การแปลงเพตริเน็ทโมเดลให้เป็นแลดเดอร์ไดอะแกรม

INTERPRETATION PETRI NET MODEL TO LADDER DIAGRAM

ประยุทธ อินแบน

PRAYUTH INBAN

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชาวิศวกรรมการวัดคุม

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2548

ISBN 974-15-1953-2

INTERPRETATION PETRI NET MODEL TO LADDER DIAGRAM

PRAYUTH INBAN

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR DEGREE OF

MASTER OF ENGINEERING IN INSTRUMENTATION ENGINEERING

SCHOOL OF GRADUATE STUDIES

KING MONGKUL'S INSTITUTE OF TECHNOLOGY LADKRABANG

2005

ISBN 974-15-1953-2

หัวข้อวิทยานิพนธ์	การแปลงเพตริเน็ทโมเดลให้เป็นแลดเดอร์ไดอะแกรม
นักศึกษา	นายประยุทธ อินแบน
รหัสประจำตัว	46061709
ปริญญา	วิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชา	วิศวกรรมการวัดคุม
พ.ศ.	2548
อาจารย์ผู้ควบคุมวิทยานิพนธ์	ผศ.ทวีพล ซื่อสัตย์

บทคัดย่อ

เครื่องควบคุมแบบตรรกะโปรแกรมได้ (PLC) ได้มีการใช้งานอย่างกว้างขวางในทางอุตสาหกรรม โดยมีมาตรฐานในการเขียนโปรแกรมคือ IEC 1131-3 ภาษาแลดเดอร์เป็นภาษาหนึ่งซึ่งได้รับความนิยมสูงที่สุด โดยมีลักษณะเหมือนกับวงจรไฟฟ้าซึ่งไม่มีการจัดลำดับและขั้นตอนที่ชัดเจน จึงเป็นการยากที่จะทำความเข้าใจและพัฒนาโปรแกรม Petri net เป็นตัวอย่างเชิงภาพที่สามารถอธิบายขั้นตอนการทำงานได้ชัดเจนและเข้าใจง่าย ในวิทยานิพนธ์ฉบับนี้ได้นำเสนอวิธี การออกแบบโปรแกรมจากเงื่อนไขการทำงานแบบวงจรแลดเดอร์ให้อยู่ในรูปแบบของ Petri net และได้พัฒนาโปรแกรมคอมพิวเตอร์ เพื่อใช้ศึกษาการออกแบบโปรแกรมในรูปแบบของ Petri net แล้วนำไปควบคุมอุปกรณ์ภายนอกผ่านโมดุลอินพุตและโมดุลเอาท์พุต ของ PLC จากผลการทดลองนำวงจรแลดเดอร์มาแปลงให้อยู่ในรูป Petri net แล้วใช้โปรแกรมดังกล่าวแปลงเป็นคำสั่งเพื่อป้อนให้กับ PLC ปรากฏว่า ผลตอบสนองของ PLC ดีเช่นเดียวกับใช้วงจรแลดเดอร์ แสดงว่าเราสามารถใช้ทฤษฎีของ Petri net ในการออกแบบโปรแกรมควบคุม PLC ได้เป็นอย่างดี

Thesis Title	Interpretation Petri net Model to Ladder Diagram
Students	Mr. Prayuth Inban
Student ID.	46061709
Degree	Master of Engineering
Programme	Instrumentation Engineering
Year	2005
Thesis Advisor	Asst. Prof. Taweepol Suesut

ABSTRACT

The Programmable Logic Controller (PLC) is widely used in industrial by using the IEC 1131-3 programming language. The ladder diagram is the most popular language that similar to the electrical schematic diagrams which does not have the step and sequence of the operation. Therefore it is difficult to understand and develop the program structure. The Petri net is one of graphical models that can be explained the system operation easily and clearly. This thesis presents the computer software that developed by Delphi to transform Petri net to Ladder diagram and implementation to control the processes through PLC's input and output. This software can be applied to another application on PLC as well.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงได้อย่างดี ด้วยคำแนะนำปรึกษาเกี่ยวกับ การแปลงเพตริเน็ทโมเดลให้เป็นแลดเดอร์ไดอะแกรม จาก ผศ.ทวีพล ซื่อสัตย์ ซึ่งเป็นอาจารย์ควบคุมวิทยานิพนธ์ ผู้วิจัยรู้สึกซาบซึ้งในความอนุเคราะห์จากท่านและขอกราบของพระคุณเป็นอย่างสูง

ท้ายที่สุดนี้ ผู้วิจัยของกราบขอบพระคุณ คุณพ่อ สจ. ณรงค์ คุณแม่พิกุล อินแบน ซึ่งเป็นผู้ให้โอกาสทางการศึกษาและกำลังใจกับผู้วิจัยตั้งแต่เริ่มต้นจนสำเร็จสมความตั้งใจ

คุณค่าและประโยชน์อันพึงมีจากวิทยานิพนธ์ฉบับนี้ ผู้วิจัยขอมอบแด่ผู้มีพระคุณทุกท่าน

ประยุทธ อินแบน

สารบัญ

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

สารบัญตาราง

ตารางที่

หน้า

สารบัญรูป

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

บทที่ 1
บทนำ

1.1 ความเป็นมาและเหตุจูงใจของการวิจัย

ในการอธิบายการทำงานของเครื่องจักรกลโดยใช้คำพูดหรือลายลักษณ์อักษรนั้น แม้จะเป็นวิธีที่ง่าย แต่ไม่มีกฎเกณฑ์ที่แน่นอน และไม่สามารถที่จะถ่ายทอดรายละเอียดของการทำงานของเครื่องจักรกลให้ง่ายต่อการทำความเข้าใจได้ ซึ่งถ้าการควบคุมเครื่องจักรกลนั้นมีความยุ่งยากและซับซ้อนมาก ๆ ถ้าอธิบายด้วยคำพูดก็คงจะต้องใช้เวลานานในการทำความเข้าใจ ดังนั้นโดยมากแล้วมักจะอธิบายการทำงานของเครื่องจักรกลในรูปของผังเวลาการทำงาน และโฟว์ชาร์ตเป็นส่วนใหญ่

ปัจจุบันการเขียนวงจรแลดเดอร์เพื่อใช้สำหรับโปรแกรมการทำงานต่าง ๆ ที่ต้องการให้กับเครื่องควบคุมแบบตรรกะที่โปรแกรมได้ (Programmable Logic Controller) มักจะเขียนด้วยวิธีการดัดแปลงมาจากวงจรรีเลย์ หรือเขียนตามผังเวลาการทำงานของเครื่องจักรกล ซึ่งทั้งสองวิธีดังกล่าวไม่มีรูปแบบขั้นตอนที่ชัดเจน โดยส่วนมากแล้วจะขึ้นอยู่กับประสบการณ์ของผู้ใช้งานเอง จึงต้องอาศัยผู้ชำนาญงานมาช่วยในการออกแบบโปรแกรมระบบควบคุม กรณีเกิดเหตุขัดข้องขึ้น หรือต้องการที่จะเปลี่ยนแปลงเงื่อนไขการทำงานของเครื่องจักรเพียงเล็กน้อย หรือในขั้นตอนการเขียนโปรแกรมนั้นเกิดปัญหาขึ้นก็จะประสบกับปัญหาในการตรวจสอบหาจุดบกพร่องของเครื่องจักรหรือจุดบกพร่องภายในวงจรแลดเดอร์ ทำให้เกิดความล่าช้าในการตรวจสอบหาจุดบกพร่องหรือข้อผิดพลาดที่เกิดขึ้น ทั้งนี้เพราะในรูปแบบของวงจรแลดเดอร์ไม่ได้แสดงลำดับขั้นตอนการทำงานของเครื่องจักรกลที่เราควบคุมอยู่ให้เห็นอย่างชัดเจน จากที่ได้ศึกษาทฤษฎีของ Petri nets. ซึ่งจะเป็นการจำลองกระบวนการของระบบการทำงานต่าง ๆ ให้อยู่ในลักษณะของรูปภาพแทนในส่วนต่าง ๆ ของระบบการทำงานที่สนใจ ซึ่งจะเป็นลำดับขั้นตอนของการทำงานที่เห็นได้ชัดเจน จึงช่วยให้ง่ายในการออกแบบโปรแกรมการควบคุมการทำงาน และแก้ไขการทำงานของเครื่องจักรกลได้ง่ายขึ้น และยังสามารถใช้เป็นเครื่องมือเพื่อช่วยในการวิเคราะห์การทำงานของระบบต่าง ๆ ได้ เช่น เกิดการหยุดนิ่งของระบบการทำงาน การวนซ้ำ เป็นต้น

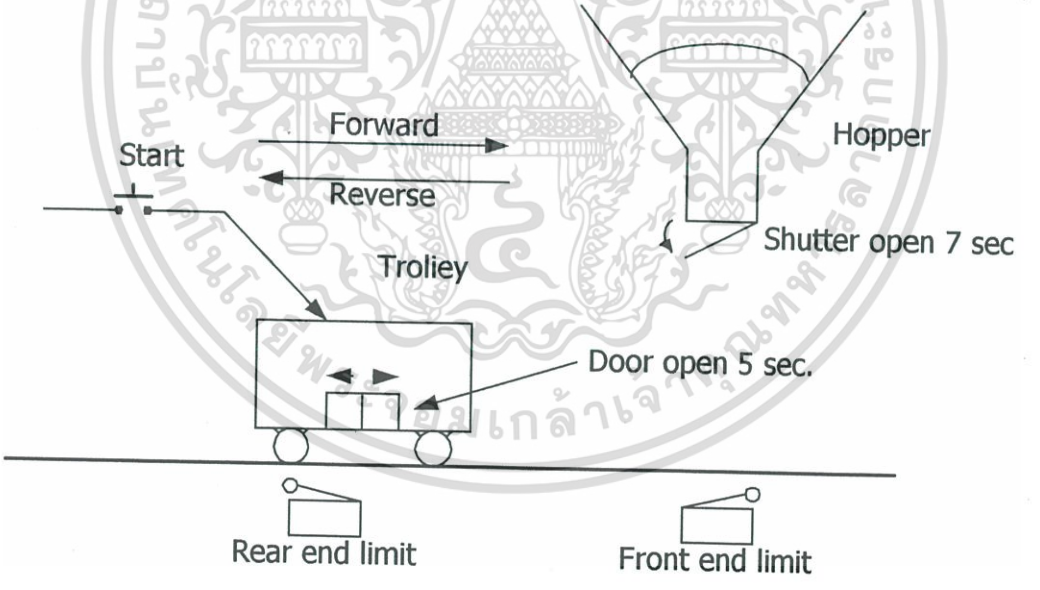
จากที่เราได้กล่าวถึงการอธิบายระบบการทำงานของเครื่องจักรกลในแบบต่าง ๆ มาแล้วนั้น เพื่อให้เห็นชัดเจนยิ่งขึ้นดังนั้น จึงยกตัวอย่างการอธิบายระบบการทำงานในรูปแบบต่าง ๆ เพื่อใช้ในการเปรียบเทียบ

