton, M. and Adamski, M., 1994, Parallel Controller Synthesis for Programmable Logic Devices, Microprocessors and Microsystems, vol. 18, no. 8, pp. 451-457.
- Peterson, J. L., 1981, Petri Net Theory and The Modelling of Systems, Prentice Hall ISBN 0-13-661983-5.
- Reddy, G. B., Murty, S. S. N. and Ghosh, K., Nov. 1993, Timed Petri Net: An Expeditious Tool for Modelling and Analysis of Manufacturing Systems, Mathematical and Computer Modelling, vol. 18, ISS 9, pp. 17-30.
- Sahraoui, A., Atabakhche, H., Courvoisier, M. and Valette, R., 1987, Joining Petri Nets and Knowledge Based Systems for Monitoring Purposes, Procs. of IEEE International Conf. On Robotics and Automation, vol. 2, pp. 1160-65.
- Satoh, T., Nose, K. and Kumagai, S., 1992, Automatic Generation System of Ladder List Program by Petri Net, Procs. of IEEE International Workshop on Emerging Technologies and Factory Automation - Technology for the Intelligent Factory, pp. 128-33.
- SEMSPLC, Software Engineering Methods for Safe Programmable Logic Controllers, 1992-95, DTI/SERC Sponsored Collaborative Project. Collaborators also include University of York, York Software Engineering Ltd, Cegelec Ltd, ICI Ltd, IDEC Ltd, Nuclear Electric Plc, the HSE, and Cincinnati Milacron Ltd.
- Shatz, S. M. and Wang, J. P., October 1987, Introduction to Distributed Software Engineering, IEEE Computer, vol. 20, no. 10, pp. 23-31.

- Simpson, J., Kocken, R. and Albus, J., 1982, The Automated Manufacturing Research Facility of the National Bureau of Standards, *Journal of Manufacturing Systems*, vol. 1, no. 1, pp. 17-32.
- Slack, N., 1988, Manufacturing Systems Flexibility - An Assessment Procedure, *Computer Integrated Manufacturing Systems*, vol. 1, no. 1, pp. 25-31.
- Sundaram, C. R. M. and Narahari, Y., 1993, Modelling And Analysis of the Variance in Parallelism in Parallel Computations, *Computers and Electrical Engineering*, vol. 19, no. 6, pp. 495-506.
- Taholakian, A and Hales, W. M. M., 1995, The Design and Modelling of PLC Programs Using Petri Nets, *Procs. of the International Conference on Maintenance, Reliability and Quality*, pp. 194-199.
- Taholakian, A and Hales, W. M. M., 1996, PN $\leftrightarrow$ PLC: A Methodology for Designing, Simulating and Coding PLC Based Control Systems Using Petri Nets, Accepted by the *International Journal of Production Research*, to be published in June 1997.
- Tudruj, M., 1992, Multi-Layer Reconfigurable Transputer Systems with Distributed Control of Link Connections, *Microprocessing and Microprogramming*, vol. 34, ISS 1-5, pp. 201-4.
- Viswanadham, N. and Johnson, T. L., 1988, Fault Detection and Diagnosis of Automated Manufacturing Systems, *Procs. of the 27<sup>th</sup> IEEE Conf. on Decision and Control*, vol. 3, pp. 2301-2306.

- Viswanadham, N. and Narahari, Y., 1992, Stochastic Modelling of Flexible Manufacturing Systems, *Mathematical and Computer Modelling*, vol. 16, no. 3, pp. 15-34.
- Viswanadham, N., Narahari, Y. and Johnson, T. L., 1990, Deadlock Prevention and Deadlock Avoidance in Flexible Manufacturing Systems Using Petri Net Models, *IEEE Transactions on Robotics and Automation*, vol. 6, no. 6, pp. 713-23.
- Wardman, D., 1994, *Systematic Logic Controller Programming; A Self Study Course*, Lecture Notes, School of Engineering, Sheffield Hallam University.
- Webb, J., 1992, *Programmable Logic Controllers; Principles and Applications*, Second Edition, Macmillan Publishing Company, ISBN 0-02-424970-X.
- Weston, R. H., 1991, CIM Enterprises for the 21<sup>st</sup> Century, *Procs. of the International Conf. on Computer Integrated Manufacturing*, ch. 135, pp. 3-6.
- Weston, R. H., 1993, Steps Towards Enterprise-Wide Integration: A Definition of Need and First-Generation Open Solutions, *International Journal of Production Research*, vol. 31, no. 9, pp. 2235-2254.
- Williams, A. M. and Lill, B. H., May 1987, Commercially Available Flexible Assembly Cell, *Procs. of the 10<sup>th</sup> Annual Conf. of the British Robot Association*, ch. 27, pp. 167-76.

## **Appendices**

**Appendix A PN $\Leftrightarrow$ PLC: The Methodology Rules, Steps and Terminology**

**Appendix B Puma Work Station PN Groups, Developed Using PN $\Leftrightarrow$ PLC**

**Appendix C Miller Work Station PN Groups, Developed Using PN $\Leftrightarrow$ PLC**

**Appendix D Lathe Work Station PN Steps, Developed Using PN $\Leftrightarrow$ PLC**

**Appendix E Error Handling PN Graphs, Developed Using PN $\Leftrightarrow$ PLC**

**Appendix F Ladder Diagrams Translated from Puma Work Station PN Groups, Using PN $\Leftrightarrow$ PLC.**

**Appendix G Ladder Diagrams Translated from Miller Work Station PN Groups, Using PN $\Leftrightarrow$ PLC.**

**Appendix H Step Ladder Diagrams Translated from Lathe Work Station PN Steps, Using PN $\Leftrightarrow$ PLC.**



**Appendix A  $PN \Leftrightarrow PLC$ : The Methodology Rules, Steps and  
Terminology**

## Design Rules

- Rule 1. The drafting of the  $PN \Leftrightarrow PLC$  graph is carried out using a right-to-left and down-the-page approach i.e. starting with an output, work backwards and determine the conditions for switching the output on. This must be represented with one or more “switching-on transitions” within the boundaries of the  $PN$  graph, and their “switching-on places”.*
- Rule 2. Consider the conditions for switching the output off. This must be represented with one or more “switching-off transitions”, to the right of the output place, and their “switching-off places”. There must be no non-determinism in the design.*
- Rule 3. Rules 1 and 2 are repeated for all output places, and also for any internal relays used.*

## Simulation Steps

- Step 1. The first step of the simulation process is to determine one or more realistic scenarios of events.*
- Step 2. Consider the first event of the chosen scenario and mark the  $PN \Leftrightarrow PLC$  graph accordingly.*
- Step 3. Given the initial marking, determine the state of each output place and internal relay. This is done by considering the “switching-on transition” and the “switching-off transition” of each place consecutively. Note, if both transitions are fired then the output place does not receive a token.*

*The final marking of the  $PN \Leftrightarrow PLC$  graph represents the state of the system after one scan of the program.*

*Step 4. The resulting marking from one scan is the initial marking for the next scan which is simulated as in step 3. A number of scans are made until the  $PN \Leftrightarrow PLC$  graph reaches a steady state, i.e. the marking does not change from one scan to the next.*

*Step 5. Once a steady state has been achieved then the markings representing the next event in the scenario are included in the  $PN \Leftrightarrow PLC$  graph.*

### Translation Rules

*Rule 1. Consider an output place and situate it as an output on the right-hand side of the Ladder Diagram. Consider the "switching-on transition" to the left of that output place. The input places to that transition are the conditions for switching the output place on, and are therefore represented as inputs on the left-hand side of the Ladder Diagram.*


*Rule 2. Always latch the output with itself.*

*Rule 3. Consider the "switching-off transition" to the right of the output place. The "switching-off arcs" joining places to this transition identify these places as the switching-off conditions of that output place. It is the inverse of these conditions which keep the output 'On' and are therefore represented on the LD using De Morgan's law.*


*Rule 4. Repeat rules 1 - 3 for all the places*

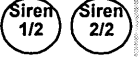
## Terminology and Symbols.


In order to make the PN graph readable, certain symbols have been adopted as follows:




 is a "**return arc**" which shows that an input place will receive its token back after the transition has fired. For example this applies to input signals over which the PLC has no control, such as signals from sensors. However, this symbol can be ignored once the following two points are appreciated:


1. Tokens are removed from an output **place** or an internal **place** only via the "switching-off transition" of that **place**.
2. Input signals, such as those from sensors, will not lose their tokens as a result of their output **transitions** firing.

 are "**switching-off arcs**" which clearly show what is being switched off and what is doing the switching off. Combined with design rule 2, the "switching-off arcs" show exactly what the designer has in mind.

 are duplicated places. Where a place is the input to several transitions, it can be duplicated in order to reduce the number of arcs that cross. This simplifies the overall PN graph and improves readability. It is, however, essential that a tally is kept of the number of duplicate places (1/2, 2/2) so that they are not overlooked by the designer and simulator.

 is a “dummy” or “drain place”, which is used to demonstrate that a token is drained from a particular place as a result of its transition firing. This symbol clearly shows the "switching-off transitions", and proves that the designer has considered how each output and internal relay will be switched off.

$P_a$   is a **transition** which has priority over any other **transition**,  $P_b$   has priority over any **transition** except for  $P_a$  , and so on.

 is a shaded place and indicates that it occurs in and is switched off in another PN graph.

**PN Grouping** is used to simplify the PN graph and make it more readable. When an input signal is connected to every “switching-on” and “switching-off transition” it transforms the PN graph to a PN Group. Therefore PN graph is entitled PN Group and arcs from the input signal are not drawn to each transition.

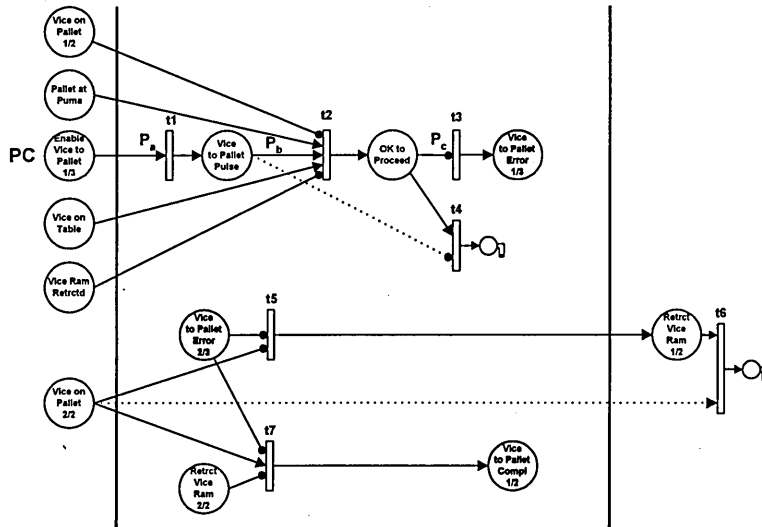
**PN Steps** are used when designing Step Ladders. In a Step LD the Step Relay is connected to all “switching-on transitions” to ensure that outputs are only switched on and off by the conditions shown in that particular Step. Therefore to simplify the PN graph and make it more readable, it is entitled PN Step and arcs from the Step Relay are not drawn to each “switching-on transition”.

**Appendix B Puma Work Station PN Groups, Developed Using**

**PN↔PLC**

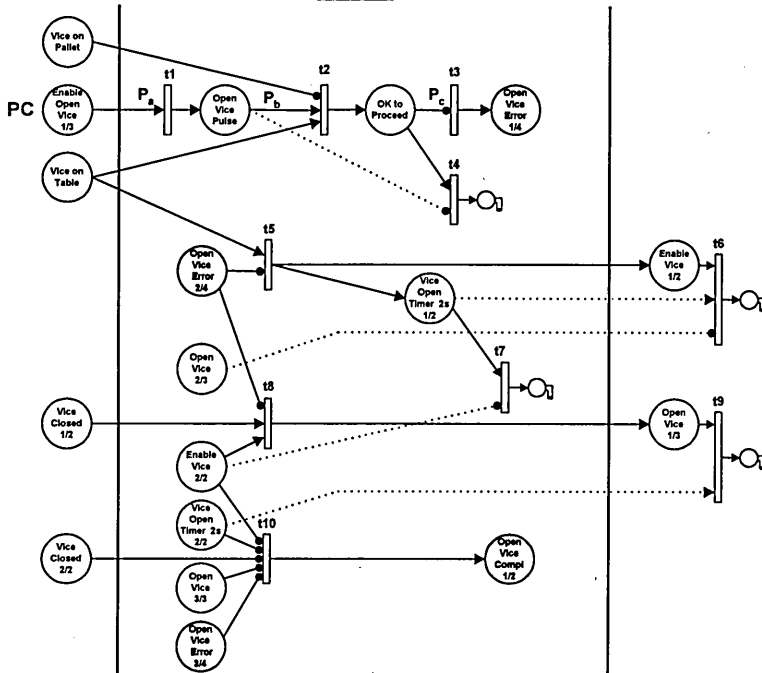


**Enable Vice to Pallet PN Group  
(Puma)**



**Figure 5-7**

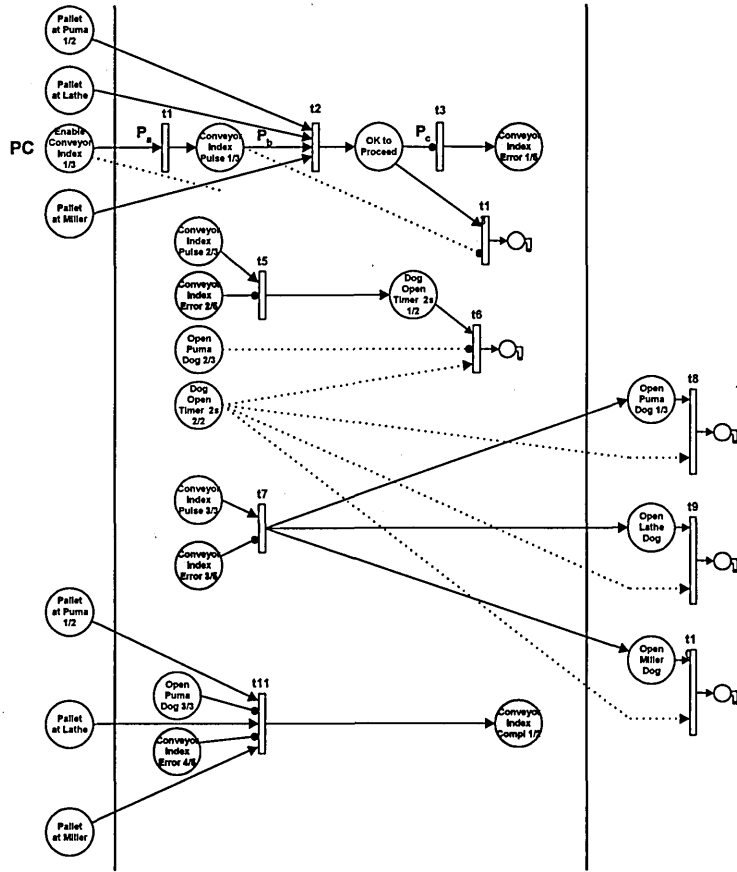
**Enable Open Vice PN Group  
(Puma)**



**Figure 5-8**



**Enable Conveyor Index PN Group  
(Puma)**



**Figure 5-9**

## **Appendix C Miller Work Station PN Groups, Developed Using**

**PN↔PLC**

**Enable Vice to Table PN Group  
(Miller)**

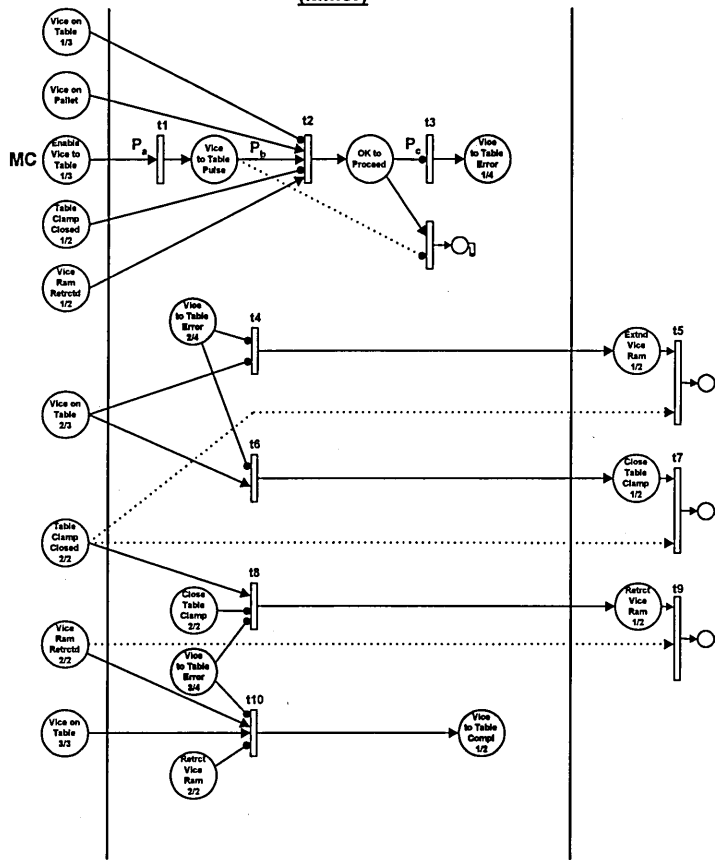
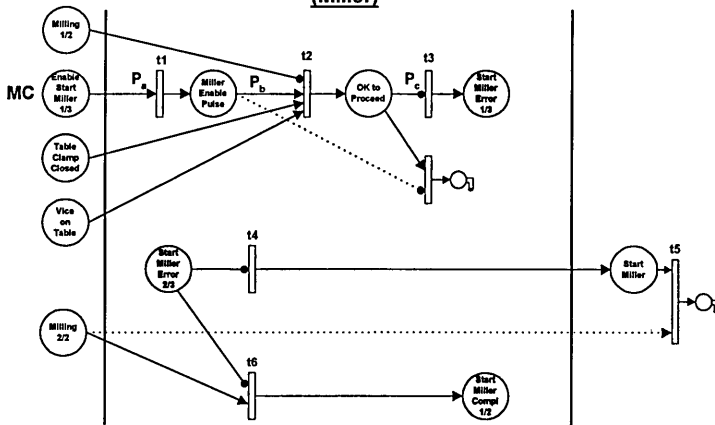


Figure 5-10

**Enable Start Miller PN Group  
(Miller)**



**Milling PN Graph  
(Miller)**

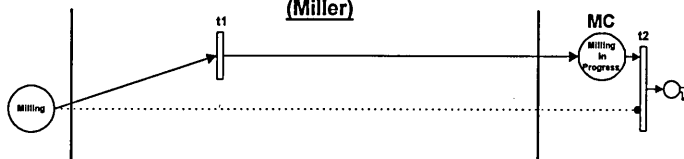
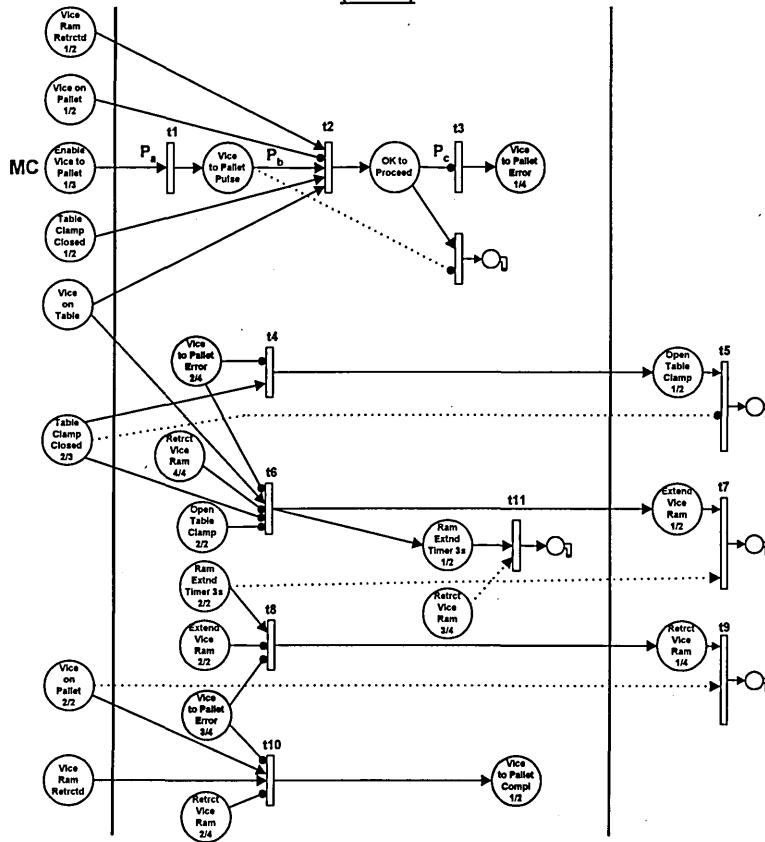


