國　立　交　通　大　學

工業工程與管理學系

碩士論文

浮標守恆裴氏圖到階梯圖的自動轉換設計與實作:
自動灌模系統案例分析

Design and Implementation of Automated Transformation from Token-Conserved Petri Net to Ladder Diagrams: Automated Mould Filling System Case Study

研究生：陳書皓

指導教授：梁高榮　博士

中華民國一百零二年七月

浮標守恆裴氏圖到階梯圖的自動轉換設計與實作:自動灌模系統案例分析

Design and Implementation of Automated Transformation from Token-Conserved Petri Net to Ladder Diagrams: Automated Mould Filling System Case Study

Student：Shu-Hao Chen

Advisor：Gau-Rong Liang

國立交通大學

工業工程與管理學系

碩士論文

A Thesis Submitted to
Department of Industrial Engineering and Management
College of Management
National Chiao Tung University
in Partial Fulfillment of the Requirements
for the Degree of Master of Engineering in
Industrial Engineering and Management
July 2013
Hsinchu, Taiwan, Republic of China

中華民國一百零二年七月

浮標守恆裴氏圖到階梯圖的自動轉換設計與實作:自動灌模系統案例分析

研究生：陳書皓　　指導教授：梁高榮博士

國立交通大學工業工程與管理學系

摘要

理論上，浮標守恆裴氏圖可用來描述彈性製造系統的製造流程。實務上，階梯圖可用來即時控制製造設備。從浮標守恆裴氏圖產生階梯圖，本論文已發展出一種三階段轉換的方法。第一階段為利用時域分解法將浮標守恆裴氏圖分解成擁有初始浮標的多張歐氏記號圖。第二階段為依據動態生產限制的詮釋而設計同步裴氏圖選擇適當的歐氏記號圖以控制生產設備。在此，浮標守恆裴氏圖與同步裴氏圖合稱為詮釋型裴氏圖。第三階段為將初始浮標、同步裴氏圖與歐氏記號圖分別轉換為初始階梯圖、同步階梯圖與歐氏階梯圖。為了驗證新方法的可行性，本論文以自動灌模系統之浮標守恆裴氏圖來產生階梯圖。

關鍵字：
自動化製造系統
歐氏記號圖
浮標守恆裴氏圖
同步裴氏圖
階梯圖

Design and Implementation of Automated Transformation from
Token-Conserved Petri Net to Ladder Diagrams:
Automated Mould Filling System Case Study

Student: Shu-Hao, Chen Advisor: Dr. Gau-Rong, Liang

Department of Institute of Industrial Engineering & Management
National Chiao Tung University

Abstract

Theoretically Token-Conserved Petri net (TCPN) is used for described the manufacturing process of flexible manufacturing system. Practically ladder diagrams are used for controlling manufacturing devices in a real-time way. In this thesis, a 3-step transformation method has been developed for generating ladder diagrams from a given Token-Conserved Petri net (TCPN). First Eulerian marked graphs with initial tokens are generated from the TCPN using temporal decomposition method. Second a synchronized Petri net is designed for selecting a proper Eulerian marked graph to control manufacturing devices according to the dynamic interpretation of production constraints. For this reason, the TCPN and the synchronized Petri net is named interpreted Petri net. Third the initial tokens, the synchronized Petri net, and Eulerian marked graphs are transformed into initial ladder diagram, synchronized ladder diagram, and Eulerian ladder diagrams respectively. For showing the feasibility of this new approach, ladder diagrams of an automated mold filling system are generated from its TCPN.

Keywords：
Automated Manufacturing System
Eulerian Marked Graph
Token-Conserved Petri net
Synchronized Petri net
Ladder Diagram

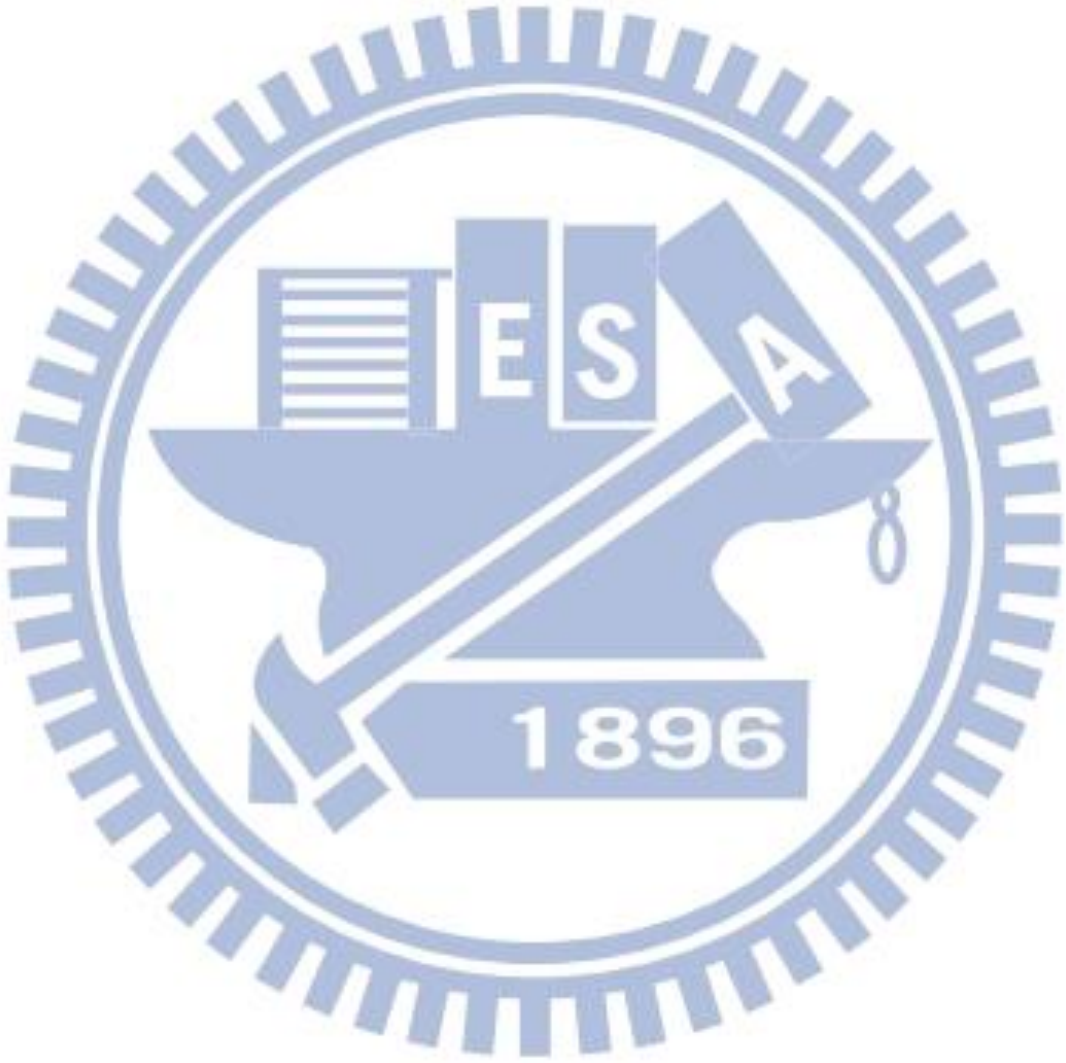
誌謝

本篇論文得以完成，首先要感謝我的指導教授梁高榮博士這一年來細心的教導與叮嚀。此外，特別感謝口試委員陳文智老師和王志軒老師參加口試並提供寶貴建議，使本論文能更加完整。

兩年的研究所生活裡，除了學術上的增長，也認識了許多共患難的好朋友。謝謝我實驗室的好夥伴王敏，在研究所的兩年幫助我很多很多，一起面對問題一起解決困難。帶來很多的歡樂與溫馨。謝謝學長姐本欣、宏智、舜龍、張弘、哲慧，婉琪，教導我很多很多東西，不管是學業上論文上還是生活上面的經驗，有你們才使得我成長很多。謝謝學弟妹們在實驗室帶來的歡樂。

最後，感謝我摯愛的父母與家人，感謝你們支持我完成研究所的學業，更感謝您們多年來的付出與鼓勵。謹以本論文獻給我最愛的家人以及陪我伴我度過這段成長的老師和朋友們。

目錄



圖目錄

表目錄

第一章　緒論

本章主要說明本論文的研究動機、目的與論文的整體架構。以下共分成三個部份：1.1 節『研究動機』，1.2 節『問題界定與研究目的』，1.3 節『論文架構』。

1.1 研究動機

對於自動化製造系統(Automated Manufacturing System)而言，主要可分為重覆性製造系統(Repetitive Manufacturing System)與彈性製造系統(Flexible Manufacturing System)兩大類[6]。前者為單一產品的大量生產，可以使生產力最大化。後者為多項產品的彈性生產，以生產設備共用方式進行適量生產。在分析自動化製造系統的運作行為時，歐氏記號圖(Eulerian Marked Graph)與浮標守恆裴氏圖(Token-Conserved Petri Net)是常用的工具[6]，這兩者比裴氏圖(Petri Net)[43]多了可描述製造系統資訊流與浮標守恆(Token-conserved)兩大特色，因此可更妥善地表達製造系統。

目前在工業界中，可程式邏輯控制器(Programmable Logic Controller, PLC)[23]是作為製造控制系統的控制工具。在 PLC 程式語言中，以階梯圖(Ladder Diagram)[22]應用最為廣泛。由於 PLC 易於維修且方便維護，故 PLC 成為工業界中常見的控制工具。

裴氏圖提供使用者進行全面性的系統規劃。並可進行系統的最佳化[4]。浮標守恆裴氏圖為裴氏圖的特例，當所有轉移點的輸入暫存點與輸出暫存點數相同時，此種裴氏圖稱為浮標守恆裴氏圖，其可用來表達製造設備的狀態，所有的浮標總數是不會改變的[7]。梁高榮[6]於 2011 年提出浮標守恆裴氏圖可用來表達彈性製造系統；梁高榮[7]於同年 10 月提出時域分解法(Temporal Decomposition Method)，將浮標守恆裴氏圖分解成歐氏記號圖，可降低偵查法則的設計複雜度，利用人工智慧語言 Prolog 開發程式，使得分解過程可以自動化。

因為裴氏圖無法直接控制系統，而階梯圖也無法提供系統規劃與數學性質分析。若能先以裴氏圖建立自動化控制系統，再轉換為階梯圖應用在製造業中，將能同時具備控制系統與數學性質分析的優點。故 Shin 與 Meng[45]整理出不同以裴氏圖為基礎，轉換為階梯圖的方法；張舜龍[8]以重覆性製造系統為例，建立法則轉換法將歐氏記號圖轉換為階梯圖。但目前沒有人以彈性製造系統為例，將浮標守恆裴氏圖轉換成階梯圖這議題進行研究。

本篇論文將針對彈性製造系統裡浮標守恆裴氏圖轉換成階梯圖進行研究與自動轉換程式的撰寫。以自動灌模系統為例[12]，將浮標守恆裴氏圖利用時域分解法，分解成推式、拉式與混合式的四張歐氏記號圖，再將此四張歐氏記號圖結合同步裴氏圖(Synchronized Petri Net)轉換為階梯圖並且進行合成。此階梯圖包含了初始階梯圖、同步階梯圖與歐氏階梯圖，利用同步階梯圖可使工廠可依景氣的好壞彈性選擇推式、拉式或混合式的方式製造產品。

在自動轉換程式方面，本論文使用 Java 程式撰寫，利用可加註語言(Extensible Markup Language, XML)[25]的技術，使歐氏記號圖和階梯圖以文件的方式在使用者介面上呈現。再使用 XML 資料繫結技術(Java Architecture XML Binding, JAXB)將文件轉為物件，最後進行物件的自動轉換。此過程稱為三層式架構轉換法(Three-layer Architecture Approach)[11]。因此，本論文將提出將浮標守恆裴氏圖轉換成初始階梯圖、同步階梯圖與歐氏階梯圖的方法，並實作程式使轉換過程得以自動化。

1.2 問題界定

在自動化製造系統中，裴氏圖具有數學建模的特色，階梯圖具有直接控制系統、成本低與普及度高的特色。故通常使用裴氏圖模擬和描述系統，接著使用階梯圖搭配可程式邏輯控制器來實際控制系統。管理人員如果僅單用階梯圖是沒辦法進行數學分析，而單用裴氏圖也無法直接用來控制製造系統。因此許多人開始提出以裴氏圖作為基礎，將裴氏圖轉換為階梯圖。如此便可進行裴氏圖的數學分析，亦可利用階梯圖實際控制製造系統。

前人以重覆性製造系統為例，發展出以法則轉換法將歐氏記號圖轉為階梯圖。但彈性製造系統無法以歐氏記號圖表達，需以浮標守恆裴氏圖來描述之，而浮標守恆裴氏圖能進行拆解成數張歐氏記號圖。故本論文欲以自動灌模系統為例，將辜婉琪設計之自動灌模系統之浮標守恆裴氏圖拆解成數張歐氏記號圖，並轉換成浮標守恆階梯圖，使浮標守恆階梯圖擁有原本浮標守恆裴氏圖的浮標行為以控制系統狀態，彈性的選擇生產方式。圖 1.1 為自動灌模系統之浮標守恆裴氏圖[12]。

在建立轉換浮標守恆階梯圖的方法後，本論文會使用三層式架構轉換法設計從浮標守恆裴氏圖到浮標守恆階梯圖的自動轉換程式。此方法除了產生同步階梯圖流程與方法尚未固定，其餘步驟所需的資訊技術都已經成熟。故本論文會以法則轉換法為基礎，加入本論文提出的同步階梯圖轉換方法，發展新的物件轉換流程。

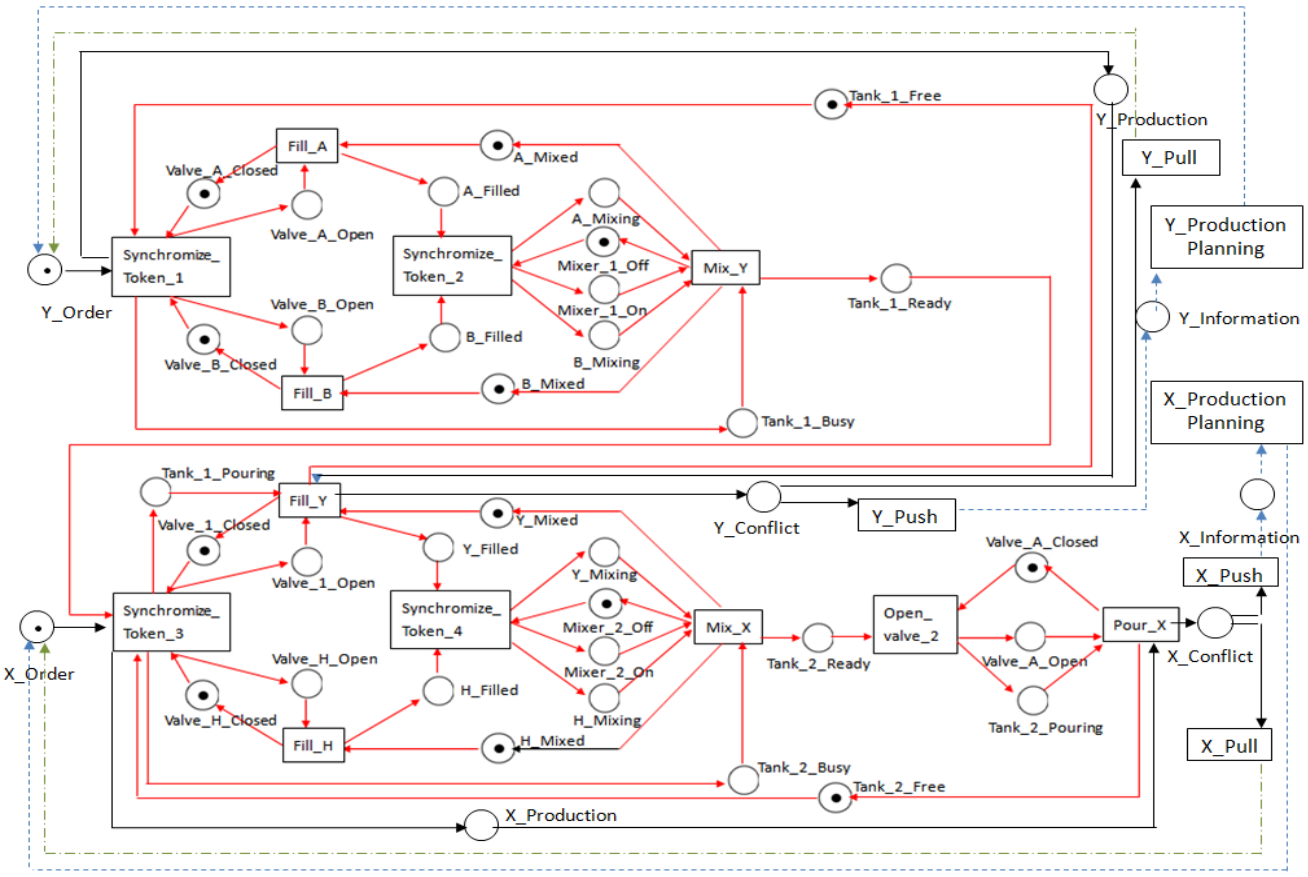


圖 1.1 自動灌模系統之浮標守恆裴氏圖[12]

1.3 研究方法與論文架構

本論文重點主要在浮標守恆裴氏圖到階梯圖轉換法的設計與程式實作。主要研究方法分四個階段。

第一階段為『說明浮標守恆裴氏圖、歐氏記號圖，PLC 與階梯圖』：根據文獻了解浮標守恆裴氏圖、歐氏記號圖的定義與數學特性。並整理階梯圖基本指令，與本論文所用到的迴路指令。

第二階段為『法則轉換法文獻回顧、可加註語言與三層式架構應用』：根據前人所提出的法則轉換法，分析比較其方法得以應用於本論文的轉換法。另外介紹可加註語言和可加註文件資料繫結技術，以做到自動轉換。

第三階段為『轉換架構設計』：先透過研究浮標守恆裴氏圖的行為，再結合前人的轉換法優點，建立浮標守恆裴氏圖轉換到階梯圖的方法。

第四階段為『實作轉換法程式』：將轉換法透過 XML 技術與 JAVA 程式語言實作自動轉換程式，使得浮標守恆裴氏圖到階梯圖的過程能自動化。

因此透過上述的研究方法，本論文分為六章，其大綱如下。

第一章：『緒論』主要說明本論文的研究動機、界定研究範圍、與論文整體架構。

第二章：『文獻回顧』整理本論文所需要的相關技術和方法，並對基礎觀念做整理。

第三章：『浮標守恆裴氏圖到階梯圖的轉換設計』提出浮標守恆裴氏圖到階梯圖轉換流程與設計方法。

第四章：『浮標守恆裴氏圖到階梯圖的轉換實作』本章利用第三章所提出的方法，加上 XML 技術與 XML 文件資料繫結技術 JAXB，搭配 JAVA 程式建立浮標守恆裴氏圖到階梯圖的自動轉換程式。

第五章：『自動轉換程式的操作說明與案例操作』本章利用自動轉換程式，將自動灌系統之浮標守恆裴氏圖進行自動轉換。並驗證其結果。

第六章：『結論』整理本論文之貢獻與成果。

第二章　文獻回顧

本章主要回顧裴氏圖、階梯圖、可加註語言技術和前人所提出的法則轉換的方法。本章分四節，2.1 節介紹裴氏圖，2.2 節介紹階梯圖，2.3 節介紹法則轉換法，2.4 節介紹三層式架構轉換法。

2.1 裴氏圖、歐氏記號圖與浮標守恆裴氏圖

本節介紹裴氏圖的基本元件和法則矩陣，以及不同的裴氏圖種類。2.1.1 節介紹裴氏圖基本元件與性質,2.1.2 節介紹裴氏圖的法則矩陣。2.1.3 到 2.1.5 節介紹不同類型的裴氏圖，2.1.3 節介紹歐氏記號圖，2.1.4 節介紹浮標守恆裴氏圖，2.1.5 節介紹詮釋型裴氏圖。2.1.6 節介紹時域分解法。

2.1.1 裴氏圖的基本元件與性質

裴氏圖於 1962 年由西德數學家 C. A. Petri 提出，其為圖形化的表達工具且具有嚴格數學計算架構的技術。由於裴氏圖具有可描述製造系統中邏輯運作的特性，包含了事件的同步性與事件的因果性，以及可對系統作定性與定量的分析。故廣泛應用在製造系統的建模與分析。

表 2.1 為裴氏圖的元件圖形、元件名稱與意義。裴氏圖是由四種元件組成，分別為暫存點 (Place)、浮標(Token)、轉移點 (Transition)與連接轉移點和暫存點的方向弧 (Arc)。暫存點用來表達系統狀態；浮標表達系統目前狀態；轉移點表達事件發生以轉移狀態；方向弧則是連接暫存點與轉移點，表達事件發生後，狀態轉移的順序。轉移點或暫存點間可以有多個方向弧連接。若有方向弧是由轉移點指向暫存點，則對於此暫存點而言，稱為輸入轉移點(Input transition)；反之，若有方向弧是由暫存點指向轉移點，則對於此暫存點而言，稱此轉移點為輸出轉移點(Output transition)。

表 2.1 裴氏圖基本元件圖形

圖形	名稱	表示意義
◯	暫存點(Place)	系統各種狀態
⊙	浮標(Token)	系統目前狀態
▮	轉移點(Transition)	觸發系統狀態的改變
→	方向弧(Arc)	狀態改變的方向

裴氏圖的性質可依初始狀態(Initail Marking)分為兩大類：行為性質(Behavioral Propertise)與結構性質(Structural Properties)[43]。行為性質與裴氏圖之初始狀態有關，如可達性(Reachability)、活性(Liveness)等性質之探討。結構性質是與初始狀態無關的性質，如安全性(Safeness)、限制性(Boundedness)、浮標不滅性(Conservativeness)、可逆性(Reversibility)與一致性(Consistent)等。

裴氏圖是由轉移點的觸發(Firing)[43]讓浮標移動。浮標的移動表示狀態的轉移[7]。轉移點主要分為可觸發的(Enabled)與觸發後(Fired)。當轉移點的輸入暫存點都包含有足夠的浮標時，稱此轉移點為可觸發狀態。在可觸發狀態下的轉移點可以進行觸發的動作。當轉移點觸發完成後，會使得轉移點的輸入暫存點浮標數減少，輸出暫存點浮標數增加，其中浮標數增加與減少的數量是由箭號的權重 W 所決定，轉移點的激發規則如圖 2.1 所示。

初始浮標(InitailMarking)為裴氏圖浮標的初始位置，表示系統的初始狀態，也就是裴氏圖運行前浮標所在的暫存點。初始浮標可表達為向量，如圖 2.2 中 P1 和 P2 行的值為 1，P3 行的值為 0，表示 P1 和 P2 有初始浮標，P3 沒有初始浮標。

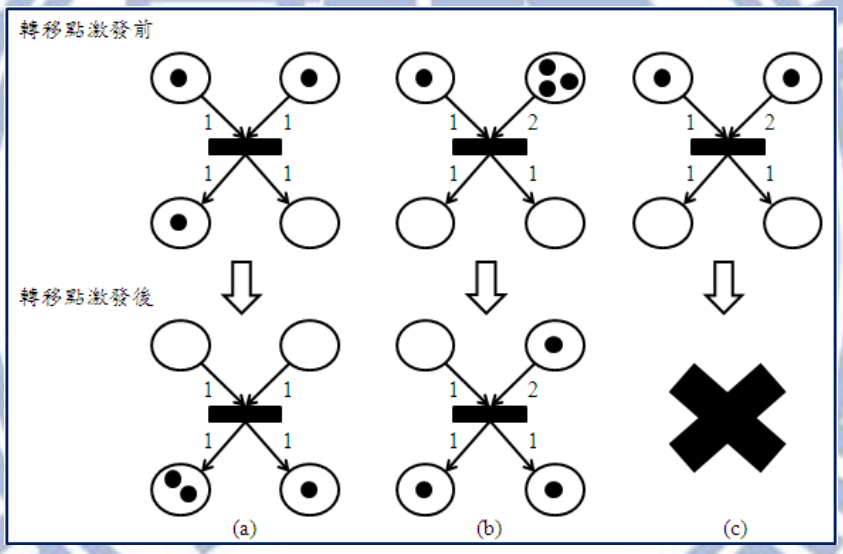


圖 2.1 轉移點激發規則

P1	P2	P3
[1	1	0]

圖 2.2 初始浮標向量的例子

2.1.2 裴氏圖的數學特性與法則矩陣

裴氏圖除了能夠描述系統之動態行為外，亦具有許多良好的數學性質供管理人員使用。裴氏圖能用代數方程式來表示，此代數方程式稱為狀態方程式(State Equation)[1]，其主要功能是以矩陣代數型式表達系統之動態行為。狀態方程式定義為$\mathbf{P(n) = P(0) + R^T T}$，其中 P 代表暫存點狀態，P(n)表示觸發後不同轉移點後的狀態，P(0)表示初始狀態，R 表示法則矩陣(Rule Matrix)[1]，T 表示轉移點的觸發。

法則矩陣的列代表轉移點，行代表暫存點。矩陣中的數字代表轉移點觸發後，所對應的暫存點浮標數的增加或減少。建立法則矩陣的方法主要是來自轉移點的輸入暫存點和輸出暫存點，若是輸入暫存點則記為-1，若是輸出暫存點則記為+1，若是沒有方向弧連接，則記為 0。如圖 2.3 為一個輸送帶控制系統裴氏圖，圖 2.4 為其相對應的法則矩陣。可由圖 2.3 知道轉移點 T1 的輸入暫存點為 P3，因此在圖 2.4 中 P3 和 T1 相交處記為-1。轉移點 T1 的輸出暫存點為 P1，因此在圖 2.4 中 P1 和 T1 相交處記為 1，最後因為其他沒有方向弧連接，故為 0。建立法則矩陣後，便可利用狀態方程式表達系統的行為，其結果如圖 2.5。

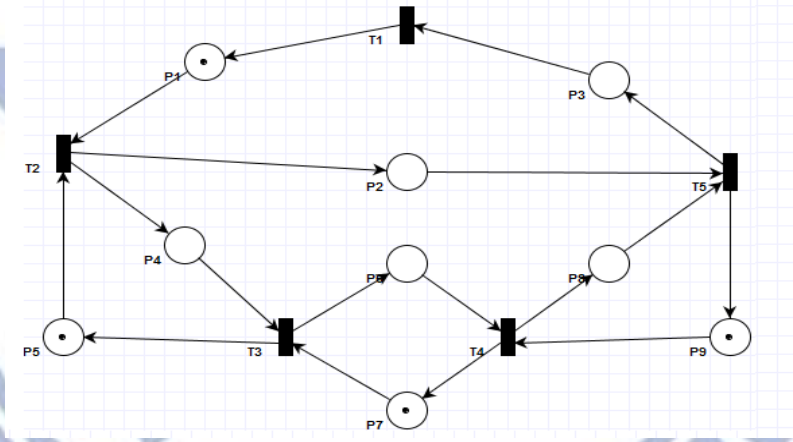


圖 2.3 輸送帶控制系統裴氏圖

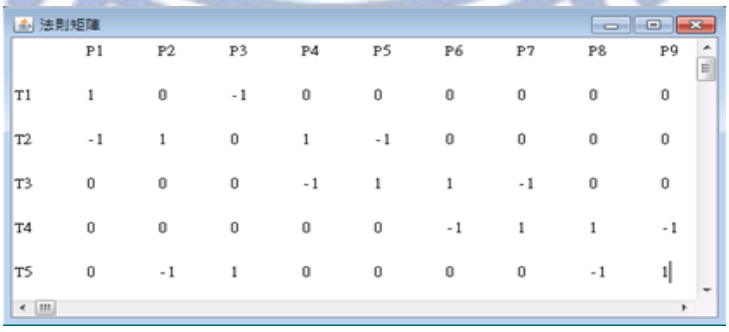


圖 2.4 法則矩陣範例

$$\begin{bmatrix}0\\1\\0\\1\\0\\0\\1\\0\\1\end{bmatrix} = \begin{bmatrix}1\\0\\0\\0\\1\\0\\1\\0\\1\end{bmatrix} + \begin{bmatrix}1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0\\ -1 & 1 & 0 & 1 & -1 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & -1 & 1 & 1 & -1 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & -1 & 1 & 1 & -1\\ 0 & -1 & 1 & 0 & 0 & 0 & 0 & -1 & 1\end{bmatrix} \cdot \begin{bmatrix}0\\1\\0\\0\\0\end{bmatrix}$$

圖 2.5 狀態方程式範例

2.1.3 歐氏記號圖

歐氏記號圖[5]是特殊的裴氏圖，數學表達為$\forall t \in T: |t\cdot| = |\cdot t|$，主要有兩個重要性質。第一個性質為每個暫存點的輸入轉移點和輸出轉移點皆只有一個。第二個性質為轉移點的輸入暫存點與輸出暫存點的個數相同。因輸入暫存點與輸出暫存點的個數相同，法則矩陣每列的-1、1 之個數會相等；每行亦僅會有一個-1 和 1 的元素。故法則矩陣每行與每列相加總合為 0。歐氏記號圖可以使擬陣理論[44]之應用更加容易。因此若管理人員使用歐氏記號圖，會更容易進行系統監控。

裴氏圖來描述製造流程是以資訊流進行控制，而歐氏記號圖中浮標的移動可表示資訊的流動，如設備的開啟與關閉。浮標的移動可視為物件屬性(Attribute)在不同時間的呈現[7]。歐氏記號圖中暫存點一進一出的性質表示物件在一個時間點不會有多種屬性，如設備不會同時開啟且關閉，這樣的性質使得歐氏記號圖有效地描述重覆性製造系統的行為。

2.1.4 浮標守恆裴氏圖

梁高榮[7]於 2011 年提出浮標守恆裴氏圖來描述彈性製造系統。近年來，歐氏記號圖與浮標守恆裴氏圖的技術漸漸用來建構製造系統之模式，可用於資訊流的觀念上。對資訊流來說，如果使用物件的屬性來描述時，則浮標在暫存點上的移動可以看成是物件屬性在不同時間的呈現，故所有的浮標總數不會改變。這裡把浮標總數不會改變的裴氏圖稱為浮標守恆裴氏圖[7]。浮標守恆裴氏圖擁有歐氏記號圖其中一個性質，其性質為轉移點的輸入暫存點與輸出暫存點的個數相同，如圖 2.6 所示。

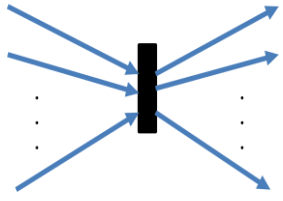


圖 2.6 浮標守恆裴氏圖的轉移點特性

2.1.5 詮釋型裴氏圖

詮釋性裴氏圖(Interpreted Petri Net)[17]是 1992 年由 René David 與 Hassane Alla 所提出。詮釋性裴氏圖為一般裴氏圖的延伸，主要是用在描述硬體、軟體與可程式邏輯控制器等實體系統的輸出、輸入信號互動模式，以輔助使用者進行系統控制邏輯的設計與分析。在自動化製造系統的應用上，詮釋性裴氏圖可以用在表達 Grafcet。Grafcet[17]為可程式邏輯控制器的圖形式程式語言之一，其表達方式與設計原理源自於裴氏圖，但本身卻不具有如裴氏圖一般的分析技術，所以可先由詮釋性裴氏圖進行系統控制邏輯的設計與驗證，再轉換為 Grafcet 語言載入可程式邏輯控制器，此種作法能夠有助於提升控制器的執行效率。

一個詮釋性裴氏圖的定義包含下列三項性質：同步性(Synchronized)、暫存點時間性(P-timed)及包含資料處理元件。對於同步性而言，是指裴氏圖轉移點激發與外部事件的發生存在關聯。一個同步性裴氏圖只有在轉移點為可激發狀態並且當所對應的外部事件發生時，轉移點才會進入激發狀態。

對於暫存點時間性而言，是指暫存點具有計時的特性。當浮標進入暫存點後即重新啟動計時，計時尚未結束以前進入暫存點的浮標為無法利用(Unavailable)的狀態，計時結束後浮標轉換為可利用(Available)的狀態。

資料處理元件是裴氏圖與實體環境之間的溝通介面。在詮釋性裴氏圖裡，每一個暫存點對應到一項作業行為 O_i；每個轉移點對應到一組激發條件 C_j。資料處理元件會根據目前裴氏圖中暫存點的浮標狀態來調整實體系統的整體運作行為 V_{k-}。此外，資料處理元件也會接收實體環境中所發生的事件來判斷裴氏圖轉移點激發條件滿足與否。當有轉移點的激發條件滿足時，會促使該轉移點進行激發並造成浮標狀態產生變化。圖 2.7(a)顯示一個詮釋性裴氏圖與外部環境的互動關係；圖 2.7(b)說明在詮釋性裴氏圖定義裡，將計時時間 d_i, 及作業行為 O_i 對應至暫存點；外部事件 E_j 及狀態轉移條件 C_j 對應至轉移點。

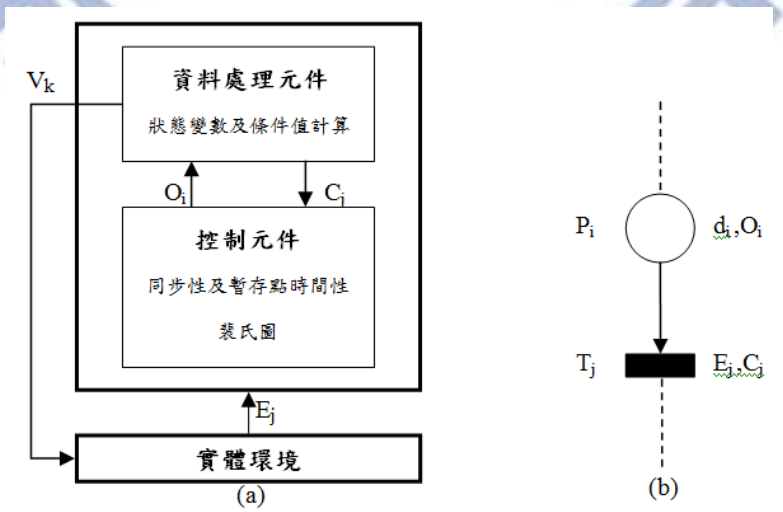


圖 2.7 詮釋型裴氏圖的例子

2.1.6 浮標守恆裴氏圖、同步裴氏圖與詮釋型裴氏圖關係

由於詮釋型裴氏圖是將裴氏圖增加實體環境的控制元件，使得可依外部環境情況選擇需觸發的轉移點。梁高榮[7]設計出控制環境的同步裴氏圖(Synchronized Petri Net)，將同步裴氏圖與浮標守恆裴氏圖進行合成，變成詮釋型裴氏圖。如圖 2.8。

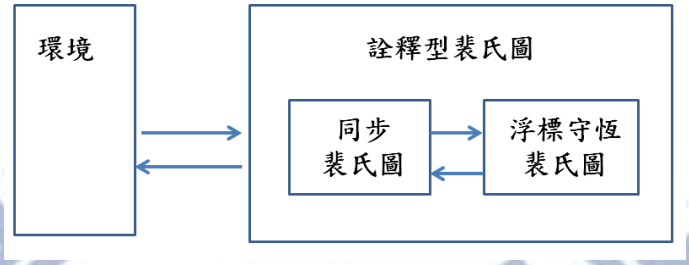


圖 2.8 詮釋型裴氏圖架構

詮釋型裴氏圖的設計主要分三個步驟[12]。第一個步驟是輸入浮標守恆裴氏圖，找出選項暫存點，計算可能暫存點的數目並加入控制暫存點。第二個步驟是設計同步裴氏圖，設計環境要素加至選項暫存點。第三個步驟則是合併完成詮釋型裴氏圖。其流程如圖 2.9。

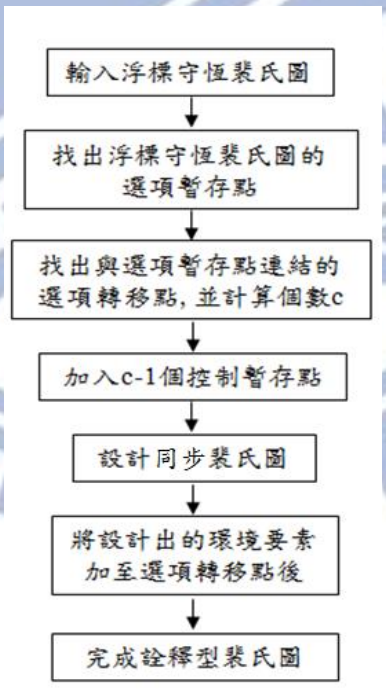


圖 2.9 詮釋型裴氏圖的設計流程圖

2.1.7 時域分解法

梁高榮[7]於 2011 年提出時域分解法(Temporal Decomposition)，其主要是將彈性製造系統之浮標守恆裴氏圖分解成歐氏記號圖。歐氏記號圖可透過擬陣理論來推導出偵查法則(Monitor Rule)[44]，但直接從浮標守恆裴氏圖設計出偵查法則是很困難的。故將其分解成數張歐氏記號圖，再將偵查法則合成，便可降低偵查法則之設計複雜度。

時域分解法可以很容易在時域裡將浮標守恆裴氏圖分解出歐氏記號圖，其原理是建立在浮標守恆裴氏圖的浮標數的守恆與降低轉移點兩大特色上。其主要分成三個步驟，第一步驟為計算可用轉移點集合，也就是找出選項暫存點與選項轉移點，利用選項暫存點互斥的特性降低轉移點，找出可用的轉移點集合與可分解的歐氏記號圖上限。第二步驟為計算可觸發轉移點集合，也就是將每一組可用轉移點集合刪除不可通行之轉移點。第三步驟為產生歐氏記號圖，利用法則矩陣刪除非可觸發轉移點集合之轉移點與整行為 0 之暫存點，便可得到歐氏記號圖。其步驟如圖 2.10。

對計算可用轉移點集合來說，這是利用選項暫存點的互斥特性來產生多組可用轉移點集合。可用轉移點集合的數目代表分解出來歐氏記號圖的張數上限。對計算可觸發轉移點集合來說，在可用轉移點集合裡用加法方式來增加轉移點數目，進而建構出可觸發轉移點集合。最後將可觸發轉移點集合與法則矩陣整合，產生多張歐氏記號圖。

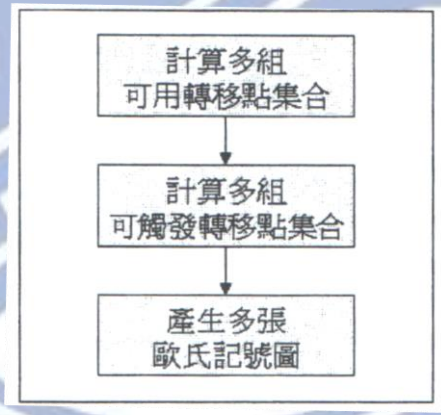


圖 2.10 時域分解法步驟

2.2 階梯圖、階梯圖指令與階梯圖和法則的關係

本節先介紹可程式邏輯控制器與階梯圖的基本元件。接著說明本論文中會使用到的階梯圖指令。然後說明裴氏圖和階梯圖如何以法則來表達。最後介紹本論文使用的三菱廠牌PLC。2.2.1 節說明可程式邏輯控制器與階梯圖基本元件，2.2.2 節說明階梯圖的指令，2.2.3 節說明裴氏圖、階梯圖與法則之間的關係，2.2.4 節介紹三菱廠牌的 PLC。

2.2.1 可程式邏輯控制器與階梯圖基本元件

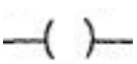
可程式邏輯控制器是工業界中用來自動化控制的數位邏輯控制器，其程式設計語言以階梯圖為主。PLC 主要將外部的輸入裝置如按鍵、感應器、開關及脈波等的狀態讀取後，依這些信號的狀態或數值寫入記憶體中，以微處理機執行邏輯、順序、計數及算式運算，產生所定應的輸出信號到輸出裝置。其運作方式可分為三個步驟，第一個步驟是檢查輸入端元件所連接之設備開關，如感應器狀態，並將其狀態寫入記憶體中。第二個步驟再根據內部儲存的程式，通常為階梯圖，透過邏輯運算出可運行的結果，再將此結果寫入記憶體中。第三個步驟將此結果輸出信號到外部設備以控制系統。

PLC 的邏輯運算是透過階梯圖[22]程式語言。階梯圖語言的基本元件包含左電軌(Left Power Rail)、右電軌(Right Power Rail)、接點(Contact)、和線圈(Coil)。表 2.2 為階梯圖的基本元件圖型。

左電軌與右電軌分別代表系統電流的輸入和輸出。接點是接收輸入端設備信號的元件，當外部輸入信號時，會進行開啟和關閉兩種狀態。輸入接點依照邏輯組合的不同來控制輸出線圈的動作。其編碼方式通常是以 X 加上編號的方式。常開接點為外部輸入為開啟時，此接點會為開啟的狀態。而常開接點的外部輸入為關閉時，此接點會為關閉的狀態。

線圈是輸出信號到外部設備的元件，代表系統目前的動作也就是所處狀態。其編碼方式是 Y 加上編號的方式。

表 2.2 階梯圖基本元件與圖件

元件圖形	名稱	功能說明
	左、右電源軌 (Left/Right Power Rail)	系統電流的輸入端和輸出端。
	常開接點 (Normal Open Contact)	為 PLC 與外部輸入點對應的內部記憶體儲存裝置，透過外部輸入信號進行開啟和關閉兩種狀態。
	常閉接點 (Normal Close Contact)	
	輸出線圈(coil)	為 PLC 與外部輸出點對應的內部記憶體儲存裝置，透過信號輸出控制系統。

在階梯圖邏輯[36]方面可分為組合邏輯(Combinational Logic)和順序邏輯(Sequential Logic)，組合邏輯階梯圖可將輸入裝置與輸出裝置藉由不同的邏輯設計法組合而成。如圖 2.11。而順序邏輯階梯圖具有回授結構之迴路，將輸出結果拉回當輸入條件。以圖 2.12 為例，當 X000 開啟且 X001 關閉時，Y000 就會開啟。而 Y000 開啟後會當作接點的輸入條件，故 X000 不管開啟或關閉皆不會影響 Y000，稱之自保持迴路。

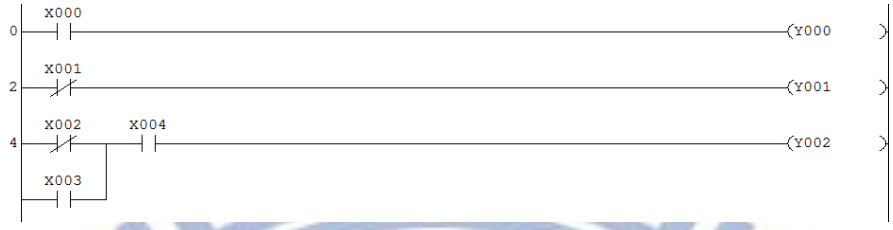


圖 2.11 組合邏輯階梯圖

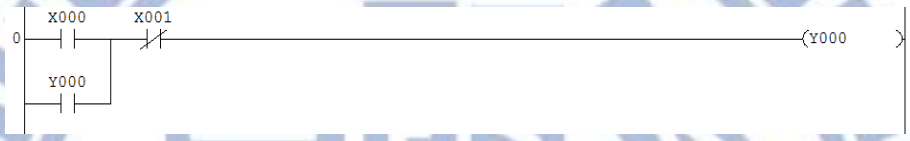


圖 2.12 順序邏輯階梯圖

2.2.2 階梯圖指令

階梯圖中有許多的指令可以應用，接著會介紹本論文會應用到的指令，其包括計時指令、自保持迴路指令、自保持迴路解除指令與呼叫指令。

計時指令是使用在線圈上面，其名稱使用 T 與 K 單位時間。當滿足輸入接點條件時，會計時 K 單位時間後，才會開啟線圈 T。如圖 2.13。

圖 2.13 計時指令範例

使用指令 SET 可讓線圈自保持迴路，而指令 RST 可讓線圈解除自保持迴路。如圖 2.14 中，當接點 X000 開啟時，線圈 Y000 直接以自保持迴路持續開啟。當接點 X001 開啟，則線圈 Y000 會解除自保持迴路而關閉。

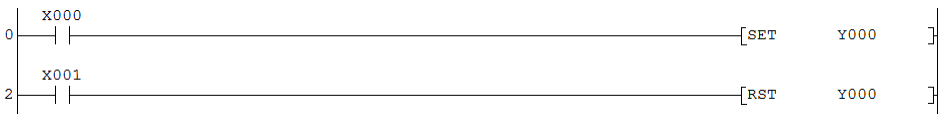


圖 2.14 自保持迴路與解除自保持迴路指令範例

呼叫指令主要是將階梯圖分成主程式與副程式兩個部份。用到的指令有 CALL、FEND、標籤 P 與 SRET。當滿足接點的輸入條件時，使用 CALL 就會呼叫標籤 P，使程式執行副程式 P 的部份。如圖 2.15 中，FEND 以上為主程式，標籤 P1 與 SRET 之間為副程式。當主程式執行到接點 X001 開啟時，接著會呼叫標籤 P1，則程式會跳到副程式 P1 的部份開始執行。