ตัวอย่างที่ 1.1 ระบบขนถ่ายสินค้าด้วยรถเลื่อน

การอธิบายด้วยลายลักษณ์อักษร

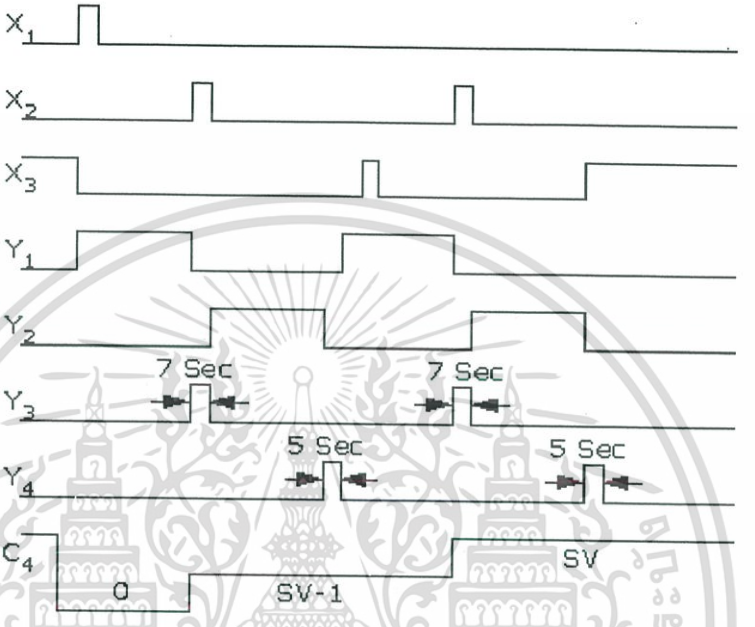
ลำดับการทำงานของเครื่องจักรมีเงื่อนไขการทำงานดังนี้

1. เมื่อกดสวิทช์เริ่มทำงานตัวรถบรรทุกของจะเลื่อนไปทางด้านขวามือและหยุดอยู่ในตำแหน่งใต้ถังเก็บของ
2. ฝาถังเก็บของเปิดออกเพื่อปล่อยสิ่งของออกมานาน 7 วินาที
3. หลังจากนั้นให้รถบรรทุกของกลับมาทางด้านซ้ายมือ แล้วเปิดประตูปล่อยของออกเป็นเวลานาน 5 วินาที
4. ตัวรถบรรทุกจะเลื่อนกลับไปรับสิ่งของเพื่อขนถ่ายสินค้า 2 รอบแล้วหยุด



รูปที่ 1.1 โครงสร้างภายนอกของเครื่องจักร

การอธิบายโดยผังเวลาการทำงาน



รูปที่ 1.2 แสดงผังเวลาในการทำงาน

X1 คือ สวิตช์เริ่มต้น

X2 คือ ลิมิตสวิตช์ด้านขวา

X3 คือ ลิมิตสวิตช์ด้านซ้าย

Y1 คือ เอาท์พุตสั่งเลื่อนไปทางขวา

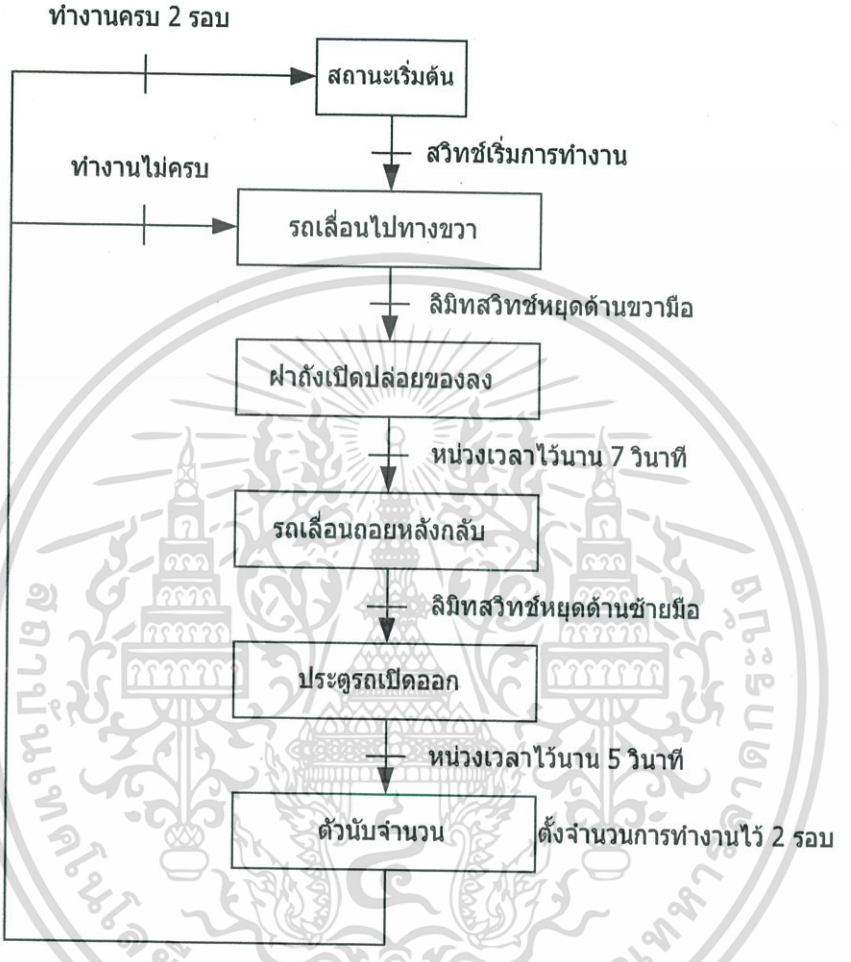
Y2 คือ เอาท์พุตสั่งเลื่อนไปทางซ้าย

Y3 คือ เอาท์พุตสั่งเปิดฝาถัง

Y4 คือ เอาท์พุตสั่งเปิดประตูรถ

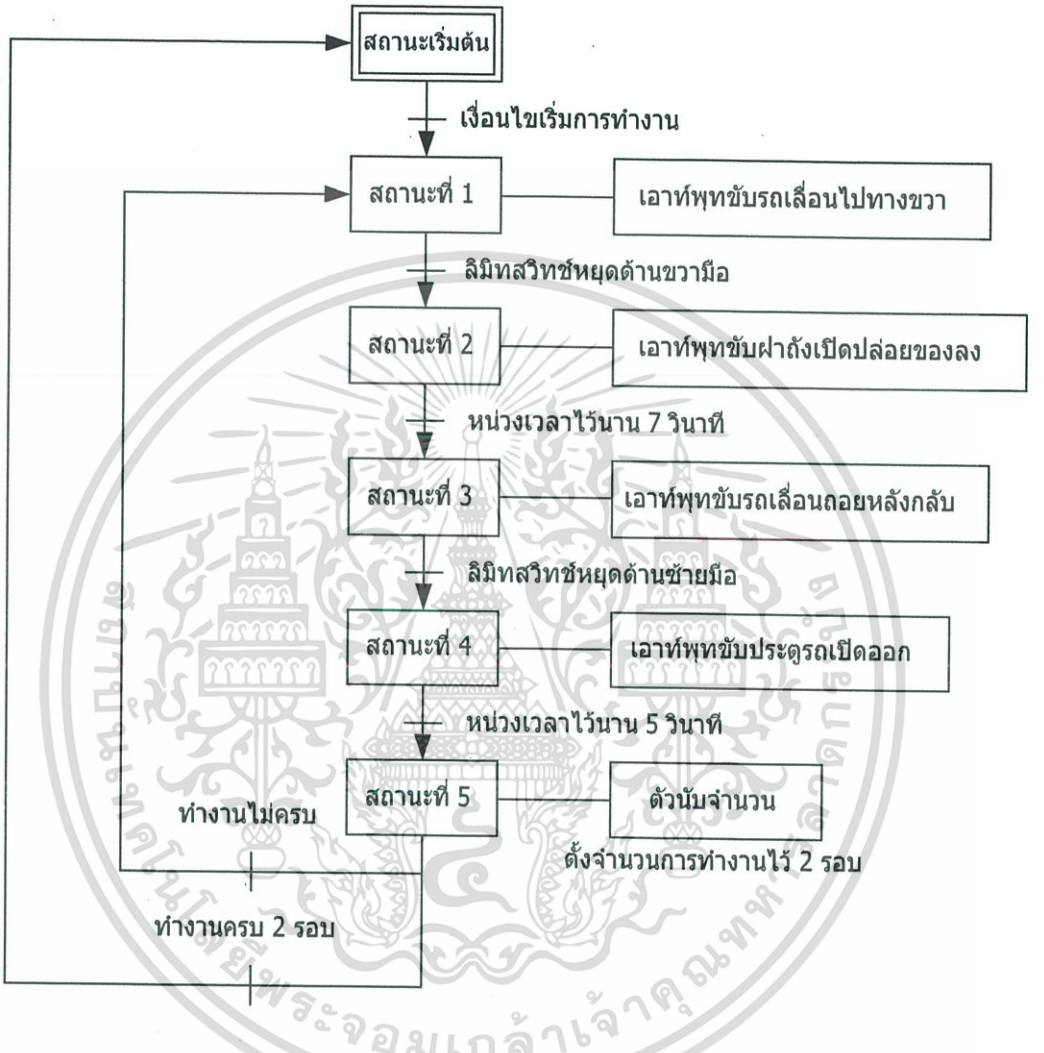
C4 คือ ตัวนับจำนวน(Counter)

การอธิบายโดยแผนผังการทำงาน



รูปที่ 1.3 แสดงแผนผังการทำงาน

การอธิบายโดย Grafcet



รูปที่ 1.4 รูปแบบของ Grafcet

การอธิบายโดยวงจรแลดเดอร์

รูปที่ 1.5 รูปแบบของวงจรแลดเดอร์

จากการอธิบายการทำงานของเครื่องจักรแบบต่าง ๆ นั้นจะเห็นได้ว่าในรูปแบบของวงจรแลดเดอร์ไม่มีลำดับขั้นตอนการทำงานของเครื่องจักรกลเลย ซึ่งแตกต่างกับรูปแบบของ Gratect ซึ่งเป็นทฤษฎีที่ได้ประยุกต์มาจาก Petri Nets. และมีลักษณะรูปแบบที่ช่วยให้เราเห็นภาพลำดับขั้นตอนการทำงานที่ชัดเจนและง่ายต่อการทำความเข้าใจ

1.2 วัตถุประสงค์ของวิทยานิพนธ์

ในวิทยานิพนธ์นี้จะเป็นการศึกษาเกี่ยวกับทฤษฎีของ Petri Nets. และวงจรแลดเดอร์ของเครื่องควบคุมแบบตรรกะที่โปรแกรมได้(Programmable Logic Controller) โดยมีวัตถุประสงค์ที่จะนำทฤษฎีของ Petri Nets. มาประยุกต์ใช้เพื่อช่วยในการเขียนโปรแกรมการควบคุมที่ง่ายขึ้น ลดเวลาและความยุ่งยากในการเขียนโปรแกรมที่ต้องการจะนำไปใช้ควบคุมเครื่องจักรกลให้ทำงานตามที่ต้องการ กล่าวคือ สามารถที่จะเขียนวงจรแลดเดอร์โดยการเขียนจากการอธิบายการทำงานของระบบการทำงานของเครื่องจักรกลในแบบต่าง ๆ ให้อยู่ในรูปแบบของ Petri Net หลังจากนั้น จึงแปลงจากรูปแบบของ Petri Nets ให้อยู่ในรูปแบบของวงจรแลดเดอร์เพื่อนำไปใช้ในการป้อนโปรแกรมให้กับเครื่องควบคุมแบบตรรกะที่โปรแกรมได้ต่อไป และเขียนโปรแกรมเพื่อให้ผู้ใช้งานเขียนโปรแกรมควบคุมอุปกรณ์ภายนอกและสามารถจำลองเหตุการณ์ในรูปแบบของ Grafcet

1.3 ขอบเขตของวิทยานิพนธ์

1) สามารถที่เขียนวงจรแลดเดอร์โดยการเขียนเป็น Petri Nets จากเงื่อนไขการทำงานแล้วทำการแปลงจาก Petri Net. มาเป็นวงจรแลดเดอร์ เพื่อใช้ในการโปรแกรมให้กับเครื่องควบคุมต่อไป
2) เขียนโปรแกรมเพื่อให้ผู้ใช้งานเขียนโปรแกรมในการรับค่าสถานะทางลอจิกจากอุปกรณ์ภายนอกผ่านทางอินพุตโมดุลของเครื่องควบคุม นำค่าที่ได้มาประมวลผลโดยโปรแกรมบนเครื่องคอมพิวเตอร์หลังจากนั้นส่งผลที่ได้จากการประมวลผลไปควบคุมอุปกรณ์ภายนอกผ่านทางเอาท์พุตของเครื่องควบคุม
3) สามารถตรวจสอบความผิดพลาดของโปรแกรมที่ผู้ใช้งานเขียนขึ้นมาได้
4) สามารถจำลองการทำงานของโปรแกรมที่ผู้ใช้งานเขียนขึ้นมาได้

บทที่ 2
ทฤษฎีและหลักการที่เกี่ยวข้อง

2.1 ทฤษฎีของเพททริเนทส์ (Petri net.)