Figure 5-11

**Enable Vice to Pallet PN Group  
(Miller)**



**Figure 5-12**

**Appendix D Lathe Work Station PN Steps, Developed Using**

**PN↔PLC**

**Enable Vice to Table PN Step  
(Lathe)**

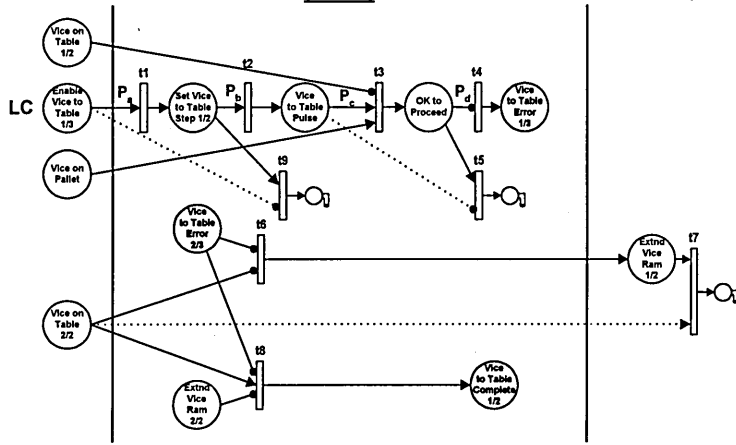


Figure 5-15

**Enable Grip Workpiece PN Step  
(Lathe)**

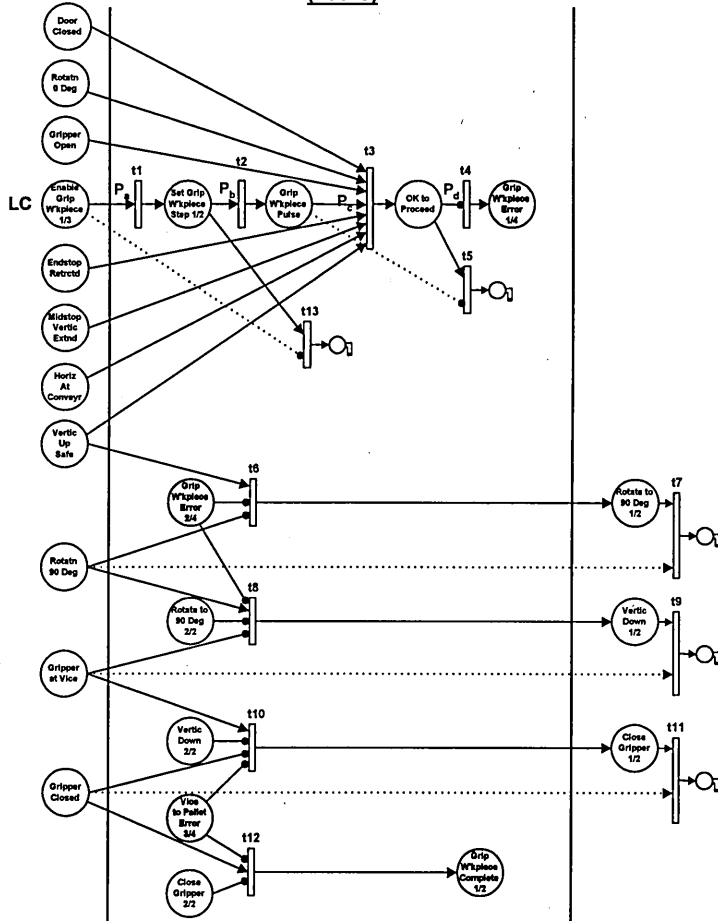


Figure 5-16

**Enable Workpiece to Chuck PN Step  
(Lathe)**

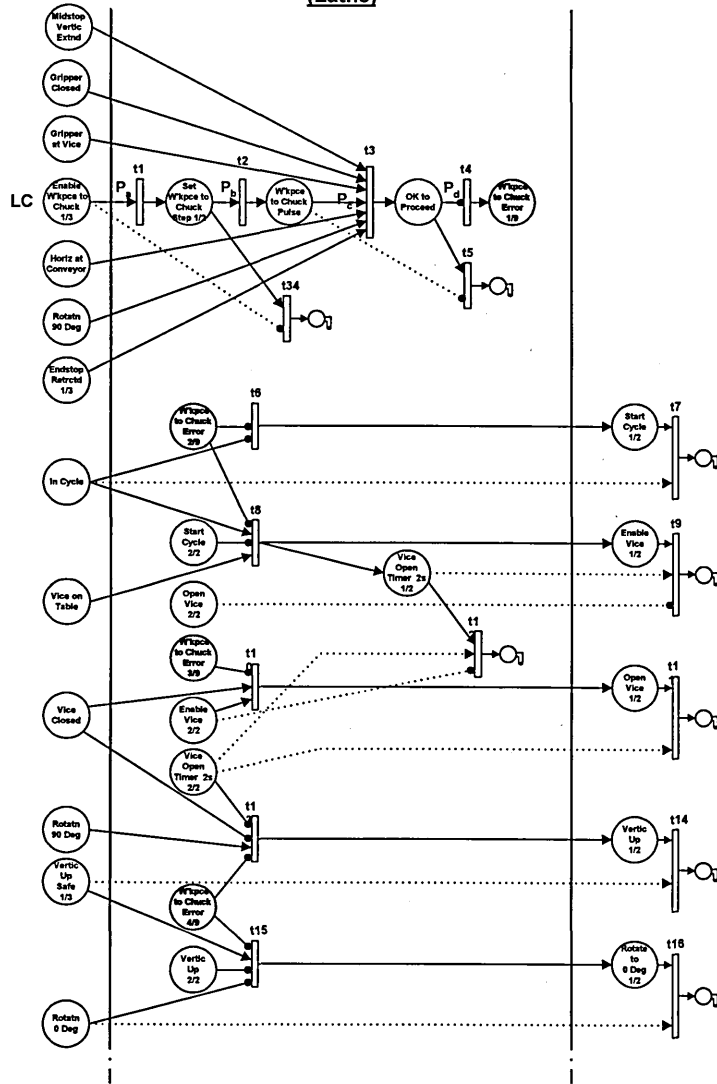
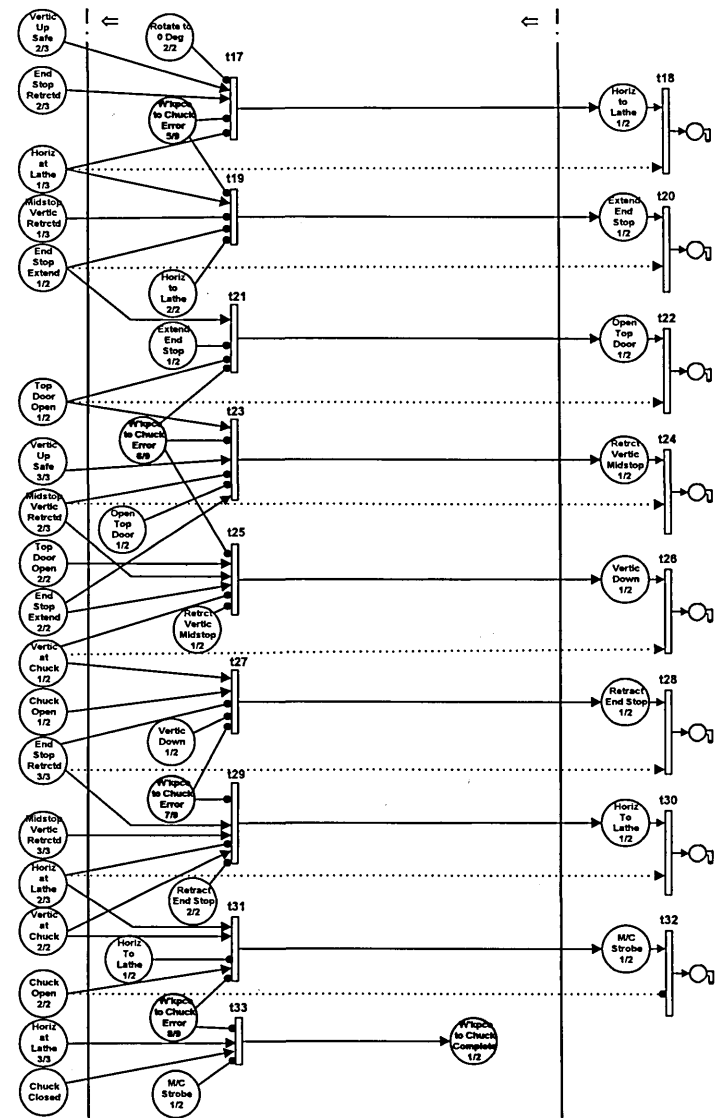
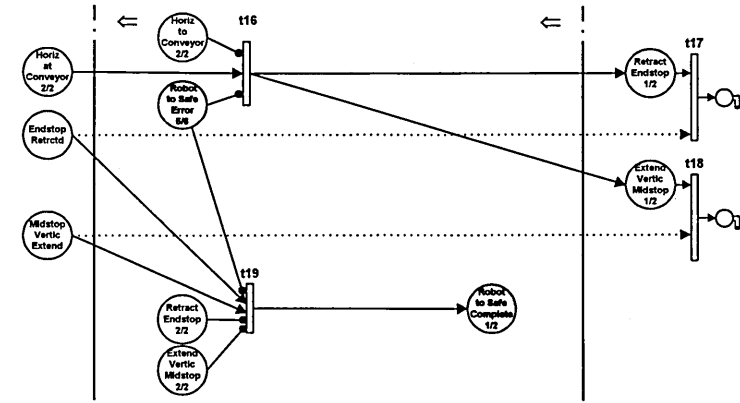
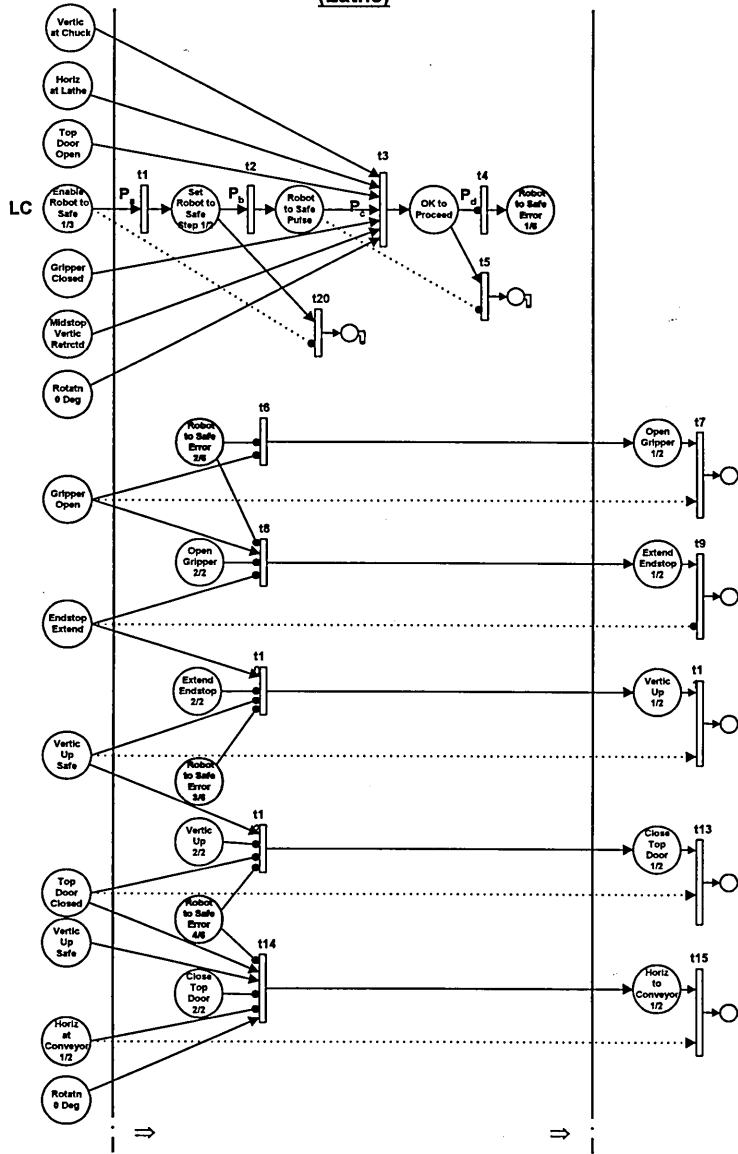


Figure 5-17 (continued =>)



**Enable Robot to Safe PN Step  
(Lathe)**



**Figure 5-18**



**Enable Vice to Pallet PN Step  
(Lathe)**

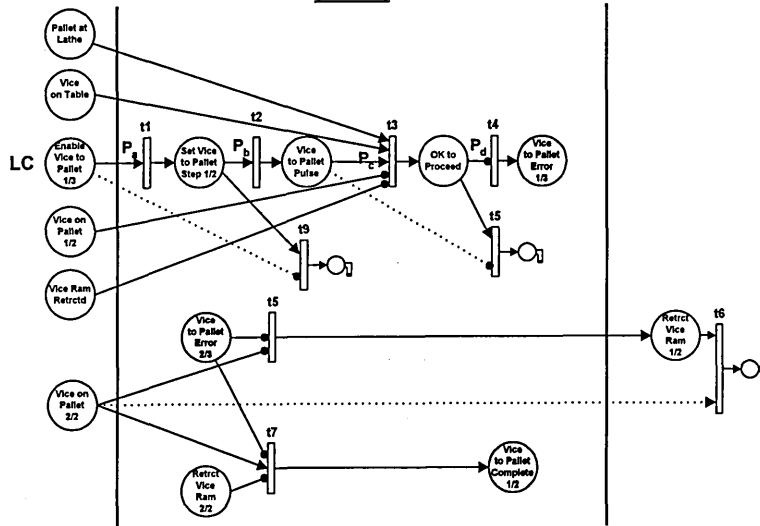


Figure 5-19

**Enable Start Maching PN Step  
(Lathe)**

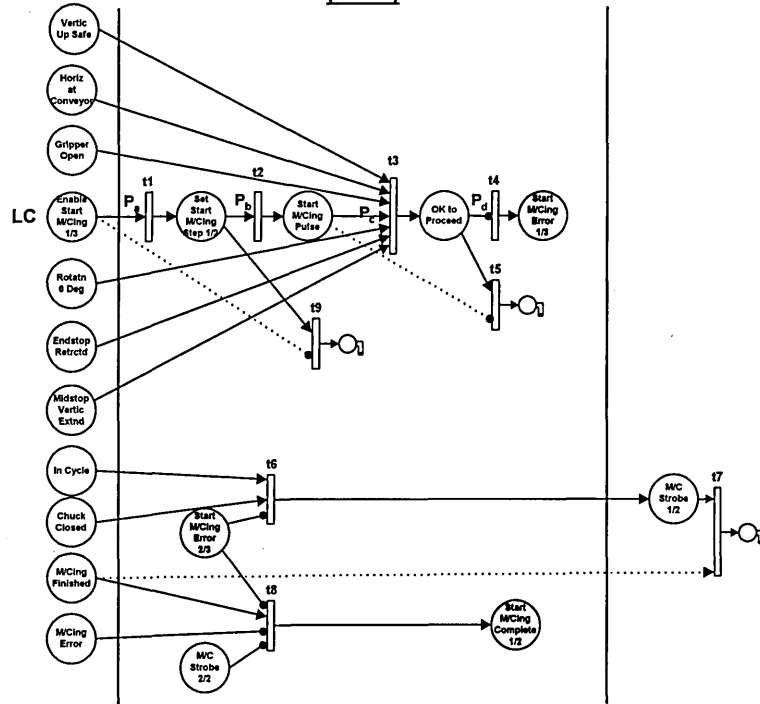
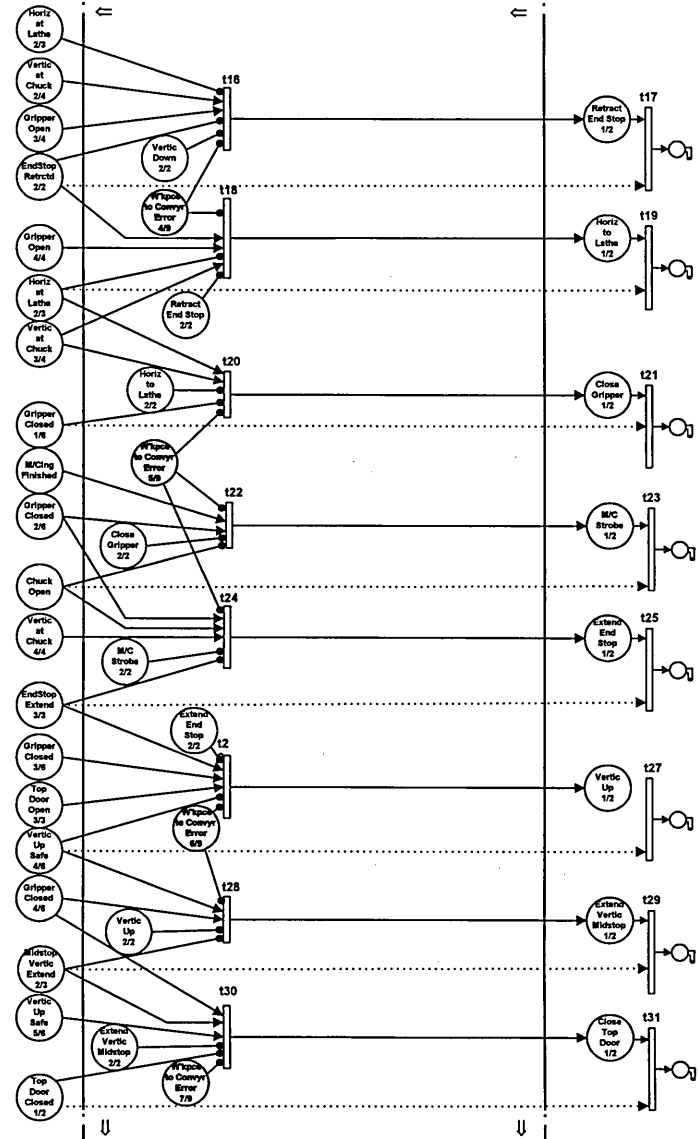
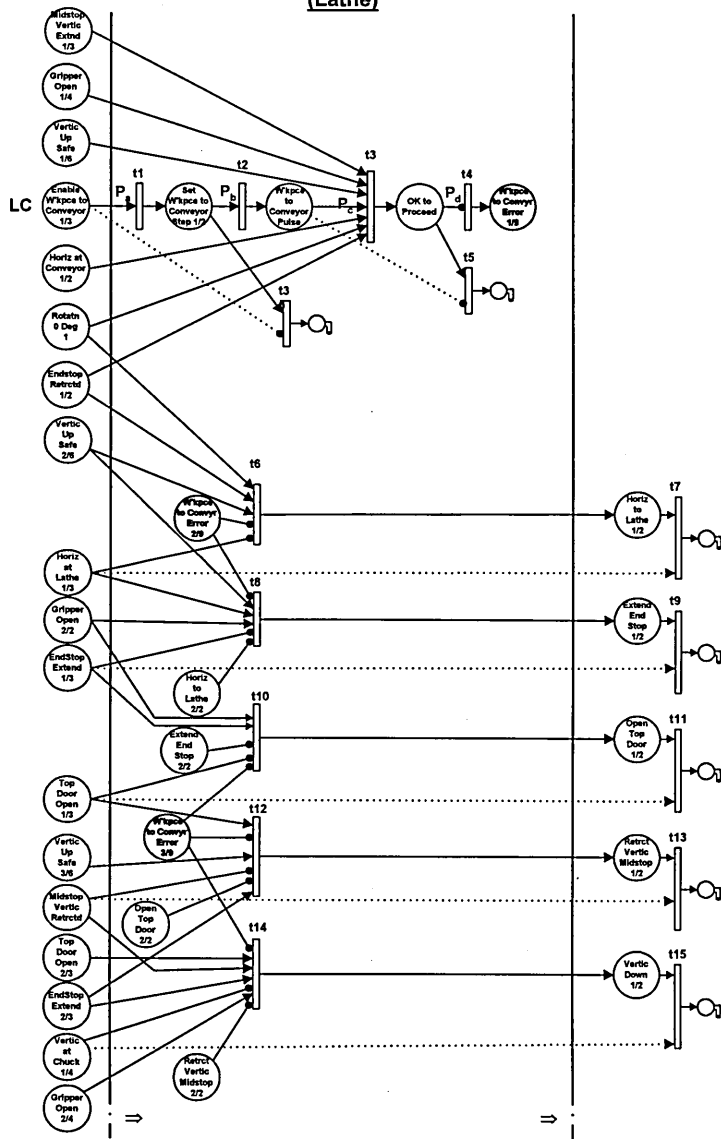


Figure 5-20

**Enable Workpiece to Conveyor PN Step  
(Lathe)**



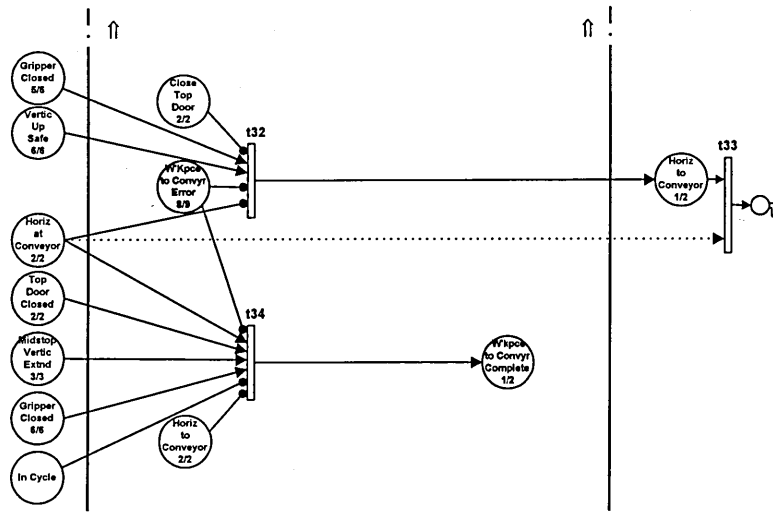


Figure 5-21

### Enable Workpiece to Vice PN Step (Lathe)

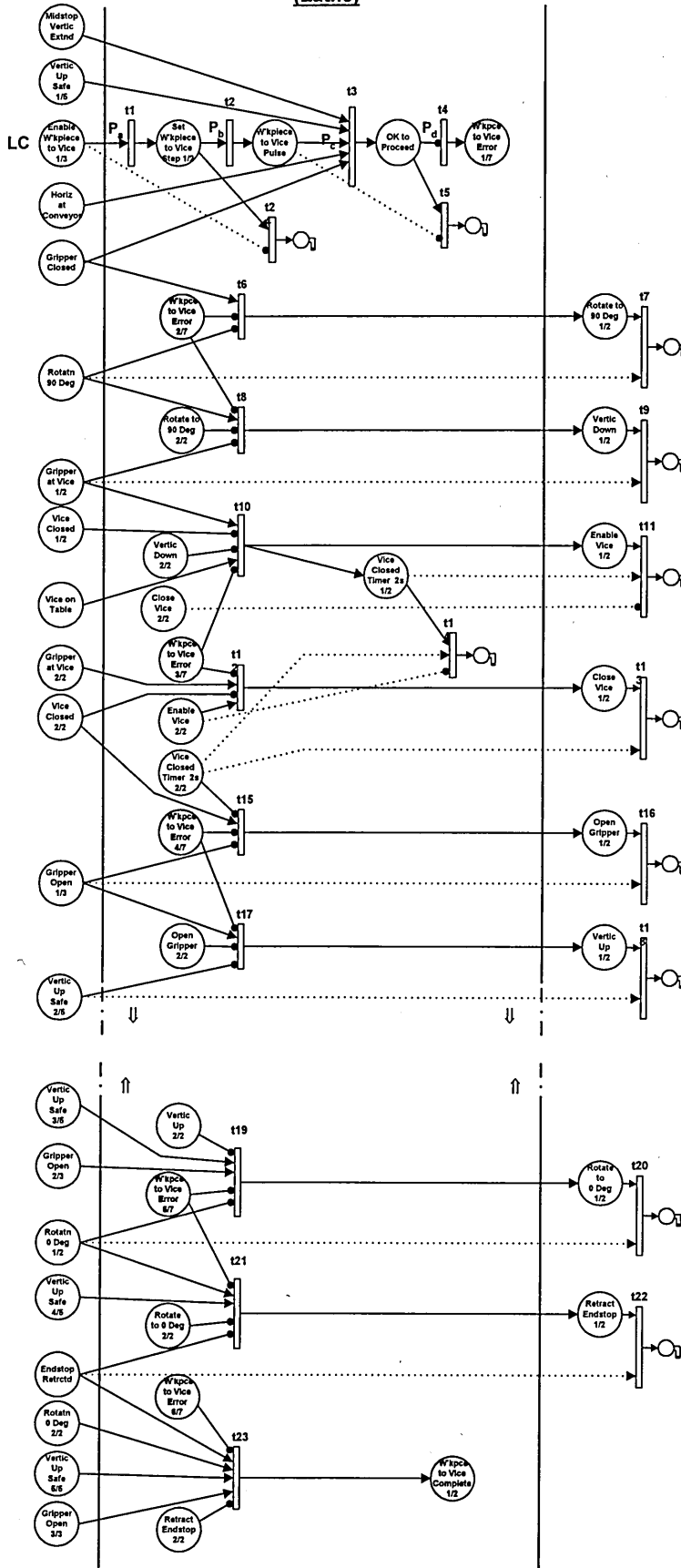
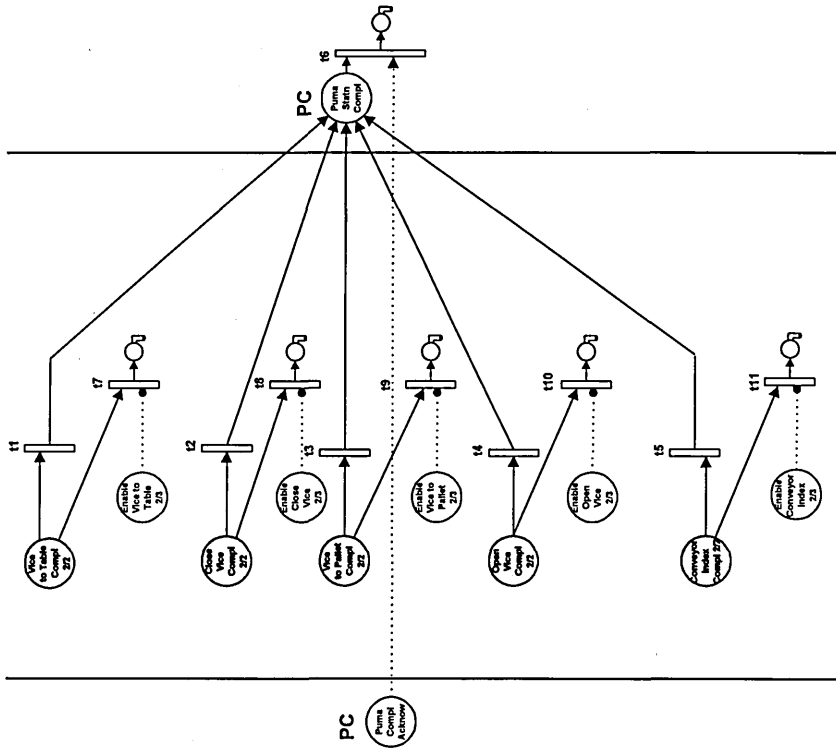