圖 2.15 呼叫指令範例

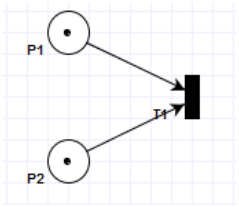
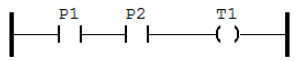
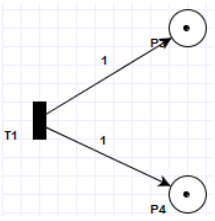
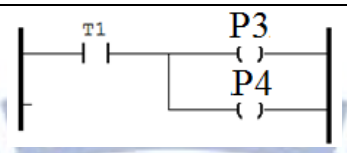
2.2.3 裴氏圖、階梯圖與法則的關係

前面分別介紹了裴氏圖與階梯圖，而裴氏圖與階梯圖皆能以法則的方式表達。因此可利用法則做為裴氏圖與階梯圖之間的轉換橋樑。

如果以法則來表示裴氏圖，以裴氏圖的轉移點來看可依輸入暫存點轉移點與輸出暫存點找出兩條法則，第一條為輸入暫存點代表法則的條件，而轉移點代表法則的事件。第二條為轉移點代表法則的條件，輸出暫存點代表法則的事件。如表 2.3 所示，T1 的輸入暫存點為 P1 和 P2，第一條法則為 P1^P2→T1。T1 的輸出暫存點為 P3 和 P4，第二條法則為 T1→P3^P4。

如果以法則來表示階梯圖，接點為法則的條件，線圈為法則的事件。一條法則就代表一條階梯。如表 2.3 所示，P1 與 P2 為接點，T1 為線圈，故第一條法則為 P1^P2→T1。T1 為接點，P3 與 P4 為線圈，故第二條法則為 T1→P3^P4。

表 2.3 法則以裴氏圖與階梯圖表達的型式

裴氏圖	法則	階梯圖
	P1∧P2→T1	
	T1→P3∧P4	

2.2.4 三菱廠牌 PLC 介紹

本論文使用的是三菱廠牌，型號為 FX3U 的 PLC[15]，本論文之階梯圖都是以此型號的規格撰寫。本節介紹內容主要是會使用到的外觀元件。

如圖 2.16 所示，此 PLC 外觀包含三個部分，分別為控制主機，輸出燈號與輸入開關。對於控制主機來說，這是存放人員撰寫的階梯圖，並依此階梯圖進行邏輯運算。對於輸出燈號來說，這是對應到階梯圖中的線圈。若燈號對應到的線圈為開啟，則此燈就會開啟。對於輸入開關來說，這是對應到階梯圖的接點。若輸入開關為開啟時，則對應到的接點就會是開啟的狀態。

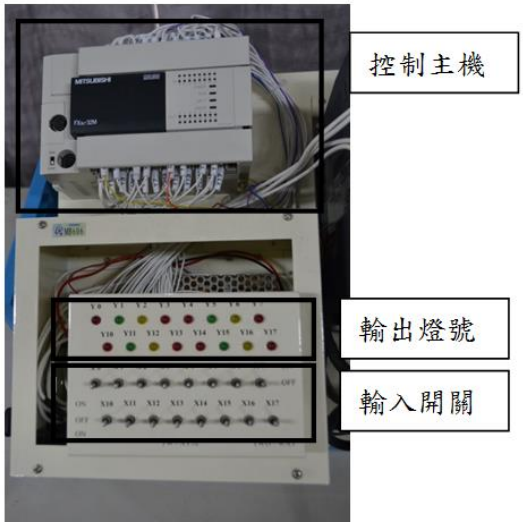


圖 2.16 三菱的 PLC 外觀元件

2.3 歐氏記號圖到階梯圖的法則轉換方法

本節主要說明前人所提出如何將歐氏記號圖轉換為階梯圖的方法。2.3.1 節主要說明歐氏記號圖之暫存點屬性分析，2.3.2 節說明法則R_I、R_E與R_F，最後 2.3.3 節說明法則轉換法。

2.3.1 歐氏記號圖之暫存點屬性分析

本小節將說明暫存點的屬性，歐氏記號圖的暫存點屬性分為三大類。分別為可觀測(Observable)且可直接量測(Directly Measurable)、可觀測但可間接量測(Indirectly Measurable)與不可觀測(Unobservable)的屬性。這些屬性的關係如圖 2.17。

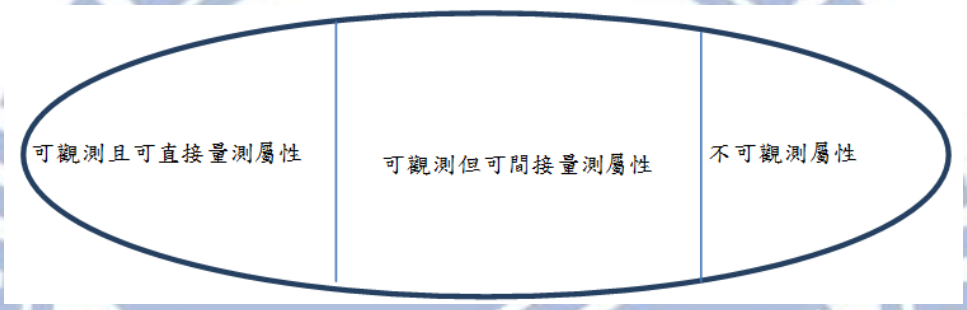


圖 2.17 暫存點的屬性

對於可觀測且可直接量測屬性的暫存點而言，這些屬性是系統能直接控制的，如設備的開啟與關閉。故這類型的暫存點可轉換為階梯圖元件，使 PLC 能進行控制。對於可觀測但可間接量測的屬性而言，其無法得知目前系統的狀態，但可透過其他屬性來推測目前狀態。如水槽的水位或液體的溫度。對於不可觀測屬性的暫存點而言，這些屬性不屬於系統內，是系統無法控制的。如工人進行補料或人員之間的訊息傳遞。若將此類暫存點轉成階梯圖，對於 PLC 是沒有任何意義。因此這類的暫存點必需刪除。

若歐氏記號圖具有可觀測且可直接量測的暫存點屬性，則稱之OM_D型歐氏記號圖；若歐氏記號圖具有可觀測但可間接量測的暫存點屬性，則稱之OM_I型歐氏記號圖；若歐氏記號圖具有不可觀測屬性的暫存點，則稱 U 型歐氏記號圖。將歐氏記號圖轉為階梯圖時，必需判斷此歐氏記號圖是屬於那一類型。如果是 U 型歐氏記號圖，則需刪除不可觀測屬性的暫存點，將其轉為OM_D或OM_I型歐氏記號圖，才能進行歐氏記號圖到階梯圖的轉換。

2.3.2 法則R_I、R_E與R_F

本節說明如何設定法則R_I、R_E與R_F，利用法則便能將歐氏記號圖轉換為階梯圖。首先將裴氏圖的初始浮標狀態設定成法則R_I，其中下標 I 為 Initial 之意。此法則的條件是放 X000，表示系統啟動。事件則是放初始暫存點並以「且」連接。「且」的符號以「∧」表示[29]。因為歐氏記號圖的初始狀態只有一種，故每張歐氏記號圖只有一條法則R_I。以圖 2.18 輸送

帶控制系統為例[41]，可知P1、P5、P7與P9為初始暫存點，故法則R_I為X000 →P1∧P5∧P7∧P9，表示當X000開啟時，則P1、P5、P7與P9會同時開啟。

轉移點觸發條件的法則稱為R_E，其中下標E為Enabled之意，將轉移點的輸入暫存點當作法則條件。以圖2.16為例，T2的輸入暫存點為P1與P5，T2的法則R_E可寫成P1∧P5→T2，代表著當P3有浮標時則開啟T1。而轉移點觸發後的狀態法則稱為R_F，其中下標F為Fired之意。其條件設定為轉移點。而事件則為轉移點的輸入暫存點和輸出暫存點以「且」連接，但是輸入暫存點要以「非」表示[29]，。以圖2.16為例，T2的輸出暫存點為P2與P4，輸入暫存點P1與P5，故法則R_F可寫成T2→P2∧P4∧$\overline{P1}$∧$\overline{P5}$，代表當T2開啟時，P2和P4開啟，P1與P5關閉。故每一個轉移點能寫出一個法則R_E與法則R_F。

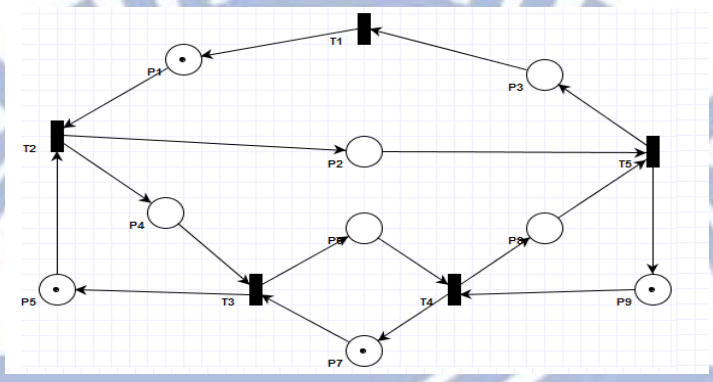


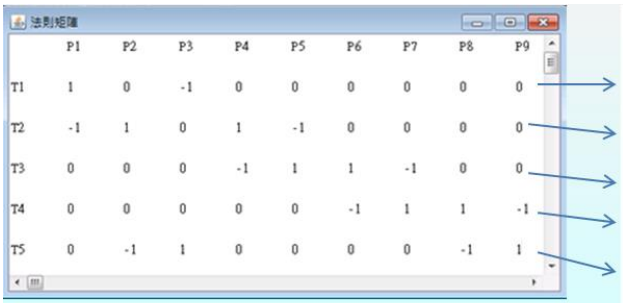
圖2.18 輸送帶控制系統之歐氏記號圖

2.3.3 法則轉換法

本節會根據前一節所提到的法則介紹法則轉換法，將歐氏記號圖轉換成階梯圖。浮標行為可分為(1)初始浮標，(2)轉移點的觸發條件，(3)觸發後的狀態三部分。對於初始浮標來說，這是指系統在開啟的時候，浮標所在的暫存點表示系統的初始狀態。對於轉移點的觸發條件來說，這是指若轉移點的輸入暫存點皆有浮標時，則此轉移點便可觸發。而以系統的物理意義來說，表示進行作業中。對於觸發後的狀態而言，這是指轉移點觸發後，浮標會從轉移點的輸入暫存點移動到輸出暫存點。表示作業結束，進入另一個狀態。因此法則轉換法將這三個部分轉換為階梯圖，使階梯圖擁有歐氏記號圖的浮標行為。

在轉換之前，必須由暫存點的屬性分類歐氏記號圖。由 2.3.1 節可知，若歐氏記號圖為U型，則必須刪除不可觀測屬性的暫存點。使得歐氏記號圖變成OM_D型或OM_I型才可以進行轉換。

以圖2.16的歐氏記號圖為例，首先以2.3.2節可知法則R_I為X000 →P1∧P5∧P7∧P9，接著將此歐氏記號圖以法則矩陣表達。首先將此歐氏記號圖以法則矩陣表達。如圖 2.19(a)。每一個轉移點皆能產生一個法則R_E與法則R_F。以T1轉移點來說，P3的值為-1，代表P3為輸入暫存點。故法則R_E為P3→T1。P1的值為1，代表P1為輸出暫存點。故法則R_F為T1→P1∧$\overline{P3}$。將每一個轉移點都轉成法則其結果如圖2.19(b)。



(a)法則矩陣

轉移點	法則R_E	法則R_F
T1	P3→T1	T1→P1∧$\overline{P3}$
T2	P1∧P5→T2	T2→P2∧P4∧$\overline{P1}$∧$\overline{P5}$
T3	P4∧P7→T3	T3→P5∧P6∧$\overline{P4}$∧$\overline{P7}$
T4	P6∧P9→T4	T4→P7∧P8∧$\overline{P6}$∧$\overline{P9}$
T5	P2∧P8→T5	T5→P3∧P9∧$\overline{P2}$∧$\overline{P8}$

(b)法則

圖 2.19 輸送帶控制系統之法則矩陣與法則

將法則R_I加上每一個轉移點的法則R_E和法則R_F後，將這些法則以階梯圖表達。以R_I：X000 →P1∧P5∧P7∧P9 為例，將其轉成階梯圖如圖 2.20。法則的條件當作接點，法則的結果當作線圈，並利用 SET 指令開啟線圈。故 X000 為接點，Y001、Y005、Y007 與 Y009 為 SET 線圈。

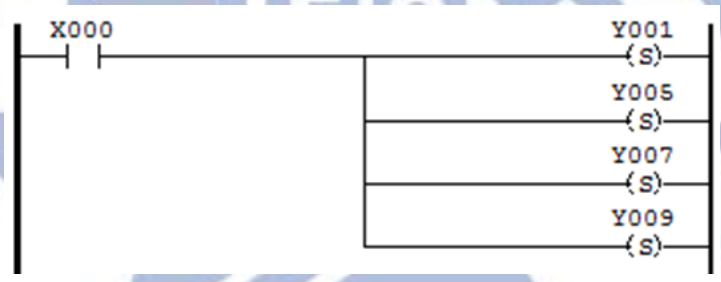


圖 2.20 法則R_I轉換成階梯圖

以法則R_E：P3→T1 為例，將其轉成階梯圖如圖 2.21。一樣以法則的條件當作接點，法則的結果當作線圈，並利用 SET 指令開啟線圈。故 Y003 為接點，T1 為 SET 線圈。

圖 2.21 法則R_E轉換成階梯圖

以法則R_F：T1→P1∧$\overline{P3}$為例，將其轉成階梯圖如圖 2.22。以法則的條件當作接點，法則的結果當作線圈，並利用 SET 指令開啟線圈，RST 指令關閉線圈。因浮標的移動是從 P3 移動到 P1，故 P1 在階梯圖裡必需關閉。故 T1 為接點，Y001 為 SET 線圈，Y003 為 RST 線圈。

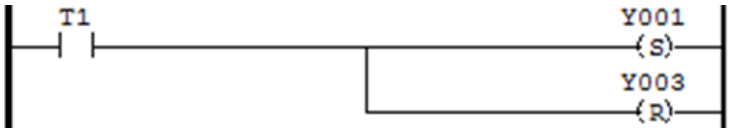


圖 2.22 法則R_F轉換成階梯圖

將法則R_I加上每一個轉移點的法則R_E和法則R_F後，將這些法則以階梯圖表達，便能使此階梯圖擁有歐氏記號圖的浮標行為。而只要原本的歐氏記號圖沒有鎖死[47]，則系統必可以回到初始狀態，即可循環製造。整體轉換流程可以整理如圖 2.23。

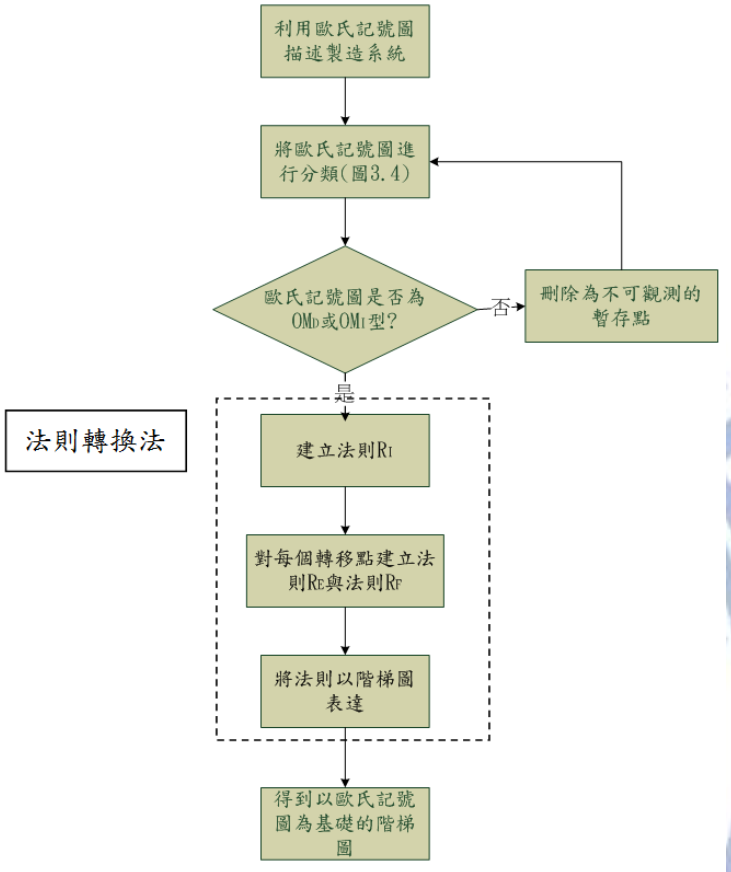


圖 2.23 歐氏記號圖轉換為階梯圖流程

2.4 三層式架構轉換法

三層式架構轉換法[11]是以 Java 程式為基礎，進行裴氏圖到階梯圖的自動轉換方法。由於本論文所使用的歐氏記號圖與浮標守恆裴氏圖皆為裴氏圖的一種，故亦可透過此轉換法進行自動轉換。下面將介紹三層式架構轉換法的背景。

因為在以前歐氏記號圖到階梯圖的自動轉換中，兩種圖形之間並沒有共同對照的標準，故難以執行自動化轉換。然而，隨著 XML 技術的發展，兩種圖形都可以使用 XML 文件表達。裴氏圖加註語言(Petri Net Markup Language, PNML)[37]為裴氏圖的 XML 文件表達法，而 PLCopen 加註語言(PLCopen eXtensible Markup Language, PLCopenXML)[34]為階梯圖的 XML 文件表達法。因此在自動轉換上便有共同對照的標準。之後隨著文件繫結技術 JAXB 的發展，可將文件轉換為 Java 物件，並透過法則進行裴氏圖物件到階梯圖物件的轉換。

因此，此方法的流程是先將裴氏圖以 PNML 文件形式表達，再透過 JAXB 的解編(Unmarshal)功能[31]將此 PNML 文件的元素轉換為 Java 物件。再利用法則轉換法進行物件的轉換。最後將轉換後的物件以 JAXB 的組編(Marshal)功能[31]轉換為 PLCopenXML 文件，再將此 PLCopenXML 文件以階梯圖的圖形表達以完成轉換流程，如圖 2.24 所示。由於轉換過程牽涉到圖件層、文件層與物件層，故稱為三層式架構轉換法。另外，在此流程中，除了物件的法則轉換會隨不同的轉換方法有所不同，其餘在圖件、文件到物件之間的轉換程序都已經發展成熟。2.4.1 節為圖件層分析。2.4.2 節為文件層分析。2.4.3 節為物件層分析。

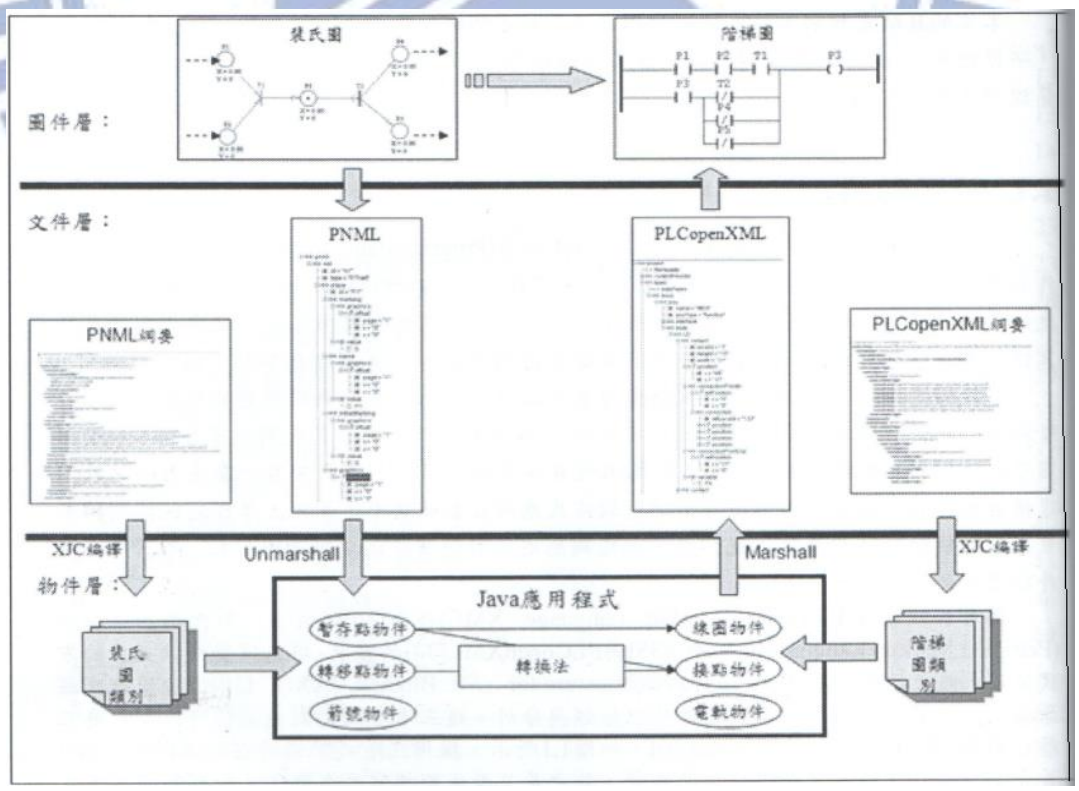


圖 2.24 三層式架構轉換法流程

2.4.1 圖件層分析

在圖件層中，裴氏圖轉階梯圖的過程沒有共同的對照標準，因此只可透過邏輯轉換的方式達成轉換。由 2.3.3 節可知，法則可作為裴氏圖到階梯圖的轉換橋樑。故使用法則進行轉換僅在人工處理上較為簡單，但在自動轉換上卻相當困難。不過隨著可統一標準的 XML 文件出現後，若把裴氏圖與階梯圖分別用 XML 文件型式表達。且因圖件與物件的轉換為一對一關係，則自動轉換便可基於這個一對一關係制定統一的標準來撰寫程式，達到自動轉換裴氏圖到階梯圖的目的。因此裴氏圖到階梯圖的轉換就可進入到文件層的分析。

2.4.2 文件層分析

對於文件層分析來說，這是指將裴氏圖與階梯圖以 XML 文件的方式表達。因為 XML 具有統一標準的優點，且可以跨平台傳遞資料，因此自動轉換程式便可基於此統一標準進行轉換。本節共分三小節。首先在 2.4.2.1 節介紹可加註語言，接著 2.4.2.2 節說明裴氏圖與裴氏圖加註語言，2.4.2.3 節說明 AutomationML 與 PLCopen 標準，最後在 0 節介紹階梯圖與 PLCopen 加註語言。

2.4.2.1 可加註語言

XML 是一種標註語言(Markup Language)，其最早是由全球資訊網聯合會(World Wide Web, W3C)[38]所發展出來，目的是要制定標準文件規格達到快速且正確的電子交換。XML 的前身為標準廣義加註語言(The Standard Generalized Markup Language, SGML)，而 XML 簡化了 SGML 中複雜與少用的特性。讓使用者可以簡單的定義需要的文件型態。

XML 主要有三個用途，分別為豐富檔案(Rich Documents)、後設資料(Metadata)與配置文件(Configuration Files)。豐富檔案可以自定檔案描述並使其更豐富。後設資料可描述其他檔案或網路資訊。配置文件可描述軟體設定的參數。

XML 為階層式的樹狀結構，每份 XML 文件都會包含 XML 宣告(XML Declaration)。以表 2.4 的 PNML 文件為例，其 XML 宣告為<?xml version="1.0" encoding="ISO-8859-1"?>，xml version 是指讀取 XML 文件時需要的規範，encoding 則是指 XML 文件中的字串所必需遵循的規範。而在 XML 中的每一列的資料項稱為元素(Element)，這些元素都會包含一個起始標籤(start-tags)與一個終止標籤(End-tags)。起始標籤以<元素名稱>命名，終止標籤以</元素名稱>命名。

XML 中主要特有的優點是有良好規格(Well-formed)文件與有效(Valid)文件。良好規格這是指文件都可以符合 XML 所要求的文件規格。這些文件的規格可見[25]。對於有效文件這是指文件有符合交換雙方所訂定的規範。而交易雙方也會設定驗證機制來確定傳送的文件是否有符合規範。文件定義技術將於下面說明兩種驗證機制。

驗證機制主要有文件型別定義(Document Type Definition, DTD)[20]與 XML 綱要(XML schema)[26]兩種。DTD 的作用在定義與規範特定 XML 文件如何編排撰寫。DTD 的驗證機

制是透過在 XML 文件的宣告與內文之間加入 DTD 語言，來定義文件規範，故 DTD 語言是內嵌在文件中。因此交易雙方會接收到 XML 文件時，會先讀取文件宣告部分，再讀取 DTD 語言，最後使用 DTD 語言驗證文件是否符合規範。XML 的驗證機制是透過外部檔案 XSD(XML Schema Definition)來驗證文件。因此交易雙方收到 XML 文件時，讀取文件宣告部分後，就會透過外部檔案 XSD 來驗證文件。

XML 綱要在文件規範上會比 DTD 更好更清楚[25]，且將文件轉換為物件時，綱要可以透過 JAXB 來產生各種所需的物件[26]。因此目前通常採用 XML 綱要作為驗證機制。

2.4.2.2 裴氏圖與裴氏圖加註語言的關係

裴氏圖加註語言是由國際標準組織(International Organization for Standardization, ISO)和國際電工協會(International Electrotechnical Commission, IEC)依照 ISO/IEC 15909 標準[33]制定。每份 PNML 文件即代表一個裴氏圖。故在 PNML 中也包含這些元件。

本論文使用裴氏圖繪圖軟體 PIPE[39]繪製裴氏圖。PIPE 軟體的優點為當繪製完裴氏圖後，可直接以 PNML 文件的儲存方式儲存。同樣地，PIPE 亦可匯入已寫好的 PNML 文件，以圖件方式開啟並顯示裴氏圖，達到圖件與文件的互相轉換。

接下來舉一個裴氏圖的圖件要如何以 PNML 文件方式表達。而在 PNML 文件中分別用標籤<place>、<transition>、<initialMarking>與<arc>來代表暫存點、轉移點、初始符標與方向弧。圖 2.25 的文件表達方式整理如表 2.4 所示。

接著說明會使用的元素屬性。首先由表 2.4 可看出每一項標籤都會有附屬標籤如 id，此屬性的目的主要是給電腦所辨識。但使用人員在電腦所看到的圖形名稱則是以元素<name>下的子元素<value>所表示。其次裴氏圖的初始浮標是利用<initialMarking>下的子元素<value>所表示，元素<value>內的數字表示暫存點在初始狀態時的浮標數，0 代表沒有符標，1 代表有浮標。方向弧內包含了 source 和 target 屬性，用來表示轉移點和暫存點之間浮標轉移的方向。若 source 是轉移點，則 target 就會是暫存點，且此暫存點為轉移點的輸出暫存點，反之若 target 是轉移點，則 source 就會是暫存點，且此暫存點為轉移點的輸出暫存點。因此可利用這兩個屬性來建立法則矩陣。

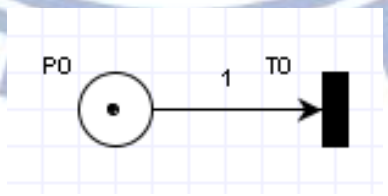
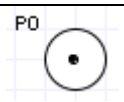


圖 2.25 裴氏圖範例

表 2.4 裴氏圖元件與文件表達方式

元件名稱	元件圖形	PNML 文件內容
		<?xml version="1.0" encoding="iso-8859-1" ?> - <pnml> - <net id="Net-One" type="P/T net"> <token id="Default" enabled="true" red="0" green="0" blue="0" />
暫存點		- <place id="P0"> - <graphics> <position y="75.0" x="165.0"/> </graphics> - <name> <value>P0</value> - <graphics> <offset y="35.0" x="-5.0"/> </graphics> </name> - <initialMarking> <value>Default,0</value> - <graphics> <offset y="0.0" x="0.0"/> </graphics> </initialMarking> - <capacity> <value>0</value> </capacity> </place>
轉移點		- <transition id="T0"> - <graphics> <position x="300.0" y="90.0" /> </graphics> - <name> <value>T0</value> - <graphics> <offset x="0.0" y="0.0" /> </graphics> </name> - <orientation> <value>0</value> </orientation> - <rate> <value>1.0</value> </rate> - <timed> <value>false</value> </timed> - <infiniteServer> <value>false</value> </infiniteServer> - <priority> <value>1</value> </priority> </transition>
方向弧		- <arc id="T0 to P0" source="T0" target="P0"> <graphics /> - <inscription> <value>1</value> <graphics /> </inscription> - <tagged> <value>false</value> </tagged> <arcpath id="000" x="331" y="177" curvePoint="false" /> <arcpath id="001" x="447" y="177" curvePoint="false" /> <type value="normal" /> </arc>
		</net> </pnml>

2.4.2.3 AutomationML 與 PLCopen 標準

AutomationML(Automation Markup Language)[27]為一種以 XML 為基礎的文件標準格式，可儲存及交換工廠機械設備的資訊。AutomationML 主要目的在整合一些工程工具，如 PLC、機械手臂控制(Robot Control)[28]等。其格式分別為佈局(Topology)、幾何(Geometry)、運動學(Kinematics)與邏輯(Logic)。

PLCopenXML 為 AutomationML 之邏輯部份使用的標準，為 PLCopen[34]所制定的規範。PLCopen 為一個工業控制程式相關主題的國際性組織，其中利用 IEC61131-3[21]中統一的標準規範，訂定了 PLCopen 加註語言。PLCopen 加註語言為 XML 文件模式，提供了統一的文件格式給所有人使用。並且可跨平台與網際網路傳送，應用於不同的領域。本論文 XML 轉換的部份與 PLC 邏輯有關，故使用 PLCopen 加註語言做為轉換的依據。

2.4.2.4 階梯圖與 PLCopenXML 的關係

本論文使用 PLCopenEditor 軟體繪製階梯圖，並且可以進行 PLCopenXML 與階梯圖的轉換。儲存的格式以 PLCopenXML 文件儲存；亦可使用軟體匯入 PLCopenXML 文件以階梯圖方式呈現。如圖 2.26 的階梯圖，以 PLCopenXML 文件表達可整理如表 2.5 所示，可看出階梯圖與 PLCopenXML 文件具有一對一對應的關係。故可以互相轉換。

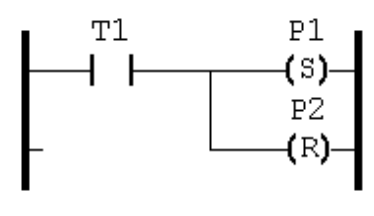


圖 2.26 階梯圖範例

由表 2.5 可知 PLCopenXML 文件開頭為基本資訊。本論文在歐氏記號圖與階梯圖的轉換過程，主要使用的是元素<body>下的四個子元素<leftPowerRail>、<rightPowerRail>、<contact>與<coil>，分別為左電軌、右電軌、接點與線圈。這四個元素都有屬性 localId，其功用是給電腦辨識其名字，使用人員所看到的接點和線圈名稱則是由子元素<variable>所表示的名稱。接著屬性 height 和 width 分別為圖形元件的高度與寬度，子元素<position>則是圖形元件的座標。以<contact>為例，其<reflocalId>為 1，表示連線起點的圖件是<leftPowerRail>。若連線沒有跨行，如 T1 到 P1 的連線，則<connection>有兩個子元素<position>，第一個是指連線終點座標，第二個則是連線起點座標。若有跨行，如 T1 到 P2 的連線，則<connection>有四個子元素<position>，第二和第三個<position>代表圖型跨行中線端點座標。最後元素<coil>中有屬性 storage，如果為 set，則表示此線圈設定為前面所提的 SET 指令線圈；若為 reset，則表示為 RST 指令線圈，其圖件外觀也會如 P1 和 P2 所示有所不同。

表 2.5 階梯圖元件與文件表達方式

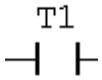
元件名稱	元件圖形	PLCopenXML 文件內容
		<?xml version="1.0" encoding="UTF-8" ?> - <project xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://www.plcopen.org/xml/tc6.xsd" xmlns:xhtml="http://www.w3.org/1999/xhtml" xsi:schemaLocation="http://www.plcopen.org/xml/tc6.xsd"> <fileHeader companyName="temp" productName="temp" productVersion="temp" creationDateTime="2012-02-28T20:28:53" /> - <contentHeader name="temp" modificationDateTime="2012-02-28T20:32:58"> - <coordinateInfo> - <fbd> <scaling x="0" y="0" /> </fbd> - <ld> <scaling x="0" y="0" /> </ld> - <sfc> <scaling x="0" y="0" /> </sfc> </coordinateInfo> </contentHeader> - <types> <dataTypes /> - <pous> - <pou name="temp" pouType="function"> - <interface> - <returnType> <BOOL /> </returnType> - <inOutVars> - <variable name="T1"> - <type> <BOOL /> </type> </variable> - <variable name="P1"> - <type> <BOOL /> </type> </variable> - <variable name="P2"> - <type> <BOOL /> </type> </variable> </inOutVars>
leftPowerRail		- <leftPowerRail localId="1" height="80" width="2"> <position x="46" y="65" /> - <connectionPointOut formalParameter=""> <relPosition x="2" y="20" /> </connectionPointOut> - <connectionPointOut formalParameter=""> <relPosition x="2" y="60" /> </connectionPointOut> </leftPowerRail>
rightPowerRail		- <rightPowerRail localId="5" height="80" width="2"> <position x="275" y="64" /> - <connectionPointIn> <relPosition x="0" y="20" /> - <connection refLocalId="3"> <position x="275" y="84" /> <position x="248" y="84" /> </connection> </connectionPointIn> - <connectionPointIn> <relPosition x="0" y="60" /> - <connection refLocalId="4"> <position x="275" y="124" /> <position x="249" y="124" /> </connection> </connectionPointIn> </rightPowerRail>
contact		- <contact localId="2" height="15" width="21"> <position x="75" y="77" /> - <connectionPointIn> <relPosition x="0" y="8" /> - <connection refLocalId="1"> <position x="75" y="85" /> <position x="48" y="85" /> </connection> </connectionPointIn> - <connectionPointOut> <relPosition x="21" y="8" /> </connectionPointOut> <variable>T1</variable> </contact>

表 2.6 階梯圖元件與文件表達方式(續)

<table>
<tr>
<td>coil</td>
<td>P1
—(S)—
P2
—(R)—</td>
<td>
<pre>
- <coil localId="3" height="15" width="21" storage="set">
    <position x="203" y="89" />
  - <connectionPointIn>
      <relPosition x="0" y="8" />
    - <connection refLocalId="2">
        <position x="203" y="97" />
        <position x="124" y="97" />
      </connection>
    </connectionPointIn>
  - <connectionPointOut>
      <relPosition x="21" y="8" />
    </connectionPointOut>
    <variable>P1</variable>
  </coil>

- <coil localId="5" height="15" width="21" storage="reset">
    <position x="204" y="129" />
  - <connectionPointIn>
      <relPosition x="0" y="8" />
    - <connection refLocalId="2">
        <position x="204" y="137" />
        <position x="164" y="137" />
        <position x="164" y="97" />
        <position x="124" y="97" />
      </connection>
    </connectionPointIn>
  - <connectionPointOut>
      <relPosition x="21" y="8" />
    </connectionPointOut>
    <variable>P2</variable>
  </coil>
</pre>
</td>
</tr>
<tr>
<td></td>
<td></td>
<td>
<pre>
      </pou>
    </pous>
  </types>
- <instances>
    <configurations />
  </instances>
</project>
</pre>
</td>
</tr>
</table>

2.4.3 物件層分析

在 2.4.2 節說明了裴氏圖和階梯圖的標準文件規範，在轉換過程中可依此標準進行裴氏圖到階梯圖的轉換。然而，使用文件進行轉換時，若裴氏圖不同時，則其文件也會不同。故必須針對每個不同的文件撰寫程式才可以得到所需的元素和屬性，導致程式設計上較為困難。因此若可以將 PNML 文件中的元素轉換為物件，則可以透過判斷物件的屬性，決定要轉換為何種階梯圖物件，且程式只需撰寫一次。本節在 2.4.3.1 節先說明文件和物件如何透過 JAXB 互相轉換。在 2.4.3.2 節介紹 Eclipse 在 2.4.3.3 節介紹如何將 PNML 文件轉換到物件，在 2.4.3.4 節介紹階梯圖物件轉換到 PLCopenXML 文件。

2.4.3.1 JAXB 架構

JAXB 技術主要提供兩種功能，第一個是 XML 綱要與 Java 類別之間的繫結(Binding)，第二個則是 XML 文件和 Java 物件的對應轉換關係，讓程式開發人員能快速地將 XML 中的資料整合到 Java 程式中進行處理。對於提供 XML 綱要與 Java 類別之間的繫結來說，JAXB 提供 XJC 和 Schemagen 兩種編譯器(Compiler)。前者可以將 XML 綱要轉換為 Java

類別，後者可以將 Java 類別轉換為 XML 綱要。因為本論文只有將 PNML 和 PLCopenXML 的綱要生成 Java 類別，所以不會使用 Schemagen 功能。JAXB 資料處理的流程如圖 2.27。

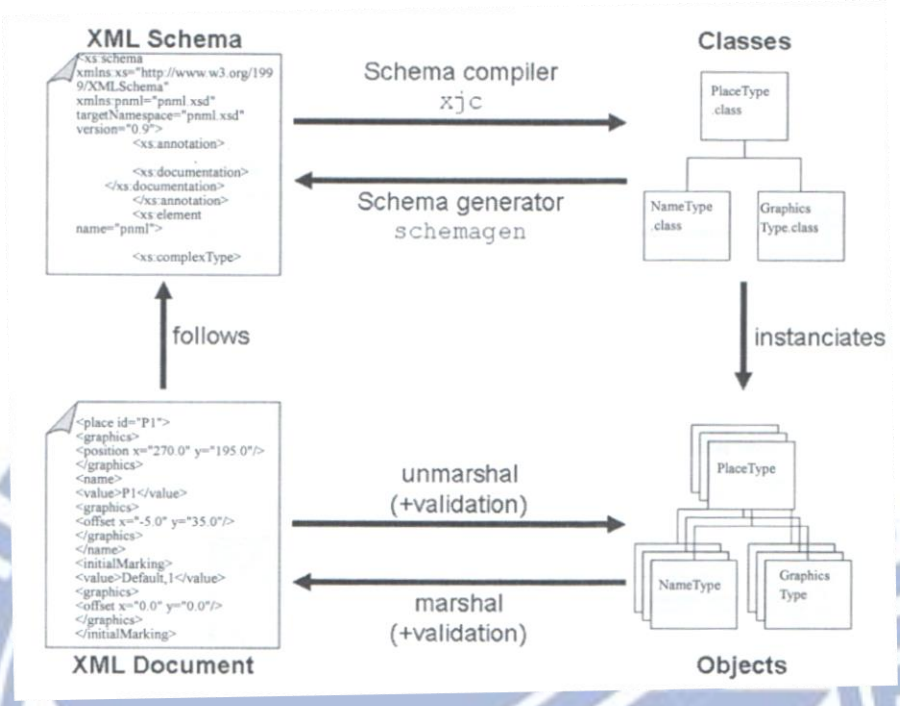


圖 2.27 JAXB 資料處理流程

對於 XML 元素和 Java 物件的對應關係來說，JAXB 提供組標籤與解標籤功能。前者可將 Java 物件轉換為 XML 文件，後者可將 XML 文件轉換為 Java 物件。透過這兩種功能，可進行文件和物件之間的轉換。組標籤與解標籤的流程如圖 2.28。

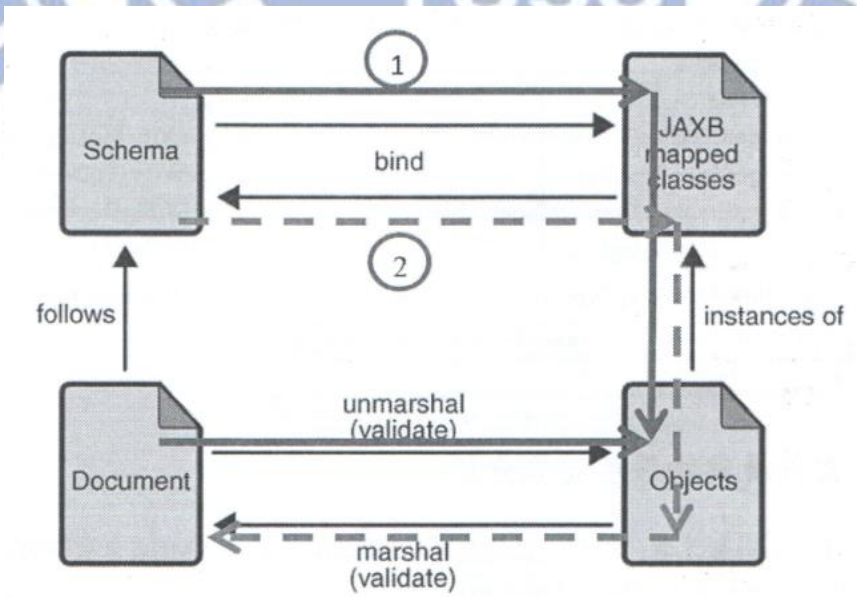


圖 2.28 組標籤與解標籤流程[32]

2.4.3.2 IDE Eclipse 與 JAXB 的關係

Eclipse 是著名跨平台的整合式開發環境(Integrated Development Environment, IDE)。其主要是以 JAVA 語言所開發。Eclipse 本身是一個框架平台，支援眾多的外掛程式，亦支援各個作業系統，使多數使用者可利用 Eclipse 設計自己的 IDE。

在 2.4.3.1 小節有完整介紹 JAXB 的架構，而 JAXB 相關工具大多會使用 Eclipse，因為 Eclipse 擁有完整 JAXB 功能，可以直接顯示在 Eclipse 介面上，去除掉使用 JAXB 在命令提示字元輸入命令的時間。而在處理一個 XML Schema 時，JAXB 與 Eclipse 差異不大，但處理多個 XML Schema 時，Eclipse 會變得簡單很多。

2.4.3.3 PNML 文件到歐氏記號圖物件

本節說明如何以 Java 程式將 PNML 文件解編到歐氏記號圖物件。在解編 PNML 文件時，首先在 Java 程式內產生物件 JAXBContext，其為 JAXB 程式的起點。在產生的時候，必須指定 PNML 綱要產生的類別的套件路徑，如圖 2.29 第一行所示。使得解編 PNML 文件後的物件會遵循這些類別的定義。接著第二行程式產生 Unmarshaller 物件，使得在第三行程式呼叫 Unmarshaller 物件中的方法 unmarshal()解標籤文件轉換為物件，再把轉換後的物件資訊儲存到 Pnml 物件中。最後第四行再利用 Pnml 物件呼叫方法 getNet()得到暫存點、轉移點、方向弧等物件的資訊，再以資料結構的清單(List)形式儲存到 NetType 物件。透過這四行程式，便可將 PNML 文件中的所有元素轉換為 Java 物件，提供給人員使用。

```
JAXBContext jaxbContext = JAXBContext.newInstance("mappedClassfromPNML");
Unmarshaller unMarshaller = jaxbContext.createUnmarshaller();
Pnml pnmlElement = (Pnml) unMarshaller.unmarshal(PNML);
NetType nettype = pnmlElement.getNet();
//若為暫存點物件, 則印出其屬性
for(int i=0;i<nettype.getPlaceOrTransitionOrArc().size();i++){
    if(nettype.getPlaceOrTransitionOrArc().get(i) instanceof PlaceType){
    PlaceType p=(PlaceType)nettype.getPlaceOrTransitionOrArc().get(i);
    System.out.println("place id="+p.getId());
    System.out.println("position x="+p.getGraphics().getPosition().get(0).getX());
    System.out.println("position y="+p.getGraphics().getPosition().get(0).getY());
    System.out.println("offset x="+p.getName().getGraphics().getOffset().getX());
    System.out.println("offset y="+p.getName().getGraphics().getOffset().getY());
    System.out.println("placeName="+p.getName().getValue());
    System.out.println("InitialMarking="+p.getInitialMarking().getValue());
    }
}
```

圖 2.29PNML 文件到歐氏記號圖物件

```
place id=P0
position x=165.0
position y=75.0
offset x=-5.0
offset y=35.0
placeName=P0
InitialMarking=Default,0
```

圖 2.30 取得暫存點物件的例子

下面以表 2.4 的 PNML 文件為例，當文件匯入程式後，透過前段所述的四行程式，其轉換後的物件會儲存在 NetType 物件中。若要取得轉換後的物件，則可利用 NetType 的 getPlaceOrTransitionOrArc()方法取得物件。以取得暫存點物件為例，其程式碼如圖 2.29 黑

線取選部分。由於 Java 程式的運算子 instanceof 是用來比對物件是否屬於特定的類別[30]。因此利用此運算子，依序比對 NetType 中的物件是否屬於類別 PlaceType。若物件屬於 PlaceType 類別，則印出其屬性，圖 2.30 為結果。