ทฤษฎีด้าน Petri netที่ยกมากล่าวนี้ได้มาจากการรวบรวม และศึกษาจากหนังสือ บทความ และเวปไซต์ ที่ได้มีอยู่ เราได้รวบรวมนำเนื้อหาบางส่วนที่เกี่ยวกับการทำปริญญานิพนธ์เพื่อเขียนเป็นทฤษฎีในส่วนนี้ด้วย

2.1.1 ส่วนประกอบของเพททริเนทส์เบื้องต้น

จากที่ได้อธิบายไปแล้วว่า Petri net เป็นแบบจำลองทางด้านกราฟฟิกสำหรับระบบต่าง ๆ โดยการสร้างและใช้แบบจำลองแทนส่วนประกอบต่าง ๆ ในระบบที่เราสนใจ ดังนั้นแบบจำลองนี้จะประกอบด้วยส่วนต่าง ๆ และการใช้งานดังนี้

1. Places ใช้สัญลักษณ์เป็นวงกลม Places สามารถแบ่งเป็น
   1.1 input places แทนเงื่อนไขเริ่มต้น(Pre Condition) สำหรับเหตุการณ์หนึ่ง ๆ
   1.2 output places แทนเงื่อนไขที่ได้ (Post Condition) จากการทำเหตุการณ์หนึ่ง ๆ ในแต่ละ Place จะมีพร๊อพเพอร์ตี้ อยู่ได้แก่
   - Capacity ระบุจำนวนสูงสุดของ Token ที่ Place สามารถเก็บไว้ได้
   - Marking ระบุจำนวน Token ที่ Place มีอยู่

2. Transitions ใช้สัญลักษณ์เป็นสี่เหลี่ยมด้านขนานหรือ จตุรัสแทนเหตุการณ์ของระบบในแต่ละ Transition จะมีพร๊อพเพอร์ตี้อยู่ได้แก่
   - Priority กำหนดความสำคัญของ Transition นั้น โดยค่า 0 จะมีค่าสูงสุด
   - Probability กำหนดความน่าจะเป็นของการเลือก Transition นั้น โดยระบบ

3. Token ใช้สัญลักษณ์เป็นจุดสีดำซึ่งจะ ปรากฏเป็นจุดสีดำเราจะแทนเงื่อนไขที่จะพิจารณาด้วยสัญลักษณ์ของ Token ดังนั้น Token จะอยู่ใน Place ซึ่งจะหมายถึงมีเงื่อนไขตามจำนวนของ Tokens เข้ามาในเหตุการณ์ (Transition)

4. Arcs ใช้สัญลักษณ์เป็นเส้นที่เชื่อมต่อระหว่าง Transitions และ Places ซึ่งจะบอกทิศทางของเงื่อนไขกับเหตุการณ์ Arcs แบ่งเป็น 2 ประเภทตามทิศทาง

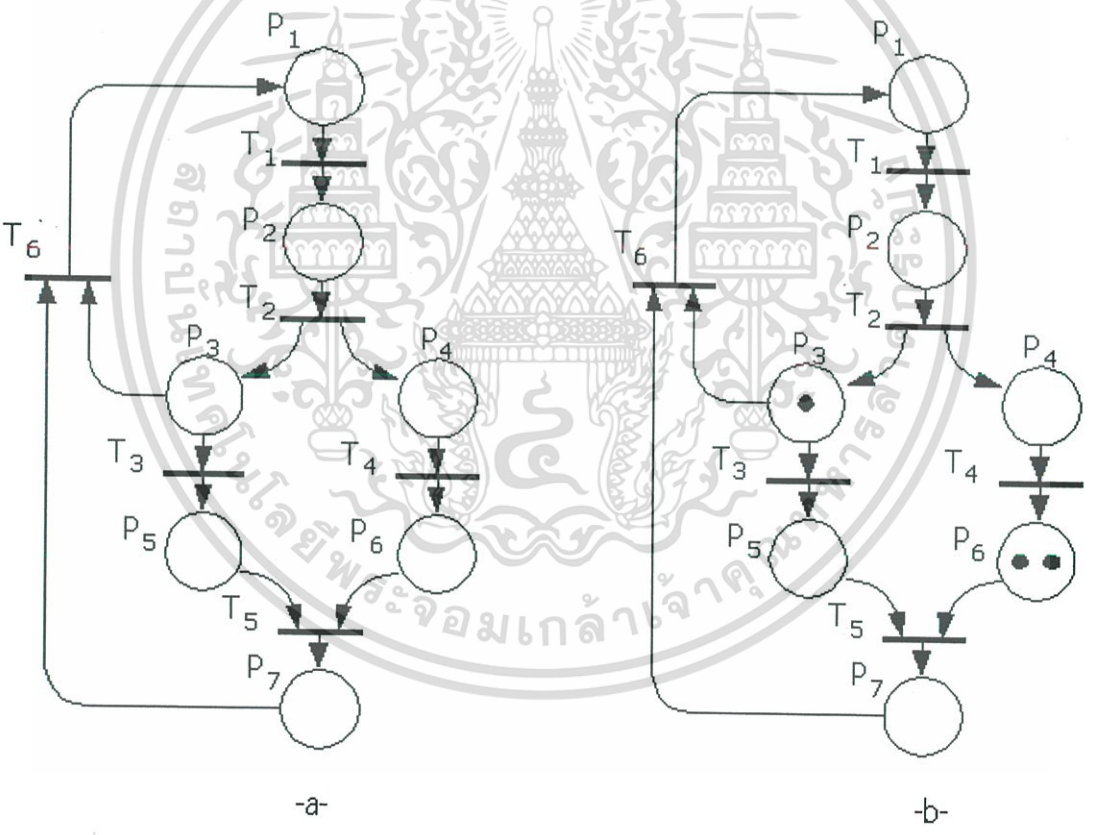
4.1 PreArcs มีทิศทางจาก Place ไป Transition

มี Arc Weight w(p,t) ระบุจำนวน Token ที่ต้องการสำหรับการผ่านในแต่ละครั้ง

4.2 PostArcs มีทิศทางจาก Transition ไป Place

มี Arc Weight w(t,p) ระบุจำนวน Token ที่จะเกิดขึ้นสำหรับการผ่าน Arc นี้ในแต่ละครั้ง

จากรูปที่ 2.1 Place จะแทนโดยรูปวงกลม Transition จะแทนด้วยบัสหรือกล่องสี่เหลี่ยม โดยที่ Place และ Transition จะเชื่อมต่อกันโดย arcs ซึ่งใน arcs นี้ก็จะมีอยู่ 2 รูปแบบ คือ arc ที่ต่อจาก Place ไป Transition และจาก Transition ไป Place



รูปที่ 2.1 a) Not marked Petri net. B) marked Petri net.

ในรูปที่ 2.1 จะมี 7 Place, 6 Transition และ 15 arc เราจะเรียกเซ็ตของ Place ว่า P. สำหรับตัวอย่างนี้จะมี P = {P1,P2,P3,P4,P5,P6,P7} เซ็ตของ Transition คือ T = {T1,T2,T3,T4,T5,T6} Place P 3 เรียกว่า Place ต้นทาง หรือ input Place ของ T3 เพราะเป็นการเชื่อมต่อเข้าโดยตรงจาก P3 ไป T3 Place P5 เป็น Place ปลายทางหรือ output Place ของ T3 เนื่องจากเป็น Place ที่ถูกเชื่อมจาก T3 ไป P5 และในทางเดียวกัน Place และ Transition อื่น ๆ ที่มีรูปแบบดังกล่าวก็สามารถที่จะเรียกได้เหมือนกัน ส่วนใน Transition ที่ไม่มี input Place เรียกว่า Source Transition และ Transition ที่ปราศจาก output Place จะเรียกว่า Sink Transition

2.1.2 การทำเครื่องหมาย (Marking)

ในรูปที่ 2.1b จะเป็นการแสดงถึงการ Marked ของ Petri net ซึ่งแต่ละ Placeจะบรรจุด้วย ค่าจำนวนเต็มบวก ซึ่งเรียกว่า Token หรือ Marks ซึ่งจะบรรจุอยู่ใน place Pi เราจะสามารถเรียกว่า M(Pi) หรือ mi สำหรับในตัวอย่างในรูปที่ 2.1b นั้นเราจะได้ว่า m1 = m3 = 1 , m6 = 2 และ m2 = m4 = m5 = m7 = 0 ซึ่งข่ายของการ marking (M) จะสามารถกำหนดโดยเวคเตอร์ได้ดังนี้คือ

M = (m1,m2,m3,m4,m5,m6,m7) ซึ่งจากรูปที่ 2.1b ก็จะได้ว่า

M = (1,0.1,0,0,2,0) โดยที่ marking นี้สามารถเปลี่ยนแปลงได้โดยการ firing ของ Transition ซึ่งการ firing เราก็จะได้นำเสนอในหัวข้อต่อไป

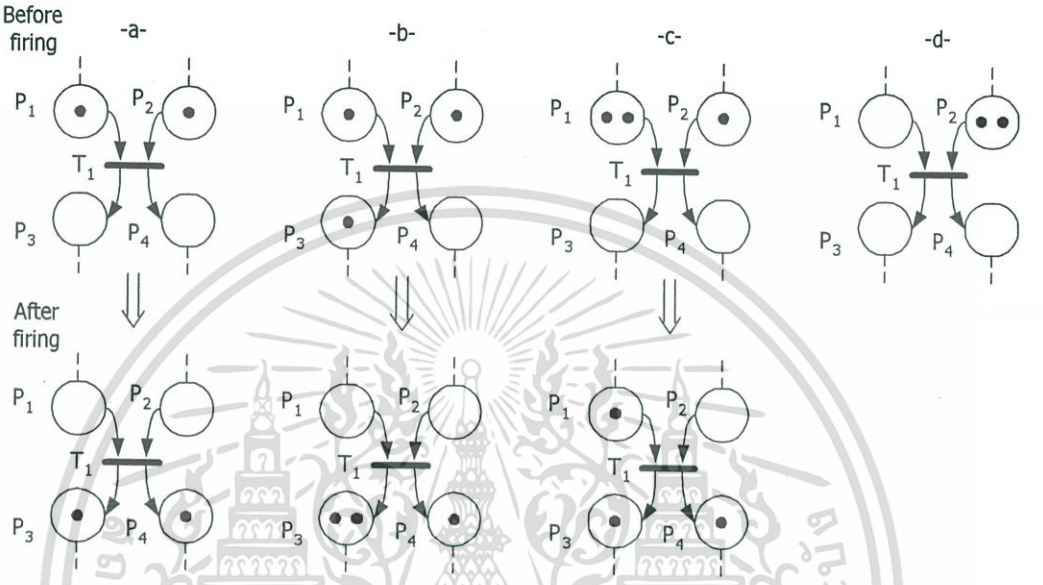
2.1.3 การส่งผ่าน Tokens ของทรานซิสชั่น (Firing of a Transition)