Figure 5-22

## **Appendix E Error Handling PN Graphs, Developed Using**

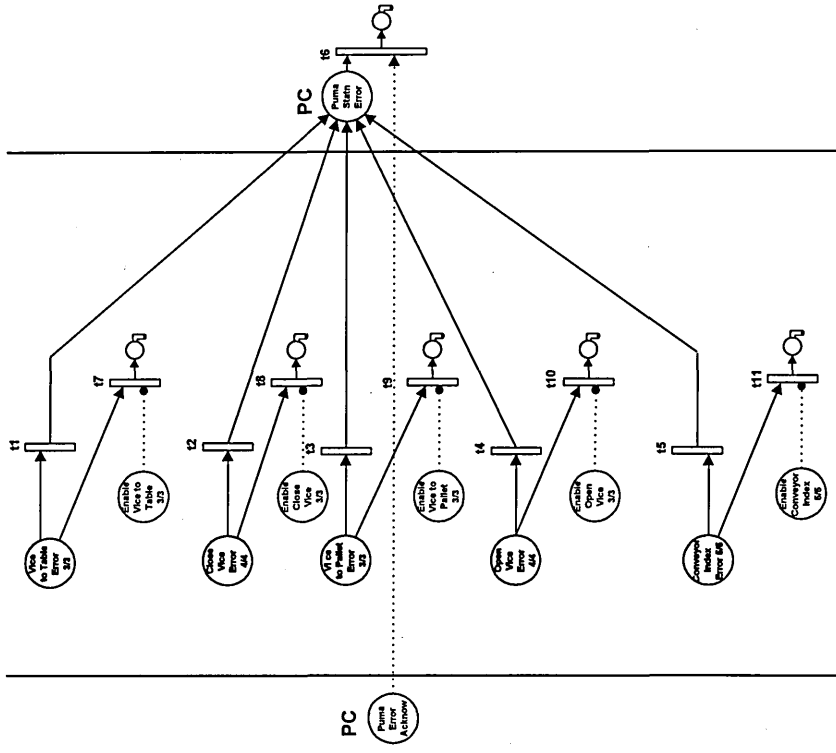
**PN $\leftrightarrow$ PLC**

**Puma WS Complete  
Status PN Graph**



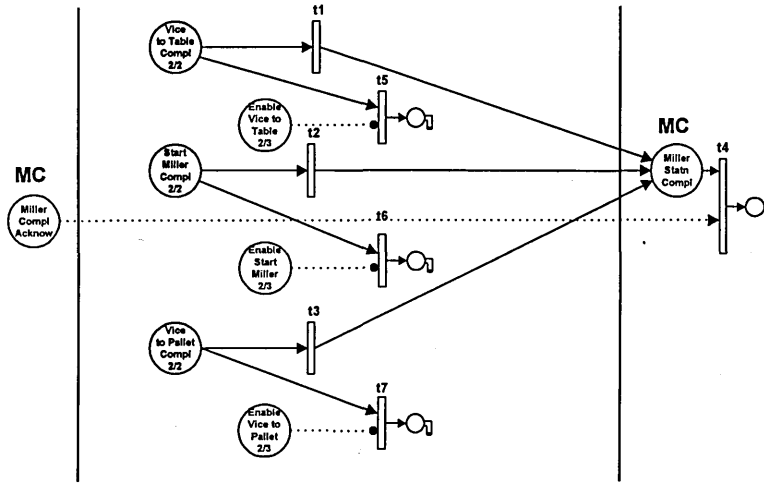
**Figure 5-23**

**Puma WS Error  
Status PN Graph**



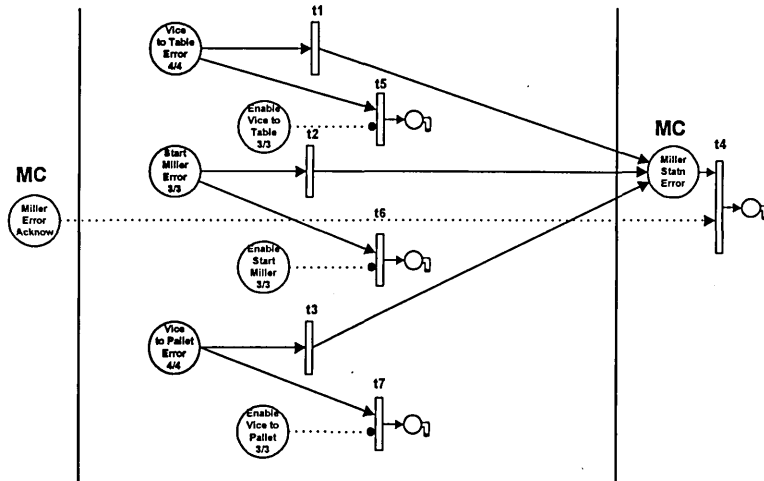
**Figure 5-24**

**Miller WS Complete  
Status PN Graph**



**Figure 5-25**

**Miller WS Error  
Status PN Graph**



**Figure 5-26**

**Lathe WS Complete  
Status PN Graph**

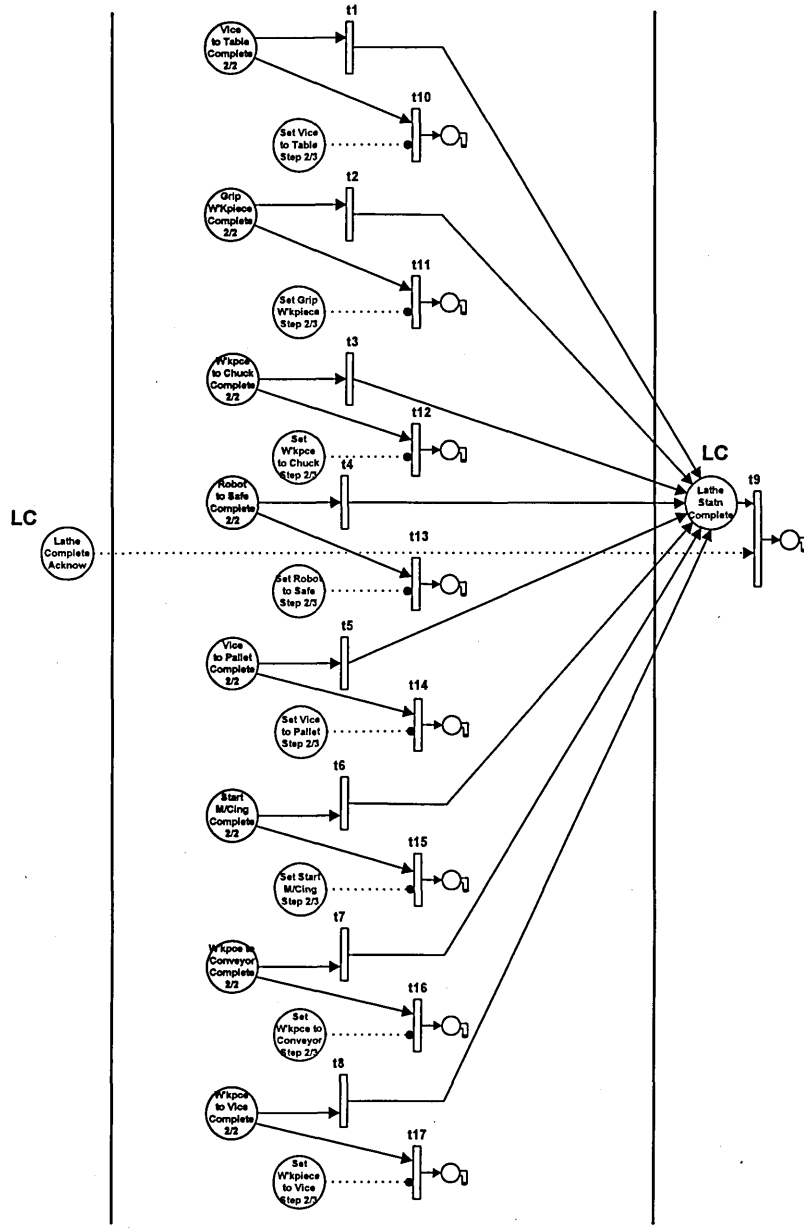


Figure 5-27



**Lathe WS Error Status PN Graph**

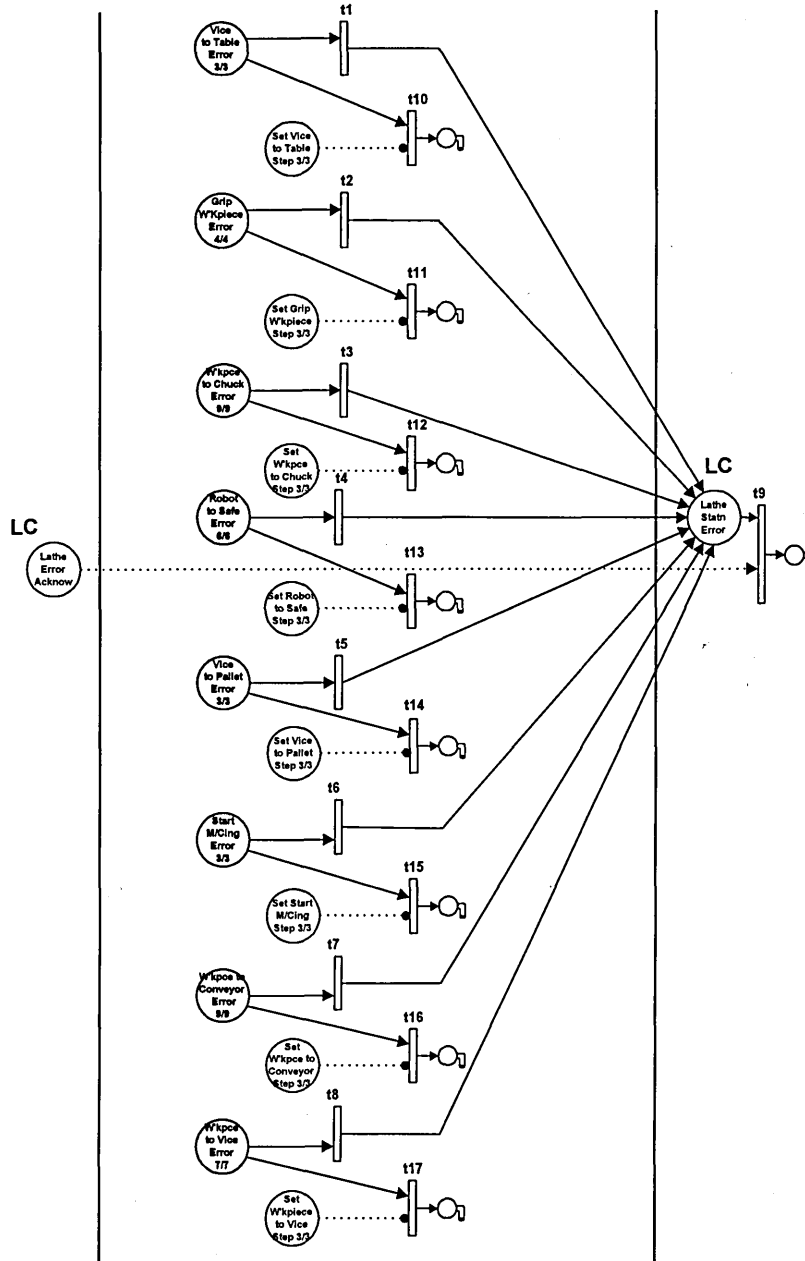


Figure 5-28

## **Appendix F Ladder Diagrams Translated from Puma Work**

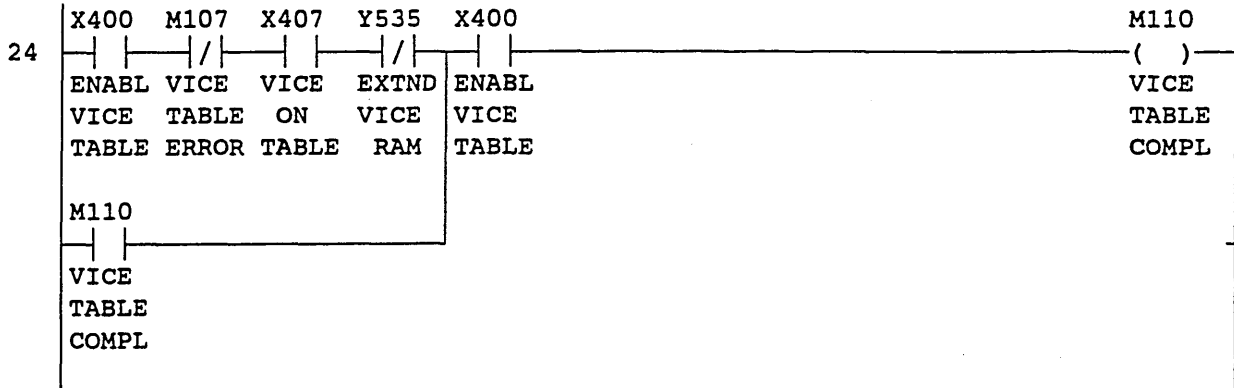
### **Station PN Groups, Using PN $\leftrightarrow$ PLC**

CONTROL OF WORK HANDLING AND CONVEYOR INDEXING FOR PUMA STATION IN SCHOOL OF ENGINEER FMC	SHEFFIELD HALLAM UNIVERSITY	Appendix F	Proj:PUMA
		Date: 12\12\95	Syst:F1/F2
		Rev.no: 3	Type:Name
	Draw.no:	Sign: A.T.	Page: 1

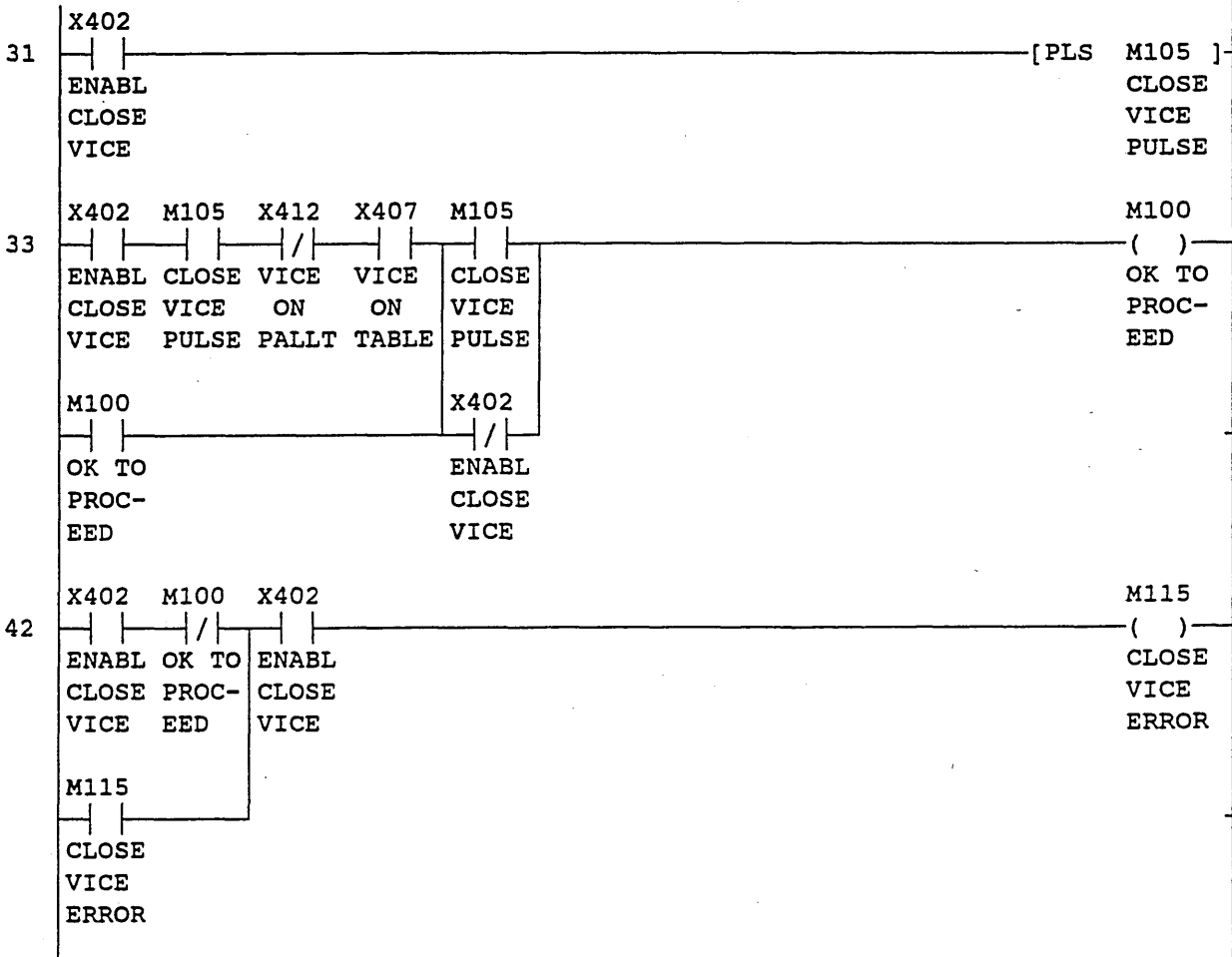
I/O	Name	Comment	Remark
X400	ENABLVICE TABLE	Input from Puma Controller	
X401	ENABLOPEN VICE	Input from Puma Controller	
X402	ENABLCLOSEVICE	Input from Puma Controller	
X403	ENABLVICE PALLT	Input from Puma Controller	
X404	ENABLCONVRINDEX	Input from Puma Controller	
X407	VICE ON TABLE	Input from Sensor 1	Ind. Prox.
X410	VICE RAM RETRD	Input from Sensor 2	Magn. Reed
X411	VICE CLOSD	Input from Sensor 3	Press. Sw.
X412	VICE ON PALLT	Input from Sensor 4	IR Refl.
X413	PALLT AT PUMA	Input from Sensor 5	Cap. Prox.
X500	PALLT AT LATHE	Input from Sensor 6	Cap. Prox.
X501	PALLT AT MILLR	Input from Sensor 7	Cap. Prox.
X502	PUMA ERRORACKNO	Input from Puma Controller	
X503	PUMA COMPLACKNO	Input from Puma Controller	
Y430	PUMA STATNCOMPL	Output to Puma Controller	
Y431	PUMA STATNERROR	Output to Puma Controller	
Y434	ENABLVICE	Output to Toggle Clamp	1Sol.Spring
Y530	OPEN VICE	Output to Vice Cyl. (Retract)	Double Sol
Y531	CLOSE VICE	Output to Vice Cyl. (Extend)	Double Sol
Y532	OPEN PUMA DOG	Output to Puma Stop Cyl.	Single Sol
Y533	OPEN LATHE DOG	Output to Lathe Stop Cyl.	Single Sol
Y534	OPEN MILLR DOG	Output to Miller Stop Cyl.	Single Sol
Y535	EXTNDVICE RAM	Output to Ram Cyl. (Extend)	Double Sol
Y536	RETRTVICE RAM	Output to Ram Cyl. (Retract)	Double Sol
M100	OK TOPROC-EED		Int. Relay
M101	VICE TABLEPULSE		Int. Relay
M102	OPEN VICE PULSE		Int. Relay
M103	VICE PALLTPULSE		Int. Relay
M104	CONVRINDEXPULSE		Int. Relay
M105	CLOSEVICE PULSE		Int. Relay
M107	VICE TABLEERROR		Int. Relay
M110	VICE TABLECOMPL		Int. Relay
M111	VICE PALLTERROR		Int. Relay
M112	VICE PALLTCOMPL		Int. Relay
M113	OPEN VICE ERROR		Int. Relay
M114	OPEN VICE COMPL		Int. Relay
M115	CLOSEVICE ERROR		Int. Relay
M116	CLOSEVICE COMPL		Int. Relay
M117	CONVRINDEX COMP		Int. Relay
M120	CONVRINDEXERROR		Int. Relay
T50	DOG OPEN TIMER		
T51	VICE OPEN TIMER		



CONTROL OF WORK HANDLING AND CONVEYOR INDEXING FOR PUMA STATION IN SCHOOL OF ENGINEERG FMC	SHEFFIELD HALLAM UNIVERSITY	Appendix F	Proj:PUMA
		Date: 12\12\95	Syst:F1/F2
		Rev.no: 3	Type:Ladder
	Draw.no:	Sign: A.T.	Page: 3

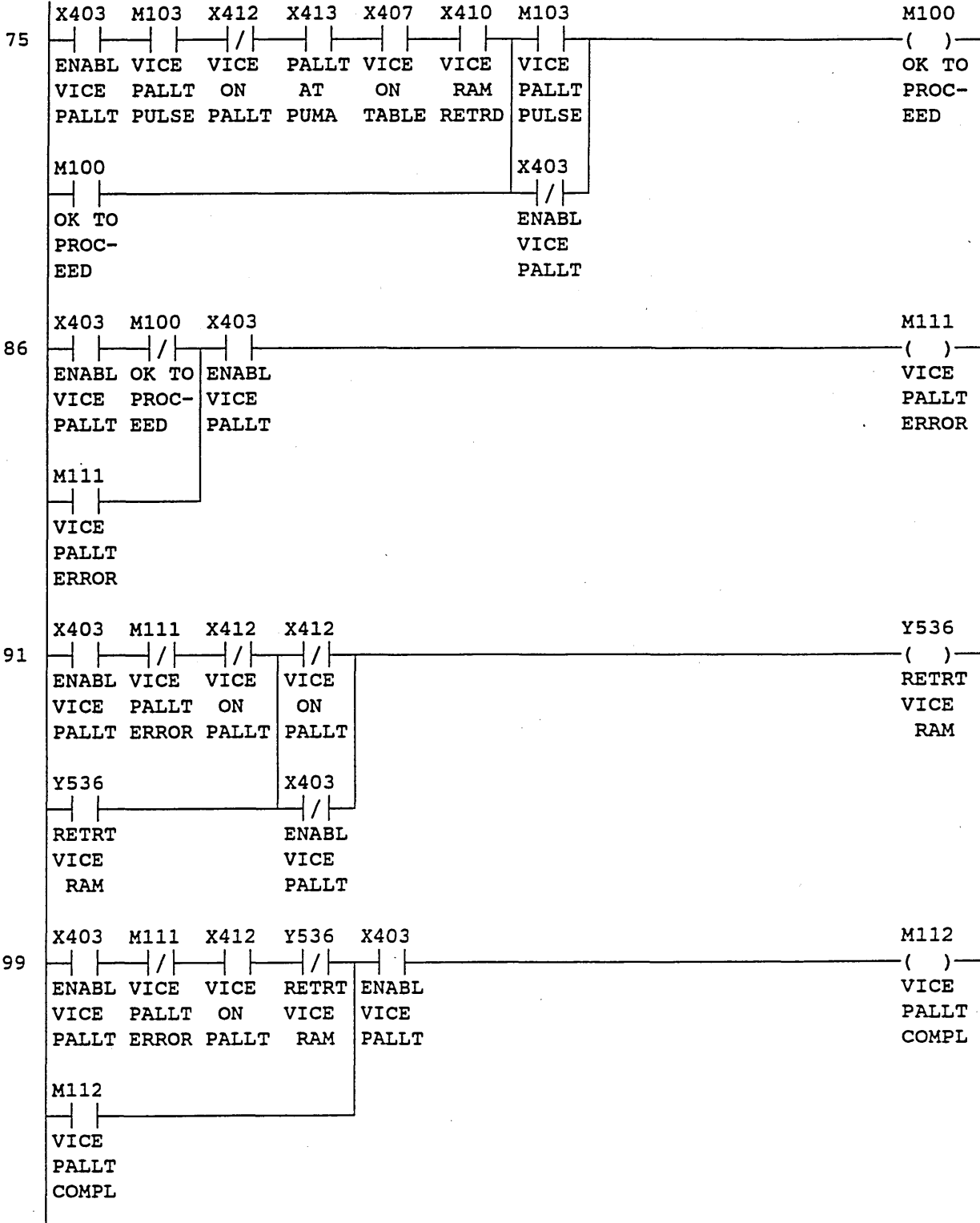


CLOSE VICE





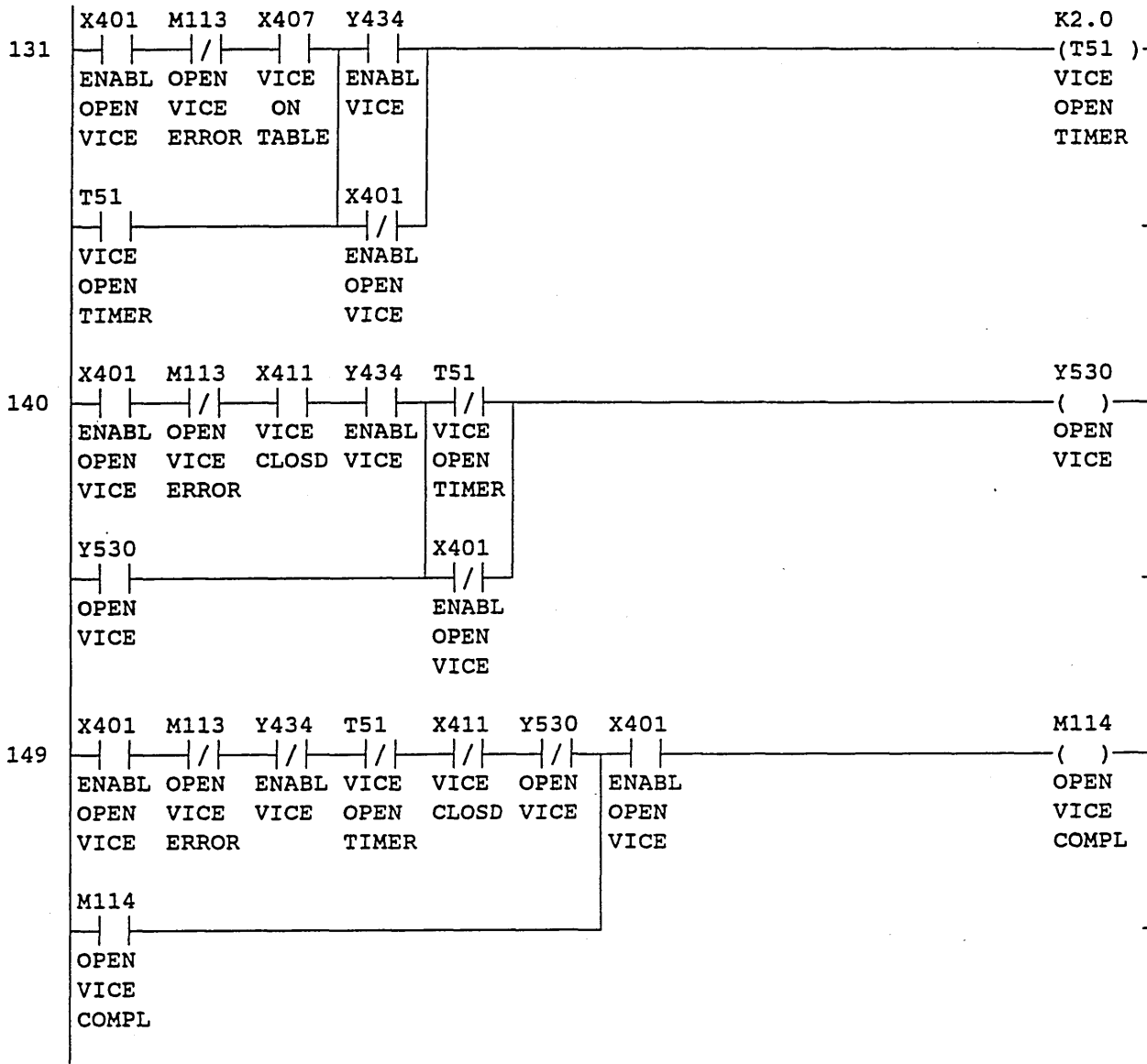
CONTROL OF WORK HANDLING AND CONVEYOR INDEXING FOR PUMA STATION IN SCHOOL OF ENGINEERG FMC	SHEFFIELD HALLAM UNIVERSITY	Appendix F	Proj:PUMA
		Date: 12\12\95	Syst:F1/F2
		Rev.no: 3	Type:Ladder
	Draw.no:	Sign: A.T.	Page: 5



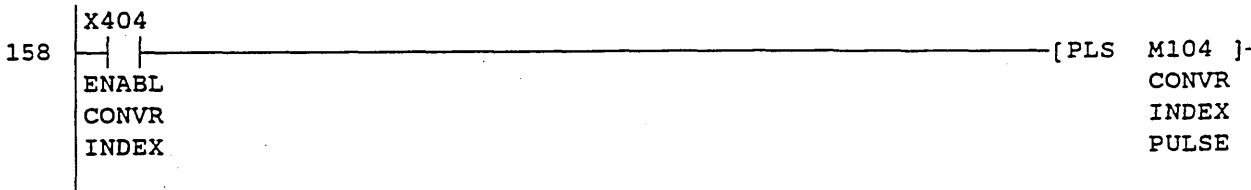




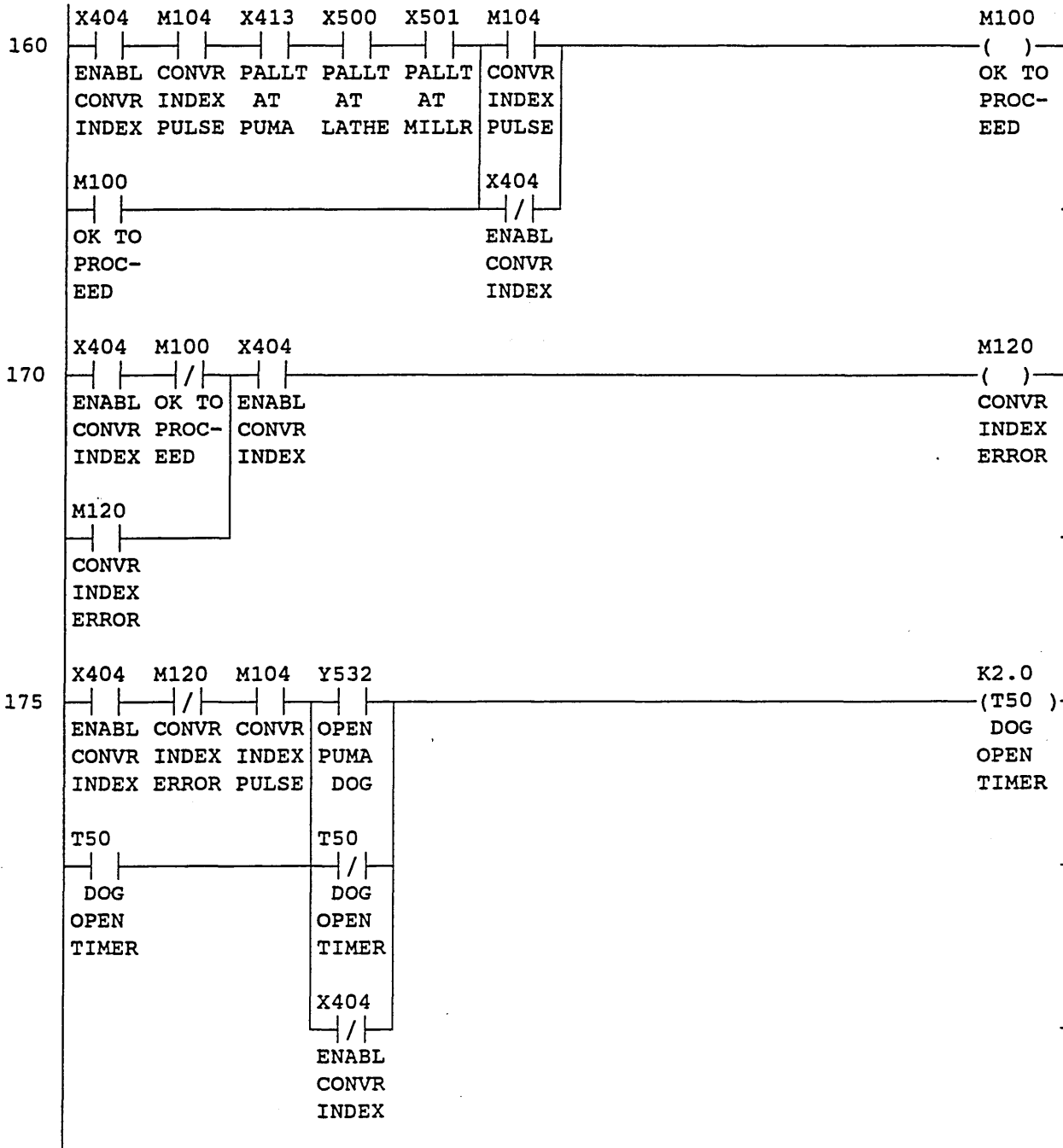
CONTROL OF WORK HANDLING AND CONVEYOR INDEXING FOR PUMA STATION IN SCHOOL OF ENGINEERG FMC	SHEFFIELD HALLAM UNIVERSITY	Appendix F	Proj:PUMA
		Date: 12\12\95	Syst:F1/F2
		Rev.no: 3	Type:Ladder
	Draw.no:	Sign: A.T.	Page: 7