同理，如需轉移點物件與方向弧物件也可透過前面的方式取得，並且印出其屬性。另外，由這個例子可知文件在解編之後，的確是以物件的形式儲存，而且是 PNML 綱要產生的類別的實體，否則無法使用運算子 instanceof 進行比對。

2.4.3.4 階梯圖物件到 PLCopenXML 文件

接下來說明如何由階梯圖物件轉換到 PLCopenXML 文件。圖 2.31 中第一行程式首先產生一個物件 JAXBContext，並指定 PLCopenXML 綱要產生類別的套件路徑。第二行程式產生組標籤物件 Marshaller。第三行程式產生一個物件 ObjectFactory，這個物件的功能是用來產生階梯圖物件。為了使輸出文件符合 PLCopenXML 綱要的規範，可透過物件 ObjectFactory 產生符合綱要規範所需的物件，包含 Project、FileHeader、ContentHeader、Instance、Types、Pou、Body 與 LD 等物件，並設定基本資訊，如圖 2.31 第一個框選的部分。接下來以產生一個接點物件為例，其屬性設定仿照表 2.5 的接點 T1，如圖 2.31 右邊所示。再來使用 Marshaller 物件的方法 setProperty("jaxb.formatted.output", Boolean, True)，使輸出的文件具有自動換行與縮排的規格。最後使用方法 marshal 組標籤階梯圖物件為 XML 文件。文件結果如圖 2.32 所示，與表 2.5 的接點 T1 屬性相同。透過本段的方式，就可以達到階梯圖物件轉換到 PLCopenXML 文件的目的。

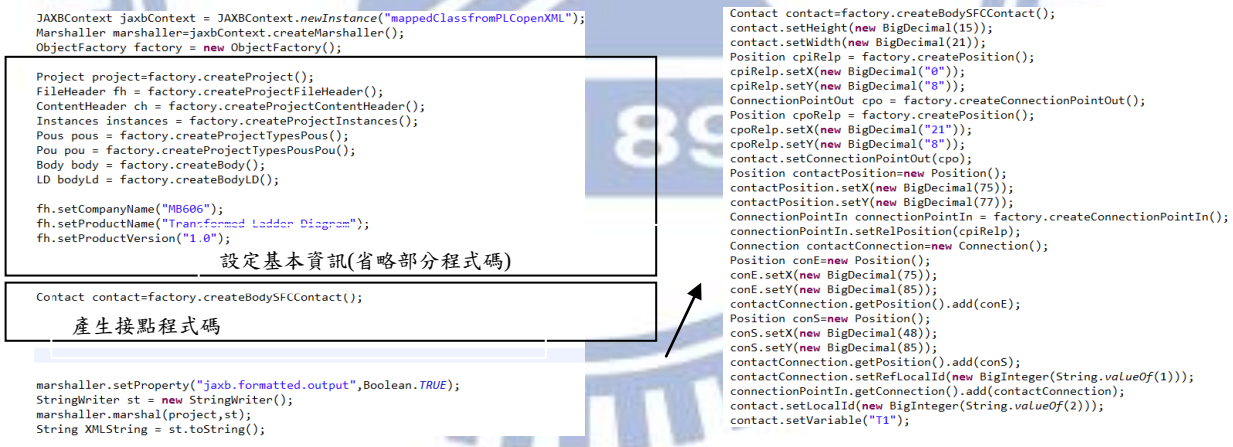


圖 2.31 階梯圖物件到 PLCopenXML 文件

```
<contact width="21" height="15" localId="2">
    <position y="77" x="75"/>
    <connectionPointIn>
        <relPosition y="8" x="0"/>
        <connection refLocalId="1">
            <position y="85" x="75"/>
            <position y="85" x="48"/>
        </connection>
    </connectionPointIn>
    <connectionPointOut>
        <relPosition y="8" x="21"/>
    </connectionPointOut>
    <variable>T1</variable>
</contact>
```

圖 2.32 接點物件到文件的例子

2.5 自動灌模系統介紹

自動灌模系統為單一產品重覆性生產的自動化製造系統，此系統之目的在於使用灌漿造模的方式進行模型的製作，如圖 2.33 所示。對於系統架構而言，自動灌模系統由兩個子系統所構成，分別為備料系統與澆模系統，其中備料系統的功能主要在於調製出模型硬化成型所需的物料；澆模系統的功能在於運載模具至指定的位置並依據模具上無線射頻(Radio Frequency Identification, RFID)[10]標籤內所記錄的製程參數進行澆模作業。

圖 2.33 自動灌模系統

對於系統架構而言，自動灌模系統依作業性質的不同可分為兩個獨立的子系統，分別為備料系統與澆模系統，如圖 2.34 所示。備料系統的硬體設備包含三個儲存槽、三個幫浦、三個液位感知器、兩個攪拌器以及兩個閥門，其主要的功能為調製出模型硬化成行所需的物料 X。調製過程中可分為三個步驟，首先會利用幫浦輸入物料 A、B 至一號儲存槽，經過攪拌後調製出物料 Y，接著，使用幫浦輸入物料 H 至二號儲存槽，與物料 Y 攪拌混和後調製出物料 X，最後，物料 X 會在三號儲存槽中等待使用。而澆模系統的硬體設備包含一個閥門、一組輸送帶、一個 RFID 讀取器暨天線以及四個紅外線感知器，當紅外線感知器感應到容器進入後，即啟動輸送帶，接著，RFID 讀取器會讀出標籤內所記錄的製程參數進行澆模作業，最後，紅外線感知器感應到容器離開後，即停止輸送帶。自動化灌模系統內各裝置的詳細資料如表 2.7 所示。

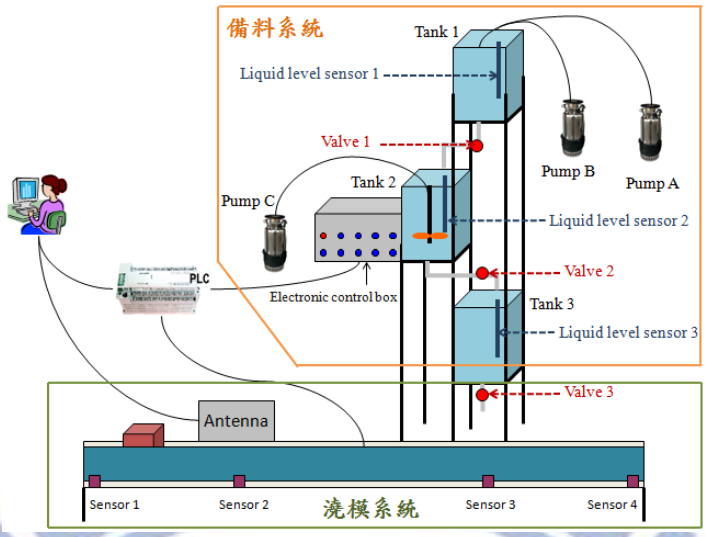


圖 2.34 自動灌模系統示意圖[10]

表 2.7 自動灌模系統內各裝置的詳細資料

隸屬系統	裝置名稱	裝置功能
備料系統	一號儲存槽	調製物料 Y
	二號儲存槽	調製物料 X
	三號儲存槽	儲存物料 X
	幫浦 A	輸入物料 A
	幫浦 B	輸入物料 B
	幫浦 C	輸入物料 C
	一號液位感知器	感應一號儲存槽最高水位
	二號液位感知器	感應二號儲存槽最高水位
	三號液位感知器	感應三號儲存槽最高及最低水位
	一號攪拌器	混合物料 A、B
	二號攪拌器	混合物料 Y、H
	一號閥門	控制物料 Y 注入二號儲存槽
	二號閥門	控制物料 X 注入三號儲存槽
澆模系統	三號閥門	控制物料 X 注入模具
	輸送帶	運送模具
	RFID 讀取器暨天線	讀取模具 RFID 標籤資料
	一號紅外線感知器	感應模具進入澆模系統
	二號紅外線感知器	模具 RFID 讀取位置定位
	三號紅外線感知器	模具澆模位置定位
	四號紅外線感知器	感應模具離開澆模系統
備料系統與澆模系統	可邏輯控制器	整合硬體裝置與控制程式

第三章　浮標守恆裴氏圖到階梯圖的轉換設計

本章提出浮標守恆裴氏圖轉換成浮標守恆階梯圖的流程與轉換方法。首先說明轉換過程中用到的浮標守恆裴氏圖、歐氏記號圖、同步裴氏圖三者與階梯圖彼此的關係，再來分析這些物件的組成與關係。最後提出歐氏記號圖與同步裴氏圖轉換到階梯圖，其階梯圖包含初始階梯圖、同步階梯圖與歐氏階梯圖。本章分四節，3.1 節為浮標守恆裴氏圖、歐氏記號圖、同步裴氏圖與階梯圖的關係，3.2 節為同步裴氏圖與同步階梯圖之設計，3.3 節為歐氏記號圖與階梯圖的物件關係，3.4 節為浮標守恆裴氏圖到浮標守恆階梯圖轉換案例說明。

3.1 浮標守恆裴氏圖、歐氏記號圖、同步裴氏圖與階梯圖的關係

彈性製造系統可由浮標守恆裴氏圖來表達，而浮標守恆裴氏圖可經由時域分解法拆成數張歐氏記號圖，每一張歐氏記號圖代表著彈性製造系統內的某一項製程造流程。本論文利用 2.1.6 節提到詮釋型裴氏圖的概念，將歐氏記號圖加入控制暫存點。同步裴氏圖的控制暫存點是由外部環境所控制此暫存點的浮標。當外部人員想使用某一個生產方式時，給予命令至 PLC 上面，控制暫存點便會擁有浮標，使得製造系統執行該張歐氏記號圖的生產方式。

浮標守恆裴氏圖、歐氏記號圖、同步裴氏圖三者與階梯圖的關係如圖 3.1。浮標守恆裴氏圖可分解成多張歐氏記號圖，將多張歐氏記號圖各自加入同步裴氏圖成為詮釋型歐氏記號圖，最後將多張加入控制暫存點之歐氏記號圖轉成階梯圖並且合併。詳細的轉換步驟於 3.4 節會詳細說明。

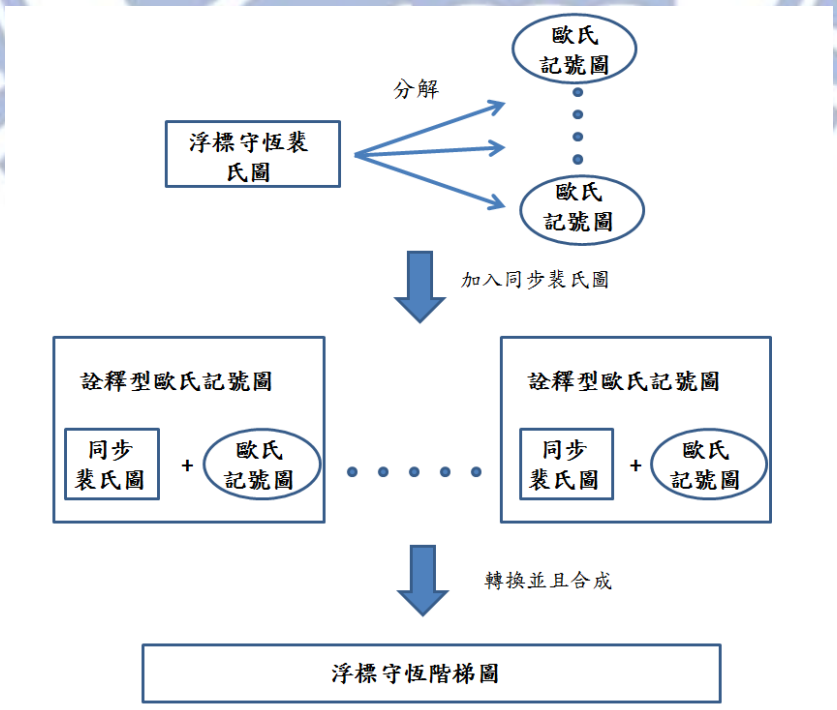


圖 3.1 浮標守恆裴氏圖到階梯圖轉換流程

3.2 同步裴氏圖與同步階梯圖之設計

在 3.1 節介紹到歐氏記號圖加入同步裴氏圖成為詮釋型歐氏記號圖。本節將介紹浮標守恆裴氏圖轉換成浮標守恆階梯圖過程中，如何設計同步裴氏圖，並且說明同步裴氏圖如何轉換成階梯圖。

舉一個簡單的例子來說，圖 3.2 為一個浮標守恆裴氏圖分解成兩張歐氏記號圖例子，此浮標守恆裴氏圖可分解歐氏記號圖 A 與歐氏記號圖 B。歐氏記號圖 A 與歐氏記號圖 B 分別代表不同的生產方式。

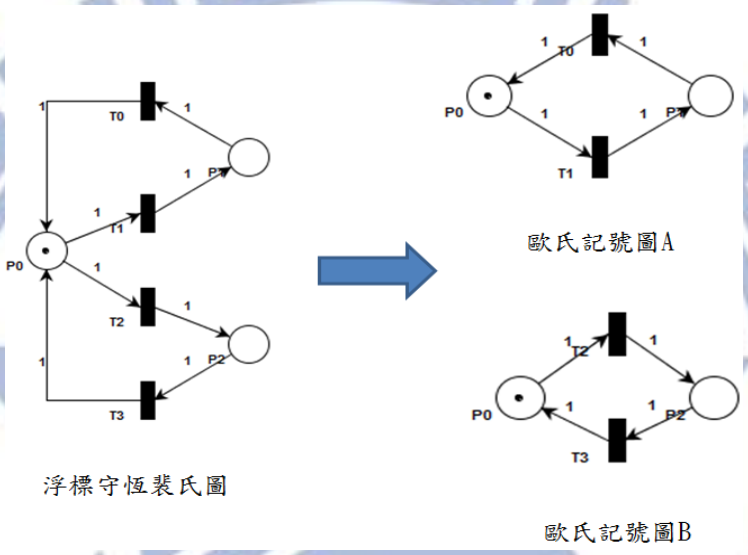


圖 3.2 浮標守恆裴氏圖拆解成歐氏記號圖例子

設計同步裴氏圖時，首先找出浮標守恆裴氏圖的選項暫存點，由圖 3.2 可知其選項暫存點為 P0，此目的是為了找出連結選項暫存點的選項轉移點，可以看出選項轉移點為 T1 與 T2。最後將分解後的歐氏記號圖 A 與歐氏記號圖 B 將選項轉移點 T1 與 T2 各加入控制暫存點 P3 與 P4。將選項轉移點加入控制暫存點便稱為同步裴氏圖，其結果如圖 3.3。詮釋型歐氏記號圖便是由歐氏記號圖加入同步裴氏圖組合而成，其控制暫存點 P3 與 P4 是由外部環境所決定是否擁有浮標。若想使用歐氏記號圖 A 的生產方式，P3 便會擁有浮標，使其能觸發 T1；若使用歐氏記號圖 B 的生產方式，P4 便會擁有浮標，使其能觸發 T2。

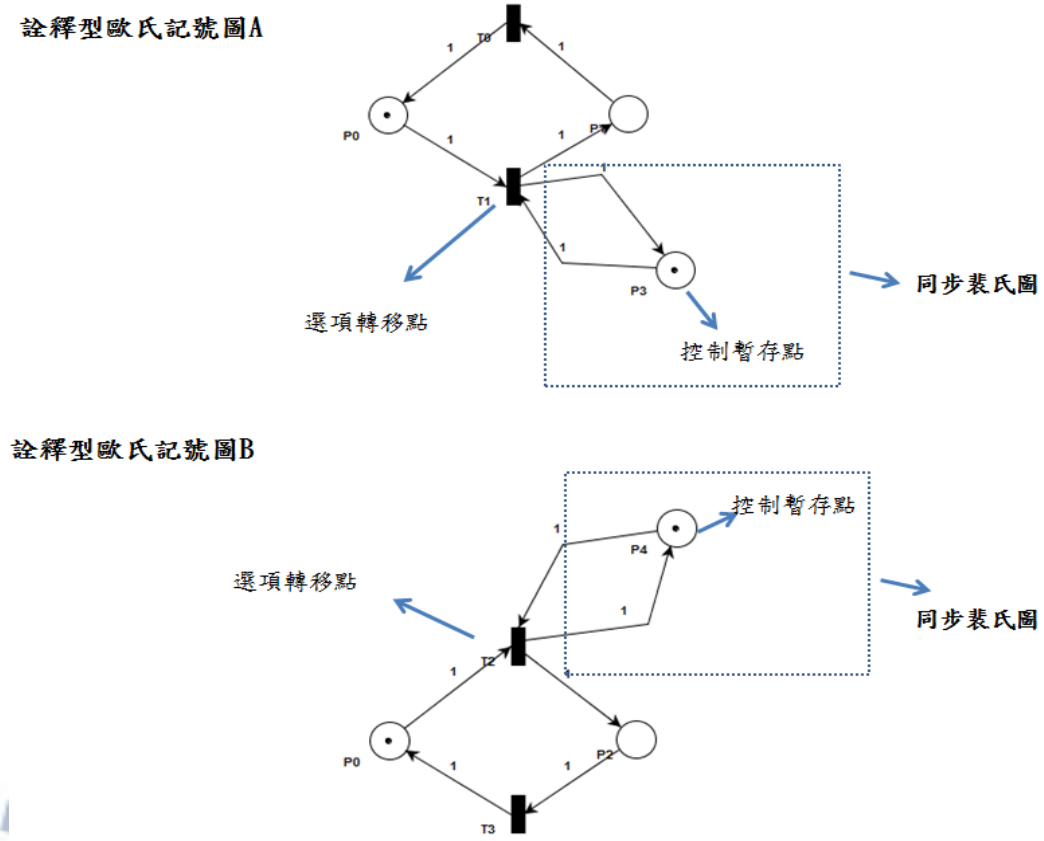


圖 3.3 歐氏記號圖加入同步裴氏圖

將這些詮釋型歐氏記號圖轉成階梯圖中，同步裴氏圖裡之控制暫存點需由外部環境決定浮標的有無。故在階梯圖設計中，需要設計由外部接點來控制其線圈開關。以上例來說，兩張詮釋型歐氏記號圖，則需設計兩個接點與控制暫存點 P3 與 P4 對應之線圈。如圖 3.4，新增一列階梯加入接點 X001 與線圈 Y003；再新增一列階梯加入接點 X002 與線圈 Y004。接點 X001 與 X002 代表著 PLC 外部控制的開關，線圈 Y003 與 Y004 的開與關代表著控制暫存點浮標的有無。若外部環境或使用人員欲選擇使用歐氏記號圖 A 的生產方式，及開啟 PLC 外部開關 X001，開啟接點 X001，則線圈 Y003 便開啟，亦代表著控制暫存點 P3 擁有浮標。當 P0 有浮標時，則會觸發 T1，使得使用歐氏記號圖 A 的生產方式。

圖 3.4 同步階梯圖設計

3.3 歐氏記號圖與階梯圖的物件關係

在 2.4.3 節物件層分析中，有介紹如何使用 JAXB 裡提供的 XJC 編譯器來解析綱要，將 PNML 與 PLCopenXML 綱要解析，產生有效物件。本節將利用 XJC 編譯器來解析 PNML 與 PLCopenXML 綱要，說明歐氏記號圖與階梯圖所組成的物件，並且說明其物件對應的關係。3.2.1 節說明歐氏記號圖的物件組成分析，3.2.2 節說明階梯圖的物件組成分析，3.2.3 節說明歐氏記號圖與階梯圖物件對應關係。

3.3.1 歐氏記號圖的物件組成分析

本節將介紹歐氏記號圖會產生那些物件，以及這些的物件功能為何。因本論文使用歐氏記號圖作為轉換的基礎，故本論文會以歐氏記號圖物件表示為 PNML 轉換後的物件。由 2.4.3.3 節的說明可知，轉換後的歐氏記號圖物件須依 PNML 綱要產生的類別定義。故首先使用 XJC 編譯器將 PNML 綱要[13]產生 17 項類別。圖 3.5 是利用 XJC 編譯器將 PNML 綱要進行解析，XJC 使用方法可參考[11]。

```
parsing a schema...
compiling a schema...
generated\ArcType.java
generated\GraphicsType.java
generated\HiddenType.java
generated\InitialMarkingType.java
generated\InscriptionType.java
generated\NameType.java
generated\NetType.java
generated\ObjectFactory.java
generated\OffsetType.java
generated\PageType.java
generated\PlaceType.java
generated\Pnml.java
generated\PositionType.java
generated\ReferencePlaceType.java
generated\ReferenceTransitionType.java
generated\ToolspecificType.java
generated\TransitionType.java
```

圖 3.5 PNML 綱要經由編譯之物件

本論文在此直接整理產生類別與其功能說明，如表 3.1 所示。PNML 文件產生的歐氏記號圖物件便會依這些類別定義。而這 17 項物件有些物件本論文並不會用到，於 3.2.3 節會詳細的說明所使用到的歐氏記號圖物件與階梯圖物件的關係。

表 3.1 PNML 產生的 17 項物件

物件名稱	功能	物件分類說明
Pnml	定義 Pnml 文件	主要構成裝氏圖的物件的類別，包含暫存點、轉移點與方向弧
NetType	定義裝氏圖包含的暫存點、轉移點、方向弧物件	
PlaceType	定義暫存點物件與屬性	
TransitionType	定義轉移點物件與屬性	
ArcType	定義方向弧物件與屬性	
GraphicsType	定義圖形資訊	多個物件會使用到這些類別的內容。如電腦顯示的名稱、物件顯示的座標
NameType	定義名稱	
OffsetType	定義相對座標	
Position	定義物件座標	
HiddenType	定義隱藏資訊	個別物件會使用到這些類別的內容。如暫存點物件的初始浮標數或方向弧物件的權重
InitailMarkingType	定義暫存點初始浮標數	
InscriptionType	定義方向弧權重	
ToolspecificType	定義自訂工具	
PageType	定義頁面	用來定義裝氏圖物件包含的子裝氏圖
ReferencePlaceType	定義暫存點內部子裝氏圖	
ReferenceTransitionType	定義轉移點內部子裝氏圖	
ObjectFactory	物件工廠	XJC 編譯後自動產生的物件

3.3.2 階梯圖的物件組成分析

本節介紹階梯圖有那些物件，以及這些物件的功能。首先使用 XJC 編譯器將 PLCopenXML 綱要[35]產生類別，圖 3.6 是利用 XJC 編譯器將 PLCopenXML 綱要解析。

```
parsing a schema...
compiling a schema...
org\plcopen\xml\tc6_0201\AccessType.java
org\plcopen\xml\tc6_0201\AddData.java
org\plcopen\xml\tc6_0201\AddDataInfo.java
org\plcopen\xml\tc6_0201\Body.java
org\plcopen\xml\tc6_0201\Connection.java
org\plcopen\xml\tc6_0201\ConnectionPointIn.java
org\plcopen\xml\tc6_0201\ConnectionPointOut.java
org\plcopen\xml\tc6_0201\DataType.java
org\plcopen\xml\tc6_0201\EdgeModifierType.java
org\plcopen\xml\tc6_0201\FormattedText.java
org\plcopen\xml\tc6_0201\ObjectFactory.java
org\plcopen\xml\tc6_0201\Position.java
org\plcopen\xml\tc6_0201\PouInstance.java
org\plcopen\xml\tc6_0201\PouType.java
org\plcopen\xml\tc6_0201\Project.java
org\plcopen\xml\tc6_0201\RangeSigned.java
org\plcopen\xml\tc6_0201\RangeUnsigned.java
org\plcopen\xml\tc6_0201\StorageModifierType.java
org\plcopen\xml\tc6_0201\Value.java
org\plcopen\xml\tc6_0201\VarList.java
org\plcopen\xml\tc6_0201\VarListAccess.java
org\plcopen\xml\tc6_0201\VarListConfig.java
org\plcopen\xml\tc6_0201\VarListPlain.java
org\plcopen\xml\tc6_0201\package-info.java
```

圖 3.6 PLCopenXML 綱要經由編譯之物件

本論文在此直接整理產生類別與其功能，整理如表 3.2。在歐氏記號圖物件對應轉換的階梯圖物件會依這些類別定義。其對應轉換的過程在下一小節所介紹。

表 3.2 PLCopenXML 綱要產生的 24 項物件

物件名稱	功能	物件分類說明
AccessType	定義存取型別	構成階梯圖所需物件的類別。最重要的是類別 Body 下的子類別 Contact 與 Coil，是用來定義接點與線圈物件
AddType	定義補充資訊	
AddDataInfo	定義補充資訊的資料內容	
Body	定義階梯圖主體	
DataType	定義資料型別	
FormattedText	定義規格化文件	
Package-Info	套件資訊	
PouInstance	定義 POU 物件	
PouType	定義 POU 型別	
Project	定義專案	
Connection	定義連接線段物件	多個物件會使用到這些類別的內容，如階梯圖物件的座標或物件間連線端點的座標
ConnectionPointIn	定義輸入接點物件	
ConnectionPointOut	定義輸出接點物件	
EdgeModifierType	定義元素的邊資訊	
StorageModifierType	定義線圈儲存指令型別	
Position	定義物件座標	
Value	定義物件的值	
RangeSigned	定義有號數範圍	定義物件的變數
RangeUnsigned	定義無號數範圍	
VarList	宣告變數清單	
VarListAccess	宣告存取變數清單	
VarListConfig	宣告環境相關變數	
VarListPlain	宣告無屬性的變數	
ObjectFactory	物件工廠	XJC 編譯後自動產生的類別

3.3.3 歐氏記號圖與階梯圖的物件對應關係

根據表 3.1 PNML 物件中，歐氏記號圖的基本物件可以分為三大類，分別為 Place、Transition 與 Arc。在這三大類物件中也包含其子物件與屬性，在設計自動轉換程式時，編譯 PNML 文件會將所需的各物件與其屬性擷取出來。而在表 3.2 PLCopenXML 物件中，階梯圖的基本元件主要分為四類，其中包含 Coil、Contact、RightPowerRail 與 LeftPowerRail。在設計自動轉換程式輸出階梯圖文件時，主要針對階梯圖這四大屬性做資訊的轉換，變可完成歐氏記號圖到階梯圖的轉換。

在自動程式轉換設計中歐氏記號圖用到 9 種物件與 10 項屬性，整理如表 3.4，並且說明其屬性之功能。而階梯圖用到 9 種物件與 15 項屬性，整理如表 3.3，並且說明其屬性之功能。

表 3.3 轉換所使用歐氏記號圖之物件

階梯圖物件名稱			屬性	說明
1	Body	Body/Coil	id	線圈名稱
			position	線圈位置
		Body/Contact	id	接點名稱
			position	接點位置
		Body/LeftPowerRail	id	左電軌名稱
			position	左電軌位置
		Body/RightPowerRail	id	右電軌名稱
			position	右電軌位置
2	DataType			定義資料型別
3	Connection		refLocalId	連接名稱
			Position	連接座標
4	ConnectionPointIn		relPosition	連接輸入座標
			localId	連接輸入名稱
5	ConnectionPointOut		relPosition	連接輸出座標
6	Position		position	定義座標
7	Variable		Value	定義值
8	PouType			定義程式結構單元
9	Project			定義專案

表 3.4 轉換所使用階梯圖之物件

歐氏記號圖物件名稱		屬性	說明
1	Pnml		歐氏記號圖架構
2	PlaceType	id	暫存點名稱
3	TransitionType	id	轉移點名稱
4	ArcType	source	來源
		target	目標
5	InitialMarkingType	value	浮標值
6	NetType	id	歐氏記號圖名稱
		type	歐氏記號圖類型
7	NameType	id	名稱
8	GraphicsType	position	座標
9	Position	position	座標

整理好所需用到的歐氏記號圖與階梯圖物件後，接著說明歐氏記號圖物件與階梯圖物件的對應關係。在 2.3.3 節提到使用法則轉換法將歐氏記號圖轉換成階梯圖，故歐氏記號圖與階梯圖的物件可以經由法則矩陣找出其對應的關係。歐氏記號圖與階梯圖的物件對應關係如圖 3.7。首先是法則R_I的物件對應，利用 InitialMarkingType 物件找出初始浮標，建立初始的接點與線圈物件，接著利用 source 與 target 這兩物件建立法則矩陣，最後用法則R_E與R_F產生接點與線圈的物件。如此便完成了歐氏記號圖與階梯圖的物件對應關係。

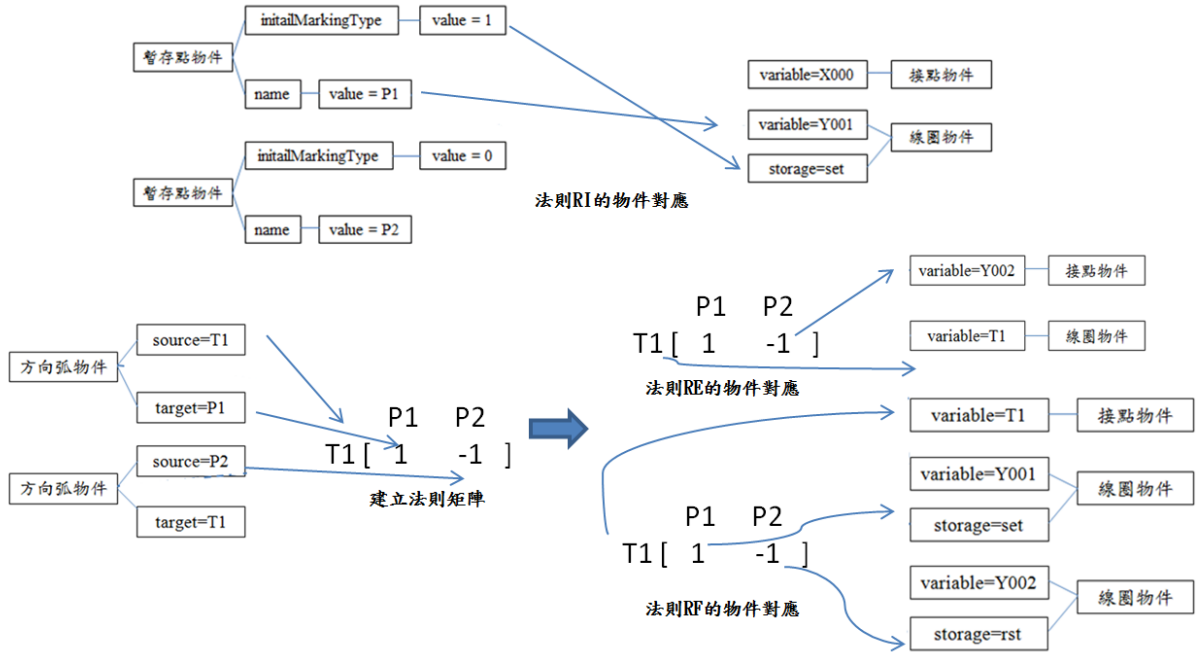


圖 3.7 歐氏記號圖與階梯圖物件對應關係

3.4 浮標守恆裴氏圖到浮標守恆階梯圖轉換兩案例說明

在 3.1 節介紹了浮標守恆裴氏圖到浮標守恆階梯圖的轉換流程，3.2 節介紹了設計同步裴氏圖與同步階梯圖的方法。本節將利用 3.1 與 3.2 節的方法，實際將兩個彈性製造系統之浮標守恆裴氏圖轉換成浮標守恆階梯圖。3.4.1 節為液體加熱系統案例分析。3.4.2 節為自動灌模系統案例分析。

3.4.1 液體加熱系統案例分析

第一個案例以一個液體加熱系統為例子進行轉換[7]，液體加熱系統之浮標守恆裴氏圖如圖 3.8。表 3.5 為暫存點與轉移點意義。根據其暫存點的物理意義分類暫存點屬性，此浮標守恆裴氏圖是屬於可觀測且可直接量測屬性。

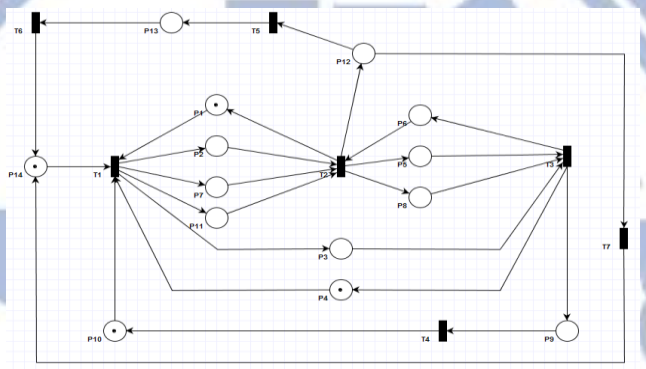
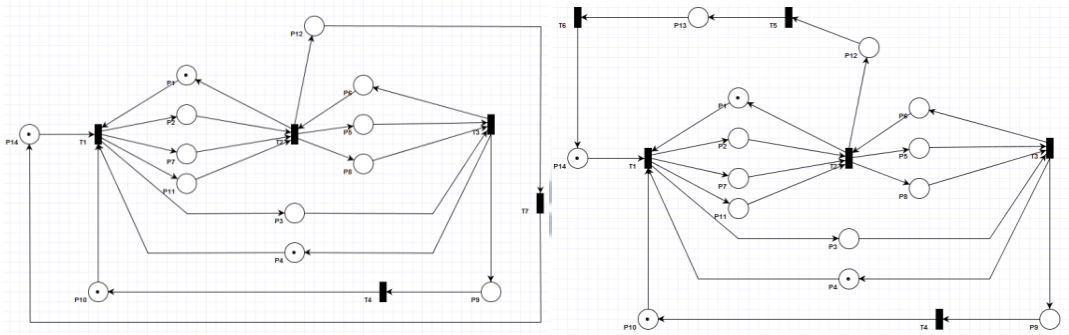


圖 3.8 液體加熱系統之浮標守恆裴氏圖

表 3.5 液體加熱系統之暫存點與轉移點意義

暫存點		轉移點	
P1	閥 1 關閉	T1	批量生產啟動
P2	閥 1 開啟	T2	桶 B 加熱
P3	閥 2 關閉	T3	桶 B 出貨
P4	閥 2 開啟	T4	批量生產完成
P5	加熱設備關閉	T5	桶 A 補料
P6	加熱設備開啟	T6	桶 A 補料完
P7	桶 B 進料	T7	桶 A 備妥
P8	桶 B 加熱		
P9	桶 B 出貨		
P10	桶 B 備妥		
P11	桶 A 出料		
P12	檢查桶 A 庫存		
P13	桶 A 補料		
P14	桶 A 備妥		

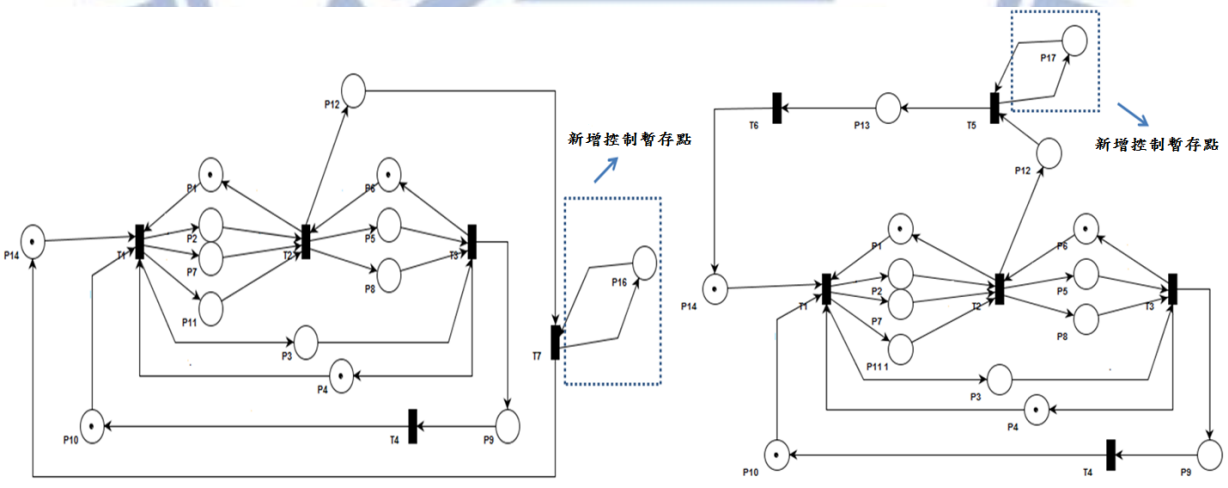
由圖 3.8 與表 3.5 可知 P12 為選項暫存點，在此狀態須判斷桶 A 庫存是否足夠，若足夠則直接進行生產；若不足夠，則桶 A 需要進行補料。因此系統會有隨著桶 A 的庫存而有兩種不同的製程，分別為批量生產與桶 A 補料。透過 2.1.7 節所介紹的時域分解法，此浮標守恆裴氏圖可以分解成兩張歐氏記號圖。如圖 3.9。圖 3.9(a)代表著批量生產的製程，圖 3.9(b)代表著桶 A 補料的製程。



(a)批量生產　　(b)桶 A 補料

圖 3.9 液體加熱系統分解之歐氏記號圖

接著將這兩張歐氏記號圖加入同步裴氏圖，由圖 3.8 可知選項轉移點為 T5 與 T7。於 T5 與 T7 轉移點上新增控制暫存點 P16 與 P17，如圖 3.10。圖 3.10(a)為批量生產之詮釋型歐氏記號圖，P16 為控制暫存點。如欲使用批量生產的製程，P16 則會新增浮標；圖 3.10(b)為桶 A 補料之詮釋型歐氏記號圖，P17 為控制暫存點。如欲使用桶 A 補料的製程，則 P17 便會新增浮標。



(a)批量生產　　(b)桶 A 補料

圖 3.10 合成後之詮釋型歐氏記號圖

轉換成階梯圖主要分成三個步驟，如圖 3.11。第一個步驟為初始狀態設定，將擁有初始浮標的暫存點所對應之線圈開啟。第二個步驟為設定同步階梯圖，將控制暫存點所對應之線圈設為輸出線圈，外部開關為輸入接點。第三步驟為加入歐氏階梯圖，在同步階梯圖下方加入詮釋型歐氏記號圖所轉換的歐氏階梯圖。經過此三個步驟便能完成浮標守恆階梯圖，外部人員或環境便能控制 PLC，依照需求使用不同的製程。

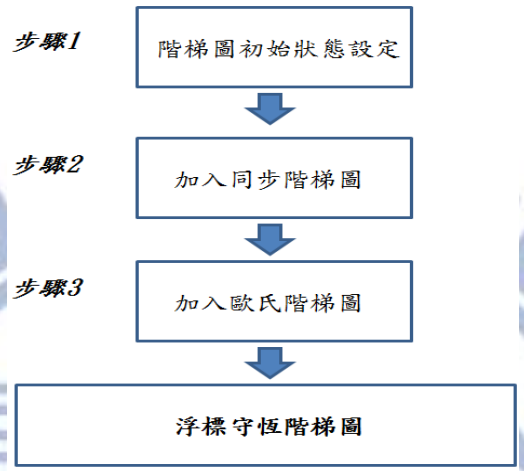


圖 3.11 設計浮標守恆階梯圖三步驟

將批量生產與桶 A 補料之詮釋型歐氏記號圖利用上述三個步驟轉換成階梯圖，轉換結果如圖 3.12。

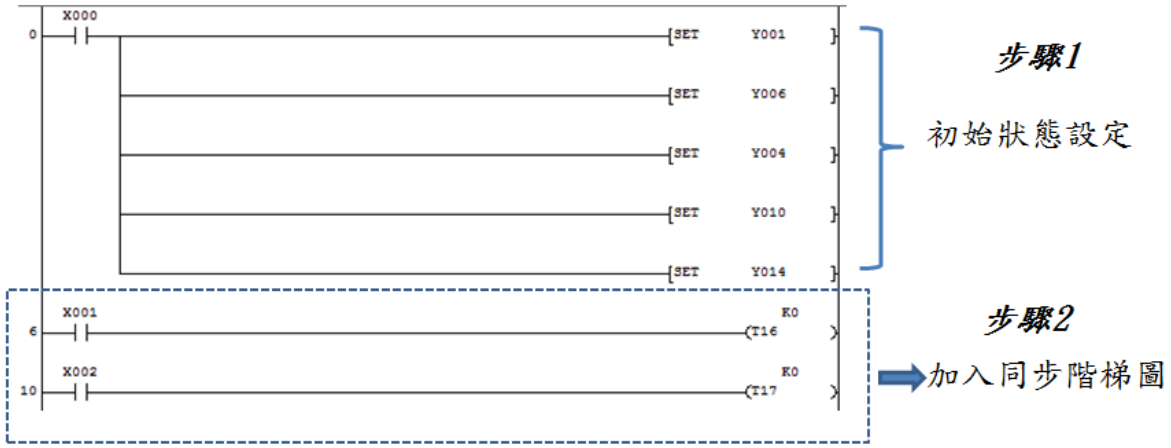


圖 3.12 轉換後之浮標守恆階梯圖

步驟3

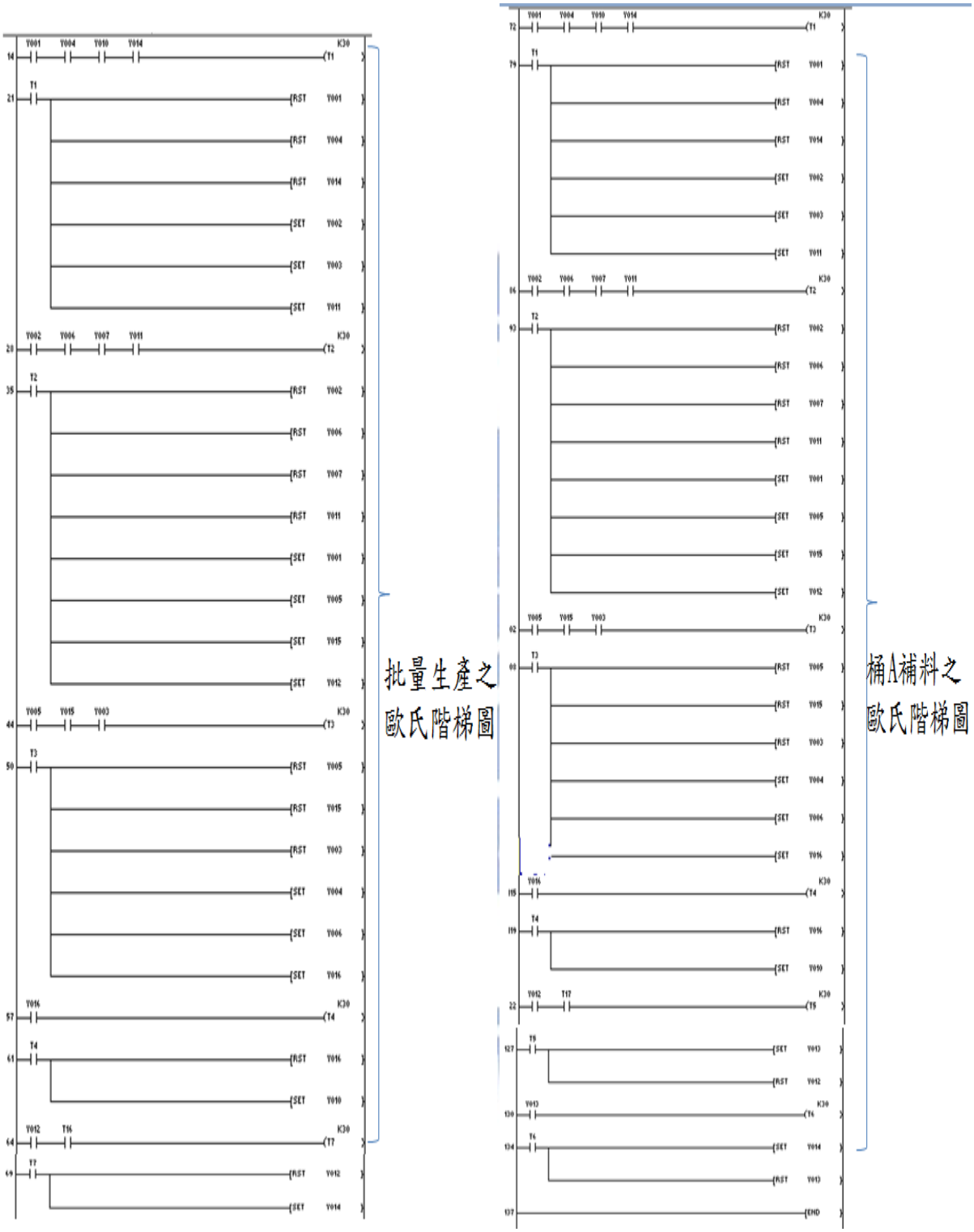


圖 3.13 轉換後之階梯圖

3.4.2 自動灌模系統案例分析

第二個案例是以 2.5 節所介紹的自動灌模系統，圖 3.14 為自動灌模系統之備料系統的浮標守恆裴氏圖。本節會將此浮標守恆裴氏圖利用 2.1.7 節所提到的時域分解法[7]進行分解，分解為多張歐氏記號圖，再將這些歐氏記號圖加入同步裴氏圖，成為詮釋型歐氏記號圖。由於此浮標守恆裴氏圖過於龐大且複雜，用人工的方法難以轉換成所對應之階梯圖。故本論文於 5.2 節會利用自動轉換程式將此浮標守恆裴氏圖轉換成階梯圖。