หมายความว่าการเคลื่อนที่และการเปลี่ยนแปลงของ Tokens ที่อยู่ใน Places แรกและจะเคลื่อนที่ไปสู่อีก Places อื่น โดยผ่าน Transition เทียบกับระบบได้ว่าการมีเงื่อนไขที่พร้อม (จำนวน token ใน places แรก) สำหรับการเกิดเหตุการณ์ (transition) และผลที่ได้จากการเกิดเหตุการณ์จะเป็นอย่างไร (จำนวน token ที่เกิดขึ้นใน place อื่น) นั้นมีเงื่อนไขอย่างไร

1. transition t จะบอกได้ว่า enable (พร้อมที่จะทำ firing) เมื่อทุก input place p ของ t มีจำนวน tokens อย่างน้อย w(p,t) เท่ากับ weight ของ arcs จาก p ไป t (Pre Arcs)
2. transition ที่ enable จะทำ firing หรือไม่ก็ได้
3. การ fire สำหรับ enable transition t จะย้าย token จำนวน w(p,t) สำหรับแต่ละ input place p ของ t และจะสร้าง tokens จำนวน w(t,p) ให้กับแต่ละ output place p ของ t ซึ่ง w (t,p) คือ weight ของแต่ละ arc จาก t ไป p

จากรูปที่ 2.2 a, 2.2b , 2.2c (ก่อนการfiring) มันจะ enabled เพราะในแต่ละ Place P1,P2 มี Token อย่างน้อยที่สุดเท่ากับหนึ่งทั้งนั้น ส่วนในรูปที่ 2.2 d นั้นไม่ enable เพราะ P1 ไม่มี Token

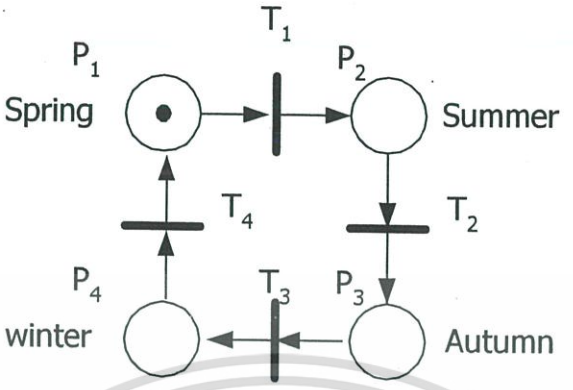
การ firing ของ Transition Tj จะประกอบไปด้วยการนำ Token จาก input place ไปยัง output place โดยจะต้องคำนึงถึงของกำหนดของ arc ที่ใช้ในการส่งผ่าน Transition และกฎการ enable ของ Transition ด้วย ซึ่งจะแสดงให้เห็นได้ดังรูปที่ 2.2



รูปที่ 2.2 Firing of Transition

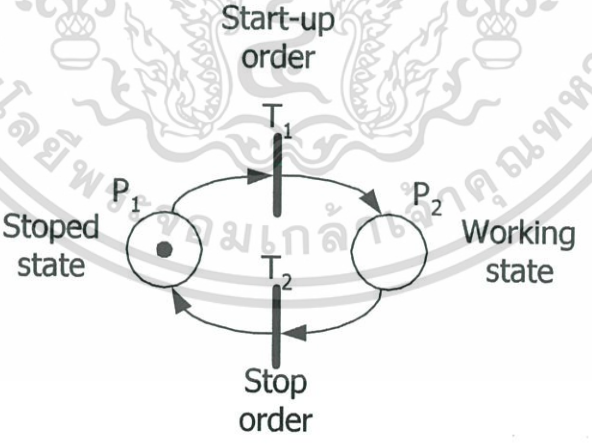
2.1.4 ระบบอัตโนมัติและระบบไม่อัตโนมัติ(Autonomous and non autonomous Petri net.)

จากรูปที่ 2.3 เป็นรูปของช่วงฤดูการณ์ซึ่งแต่ละ Place นั้นจะเป็นฤดูกาล และ ฤดูกาลนี้มันก็จะเปลี่ยนไปได้โดยการที่ รอให้ Transition enable ซึ่งมันจะเปลี่ยนจากฤดูกาลหนึ่งไปยังอีกฤดูกาลเมื่อเราพิจารณาพฤติกรรมของการเปลี่ยนแปลงฤดูกาลนี้ มันจะค่อยเปลี่ยนแปลงขึ้นและจะทำให้เกิด Transition enable ขึ้นและ firing ไปตามกาลเวลาทันทีซึ่งการเกิดเหตุการณ์ลักษณะนี้เราจะสามารถพูดได้ว่ามันเป็นพฤติกรรมของ Autonomous Petri net



รูปที่ 2.3 Autonomous Petri net

รูปที่ 2.4 จะเป็นการแสดงวงรอบการทำงานของระบบมอเตอร์ในการเริ่มต้นทำงานและหยุดทำงาน โดยจะเริ่มจากอยู่ที่ตำแหน่งหยุดการทำงานไปตำแหน่งเริ่มทำงาน และกลับมายังจุดหยุดการทำงานอีกครั้ง ในรูปที่ 2.4 นั้น สถานะเริ่มต้นนั้นจะมี Marking อยู่ที่ สถานะหยุดการทำงานซึ่งมันจะเริ่มทำงานได้ก็ต่อเมื่อ Transition ของการสตาร์ท (T1) enable แต่ T1 จะ enable ได้นั้นจะอาศัยการสั่งงานมาจากภายนอก ซึ่งพฤติกรรมแบบนี้จะเรียกได้ว่าเป็น non-autonomous Petri net



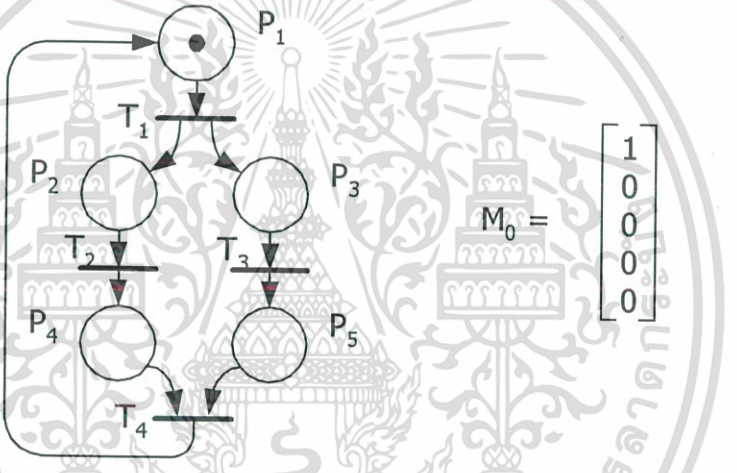
รูปที่ 2.4 non-autonomous Petri net

2.1.5 สัญลักษณ์และคำจำกัดความ

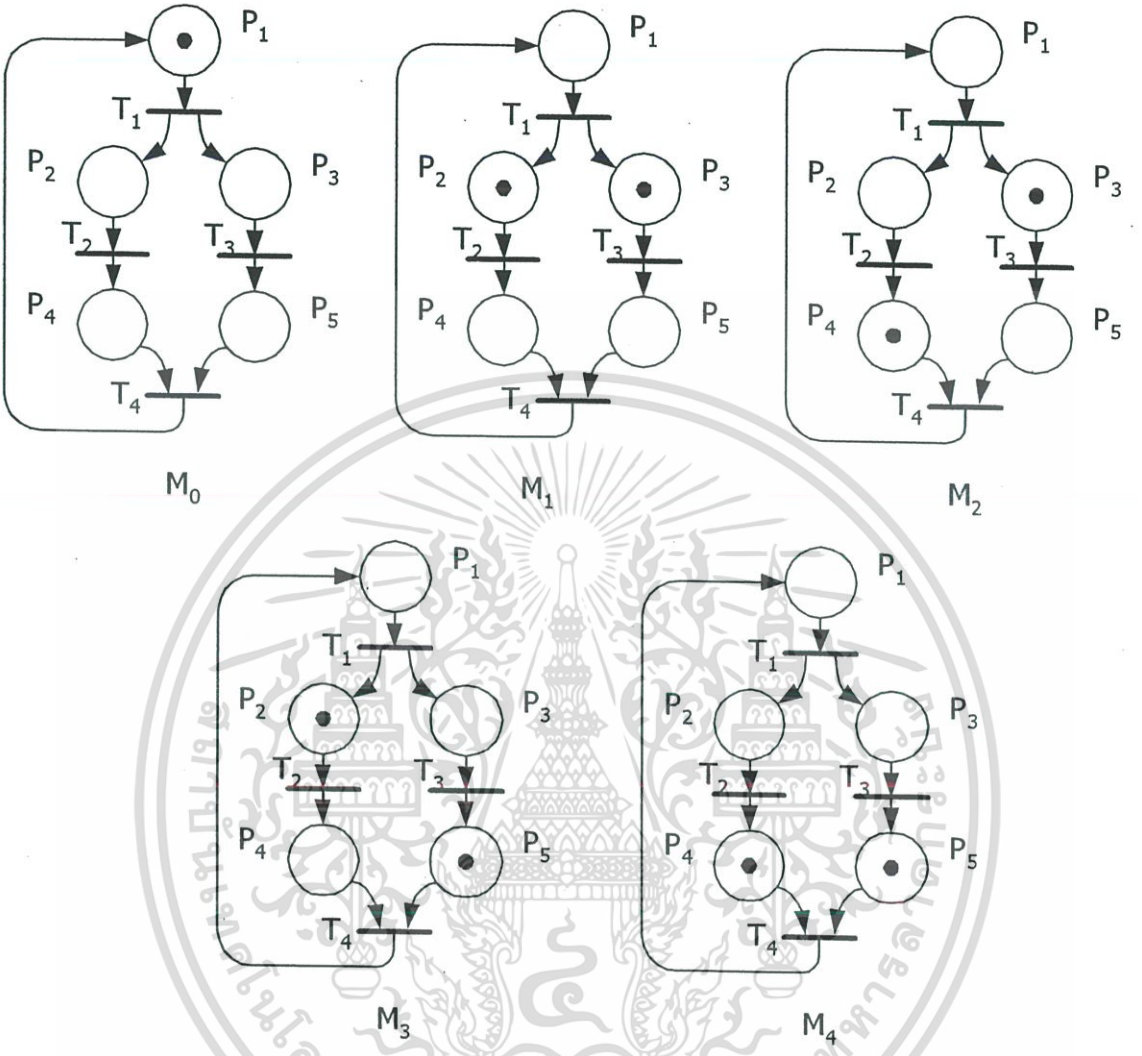
ในการที่เราใช้สัญลักษณ์และคำจำกัดความต่าง ๆ แทนนั้นจะทำให้ง่ายในการที่เราจะเขียนและอธิบาย Petri net

ในรูปที่ 2.5 เป็นการแสดงการ Mark ของ Petri netโดยกำหนดให้เขียนอยู่ในรูปของ Column Vector ซึ่งเราสามารถให้มันง่ายขึ้นโดยการ Transposed เราจะใช้สัญลักษณ์ [ ] แทนเมตริกซ์ และ ( ) แทน transposed

$$M_0 = (1,0,0,0,0) = [\,1\,0\,0\,0\,0\,]^T = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$