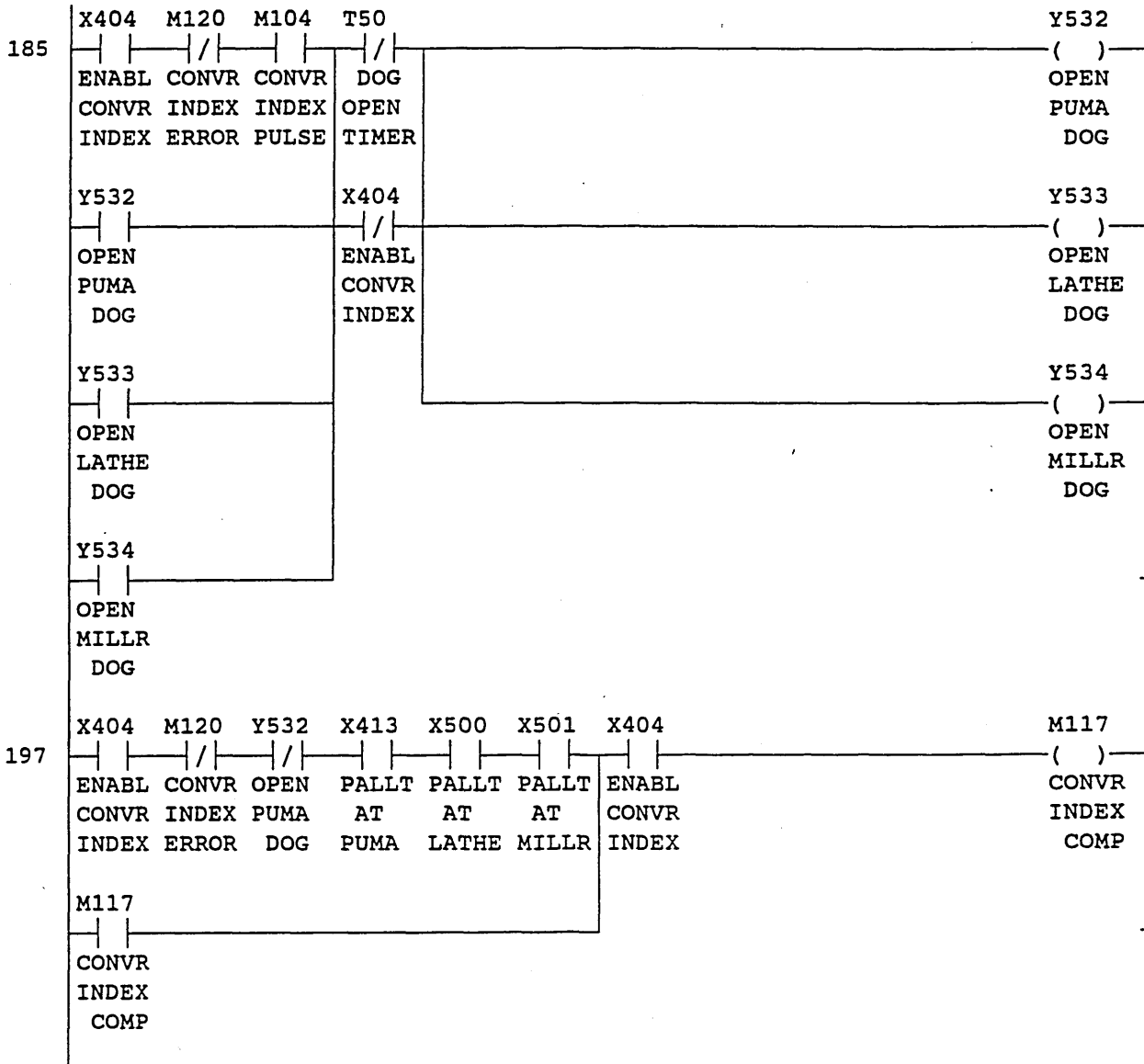
CONVEYOR INDEX



CONTROL OF WORK HANDLING AND CONVEYOR INDEXING FOR PUMA STATION IN SCHOOL OF ENGINEERG FMC	SHEFFIELD HALLAM UNIVERSITY	Appendix F	Proj:PUMA
		Date: 12\12\95	Syst:F1/F2
		Rev.no: 3	Type:Ladder
	Draw.no:	Sign: A.T.	Page: 8

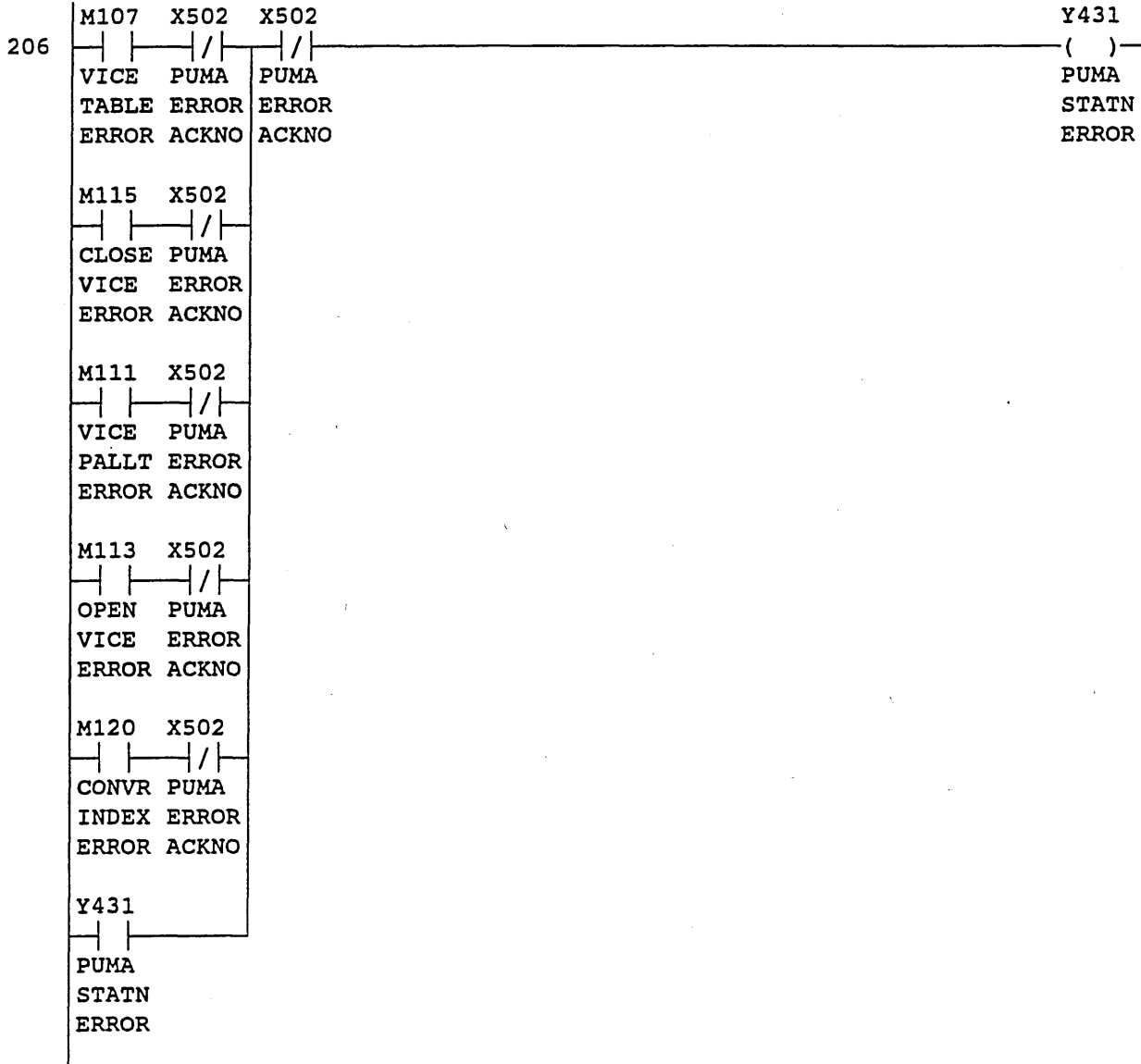


CONTROL OF WORK HANDLING AND CONVEYOR INDEXING FOR PUMA STATION IN SCHOOL OF ENGINEERG FMC	SHEFFIELD HALLAM UNIVERSITY	Appendix F	Proj:PUMA
		Date: 12\12\95	Syst:F1/F2
		Rev.no: 3	Type:Ladder
	Draw.no:	Sign: A.T.	Page: 9



CONTROL OF WORK HANDLING AND CONVEYOR INDEXING FOR PUMA STATION IN SCHOOL OF ENGINEERG FMC	SHEFFIELD HALLAM UNIVERSITY	Appendix F	Proj:PUMA
		Date: 12\12\95	Syst:F1/F2
		Rev.no: 3	Type:Ladder
	Draw.no:	Sign: A.T.	Page: 10

GENERATE ERROR STATUS



CONTROL OF WORK HANDLING AND CONVEYOR INDEXING FOR PUMA STATION IN SCHOOL OF ENGINEERG FMC	SHEFFIELD HALLAM UNIVERSITY	Appendix F	Proj:PUMA
		Date: 12\12\95	Syst:F1/F2
		Rev.no: 3	Type:Ladder
	Draw.no:	Sign: A.T.	Page: 11

GENERATE COMPLETION STATUS

223	M110 X503 X503	Y430
	/	( )
	VICE PUMA PUMA	PUMA
	TABLE COMPL COMPL	STATN
	COMPL ACKNO ACKNO	COMPL
	M116 X503	
	/	
	CLOSE PUMA	
	VICE COMPL	
	COMPL ACKNO	
	M112 X503	
	/	
	VICE PUMA	
	PALLT COMPL	
	COMPL ACKNO	
	M114 X503	
	/	
	OPEN PUMA	
	VICE COMPL	
	COMPL ACKNO	
	M117 X503	
	/	
	CONVR PUMA	
	INDEX COMPL	
	COMP ACKNO	
	Y430	
	PUMA	
	STATN	
	COMPL	
240		[END ]

## **Appendix G Ladder Diagrams Translated from Miller Work**

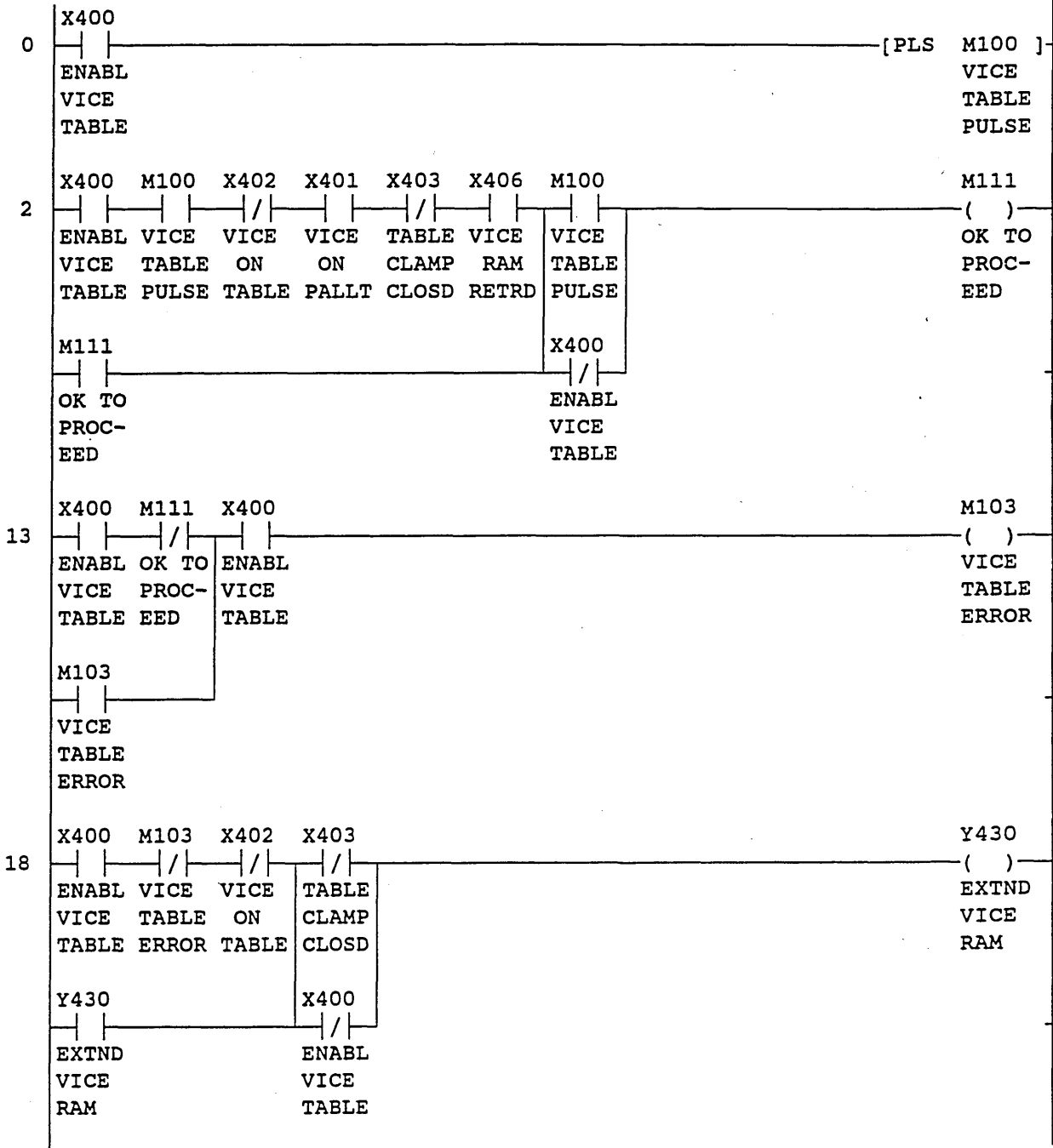
### **Station PN Groups, Using PN $\leftrightarrow$ PLC**

CONTROL OF WORK HANDLING AND M/CING FOR MILLER STATION IN THE SCHOOL OF ENGINEERING'S FMC	SHEFFIELD HALLAM UNIVERSITY	Appendix G	Proj:MILLER
		Date: 08/01/96	Syst:F1/F2
		Rev.no: 2	Type:Name
	Draw.no:	Sign: A.T.	Page: 1

I/O	Name	Comment	Remark
X400	ENABLVICE TABLE	Input from Miller Controller	
X401	VICE ON PALLT	Input from Sensor 8	I.R. Refl.
X402	VICE ON TABLE	Input from Sensor 9	Ind. Prox.
X403	TABLECLAMPCLD	Input from Sensor 10	Ind. Prox.
X404	ENABLSTARTMILLR	Input from Miller Controller	
X405	MILL-ING	Input from Miller Relay	Via M52
X406	VICE RAM RETRD	Input from Sensor 11	Ind. Prox.
X407	ENABLVICE PALLT	Input from Miller Controller	
X410	MILLRERRORACKNO	Input from Miller Controller	
X411	MILLRCOMPLACKNO	Input from Miller Controller	
Y430	EXTNDVICE RAM	Output to Cylinder 1: Extend	Double Sol
Y431	CLOSETABLECLAMP	Output to Cylinder 2: Extend	Double Sol
Y433	MILLG IN PROGR	Output to Miller Controller	
Y435	RETR VICE RAM	Output to Cylinder 1: Retract	Double Sol
Y436	OPEN TABLECLAMP	Output to Cylinder 2: Retract	Double Sol
Y530	MILLRSTATNERROR	Output to Miller Controller	
Y531	MILLRSTATNCOMPL	Output to Miller Controller	
Y534	STARTMILLR	Output to Miller	M52 Reset
M100	VICE TABLEPULSE		Int. Relay
M101	MILLRENABLPULSE		Int. Relay
M102	VICE PALLTPULSE		Int. Relay
M103	VICE TABLEERROR		Int. Relay
M104	STARTMILLRERROR		Int. Relay
M105	VICE PALLTERROR		Int. Relay
M106	VICE TABLECOMPL		Int. Realy
M107	STARTMILLRCOMPL		Int. Relay
M110	VICE PALLTCOMPL		Int. Relay
M111	OK TOPROC-EED		Int. Relay
T51	RAM EXTNDTIMER		

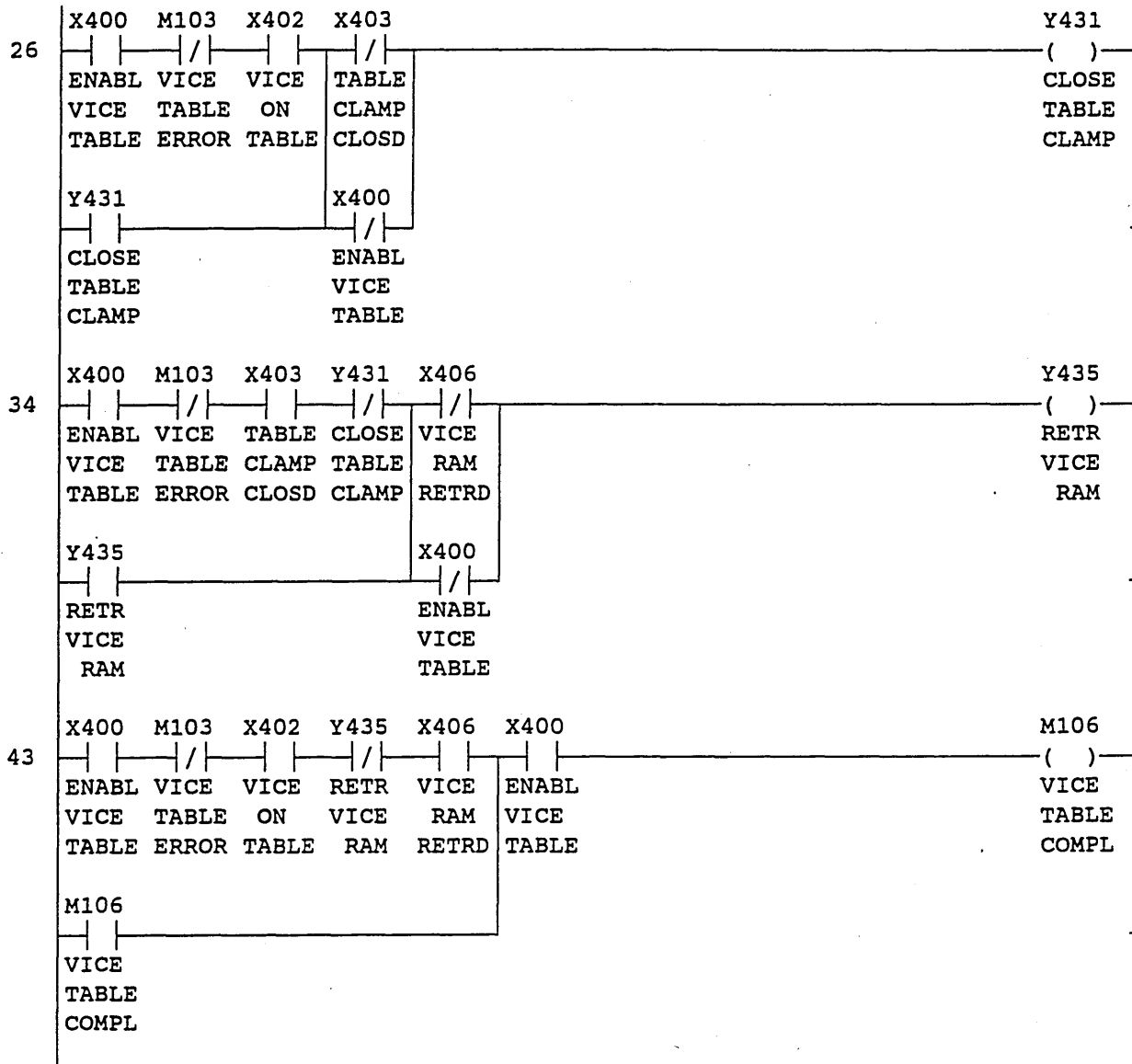
CONTROL OF WORK HANDLING AND M/CING FOR MILLER STATION IN THE SCHOOL OF ENGINEERING'S FMC	SHEFFIELD HALLAM UNIVERSITY	Appendix G	Proj: MILLER
		Date: 08/01/96	Syst: F1/F2
		Rev.no: 2	Type: Ladder
	Draw.no:	Sign: A.T.	Page: 2

VICE TO TABLE

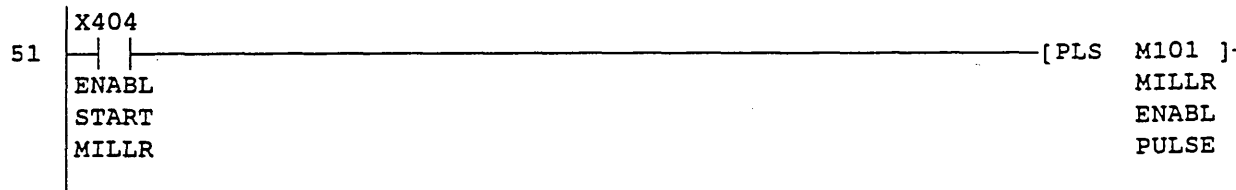




CONTROL OF WORK HANDLING AND M/CING FOR MILLER STATION IN THE SCHOOL OF ENGINEERING'S FMC	SHEFFIELD HALLAM UNIVERSITY	Appendix G	Proj:MILLER
		Date: 08/01/96	Syst:F1/F2
		Rev.no: 2	Type:Ladder
	Draw.no:	Sign: A.T.	Page: 3



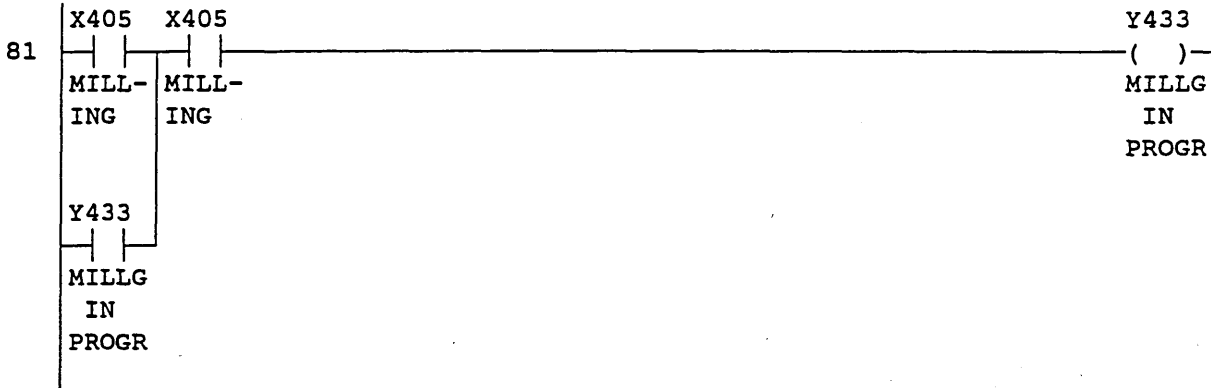
START MILLER



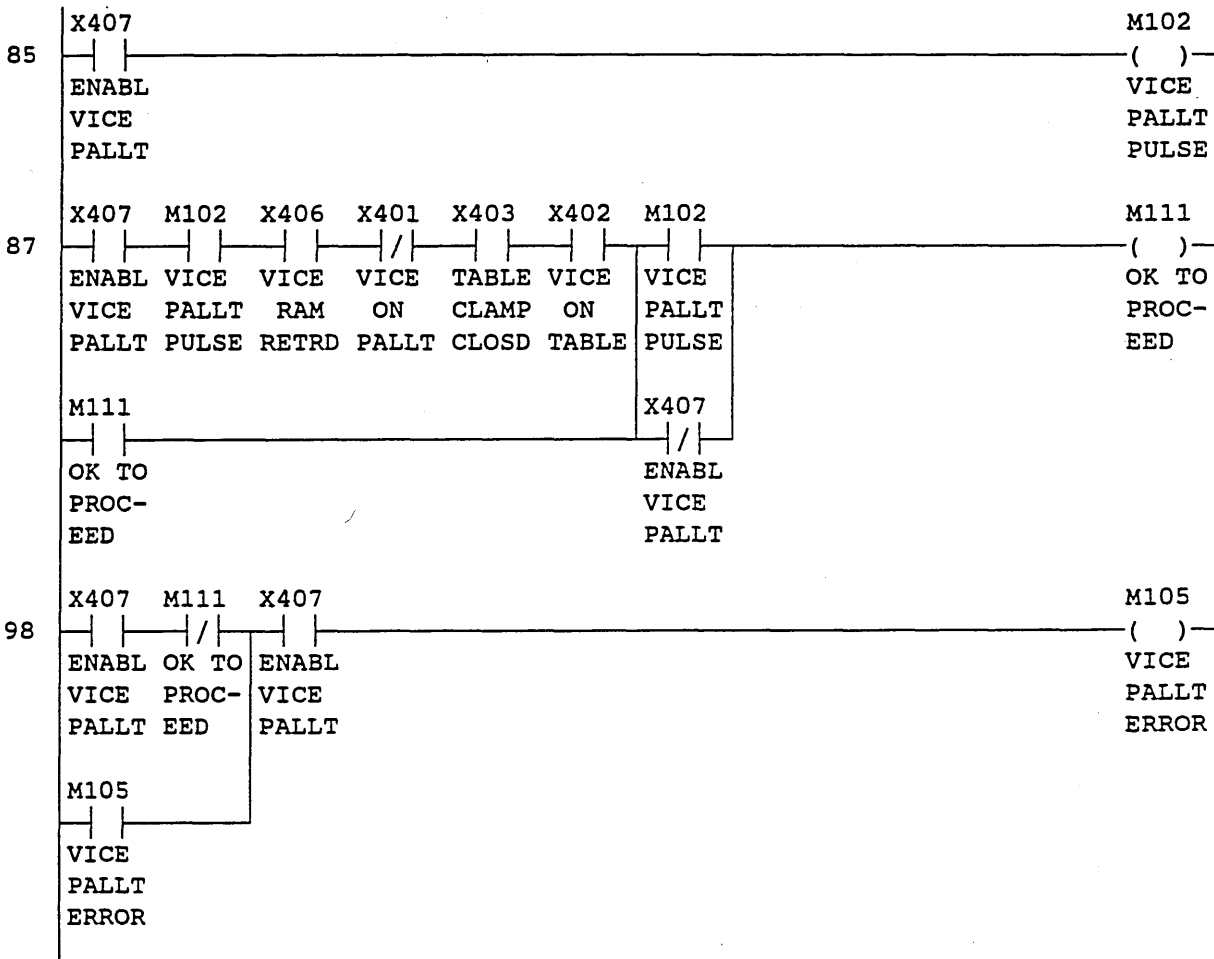


CONTROL OF WORK HANDLING AND M/CING FOR MILLER STATION IN THE SCHOOL OF ENGINEERING'S FMC	SHEFFIELD HALLAM UNIVERSITY	Appendix G	Proj: MILLER
		Date: 08/01/96	Syst: F1/F2
		Rev.no: 2	Type: Ladder
	Draw.no:	Sign: A.T.	Page: 5