由圖 3.14 圈選處可知，選項轉移點有兩組：第一組為 T14 與 T15，第二組為 T16 與 T17。T14 與 T15 分別代表使用拉式與推式在槽一調製 Y 物料，故只能選擇一種生產方式生產 Y 物料；T16 與 T17 分別代表使用拉式與推式在槽二調製 X 物料，故只能選擇一種生產方式生產 X 物料。接著使用時域分解法的三個步驟進行分解。

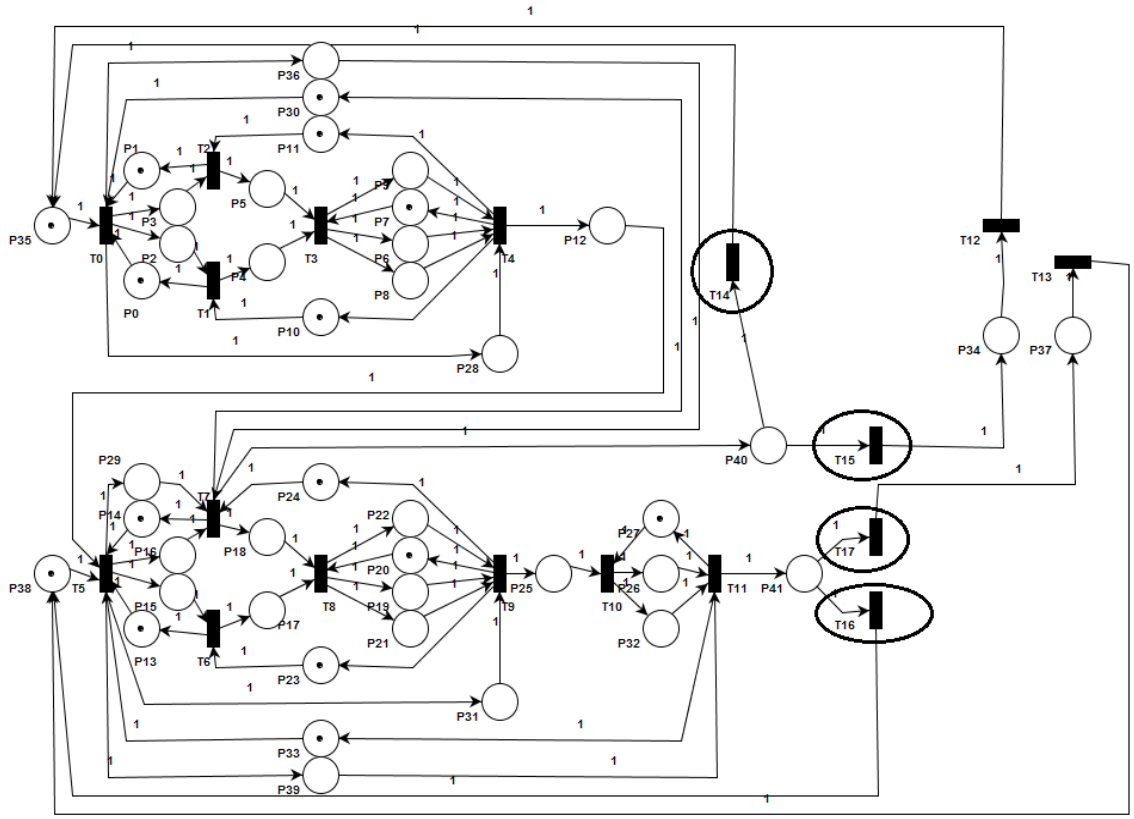
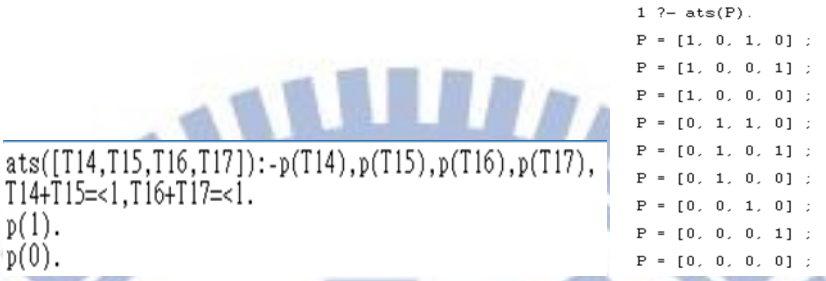


圖 3.14 備料系統之浮標守恆裴氏圖

第一步驟計算可用轉移點集合，利用選項轉移點互斥的特性找出多組可用轉移點集合。透過圖 3.15 左邊的計算可用轉移點程式，可以產生四組最大上限解，即[1,0,1,0]、[1,0,0,1]、[0,1,1,0]及[0,1,0,1]。其結果如圖 3.15 右邊的結果。這四組完整的可用轉移點集合分別為：

(1) [t0,t1,t2,t3,t4,t5,t6,t7,t8,t9,t10,t11,t12,t13,**t14**,**t16**]
(2) [t0,t1,t2,t3,t4,t5,t6,t7,t8,t9,t10,t11,t12,t13,**t14**,**t17**]
(3) [t0,t1,t2,t3,t4,t5,t6,t7,t8,t9,t10,t11,t12,t13,**t15**,**t16**]
(4) [t0,t1,t2,t3,t4,t5,t6,t7,t8,t9,t10,t11,t12,t13,**t15**,**t17**]



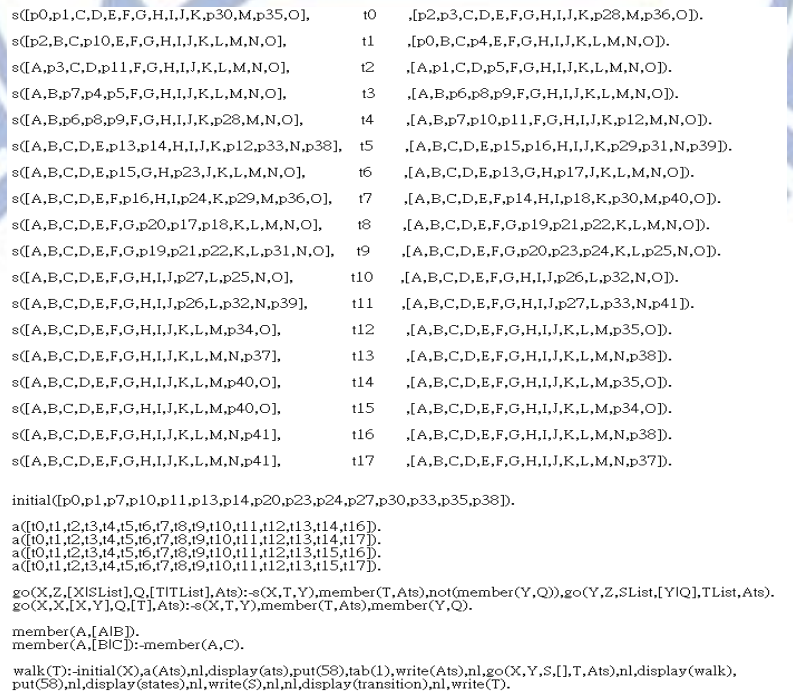
```
ats([T14,T15,T16,T17]):-p(T14),p(T15),p(T16),p(T17),
T14+T15=<1,T16+T17=<1.
p(1).
p(0).
```

```
1 ?- ats(P).
P = [1, 0, 1, 0] ;
P = [1, 0, 0, 1] ;
P = [1, 0, 0, 0] ;
P = [0, 1, 1, 0] ;
P = [0, 1, 0, 1] ;
P = [0, 1, 0, 0] ;
P = [0, 0, 1, 0] ;
P = [0, 0, 0, 1] ;
P = [0, 0, 0, 0] ;
```

圖 3.15 計算可用轉移點程式與輸出結果

第二步驟計算可觸發轉移點集合，利用可用轉移點集合產生的多組繞徑取聯集。透過圖 3.16 計算可觸發轉移點程式，可產生四組可用轉移點集合之繞徑，其結果如圖 3.17。並取多組繞徑轉移點集合的聯集，便可產生可觸發轉移點集合，分別為：

(1) [t0,t1,t2,t3,t4,t5,t6,t7,t8,t9,t10,t11,t14,t16]
(2) [t0,t1,t2,t3,t4,t5,t6,t7,t8,t9,t10,t11,t13,t14,t17]
(3) [t0,t1,t2,t3,t4,t5,t6,t7,t8,t9,t10,t11,t12,t15,t16]
(4) [t0,t1,t2,t3,t4,t5,t6,t7,t8,t9,t10,t11,t12,t13,t15,t17]



```
s([p0,p1,C,D,E,F,G,H,I,J,K,p30,M,p35,O],      t0    ,[p2,p3,C,D,E,F,G,H,I,J,K,p28,M,p36,O]).
s([p2,B,C,p10,E,F,G,H,I,J,K,L,M,N,O],         t1    ,[p0,B,C,p4,E,F,G,H,I,J,K,L,M,N,O]).
s([A,p3,C,D,p11,F,G,H,I,J,K,L,M,N,O],         t2    ,[A,p1,C,D,p5,F,G,H,I,J,K,L,M,N,O]).
s([A,B,p7,p4,p5,F,G,H,I,J,K,L,M,N,O],         t3    ,[A,B,p6,p8,p9,F,G,H,I,J,K,L,M,N,O]).
s([A,B,p6,p8,p9,F,G,H,I,J,K,p28,M,N,O],       t4    ,[A,B,p7,p10,p11,F,G,H,I,J,K,p12,M,N,O]).
s([A,B,C,D,E,p13,p14,H,I,J,K,p12,p33,N,p38],  t5    ,[A,B,C,D,E,p15,p16,H,I,J,K,p29,p31,N,p39]).
s([A,B,C,D,E,p15,G,H,p23,J,K,L,M,N,O],        t6    ,[A,B,C,D,E,p13,G,H,p17,J,K,L,M,N,O]).
s([A,B,C,D,E,F,p16,H,I,p24,K,p29,M,p36,O],    t7    ,[A,B,C,D,E,F,p14,H,I,p18,K,p30,M,p40,O]).
s([A,B,C,D,E,F,G,p20,p17,p18,K,L,M,N,O],      t8    ,[A,B,C,D,E,F,G,p19,p21,p22,K,L,M,N,O]).
s([A,B,C,D,E,F,G,p19,p21,p22,K,L,p31,N,O],    t9    ,[A,B,C,D,E,F,G,p20,p23,p24,K,L,p25,N,O]).
s([A,B,C,D,E,F,G,H,I,J,p27,L,p25,N,O],        t10   ,[A,B,C,D,E,F,G,H,I,J,p26,L,p32,N,O]).
s([A,B,C,D,E,F,G,H,I,J,p26,L,p32,N,p39],      t11   ,[A,B,C,D,E,F,G,H,I,J,p27,L,p33,N,p41]).
s([A,B,C,D,E,F,G,H,I,J,K,L,M,p34,O],          t12   ,[A,B,C,D,E,F,G,H,I,J,K,L,M,p35,O]).
s([A,B,C,D,E,F,G,H,I,J,K,L,M,N,p37],          t13   ,[A,B,C,D,E,F,G,H,I,J,K,L,M,N,p38]).
s([A,B,C,D,E,F,G,H,I,J,K,L,M,p40,O],          t14   ,[A,B,C,D,E,F,G,H,I,J,K,L,M,p35,O]).
s([A,B,C,D,E,F,G,H,I,J,K,L,M,p40,O],          t15   ,[A,B,C,D,E,F,G,H,I,J,K,L,M,p34,O]).
s([A,B,C,D,E,F,G,H,I,J,K,L,M,N,p41],          t16   ,[A,B,C,D,E,F,G,H,I,J,K,L,M,N,p38]).
s([A,B,C,D,E,F,G,H,I,J,K,L,M,N,p41],          t17   ,[A,B,C,D,E,F,G,H,I,J,K,L,M,N,p37]).

initial([p0,p1,p7,p10,p11,p13,p14,p20,p23,p24,p27,p30,p33,p35,p38]).

a([t0,t1,t2,t3,t4,t5,t6,t7,t8,t9,t10,t11,t12,t13,t14,t16]).
a([t0,t1,t2,t3,t4,t5,t6,t7,t8,t9,t10,t11,t12,t13,t14,t17]).
a([t0,t1,t2,t3,t4,t5,t6,t7,t8,t9,t10,t11,t12,t13,t15,t16]).
a([t0,t1,t2,t3,t4,t5,t6,t7,t8,t9,t10,t11,t12,t13,t15,t17]).

go(X,Z,[X|SList],Q,[T|TList],Ats):-s(X,T,Y),member(T,Ats),not(member(Y,Q)),go(Y,Z,SList,[Y|Q],TList,Ats).
go(X,X,[X,Y],Q,[T],Ats):-s(X,T,Y),member(T,Ats),member(Y,Q).

member(A,[A|B]).
member(A,[B|C]):-member(A,C).

walk(T):-initial(X),a(Ats),nl,display(ats),put(58),tab(1),write(Ats),nl,go(X,Y,S,[],T,Ats),nl,display(walk),
put(58),nl,display(states),nl,write(S),nl,nl,display(transition),nl,write(T).
```

圖 3.16 計算可觸發轉移點程式

：

：

以下省略

圖 3.17 可觸發轉移點集合的繞徑畫面

第三步驟產生歐氏記號圖，將原本浮標守恆裝氏圖之法則矩陣刪除不存在於可觸發轉移點集合的轉移點，接著將暫存點整欄為 0 值也刪除，便能成功產生歐氏記號圖。利用這四組可觸發轉移點集合便可產生四張歐氏記號圖。此四張歐氏記號圖的生產方式分別為：(1)槽一槽二皆用拉式生產。(2)槽一使用拉式生產槽二使用推式生產(混合式 A)。(3)槽一使用推式生產槽二使用拉式生產(混合式 B) (4)槽一槽二皆用推式生產。圖 3.18 為拉式備料系統之歐氏記號圖，生產方式為槽一槽二皆用拉式生產。圖 3.19 為混合式(a)備料系統之歐氏記號圖，生產方式為槽一使用拉式生產槽二使用推式生產。圖 3.20 為混合式(b)備料系統之歐氏記號圖，生產方式為槽一使用推式生產槽二使用拉式生產。圖 3.21 為推式備料系統之歐氏記號圖，生產方式為槽一槽二皆用推式生產。

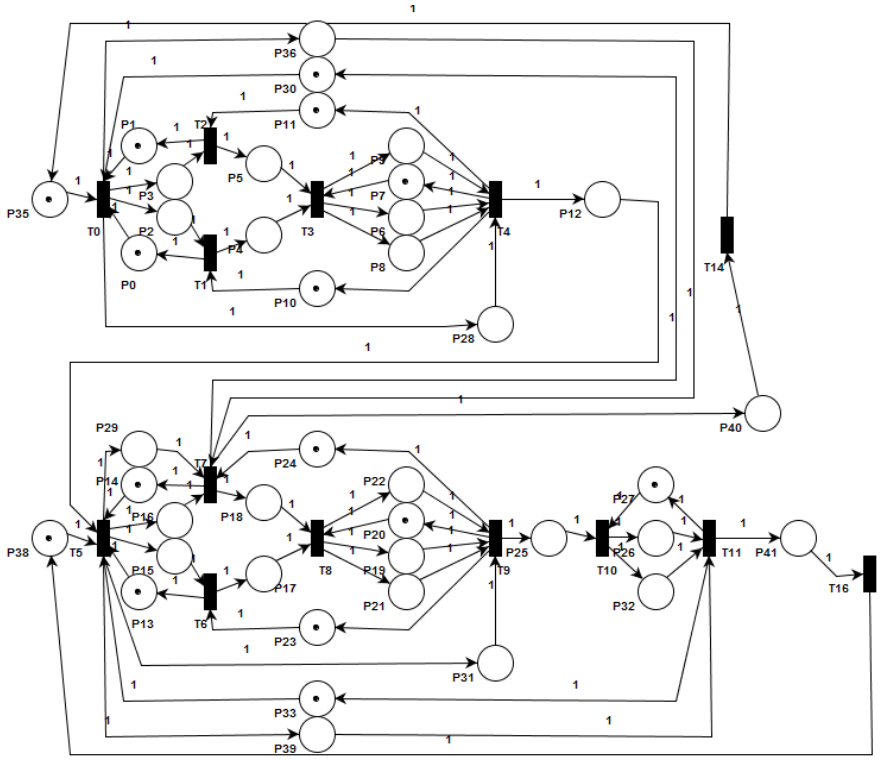


圖 3.18 拉式備料系統之歐氏記號圖

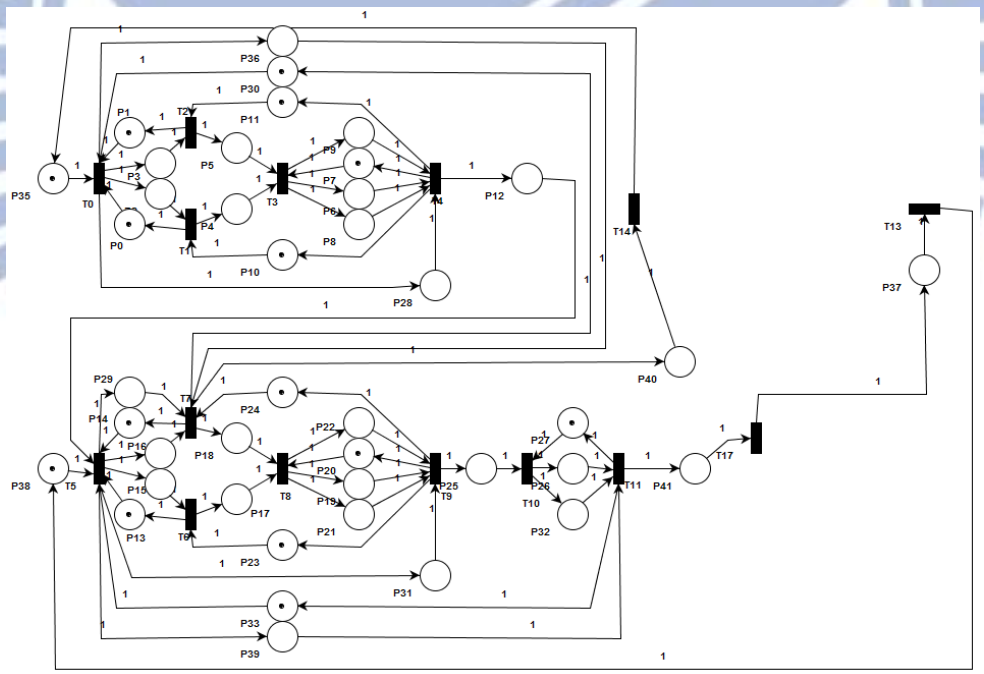


圖 3.19 混合式(a)備料系統之歐氏記號圖

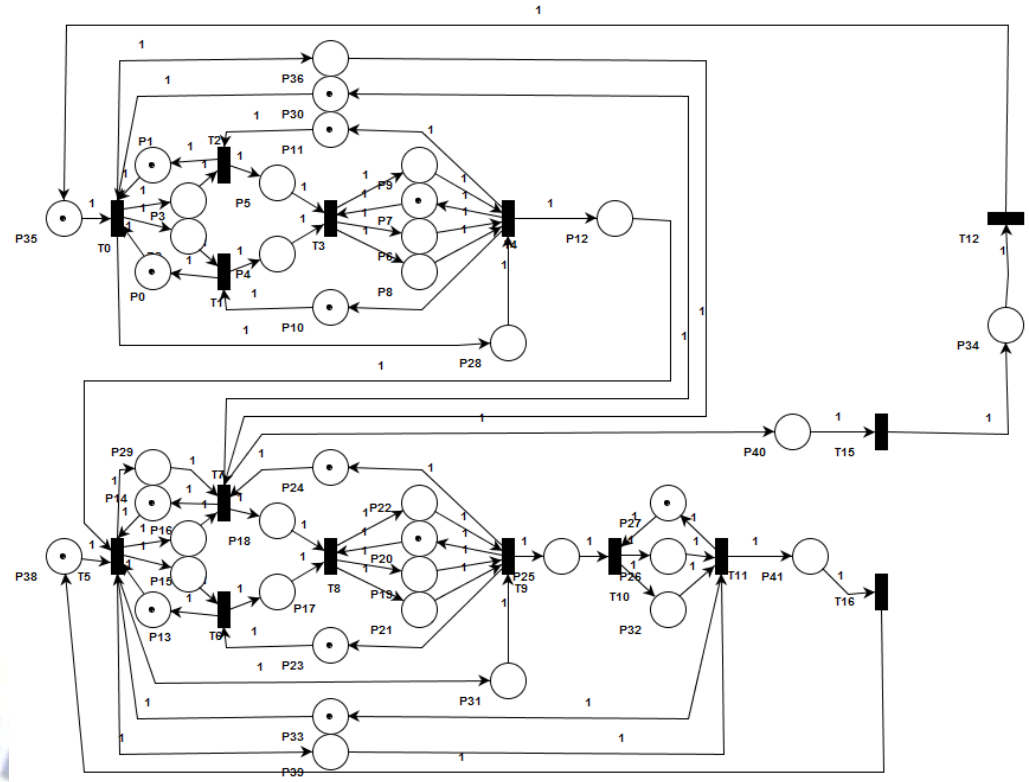


圖 3.20 混合式(b)備料系統之歐氏記號圖

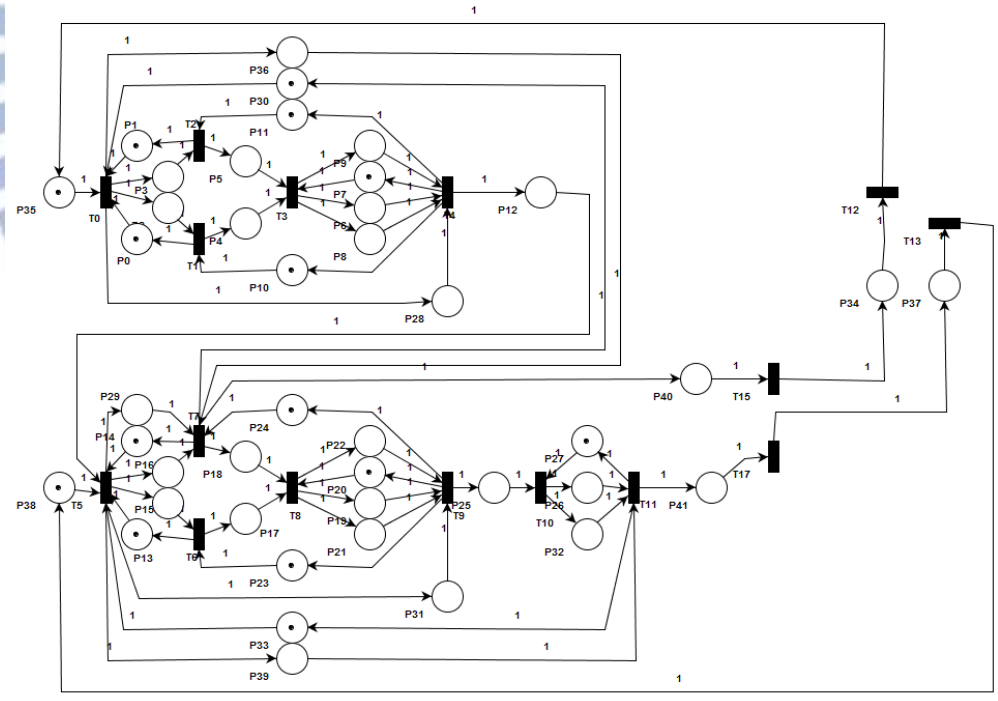


圖 3.21 推式備料系統之歐氏記號圖

產生出四張歐氏記號圖後，最後再利用 3.2 節所提出來的方法加入控制暫存點。以拉式備料系統之歐氏記號圖為例，兩組選項轉移點所取的分別為 T14 與 T16，在這兩個轉移點上加入同一個輸入與輸出的控制暫存點 P42。如圖 3.22 所示。

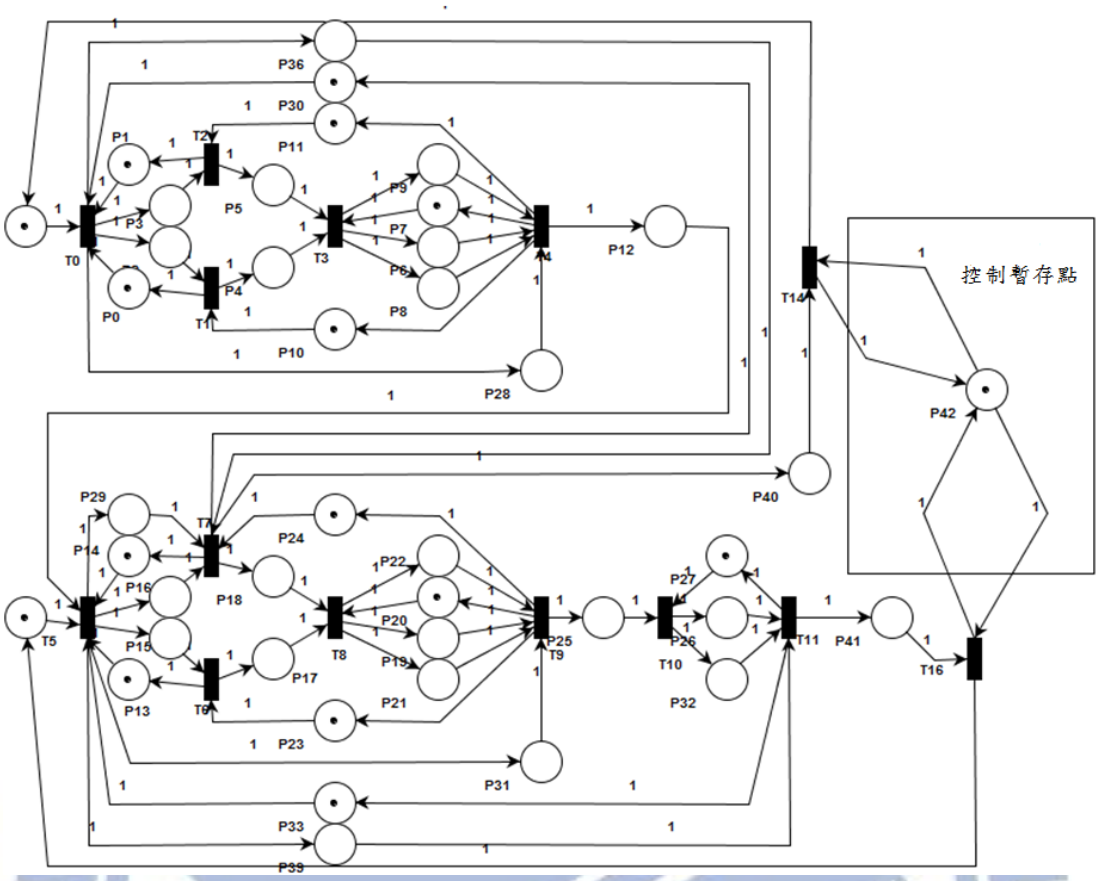


圖 3.22 加入控制暫存點之拉式歐氏記號圖

以圖 3.22 為例，當控制暫存點 P42 擁有浮標時，代表著系統使用拉式備料系統。外部環境可以依情況選擇拉式生產，控制暫存點 P42 便會擁有浮標。使槽一調製 Y 與槽二調製 X 皆使用拉式生產，故當 P40 與 P41 擁有浮標時，則可以觸發 T14 與 T16。如果不使用拉式生產，則消除控制暫存點 P42 的浮標，系統便不會觸發 T14 與 T16 使用拉式生產。故同步階梯圖就是利用同步裴氏圖的特性，利用控制暫存點來控制生產系統。

由於總共有 4 張詮釋型歐氏記號圖，系統狀態多且複雜，難以用人工的方式轉換成階梯圖。故於 5.2 節自動轉換程式案例分析時，會將這些詮釋型歐氏記號圖輸入到自動轉換程式中，接著藉由物件的轉換，自動產生初始階梯圖、同步階梯圖與歐氏階梯圖，最後合成為一張大型的階梯圖。

最後本論文將自動灌模系統之浮標守恆裴氏圖加入同步裴氏圖成為詮釋型裴氏圖。詮釋型裴氏圖代表著可利用外部環境控制此彈性製造系統。以圖 3.23 為例，先前在分解時已知道分解後的四張歐氏記號圖其選項轉移點分別為 T14 與 T16、T14 與 T17 、T15 與 T16 以及 T15 與 T17。設計四個控制暫存點 P50、P51、P52 與 P53 分別連接到 T14 與 T16、T14 與 T17 、T15 與 T16 以及 T15 與 T17。當 P50 有浮標時，則觸發 T14 與 T16，代表槽一與槽二皆使用拉式生產。同理，當 P51 有浮標時，則觸發 T14 與 T17，代表槽一使用拉式生產槽二使用混合式生產。由於控制暫存點 P50 至 P53 同時只會存在一個浮標，故設計一個暫存點 P54 來控制並選擇那個控制暫存點需擁有浮標。當欲使用拉式生產，觸發轉移點 T18 使 P50 擁有浮標。當不使用拉式生產時，觸發轉移點 T20 使 P50 失去浮標。接著又可以在次選擇欲使用的生產方式。

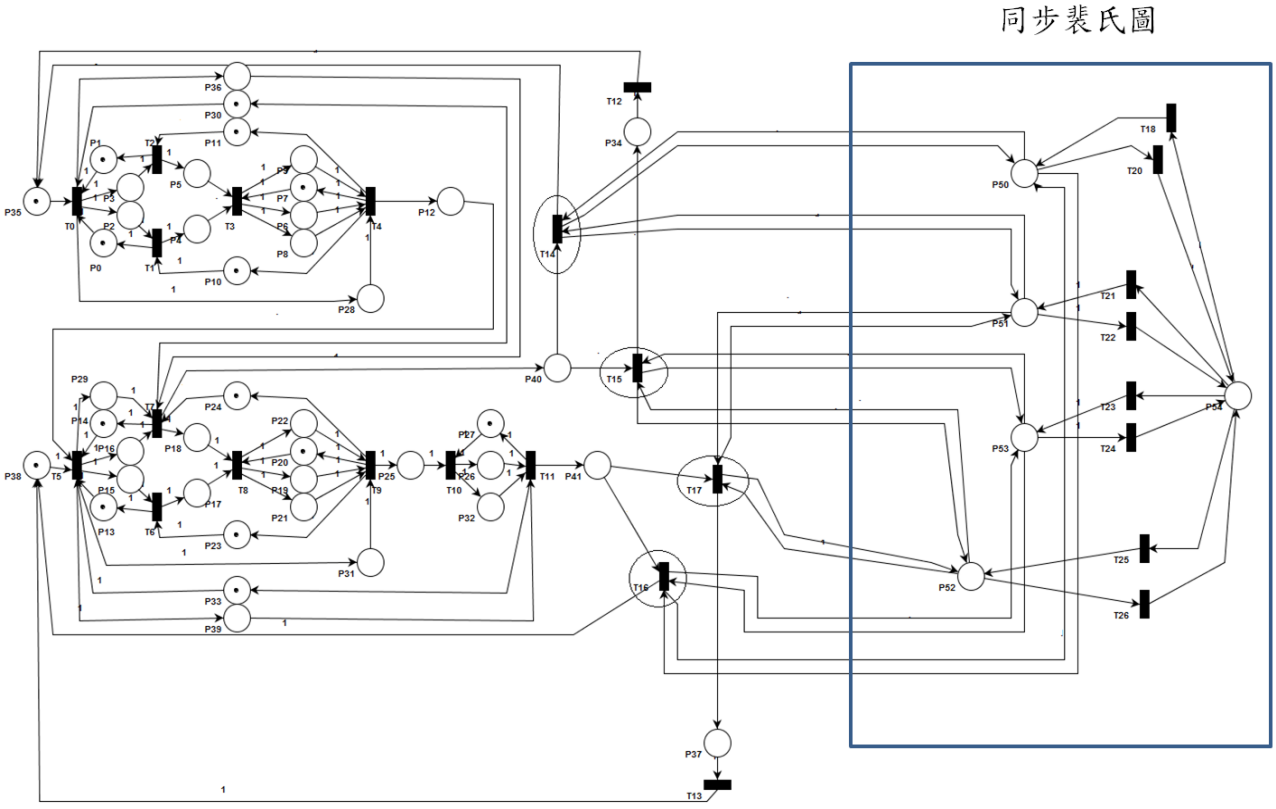


圖 3.23 自動灌模系統之詮釋型裴氏圖

自動灌模系統之詮釋型裴氏圖可以找出其狀態方程式$P' = P + R_I t$，其中R_I為詮釋型裴氏圖之法則矩陣。而詮釋型裴氏圖為浮標守恆裴氏圖與同步裴氏圖的合成，故$R_I = R_T + R_S$。其中R_T為浮標守恆裴氏圖之法則矩陣，R_S為同步裴氏圖之法則矩陣。以下為自動灌模系統之詮釋型裴氏圖的R_I、R_T與R_S。

R_I^T :

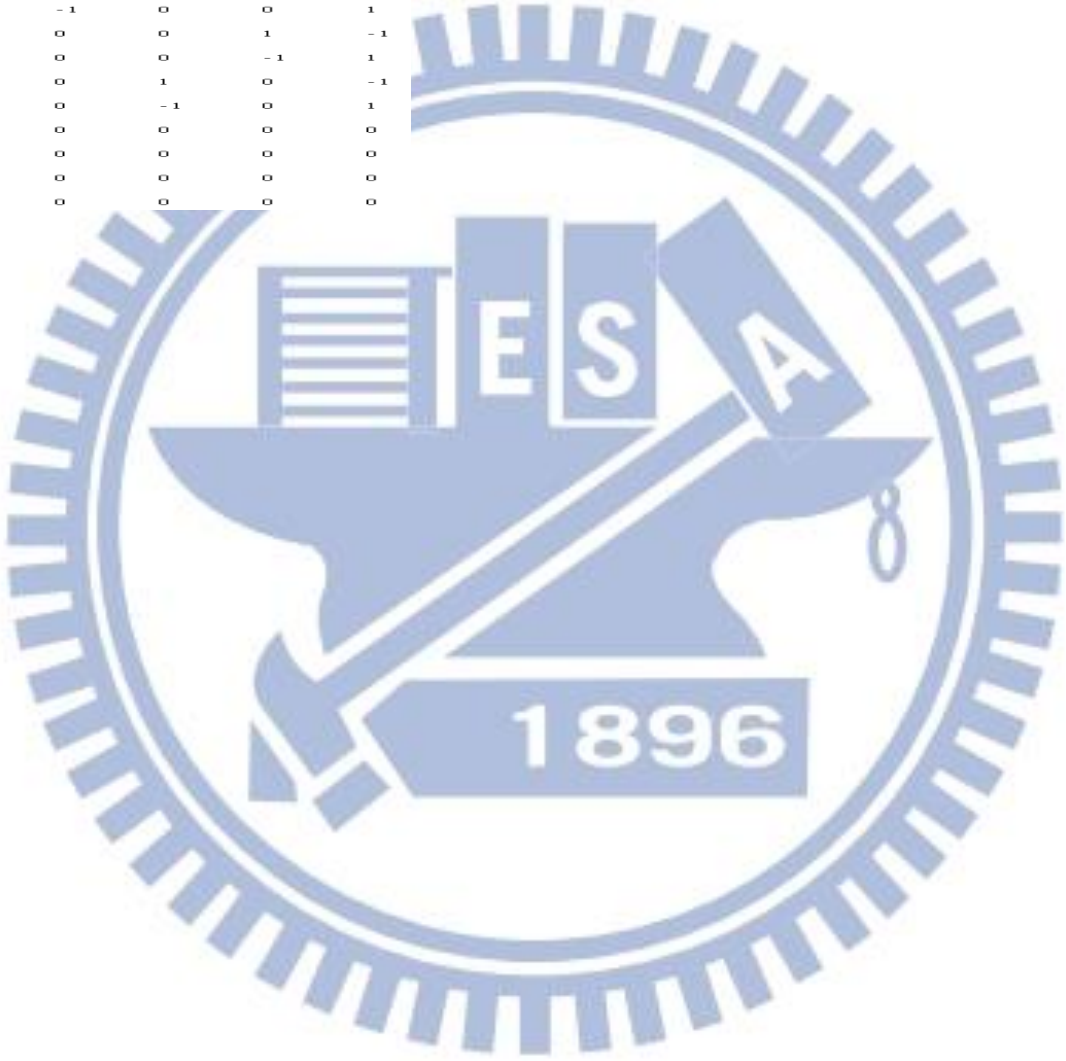
	P0	P1	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19	P2	P20	P21	P22	P23	P24	P27	P28	P29	P3	P31	P32	P33	P34	P35	P37	P38	P39	P4	P40	P41	P30	P36	P25	P26	P5	P50	P51	P52	P53	P6	P7	P8	P9
T0	-1	-1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0	0	0	0	-1	0	0	0	0	0	0	-1	1	0	0	0	0	0	0	0	0	0	0	0
T1	1	0	-1	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
T10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	-1	1	0	0	0	0	0	0	0	0	0
T11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	-1	1	0	0	0	0	-1	0	0	1	0	0	0	-1	0	0	0	0	0	0	0	0	0
T12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
T14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
T15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
T16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
T17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
T13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
T2	0	1	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
T3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	-1	0	0	0	0	1	-1	1	1
T4	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	1	-1	-1
T5	0	0	0	0	-1	-1	-1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	1	0	-1	0	0	0	-1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
T6	0	0	0	0	0	1	0	-1	0	1	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
T7	0	0	0	0	0	0	1	0	-1	0	1	0	0	0	0	0	0	-1	0	0	-1	0	0	0	0	0	0	0	0	0	0	1	0	1	-1	0	0	0	0	0	0	0	0	0	0	0
T8	0	0	0	0	0	0	0	0	0	-1	-1	1	0	-1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
T9	0	0	0	0	0	0	0	0	0	0	0	-1	0	1	-1	-1	1	1	0	0	0	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

R_T^T :

	P0	P1	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19	P2	P20	P21	P22	P23	P24	P27	P28	P29	P3	P31	P32	P33	P34	P35	P37	P38	P39	P4	P40	P4	P30	P36	P25	P26	P5	P6	P7	P8	P9
T0	-1	-1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0	0	0	0	-1	0	0	0	0	0	0	-1	1	0	0	0	0	0	0	0
T1	1	0	-1	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
T10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	-1	1	0	0	0	0	0
T11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	-1	1	0	0	0	0	-1	0	0	1	0	0	0	-1	0	0	0	0	0
T12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
T14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	-1	0	0	0	0	0	0	0	0	0	0	0
T15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	-1	0	0	0	0	0	0	0	0	0	0	0
T16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	-1	0	0	0	0	0	0	0	0	0	0
T17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	-1	0	0	0	0	0	0	0	0	0	0
T13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
T2	0	1	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
T3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	-1	1	-1	1	1
T4	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	1	-1	-1
T5	0	0	0	0	-1	-1	-1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	1	0	-1	0	0	0	-1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
T6	0	0	0	0	0	1	0	-1	0	1	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
T7	0	0	0	0	0	0	1	0	-1	0	1	0	0	0	0	0	0	-1	0	0	-1	0	0	0	0	0	0	0	0	0	0	1	0	1	-1	0	0	0	0	0	0	0
T8	0	0	0	0	0	0	0	0	0	-1	-1	1	0	-1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
T9	0	0	0	0	0	0	0	0	0	0	0	-1	0	1	-1	-1	1	1	0	0	0	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

R_S^T :

PS0	PS1	PS2	PS3	PS4
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
1	0	0	0	-1
0	0	0	0	0
0	0	0	0	0
-1	0	0	0	1
0	1	0	0	-1
0	-1	0	0	1
0	0	0	1	-1
0	0	0	-1	1
0	0	1	0	-1
0	0	-1	0	1
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0



第四章　浮標守恆裴氏圖到階梯圖的轉換實作

本論文在第三章提出建立浮標守恆裴氏圖到階梯圖的轉換方法。但此過程是以人工方式，容易發生人為錯誤。此外當浮標守恆裴氏圖過於龐大時，其時間成本會很高。故本章以 2.4 節的三層式架構轉換法為基礎，發展出浮標守恆裴氏圖到階梯圖的自動轉換程式。本章節首先會說明 PNML、PLCopenXML 與物件的關係，建立物件轉換流程。並說明如何將多張分解後的詮釋型歐氏記號圖，轉換到對應的階梯圖物件。其轉換成階梯圖主要可分成三個部份：分別為初始階梯圖、同步階梯圖與歐氏階梯圖。接著再說明程式所需功能、架構、執行流程與程式碼撰寫，以完成浮標守恆裴氏圖到階梯圖的自動轉換。4.1 節為解標籤：PNML 文件與物件的關係說明。4.2 節為組標籤：物件與 PLCopenXML 的關係說明。4.3 節為物件與初始階梯圖、同步階梯圖與歐氏階梯圖的關係。4.4 節為浮標守恆裴氏圖到階梯圖的轉換流程分析。4.5 節為自動轉換程式之實作。

4.1 解標籤：PNML 文件與物件的關係說明

解標籤(Unmarshaller)是用來將 PNML 文件轉換為代表該 PNML 內容的 JAVA 物件。Unmarshaller 可接受符合綱要規範的 XML 文件，若在 Unmarshaller 的驗證過程中有發生錯誤，則會拋出 UnmarshallEeceptiom。

利用 2.4.3.3 節所介紹在 JAVA 程式裡加入 Unmarshaller 的語法，產生 PNML 文件所對應之物件。為了能更清楚看出 PNML 物件的整體架構，利用軟體 SPY[25]來產生物件的架構圖，如圖 4.1 所示。

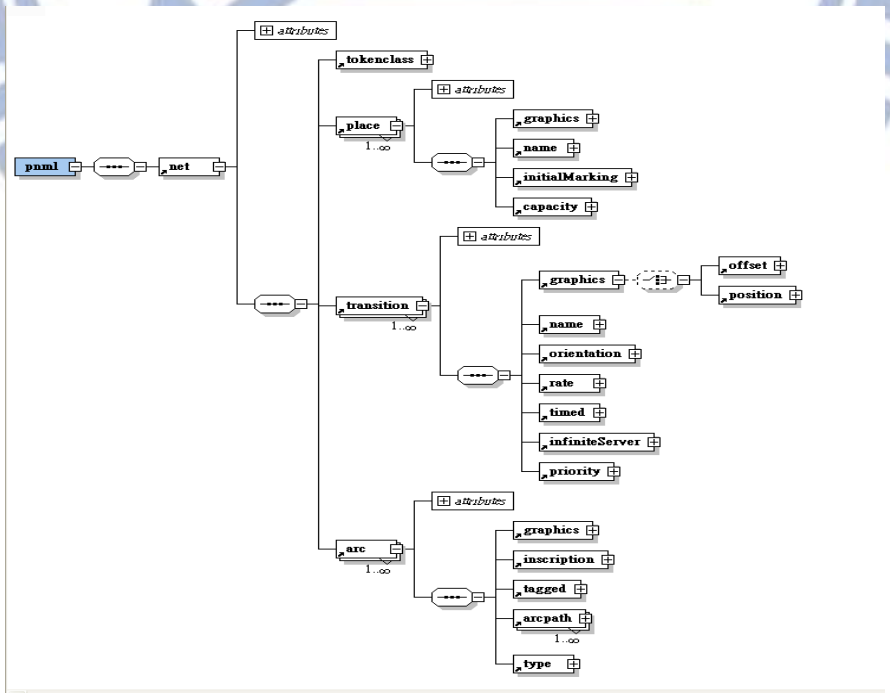


圖 4.1 PNML 物件結構

在<net>物件中共含有三個子物件分別為<place>、<transition>與<arc>。子物件<place>、<transition>與<arc>紀錄暫存點、轉移點與方向弧資訊。整理其物件、物件屬性、屬性說明與 PNML 文件的對應關係，如表 4.1。找出 PNML 文件與物件的對應關係，在設計自動轉換程式時，便能抓取所需的物件，建立物件轉換的方法。

表 4.1 PNML 文件與物件的關係

<table>
<tr><th>物件</th><th>物件</th><th>物件屬性</th><th>屬性說明</th><th>PNML 文件</th></tr>
<tr><td rowspan="4">Place</td><td>Place</td><td>ID</td><td>Place 的 ID</td><td rowspan="4"><place id="P0">
<graphics>
<position x="90.0" y="120.0" />
</graphics>
<name>
<value>P0</value>
<graphics>
<offset x="-5.0" y="35.0" />
</graphics>
</name>
<initialMarking>
<value>Default,1</value>
<graphics>
<offset x="0.0" y="0.0" />
</graphics>
</initialMarking>
<capacity>
<value>0</value>
</capacity>
</place></td></tr>
<tr><td>Name</td><td>Value</td><td>Place 的名稱</td></tr>
<tr><td>Graphic</td><td>Position</td><td>Place 的座標</td></tr>
<tr><td>InitialMarking</td><td>Value</td><td>Place 的初始浮標</td></tr>
<tr><td rowspan="3">Transition</td><td>Transition</td><td>ID</td><td>Transition 的 ID</td><td rowspan="3"><transition id="T0">
<graphics>
<position x="225.0" y="65.0" />
</graphics>
<name>
<value>T0</value>
<graphics>
<offset x="-5.0" y="35.0" />
</graphics>
</name>
<orientation>
<value>0</value>
</orientation>
<rate>
<value>1.0</value>
</rate>
<timed>
<value>false</value>
</timed>
<infiniteServer>
<value>false</value>
</infiniteServer>
<priority>
<value>1</value>
</priority>
</transition></td></tr>
<tr><td>Name</td><td>Value</td><td>Transition 的名稱</td></tr>
<tr><td>Graphic</td><td>Position</td><td>Transition 的座標</td></tr>
<tr><td rowspan="3">Arc</td><td>Arc</td><td>Source</td><td>Arc 的來源</td><td rowspan="3"><arc id="P0 to T0" source="P0" target="T0">
<graphics />
<inscription>
<value>Default,1</value>
<graphics />
</inscription>
<tagged>
<value>false</value>
</tagged>
<arcpath id="000" x="115" y="126" curvePoint="false" />
<arcpath id="001" x="231" y="77" curvePoint="false" />
<type value="normal" />
</arc></td></tr>
<tr><td></td><td>Target</td><td>Arc 的目標</td></tr>
<tr><td></td><td>ID</td><td>Arc 的 ID</td></tr>
</table>