รูปที่ 2.5 แสดงสถานะเริ่มต้นของ Petri net



รูปที่ 2.6 แสดงการ Reachable marking

ที่ตำแหน่ง Mo จะสามารถ enable ที่ T_1 เพียงแห่งเดียวและถ้าเรา firing ก็จะได้ M_1 = (0,1,1,0,0) และจะใช้สัญลักษณ์เป็น

$$Mo\ [T_1 > M_1$$

ที่ตำแหน่ง M1 จะมี สอง Transition ที่ enable อยู่คือ T_2 และ T_3 ถ้าเรา firing แล้วก็จะได้ M_2 และ M_3 ซึ่งก็จะได้ (ในรูปที่ 2.6)

$$M_1\ [T_2 > M_2 = (0,0,1,1,0)$$

$$M_1\ [T_2 > M_3 = (0,0,0,1,1)$$

ที่ตำแหน่ง M_2 มี T_3 enable เพียง Transition เดียว และถ้า firing แล้วก็จะได้ $M_2\ [T_3 > M_4$ = (0,0,0,1,1)

สำหรับที่ M_3 T_2 enable ตำแหน่งเดียว ถ้า firing ก็จะได้ M_4 ซึ่งเมื่ออยู่ในตำแหน่ง M_4 แล้วนั้น T_4 ก็ enable เพียง Transition เดียวและเมื่อ firing T_4 แล้วนั้นก็จะได้ผลลัพท์เป็น Mo อีกครั้ง

ดังเป็นเหตุการณ์เช่นนี้แล้วสามารถเขียนได้ว่า *Mo = {Mo,M_1,M_2,M_3,M_4} และสามารถแสดงได้ดังรูปที่ 2.6

เมื่อเราเริ่มต้นที่ Mo ในขั้นแรก T_1 enable ต่อมา T_2 enable หลังจากการ firing ก็จะได้ M2 เราสามารถเขียนได้อีกแบบหนึ่ง คือ

$$Mo\ [\ T_1T_2 > M_2$$

T1T2 เป็นลำดับการ firing ซึ่งเขียนได้คือ

$$S = T_1T_2 \quad \text{และ} \quad Mo[S>M_2$$

2.1.6 คุณสมบัติที่สำคัญ (The essential characteristics)ของเพรททริเนทส์

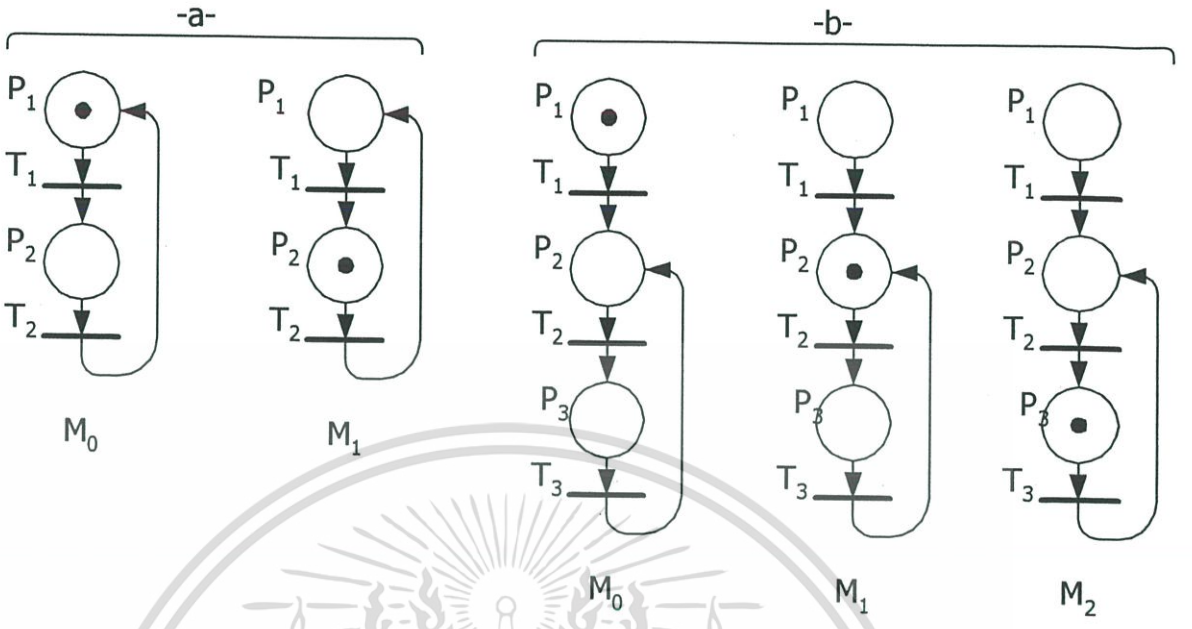
Petri netเป็นสิ่งที่นำไปใช้ในการวิเคราะห์การทำงานของระบบ โดยเราจะต้องเรียนรู้ถึงคุณสมบัติของมัน ซึ่งในที่นี้จะได้กล่าวถึงเฉพาะคุณสมบัติที่เกี่ยวข้องกับการทำโครงงานนี้เท่านั้น

การคงสภาวะทำงานและอยู่นิ่งของระบบ (Livens and deadlock)

การเกิด marking ที่ค่อย ๆ เกิดขึ้นโดยการ fireable ของ Transitions เมื่อใน Petri net นั้น ไม่สามารถ fireable ต่อไปได้อีกในทุก ๆ ทางของ Petri netแล้วจะทำให้ระบบที่ทำการออกแบบมีความน่าจะเป็นสูง และค่อนข้างจะแน่นอนที่จะเกิดปัญหา

Transition Tj จะคงสภาวะทำงาน (live) สำหรับสถานะเริ่มต้นของ Marking Mo ถ้า MiE*Mo จะต้องผ่าน Tj ทุกครั้งและเหมือนเดิมตลอด Transition live นี้สามารถอธิบายดังรูปที่ 2.7 จากรูป 2.16 a จะเห็นว่าในลำดับการเข้าหา Mo จะประกอบด้วยการ firing อยู่ 2 ลำดับคือ T1,T2, Mo[T_1>M1 และ M_1[T_2T_1>M_1 และจะเป็นอย่างนี้เรื่อย ๆ เมื่อมีการ firing เข้าหา Mo จะเรียก T1,T2 ว่า live ส่วนในรูปที่ 2.7b จะเห็นได้ว่าจะไม่สามารถเข้าหา Mo ได้ ซึ่งเมื่อทำการ firing T1 ไปแล้ว T_1 ไม่สามารถ enabled ได้อีก ดังนั้น T_1 จึงไม่ใช่ live Transition และ Petri netจะ Live ได้เมื่อทุก Transition ของ Petri netนั้น live

Transition เป็น quasi-live ถ้า Transition นั้นมีโอกาสที่จะ fired ดังในรูปที่ 2.7b ที่ T1 ซึ่งสามารถ fired หนึ่งครั้งก่อนที่จะไม่สามารถ enable ได้ตลอดไป ในรูปที่ 2.8a แสดงให้เห็นถึง quasi-live Petri net ในการ firing จากสถานะของการทำงานเริ่มต้น Mo. Mo[T_1>(0,1,0,0,) และ Mo[$T_2T_3T_4$>(0,0,1,0) ซึ่งเป็นการแย่งกัน firing ระหว่าง T_1,T_2 (<P_1{T_1,T_2} ถ้า Transition ที่ firing ก่อนเป็น T_1 ก็จะไม่สามารถ firing Transition อื่นได้อีก

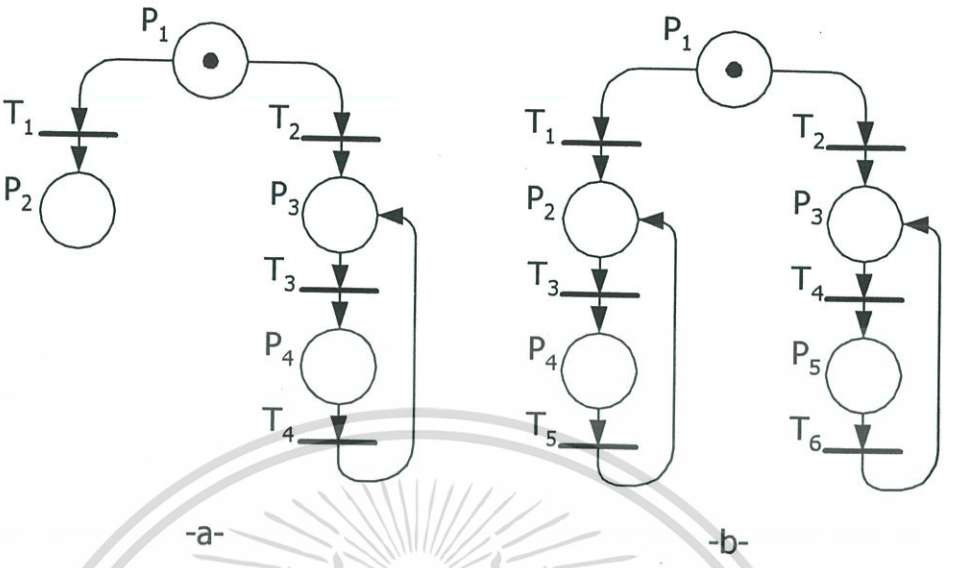


รูปที่ 2.7 แสดงการ Livens

Deadlock (หรือ sink state) เป็นปรากฏการณ์ที่ไม่สามารถที่จะ firing ต่อไปได้อีก ซึ่งเกือบจะถูกรูปแบบทุกระบบเมื่อเกิดสภาวะ deadlock และจะเป็นสิ่งที่แย่มากซึ่งก็จะกล่าวได้คือในระบบส่วนมากไม่ต้องการให้เกิด deadlock หรือกล่าวอีกนัยหนึ่ง deadlock คือ marking ที่ไม่สามารถจะ enabled ได้อีก จากในรูปที่ 2.8a เมื่อ firing T1 แล้วจะได้ผลลัพธ์ คือ M1 = (0,1,0,0) ซึ่ง Marking นี้คือ deadlock และPetri net จะสามารถพูดได้ว่าเป็น deadlock-free สำหรับสถานะเริ่มต้น Mo ถ้าเมื่อ firing ไปแล้วไม่สามารถเข้าหา Mo ได้โดยจะวนอยู่ที่เดิม

ในรูปที่ 2.8b เป็น deadlock free Petri net ถ้า firing T1 จะได้ Marking M1= (0,1,0,0,0) และ T3,T5 จะเป็น live Transition จากการ Marking นี้และเมื่อ firing T2 ก็จะได้ลักษณะเดียวกัน ซึ่งมันเรียกว่า deadlock free Petri nets

หมายเหตุ การเกิด livens และ deadlock นั้นจะขึ้นอยู่กับสถานะเริ่มต้นของการ Marking ด้วยเช่นถ้าสถานะเริ่มต้นของการ Marking ในรูปที่ 2.7a เป็นศูนย์ Mo = (0,0) นั้นจะเป็น deadlock และไม่มี Transition ใด enable เลย จากรูปที่ 2.8a จะเห็นได้ว่าเป็น quasi-live และจะเป็น deadlock เมื่อ Mo =(1,0,0,0) แต่ถ้า Mo = (0,0,1,0) จะไม่สามารถเรียกว่า live Petri net ได้ มันจะเป็น deadlock free แทน