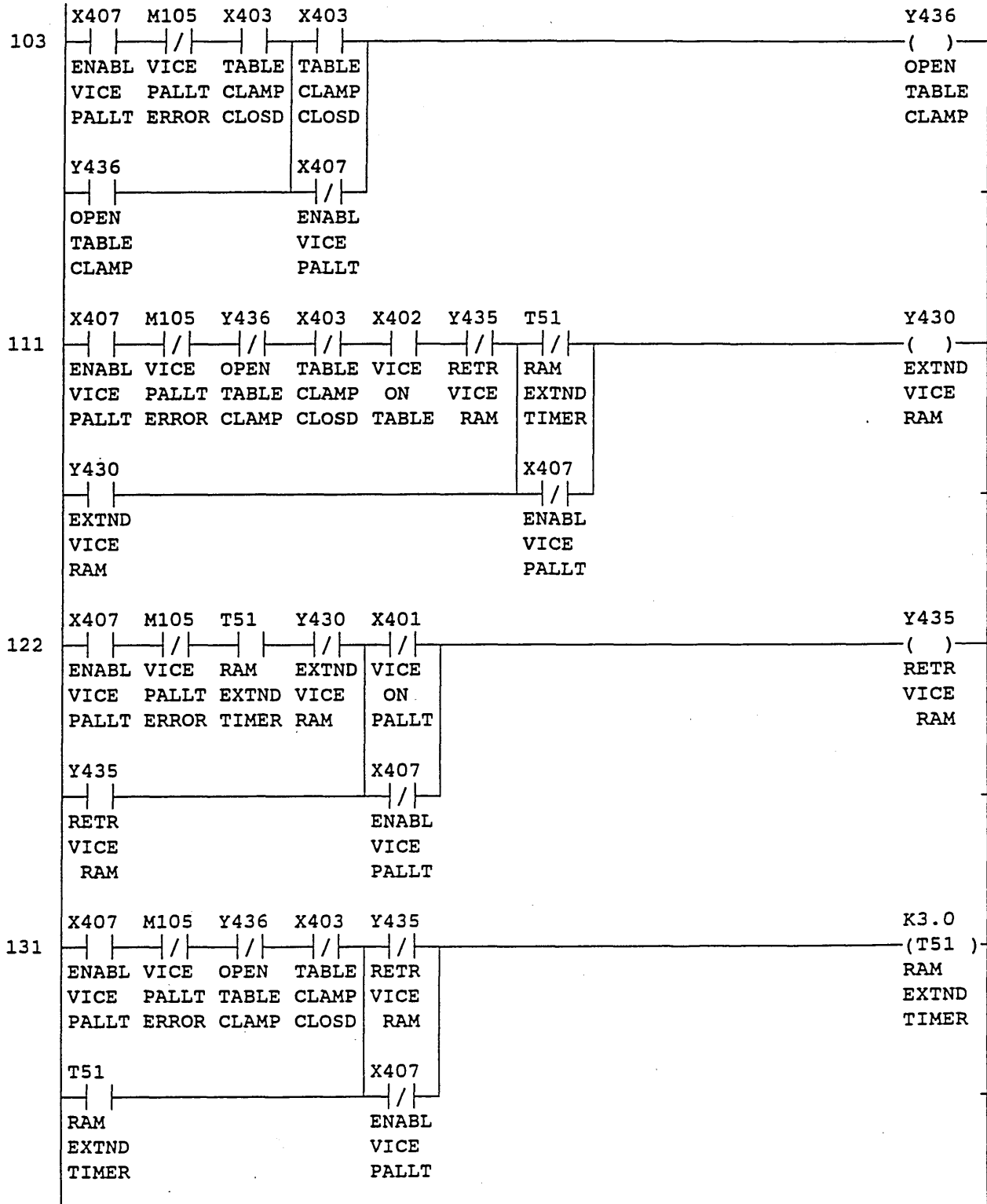
MILLING IN PROGRESS UPDATE TO MC



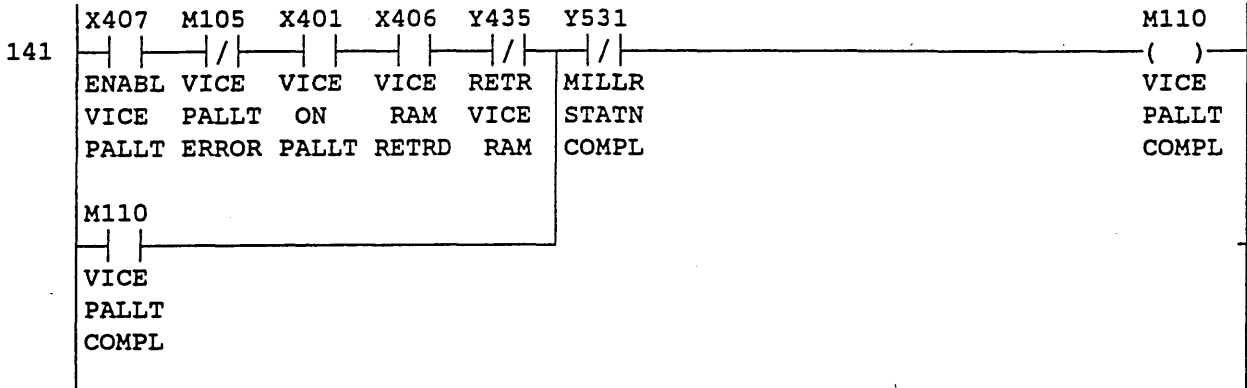
VICE TO PALLET



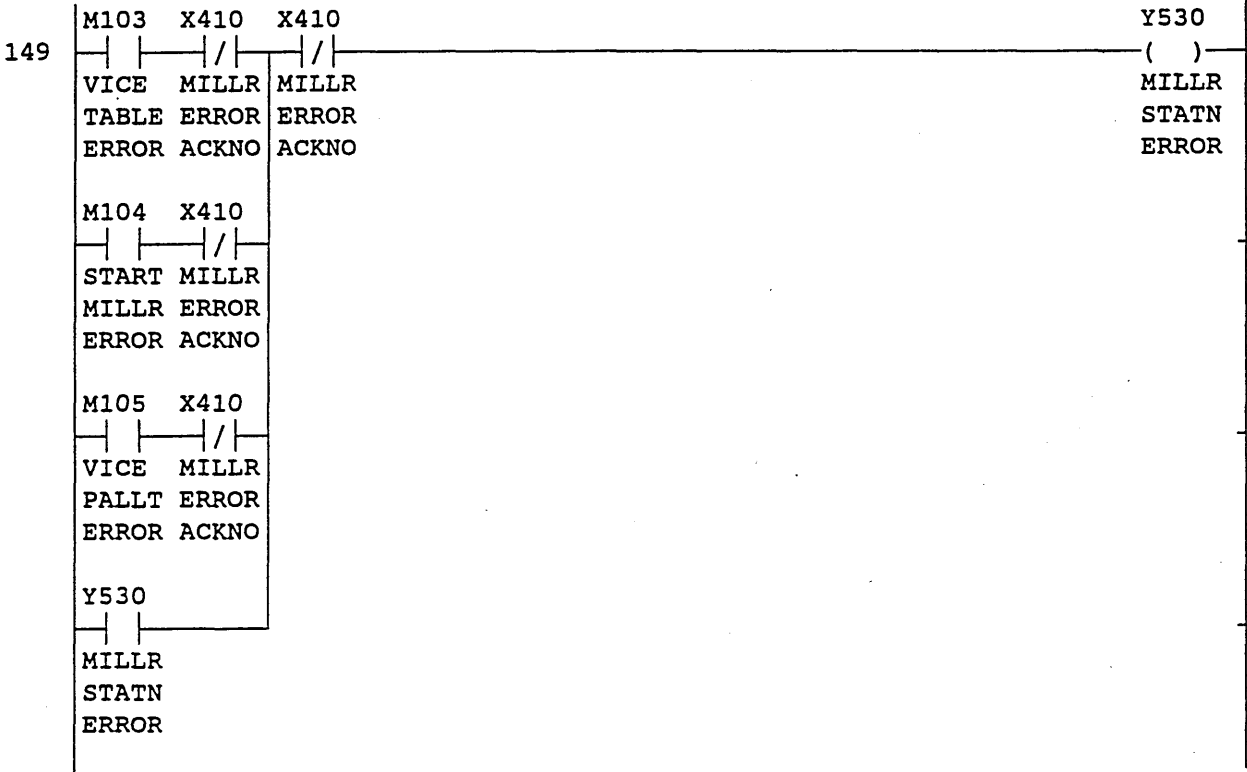
CONTROL OF WORK HANDLING AND M/CING FOR MILLER STATION IN THE SCHOOL OF ENGINEERING'S FMC	SHEFFIELD HALLAM UNIVERSITY	Appendix G	Proj: MILLER
		Date: 08/01/96	Syst: F1/F2
		Rev.no: 2	Type: Ladder
	Draw.no:	Sign: A.T.	Page: 6



CONTROL OF WORK HANDLING AND M/CING FOR MILLER STATION IN THE SCHOOL OF ENGINEERING'S FMC	SHEFFIELD HALLAM UNIVERSITY	Appendix G	Proj:MILLER
		Date: 08/01/96	Syst:F1/F2
		Rev.no: 2	Type:Ladder
	Draw.no:	Sign: A.T.	Page: 7



GENERATE ERROR STATUS



CONTROL OF WORK HANDLING AND M/CING FOR MILLER STATION IN THE SCHOOL OF ENGINEERING'S FMC	SHEFFIELD HALLAM UNIVERSITY	Appendix G	Proj:MILLER
		Date: 08/01/96	Syst:F1/F2
		Rev.no: 2	Type:Ladder
	Draw.no:	Sign: A.T.	Page: 8

GENERATE COMPLETION STATUS

160	M106 X411 X411	Y531
	/     /	( )
	VICE MILLR MILLR	MILLR
	TABLE COMPL COMPL	STATN
	COMPL ACKNO ACKNO	COMPL
	M107 X411	
	/	
	START MILLR	
	MILLR COMPL	
	COMPL ACKNO	
	M110 X411	
	/	
	VICE MILLR	
	PALLT COMPL	
	COMPL ACKNO	
	Y531	
	MILLR	
	STATN	
	COMPL	
171		[END ]

**Appendix H Step Ladder Diagrams Translated from Lathe  
Work Station PN Steps, Using PN $\Leftrightarrow$ PLC**

CONTROL OF WORK HANDLING, GANTRY ROBOT AND MACHINING FOR LATHE STATION IN THE SCHOOL OF ENGINEERING'S FMC	SHEFFIELD HALLAM UNIVERSITY	Appendix H	Proj:LATHE
		Date: 18/01/96	Syst:F1/F2
		Rev.no: 2	Type:Name
	Draw.no:	Sign: A.T.	Page: 1

I/O	Name	Comment	Remark
X0	ENABLVICE TABLE	Input from Lathe Controller	
X1	ENABLGRIP WKPCE	Input from Lathe Controller	
X2	ENABLWKPCE CHUK	Input from Lathe Controller	
X3	ENABLROBOT SAFE	Input from Lathe Controller	
X4	ENABLVICE PALLT	Input from Lathe Controller	
X5	ENABLSTARTM/C'G	Input from Lathe Controller	
X6	ENABLWKPCE CONV	Input from Lathe Controller	
X7	ENABLWKPCE VICE	Input from Lathe Controller	
X10	VICE ON TABLE	Input from sensor 12	Ind. Prox.
X11	VICE RAM RETRD	Input from sensor 13	Magn. Reed
X12	VICE CLOSE	Input from sensor 14	Press. Sw.
X13	VICE ON PALLT	Input from sensor 15	I.R. Refl.
X14	HORIZ AT CONVR	Input from sensor B01	Ind. Prox.
X15	HORIZ AT LATHE	Input from sensor B02	Ind. Prox.
X16	VERT AT CHUCK	Input from sensor B03	Ind. Prox.
X17	VERT UP SAFE	Input from sensor B04	Ind. Prox.
X400	MIDSTVERT EXT	Input from sensor B05	Press. Sw.
X401	MIDSTVERT RET	Input from sensor B06	Magn. Reed
X402	ROTATO DEG	Input from sensor B08	Ind. Prox.
X403	ROTAT 90 DEG	Input from sensor B09	Ind. Prox.
X404	ENDSTRETRT	Input from sensor B11	Magn. Reed
X405	ENDSTEXTND	Input from sensor B12	Magn. Reed
X406	GRIPRCLOSE	Input from sensor B13	Magn. Reed
X407	GRIPR OPEN	Input from sensor B14	Press. Sw.
X410	GRIPR AT VICE	Input from sensor B17	Ind. Prox.
X411	CHUCKOPEN	Input from CNC Lathe	Code M61
X412	CHUCKCLOSE	Input from CNC Lathe	Code M62
X413	M/C'GFINSH	Input from CNC Lathe	Code M63
X500	IN CYCLE	Input from CNC Lathe	Int. Relay
X501	M/C'GERROR	Input from CNC Lathe	Int. Relay
X503	TOP DOOR OPEN	Input from sensor 16	Magn. Reed
X504	TOP DOOR CLOSE	Input from sensor 17	Magn. Reed
X505	PALET AT LATHE	Input from sensor 6	Cap. Prox.
X506	LATHEERRORACKNO	Input from Lathe Controller	
X507	LATHESTATNCOMPL	Input from Lathe Controller	
Y30	LATHESTATNCOMP	Output to Lathe Controller	
Y31	LATHESTATNEROR	Output to Lathe Controller	
Y32	EXTNDVICE RAM	Output to Ram cyl.(Extend)	Double Sol
Y33	RETRTVICE RAM	Output to Ram cyl.(Retract)	Double Sol
Y34	OPEN VICE	Output to Vice cyl.(Retract)	Double Sol
Y35	CLOSE VICE	Output to Vice cyl.(Extend)	Double Sol
Y36	ENABLVICE	Output to Toggle Clamp	1Sol Sprng



CONTROL OF WORK HANDLING, GANTRY ROBOT AND MACHINING FOR LATHE STATION IN THE SCHOOL OF ENGINEERING'S FMC	SHEFFIELD HALLAM UNIVERSITY	Appendix H	Proj:LATHE
		Date: 18/01/96	Syst:F1/F2
		Rev.no: 2	Type:Name
	Draw.no:	Sign: A.T.	Page: 2

I/O	Name	Comment	Remark
Y37	HORIZ TO CONVR	Output to Sol. Y1	3 Pos 2Sol
Y40	HORIZ TO LATHE	Output to Sol. Y2	3 Pos 2Sol
Y41	VERT DOWN	Output to Sol. Y3	3 Pos 2Sol
Y42	VERT UP	Output to Sol. Y4	3 Pos 2Sol
Y43	EXTNDVERT MIDST	Output to Sol. Y5	Double Sol
Y44	RETRTVERT MIDST	Output to Sol. Y6	Double Sol
Y45	ROTATTO 0 DEG	Output to Sol. Y8	1Sol Sprng
Y430	ROTATTO 90DEG	Output to Sol. Y9	1Sol Sprng
Y431	EXTND END STOP	Output to Sol. Y10	Double Sol
Y432	RETRT END STOP	Output to Sol. Y11	Double Sol
Y433	CLOSEGRIPR	Output to Sol. Y12	Double Sol
Y434	OPEN GRIPR	Output to Sol. Y13	Double Sol
Y435	STARTCYCLE	Output to Lathe	Int. Relay
Y436	STOPCYCLE	Output to Lathe	Int. Relay
Y437	M/C STROB	Output to Lathe	Int. Relay
Y530	OPEN TOP DOOR	Output to Sol	Double Sol
Y531	CLOSETOP DOOR	Output to Sol	Double Sol
M100	VICE/TABLECOMPL		Int. Relay
M101	VICE/TABLE ERR		Int. Relay
M102	GRIP WKPCECOMPL		Int. Relay
M103	GRIP WKPCE ERR		Int. Relay
M104	WKPCE/CHUKCOMPL		Int. Relay
M105	WKPCE/CHUK ERR		Int. Relay
M106	ROBOT/SAFECOMPL		Int. Relay
M107	ROBOT/SAFE ERR		Int. Relay
M110	VICE/PALLTCOMPL		Int. Relay
M111	VICE/PALLT ERR		Int. Relay
M112	M/C'GCOMPL		Int. Relay
M113	STARTM/C'G ERR		Int. Relay
M114	WKPCE/CONVCOMPL		Int. Relay
M115	WKPCE/CONV ERR		Int. Relay
M116	WKPCE/VICECOMPL		Int. Relay
M117	WKPCE/VICE ERR		Int. Relay
M122	VICE/TABLE PLS		Int. Relay
M123	GRIP WKPCE PLS		Int. Relay
M124	WKPCECHUCK PLS		Int. Relay
M125	ROBOT SAFE PLS		Int. Relay
M126	VICE PALLT PLS		Int. Relay
M127	STARTM/C'G PLS		Int. Relay
M130	WKPCECONVR PLS		Int. Relay
M131	WKPCEVICE PLS		Int. Relay
M132	OK TOPROC-EED		Int. Relay
T50	VICE OPEN TIMER		

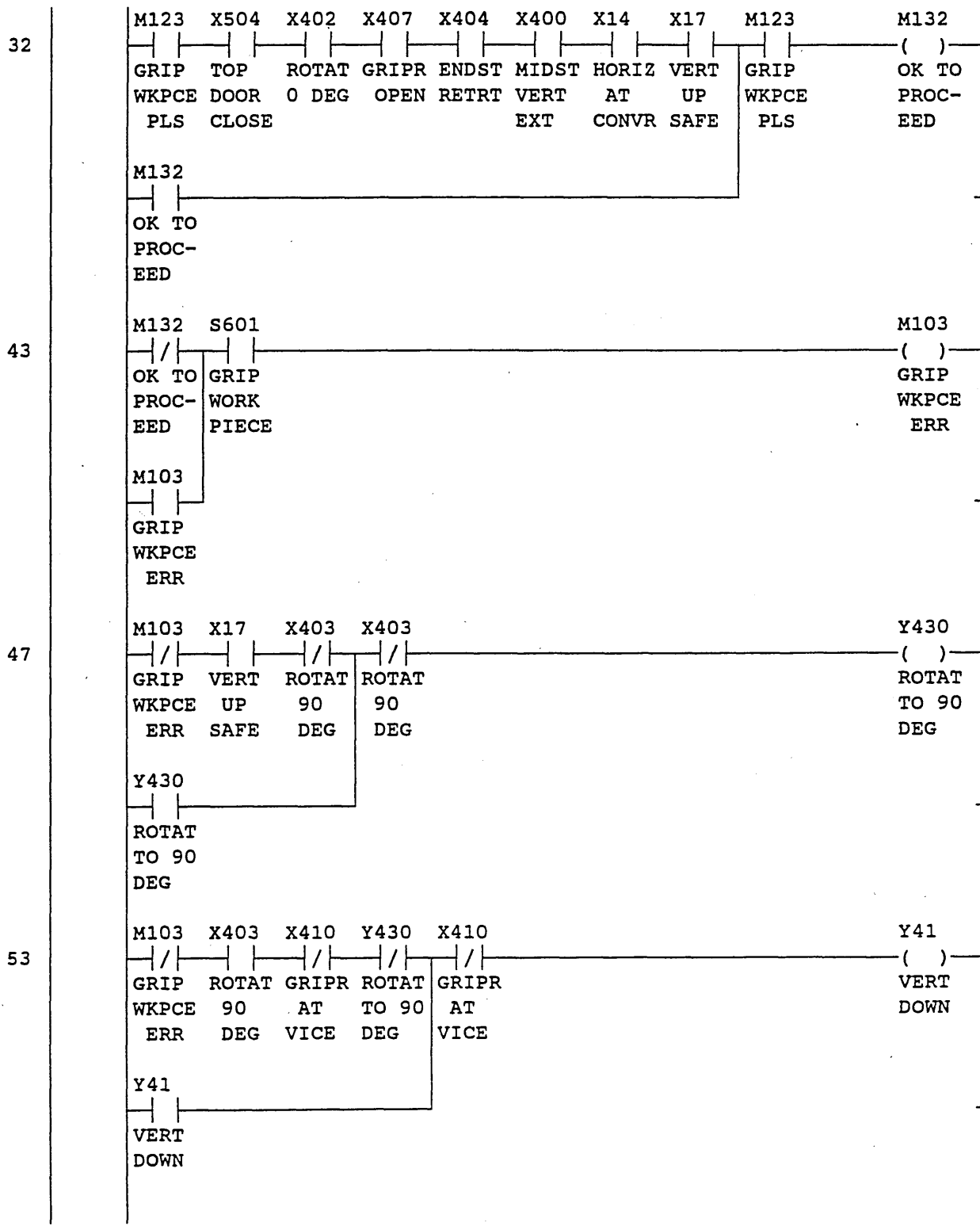
CONTROL OF WORK HANDLING, GANTRY ROBOT AND MACHINING FOR LATHE STATION IN THE SCHOOL OF ENGINEERING'S FMC	SHEFFIELD HALLAM UNIVERSITY	Appendix H	Proj:LATHE
		Date: 18/01/96	Syst:F1/F2
		Rev.no: 2	Type:Name
	Draw.no:	Sign: A.T.	Page: 3

I/O	Name	Comment	Remark
T51	VICE CLOSETIMER		
S600	VICE TO TABLE		Step
S601	GRIP WORK PIECE		Step
S602	WKPCE TO CHUCK		Step
S603	ROBOT SAFE POSN		Step
S604	VICE TO PALLT		Step
S605	STARTM/C'G		Step
S606	WKPCE TO CONVR		Step
S607	WKPCE TO VICE		Step

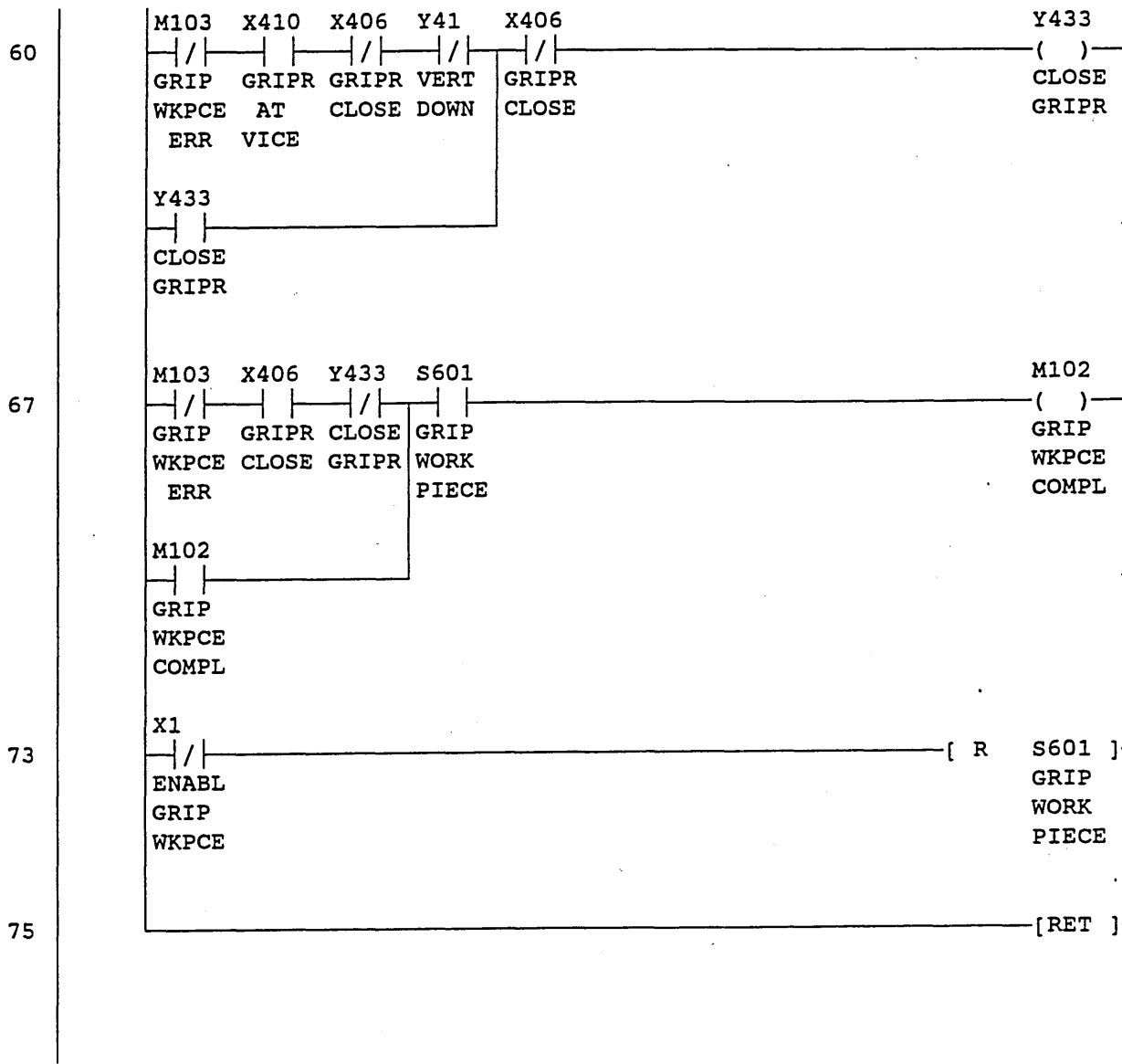




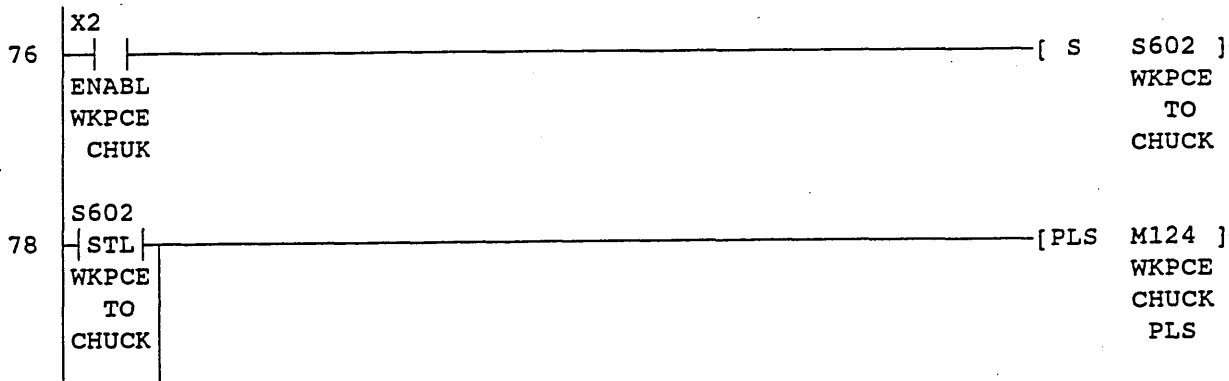
CONTROL OF WORK HANDLING, GANTRY ROBOT AND MACHINING FOR LATHE STATION IN THE SCHOOL OF ENGINEERING'S FMC	SHEFFIELD HALLAM UNIVERSITY	Appendix H	Proj:LATHE
		Date: 18/01/96	Syst:F1/F2
		Rev.no: 2	Type:Ladder
	Draw.no:	Sign: A.T.	Page: 6



CONTROL OF WORK HANDLING, GANTRY ROBOT AND MACHINING FOR LATHE STATION IN THE SCHOOL OF ENGINEERING'S FMC	SHEFFIELD HALLAM UNIVERSITY	Appendix H	Proj: LATHE
		Date: 18/01/96	Syst: F1/F2
		Rev.no: 2	Type: Ladder
	Draw.no:	Sign: A.T.	Page: 7