4.2 組標籤：物件與 PLCopenXML 文件的關係說明

在 JAXB 中，組標籤(Marshaller)包括解析 Java 內容樹並寫出一個 XML 文件，此文件是經由 XML 綱要驗證並是有效符合綱要規定。對於階梯圖物件轉換成 PLCopenXML 文件的方法在 2.4.3.4 節有介紹。為了更清楚的了解階梯圖物件的架構，利用 SPY 軟體來產生階梯圖物件的架構圖，如圖 4.2。PLCopenXML 文件中會以<project>作為物件的根物件，而與階梯圖相關的資訊放置在子物件<body>下的<LD>物件。<LD>物件下總共有四個子物件，<Coil>與<Contact>用來記錄接點與線圈資訊，<LeftPowerRail>與<RightPowerRail>紀錄左右電軌的資訊。

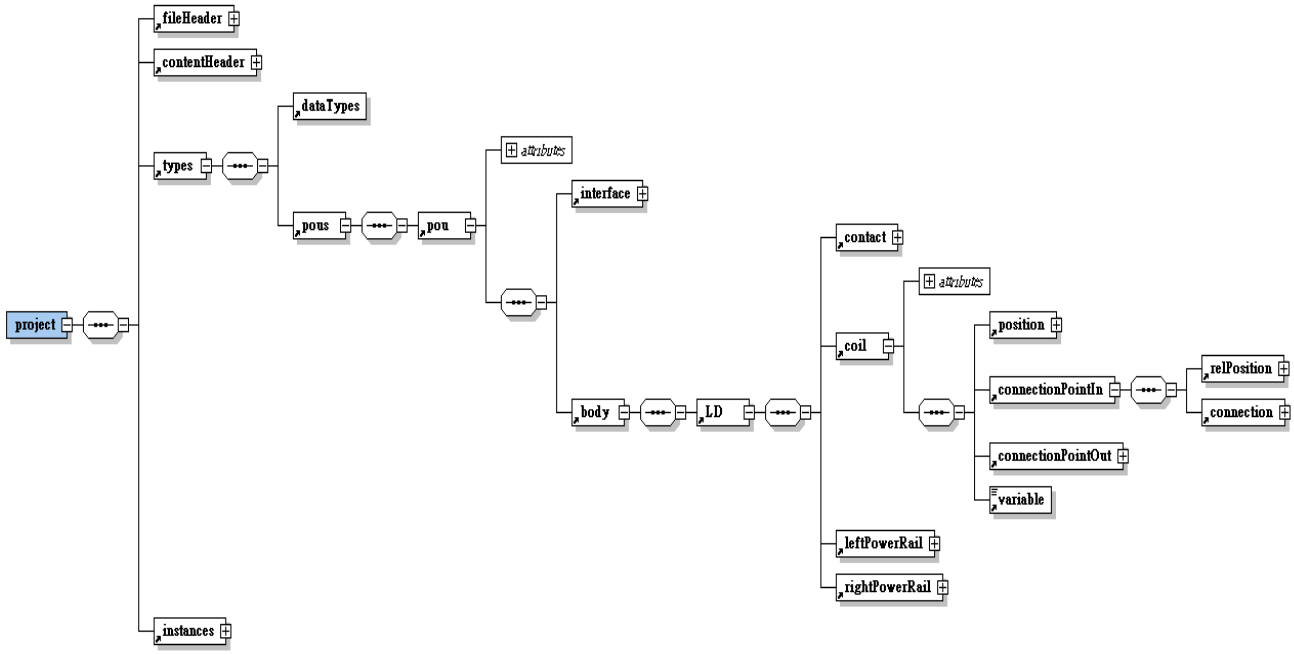


圖 4.2 PLCopenXML 物件結構

根據 PLCopenXML 所需用到的物件、屬性、屬性說明與文件之對應，將這些說明與關係整理成表 4.2。經由表 4.2，可以知道 PLCopenXML 物件與文件的對應關係，在設計自動轉換程式時，便能依轉換方法產生所需的物件，設定物件的屬性，並且轉換成正確的 PLCopenXML 文件。

表 4.2 PLCopenXML 物件與文件的關係

	物件	物件屬性	屬性說明	PLCopenXML 文件
Coil	Coil	localId	ID	<coil storage="set" width="21" height="15" localId="3"> <position y="42" x="450" /> <connectionPointIn> <relPosition y="8" x="0" /> <connection refLocalId="2"> <position y="50" x="450" /> <position y="50" x="121" /> </connection> </connectionPointIn> <connectionPointOut> <relPosition y="8" x="21" /> </connectionPointOut> <variable>Y000</variable> </coil>
		Variable	名稱	
	ConnectionPoinIn	relPosition	相對座標	
		localId	輸入連線 ID	
	ConnectionPoinOut	localId	輸出連線 ID	
	Connection	reflocalId	ID	
		Position	座標	
Contact	Contact	Variable	名稱	<contact width="**21**" height="**15**" localId="**2**"> <position y="**42**" x="**100**" /> <connectionPointIn> <relPosition y="**8**" x="**0**" /> <connection refLocalId="**1**"> <position y="**50**" x="**100**" /> <position y="**50**" x="**71**" /> </connection> </connectionPointIn> <connectionPointOut> <relPosition y="**8**" x="**21**" /> </connectionPointOut> <variable>**X000**</variable> </contact>
		localId	ID	
		Negated	常閉開關	
	ConnectionPoinIn	relPosition	相對座標	
		localId	輸入連線 ID	
	ConnectionPoinOut	localId	輸出連線 ID	
	Connection	reflocalId	ID	
		Position	座標	
LeftPowerRail	LeftPowerRail	localId	ID	<leftPowerRail width="**21**" height="**322**" localId="**1**"> <position y="**30**" x="**50**" /> <connectionPointOut formalParameter=""> <relPosition y="**20**" x="**21**" /> </connectionPointOut> <connectionPointOut formalParameter=""> <relPosition y="**60**" x="**21**" /> </connectionPointOut> <connectionPointOut formalParameter=""> <relPosition y="**100**" x="**21**" /> </connectionPointOut> <connectionPointOut formalParameter=""> <relPosition y="**180**" x="**21**" /> </connectionPointOut> <connectionPointOut formalParameter=""> <relPosition y="**220**" x="**21**" /> </connectionPointOut> </leftPowerRail>
		Position	座標	
	ConnectionPoinOut	relPosition	相對座標	
RightPowerRail	RightPowerRail	localId	ID	<rightPowerRail width="**21**" height="**322**" localId="**14**"> <position y="**30**" x="**500**" /> <connectionPointIn> <relPosition y="**20**" x="**0**" /> <connection refLocalId="**3**"> <position y="**50**" x="**500**" /> <position y="**50**" x="**471**" /> </connection> </connectionPointIn> <connectionPointIn> <relPosition y="**60**" x="**0**" /> <connection refLocalId="**5**"> <position y="**90**" x="**500**" /> <position y="**90**" x="**471**" /> </connection> </connectionPointIn>
		Position	座標	
	ConnectionPoinIn	relPosition	相對座標	
	Connection	reflocalId	ID	
		Position	座標	

4.3 物件與初始階梯圖、同步階梯圖與歐氏階梯圖的關係

本節說明程式如何由加入了同步裴氏圖之歐氏記號圖物件對應轉換到階梯圖物件。也就是程式透過詮釋型歐氏記號圖物件的屬性，搭配 2.3.3 節所介紹的法則轉換法，產生對應的階梯圖物件。程式產生階梯圖物件的方法是由 2.4.3.4 節所提到的物件 Objectfactory 所產生。而對應的意思是指歐氏記號圖與階梯圖物件的名稱互相對應。但在其他屬性如位置，元件大小等則沒有對應關係。而對應到階梯圖物件主要分為三個部份：第一部份為對應到初始階梯圖物件。第二部份為對應到同步階梯圖物件。第三部份為對應到歐氏階梯圖物件。

對於產生初始階梯圖物件來說，是以 2.3.2 節所介紹的法則R_I進行物件轉換。首先程式用 Objectfactory 產生接點物件並命名為 X000，此接點表示系統開啟。在 2.4.3.3 節有提到暫存點物件的 initailMarking 屬性中的 value 是用來表示暫存點的初始浮標數。因此如果暫存點物件的 initailMarkinge 中的 value = 1 時，則程式會產生線圈物件。並設定名稱屬性 variable 為對應的初始暫存點名稱，與 0 節說明的 storage 屬性為 set 並與接點 X000 連接。以圖 4.3 為例，若歐氏記號圖的暫存點 P1 和 P3 的屬性 initailMarkingType 中的 value = 1 時，則程式產生線圈物件並設定對應名稱。在本論文中，名稱對應方式是如果暫存點名稱為 P1，則階梯圖物件就是以 Y001 進行命名。其方式可見圖 4.3 中的虛線對應。最後線圈物件再設定屬性 storage 為 set 並與接點 X000 連接，完成初始階梯圖的物件轉換。

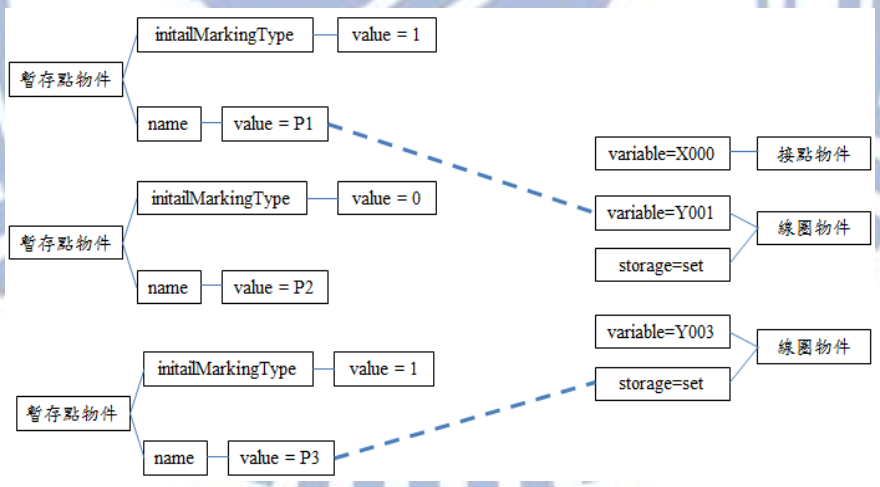


圖 4.3 初始階梯圖物件對應轉換

對於產生同步階梯圖物件與歐氏階梯圖物件來說，是以法則轉換法為基楚，故需利用加入了同步裴氏圖之歐氏記號圖物件來產生法則矩陣。先建立每列的值，也就是程式依序讀取轉移點物件，透過讀取到的轉移點名稱屬性，比對每個方向弧中的屬性 source 和 target 是否有此名稱。在 2.4.2.2 節有提到方向弧的屬性 source 和 target 是表示轉移點的輸出暫存點與輸入暫存點。故若轉移點出現在 source，則在 target 的暫存點就是輸出暫存點。在法則矩陣中，此轉移點和其輸入暫存點的交叉值就為-1，反之為 1。以圖 4.4 為例，若讀取到轉移點 T1，則在那一列設定 T1 的值。接著比對每個方向弧的屬性 source 和 target，比對到第一個方向弧物件的 source 為 T1。此方向弧物件的 target 為 P1，P1 為 T1 之輸出暫存點，故 T1 列與 P1 行的交叉值設定為 1。同理第二個方向弧物件的 target 為 T1，其 sourse 為 P2，P2 為 T1 之輸入暫存點，故 T1 列與 P2 行的交叉值設定為-1。透過這樣的設定方式，依序讀取每個轉移點的物件，就可以建立出法則矩陣。

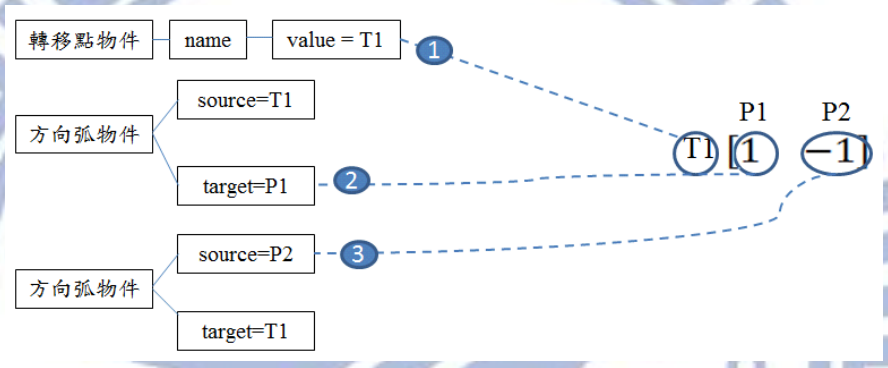


圖 4.4 以歐氏記號圖物件建立法則矩陣

由於在自動轉換程式中，需要輸入多張歐氏記號圖。因此也會產生多個法則矩陣。為了讓程式方便判讀所有法則矩陣，本論文將多張法則矩陣進行合併。以圖 4.5 為例，這是兩張法則矩陣合併的案例。在實線框選的地方為第一張法則矩陣，而虛線框選的地方為第二張法則矩陣。用對角線合併的方向將法則矩陣合併，其他空白的地方記為 0。如此便能只用一張大張的法則矩陣來轉換成同步階梯圖與歐氏階梯圖的物件。

	P0	P11	P2	P0	P22	P3
T1	1	-1	0	0	0	0
T11	-1	1	-1	0	0	0
T1	0	0	0	1	-1	0
T22	0	0	0	-1	1	-1

圖 4.5 合併法則矩陣

對於產生同步階梯圖物件來說，需利用到前面所合併的法則矩陣進行轉換。而程式要如何判斷同步裴氏圖裡的控制暫存點名稱是很重要事情。因此利用法則矩陣可以很容易的找出同步裴氏圖裡的控制暫存點，設定同步階梯圖。主要是利用到 2.1.3 節所介紹歐氏記號的性質，歐氏記號圖的轉移點皆有相同個數的輸入轉移點與輸出轉移點。故法則矩陣中每行的合必為 0。而同步裴氏圖裡的控制暫存點為額外增加的一個輸入暫存點，所以在法則矩陣之中該行的合為-1。程式只要讀取法則矩陣，發現該行合為-1，則此暫存點必為同步裴氏圖的控制暫存點，並將其名稱轉為同步階梯圖物件。以圖 4.6 為例，在法則矩陣中 P2 行的合為-1，故 P2 為控制暫存點。程式首先建立同步階梯圖的接點物件 X001 與 X002，接著將 P2 轉為線圈物件 Y002。如此一來便成功的利用法則矩陣轉換為同步階梯圖物件。

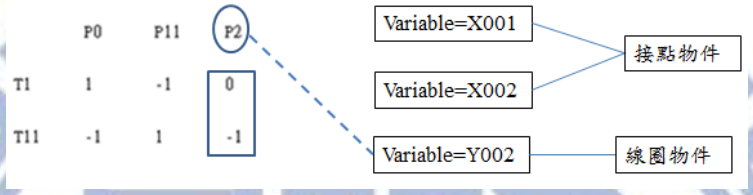


圖 4.6 以法則矩陣建立同步裴氏圖物件

對於產生歐氏階梯圖來說，也是需要利用法則矩陣來進行轉換。並且會用到在 2.3.2 節所介紹的法則R_E與法則R_F產生歐氏階梯圖物件。在法則R_E的物件轉換上，程式讀取到列後，再讀取值為-1 的行，表示該行的暫存點為輸入暫存點。讀取到輸入暫存點後，程式產生接點物件並設定名稱為此輸入暫存點。讀取完所有值為-1 的行時，程式產生線圈物件並設定名稱為此列的轉移點。以圖 4.7 為例，程式首先讀取 T1 列，並讀取值為-1 的 P2 行。故產生接點物件並設定對應名稱為 Y002。程式再產生線圈並設定名稱為 T1。這樣便完成了轉移點 T1 的法則R_E的物件轉換。

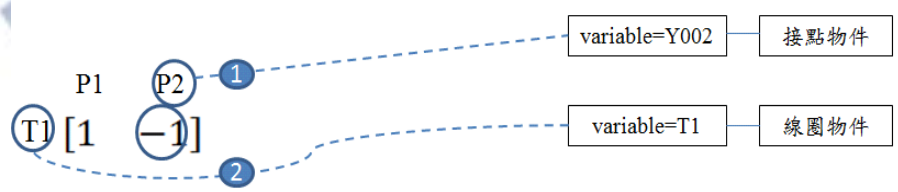


圖 4.7 法則R_E的物件對應轉換

接著再以相同列的轉移點進行法則R_F的物件轉換。程式產生接點物件，並設定名稱為此列的轉移點。接著程式讀取值為 1 或-1 值的行，若為 1，則產生線圈物件並設定名稱為該行的暫存點，且設定 storage 屬性為 set；若為-1，則產生線圈物件並設定名稱為該行的暫存點，且設定 storage 屬性為 rst。以圖 4.8 為例，首先程式產生接點物件且名稱對應到 T1。接著程式讀取值為 1 或-1 的行，先讀取到 P1 行，因此產生線圈物件並設定名稱為 Y001。因為值為 1，故設定 storage 屬性為 set。接著讀取到 P2 行，故產生線圈物件並設定名稱為 Y002。因為值為-1，故設定 storage 屬性為 set。這樣便完成了 T1 的法則R_F的物件轉換。

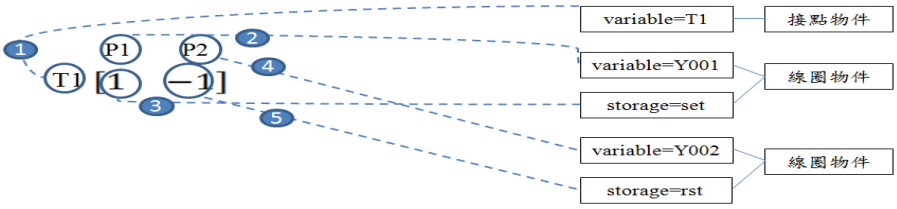


圖 4.8 法則R_F的物件對應轉換

程式依照順序讀取每列法則矩陣並重覆法則R_E與法則R_F的物件轉換後，便完成產生歐氏階梯圖物件的流程。接著以圖 4.9 至圖 4.11 介紹初始階梯圖、同步階梯圖與歐氏階梯圖的轉換流程。圖 4.9 為初始階梯圖建立流程。圖 4.10 為同步階梯圖建立流程。圖 4.11 為歐氏階梯圖建立流程。若法則矩陣為m × n的矩陣。設 i 為法則矩陣第 i 列，j 為法則矩陣第 j 行。P_j為初始暫存點向量中的值；R_{ij}為法則矩陣中轉移點T_i列和暫存點P_j行交叉的值。本論文將在 4.5 節說明如何設計此流程圖的程式碼。

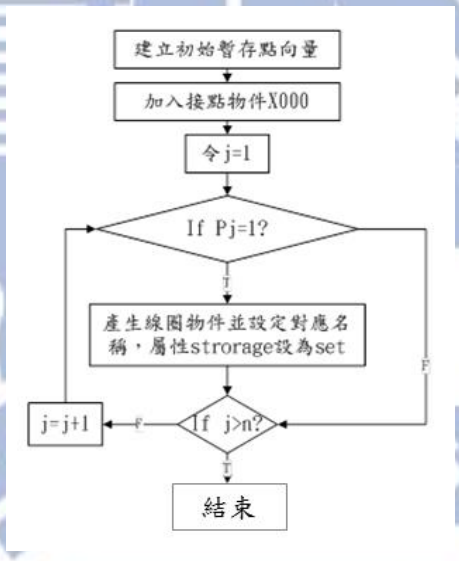


圖 4.9 初始階梯圖建立流程

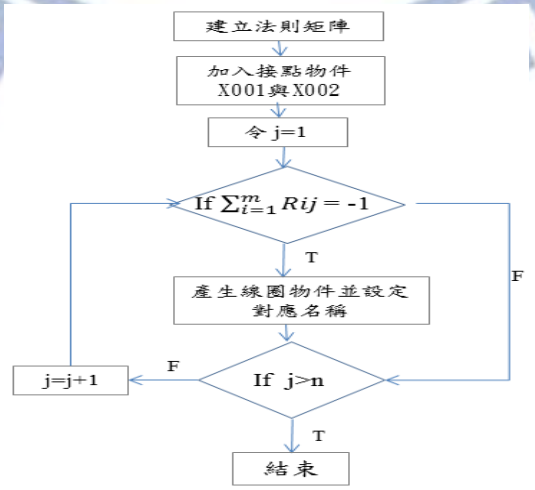


圖 4.10 同步階梯圖建立流程

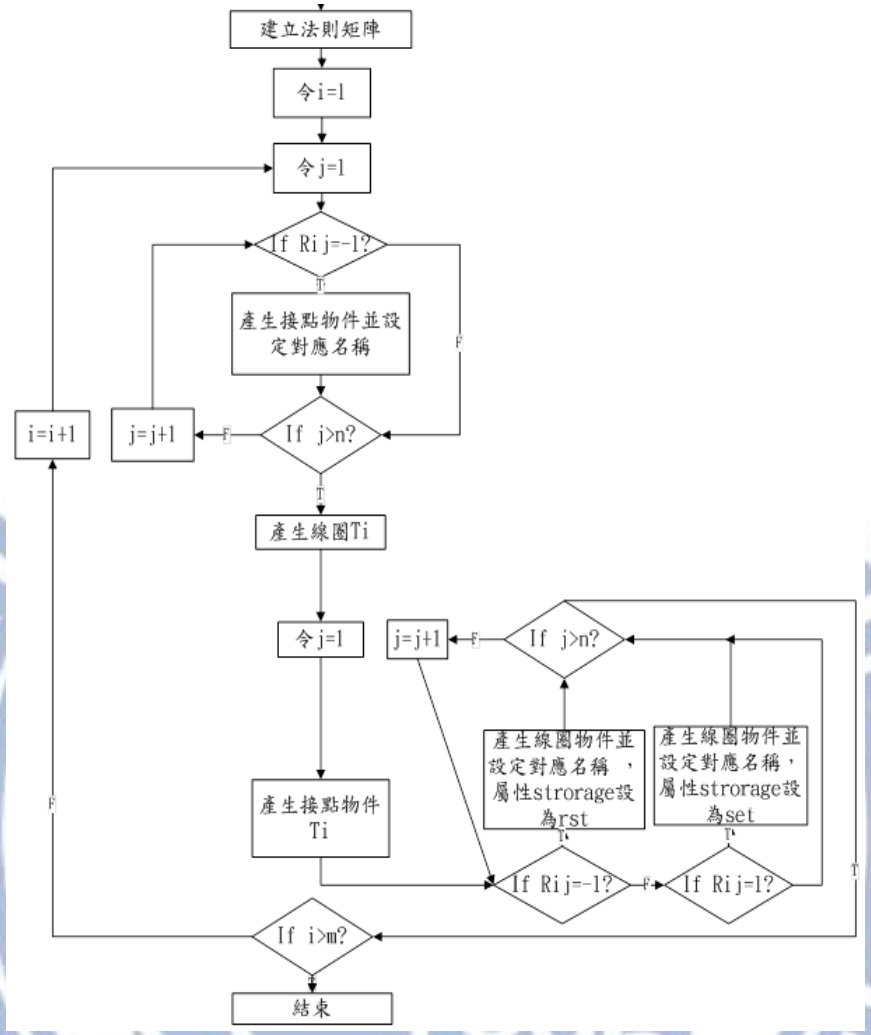


圖 4.11 歐氏階梯圖建立流程

4.4 浮標守恆裴氏圖到階梯圖的轉換流程分析

在進行程式實作前，本節先分析浮標守恆裴氏圖到階梯圖的自動轉換流程，並確定程式運行所需的功能及結果。首先製造系統的管理人員將代表彈性製造系統的浮標守恆裴氏圖經由時域分解法分解後，產生 N 張歐氏記號圖。再將 N 張歐氏記號圖加入控制暫存點後，透過 2.4.2.2 節所提的軟體 PIPE 將這些歐氏記號圖轉為 PNML 文件。執行此自動轉換程式時，需載入這些 PNML 文件，程式首先會對 PNML 的文件進行驗證。驗證內容包含三件事情：(1)檔案規格是否為.pnml 或.xml。若不是屬於這兩種文件的檔案，則程式就必須告知管理人員檔案規格有誤。(2)驗證歐氏記號圖是否有初始浮標。由於轉換為階梯圖時，是以法則R_I來設定系統的初始狀態。故若轉換的 PNML 文件沒有設定初始浮標時，則程式必須告知管理人員須設定初始浮標。

驗證 PNML 文件規格與內容正確後，程式整理這些文件包含的暫存點、轉移點與方向弧的資訊，並建立法則矩陣。再將法則矩陣進行合併。合併後程式會先展示此 PNML 文件的內容與法則矩陣，點擊按鈕後，程式就開始將文件轉換為初始階梯圖、同步階梯圖與歐氏階梯圖，如圖 4.12 圈選處所示。過程如 4.3 節所述，而詳細的程式碼將在 4.5 節介紹。由於階梯圖是以 PLCopenEditor 軟體展示，故轉換後的階梯圖要以 PLCopenXML 文件匯出。因此程式提供功能讓管理人員能選擇匯出檔案儲存的位置。最後管理人員將 PLCopenXML 文件匯入軟體 PLCopenEditor 展示轉換後的階梯圖。最後可利用轉換後的階梯圖，直接控制彈性製造系統。整體流程如圖 4.12 所示。

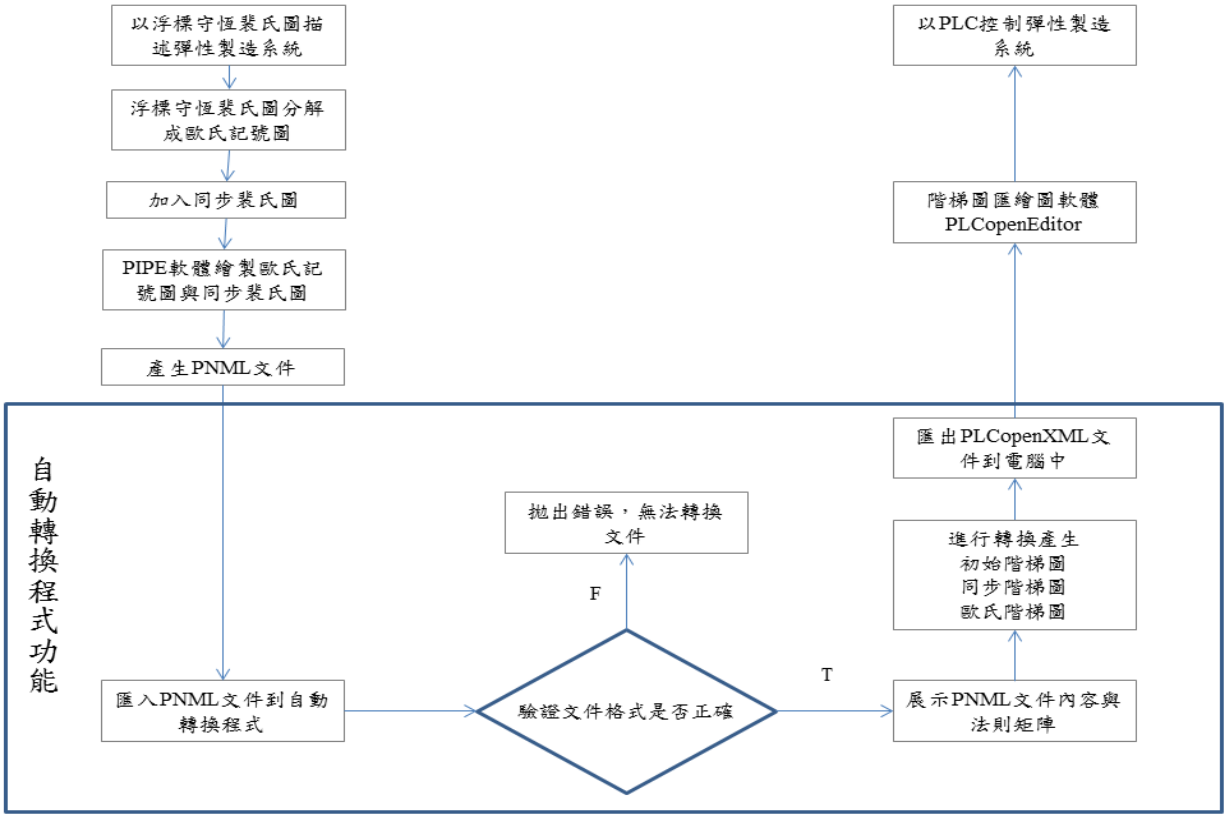


圖 4.12 浮標守恆裴氏圖到階梯圖的自動轉換流程

4.5 自動轉換程式之實作

本節說明本論文所設計的自動化轉換程式功能與執行的流程。4.5.1 節說明程式架構，也就是程式的主要套件(Package)。4.5.2 節說明物件對應轉換的程式碼如何設計。

4.5.1 程式架構

本論文的自動轉換程式是以 Java 程式開發。圖 4.12 所框選的部分為本程式的執行流程，與擁有的功能。透過此流程，本論文將自動轉換程式分為五個套件：(1)mainFile，(2) transformTools，(3) mappedClassFromPNML，(4) mappedClassFromPLCopenXML，(5) GUI。各套件功能整理如表 4.3。

表 4.3 自動轉換 Java 程式套件功能

套件名稱	套件功能
mainFile	主執行程式套件
transformTools	轉換功能套件
mappedClassFromPNML	存放 PNML 綱要類別
mappedClassFromPLCopenXML	存放 PLCopenXML 綱要類別
GUI	使用者介面

對於套件 mainFile 而言，此套件包含方法 main()，表示 Java 程式的進入點。此套件功能是掌管程式執行的流程，也就是圖 4.12 所框選部分的流程。

對於套件 transformTools 而言，此套件包含兩個類別 EMG 與 EMG2LD，提供執行過程中所需要的方法。EMG 提供的是解編 PNML 文件和把解編後的物件分類，並建立出法則矩陣的方法；EMG2LD 提供的是轉換過程的程式碼。內部方法整理如表 4.4。

對於 mappedClassFromPNML 而言，其為經由 JAXB 所產生的 PNML 綱要所存放的位置，此套件中共有 17 項套件，其名稱與內容表 3.1 有詳細說明。

對於 mappedClassFromPLCopenXML 而言，其為經由 JAXB 所產生的 PLCopenXML 綱要所存放的位置，此套件中共有 24 項套件，其名稱與內容表 3.2 有詳細說明。

對於 GUI 而言，這主要是用來建置程式之使用者介面，負責管理程式之畫面呈現與處理使用者間的互動。使用者可利用滑鼠點擊及滑鼠選擇對介面中的元件進行動作，GUI 套件會將這些動作是為一次事件的觸發，依照不同觸發的事件呼叫預先設定的功能進入作業程序，事件完成後將結果反應於使用者介面中。

表 4.4 類別 EMG 與 EMG2LD 方法說明

類別 EMG	方法說明	類別 EMG2LD	方法說明
unMarshall()	解編 PNML 文件到歐氏記號圖物件。並將解編後的物件分別以 place、transition 與 arc 三個清單式資料結構儲存	Marshall()	將轉換後的階梯圖物件組編為 PLCopenXML 文件
getPlace()	回傳暫存點物件資訊	transform()	物件轉換
getTransition()	回傳轉移點物件資訊	generateContact()	產生接點物件
getArc()	回傳方向弧物件資訊	generateCoil()	產生線圈物件
setRuleMatrix()	建立法則矩陣陣列物件	getXMLString()	將文件顯示在使用者介面
getRuleMatrix()	回傳法則矩陣陣列物件		
checkEMG()	透過法則矩陣驗證此文件是否為歐氏記號圖		
checkInitailTokens()	透過暫存點物件資訊驗證此文件是否有設定初始浮標		
getXMLString()	將文件資訊顯示在使用者介面		

最後利用統一塑模語言(Unified Modeling Language, UML)[24]中的類別圖(Class Diagram)[19]表示這些套件類別間的關係，如圖 4.13。由圖 4.13 可看出 Main 類別是繼承自 GUI 類別，並實作當中的抽象方法。因為 Main 類別在執行時，必須使用到 EMG 與 EMG2LD 類別中的方法，故具有依存(Dependency)的關係。

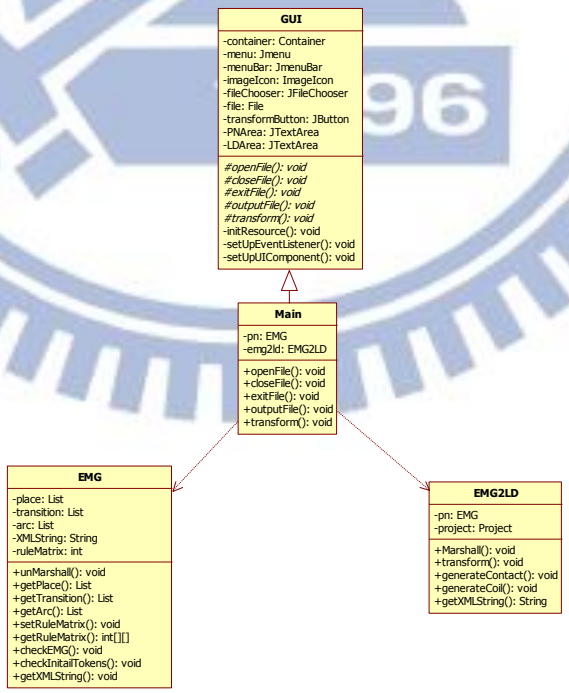


圖 4.13 程式的類別圖

4.5.2 程式流程與物件轉換程式碼設計

本節說明程式如何執行圖 4.12 圈選處的流程，以及說明如何設計物件轉換程式。首先程式啟動後，其使用者介面如圖 4.14。介面左邊是展示 PNML 文件內容，介面右邊是展示轉換後的 PLCopenXML 文件內容。兩者中間有一個 TCPN->LD 的按鈕，在程式匯入 PNML 並驗證文件為正確後，才可以點擊。

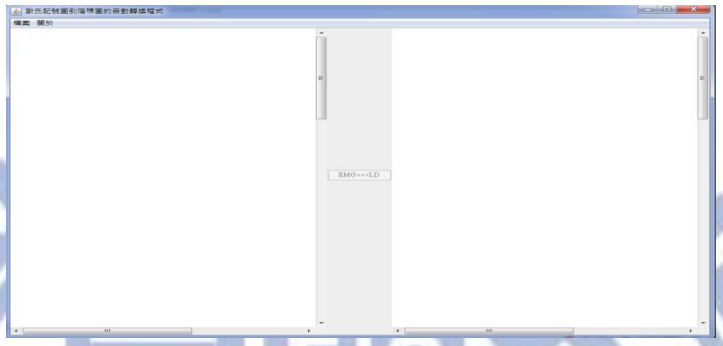
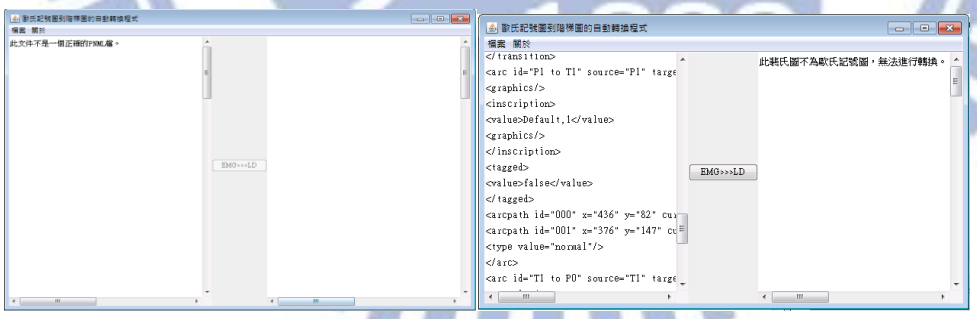
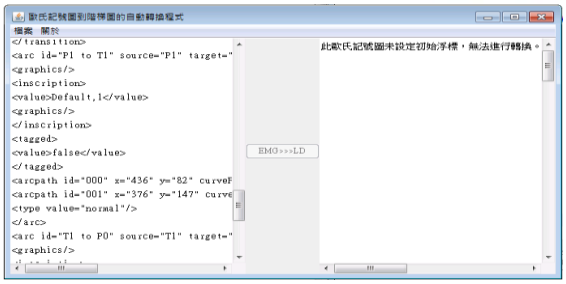


圖 4.14 自動轉換程式初始介面

接著人員透過程式匯入要轉換的 PNML 文件，匯入後程式會驗證文件是否為 PNML 檔案，若有錯誤，則程式會顯示此文件不是正確的 PNML 文件，如圖 4.15 (a)。若正確，程式就會以類別 EMG 的方法 getXMLString()在使用者介面的左邊展示文件內容，並透過方法 unMarshall()解編文件到歐氏記號圖物件，程式碼可見圖 2.29。程式透過方法 checkEMG()和 checkInitailTokens()驗證 PNML 是否為歐氏記號圖與是否有設定初始浮標。若兩者其中有一個為否，則程式會告知無法轉換。無法轉換介面如圖 4.15 的(b)與(c)。



(a)文件並非 PNML 檔案　　(b)文件並非歐氏記號圖



(c)文件並未設定初始浮標

圖 4.15 各種文件規格錯誤畫面

接著是建立與合併法則矩陣，程式以方法 setRuleMatrix()設定法則矩陣，將多張法則矩陣合併並且展示出來。建立法則矩陣的方法主要是截取方向弧物件，找到每個方向弧的 source 與 target，若 T1 為 target，P1 為 source，則法則矩陣記作-1，反之記作 1。建立法則矩陣的程式碼與輸出法則矩陣的結果如圖 4.16 所示。

```
//合併法則矩陣
public void setRuleMatrix() {
    ruleMatrix = new int[transition5.size()][place5.size()];

    String source = "", target = "";

    for (int i = 0; i < Main.arc1.size(); i++) {
        source = Main.arc1.get(i).getSource();
        target = Main.arc1.get(i).getTarget();

        for (int j = 0; j < Main.place1.size(); j++) {
            // place的input_arc
            if (target.equals(place5.get(j).getId())) {
                for (int k = 0; k < Main.transition1.size(); k++) {
                    if (source.equals(transition5.get(k).getId())) {
                        ruleMatrix[k][j]++;
                    }
                }
            }
            // place的output_arc
            if (source.equals(place5.get(j).getId())) {
                for (int k = 0; k < Main.transition1.size(); k++) {
                    if (target.equals(transition5.get(k).getId())) {
                        ruleMatrix[k][j]--;
                    }
```

法則矩陣

	P0	P11	P2	P0	P22	P3	P0	P33	P4	P0	P44	P5
T1	1	-1	0	0	0	0	0	0	0	0	0	0
T11	-1	1	-1	0	0	0	0	0	0	0	0	0
T1	0	0	0	1	-1	0	0	0	0	0	0	0
T22	0	0	0	-1	1	-1	0	0	0	0	0	0
T1	0	0	0	0	0	0	1	-1	0	0	0	0
T33	0	0	0	0	0	0	-1	1	-1	0	0	0
T1	0	0	0	0	0	0	0	0	0	1	-1	0
T44	0	0	0	0	0	0	0	0	0	-1	1	-1

圖 4.16 建立法則矩陣程式碼與輸出結果

當人員點擊按鈕 TCPN->LD 後，程式開始進行物件的對應轉換，其轉換流程即圖 4.12 圈選處。主要分為設定初始階梯圖、設定同步階梯圖與設定歐氏階梯圖三個部份產生階梯圖。接點與線圈物件的產生與屬性，是分別透過在類別 EMD2LD 的方法 generateContact() 和 generateCoil()進行設定。

對於設定初始狀態而言，其程式碼如圖 4.17(a)所示。由圖 4.17(a)可看出程式呼叫方法 generateContact()產生接點物件名稱設為 X000，再透過 for 迴圈對每個有初始浮標的暫存點呼叫方法 generateCoil()產生線圈，名稱對應到初始浮標所在的暫存點，並設定 storage 屬性為 SET。由於本論文是輸入多張歐氏記號圖，故在初始浮標裡會許很多是重複的。因此必需刪除重覆的初始狀態。利用兩個 for 迴圈者出重覆的初始狀態名稱並將其刪除，其程式碼如圖 4.17(b)。

```
//設定初始狀態
Contact initialContact=generateContact(100, positionY, 100, positionY+8, first, 1 , ++id ,
        "X000", factory);

contactList.add(initialContact);
bodyLd.getCommentOrErrorOrConnector().add(initialContact);

boolean second=false;//看線圈是不是有跨層
int last_positionY=positionY;//last都是最後接點的位置y
int last_Id=id;//最後接點的id

for(int i=0;i<n.size();i++){//n是初始暫存點 設定初始狀態線圈
    Coil coil=generateCoil(positionY, positionY+8, second, positionX, last_positionY+8,
            last_Id, ++id, n.get(i), factory);
    coil.setStorage(StorageModifierType.SET);
    bodyLd.getCommentOrErrorOrConnector().add(coil);
    coilList.add(coil);//他是第一個接點
    positionY+=40;//y=y+40
    second=true;//因為他有跨行
}
```

(a) 初始階梯圖設定

```
for(int i=0;i<n.size();i++){
for(int j=0;j<n.size();j++){

if(i!=j){
if(n.get(i).equals(n.get(j))){

n.remove(j);
//刪掉重覆的初始place
                }
            }
        }
}
```

(b)刪掉重覆初始狀態

圖 4.17 初始階梯圖設定程式碼

對於設定同步階梯圖而言，其程式碼如圖 4.18。程式先以 for 迴圈開始找出法則矩陣中每行合為-1 的 place。程式透過法則矩陣物件 matrix 的元素是否行加總合為-1，來判斷是否為控制暫存點，若是，則呼叫 generateCoil()產生線圈物件，名稱對應到控制暫存點的名稱，也就是程式碼中的 syPlace。以 generateContact()產生接點物件，也就是程式碼中的 X001 與 X002，並讓其串聯。利用 X001 與 X002 的開啟與關閉的組合來控制控制暫存點所對應之線圈。

```
for(int column=0;column<placeY.size();column++){
    for(int row=0;row<matrix.length;row++){
        a=a+matrix[row][column];
    }
    if(a==-1){
         syplace.add(placeY.get(column));//找出控制place
    }
    a=0;
}
////////設定同步梯氏圖
Contact X001=generateContact(100, positionY, 71, positionY+8, true, id++ , id ,
        "X001", factory);
bodyLd.getCommentOrErrorOrConnector().add(X001);
contactList.add(X001);//他是第一個接點

Contact X002=generateContact(150, positionY, 121, positionY+8, false, id++ , id ,
        "X002", factory);
bodyLd.getCommentOrErrorOrConnector().add(X002);
last_Id=id;
String Y1=syplace.get(0);
    Coil coil1=generateCoil(positionY, positionY+8, second, 171, positionY+8,
            last_Id, ++id, Y1, factory);
    bodyLd.getCommentOrErrorOrConnector().add(coil1);
    coilList.add(coil1);
    positionY+=40;
    second=false;
```

圖 4.18 同步階梯圖設定程式碼

對於設定歐氏記號圖而言，主要是用到 2.3.2 節所提到的法則R_E與法則R_F，其程式碼如圖 4.19 所示。程式先以 for 迴圈開始對每列的暫存點設定法則R_E。程式透過法則矩陣物件 matrix 的元素是否為-1，來判斷是否為轉移點的輸入暫存點，若為-1，則呼叫 generateContact()產生接點物件，也就是程式碼中的 inputPlace。程式在轉換所有輸入暫存點後，再呼叫 generateCoil()產生線圈，也就是程式碼中的 transitionName。再來程式呼叫 generateContact()產生接點物件，名稱對應到前面設定為法則R_E的轉移點。接著程式透過判斷 matrix 中的值是 1 或-1 以產生線圈。如果是-1，則程式呼叫 generateCoil()產生線圈，名稱對應到輸入暫存點，並設定其屬性為 RESET；如果是 1，則呼叫 generateCoil()產生線圈，即程式碼中的 outputPlace，並設定其屬性為 SET。