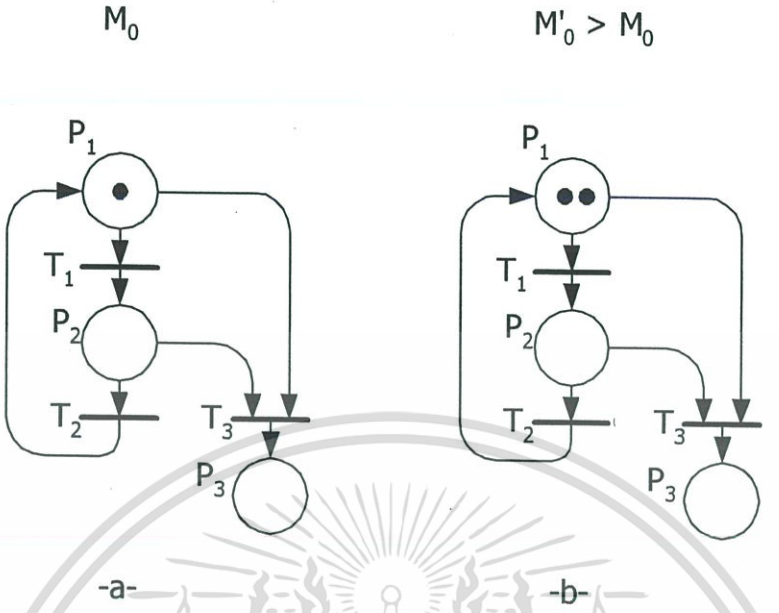


รูปที่ 2.8 Quasi live Petri nets. (a) With deadlock. (b) Deadlock-free

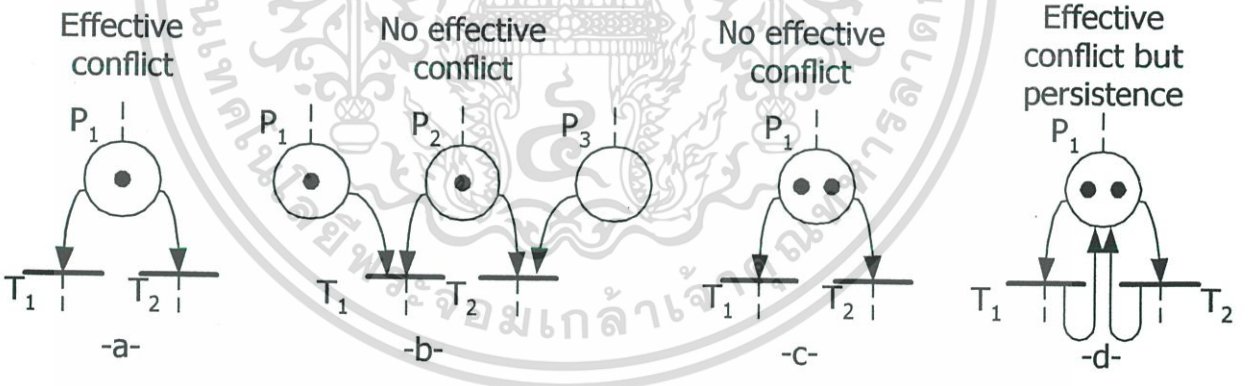
การต่อสู้ของเพททริเนทส์ (conflicts Petri nets)

Conflict ในโครงสร้างของระบบ conflict ก็คือเมื่อ Place หนึ่งเป็น input ของ Transition จำนวน 2 Transition หรือมากกว่านั้น ดังแสดงในรูปที่ 2.1 ที่ Place P3 เป็น input ของ Transition T3 และ T6 ซึ่งเมื่อ T3 และ T6 เกิดการ enable พร้อมกันนั้น จะทำให้ต้องตัดสินใจว่าจะ firing ในทางใดซึ่งสามารถเขียนได้เป็น $K = <P_1,\{T_1,T_2\ldots\}>$

ผลของการแข่งขันกัน firing ของ Transition จะเป็นการคงอยู่ของโครงสร้าง(K) และการ Marking(M) เช่น จำนวนของ Tokens ใน Pi น้อยกว่า จำนวน output Transition ของ Pi ซึ่งจะ enabled โดย M



รูปที่ 2.9 (a) T1 และ T2 เป็น Live T3 ไม่เป็น quasi-live.
(b) T1 และ T2 ไม่เป็น live , T3 เป็น quasi-live.



รูปที่ 2.10 Conflict

ในรูปที่ 2.10a เรามีโครงสร้างการแก่งแย่ง K_1=<P1,{T_1,T_2}> Transition T_1 และ T_2 enable แต่ใน place มี Token เพียงตัวเดียว ซึ่งจะทำให้เกิดผลของการแย่งกัน firing ระหว่าง T_1 กับ T_2 คือ Ke = <P_1,{T_1,T_2},M>

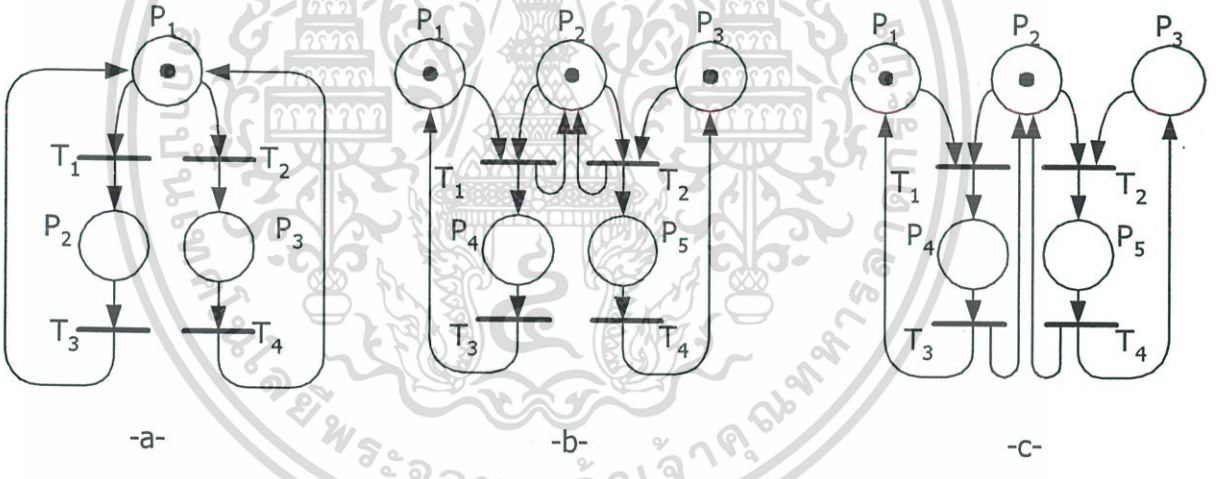
ในรูปที่ 2.10b จะมี T_1 enable เพียงตัวเดียว จึงไม่มีการแก่งแย่งกัน firing

ในรูปที่ 2.10c Transition T_1 และ T_2 enable ทั้งคู่ แต่ใน place มีจำนวน Token อยู่ 2 Token ดังนั้น Transition ทั้งสองจึงสามารถ firing พร้อมกันได้

ในรูปที่ 2.10d ถึงแม้ว่ามันจะเป็นการแย่งกันfiring ระหว่าง T_1 และ T_2 แต่จะมีโครงสร้างเฉพาะ ตามรูป จะไม่สามารถ firing พร้อมกันได้แต่เมื่อ firing ไปแล้ว จะมี Marking กลับมาดังเดิม จึงทำให้ firing ได้อีกซึ่งเหตุการณ์แบบนี้จะเรียกว่า persistent Petri net

ถ้าทุก Marking เป็นส่วนหนึ่งของการเข้าหา Mo(MiE*Mo) Petri nets นั้นจะปราศจากการแก่งแย่งเพื่อ firing Transition

หมายเหตุ Multiple firing คือการที่ Petri net สามารถ firing ไปพร้อมกันได้ และมีค่าเท่ากัน เช่น ในรูปที่ 2.6 ที่ M_1 จะเป็น Multiple firing ซึ่งจะลำดับการ firing เป็น $[T_2T_3]$ ซึ่งจะได้เป็น M_4 หรืออาจจะเขียนได้เป็น $M_1[[T_2T_3]> M4$



รูปที่ 2.11 Effective conflict and persistence for a Petri nets

2.1.7 กราฟของเครื่องหมาย และ กราฟแบบต้นไม้

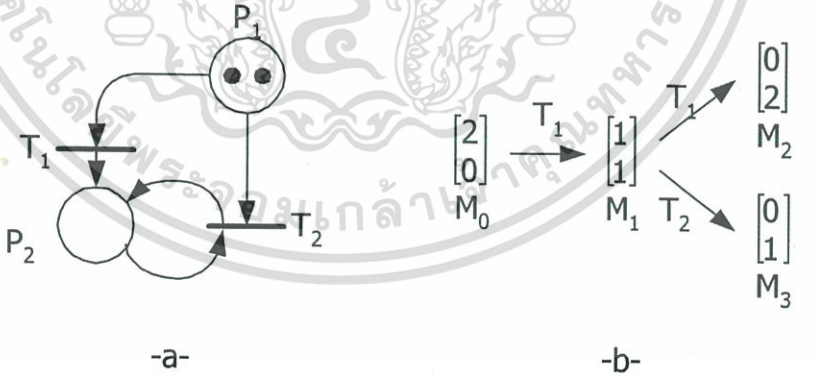
กราฟของเครื่องหมาย (Graph of markings)

Graph of markings สร้างขึ้นมาให้สอดคล้องกับการ firing ของ Transition ต่าง ๆ และเป็นเส้นทางการ firing ซึ่งจะเขียนขึ้นมาจากจำนวน Tokens ที่อยู่ใน Place ของแต่ละขั้นในการ firing

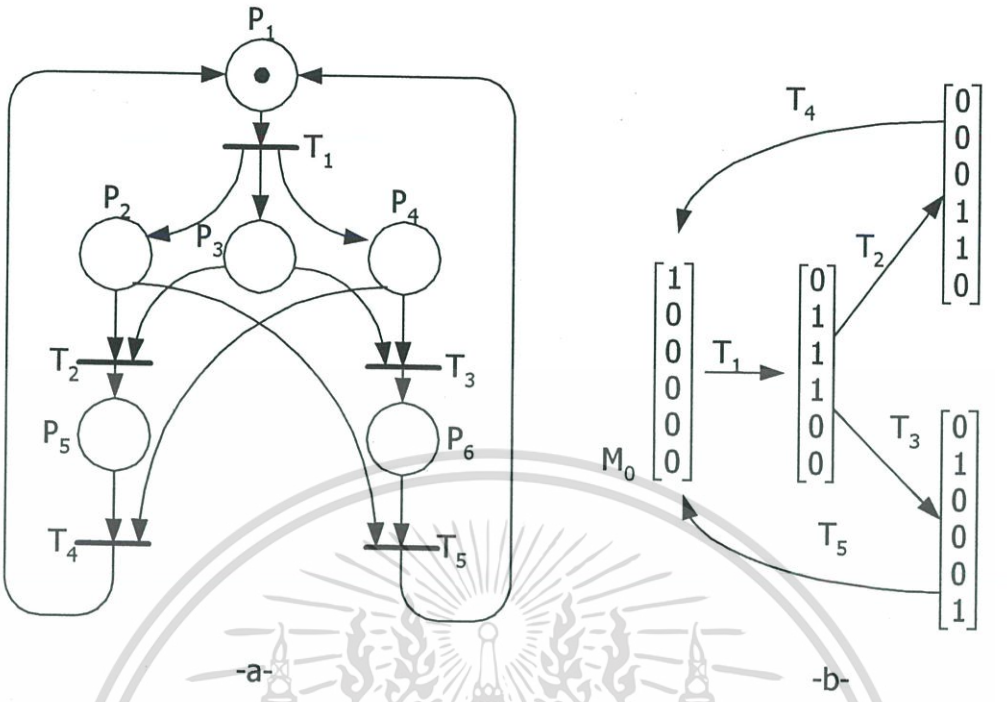
ในรูปที่ 2.12a คือ สถานะการทำงานของ Mo = (2,0) และในรูป 2.21b ก็คือ graph of marking ของรูปที่ 2.21a ซึ่งจะเห็นได้ว่าจาก Mo T_1 พร้อมที่จะทำงานถ้า firing ก็จะได้ M_1 เท่ากับ $M_1[T_1>M_2$, $M_1[T_2>M_3$ ซึ่งจะเห็นได้ว่าในรูปที่ 2.12a และ b มีความสัมพันธ์ในการ firing ของ Transition แต่ละ Transition ตรงกัน