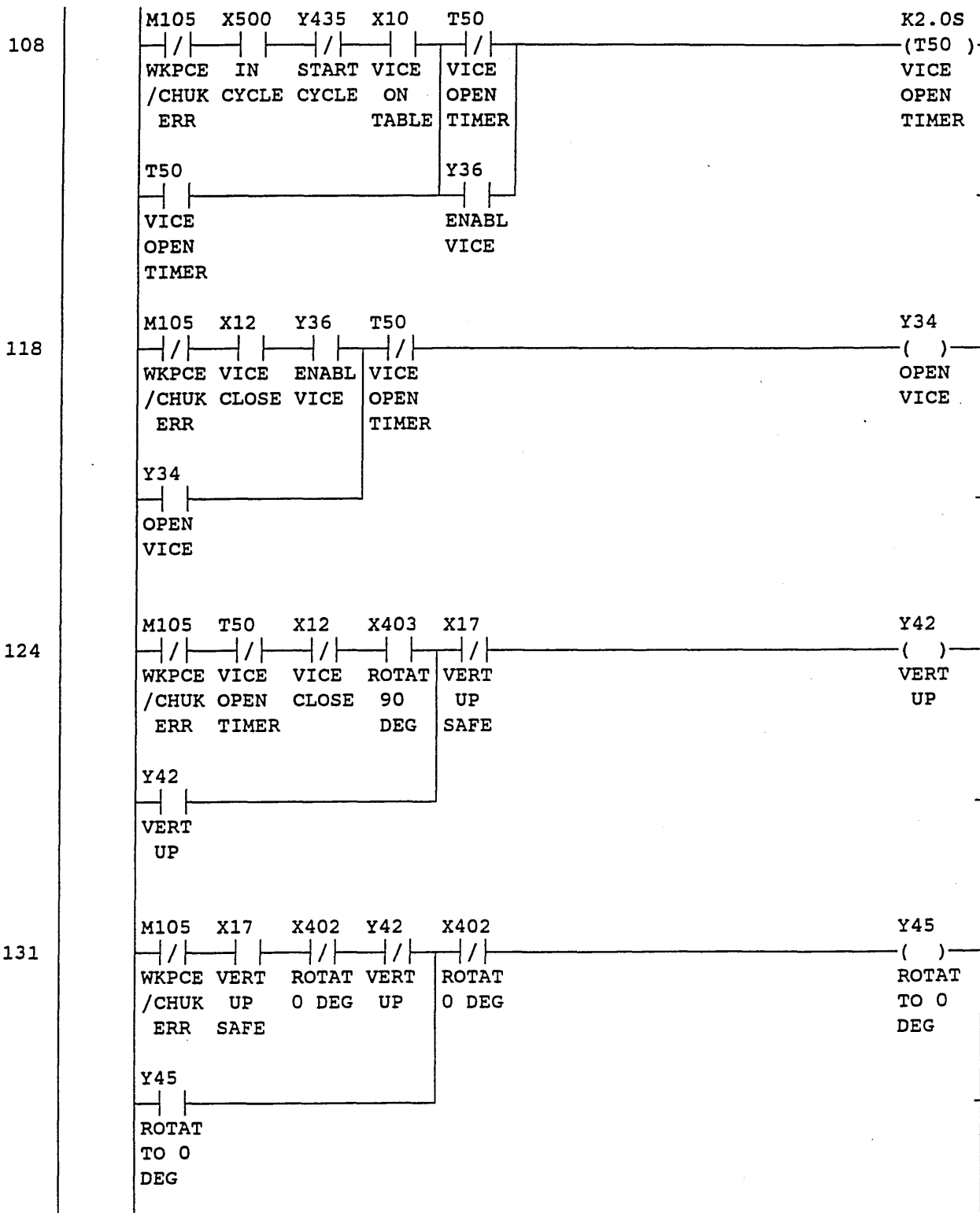


WORKPIECE TO CHUCK





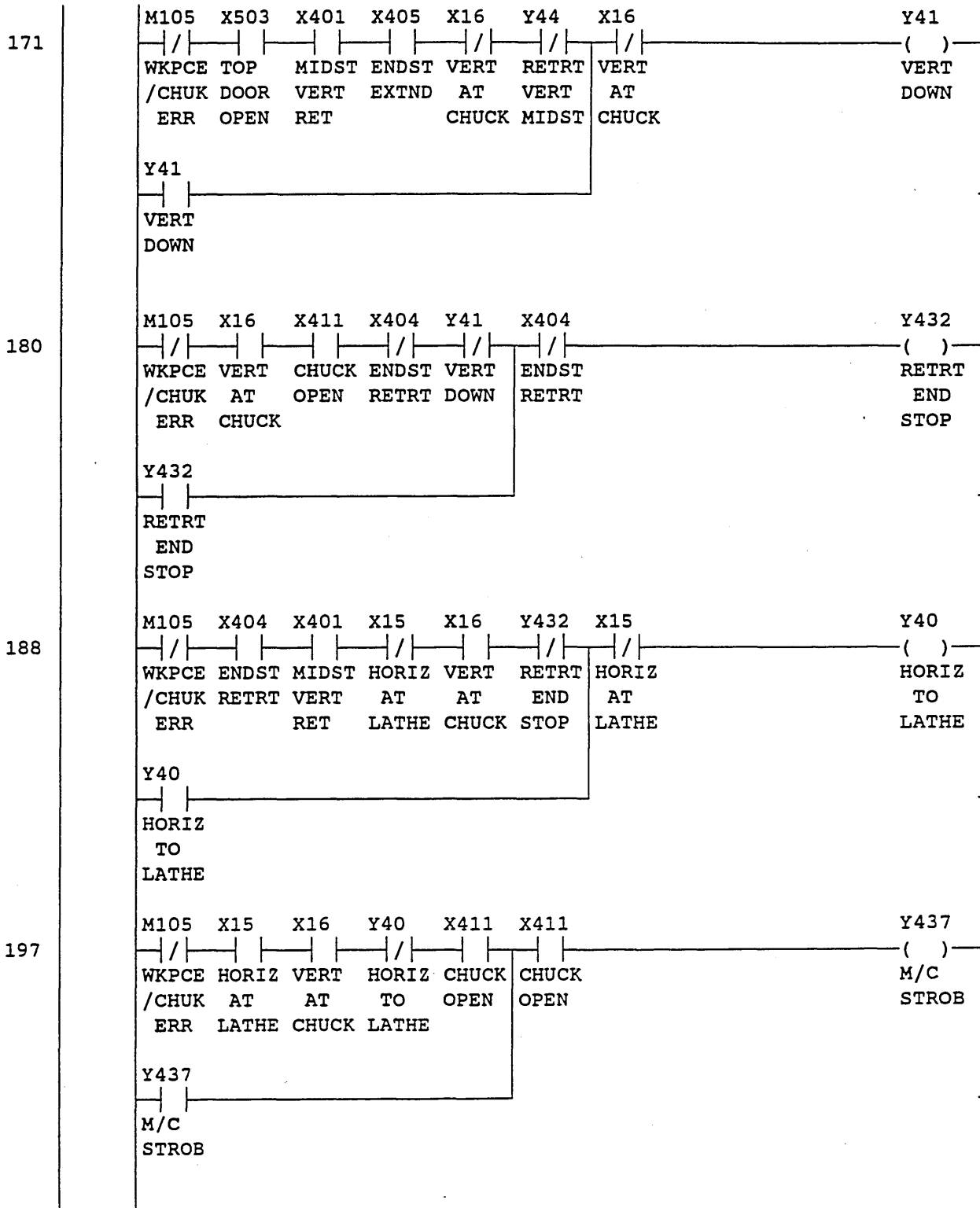
CONTROL OF WORK HANDLING, GANTRY ROBOT AND MACHINING FOR LATHE STATION IN THE SCHOOL OF ENGINEERING'S FMC	SHEFFIELD HALLAM UNIVERSITY	Appendix H	Proj:LATHE
		Date: 18/01/96	Syst:F1/F2
		Rev.no: 2	Type:Ladder
	Draw.no:	Sign: A.T.	Page: 9



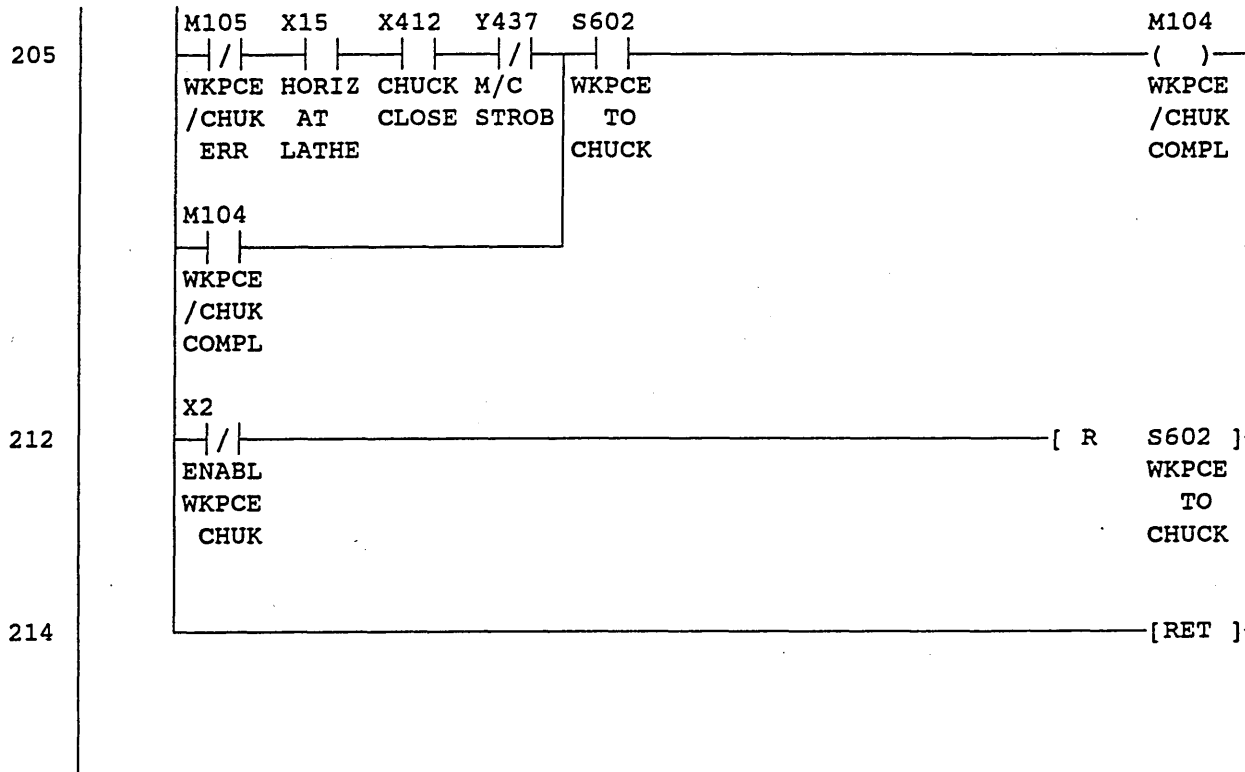




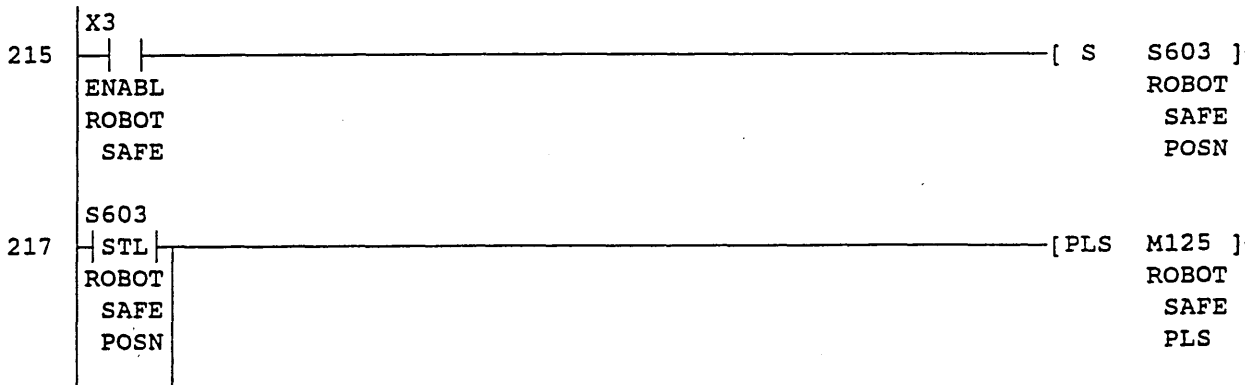
CONTROL OF WORK HANDLING, GANTRY ROBOT AND MACHINING FOR LATHE STATION IN THE SCHOOL OF ENGINEERING'S FMC	SHEFFIELD HALLAM UNIVERSITY	Appendix H	Proj:LATHE
		Date: 18/01/96	Syst:F1/F2
		Rev.no: 2	Type:Ladder
	Draw.no:	Sign: A.T.	Page: 11



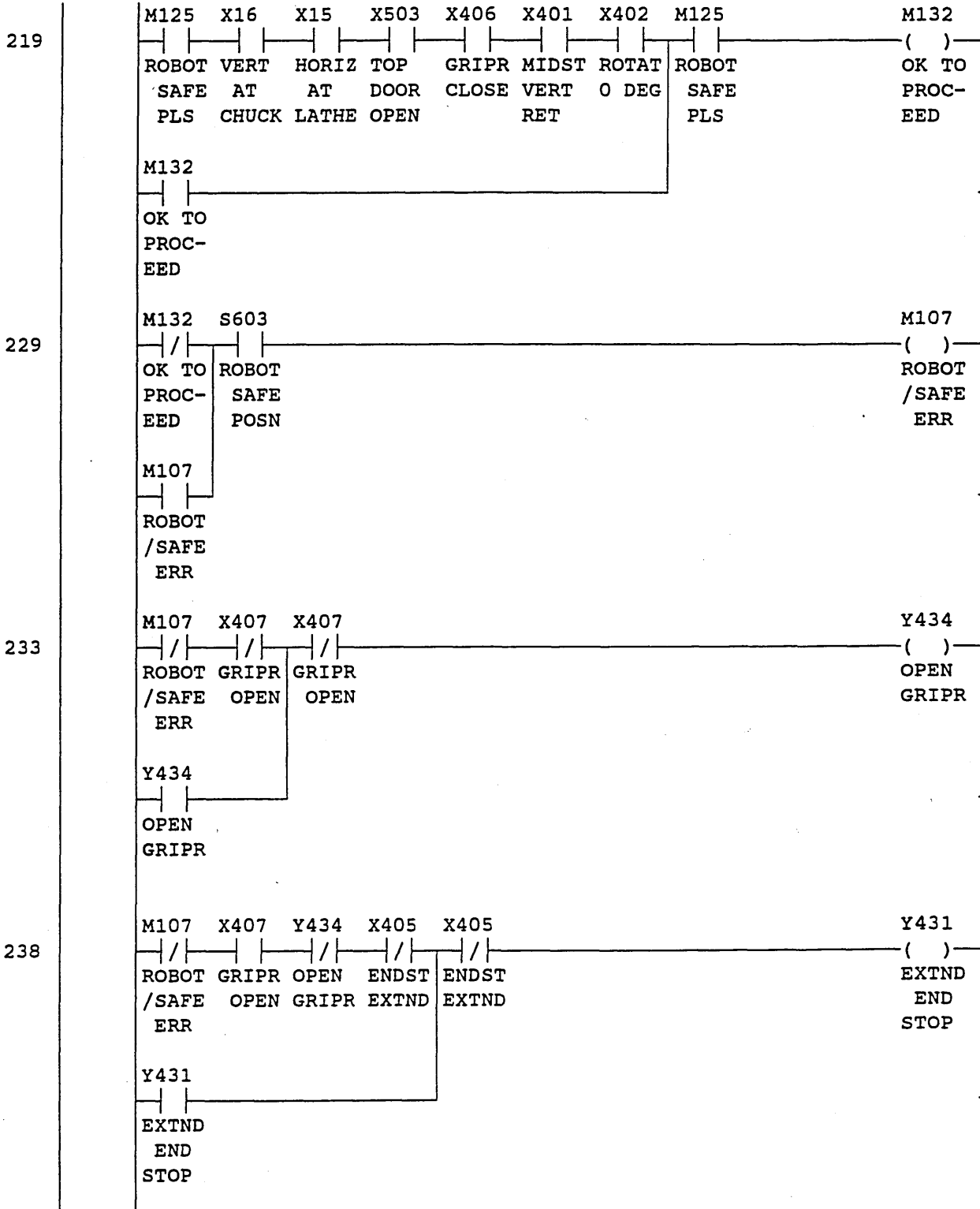
CONTROL OF WORK HANDLING, GANTRY ROBOT AND MACHINING FOR LATHE STATION IN THE SCHOOL OF ENGINEERING'S FMC	SHEFFIELD HALLAM UNIVERSITY	Appendix H	Proj:LATHE
		Date: 18/01/96	Syst:F1/F2
		Rev.no: 2	Type:Ladder
	Draw.no:	Sign: A.T.	Page: 12



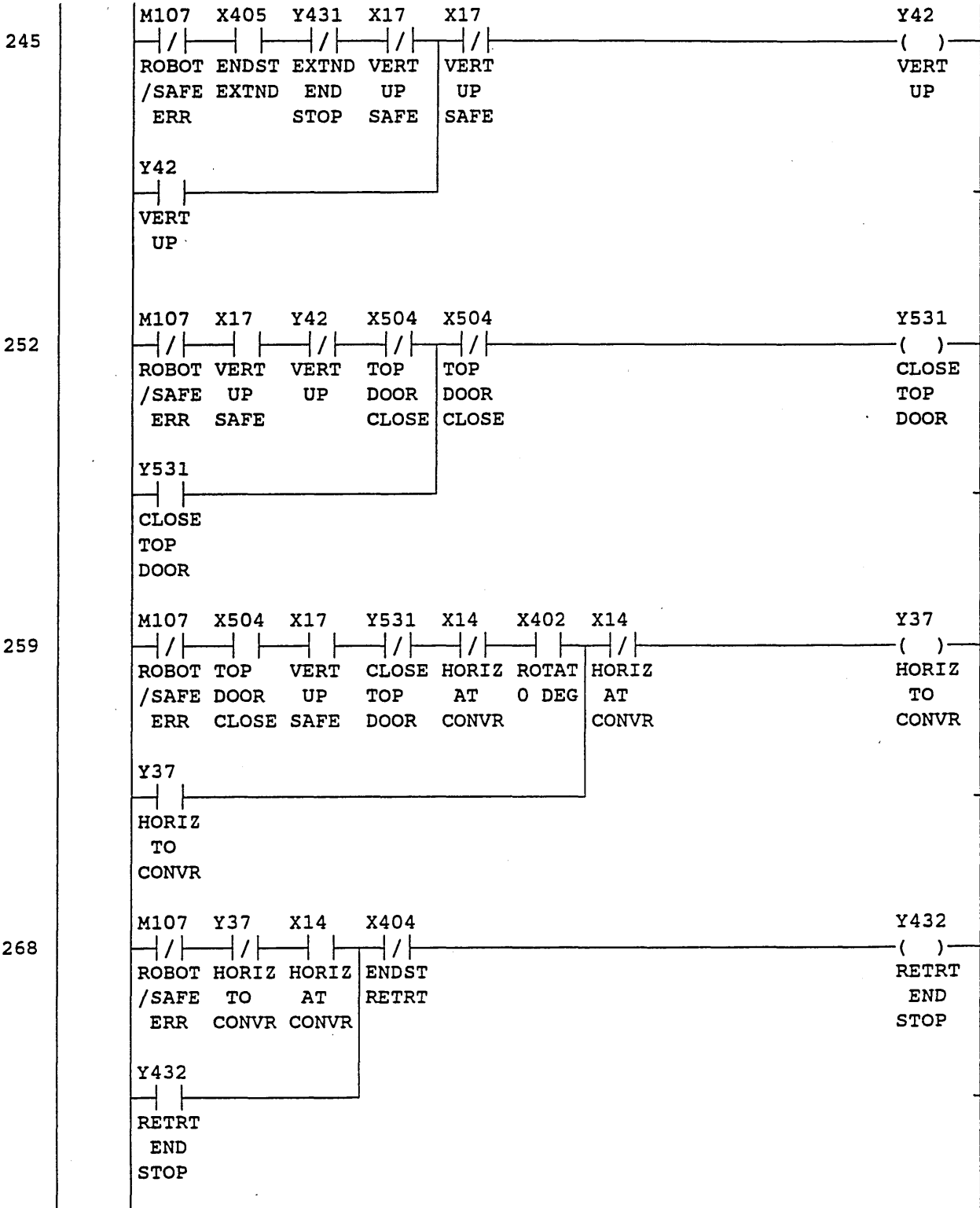
ROBOT TO SAFE POSITION



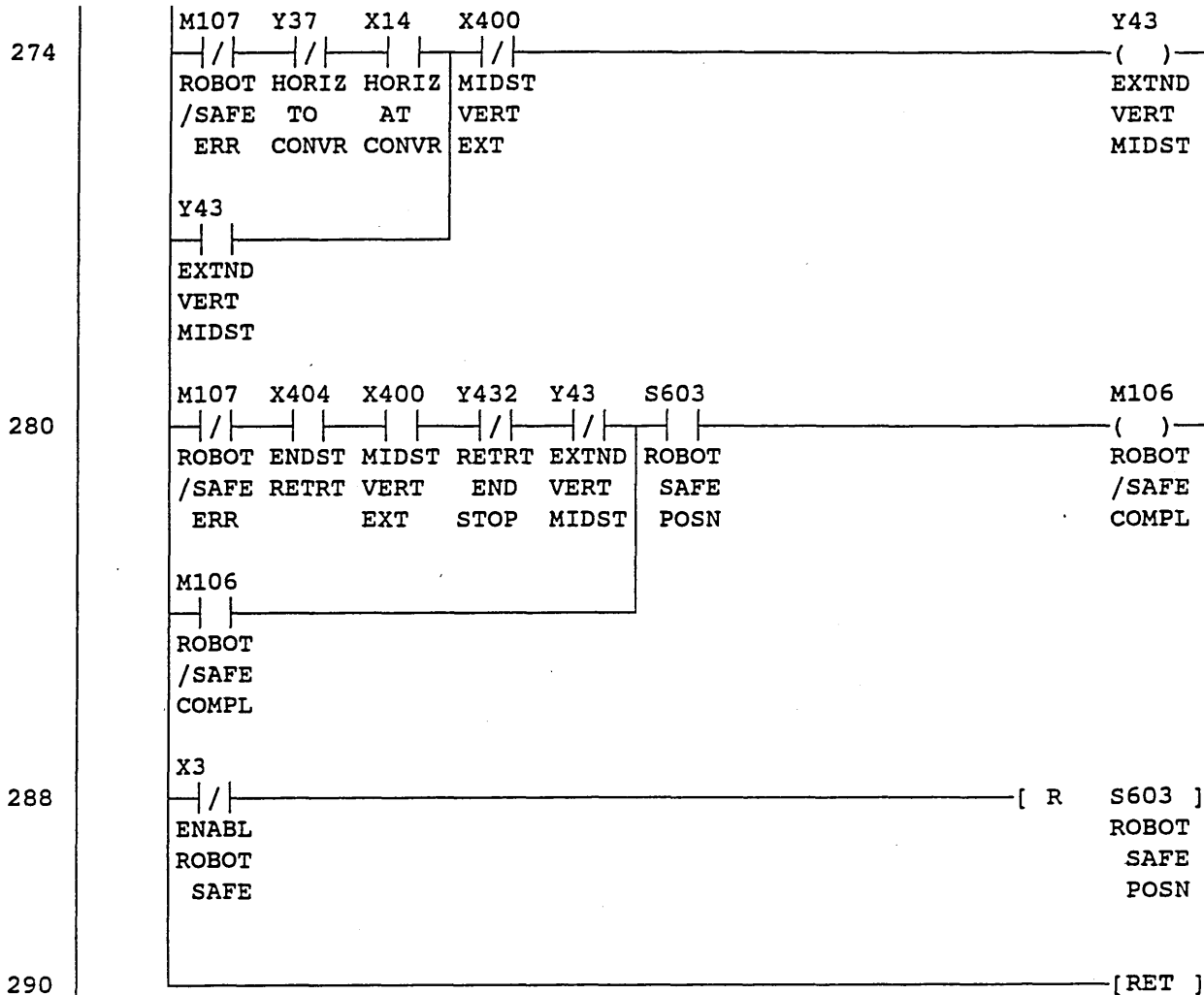
CONTROL OF WORK HANDLING, GANTRY ROBOT AND MACHINING FOR LATHE STATION IN THE SCHOOL OF ENGINEERING'S FMC	SHEFFIELD HALLAM UNIVERSITY	Appendix H	Proj:LATHE
		Date: 18/01/96	Syst:F1/F2
		Rev.no: 2	Type:Ladder
	Draw.no:	Sign: A.T.	Page: 13



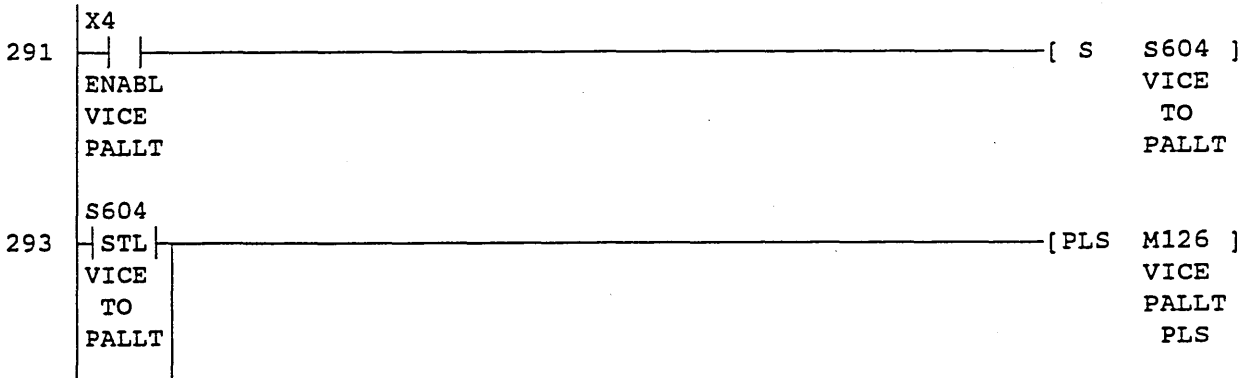
CONTROL OF WORK HANDLING, GANTRY ROBOT AND MACHINING FOR LATHE STATION IN THE SCHOOL OF ENGINEERING'S FMC	SHEFFIELD HALLAM UNIVERSITY	Appendix H	Proj:LATHE
		Date: 18/01/96	Syst:F1/F2
		Rev.no: 2	Type:Ladder
	Draw.no:	Sign: A.T.	Page: 14



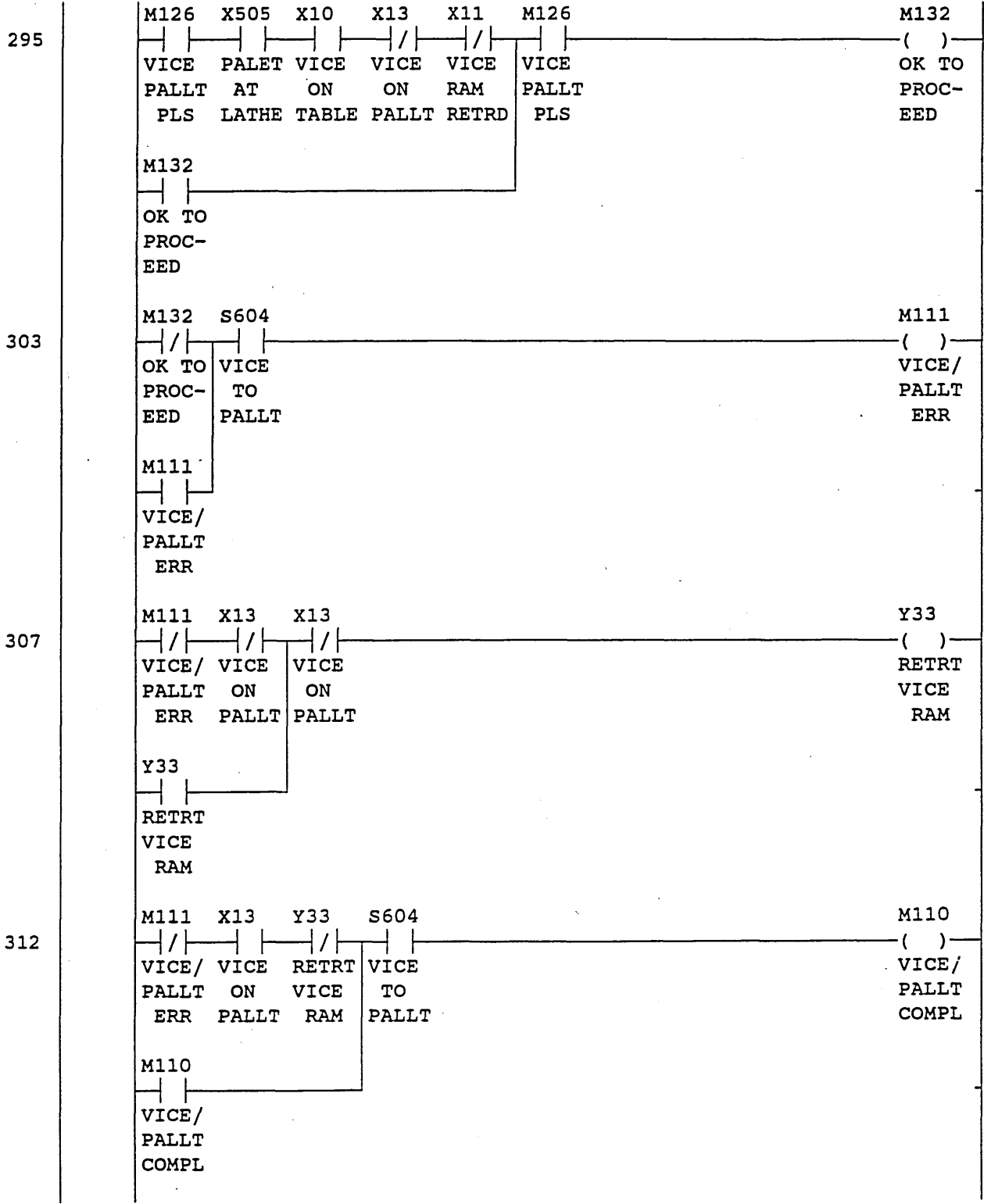
CONTROL OF WORK HANDLING, GANTRY ROBOT AND MACHINING FOR LATHE STATION IN THE SCHOOL OF ENGINEERING'S FMC	SHEFFIELD HALLAM UNIVERSITY	Appendix H	Proj:LATHE
		Date: 18/01/96	Syst:F1/F2
		Rev.no: 2	Type:Ladder
	Draw.no:	Sign: A.T.	Page: 15