最後程式利用 for 迴圈對每個轉移點都設定完初始狀態、同步裴氏圖與歐氏記號圖後，會呼叫方法 Marshall()將物件組編為文件，並將文件展示在使用者介面的右邊。轉換結果如圖 4.20。

```
//設定法則Ri
Contact initialContact=generateContact(100, positionY, 100, positionY+8, first, 1 , ++id ,
        "X000", factory);
contactList.add(initialContact);
bodyLd.getCommentOrErrorOrConnector().add(initialContact);

boolean second=false;
int last_positionY=positionY;
int last_Id=id;

for(int i=0;i<n.size();i++){
    Coil coil=generateCoil(positionY, positionY+8, second, positionX, last_positionY+8,
            last_Id, ++id, n.get(i), factory);
    coil.setStorage(StorageModifierType.SET);
    bodyLd.getCommentOrErrorOrConnector().add(coil);
    coilList.add(coil);
    positionY+=40;
    second=true;
}
```

```
//設定轉移點觸發條件
for(int row=0;row<matrix.length;row++){
    first = true;
    positionX=100;

    for(int column=0;column<matrix[row].length;column++)
    if(matrix[row][column]==-1){
        String inputPlace=placeY.get(column);
        Contact contact=generateContact(positionX, positionY, positionX-29, positionY+8
        , first, id++, id, inputPlace, factory);
        bodyLd.getCommentOrErrorOrConnector().add(contact);
        if(first){
            contactList.add(contact);
        }
        first=false;
        positionX+=50;
    }
    last_Id=id;
    second=false;

    String transitionName=((NameType)((TransitionType)pn.getTransition().get(row)).getName()).getValue();
    Coil coil=generateCoil(positionY,positionY+8,second,positionX-29,positionY,last_Id,++id,transitionName,factory);
    bodyLd.getCommentOrErrorOrConnector().add(coil);
    coilList.add(coil);
    positionY+=40;
```

圖 4.19 歐氏階梯圖設定程式碼

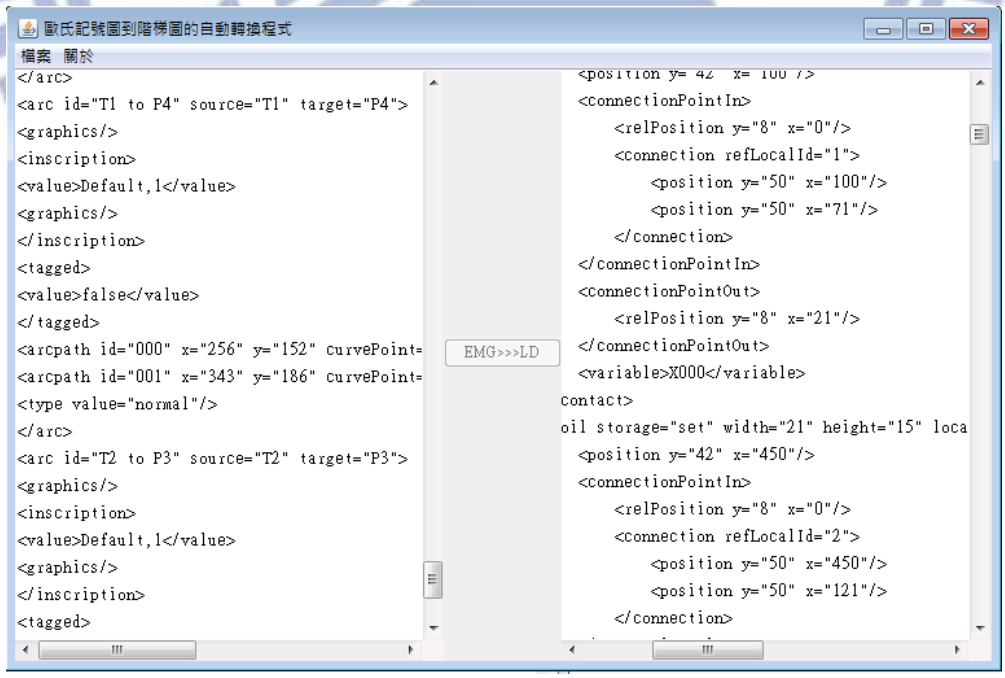


圖 4.20 轉換結果

最後使用人員可以透過程式的匯出檔案功能將轉換後的 PLCopenXML 文件匯出，並選擇想要儲存的位置。而使用人員將轉換後的文件匯入到軟體 PLCopenEditor，就可以得到階梯圖的圖件。此即分解後的多張歐氏記號圖到浮標守恆階梯圖的整體自動轉換過程。圖 4.21 為圖 2.3 在 PLCopenEditor 顯示的轉換結果。透過此程式，浮標守恆裴氏圖便可自動轉換為浮標守恆階梯圖。

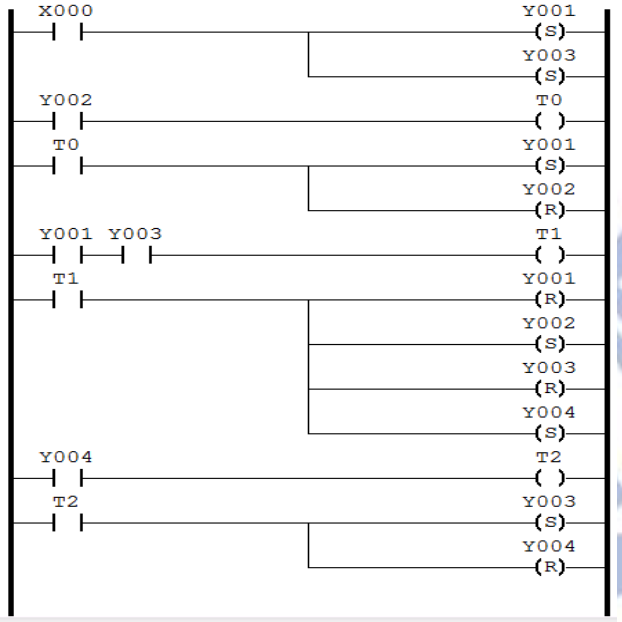


圖 4.21 轉換後的階梯圖

第五章　自動轉換程式的操作說明與案例操作

本章介紹自動轉換程式的操作流程，接著以自動灌模系統為例，將 3.4.2 節所提到的自動灌模系統，將分解後的四張詮釋型歐氏記號圖自動轉換成浮標守恆階梯圖。5.1 節說明自動轉換程式操作流程，5.2 節為自動灌模系統案例操作。

5.1 自動轉換程式操作流程

本節將說明自動轉換程式的操作流程，將圖 4.12 所框選的自動轉換流程仔細的說明如何操作。5.1.1 小節為 Pipe 產生 PNML 操作說明，此小節會詳細的介紹如何利用 Pipe 軟體繪製裴氏圖並且將其匯出成 PNML 檔案。5.1.2 小節為程式轉換操作說明，此小節將介紹自動轉換程式的介紹以及如何操作自動轉換程式。5.1.3 小節為 PLCopenXML 產生階梯圖操作說明，此小節介紹如何將匯出的階梯圖 PLCopenXML 檔案利用 PLCopenEditor 展示階梯圖。5.1.4 小節為三菱廠牌 PLC 操作說明，此小節說明如何操作三菱廠牌的 PLC。

5.1.1Pipe 產生 PNML 操作說明

裴氏圖的圖件需要可以轉成 PNML 的文件，本論文利用 Pipe 軟體繪製裴氏圖。而 Pipe 軟體優點是好操作容易繪圖也能計算裴氏圖的不變量，最重要的是可以將圖件儲存成 PNML 文件。首先將 Pipe 軟體開啟，將詮釋型歐氏記號圖畫在 Pipe 軟體上，如圖 5.1 所示。繪製完後點選檔案>另存新檔，將圖件存檔在人員指定的位置，存成 PNML 文件檔，如圖 5.2 所示。

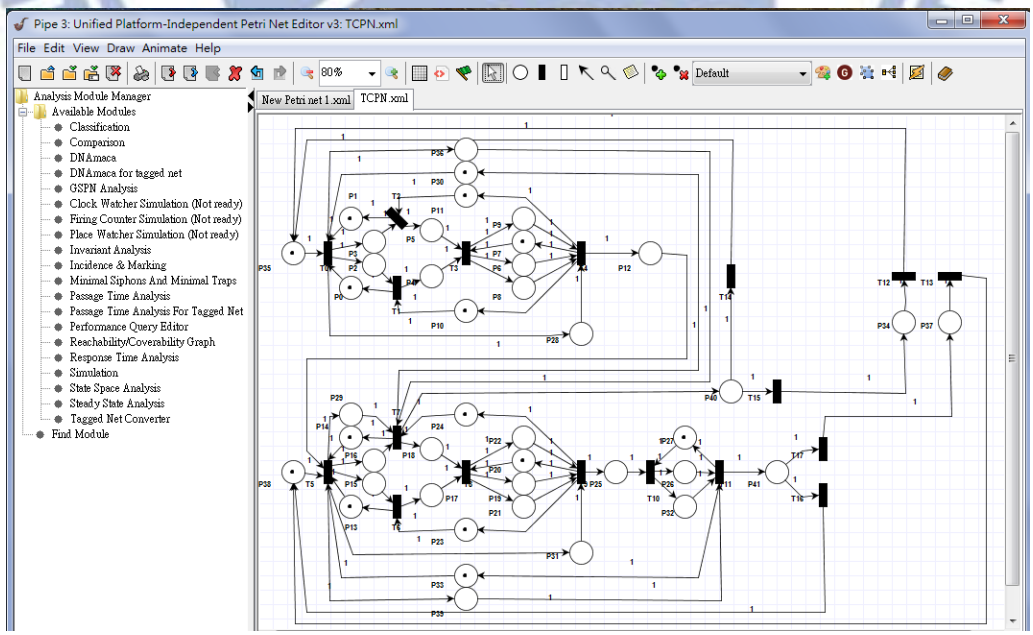


圖 5.1 Pipe 繪製裴氏圖

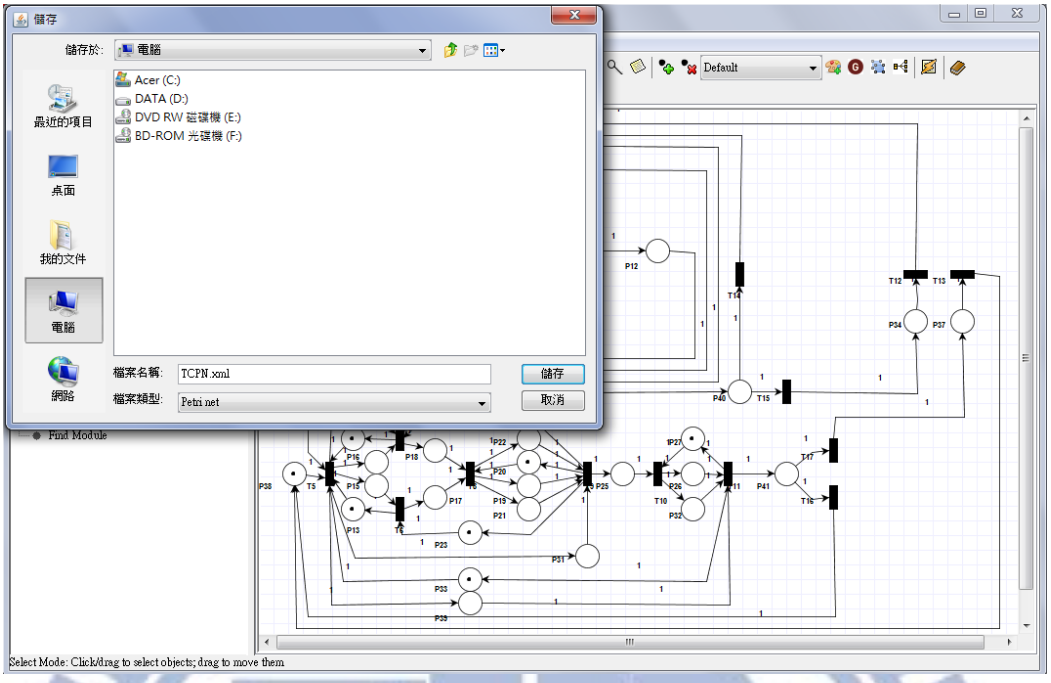


圖 5.2 裴氏圖圖件存檔成 PNML 文件

5.1.2 程式轉換操作說明

此小節會介紹如何操作自動轉換程式。開啟程式後，介面左邊是展示 PNML 文件內容，介面右邊是展示轉換後的 PLCopenXML 文件內容。兩者中間有一個 TCPN->LD 的按鈕。首先要輸入 PNML 檔案，點選左上角檔案，選取欲選擇的 PNML 檔案，如圖 5.3 所示。

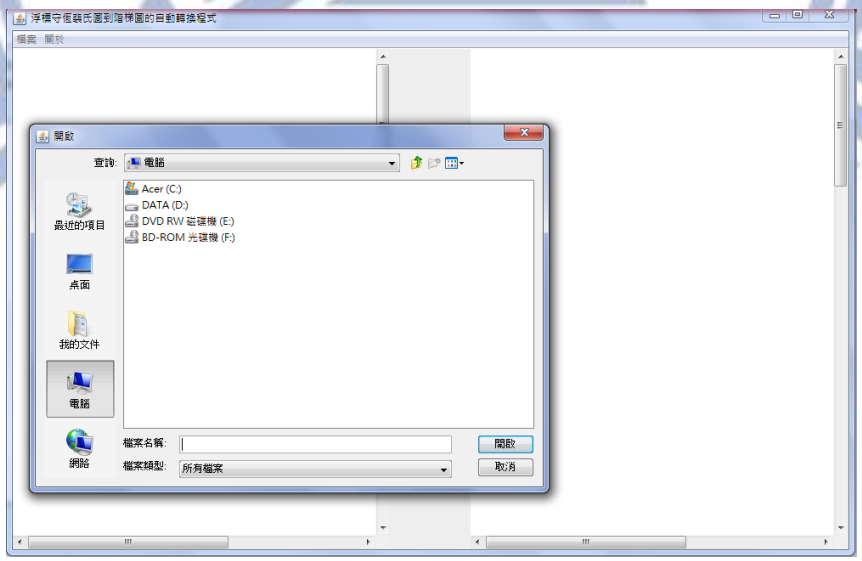


圖 5.3 選取 PNML 檔案

選取多張歐氏記號圖的 PNML 檔案後，左邊會出現 PNML 文件資訊，以及跳出視窗出現物件資訊與法則矩陣。其結果如圖 5.4。

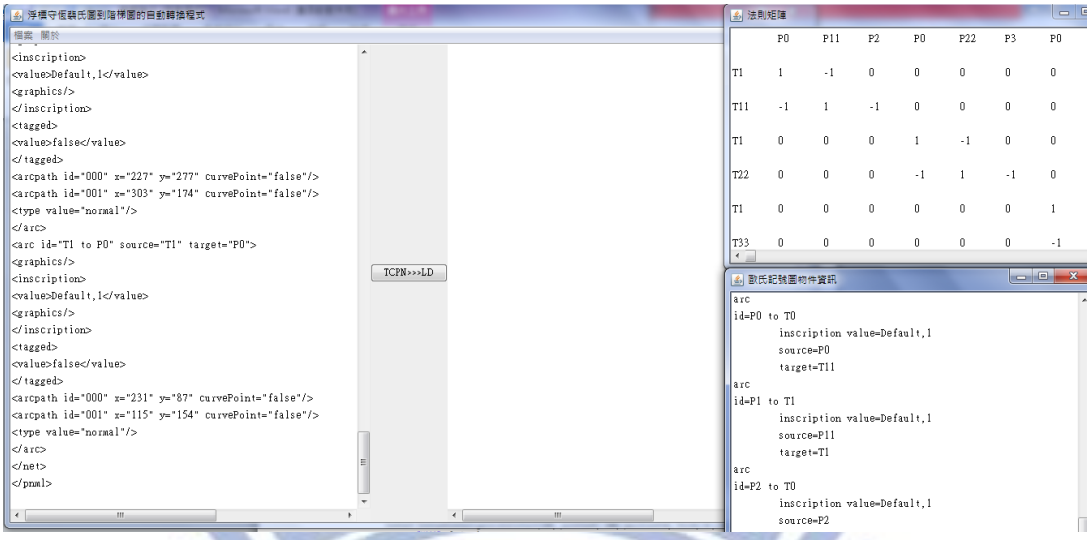


圖 5.4 輸入 PNML 後產生文件資訊、物件資訊與法則矩陣

接著點選中間的 TCPN->LD 按鈕，將輸入的這些詮釋型歐氏記號圖轉換成浮標守恆階梯圖。轉換後產生階梯圖物件資訊與 PLCopenXML 文件資訊顯示在右邊框框。其結果如圖 5.5 所示。

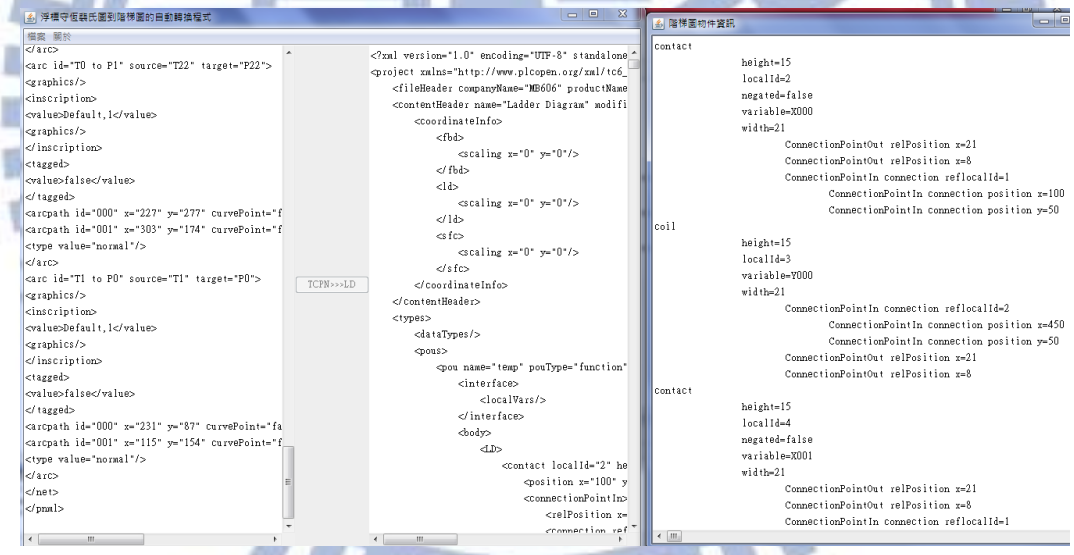


圖 5.5 轉換後產生階梯圖物件資訊與 PLCopenXML 文件資訊

最後需要將檔案匯出成 PLCopenXML 檔案並且儲存，點選左上角檔案，再點選匯出檔案，並選擇想要儲存的位置，如圖 5.6 所示。自動轉換程式的操作就如此簡單，利用此程式可以成功的轉換成浮標守恆階梯圖並且儲存起來。本論文會在下一小節介紹如何開啟階梯圖。

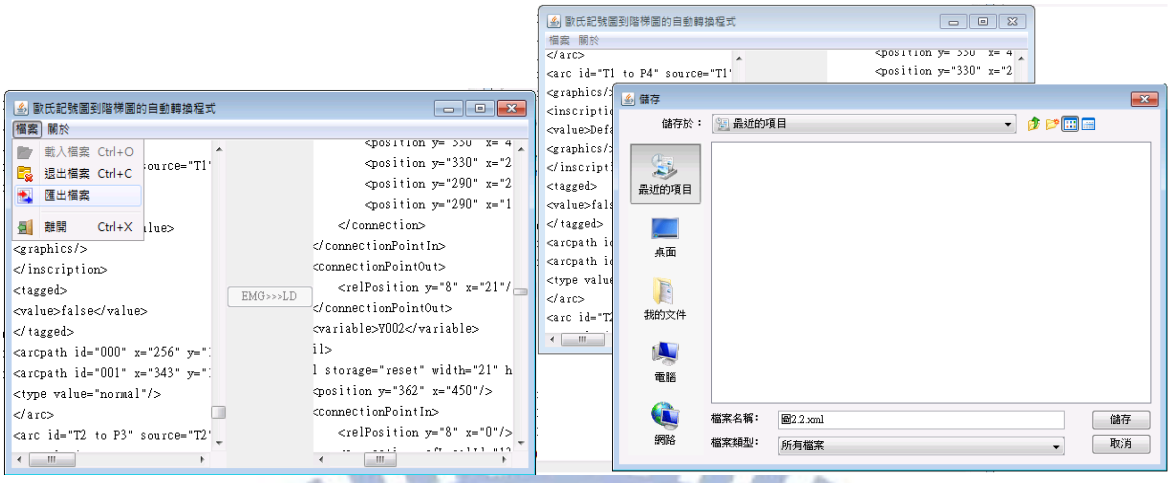


圖 5.6 檔案匯出介面

5.1.3PLCopenXML 產生階梯圖操作說明

在前一小節轉換出 PLCopenXML 文件後，這時需要軟體讀取其文件，並顯示階梯圖的圖件。本論文使用 PLCopenEditor 軟體，此軟體可以讀取 PLCopenXML 文件，並且容易操作。首先開啟 PLCopenEditor 軟體，點選左上角檔案再點選開啟檔案，將 PLCopenXML 開啟後，點選左邊的 Functions 便能顯示其階梯圖，其顯示結果如圖 5.7 所示。

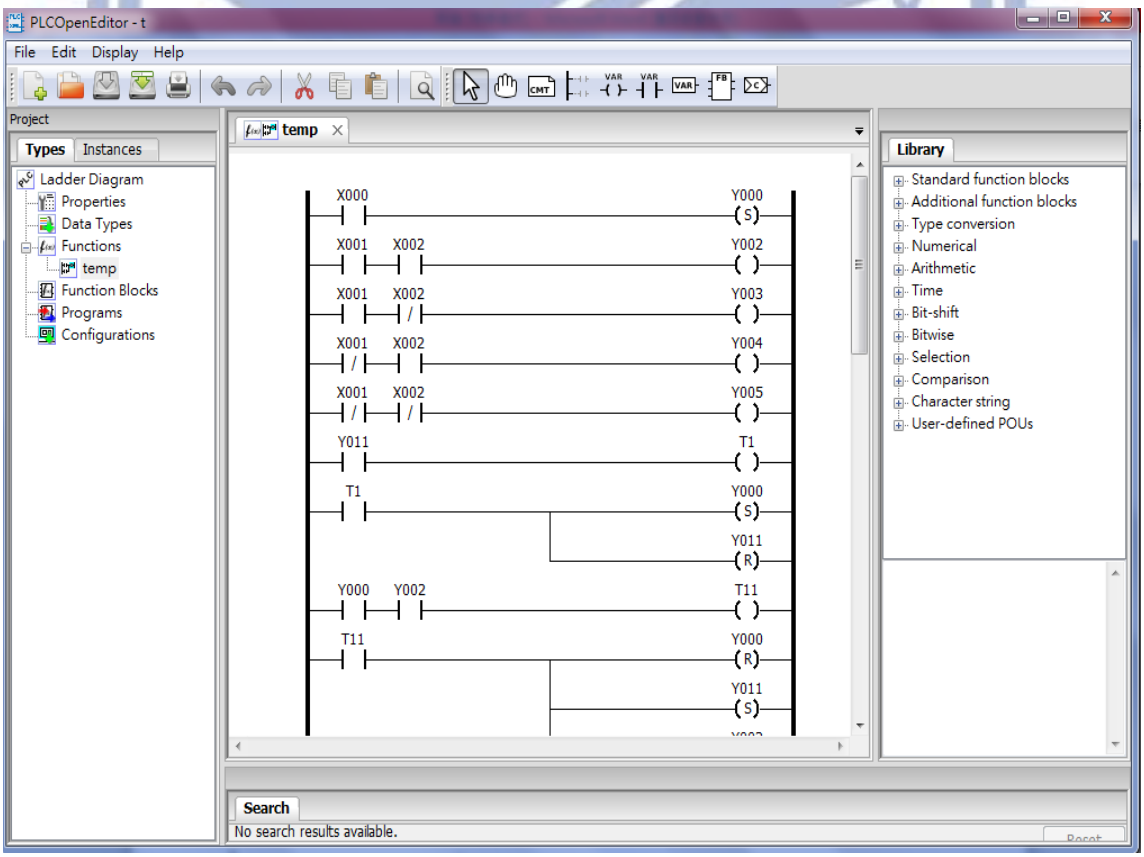


圖 5.7PLCopenEditor 軟體顯示階梯圖

5.1.4 三菱廠牌 PLC 操作說明

由於 PLCopenEditor 軟體產生的階梯圖為標準表達階梯圖，而本論文實作採用三菱廠牌 PLC，三菱廠牌 PLC 的階梯圖[14]為特殊表達階梯圖。在使用三菱廠牌的 PLC 之前，先將 PLCopenEditor 軟體產生的階梯圖繪製成特殊表達階梯圖，這兩者的差異與不同可以參考[8]。繪製成特殊表達階梯圖後，便能輸入進 PLC 中，使其運作。

欲繪製特殊表達階梯圖，首先開啟 GX Developer 軟體，繪製特殊表達階梯圖。之後點選上方 Online→write to PLC，如圖 5.8 所示。便能將階梯圖寫入進 PLC 之中控制 PLC。操作 PLC 部份可看圖 2.16，下方開關為人員給予 PLC 外部指令，中間燈號則是對應裴氏圖中的暫存點。由燈號的情況便可以看出系統目前處於什麼狀態，進行什麼工作。開關的部份則可以由前面所提的同步階梯圖來由外部人員進行控制，選擇欲生產的方式或製程。

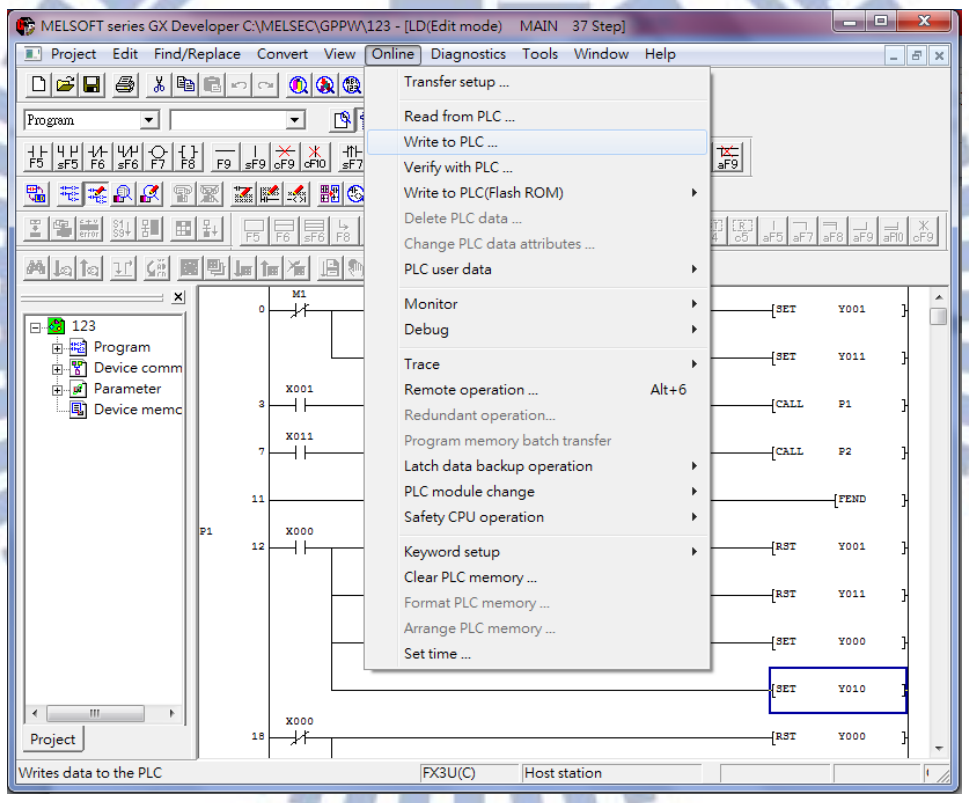


圖 5.8 階梯圖寫入 PLC

5.2 自動灌模系統案例操作

本節以自動灌模系統為例，在前面 3.4.2 節案例分析已經將浮標守恆裴氏圖分解成四張歐氏記號圖，並加入控制暫存點成為四張歐氏記號圖。接下來將這四張歐氏記號圖利用 Pipe 繪出，其結果如圖 5.9 至圖 5.12。圖 5.9 為加入控制暫存點之拉式歐氏記號圖。圖 5.10 為加入控制暫存點之混合式 A 歐氏記號圖。圖 5.11 為加入控制暫存點之混合式 B 歐氏記號圖。圖 5.12 為加入控制暫存點之推式歐氏記號圖。

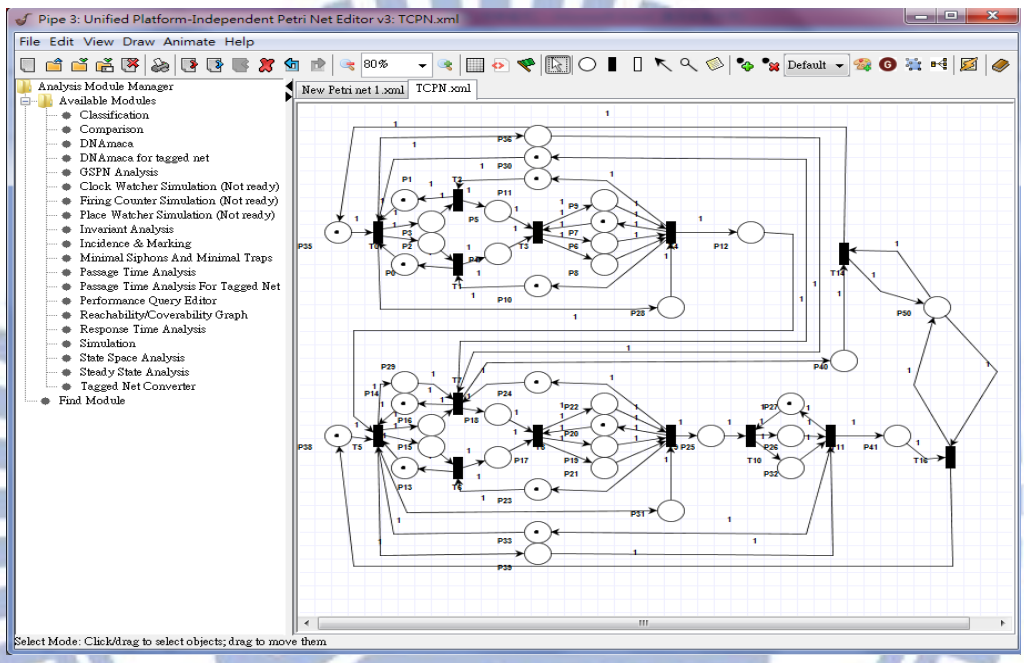


圖 5.9 加入控制暫存點之拉式歐氏記號圖

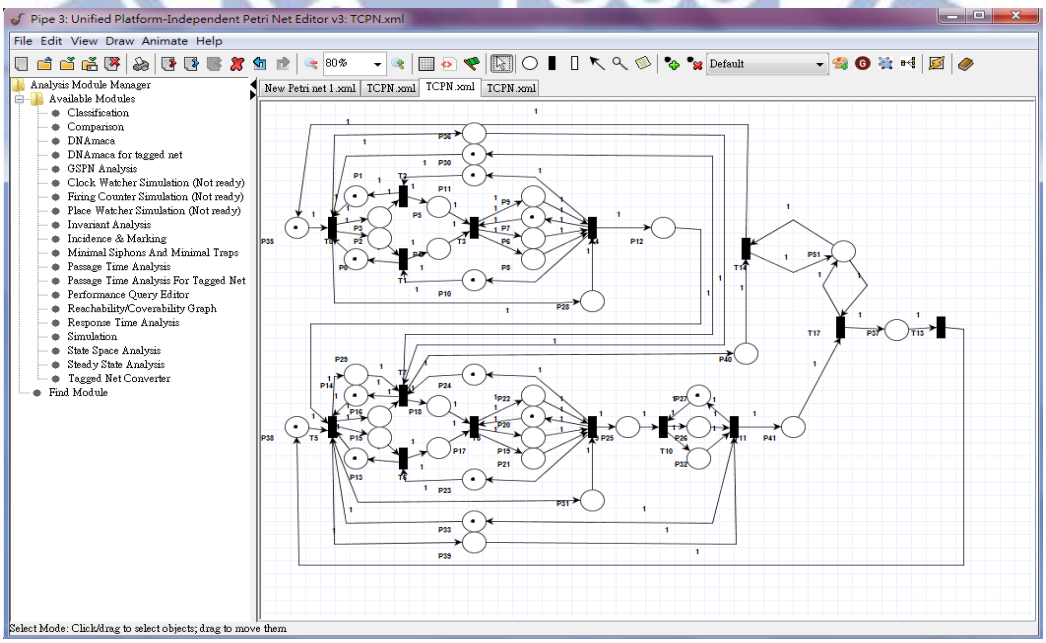


圖 5.10 加入控制暫存點之混合式 A 歐氏記號圖

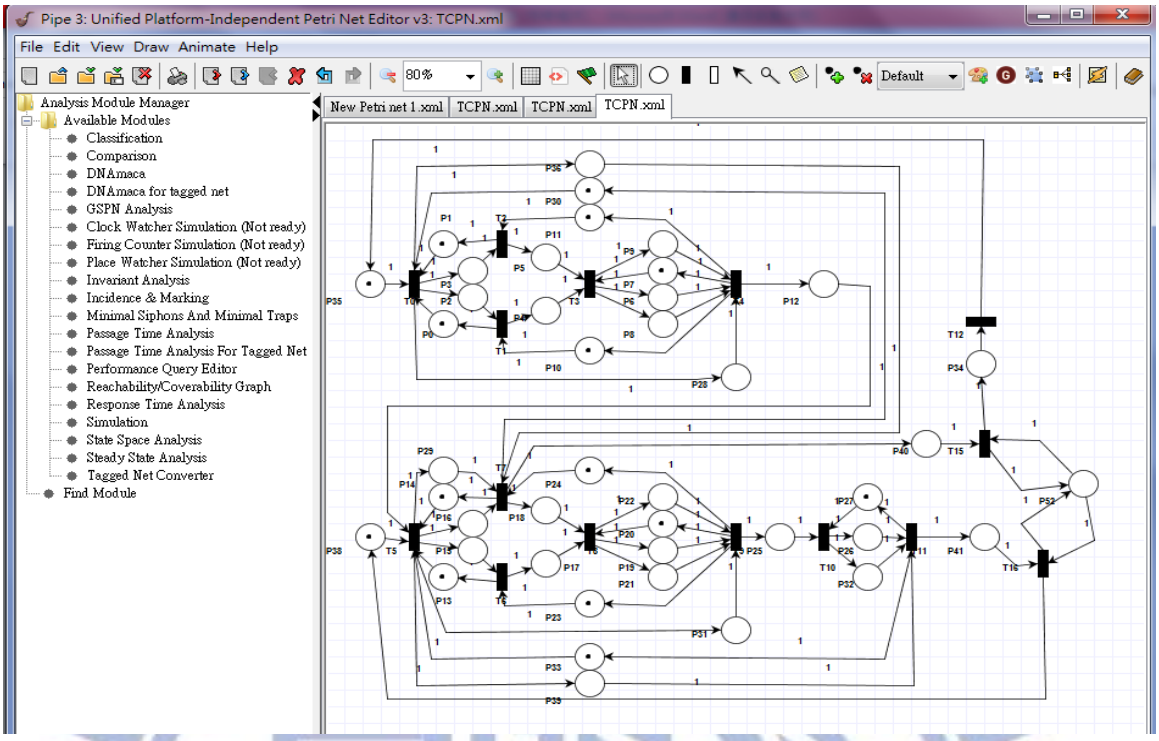


圖 5.11 加入控制暫存點之混合式 B 歐氏記號圖

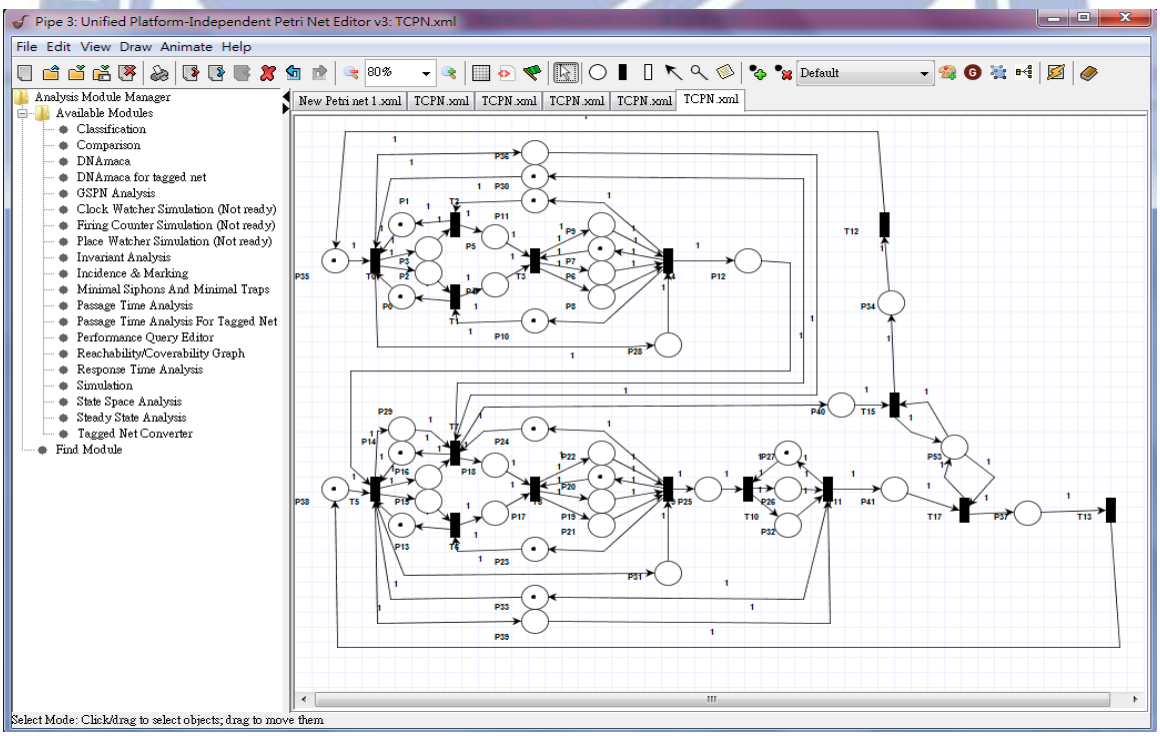


圖 5.12 加入控制暫存點之推式歐氏記號圖

接著使用人員透過自動轉換程式，將上面這四個 PNML 文件匯入到自動轉換程式裡。匯入文件後，程式會先驗證文件規格。若規格符合，則程式顯示文件內容在使用者介面左邊，並顯示其法則矩陣與四個歐氏記號圖的物件資訊，顯示結果如圖 5.13。

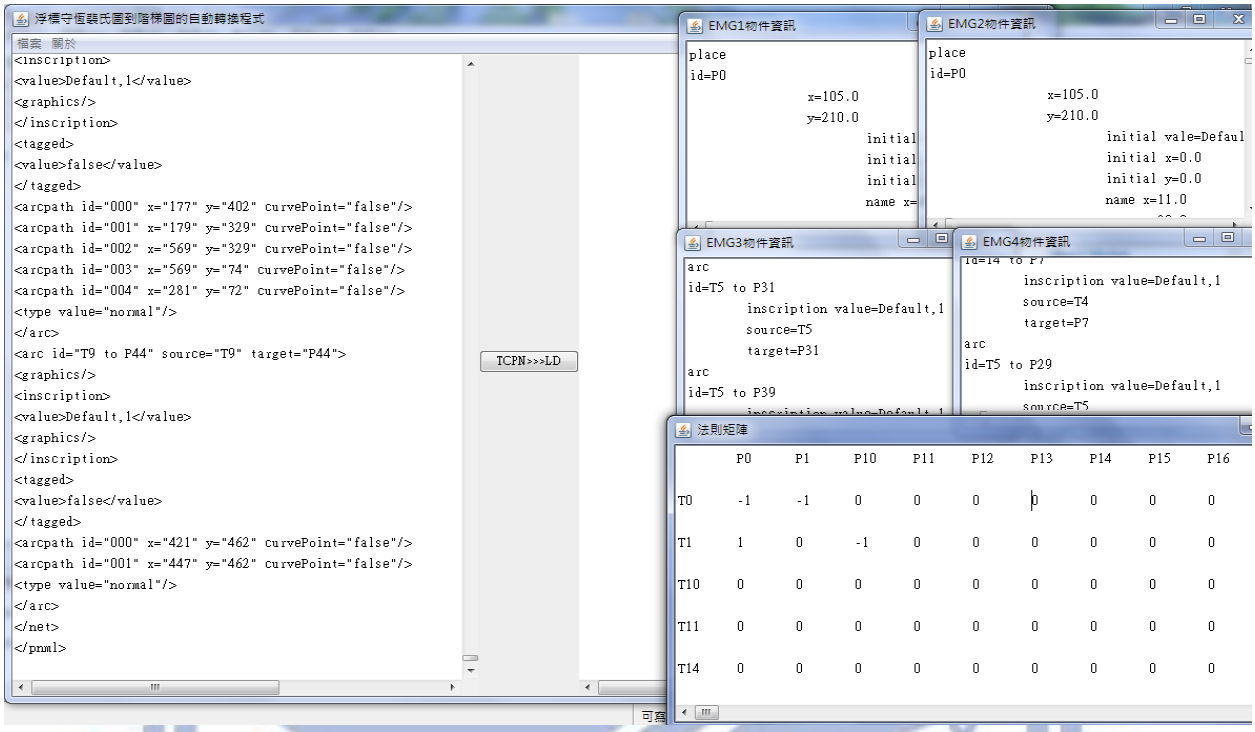


圖 5.13 自動灌模系統之文件資訊、物件資訊與法則矩陣

程式匯入成功後，人員點擊按鈕 TCPN->LD 後，程式會產生 PLCopenXML 文件，並將文件內容展示在使用者介面右邊，如圖 5.14 所示。接著人員透過轉換程式中的匯出檔案將文件以 PLCopenXML 格式匯出到電腦中。

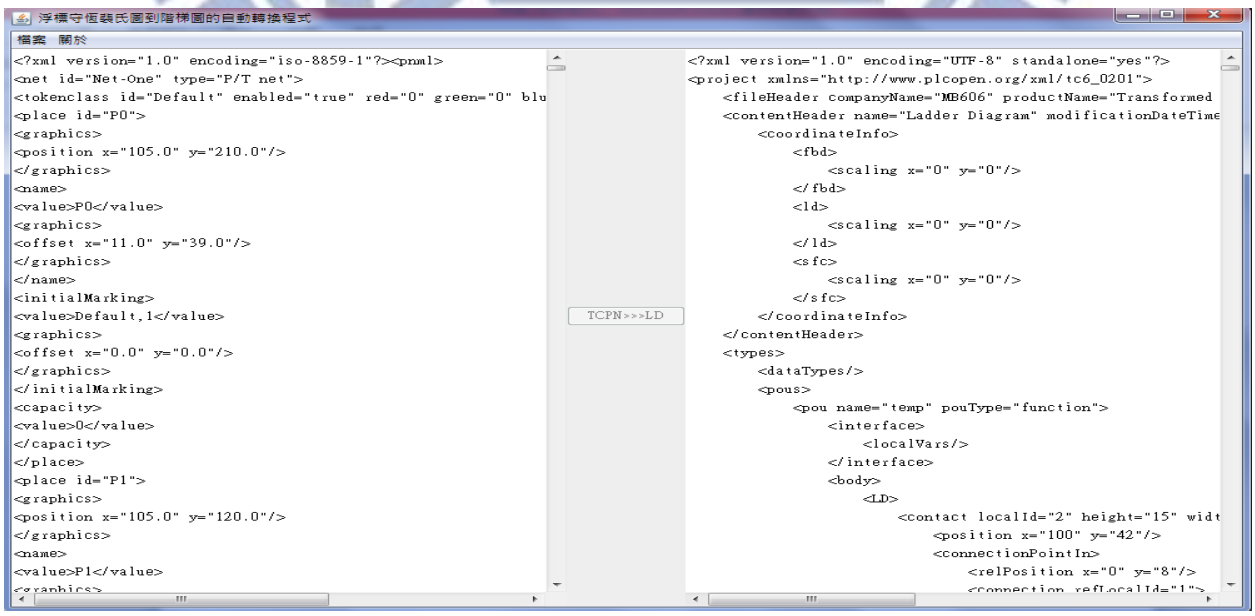


圖 5.14 程式產生自動灌模系統的階梯圖文件

最後人員透過 PLCopenEditor 軟體開啟 PLCopenXML 文件以展示自動灌模系統之階梯圖，如圖 5.15 至圖 5.19 所示。圖 5.15 為自動灌模系統之初始階梯圖與同步階梯圖，接點 X000 所並聯之線圈為初始階梯圖，其代表初始浮標的設定。接點 X001 與 X002 所建立的四節階梯為同步階梯圖，其連接之線圈 Y050、Y051、Y052 與 Y053 為同步裴氏圖之控制暫存點。人員可由外部開關 X001 與 X002 的開關組合控制 Y050、Y051、Y052 與 Y053 的符標有無。選擇所需的製程或生產方式。圖 5.16 至圖 5.19 為歐氏階梯圖的部份。圖 5.16 代表著拉式系統之歐氏階梯圖。圖 5.17 代表著混合式 A 系統之歐氏階梯圖。圖 5.18 代表著混合式 B 系統之歐氏階梯圖。圖 5.19 代表著推式系統之歐氏階梯圖。