Graph of markings สามารถนำไปใช้ประโยชน์ในการหาคุณสมบัติของ Petri net อย่างเช่นเราจะเห็นได้ว่า จากรูปที่ 2.12b เราจะรู้ว่ามันเกิด deadlock ขึ้นที่ M_2 และ M_3 ซึ่งมันมีขอบเขตไม่ใช่ live Petri nets

ในรูปที่ 2.13 เป็นการใช้ graph of marking หาคุณสมบัติของ Petri net จะสังเกตเห็นได้ว่า Petri net เป็น safe, live,reversible และ $T_1\ T_2\ T_4\ T_1\ T_3\ T_5$ เป็นลำดับการ firing ที่กลับคืนได้



รูปที่ 2.12 ตัวอย่าง กราฟของ Marking

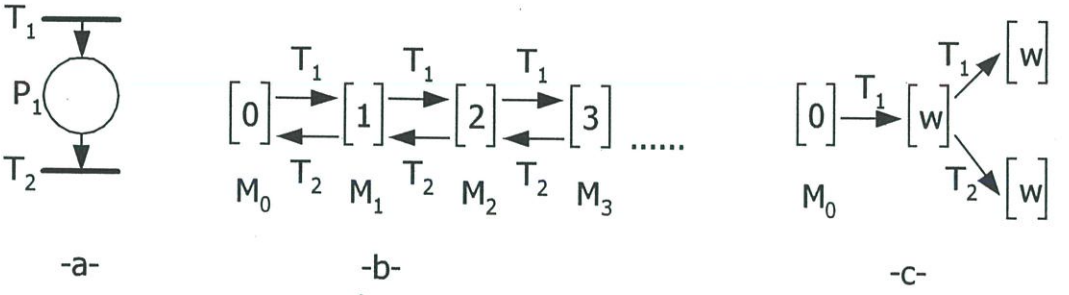


รูปที่ 2.13 ตัวอย่าง กราฟของ Marking

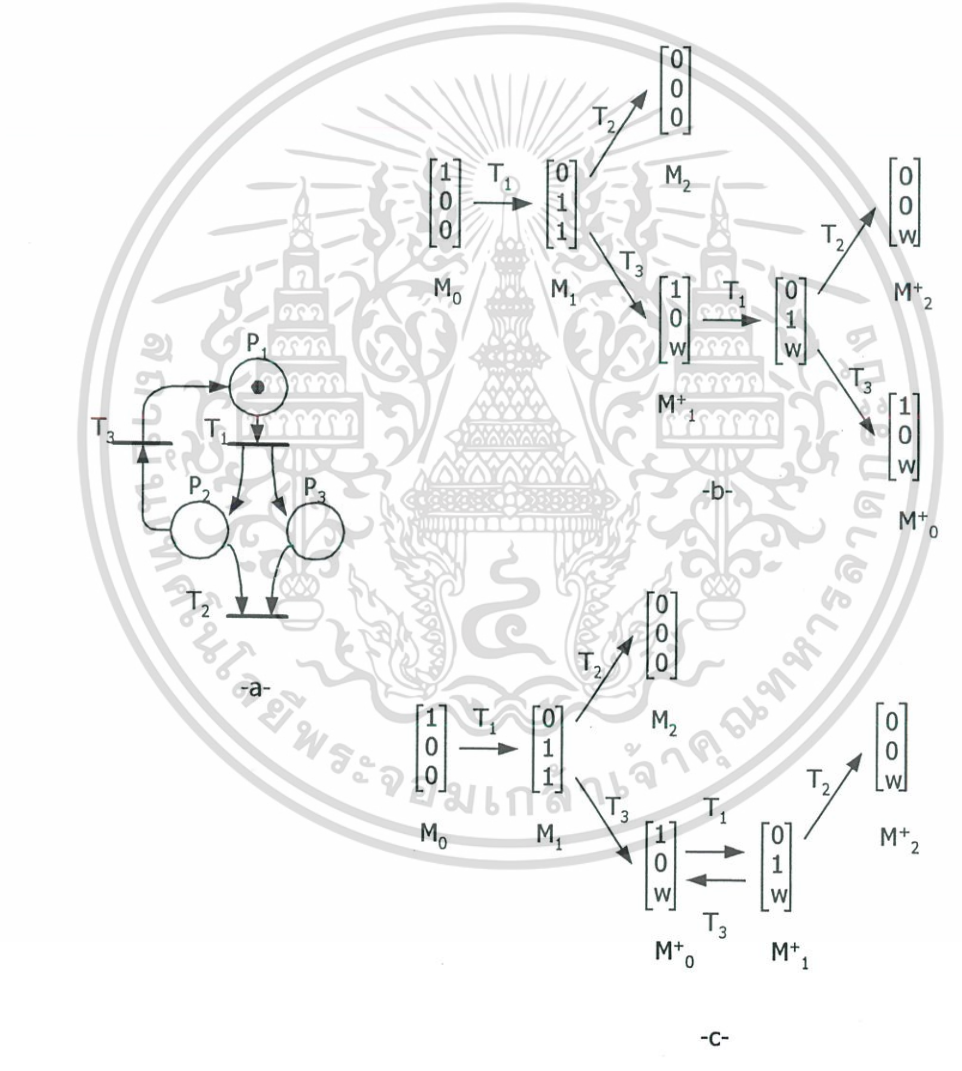
กราฟแบบต้นไม้ (Coverability graph and tree)

Coverability graph and tree เขียนขึ้นมาเพื่อรองรับการ firing ที่สร้าง Tokens ใน place ขึ้น ซึ่งมีจำนวนไม่จำกัด และมีลักษณะการ firing ซ้ำ ๆ กัน ดังเช่นในรูปที่ 2.14a ถ้า Transition T1 enabled และ fired ออกไปก็จะมีหนึ่ง Token ที่ P1 และ Transition T1 และ T2 ก็ยังคง enable อยู่ ถ้า T1 tired ก็จะมี Token ในP1 เป็นสอง Token แต่ถ้า T2 fired ก็จะทำให้ Tokens ลดลงและยังคงเป็นอย่างนี้ไปเรื่อย ๆ ในรูปที่ 2.14b เป็นการสร้าง graph of marking อย่างไรก็ตามแล้ว graph of marking นี่ก็ไม่จะสามารถที่จะแสดงความเป็นไปได้ของ marking ทั้งหมดได้เพราะการเข้าถึง Mo นั้นเป็นอนันต์ เพราะฉะนั้นเราจึงได้สร้างกราฟที่มีความสามารถที่จะแสดง marking ได้ครอบคลุมทั้งหมด

ในรูปที่ 2.14c ได้แสดง coverability tree(coverability graph and tree) ของรูป 2.23a โดยเริ่มที่ Mo = (O),Mo[T_1>M_1=(1) ซึ่ง M_1 covers Mo (M_1>Mo) และ T_1 สามารถกลับมา enable ได้อีก จำนวน Tokens ใน T1 สามารถเข้าถึง +k ซึ่งเราจะเรียกมันว่า(w) ซึ่งจะแสดงลักษณะของ marking ที่เป็นอนันต์ ซึ่งในรูป 1.26 จะมี Tokens ใน P_1 เป็น K+1 เมื่อ T_1 fired และ k-1 เมื่อ T_2 fired ซึ่งเราจะใช้ w นี้ เป็นตัวแทนสำหรับ place ที่มี Tokens มากขึ้นเป็นอนันต์ (แต่จะต้องไม่น้อยไปกว่าศูนย์)



รูปที่ 2.14 ตัวอย่างของ coverability tree



รูปที่ 2.15 coverability tree and coverability graph

Coverability tree จะมี Algorithm ในการสร้างดังนี้

ขั้นตอนที่ 1. จากสถานะเริ่มต้น Mo เขียน Mi (Marking ต่าง ๆ) ที่เกิดจากการ fired ของ Transition ที่ enable แต่ถ้ามีส่วนประกอบใดที่มากกว่าและสอดคล้องกับส่วนประกอบใน Mo ให้แทนด้วย w

ขั้นตอนที่ 2. สำหรับ M1(Marking ที่เกิดขึ้นใหม่) ซึ่งจะแยกได้ออกเป็นสองแบบคือ ขั้นตอน 2.1 หรือ ขั้นตอน 2.2

ขั้นตอน 2.1 ถ้า M ใด ๆ ที่เกิดขึ้นใหม่ มีค่าเท่ากับ M ที่ผ่านมา ก็แสดงว่าจะไม่มี Marking ต่อไปจะหยุดแค่นั้นและนำมาเชื่อมต่อเข้าด้วยกัน

ขั้นตอน 2.2 ถ้า M ที่เกิดขึ้นใหม่ไม่เท่ากับ M เดิม ก็จะสร้างตัวรับหรือจะพูดได้ว่าสร้าง Marking ใหม่ขึ้นมา และเขียนใหม่เช่นเดียวกันกับ step1 ถ้ามีส่วนประกอบใน M ใหม่ใด ที่มากกว่าและสอดคล้องกับส่วนประกอบใน M เดิมก็ให้แทนด้วย w

ตัวอย่าง (ในรูปที่ 2.15a และ b)

Step 1. $Mo[T_1>M_1$

Step 2.2 สำหรับ M_1

$M1[T_2>M_2$

$M1[T_3>(1,0,1)$ ซึ่ง $(1,0,1) > (1,0,0) = Mo$ เราจะเขียน

$M1[T_3>(1,0,w)\ M^+0$

Step 2.2 สำหรับ M_2 ไม่มี Transition ที่ enable ซึ่งมันคือ deadlock

Step 2.2 สำหรับ M^+0

$M^+1[T_2>(0,0,w) = M^+_2$

$M^+1[T_3>(1,0,w) = M^+o$

Step 2.2 สำหรับ M^+_2 ไม่มี Transition ที่ enable ซึ่งมันคือ deadlock

Step 2.1 สำหรับ M^+o เจอ M^+o ซึ่งได้รับการ fired จาก T_3 ซึ่งมีค่าเหมือนกันและเป็นการ fired ที่ซ้ำกัน $T_1\ T_3\ T_1\ T_3$ จาก Mo

2.2 ทฤษฎีของ Grafcet

ทฤษฎีของ Grafcet เป็นทฤษฎีที่ประยุกต์มาจากทฤษฎี Petri netซึ่งจะใช้ในการออกแบบควบคุมทางลอจิก ซึ่งมีค่าที่เป็นไปได้แค่เพียง “0” กับ “1” และจะดำเนินการด้านตรรกะ คือ การดำเนินการกับข้อมูลแล้วให้ค่าผลลัพธ์เป็น จริง หรือ ไม่จริง เท่านั้น โดยที่ “0” แทนเหตุการณ์ที่ไม่เป็นจริง หรือไม่ทำงาน ส่วน “1” จะแทนเหตุการณ์ที่เป็นจริงหรือทำงาน