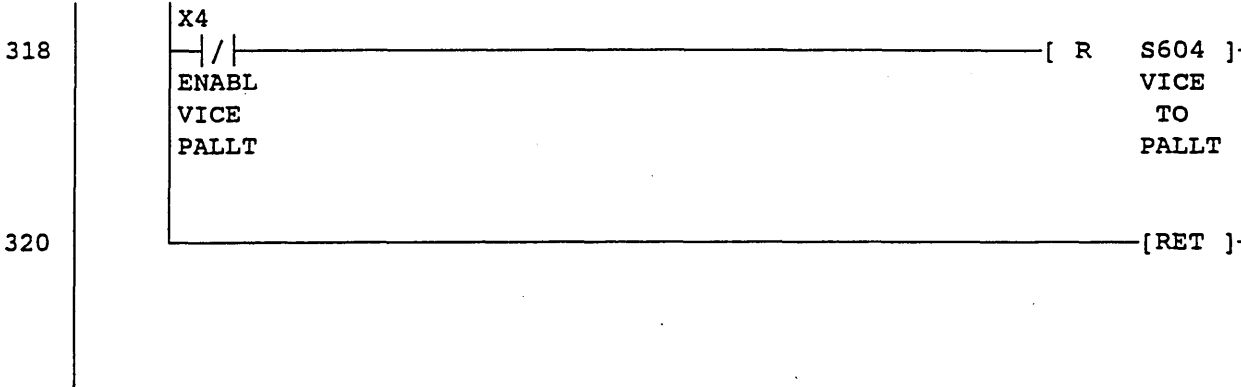
VICE TO PALLET



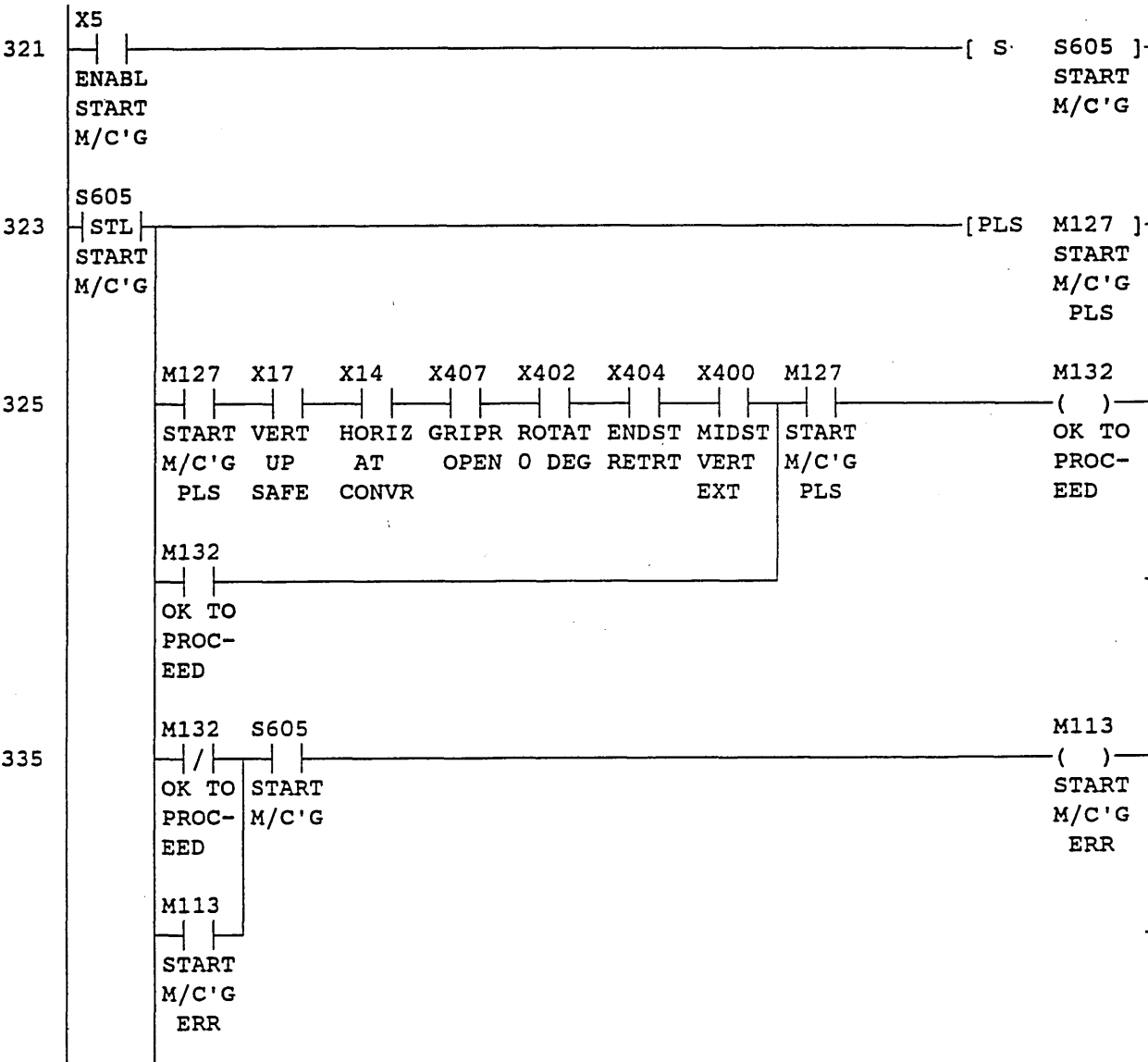
CONTROL OF WORK HANDLING, GANTRY ROBOT AND MACHINING FOR LATHE STATION IN THE SCHOOL OF ENGINEERING'S FMC	SHEFFIELD HALLAM UNIVERSITY	Appendix H	Proj:LATHE
		Date: 18/01/96	Syst:F1/F2
		Rev.no: 2	Type:Ladder
	Draw.no:	Sign: A.T.	Page: 16



CONTROL OF WORK HANDLING, GANTRY ROBOT AND MACHINING FOR LATHE STATION IN THE SCHOOL OF ENGINEERING'S FMC	SHEFFIELD HALLAM UNIVERSITY	Appendix H	Proj:LATHE
		Date: 18/01/96	Syst:F1/F2
		Rev.no: 2	Type:Ladder
	Draw.no:	Sign: A.T.	Page: 17

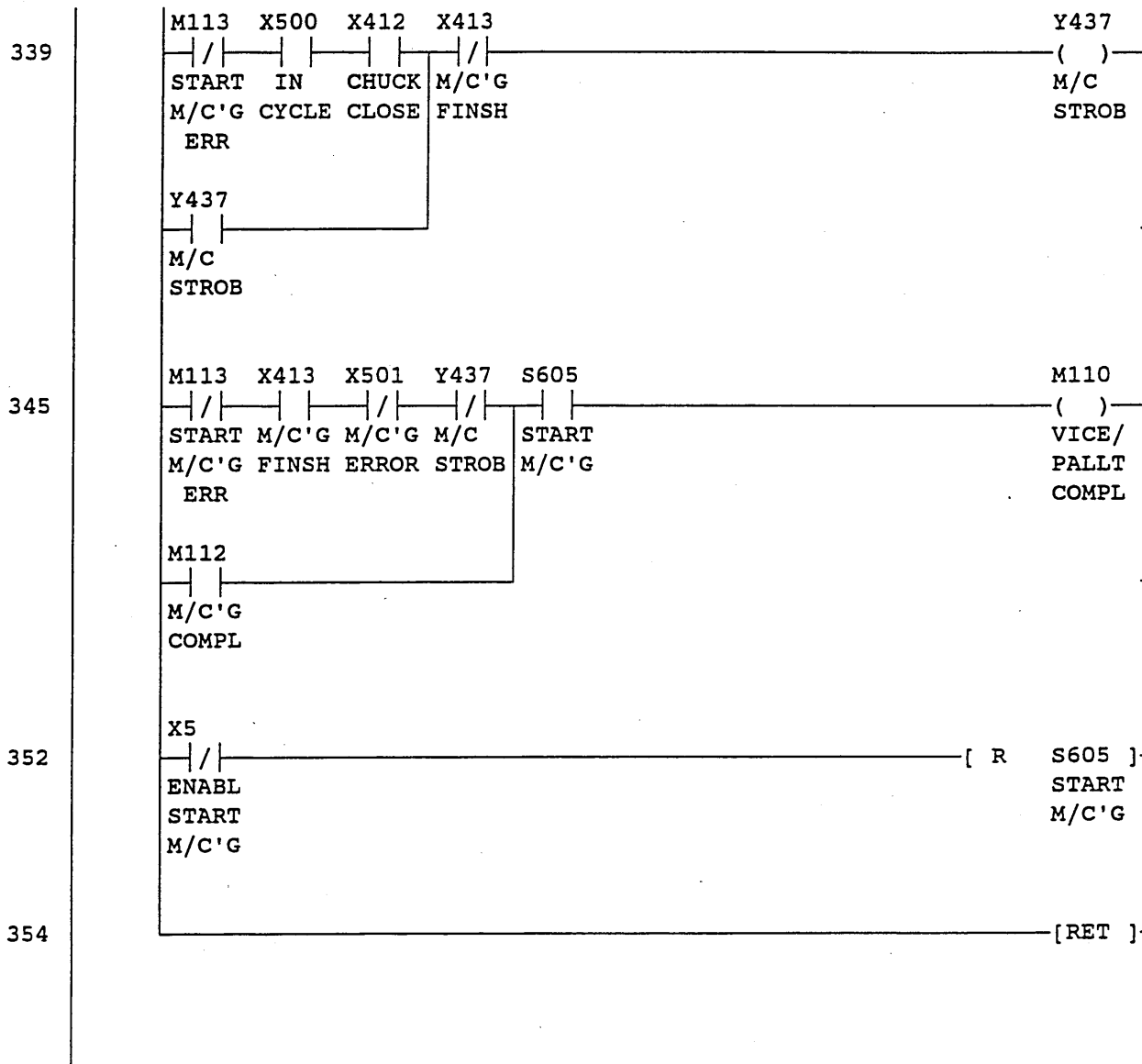


START MACHINING

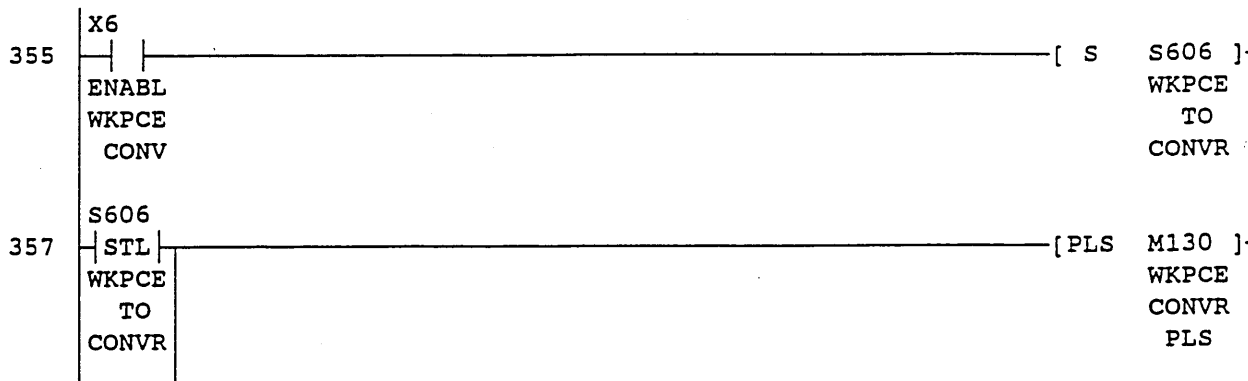




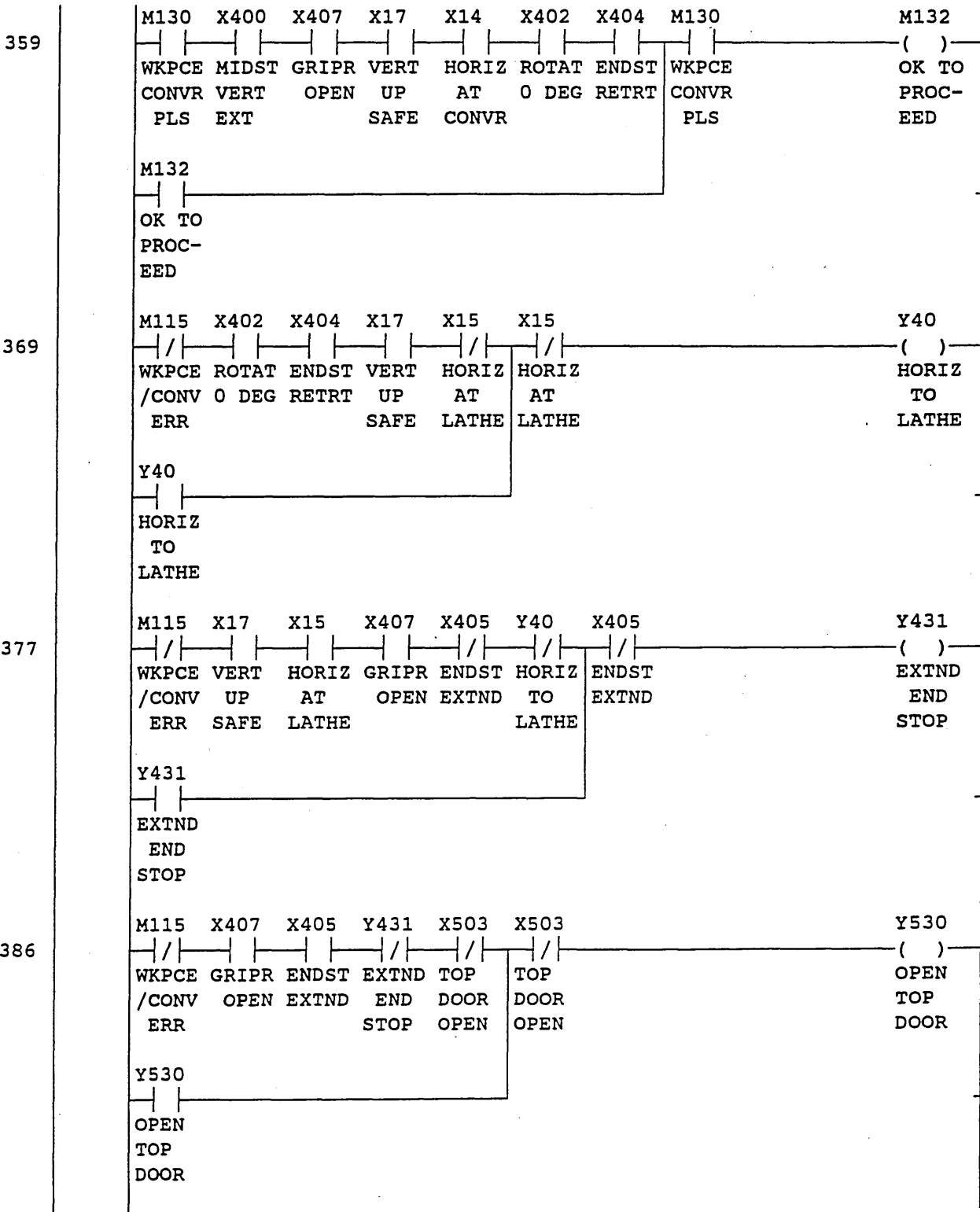
CONTROL OF WORK HANDLING, GANTRY ROBOT AND MACHINING FOR LATHE STATION IN THE SCHOOL OF ENGINEERING'S FMC	SHEFFIELD HALLAM UNIVERSITY	Appendix H	Proj:LATHE
		Date: 18/01/96	Syst:F1/F2
		Rev.no: 2	Type:Ladder
	Draw.no:	Sign: A.T.	Page: 18



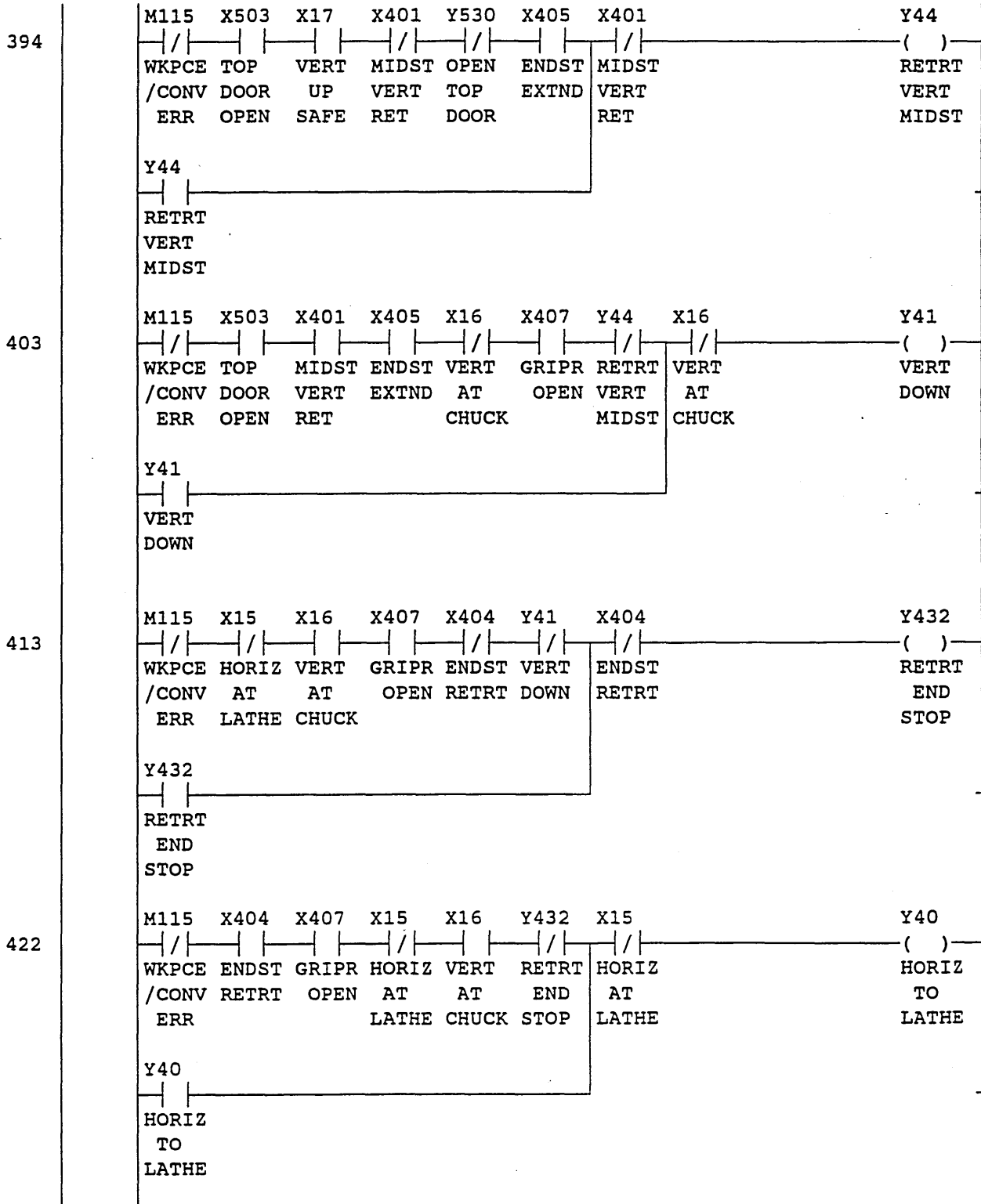
WORKPIECE TO CONVEYOR



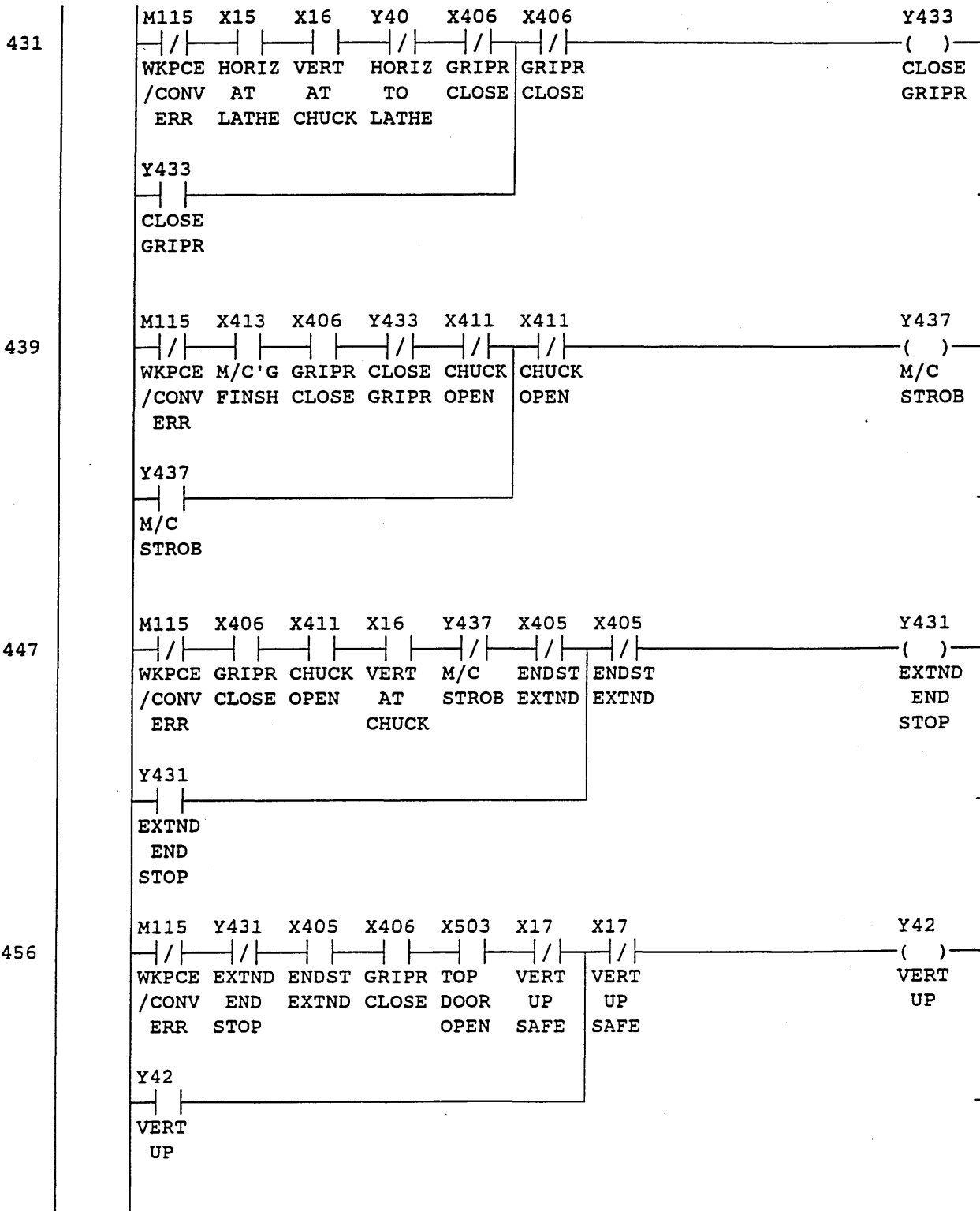
CONTROL OF WORK HANDLING, GANTRY ROBOT AND MACHINING FOR LATHE STATION IN THE SCHOOL OF ENGINEERING'S FMC	SHEFFIELD HALLAM UNIVERSITY	Appendix H	Proj:LATHE
		Date: 18/01/96	Syst:F1/F2
		Rev.no: 2	Type:Ladder
	Draw.no:	Sign: A.T.	Page: 19



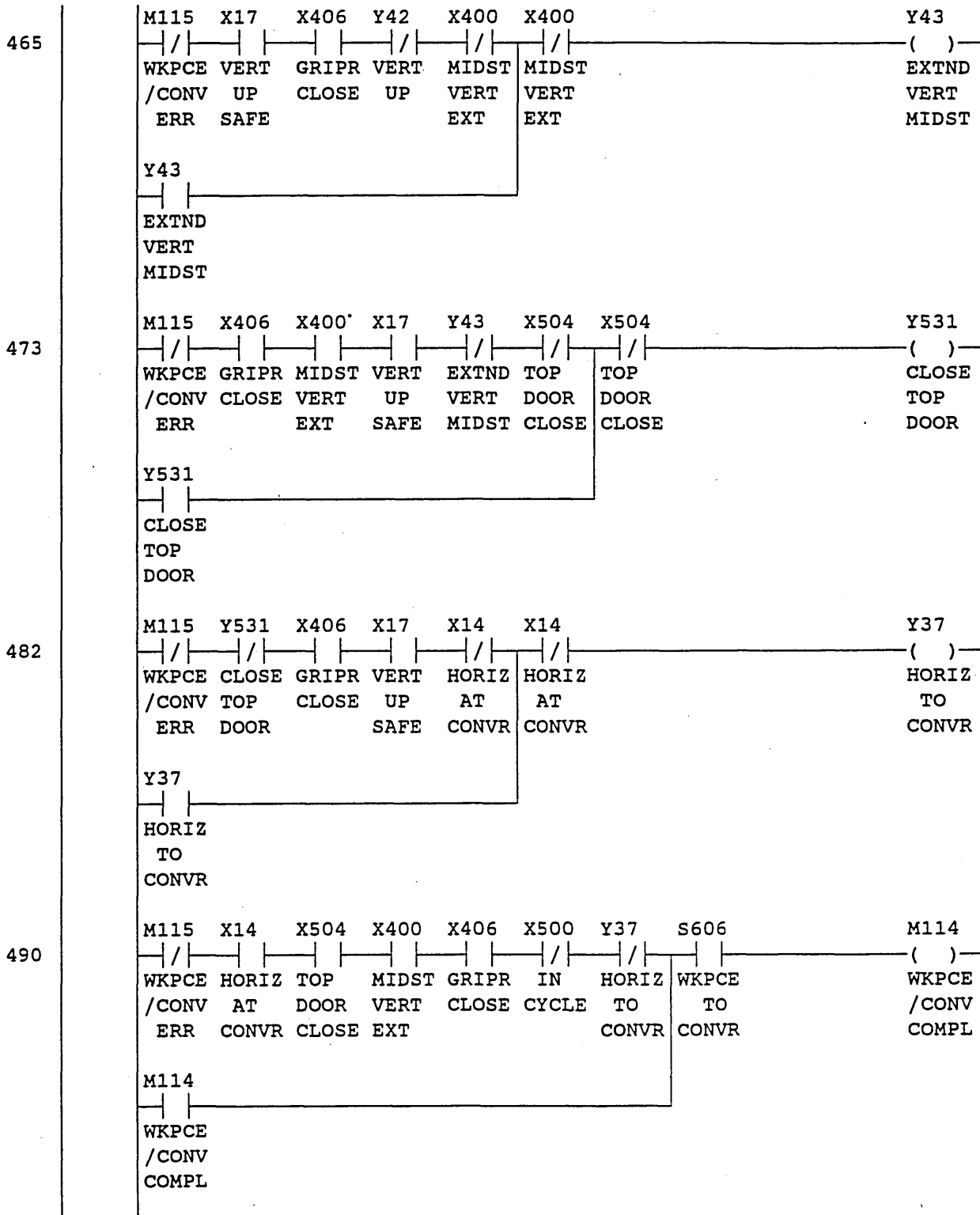
CONTROL OF WORK HANDLING, GANTRY ROBOT AND MACHINING FOR LATHE STATION IN THE SCHOOL OF ENGINEERING'S FMC	SHEFFIELD HALLAM UNIVERSITY	Appendix H	Proj:LATHE
		Date: 18/01/96	Syst:F1/F2
		Rev.no: 2	Type:Ladder
	Draw.no:	Sign: A.T.	Page: 20