最後本論文整理出自動灌模系統在四種情境下階梯圖的控制狀態，如表 5.1 所示。情境一為使用拉式生產策略，X001 與 X002 開關皆開啟。情境二為使用混合式 A 生產策略，X001 開啟與 X002 關閉。情境三為使用混合式 B 生產策略，X001 關閉與 X002 開啟。情境四為使用推式生產策略，X001 與 X002 開關皆關閉。

由結果可以看得出自動灌模系統的階梯圖很龐大也很複雜，其因內部製程分成推式、拉式、混合式 A 與混合式 B 四種製程。如果用人工方式去轉換此階梯圖，很容易造成錯誤也需要花費相當多的時間。此自動轉換程式可以解決人工轉換上容易出錯的問題，也可以省下非常大量的時間。故只要將系統的浮標守恆裴氏圖經由時域分解法分解出的歐氏記號圖並加入同步裴氏圖，使用此自動轉換程式進行轉換，便可以很簡單的得到此系統的階梯圖。

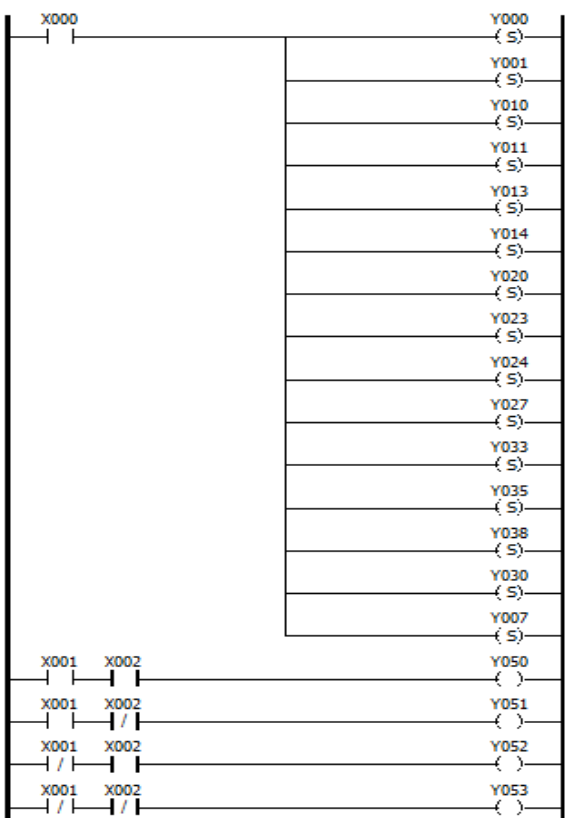


圖 5.15 自動灌模系統之初始階梯圖與同步階梯圖

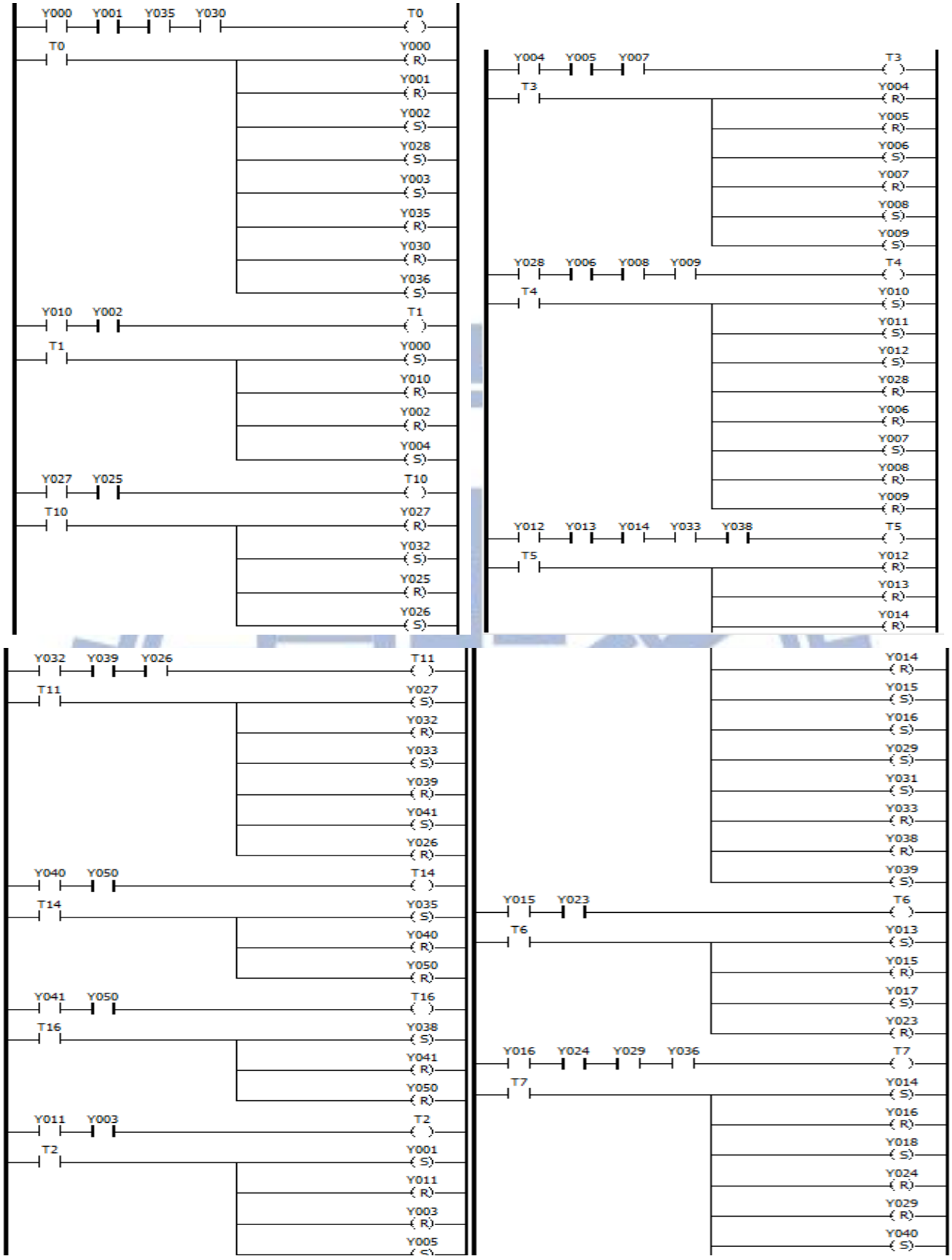


圖 5.16 拉式備料系統之歐氏階梯圖(EMG_1)

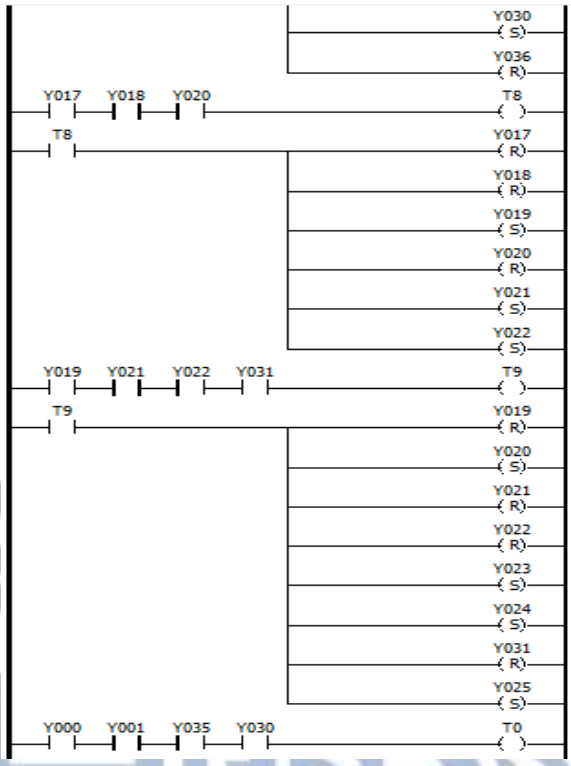


圖 5.16 (續)拉式備料系統之歐氏階梯圖(EMG_1)

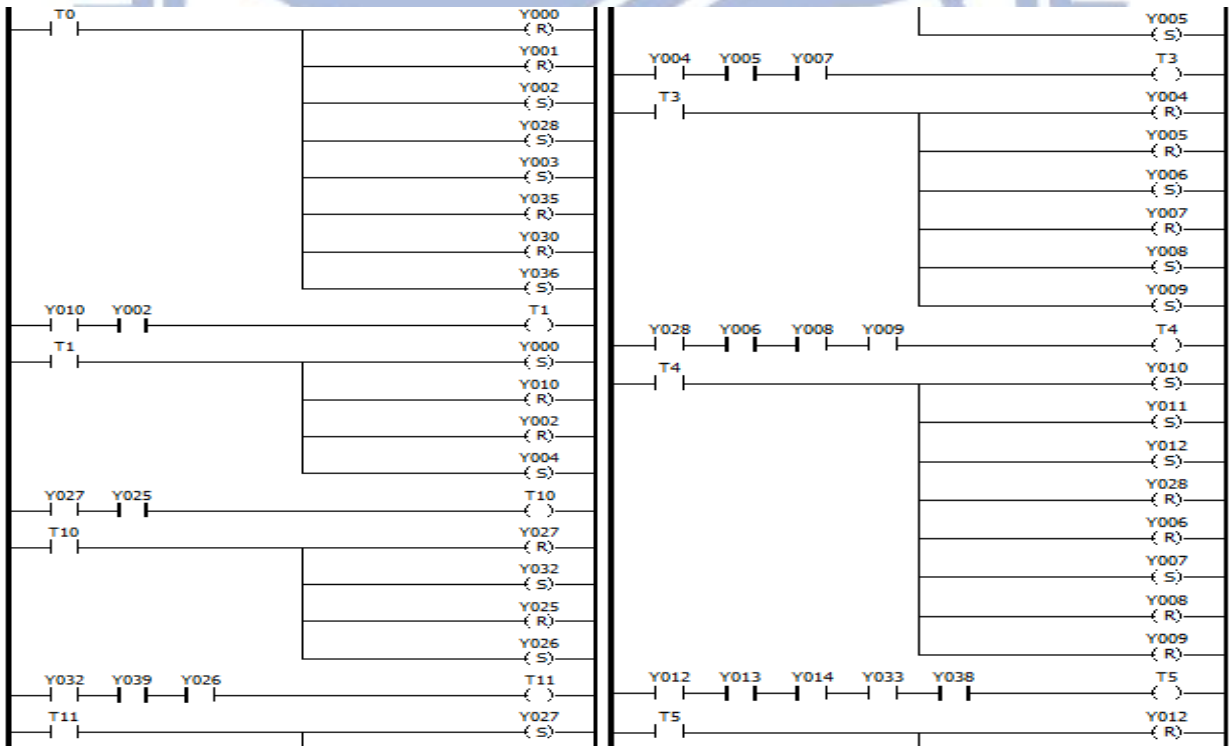


圖 5.17 混合式 A 備料系統之歐氏階梯圖(EMG_2)

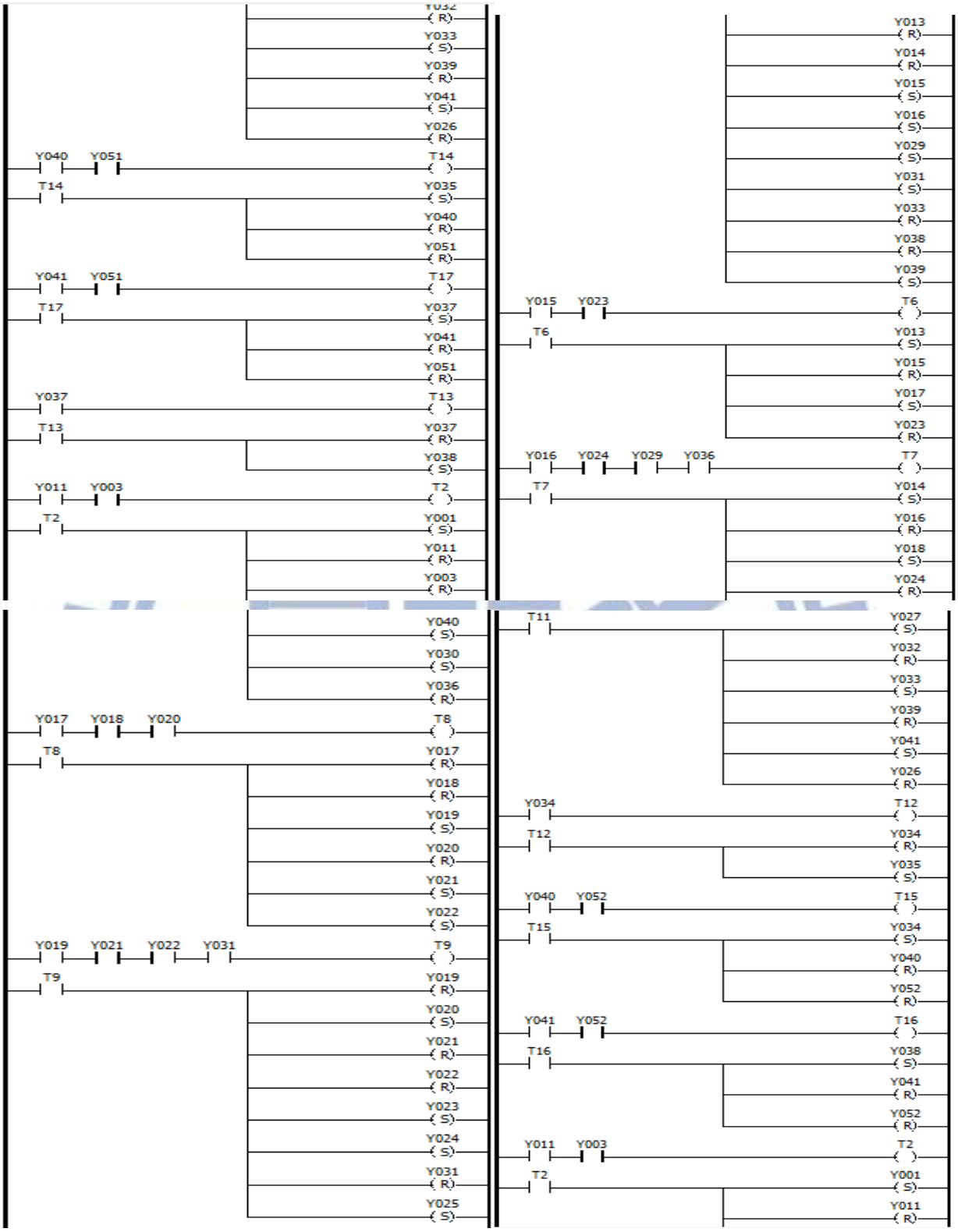


圖 5.17(續)混合式 A 備料系統之歐氏階梯圖(EMG_2)

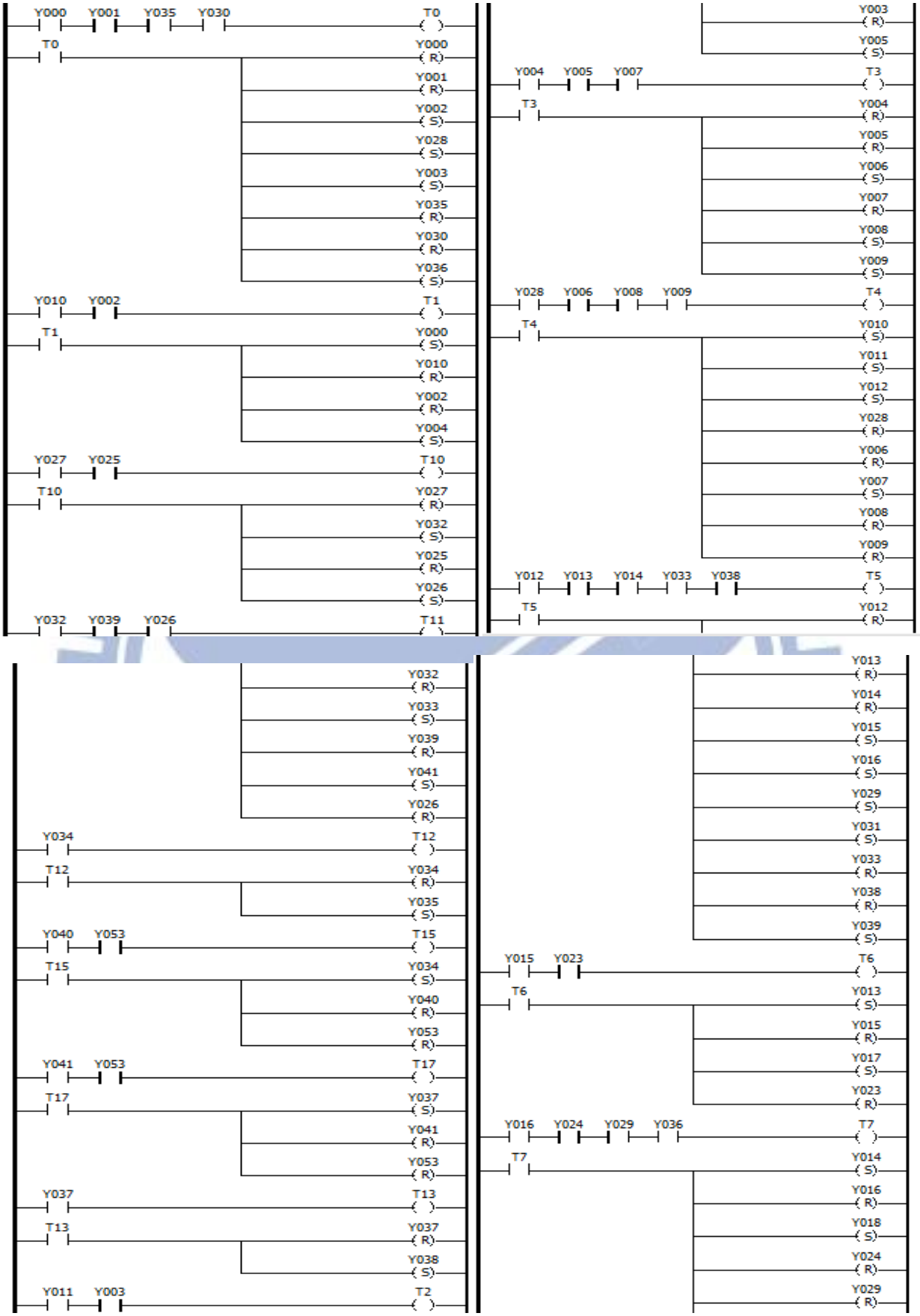


圖 5.18 混合式 B 備料系統之歐氏階梯圖(EMG_3)

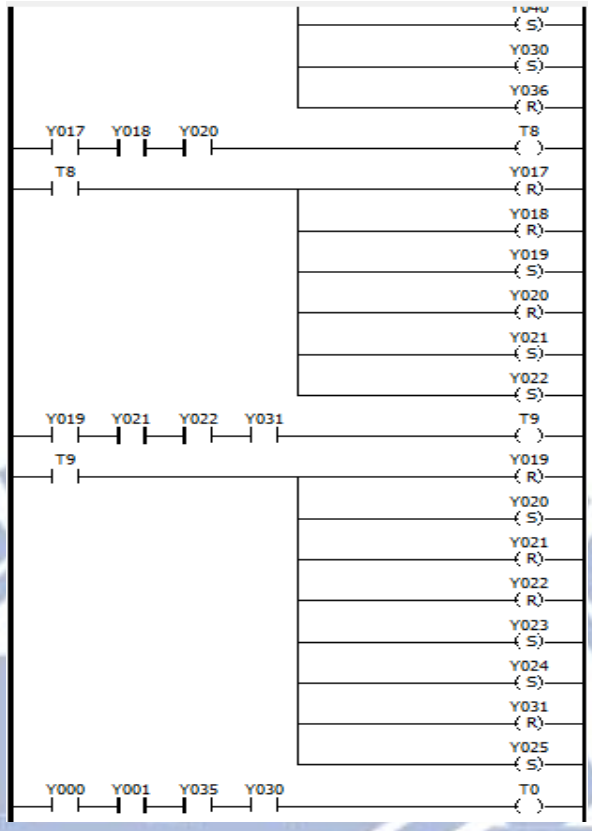


圖 5.18 (續) 混合式 B 備料系統之歐氏階梯圖(EMG$_3$)

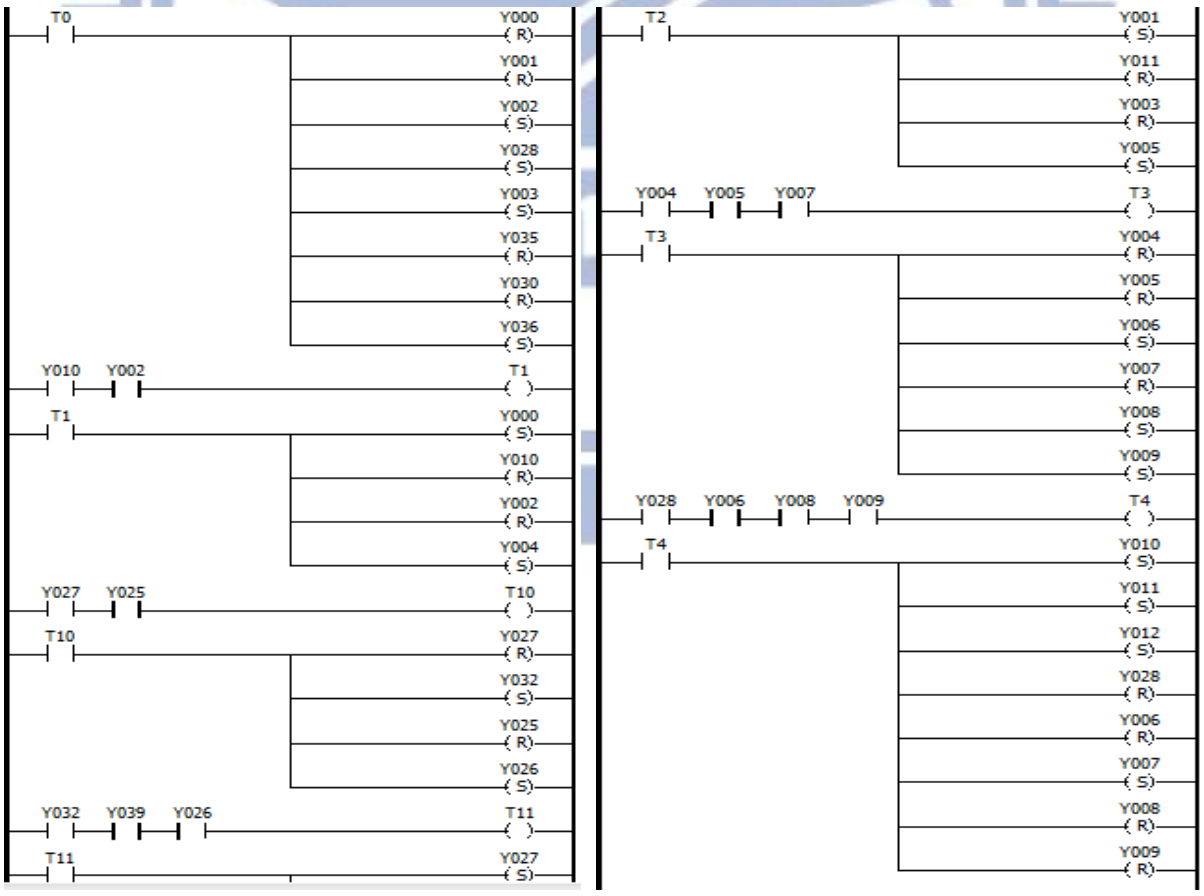


圖 5.19 推式備料系統之歐氏階梯圖(EMG$_4$)

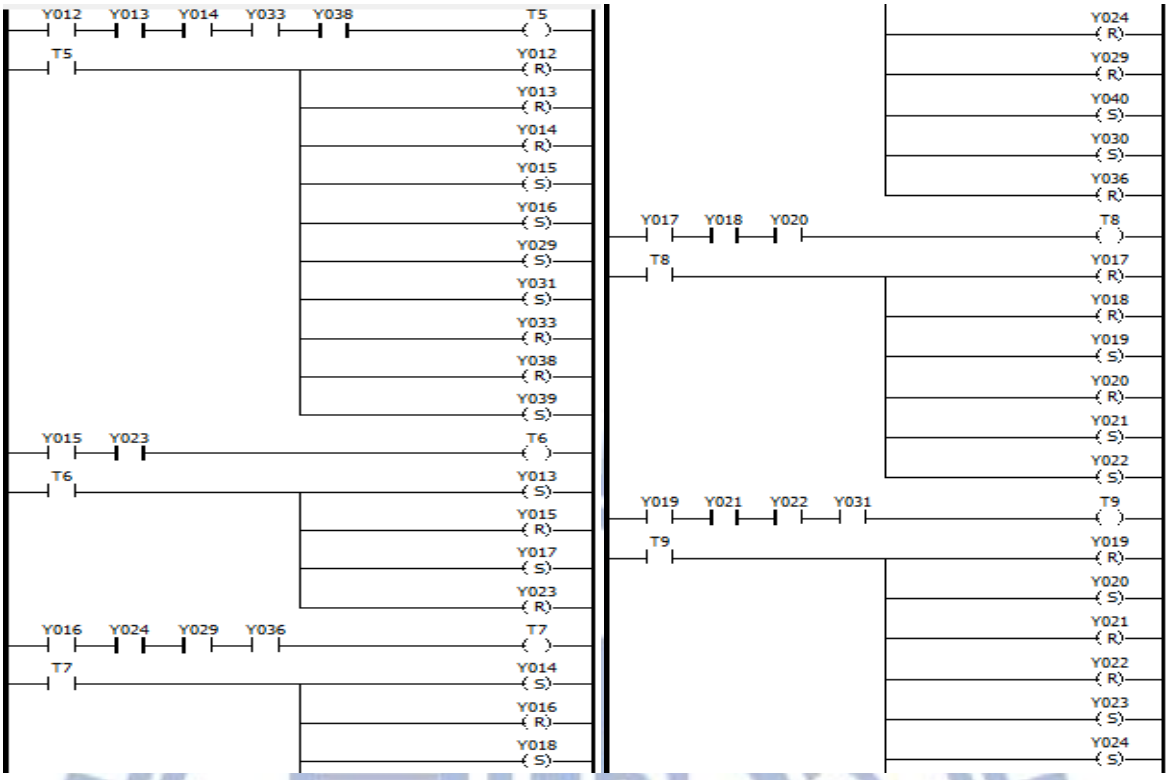


圖 5.19 (續)推式備料系統之歐氏階梯圖(EMG_4)

表 5.1 四種情境下階梯圖的控制狀態

	開關 X001	開關 X002	生產策略	對應 EMG
情境一	開	開	拉式生產策略	ENG_1
情境二	開	關	混合式 A 生產策略	ENG_2
情境三	關	開	混合式 B 生產策略	ENG_3
情境四	關	關	推式生產策略	ENG_4

最後可以由開關 X001 與 X002 來選擇控制四種情境，圖 5.20 為 PLC 上外部開關的位置關係。第一個開關為 X000 控制初始階梯圖也就是系統的開啟，第二個開關與第三個開關為 X001 與 X0002 控制同步階梯圖。而上面的燈號代表系統目前的狀態。

圖 5.20 不同情境與 PLC 開關的對應關係

第六章　結論

6.1 結論

在自動化彈性製造系統中，可利用浮標守恆裴氏圖來進行系統的建模與分析。但對於工程師來說，將浮標守恆裴氏圖轉換成階梯圖來控制彈性製造系統是很困難的一件事情。本研究目的為轉換浮標守恆裴氏圖到一個大型的階梯圖，此階梯圖包含初始階梯圖、同步階梯圖與歐氏階梯圖的合成。使其擁有浮標守恆裴氏圖的浮標行為。並將轉換過程自動化。在轉換的部份，本文將浮標守恆裴氏圖進行分解，在分解的過程中擁有智慧型的行為，會找出系統中被忽略或未被發現的製程作業，這些分解出來的製程與作業皆為歐氏記號圖。接著將這些歐氏記號圖加入同步裴氏圖並轉換成階梯圖。使其擁有同步裴氏圖的特性，可由外部環境或人員給予指令，彈性的控制生產系統。

在自動化轉換過程中，主要是自動化彈性製造系統通常複雜且龐大，對應之浮標守恆裴氏圖也是相當複雜。故需避免以人工方式產生階梯圖的過程發生錯誤並節省時間。而使用的自動轉換架構為是以三層式架構轉換法為基楚，避免直接以圖件進行轉換的困難，改以物件的方式對應轉換，使得程式可以撰寫的容易。

本論文是以張舜龍的法則轉換法為基楚，並加以延伸。主要的差別在於法則轉換法可將歐氏記號圖轉換為初始階梯圖與歐氏階梯圖，使其可以控制自動化重覆性製造性系統。而本論文多提出同步裴氏圖與同步階梯圖的概念，將浮標守恆裴氏圖轉換為初始階梯圖、同步階梯圖與歐氏階梯圖的合成。便可控制多個製造程序，控制多張歐氏記號圖。最後可以應用於自動化彈性製造系統上面，直接控制工廠裡的生產設備。

本論文發展出浮標守恆裴氏圖到階梯圖的轉換方法，以及實作出自動轉換程式。如此一來便能成功的使用浮標守恆裴氏圖來分析系統，也可以簡單的利用自動轉換程式轉換成階梯圖來直接控制系統。展望未來，本論文使得分析與操作龐大複雜的彈性製造系統更為容易。

參考文獻

[1] 梁高榮，「擬陣理論在現場監控的應用」，機械工業，六月，176-188 頁，2001。

[2] 梁高榮，「自動化製造系統內的即時故障察覺」，機械工業，五月，2006。

[3] 梁高榮，「虛體製造系統的多緒架構」，機械工業，十二月，111-121 頁，2006。

[4] 梁高榮，「裴氏圖與記號圖：可程式控制器的分析工具」，機械工業，十月，119-130 頁，2009。

[5] 梁高榮，「利用全跨模與覆點擬陣設計自動化工廠的偵查法則」，機械工業，六月，98-109 頁，2011。

[6] 梁高榮，「利用動態擬陣理論設計彈性製造系統的偵查法則」，機械工業，七月，87-100 頁，2011。

[7] 梁高榮，「自動化製造系統裡浮標守恆裴氏圖的時域分解」，機械工業，十月，76-89 頁，2011。

[8] 張舜龍，「用法則轉換法從歐氏記號圖產生階梯圖」，國立交通大學工業工程與管理研究所碩士論文，2012。

[9] 陳書皓、梁高榮，「五層式架構解歐氏記號圖/階梯圖轉換問題」，機械工業，六月，158-170 頁，2013。

[10] 黃柏勳，「建立自動化製造系統的開放資料伺服器連接架構」，國立交通大學工業工程與管理研究所碩士論文，2010。

[11] 黃柏勳、梁高榮，「三層式架構解記號圖/階梯圖轉換問題」，機械工業，八月，114-126 頁，2010。

[12] 辜婉琪，「自動灌模系統的浮標守恆裴氏圖合成」，國立交通大學工業工程與管理研究所碩士論文，2012。

[13] 鄭仲元，「記號圖到階梯圖的自動物件轉換」，國立交通大學工業工程與管理研究所碩士論文，2011。

[14] 廖文賢，三菱可程式控制器：指令應用例 100 題，双象貿易股份有限公司，2009。

[15] 廖成旺，三菱可程式控制器 FX3U 中文使用手冊，双象貿易股份有限公司，2009。

[16] Chirn, J. L. and MacFarlane D. C., “Petri Net Based Design of Ladder Logic Diagrams,” Technical Report, 2000.

[17] David, R and Alla, “Petri Nets & Grafcet: Tools for Modeling Discrete Event Systems,” Prentice Hall, 1992.

[18] http://docs.oracle.com/javase/tutorial/jaxb/index.html, Dec.20, 2012.

[19] http://en.wikipedia.org/wiki/Class_diagram, Dec.20, 2012.

[20] http://en.wikipedia.org/wiki/Document_Type_Definition, Dec.20, 2012.

[21] http://en.wikipedia.org/wiki/IEC_61131-3, Dec.20, 2012.

[22] http://en.wikipedia.org/wiki/Ladder_logic, Dec.20, 2012.

[23] http://en.wikipedia.org/wiki/Programmable_logic_controller, Dec.20, 2012.

[24] http://en.wikipedia.org/wiki/Unified_Modeling_Language, Dec.20, 2012.

[25] http://en.wikipedia.org/wiki/Xml, Dec.20, 2012.

[26] http://en.wikipedia.org/wiki/XML_Schema_%28W3C%29, Dec. 20, 2012.
[27] http://en.wikipedia.org/wiki/Robot_control,Dec.20,2012.
[28] http://www.automationml.org/, Dec.20, 2012.
[29] http://www.beremiz.org/documentation/the-plcopen-editor, Dec. 20, 2012.
[30] http://www.java2s.com/Tutorial/Java/0060__Operators/TheinstanceofKeyword.htm, Dec. 20, 2012.
[31] http://www.oracle.com/technetwork/articles/javase/index-140168.html#binsch2, Dec. 20, 2012.
[32] http://www.oracle.com/technetwork/articles/javase/index-140168.html#introjb, Dec. 20, 2012.
[33] http://www.petrinets.info/standard.php, Dec. 20, 2012.
[34] http://www.plcopen.org/, Dec. 20, 2012.
[35] http://www.plcopen.org/pages/tc6_xml/, Dec. 20, 2012.
[36] http://www.plctutor.com/relay-ladder-logic/relay-ladder-logic-contacts-coils.html, Dec. 20, 2012.
[37] http://www.pnml.org/, Dec. 20, 2012.
[38] http://www.w3.org/, Dec. 20, 2012.
[39] http://sourceforge.net/projects/pipe2/files/PIPE%203/, Dec. 20, 2012.
[40] Jones, A. H., Uzam, M., and Ajlouni, N., "Design of Discrete Event Control System for Programmable Logic Controllers Using T-timed Petri Nets," IEEE Int. Symp. Computer-Aided Control System Design, Sep. 15-18, 1996.
[41] Liang, G. R. and Hong, H. M., "Hierarchy Transformation Method for Repetitive Manufacturing System Specification Design, Verification and Implementation," Computer-integrated Manufacturing Systems, Vol. 7, No. 3, pp. 191-205, Aug., 1994.
[42] Milner, A Calculus of Communicating Systems, Springer, 1982.
[43] Murata, T., "Petri Net: Properties, Analysis and Application," Proceedings of the IEEE, Vol. 44, pp. 541-579, 1989.
[44] Oxley, J. G., Matroid Theory, Oxford University Press, 2011.
[45] Shih, S. P. and Meng, C. Z., "Ladder Diagram and Petri-Net-Based Discrete-Event Control Design Methods," IEEE Trans. on System, Man and Cybernetics, Vol. 34, 2004.
[46] Taholakian, A. and Hales, W. M. M., "PN PLC: A Methodology for Designing, Simulating and Coding PLC based Control Systems Using Petri Nets," International Journal of Production Research, vol. 35, pp. 1743-1762, 1997.
[47] Viswanadham, N., Narahari, Y., and Johnson, L., "Deadlock Prevention and Deadlock Avoidance in Flexible Manufacturing Systems Using Petri Net Models," IEEE Tran. Robot Automat, Vol. 6, No. 6, pp. 713-723, 1990.
[48] Zhou, M., McDermott, K., and Patel, P. A., "Petri Net Synthesis and Analysis of a Flexible Manufacturing System Cell," IEEE Systems, Man, and Cybernetics Society, Vol. 23, No. 2, 1993.

附錄：自動轉換程式程式碼

在自動轉換程式中，主要可分為四個類別來執行程式中的各個功能。以下 A. 1.至 A. 4. 為四個程類別的程式碼。第一個為主程式程式碼(Main.java)，主要其功能為設定呼叫介面、輸入檔案、輸出檔案與轉換檔案的事件發生。第二個為產生文件、物件與法則矩陣程式碼(EMG.java)，主要其功能為利用解標籤產生裴氏圖之文件、物件與設定法則矩陣。第三個為產生階梯圖物件、文件與轉換方法程式碼(EMG2LD.java)，主要其功能為利用物件的轉換從裴氏圖之物件產生階梯圖之物件，並利用組標籤產生階梯圖之文件。第四個為程式介面 (GUI.java)，主要其功能為設定使用者介面與視窗的顯示。

A. 1. 主程式程式碼(Main.java)

```
package MainFile;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.List;
import javax.swing.JFileChooser;
import javax.swing.JOptionPane;
import javax.xml.bind.JAXBException;
import derivedClassFromPNML.ArcType;
import derivedClassFromPNML.NameType;
import derivedClassFromPNML.PlaceType;
import derivedClassFromPNML.TransitionType;
import transformtools.EMG2LD;
import transformtools.EMG;
import window.Info;
import GUI.GUI;
public class Main extends GUI {
    public static int bb=0;
     public static void main(String[] args) {
          Main main = new Main();
          main.setVisible(true);//GUI顯現出來
     }

     private EMG pn;
     private EMG2LD emg2ld;
     protected void closeFile() {

          PNArea.setText("");
          LDArea.setText("");
          menuOpen.setEnabled(true);
          menuClose.setEnabled(false);
          menuOutput.setEnabled(false);
          transformButton.setEnabled(false);
     }
     protected void exitFile() {
          JOptionPane.setDefaultLocale(null);//setLocationRelativeTo(null);
          int option = JOptionPane.showConfirmDialog(this,
                    "確定離開程式?","提示",JOptionPane.YES_NO_OPTION);

          if(option == JOptionPane.YES_OPTION)
               {
                    System.exit(0);
               }
               else{
                    return ;
               }
     }
```

```
	public static List<PlaceType> place1 = new ArrayList();
	public static List<TransitionType> transition1 = new ArrayList();
	public static List<ArcType> arc1 = new ArrayList();
	public static List<PlaceType> place2 = new ArrayList();
	public static List<TransitionType> transition2 = new ArrayList();
	public static List<ArcType> arc2 = new ArrayList();
	public static List<PlaceType> place3 = new ArrayList();
	public static List<TransitionType> transition3 = new ArrayList();
	public static List<ArcType> arc3 = new ArrayList();
	public static List<PlaceType> place4 = new ArrayList();
	public static List<TransitionType> transition4 = new ArrayList();
	public static List<ArcType> arc4 = new ArrayList();
	protected void openFile() {
		// 顯示檔案選取的對話方塊
		int option = fileChooser.showOpenDialog(fileChooser);
		// 使用者按下確認鍵
		if (option == JFileChooser.APPROVE_OPTION) {
				file = fileChooser.getSelectedFile();
				System.out.println("擷取  " + file.getAbsolutePath() + " 資料");
				pn = new EMG();
				try{
					pn.unMarshall(file);//解編  進入EMG
					pn.EMG1234();
					if(bb==4){
					pn.plusEMG();
					pn.setRuleMatrix();
					//print法則矩陣
					int matrix[][] = pn.getRuleMatrix();
					String ruleMatrixInfo="";
					for(int column=0; column<matrix[0].length; column++){
						String placeInfo =
"\t"+((NameType)((PlaceType)pn.getPlace().get(column)).getName()).getValue();//法則矩陣中的暫存點列表
						ruleMatrixInfo+=placeInfo;
					}
					for(int row=0; row<matrix.length; row++){
						String
transitionInfo="\n\n"+((NameType)((TransitionType)pn.getTransition().get(row)).getName()).getValue();
						ruleMatrixInfo+=transitionInfo;

						for(int column=0;column<matrix[row].length;column++){
							String element = "\t" + matrix[row][column];
							ruleMatrixInfo+=element;
						}
					}
					new Info(ruleMatrixInfo,"法則矩陣");
					PNArea.setText(pn.getXMLString());

					if(pn.checkInitialTokens()){
						transformButton.setEnabled(true);
						menuOutput.setEnabled(true);
					}else{
						LDArea.setText("此歐氏記號圖未設定初始浮標，無法進行轉換。");
					}
					}}
				catch (FileNotFoundException e) {
						PNArea.setText(pn.getXMLString());
						PNArea.setText("此文件不是一個正確的PNML檔，無法進行轉換。");
					} catch (JAXBException e) {

						PNArea.setText(pn.getXMLString());
						PNArea.setText("此文件不是一個正確的PNML檔，無法進行轉換。");
					}
				menuClose.setEnabled(true);
		}
	}
```

```
protected void outputFile() {
           int option = JOptionPane.showConfirmDialog(this,
                      "開始輸出檔案?","提示",JOptionPane.YES_NO_OPTION);
           if(option == JOptionPane.YES_OPTION){
                      option = fileChooser2.showSaveDialog(fileChooser);
                      if (option == JFileChooser.APPROVE_OPTION) {
                      file = fileChooser2.getSelectedFile();
                      try {
                            emg2ld.Marshaller(file);
                            JOptionPane.showMessageDialog(this,"檔案匯出完成","訊息
",JOptionPane.INFORMATION_MESSAGE);
                      } catch (FileNotFoundException e) {
                            e.printStackTrace();
                            JOptionPane.showMessageDialog(this,"檔案匯出失敗","訊息
",JOptionPane.INFORMATION_MESSAGE);
                      } catch (JAXBException e) {
                            e.printStackTrace();
                            JOptionPane.showMessageDialog(this,"檔案匯出失敗","訊息
",JOptionPane.INFORMATION_MESSAGE);
                      }
                      }
                }
     }
     //轉換程式
     protected void transform(){
                emg2ld = new EMG2LD(pn);
                emg2ld.transform();
                LDArea.setText(emg2ld.getXMLString());
                transformButton.setEnabled(false);

     }
}
```

A. 2.產生文件、物件與法則矩陣程式碼(EMG.java)

```
package transformtools;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import javax.xml.bind.JAXBContext;
import javax.xml.bind.JAXBException;
import javax.xml.bind.Unmarshaller;
import MainFile.Main;
import derivedClassFromPNML.ArcType;
import derivedClassFromPNML.InitialMarkingType;
import derivedClassFromPNML.NameType;
import derivedClassFromPNML.NetType;
import derivedClassFromPNML.PlaceType;
import derivedClassFromPNML.Pnml;
import derivedClassFromPNML.PositionType;
import derivedClassFromPNML.TransitionType;
import window.Info;
public class EMG {
     //採用多型的方式設定ArrayList,以便存取不同類別或primative資料
     private List<PlaceType> place = new ArrayList();
     private List<TransitionType> transition = new ArrayList();
     private List<ArcType> arc = new ArrayList();
     private List<Object> list = new ArrayList<Object>();
     private List<PlaceType> place5 = new ArrayList();
     private List<TransitionType> transition5 = new ArrayList();
     private List<ArcType> arc5 = new ArrayList();
     private String XMLString;
     private int ruleMatrix[][];
     public void unMarshall(File PNML) throws JAXBException, FileNotFoundException {
               JAXBContext jaxbContext = JAXBContext.newInstance("derivedClassFromPNML");
               Unmarshaller unMarshaller = jaxbContext.createUnmarshaller();
               Pnml pnmlElement = (Pnml) unMarshaller.unmarshal(PNML);
               NetType nettype = pnmlElement.getNet();
               BufferedReader input = new BufferedReader(new FileReader(PNML));
               StringBuffer buffer = new StringBuffer();
               ArrayList<String> ms=new ArrayList<String>();
               Main.bb=Main.bb+1;
               System.out.println(Main.bb);
               try {
                    String text;
                    XMLString = "";
                    while ((text = input.readLine()) != null) {
                         XMLString = XMLString + text + "\n";
                    }
               } catch (IOException e) {
                    e.printStackTrace();
               }
               list = nettype.getPlaceOrTransitionOrArc();
               for (int i = 0; i < list.size(); i++) {
                    if (list.get(i) instanceof PlaceType) {
                         place.add((PlaceType) list.get(i));
                         PlaceType p=(PlaceType) list.get(i);
     ms.add("place");
     ms.add("id="+p.getId());
     List<PositionType> posis=p.getGraphics().getPosition();
     ms.add("\t\tx="+posis.get(0).getX());
     ms.add("\t\ty="+posis.get(0).getY());
     InitialMarkingType init=(InitialMarkingType)p.getInitialMarking();
     ms.add("\t\t\tinitial vale="+init.getValue());
     ms.add("\t\t\tinitial x="+init.getGraphics().getOffset().getX());
     ms.add("\t\t\tinitial y="+init.getGraphics().getOffset().getY());
     ms.add("\t\t\tname x="+p.getName().getGraphics().getOffset().getX());
     ms.add("\t\t\tname y="+p.getName().getGraphics().getOffset().getY());
     ms.add("\t\t\tname valuey="+p.getName().getValue());
                    }
```

```
if (list.get(i) instanceof TransitionType) {
                        transition.add((TransitionType) list.get(i));
                        TransitionType t=(TransitionType) list.get(i);
                        ms.add("transition");
                        ms.add("id="+t.getId());
                        List<PositionType> posis=t.getGraphics().getPosition();
                        ms.add("\t\tx="+posis.get(0).getX());
          ms.add("\t\ty="+posis.get(0).getY());
          ms.add("\t\t\tname x="+t.getName().getGraphics().getOffset().getX());
          ms.add("\t\t\tname y="+t.getName().getGraphics().getOffset().getY());
          ms.add("\t\t\tname valuey="+t.getName().getValue());
                    }
                    if (list.get(i) instanceof ArcType) {
                        arc.add((ArcType) list.get(i));
                        ArcType a=(ArcType) list.get(i);
                        ms.add("arc");
                        ms.add("id="+a.getId());
                        ms.add("\tinscription value="+a.getInscription().getValue());
                        ms.add("\tsource="+a.getSource());
                        ms.add("\ttarget="+a.getTarget());
                    }
               }
               if(Main.bb==1){
               new Info(ms,"EMG1物件資訊");
               }
               if(Main.bb==2){
                    new Info(ms,"EMG2物件資訊");
               }
               if(Main.bb==3){
                    new Info(ms,"EMG3物件資訊");
               }
               if(Main.bb==4){
                    new Info(ms,"EMG4物件資訊");
               }
     }
     public void EMG1234()  {
          if(Main.bb==1){
               for (int i = 0; i < place.size(); i++) {
                    Main.place1.add((PlaceType) place.get(i));
               }
               for (int i = 0; i < transition.size(); i++) {
                    Main.transition1.add((TransitionType) transition.get(i));
               }
               for (int i = 0; i < arc.size(); i++) {
                    Main.arc1.add((ArcType) arc.get(i));
               }
          }
          if(Main.bb==2){
                    for (int i = 0; i < place.size(); i++) {
                         Main.place2.add((PlaceType) place.get(i));
                    }
                    for (int i = 0; i < transition.size(); i++) {
                         Main.transition2.add((TransitionType) transition.get(i));
                    }
                    for (int i = 0; i < arc.size(); i++) {
                         Main.arc2.add((ArcType) arc.get(i));
                    }
          }
          if(Main.bb==3){
                    for (int i = 0; i < place.size(); i++) {
                         Main.place3.add((PlaceType) place.get(i));
                    }
                    for (int i = 0; i < transition.size(); i++) {
                         Main.transition3.add((TransitionType) transition.get(i));
                    }
                    for (int i = 0; i < arc.size(); i++) {
                         Main.arc3.add((ArcType) arc.get(i));
                    }
          }
```

```
      if(Main.bb==4){
                        for (int i = 0; i < place.size(); i++) {
                              Main.place4.add((PlaceType) place.get(i));
                        }
                        for (int i = 0; i < transition.size(); i++) {
                              Main.transition4.add((TransitionType) transition.get(i));
                        }
                        for (int i = 0; i < arc.size(); i++) {
                              Main.arc4.add((ArcType) arc.get(i));
                        }
            }
      }
      public void plusEMG()  {
            for (int i = 0; i < Main.place1.size(); i++) {
               place5.add((PlaceType) Main.place1.get(i));
            }
            for (int i = 0; i < Main.place2.size(); i++) {
               place5.add((PlaceType) Main.place2.get(i));
            }
            for (int i = 0; i < Main.place3.size(); i++) {
               place5.add((PlaceType) Main.place3.get(i));
            }
            for (int i = 0; i < Main.place4.size(); i++) {
               place5.add((PlaceType) Main.place4.get(i));
            }
            for (int i = 0; i < Main.transition1.size(); i++) {
                transition5.add((TransitionType) Main.transition1.get(i));
            }
            for (int i = 0; i < Main.transition2.size(); i++) {
                transition5.add((TransitionType) Main.transition2.get(i));
            }
            for (int i = 0; i < Main.transition3.size(); i++) {
                transition5.add((TransitionType) Main.transition3.get(i));
            }
            for (int i = 0; i < Main.transition4.size(); i++) {
                transition5.add((TransitionType) Main.transition4.get(i));
            }
            for (int i = 0; i < Main.arc1.size(); i++) {
                arc5.add((ArcType) Main.arc1.get(i));
            }
            for (int i = 0; i < Main.arc2.size(); i++) {
                arc5.add((ArcType) Main.arc2.get(i));
            }
            for (int i = 0; i < Main.arc3.size(); i++) {
                arc5.add((ArcType) Main.arc3.get(i));
            }
            for (int i = 0; i < Main.arc4.size(); i++) {
                arc5.add((ArcType) Main.arc4.get(i));
            }
      }
      public String getXMLString() {
            return XMLString;
      }
      public List getArc() {
            return arc5;
      }
      public List getPlace() {
            return place5;
      }
      public List getTransition() {
            return transition5;
      }
```

```
//合併法則矩陣
      public void setRuleMatrix() {
            ruleMatrix = new int[transition5.size()][place5.size()];
            String source = "", target = "";
            for (int i = 0; i < Main.arc1.size(); i++) {
                  source = Main.arc1.get(i).getSource();
                  target = Main.arc1.get(i).getTarget();
                  for (int j = 0; j < Main.place1.size(); j++) {
                        // place的input_arc
                        if (target.equals(place5.get(j).getId())) {
                              for (int k = 0; k < Main.transition1.size(); k++) {
                                    if (source.equals(transition5.get(k).getId())) {
                                          ruleMatrix[k][j]++;
                                    }
                              }
                        }
                        // place的output_arc
                        if (source.equals(place5.get(j).getId())) {
                              for (int k = 0; k < Main.transition1.size(); k++) {
                                    if (target.equals(transition5.get(k).getId())) {
                                          ruleMatrix[k][j]--;
                                    }
                              }
                        }
                  }
            }
////////////////////////////////2222////////////////////////////////
            for (int i = 0; i < Main.arc2.size(); i++) {
                  source = Main.arc2.get(i).getSource();
                  target = Main.arc2.get(i).getTarget();for (int j =  Main.place1.size(); j <
Main.place1.size()+Main.place2.size(); j++) {
                        // place的input_arc
                        if (target.equals(place5.get(j).getId())) {
                              for (int k = Main.transition1.size(); k <
Main.transition1.size()+Main.transition2.size(); k++) {
                                    if (source.equals(transition5.get(k).getId())) {
                                          ruleMatrix[k][j]++;
                                    }
                              }
                        }
                        // place的output_arc
                        if (source.equals(place5.get(j).getId())) {
                              for (int k = Main.transition1.size(); k <
Main.transition1.size()+Main.transition2.size(); k++) {
                                    if (target.equals(transition5.get(k).getId())) {
                                          ruleMatrix[k][j]--;
                                    }
                              }
                        }
                  }
            }
////////////////////////////////3333////////////////////////////////
            for (int i = 0; i < Main.arc3.size(); i++) {
                  source = Main.arc3.get(i).getSource();
                  target = Main.arc3.get(i).getTarget();
                  for (int j =   Main.place1.size()+Main.place2.size(); j <
Main.place1.size()+Main.place2.size()+Main.place3.size(); j++) {
                        // place的input_arc
                        if (target.equals(place5.get(j).getId())) {
                              for (int k = Main.transition1.size()+Main.transition2.size(); k <
Main.transition1.size()+Main.transition2.size()+Main.transition3.size(); k++) {
                                    if (source.equals(transition5.get(k).getId())) {
                                          ruleMatrix[k][j]++;
                                    }
                              }
                        }
                        // place的output_arc
                        if (source.equals(place5.get(j).getId())) {
                              for (int k = Main.transition1.size()+Main.transition2.size(); k <
Main.transition1.size()+Main.transition2.size()+Main.transition3.size(); k++) {
                                    if (target.equals(transition5.get(k).getId())) {
                                          ruleMatrix[k][j]--;
                                    }
                              }
                        }
```

```
}
		}
////////////////////////////44444//////////////////////////////
		for (int i = 0; i < Main.arc4.size(); i++) {
			source = Main.arc4.get(i).getSource();
			target = Main.arc4.get(i).getTarget();
			for (int j = Main.place1.size()+Main.place2.size()+Main.place3.size(); j < place5.size();
j++) {
				// place的input_arc
				if (target.equals(place5.get(j).getId())) {
					for (int k =
Main.transition1.size()+Main.transition2.size()+Main.transition3.size(); k < transition5.size() ; k++) {
						if (source.equals(transition5.get(k).getId())) {
							ruleMatrix[k][j]++;
						}
					}
				}
				// place的output_arc
				if (source.equals(place5.get(j).getId())) {
					for (int k =
Main.transition1.size()+Main.transition2.size()+Main.transition3.size(); k < transition5.size(); k++) {
						if (target.equals(transition5.get(k).getId())) {
							ruleMatrix[k][j]--;
						}
					}
				}
			}
		}
	}
	public int[][] getRuleMatrix(){
		return ruleMatrix;
	}

	public boolean checkInitialTokens(){
		int n=0;
		for(int column=0; column < place.size(); column++){
			String initoken=place.get(column).getInitialMarking().getValue();
			if(initoken.substring(initoken.indexOf(",")+1).equals("1")){
				n++;
			}
		}
		if(n==0){
			return false;
		}else
			return true;
	}
	public void reset() {
		this.arc = null;
		this.place = null;
		this.transition = null;
		System.out.println("物件資訊清空");
	}

}
```

A. 3.產生階梯圖物件、文件與轉換方法程式碼(EMG2LD.java)

```
package transformtools;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.OutputStream;
import java.io.StringWriter;
import java.math.BigDecimal;
import java.math.BigInteger;
import java.util.ArrayList;
import java.util.GregorianCalendar;
import java.util.List;
import javax.xml.bind.JAXBContext;
import javax.xml.bind.JAXBException;
import javax.xml.bind.Marshaller;
import javax.xml.bind.PropertyException;
import javax.xml.datatype.DatatypeConfigurationException;
import javax.xml.datatype.DatatypeConstants;
import javax.xml.datatype.DatatypeFactory;
import javax.xml.datatype.XMLGregorianCalendar;
import MainFile.Main;
import derivedClassFromPLCopenXML.Body;
import derivedClassFromPLCopenXML.Connection;
import derivedClassFromPLCopenXML.ConnectionPointIn;
import derivedClassFromPLCopenXML.ConnectionPointOut;
import derivedClassFromPLCopenXML.ObjectFactory;
import derivedClassFromPLCopenXML.Position;
import derivedClassFromPLCopenXML.PouType;
import derivedClassFromPLCopenXML.Project;
import derivedClassFromPLCopenXML.StorageModifierType;
import derivedClassFromPLCopenXML.Body.LD;
import derivedClassFromPLCopenXML.Body.SFC.Coil;
import derivedClassFromPLCopenXML.Body.SFC.Contact;
import derivedClassFromPLCopenXML.Body.SFC.LeftPowerRail;
import derivedClassFromPLCopenXML.Body.SFC.RightPowerRail;
import derivedClassFromPLCopenXML.Project.ContentHeader;
import derivedClassFromPLCopenXML.Project.FileHeader;
import derivedClassFromPLCopenXML.Project.Instances;
import derivedClassFromPLCopenXML.Project.Types;
import derivedClassFromPLCopenXML.Project.ContentHeader.CoordinateInfo;
import derivedClassFromPLCopenXML.Project.ContentHeader.CoordinateInfo.Fbd;
import derivedClassFromPLCopenXML.Project.ContentHeader.CoordinateInfo.Ld;
import derivedClassFromPLCopenXML.Project.ContentHeader.CoordinateInfo.Sfc;
import derivedClassFromPLCopenXML.Project.Instances.Configurations;
import derivedClassFromPLCopenXML.Project.Types.DataTypes;
import derivedClassFromPLCopenXML.Project.Types.Pous;
import derivedClassFromPLCopenXML.Project.Types.Pous.Pou;
import derivedClassFromPLCopenXML.Project.Types.Pous.Pou.Interface;
import derivedClassFromPLCopenXML.Project.Types.Pous.Pou.Interface.LocalVars;
import derivedClassFromPNML.InitialMarkingType;
import derivedClassFromPNML.NameType;
import derivedClassFromPNML.PlaceType;
import derivedClassFromPNML.TransitionType;
import window.Info;
public class EMG2LD {
private EMG pn;
private final int Height = 15;
private final int Width = 21;
private final int lprX=50,lprY=30;
ArrayList<String> syplace = new ArrayList<String>();
private String XMLString;
private Marshaller marshaller;
private Project project;
     public EMG2LD(EMG pn){
          this.pn = pn;
     }
```

```
public void Marshaller(File xmlDocument) throws FileNotFoundException, JAXBException{
         OutputStream os = new FileOutputStream(xmlDocument);
           marshaller.setProperty("jaxb.formatted.output",Boolean.TRUE);
           marshaller.marshal(project, os);
     }
     public void transform(){
     JAXBContext jaxbContext;
     try {
           jaxbContext = JAXBContext.newInstance("derivedClassFromPLCopenXML");//JAXBContext類別的實體化
           marshaller=jaxbContext.createMarshaller();
           ObjectFactory factory = new ObjectFactory();

           project=factory.createProject();
           //設定FileHeader  BERE 開新檔案 設基本資訊
           FileHeader fh = factory.createProjectFileHeader();
           fh.setCompanyName("MB606");
           fh.setProductName("Transformed Ladder Diagram");
           fh.setProductVersion("1.0");
           DatatypeFactory f = DatatypeFactory.newInstance();
           GregorianCalendar g = new GregorianCalendar();
           XMLGregorianCalendar c = f.newXMLGregorianCalendar(g);
           c.setTimezone(DatatypeConstants.FIELD_UNDEFINED);
        fh.setCreationDateTime(c);
           project.setFileHeader(fh);
           //設定ContentHeader,與Fbd,Ld,Sfc
           ContentHeader ch = factory.createProjectContentHeader();
           ch.setName("Ladder Diagram");
           ch.setModificationDateTime(c);
           CoordinateInfo ci = factory.createProjectContentHeaderCoordinateInfo();
           Fbd fbd = factory.createProjectContentHeaderCoordinateInfoFbd();
           Ld ld = factory.createProjectContentHeaderCoordinateInfoLd();
           Sfc sfc = factory.createProjectContentHeaderCoordinateInfoSfc();
           derivedClassFromPLCopenXML.Project.ContentHeader.CoordinateInfo.Fbd.Scaling fbdScale =
factory.createProjectContentHeaderCoordinateInfoFbdScaling();
           derivedClassFromPLCopenXML.Project.ContentHeader.CoordinateInfo.Ld.Scaling ldScale =
factory.createProjectContentHeaderCoordinateInfoLdScaling();
           derivedClassFromPLCopenXML.Project.ContentHeader.CoordinateInfo.Sfc.Scaling sfcScale =
factory.createProjectContentHeaderCoordinateInfoSfcScaling();
           fbdScale.setX(new BigDecimal(0));
           fbdScale.setY(new BigDecimal(0));
           ldScale.setX(new BigDecimal(0));
           ldScale.setY(new BigDecimal(0));
           sfcScale.setX(new BigDecimal(0));
           sfcScale.setY(new BigDecimal(0));
           fbd.setScaling(fbdScale);
           ld.setScaling(ldScale);
           sfc.setScaling(sfcScale);
           ci.setFbd(fbd);
           ci.setLd(ld);
           ci.setSfc(sfc);
           ch.setCoordinateInfo(ci);
           project.setContentHeader(ch);

           //設定Instance
           Instances instances = factory.createProjectInstances();
           Configurations config = factory.createProjectInstancesConfigurations();
           instances.setConfigurations(config);
           project.setInstances(instances);
           Types types = factory.createProjectTypes();
           DataTypes dataTypes = factory.createProjectTypesDataTypes();
           types.setDataTypes(dataTypes);
           Pous pous = factory.createProjectTypesPous();
           Pou pou = factory.createProjectTypesPousPou();
           pou.setName("temp");
           pou.setPouType(PouType.FUNCTION);
           Interface interf = factory.createProjectTypesPousPouInterface();
           LocalVars lv = factory.createProjectTypesPousPouInterfaceLocalVars();
           interf.getLocalVarsOrTempVarsOrInputVars().add(lv);
           pou.setInterface(interf);
           Body body = factory.createBody();
           LD bodyLd = factory.createBodyLD();
           int[][] matrix = pn.getRuleMatrix();
```

```
//column讀取暫存點  ROW讀取轉移點
            ArrayList<String> placeY = new ArrayList<String>();
            for(int column=0;column<matrix[0].length;column++){
                  String place = ((NameType)((PlaceType)pn.getPlace().get(column)).getName()).getValue();
                  if(place.length()==2){
                        place="Y00"+place.charAt(1);//變Y002  從0開始
                  }else{
                        place="Y0"+place.substring(1,3);
                  }
                  placeY.add(place);
            }
            //存取有initialtoken的暫存點
            ArrayList<String> n=new ArrayList<String>();
            for(int column=0;column<matrix[0].length;column++){
                  String
initoken=((InitialMarkingType)((PlaceType)pn.getPlace().get(column)).getInitialMarking()).getValue();
                  String iniPlace=placeY.get(column);
                  if(initoken.substring(initoken.indexOf(",")+1).equals("1"))//比較","之後是0或1
                        n.add(iniPlace);
            }
            for(int i=0;i<n.size();i++){
                  for(int j=0;j<n.size();j++){
                        if(i!=j){
                        if(n.get(i).equals(n.get(j))){
                              n.remove(j);//刪掉重覆的初始place
                        }
                        }
                  }
            }
            //產生左電軌物件
            int id=1;
            LeftPowerRail leftPowerRail=factory.createBodySFCLeftPowerRail();
            leftPowerRail.setLocalId(new BigInteger(String.valueOf(id)));
            Position leftp=factory.createPosition();
            leftp.setX(new BigDecimal(lprX));
            leftp.setY(new BigDecimal(lprY));
            leftPowerRail.setPosition(leftp);
            leftPowerRail.setHeight(new BigDecimal(70));
            leftPowerRail.setWidth(new BigDecimal(Width));
            int positionX=121;
            int positionY=42;
            boolean first=true;//判斷是不是第一個接點  來跟左電軌連接
            List<Contact> contactList=new ArrayList<Contact>();//第一個接點都會放到這LIST裡面
            List<Coil> coilList=new ArrayList<Coil>();//存放所有線圈
            //設定法則Ri
            Contact initialContact=generateContact(100, positionY, 100, positionY+8, first, 1 , ++id ,
                        "X000", factory);
            //initialContact.setNegated(true);
            contactList.add(initialContact);
            bodyLd.getCommentOrErrorOrConnector().add(initialContact);
            boolean second=false;//看線圈是不是有跨層
            int last_positionY=positionY;//last都是最後接點的位置y
            int last_Id=id;//最後接點的id
                        for(int i=0;i<n.size();i++){//n是初始暫存點 設定初始狀態線圈
                  Coil coil=generateCoil(positionY, positionY+8, second, positionX, last_positionY+8,
                              last_Id, ++id, n.get(i), factory);
                  coil.setStorage(StorageModifierType.SET);
                  bodyLd.getCommentOrErrorOrConnector().add(coil);
                  coilList.add(coil);//他是第一個接點
                  positionY+=40;//y=y+40
                  second=true;//因為他有跨行
            }
            int a=0;
            for(int column=0;column<placeY.size();column++){
                  for(int row=0;row<matrix.length;row++){
                        a=a+matrix[row][column];
                  }
                  if(a<0){
                        syplace.add(placeY.get(column));//找出控制place
                  }
                  a=0;
            }
```

```
//////////設定同步裴氏圖
      Contact X001=generateContact(100, positionY, 71, positionY+8, true, id++ , id ,
                  "X001", factory);
      bodyLd.getCommentOrErrorOrConnector().add(X001);
      contactList.add(X001);//他是第一個接點
      Contact X002=generateContact(150, positionY, 121, positionY+8, false, id++ , id ,
                  "X002", factory);
      bodyLd.getCommentOrErrorOrConnector().add(X002);
      last_Id=id;
      String Y1=syplace.get(0);
            Coil coil1=generateCoil(positionY, positionY+8, second, 171, positionY+8,
                        last_Id, ++id, Y1, factory);
            bodyLd.getCommentOrErrorOrConnector().add(coil1);
            coilList.add(coil1);
            positionY+=40;
            second=false;
            ////////////////////////////////////
            Contact X0001=generateContact(100, positionY, 71, positionY+8, true, id++ , id ,
                        "X001", factory);
            //syContact.setNegated(true);
            bodyLd.getCommentOrErrorOrConnector().add(X0001);
            contactList.add(X0001);
            Contact X0002=generateContact(150, positionY, 121, positionY+8, false, id++ , id ,
                        "X002", factory);
            X0002.setNegated(true);
            bodyLd.getCommentOrErrorOrConnector().add(X0002);
            last_Id=id;
            String Y2=syplace.get(1);
                  Coil coil11=generateCoil(positionY, positionY+8, second, 171, positionY+8,
                              last_Id, ++id, Y2, factory);
                  //coil.setStorage(StorageModifierType.SET);
                  bodyLd.getCommentOrErrorOrConnector().add(coil11);
                  coilList.add(coil11);
                  positionY+=40;//y=y+40
                  second=false;
                  Contact X1=generateContact(100, positionY, 71, positionY+8, true, id++ , id ,
                              "X001", factory);
                  X1.setNegated(true);
                  bodyLd.getCommentOrErrorOrConnector().add(X1);
                  contactList.add(X1);
                  Contact X2=generateContact(150, positionY, 121, positionY+8, false, id++ , id ,
                              "X002", factory);
                  bodyLd.getCommentOrErrorOrConnector().add(X2);
                  last_Id=id;
                  String Y3=syplace.get(2);
                  Coil coil111=generateCoil(positionY, positionY+8, second, 171, positionY+8,
                              last_Id, ++id, Y3, factory);
                  bodyLd.getCommentOrErrorOrConnector().add(coil111);
                  coilList.add(coil111);
                  positionY+=40;//y=y+40
                  second=false;
                  Contact X01=generateContact(100, positionY, 71, positionY+8, true, id++ , id ,
                              "X001", factory);
                  X01.setNegated(true);
                  bodyLd.getCommentOrErrorOrConnector().add(X01);
                  contactList.add(X01);
                  Contact X02=generateContact(150, positionY, 121, positionY+8, false, id++ , id ,
                              "X002", factory);
                  X02.setNegated(true);
                  bodyLd.getCommentOrErrorOrConnector().add(X02);
                  last_Id=id;
                  String Y4=syplace.get(3);

                  Coil coil1111=generateCoil(positionY, positionY+8, second, 171, positionY+8,
                              last_Id, ++id, Y4, factory);
                  //coil.setStorage(StorageModifierType.SET);
                  bodyLd.getCommentOrErrorOrConnector().add(coil1111);
                  coilList.add(coil1111);
                  positionY+=40;//y=y+40
                  second=false;
```

```
//設定轉移點觸發條件
            for(int row=0;row<matrix.length;row++){
                  first = true;
                  positionX=100;
                  for(int column=0;column<matrix[row].length;column++)
                  if(matrix[row][column]==-1){//如果-1的話 產生一個接點
                        String inputPlace=placeY.get(column);//暫存點的名字
                        Contact contact=generateContact(positionX, positionY, positionX-29, positionY+8
                        , first, id++, id, inputPlace, factory);
                        bodyLd.getCommentOrErrorOrConnector().add(contact);
                        if(first){
                              contactList.add(contact);
                        }
                        first=false;
                        positionX+=50;
                  }
                  last_Id=id;
                  second=false;
String transitionName=((NameType)((TransitionType)pn.getTransition().get(row)).getName()).getValue();
            Coil coil=
generateCoil(positionY,positionY+8,second,positionX-29,positionY,last_Id,++id,transitionName,factory);
                  bodyLd.getCommentOrErrorOrConnector().add(coil);
                  coilList.add(coil);
                  positionY+=40;
                  //設定轉移點觸發狀態
                  Contact
contact=generateContact(100,positionY,100,positionY+8,true,1,++id,transitionName,factory);
                  bodyLd.getCommentOrErrorOrConnector().add(contact);
                  contactList.add(contact);
                  second=false;
                  last_Id=id;
                  last_positionY=positionY;
                  for(int column=0;column<matrix[row].length;column++){
                        if(matrix[row][column]==-1){
                              String inputPlace=placeY.get(column);
                              coil=generateCoil(positionY, positionY+8, second, 121,
                                          last_positionY+8, last_Id, ++id, inputPlace, factory);
                              coil.setStorage(StorageModifierType.RESET);
                              bodyLd.getCommentOrErrorOrConnector().add(coil);
                              coilList.add(coil);
                              positionY+=40;
                        }else if(matrix[row][column]==1){
                              String outputPlace=placeY.get(column);
                              coil=generateCoil(positionY, positionY+8, second, 121,
                                          last_positionY+8, last_Id, ++id, outputPlace, factory);
                              coil.setStorage(StorageModifierType.SET);
                              bodyLd.getCommentOrErrorOrConnector().add(coil);
                              coilList.add(coil);
                              positionY+=40;
                        }
                        second=true;
                  }
            }
            leftPowerRail.setHeight(new BigDecimal(positionY));
                  for(int i=0;i<contactList.size();i++){
                        int y=contactList.get(i).getPosition().getY().intValue();
                        Body.SFC.LeftPowerRail.ConnectionPointOut lprOut =
                        factory.createBodySFCLeftPowerRailConnectionPointOut();
                        Position lprRelPosition=factory.createPosition();
                        lprRelPosition.setX(new BigDecimal(21));
                        lprRelPosition.setY(new BigDecimal(y+8-lprY));
                        lprOut.setRelPosition(lprRelPosition);
                        lprOut.setFormalParameter("");
                        leftPowerRail.getConnectionPointOut().add(lprOut);
                  }
                  bodyLd.getCommentOrErrorOrConnector().add(leftPowerRail);
```

```
                //右電軌
                RightPowerRail rightPowerRail = factory.createBodySFCRightPowerRail();
                Position rightp = factory.createPosition();
                rightp.setX(new BigDecimal(500));
                rightp.setY(new BigDecimal(30));
                rightPowerRail.setPosition(rightp);
                for(int i=0;i<coilList.size();i++){
                      int x=coilList.get(i).getPosition().getX().intValue();
                      int y=coilList.get(i).getPosition().getY().intValue();
                      BigInteger ref_id=coilList.get(i).getLocalId();
                      ConnectionPointIn rprIn = factory.createConnectionPointIn();
                      Position rprelPosition = factory.createPosition();
                      rprelPosition.setX(new BigDecimal(0));
                      rprelPosition.setY(new BigDecimal(y+8-lprY));
                      rprIn.setRelPosition(rprelPosition);
                      Connection rprConnection = factory.createConnection();
                      rprConnection.setRefLocalId(ref_id);
                      Position rprconnectionE=factory.createPosition();
                      rprconnectionE.setX(new BigDecimal(500));
                      rprconnectionE.setY(new BigDecimal(y+8));
                      rprConnection.getPosition().add(rprconnectionE);
                      Position rprconnectionS=factory.createPosition();
                      rprconnectionS.setX(new BigDecimal(x+Width));
                      rprconnectionS.setY(new BigDecimal(y+8));
                      rprConnection.getPosition().add(rprconnectionS);
                      rprIn.getConnection().add(rprConnection);
                      rightPowerRail.getConnectionPointIn().add(rprIn);
                }
                rightPowerRail.setHeight(new BigDecimal(positionY));
                rightPowerRail.setWidth(new BigDecimal(Width));
                rightPowerRail.setLocalId(new BigInteger(String.valueOf(++id)));
                bodyLd.getCommentOrErrorOrConnector().add(rightPowerRail);
          //建立物件資訊
          ArrayList<String> rs=new ArrayList<String>();
          List ls=bodyLd.getCommentOrErrorOrConnector();
          for(Object o:ls){
                if(o instanceof Contact){
                      Contact co=(Contact)o;
                      List<Connection> conns=co.getConnectionPointIn().getConnection();
                      rs.add("contact");
                      rs.add("\t\theight="+co.getHeight());
                      rs.add("\t\tlocalId="+co.getLocalId());
                      rs.add("\t\tnegated="+co.isNegated());
                      rs.add("\t\tvariable="+co.getVariable());
                      rs.add("\t\twidth="+co.getWidth());
     rs.add("\t\t\tConnectionPointOut relPosition x="+co.getConnectionPointOut().getRelPosition().getX());
     rs.add("\t\t\tConnectionPointOut relPosition x="+co.getConnectionPointOut().getRelPosition().getY());
     rs.add("\t\t\tConnectionPointIn connection reflocalId="+conns.get(0).getRefLocalId());
     rs.add("\t\t\t\tConnectionPointIn connection position x="+conns.get(0).getPosition().get(0).getX());
     rs.add("\t\t\t\tConnectionPointIn connection position y="+conns.get(0).getPosition().get(0).getY());
                }

if(o instanceof Coil){
Coil coi=(Coil)o;
List<Connection> conns=coi.getConnectionPointIn().getConnection();
rs.add("coil");
rs.add("\t\theight="+coi.getHeight());
rs.add("\t\tlocalId="+coi.getLocalId());
rs.add("\t\tvariable="+coi.getVariable());
rs.add("\t\twidth="+coi.getWidth());
rs.add("\t\t\tConnectionPointIn connection reflocalId="+conns.get(0).getRefLocalId());
rs.add("\t\t\t\tConnectionPointIn connection position x="+conns.get(0).getPosition().get(0).getX());
rs.add("\t\t\t\tConnectionPointIn connection position y="+conns.get(0).getPosition().get(0).getY());
rs.add("\t\t\tConnectionPointOut relPosition x="+coi.getConnectionPointOut().getRelPosition().getX());
rs.add("\t\t\tConnectionPointOut relPosition x="+coi.getConnectionPointOut().getRelPosition().getY());
                }
```

```
                if(o instanceof LeftPowerRail){
                      LeftPowerRail lf=(LeftPowerRail)o;
                      rs.add("LeftPowerRail");
                      rs.add("\theight="+lf.getHeight());
                      rs.add("\tlcoalId="+lf.getLocalId());
                      rs.add("\twidth="+lf.getWidth());
                      rs.add("\t\tPosition X="+lf.getPosition().getX());
                      rs.add("\t\tPosition Y="+lf.getPosition().getY());
rs.add("\t\tconnectionPointOut FormalParameter="+lf.getConnectionPointOut().get(0).getFormalParameter());
rs.add("\t\t\tconnectionPointOut RelPosiotion
X="+lf.getConnectionPointOut().get(0).getRelPosition().getX());
rs.add("\t\t\tconnectionPointOut RelPosiotion
Y="+lf.getConnectionPointOut().get(0).getRelPosition().getY());
                }
                if(o instanceof RightPowerRail){
                      RightPowerRail lf=(RightPowerRail)o;
                      rs.add("RightPowerRail");
                      rs.add("\theight="+lf.getHeight());
                      rs.add("\tlcoalId="+lf.getLocalId());
                      rs.add("\twidth="+lf.getWidth());
                      rs.add("\t\tPosition X="+lf.getPosition().getX());
                      rs.add("\t\tPosition Y="+lf.getPosition().getY());
                      rs.add("\t\tconnectionPointIn Connection
RefLoaclId="+lf.getConnectionPointIn().get(0).getConnection().get(0).getRefLocalId());
                      rs.add("\t\t\tconnectionPointIn Connection Position
X="+lf.getConnectionPointIn().get(0).getConnection().get(0).getPosition().get(0).getX());
                      rs.add("\t\t\tconnectionPointIn Connection Position
Y="+lf.getConnectionPointIn().get(0).getConnection().get(0).getPosition().get(0).getY());
                }
          }
          new Info(rs,"階梯圖物件資訊");
          //設定bodyLd,types,pous,types
          body.setLD(bodyLd);
          pou.getBody().add(body);
          pous.getPou().add(pou);
          types.setPous(pous);
          project.setTypes(types);
          marshaller.setProperty("jaxb.formatted.output",Boolean.TRUE);
          StringWriter st = new StringWriter();
          marshaller.marshal(project, st);
          XMLString = st.toString();
     }
     catch (JAXBException e) {
          e.printStackTrace();
     }catch (DatatypeConfigurationException e) {
          e.printStackTrace();
      }
     }

private Contact generateContact(int positionX,int positionY,int connectionX,int connectionY
                ,boolean first,int ref_id,int _id,String name,ObjectFactory factory){//X位置 Y位置 接點X位
置 接點Y的位置  是否為第一個接點
          Position cpiRelp = factory.createPosition();
          cpiRelp.setX(new BigDecimal("0"));
          cpiRelp.setY(new BigDecimal("8"));
          ConnectionPointOut cpo = factory.createConnectionPointOut();
          Position cpoRelp = factory.createPosition();
          cpoRelp.setX(new BigDecimal("21"));
          cpoRelp.setY(new BigDecimal("8"));
          cpo.setRelPosition(cpoRelp);
          Contact contact=factory.createBodySFCContact();
          contact.setHeight(new BigDecimal(Height));
          contact.setWidth(new BigDecimal(Width));
          contact.setConnectionPointOut(cpo);
          Position contactPosition=new Position();
          contactPosition.setX(new BigDecimal(positionX));
          contactPosition.setY(new BigDecimal(positionY));
          contact.setPosition(contactPosition);
          ConnectionPointIn connectionPointIn = factory.createConnectionPointIn();
          connectionPointIn.setRelPosition(cpiRelp);
```

```
if(first){
                Connection contactConnection=new Connection();
                Position conE=new Position();
                conE.setX(new BigDecimal(positionX));
                conE.setY(new BigDecimal(connectionY));
                contactConnection.getPosition().add(conE);
                Position conS=new Position();
                conS.setX(new BigDecimal(71));
                conS.setY(new BigDecimal(connectionY));
                contactConnection.getPosition().add(conS);
                contactConnection.setRefLocalId(new BigInteger(String.valueOf(1)));
                connectionPointIn.getConnection().add(contactConnection);
                contact.setConnectionPointIn(connectionPointIn);
                contact.setLocalId(new BigInteger(String.valueOf(_id)));
                contact.setVariable(name);
          }else{
                Connection contactConnection=new Connection();
                Position conE=new Position();
                conE.setX(new BigDecimal(positionX));
                conE.setY(new BigDecimal(connectionY));
                contactConnection.getPosition().add(conE);
                Position conS=new Position();
                conS.setX(new BigDecimal(connectionX));
                conS.setY(new BigDecimal(connectionY));
                contactConnection.getPosition().add(conS);
                contactConnection.setRefLocalId(new BigInteger(String.valueOf(ref_id)));
                connectionPointIn.getConnection().add(contactConnection);
                contact.setConnectionPointIn(connectionPointIn);
                contact.setLocalId(new BigInteger(String.valueOf(_id)));
                contact.setVariable(name);
          }
          return contact;
     }
     private Coil generateCoil(int positionY,int connectionY,boolean second
                ,int lastConnectionX,int lastConnectionY,int ref_Id,int _id,String name,ObjectFactory
factory){
          Position cpiRelp = factory.createPosition();
          cpiRelp.setX(new BigDecimal("0"));
          cpiRelp.setY(new BigDecimal("8"));
          ConnectionPointOut cpo = factory.createConnectionPointOut();
          Position cpoRelp = factory.createPosition();
          cpoRelp.setX(new BigDecimal("21"));
          cpoRelp.setY(new BigDecimal("8"));
          cpo.setRelPosition(cpoRelp);
          Coil coil=factory.createBodySFCCoil();
          ConnectionPointIn coilcpi = factory.createConnectionPointIn();
          coilcpi.setRelPosition(cpiRelp);
          Position coilPosition = factory.createPosition();
          coilPosition.setX(new BigDecimal(450));//position x
          coilPosition.setY(new BigDecimal(positionY));//position y
          coil.setPosition(coilPosition);
          //設定連接座標
          Connection coilConnection = factory.createConnection();
          coilConnection.setRefLocalId(new BigInteger(String.valueOf(ref_Id)));
          if(second){
                Position conECoi=factory.createPosition();
                conECoi.setX(new BigDecimal(450));
                conECoi.setY(new BigDecimal(connectionY));
                coilConnection.getPosition().add(conECoi);
                //connection center1(下) of coil
                Position conC1Coi=factory.createPosition();
                conC1Coi.setX(new BigDecimal(285));
                conC1Coi.setY(new BigDecimal(connectionY));
                coilConnection.getPosition().add(conC1Coi);
                //connection center1(上) of coil
                Position conC2Coi=factory.createPosition();
                conC2Coi.setX(new BigDecimal(285));
                conC2Coi.setY(new BigDecimal(lastConnectionY));
                coilConnection.getPosition().add(conC2Coi);
                Position conSCoi=factory.createPosition();
                conSCoi.setX(new BigDecimal(lastConnectionX));
                conSCoi.setY(new BigDecimal(lastConnectionY));
                coilConnection.getPosition().add(conSCoi);
                coilcpi.getConnection().add(coilConnection);
```

```
	}else{
				Position conECoi=factory.createPosition();
				conECoi.setX(new BigDecimal(450));
				conECoi.setY(new BigDecimal(connectionY));
				coilConnection.getPosition().add(conECoi);
				Position conSCoi=factory.createPosition();
				conSCoi.setX(new BigDecimal(lastConnectionX));
				conSCoi.setY(new BigDecimal(connectionY));
				coilConnection.getPosition().add(conSCoi);
				coilcpi.getConnection().add(coilConnection);
			}
			coil.setLocalId(new BigInteger(String.valueOf(_id)));
			coil.setHeight(new BigDecimal(Height));
			coil.setWidth(new BigDecimal(Width));
			coil.setConnectionPointOut(cpo);
			coil.setConnectionPointIn(coilcpi);
			coil.setVariable(name);//name
			return coil;
	}
	public String getXMLString() {
			return XMLString;
	}

}
```

A. 4.程式介面 (GUI.java)

```
package GUI;
import java.awt.Color;
import java.awt.Container;
import java.awt.Dimension;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.InputEvent;
import java.awt.event.KeyEvent;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.io.File;
import javax.swing.BorderFactory;
import javax.swing.Box;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JOptionPane;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.KeyStroke;
import javax.swing.ScrollPaneConstants;
import javax.swing.UIManager;
public abstract class GUI extends JFrame {
	private Container container;
	private JMenu fileMenu, aboutMenu; //選單
	private JMenuItem menuAbout;
	private JMenuBar menuBar;
	private ImageIcon iconAbout;
	private ImageIcon iconOpen, iconClose, iconExit, iconInfo,iconOutput;
	protected JMenuItem menuOpen, menuClose, menuExit,menuOutput,menuTransform;
	protected JScrollPane jScrollPane, jScrollPane2, jScrollPane3;
	protected JFileChooser fileChooser;//選取
	protected JFileChooser fileChooser2;
	protected File file;
	protected JButton transformButton;
	protected JButton transformButton1;
	protected JTextArea PNArea,LDArea;//白色放文字
	private static boolean useSystemLookAndFeel = true;
	public GUI() {
		super("浮標守恆裴氏圖到階梯圖的自動轉換程式");
		createAndShowGUI();
		initResource();
		setUpUIComponent();
		setUpEventListener();
	}
	protected abstract void openFile();
	protected abstract void closeFile();
	protected abstract void exitFile();
	protected abstract void outputFile();
	protected abstract void transform();
	private void initResource() {
		// 載入圖示檔案
		iconOpen = new ImageIcon(GUI.class.getResource("images/open.gif"));
		iconClose = new ImageIcon(GUI.class.getResource("images/close.gif"));
		iconExit = new ImageIcon(GUI.class.getResource("images/exit.gif"));
		iconAbout = new ImageIcon(GUI.class.getResource("images/about.png"));
		iconInfo = new ImageIcon(GUI.class
				.getResource("images/information.png"));
		iconOutput = new ImageIcon(GUI.class
				.getResource("images/data_output.png"));
	}
```

```
	private void setUpEventListener() {
	//按下視窗關閉鈕事件處理
	addWindowListener(new WindowAdapter() {
		public void windowClosing(WindowEvent e) {
			exitFile();
		}
	});

	//選單-載入檔案
	menuOpen.addActionListener(new ActionListener() {
		public void actionPerformed(ActionEvent e) {
			openFile();
		}
	});
	//選單-退出檔案
	menuClose.addActionListener(new ActionListener() {
		public void actionPerformed(ActionEvent e) {
			closeFile();
		}
	});

	menuOutput.addActionListener(
			new ActionListener() {
				public void actionPerformed(ActionEvent e) {
					outputFile();
				}
			});
	//選單-離開
	menuExit.addActionListener(new ActionListener() {
		public void actionPerformed(ActionEvent e) {
			exitFile();
		}
	});
	//選單-關於
	menuAbout.addActionListener(
			new ActionListener() {
				public void actionPerformed(ActionEvent e) {
					//顯示對話方塊
					JOptionPane.showOptionDialog(null,
							"此軟體非屬任何營利行為，\n"
							+ "為純學術研究開發使用，" + "由交通大學工業工程系學生開發\n"
							,"關於本軟體",
							JOptionPane.DEFAULT_OPTION,
							JOptionPane.INFORMATION_MESSAGE,
							iconAbout, null, null);
				}
			});
	  setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
}


	private void setUpUIComponent() {
	Box box = Box.createHorizontalBox();
	//檔案選取方塊
	fileChooser    = new JFileChooser("c:\\");
	fileChooser.addChoosableFileFilter(new PNMLFileFilter());
	fileChooser2    = new JFileChooser("c:\\");
	container = getContentPane();
	// 選單列
	menuBar = new JMenuBar();
	//設置"檔案"選單
	fileMenu = new JMenu("\u6a94\u6848");
	menuOpen = new JMenuItem("\u8f09\u5165\u6a94\u6848", iconOpen);
```

```
        //快速鍵設置
        menuOpen.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_O,
                InputEvent.CTRL_MASK));
        menuClose = new JMenuItem("退出檔案", iconClose);
        menuClose.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_C,
                InputEvent.CTRL_MASK));
        menuClose.setEnabled(false);
        menuTransform = new JMenuItem("檔案轉換");
        menuOutput = new JMenuItem("匯出檔案",iconOutput);
        menuOutput.setEnabled(false);
        menuExit = new JMenuItem("離開", iconExit);
        menuExit.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_X,
                InputEvent.CTRL_MASK));
        fileMenu.add(menuOpen);
        fileMenu.add(menuClose);
        //fileMenu.add(menuTransform);
        fileMenu.add(menuOutput);
        fileMenu.addSeparator(); //分隔線
        fileMenu.add(menuExit);
        // 設置"關於"選單
        aboutMenu = new JMenu("關於");
        menuAbout = new JMenuItem("關於....", iconInfo);
        aboutMenu.add(menuAbout);
        menuBar.add(fileMenu);
        menuBar.add(aboutMenu);
        setJMenuBar(menuBar);
      transformButton = new JButton("TCPN>>>LD");
      transformButton.setEnabled(false);
      transformButton1 = new JButton("TCPN");
      transformButton1.setEnabled(false);
      transformButton.addActionListener(
      new ActionListener(){
        public void actionPerformed(ActionEvent event){
            transform();
        }
      });
      box.add(getJScrollPane1());
      //box.add(transformButton1);
      box.add(transformButton);
      box.add(getJScrollPane2());
      container.add(box);
        setVisible(true);
        pack(); // 根據默認計算決定視窗大小
        setSize(new Dimension(1000, 700));
        setLocationRelativeTo(null);
    }
    private static void createAndShowGUI() {
        if (useSystemLookAndFeel) {
            try {
                UIManager.setLookAndFeel(UIManager
                        .getSystemLookAndFeelClassName());
            } catch (Exception e) {
                System.err.println("Couldn't use system look and feel.");
            }
        }
    }
    private JScrollPane getJScrollPane1() {
        if (jScrollPane == null) {
            jScrollPane = new JScrollPane(getPNArea(),
                    ScrollPaneConstants.VERTICAL_SCROLLBAR_ALWAYS,
                    ScrollPaneConstants.HORIZONTAL_SCROLLBAR_ALWAYS);
            jScrollPane.setBorder(BorderFactory.createEmptyBorder(0, 0, 0, 0));
        }
        return jScrollPane;
    }
    private JTextArea getPNArea() {
        if (PNArea == null) {
            PNArea = new JTextArea(100,100);
            PNArea.setEnabled(false);
            PNArea.setDisabledTextColor(Color.BLACK);
            //PNDrawPane.setPreferredSize(new Dimension(1000,5000));
        }
        return PNArea;
    }
    private JScrollPane getJScrollPane2() {
```

```
	private JScrollPane getJScrollPane2() {
		if (jScrollPane2 == null) {
			jScrollPane2 = new JScrollPane(getLDArea(),
					ScrollPaneConstants.VERTICAL_SCROLLBAR_ALWAYS,
					ScrollPaneConstants.HORIZONTAL_SCROLLBAR_ALWAYS);
			jScrollPane2.setBorder(BorderFactory.createEmptyBorder(0, 0, 0, 0));
		}
		return jScrollPane2;
	}
	private JTextArea getLDArea() {
		if (LDArea == null) {
			LDArea = new JTextArea(100,100);
			LDArea.setEnabled(false);
			LDArea.setDisabledTextColor(Color.BLACK);
		}
		return LDArea;
	}
}
```