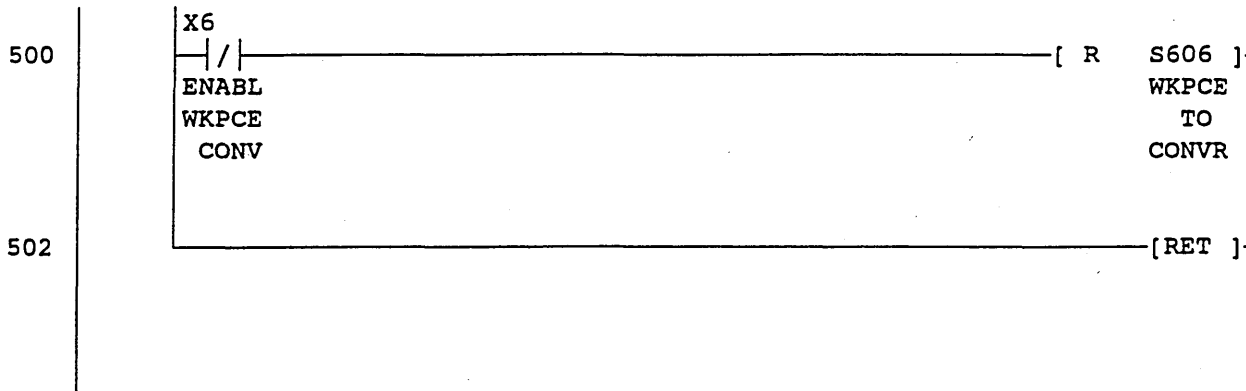
CONTROL OF WORK HANDLING, GANTRY ROBOT AND MACHINING FOR LATHE STATION IN THE SCHOOL OF ENGINEERING'S FMC	SHEFFIELD HALLAM UNIVERSITY	Appendix H	Proj: LATHE
		Date: 18/01/96	Syst: F1/F2
		Rev.no: 2	Type: Ladder
	Draw.no:	Sign: A.T.	Page: 21



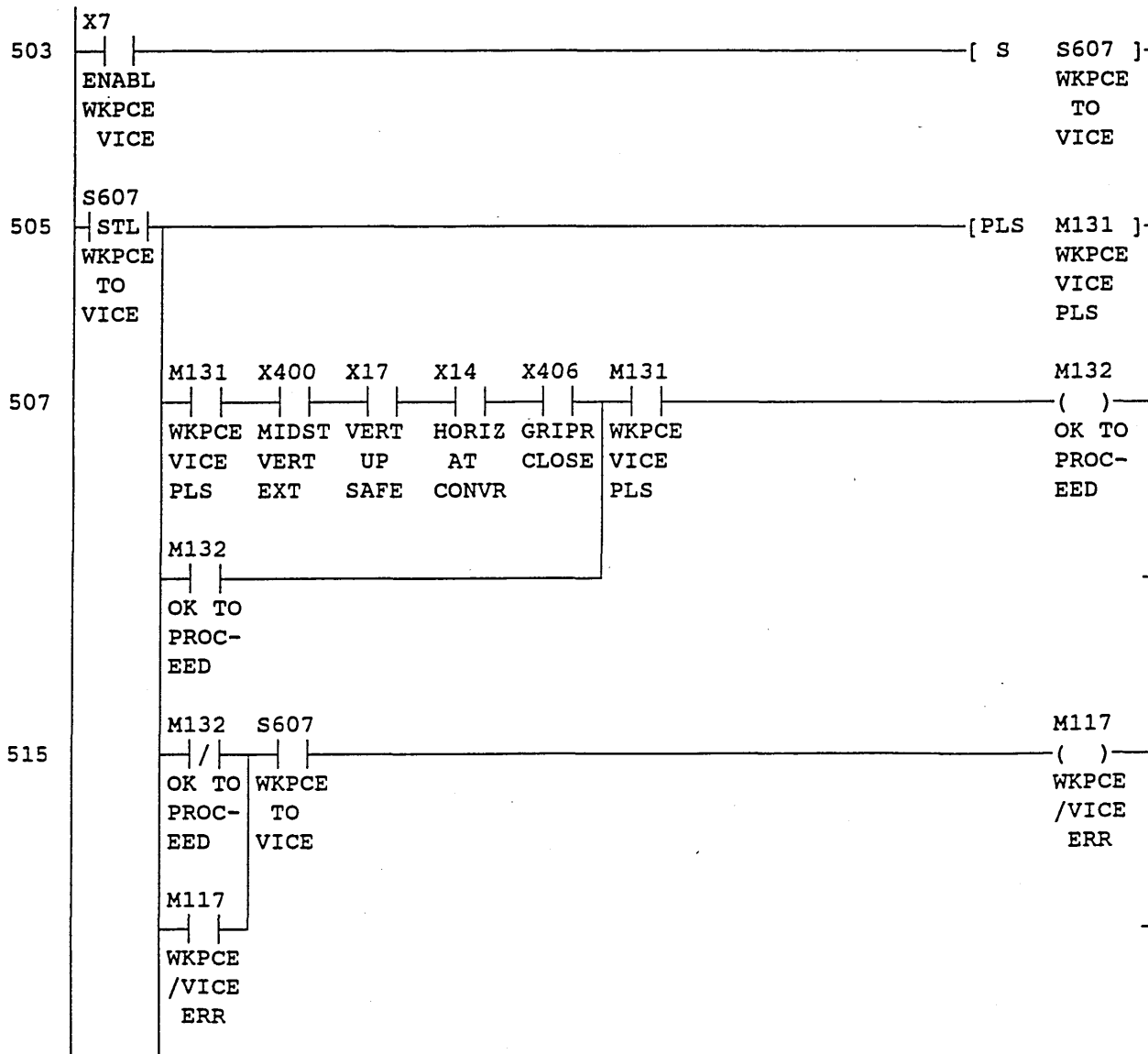
CONTROL OF WORK HANDLING, GANTRY ROBOT AND MACHINING FOR LATHE STATION IN THE SCHOOL OF ENGINEERING'S FMC	SHEFFIELD HALLAM UNIVERSITY	Appendix H	Proj:LATHE
		Date: 18/01/96	Syst:F1/F2
		Rev.no: 2	Type:Ladder
	Draw.no:	Sign: A.T.	Page: 22



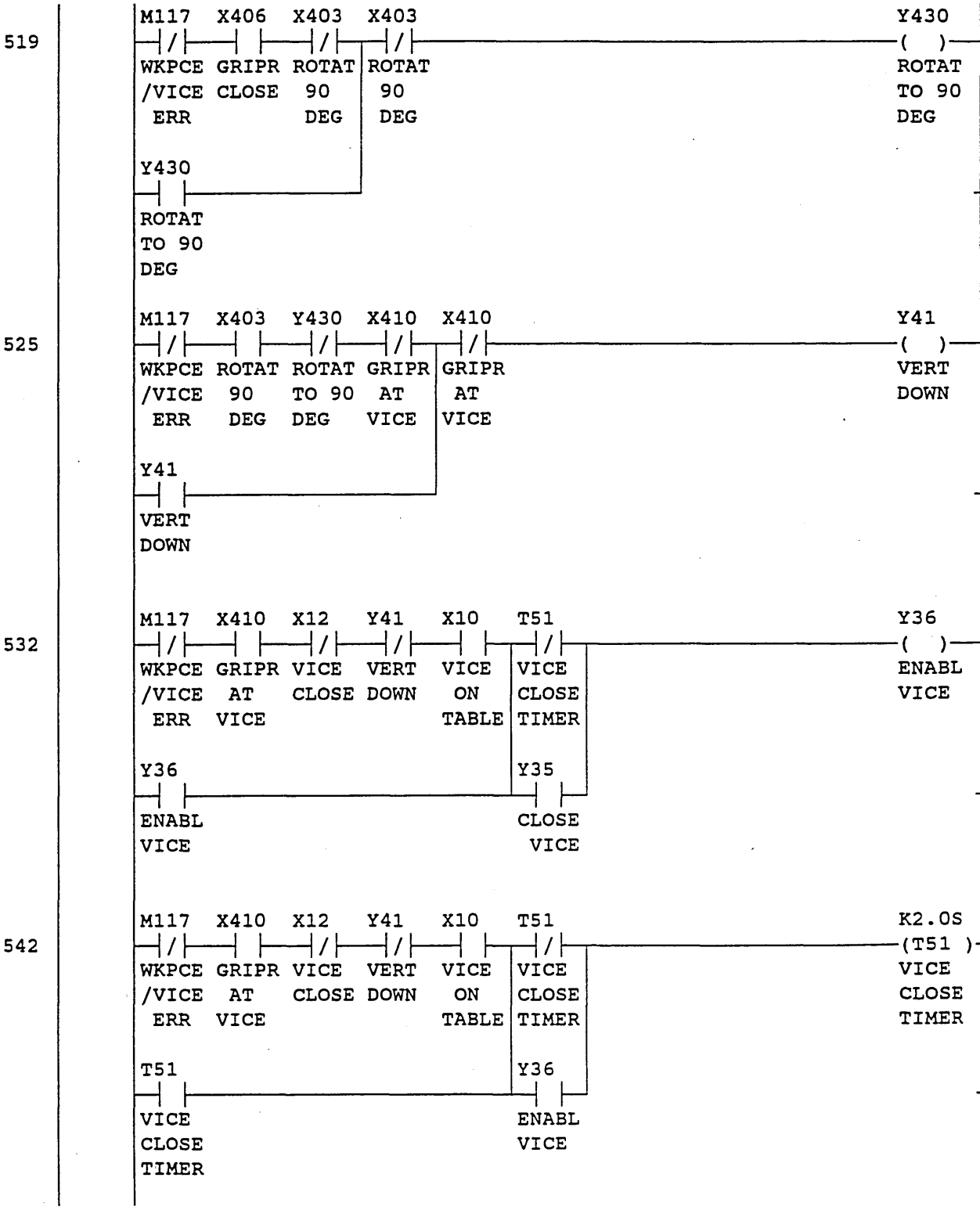
CONTROL OF WORK HANDLING, GANTRY ROBOT AND MACHINING FOR LATHE STATION IN THE SCHOOL OF ENGINEERING'S FMC	SHEFFIELD HALLAM UNIVERSITY	Appendix H	Proj:LATHE
		Date: 18/01/96	Syst:F1/F2
		Rev.no: 2	Type:Ladder
	Draw.no:	Sign: A.T.	Page: 23



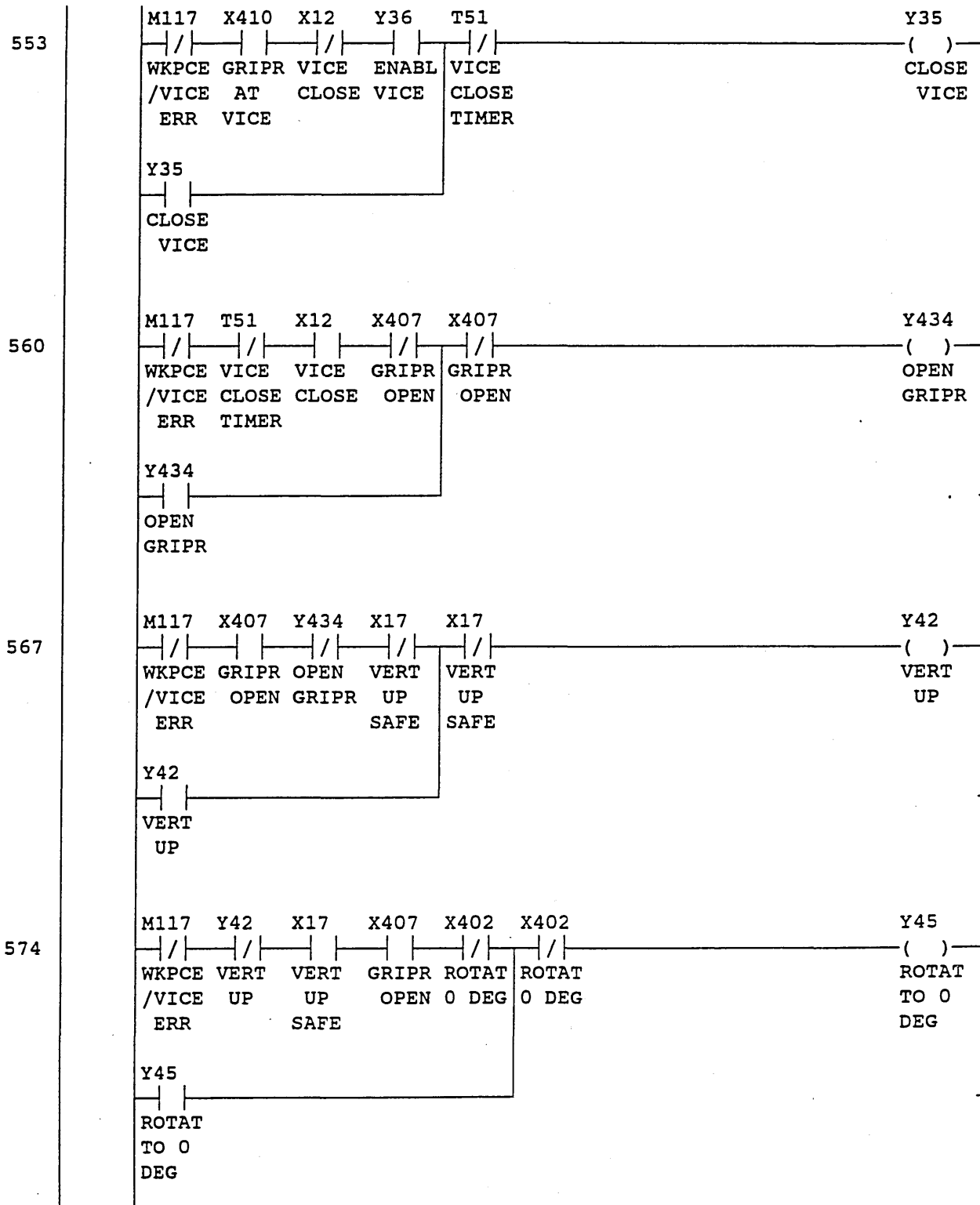
WORKPIECE TO VICE



CONTROL OF WORK HANDLING, GANTRY ROBOT AND MACHINING FOR LATHE STATION IN THE SCHOOL OF ENGINEERING'S FMC	SHEFFIELD HALLAM UNIVERSITY	Appendix H	Proj:LATHE
		Date: 18/01/96	Syst:F1/F2
		Rev.no: 2	Type:Ladder
	Draw.no:	Sign: A.T.	Page: 24



CONTROL OF WORK HANDLING, GANTRY ROBOT AND MACHINING FOR LATHE STATION IN THE SCHOOL OF ENGINEERING'S FMC	SHEFFIELD HALLAM UNIVERSITY	Appendix H	Proj:LATHE
		Date: 18/01/96	Syst:F1/F2
		Rev.no: 2	Type:Ladder
	Draw.no:	Sign: A.T.	Page: 25









CONTROL OF WORK HANDLING, GANTRY ROBOT AND MACHINING FOR LATHE STATION IN THE SCHOOL OF ENGINEERING'S FMC	SHEFFIELD HALLAM UNIVERSITY	Appendix H	Proj:LATHE
		Date: 18/01/96	Syst:F1/F2
		Rev.no: 2	Type:Ladder
	Draw.no:	Sign: A.T.	Page: 28

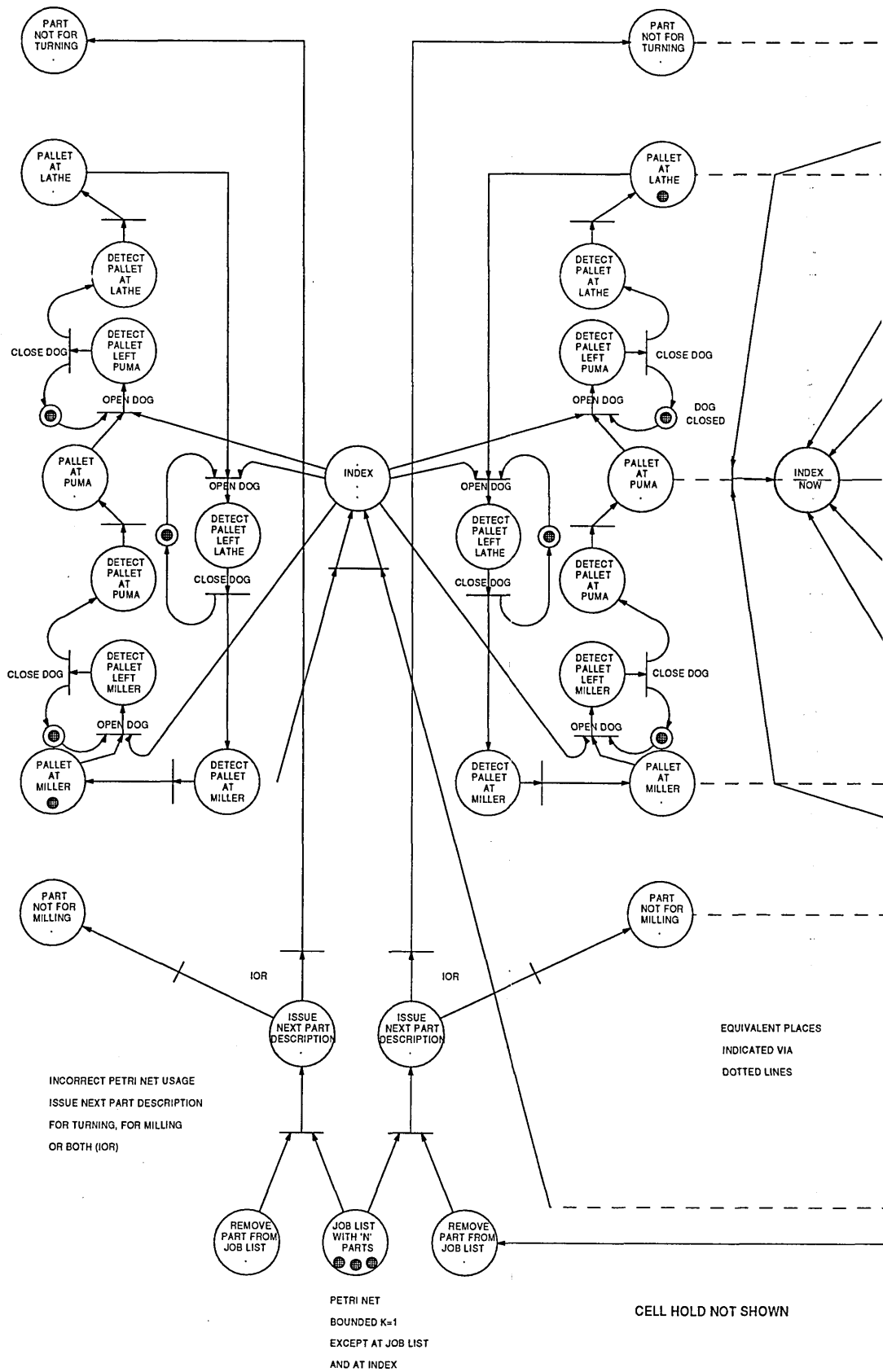
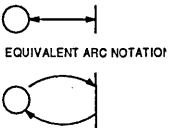
GENERATE COMPLETION STATUS

627	M100	X507	X507	Y30
			/	( )
	VICE/	LATHE	LATHE	LATHE
	TABLE	STATN	STATN	STATN
	COMPL	COMPL	COMPL	COMP
	M102	X507		
			/	
	GRIP	LATHE		
	WKPCE	STATN		
	COMPL	COMPL		
	M104	X507		
			/	
	WKPCE	LATHE		
	/CHUK	STATN		
	COMPL	COMPL		
	M106	X507		
			/	
	ROBOT	LATHE		
	/SAFE	STATN		
	COMPL	COMPL		
	M110	X507		
			/	
	VICE/	LATHE		
	PALLT	STATN		
	COMPL	COMPL		
	M112	X507		
			/	
	M/C'G	LATHE		
	COMPL	STATN		
		COMPL		
	M114	X507		
			/	
	WKPCE	LATHE		
	/CONV	STATN		
	COMPL	COMPL		
	M116	X507		
			/	
	WKPCE	LATHE		
	/VICE	STATN		
	COMPL	COMPL		

CONTROL OF WORK HANDLING, GANTRY ROBOT AND MACHINING FOR LATHE STATION IN THE SCHOOL OF ENGINEERING'S FMC	SHEFFIELD HALLAM UNIVERSITY	Appendix H	Proj:LATHE
		Date: 18/01/96	Syst:F1/F2
		Rev.no: 2	Type:Ladder
	Draw.no:	Sign: A.T.	Page: 29

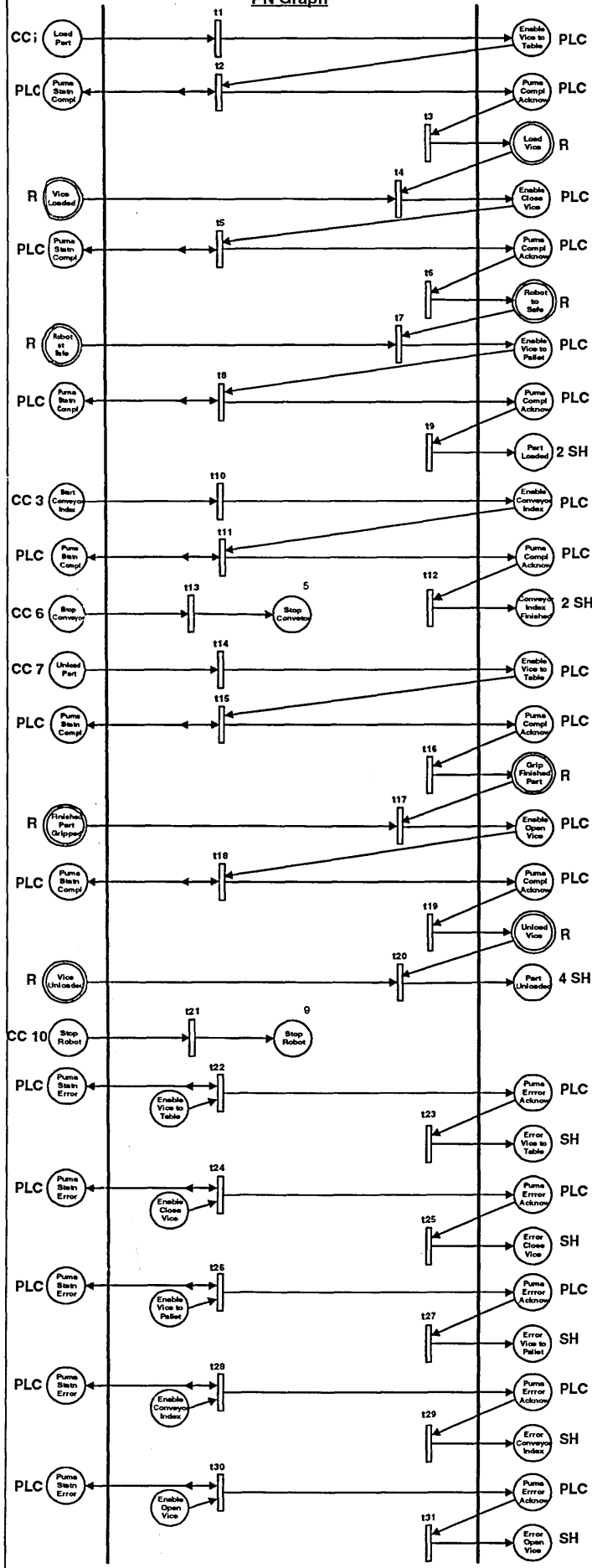
652

[END ]

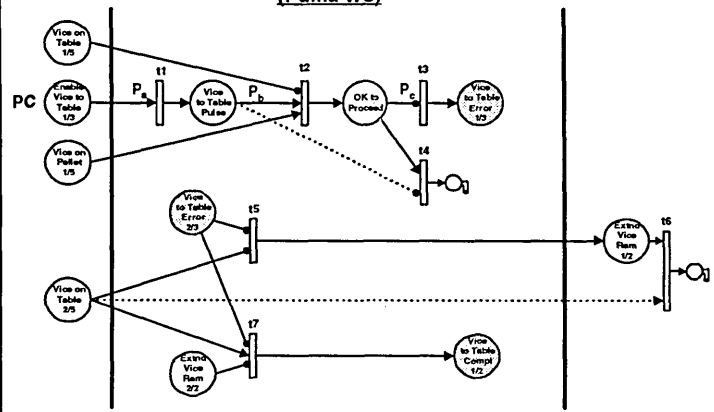




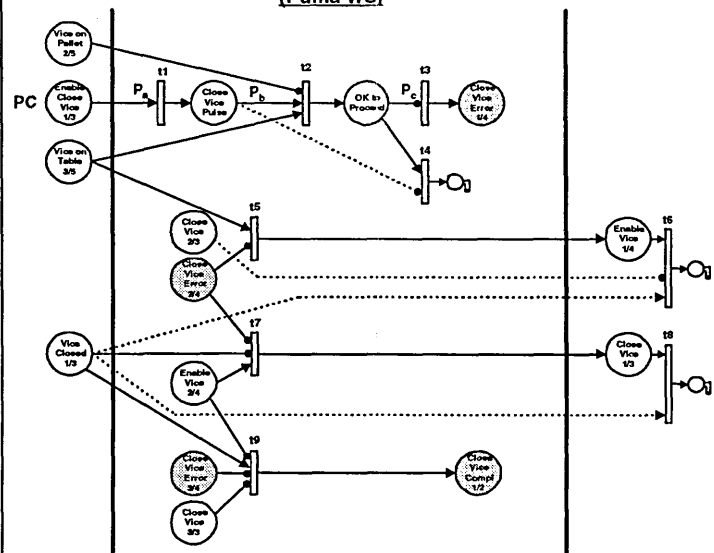
**Updated Puma Controller  
PN Graph**



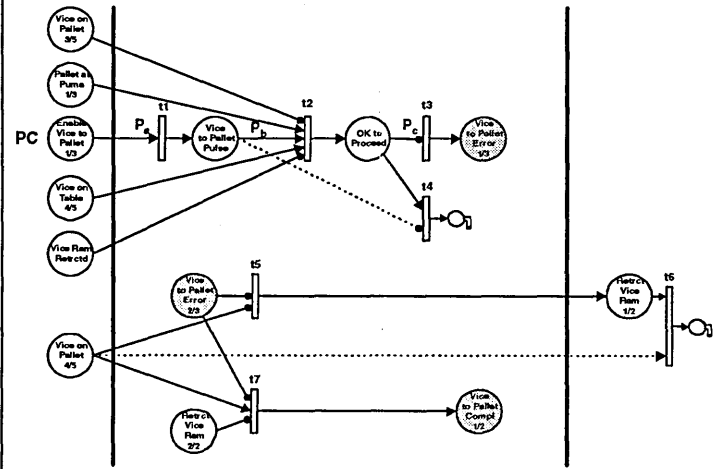
**Enable Vice to Table PN Group  
(Puma WS)**



**Enable Close Vice PN Group  
(Puma WS)**



**Enable Vice to Pallet PN Group  
(Puma WS)**



**The Unification**