การดำเนินการทางตรรกะ เช่น Not, And, Or, Xor

ตารางที่ 2.1 การดำเนินการทางตรรกะ Not

Input	Output
0	1
1	0

ตารางที่ 2.2 การดำเนินการทางตรรกะ And

Input	Input 2	Output
0	0	0
0	1	0
1	0	0
1	1	1

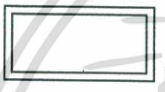
ตารางที่ 2.3 การดำเนินการทางตรรกะ Xor

Input 1	Input 2	Output
0	0	0
0	1	1
1	0	1
1	1	0

ตารางที่ 2.4 การดำเนินการทางตรรกะ Or

Input 1	Input 2	Output
0	0	0
0	1	1
1	0	1
1	1	1

2.2.1 สัญลักษณ์ที่ใช้



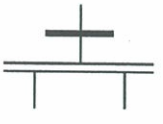
Place (Step) เริ่มต้น คือสถานะที่ใช้ในการเริ่มทำงานจะมี Logic = "1" เมื่อเริ่มทำงาน

Place (Step) คือ ขั้นการทำงานของโปรแกรม ซึ่งจะแสดงการทำงานของเครื่องจักรกลว่าตอนนี้จะให้มันทำงานอะไร

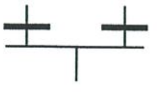
Transition มีลักษณะการทำงานเหมือนหน้าสัมผัส หรือสวิตช์ของวงจรไฟฟ้า

Arc คือ ใช้สำหรับต่อเชื่อมระหว่าง Place และ Transition

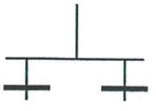
junction AND



Distribution AND



Junction OR



Distribution OR

2.2.2 เงื่อนไขของการทำงาน

1. Transition จะส่งผ่านสัญญาณออกไปเมื่อ Transition นั้น มีสถานะที่พร้อมที่จะทำงาน คือ มีสถานะเท่ากับ "1" และ Place ก่อนหน้าก็มีค่าเท่ากับ "1"

2. เมื่อ Transition หลาย ๆ ตัวมีสถานะพร้อมที่จะทำงานพร้อมกันมันก็จะทำงานไปพร้อมกัน

3. เมื่อ Transition ถัดไปจาก Step ไม่พร้อมที่จะทำงาน Step นั้นก็จะยังคงสภาพทำงานต่อไป

2.3 การติดต่อสื่อสาร

การติดต่อสื่อสารแบบ Host link เป็นการเชื่อมต่อระหว่าง PLC กับคอมพิวเตอร์ทั่วไปผ่านพอร์ตการสื่อสารของคอมพิวเตอร์ที่เรียกว่า COM 1: หรือ COM 2: ซึ่งเป็นพอร์ตการสื่อสารแบบอนุกรม เพื่อให้สามารถควบคุม PLC จากคอมพิวเตอร์ได้ สำหรับคอมพิวเตอร์ 1 เครื่อง สามารถต่อเข้ากับ PLC ได้จำนวนมาก โดยใช้การเชื่อมต่อ PLC หลาย ๆ ตัวเข้าด้วยกัน ซึ่งเรียกว่า PC Link ในการติดต่อแบบ Host link จะต้องผ่านอุปกรณ์ที่เรียกว่า Host link Unit

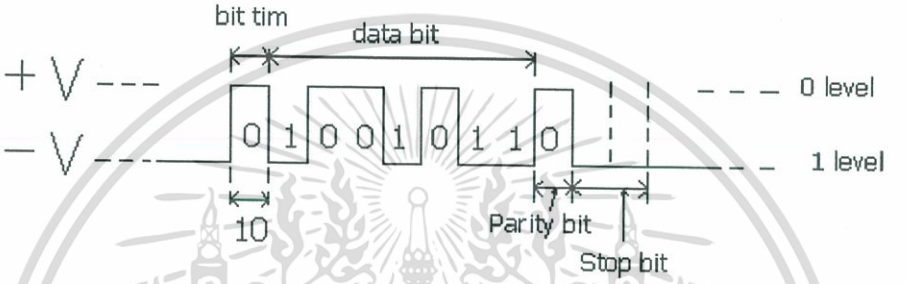
2.3.1 พอร์ตการสื่อสารแบบอนุกรม

ช่องการสื่อสารอนุกรมนี้สามารถเรียกได้หลายอย่าง เช่น Serial Port, RS-232, comport เครื่องคอมพิวเตอร์โดยปกติจะมีพอร์ตชนิดนี้อยู่แล้ว 2 พอร์ต คือ พอร์ตขนาด 9 ขา มีรูปร่างเหมือนสี่เหลี่ยมคางหมูมีเข็มยื่นออกมา 9 เข็ม เรียกหัวชนิดนี้ว่า "DB – 9 Connector Male Type" และอีกพอร์ตหนึ่ง คือพอร์ตขนาด 25 ขา มีลักษณะเหมือนกับแบบ 9 ขา ต่างกันตรงมีขามากกว่าเท่านั้น และเรียกว่า "DB – 25"

รูปที่ 2.16 ตำแหน่งของ DB – 9

ลักษณะที่สำคัญของพอร์ตอนุกรม คือ ข้อมูลจะส่งในลักษณะอนุกรม กล่าวคือ แทนที่จะต้องมีสายสัญญาณ 8 เส้น สำหรับการส่ง 1 ไบต์ ระบบนี้อาจใช้เพียง 2 เส้น หรือมากกว่า (อย่างน้อยต้องมีสายสัญญาณ 1 เส้น และ Ground 1 เส้น) โดยจะให้สัญญาณผ่านทีละบทเรียงต่อกันไปจนครบไบต์ ซึ่งลักษณะเช่นนี้มีข้อดี คือ จำนวนสายสัญญาณจะลดลงไปมาก แม้ความเร็วในการส่งจะน้อยลงก็ตาม โดยสัญญาณที่ส่งไปนี้จะส่งไปในลักษณะเป็น Asynchronous Serial Data Format กล่าวคือ ข้อมูลที่ส่งจะประกอบด้วยสัญญาณ 2 ระดับ มาเรียงต่อกัน (เช่น + 5 V สำหรับระดับ "0"

และ – 5 V สำหรับระดับ "1") โดยก่อนจะเริ่มส่งข้อมูลระดับสัญญาณจะอยู่ที่ระดับ "1" จากนั้นก็จะมีสัญญาณเริ่มส่ง (Start Bit) ซึ่งเป็นสัญญาณระดับต่ำ "0" ส่งไปเพื่อบอกว่ากำลังจะส่งข้อมูลตามมา จาก Start Bit จะตามด้วยสัญญาณข้อมูลขนาด 8 บิท เรียงต่อกันไป จากนั้นก็จะมีสัญญาณบอกว่าจบการส่งข้อมูล (เรียกว่า Stop Bit) ซึ่งเป็นสัญญาณระดับสูง ("1") เมื่อหมดชุดข้อมูลก็ปรับให้สัญญาณมาอยู่ที่ระดับ "1" เพื่อคอยรับ Start Bit ต่อไป



รูปที่ **2.17** ลักษณะสัญญาณขณะส่งผ่านพอร์ทอนุกรม

การส่งสัญญาณลักษณะที่ค่อนข้างช้า เพราะจำนวนบิทเรียงต่อกันอย่างอนุกรม โดยอัตราความเร็วของการส่ง เรียกว่า Boad Rate: จำนวนบิท/วินาที จะมีค่าอยู่ในช่วง 1200, 2400, 9600 จะเห็นว่าการส่งสัญญาณลักษณะนี้ 1 ไบต์ จะต้องใช้จำนวนบิทถึง 10 บิท (Start Bit + Data Bit + Stop Bit) เรียงต่อกันแบบอนุกรมกันไป จึงเป็นเหตุให้การส่งข้อมูลช้า

ในการส่งข้อมูลผ่านพอร์ตอนุกรมจะต้องคำนึงถึงสิ่งต่าง ๆ เช่น

- ต้องมีระดับสัญญาณตั้งแต่ +3 ถึง +25 โวลท์ (สำหรับการส่งสัญญาณโดยคอมพิวเตอร์ไม่มีปัญหา แต่ในกรณีรับสัญญาณเข้าอาจต้องมีระบบป้องกัน เพราะหากสัญญาณเกิน +25 โวลท์ อาจต่อให้เกิดความเสียหายได้
- ความเร็วในการส่ง (Baud Rate) ถ้าเป็นสายสัญญาณธรรมดาอาจส่งได้เร็วถึง 9600 บิท/วินาที แก่กรณีใช้ผ่านระบบโทรศัพท์ความเร็วจะลดลง
- จำนวนบิทสำหรับข้อมูลว่าใช้ระบบใด เช่น 5, 6, 7 หรือ 8 บิท/ตัวอักษร
- จำนวน Stop Bit มีขนาดเท่าใด เช่น 1 Stop Bit หรือ 2 Stop Bit
- จะต้องมี Parity Check หรือไม่ ฯลฯ

ตารางที่ 2.5 ตารางหน้าที่ของแต่ละขาของพอร์ตอนุกรม DB-9

ขา	หน้าที่
1	Data Carrier Detect
2	Receive Data
3	Transmit Data
4	Data Terminal Ready
5	Signal Ground
6	Data Set Ready
7	Request to Send
8	Clear to Send
9	Ring Indicator

ตารางที่ 2.6 ตารางหน้าที่ของแต่ละขาของพอร์ตอนุกรม แบบ DB-25

ขา	หน้าที่
1	ไม่ได้ใช้
2	Transmit Data
3	Receive Data
4	Request to Send
5	Clear to Send
6	Data Set Ready
7	Signal Ground
8	Data Carrier Detect
20	Data Terminal Ready
22	Ring Indicator
9-19, 21< 23-25	ไม่ได้ใช้

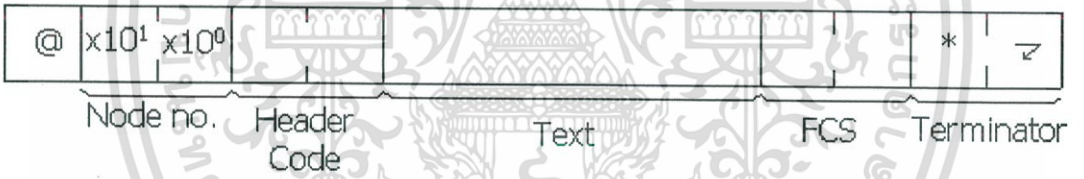
- ขา Data Carrier Detect : DCD หรืออาจเรียกว่า Carrier Detect : CD ขาที่จะแอกทีฟเมื่อมีการส่งสัญญาณพาหะจากอุปกรณ์สื่อสารข้อมูล เช่น โมเด็ม สำหรับการใช้งานปกติขานี้จะไม่ถูกใช้งานมากนัก
- ขา Receiver Data : หรือ RXD ขานี้เพื่อรับสัญญาณอนุกรมเข้ามายังคอมพิวเตอร์โดยนำข้อมูลที่อ่านได้ไปเก็บไว้ในรีจีสเตอร์บัฟเฟอร์
- ขา Transmitter Data หรือ TXD ขานี้ใช้เพื่อส่งข้อมูลอนุกรมออกมาจากคอมพิวเตอร์โดยในการนำข้อมูลที่เก็บอยู่ในบัฟเฟอร์ส่งข้อมูลออกไป
- ขา Data Terminal Ready : DTR เป็นขาเอาท์ฟุตที่ใช้ในการส่งสัญญาณข้อมูลออกจากคอมพิวเตอร์เพื่อใช้อุปกรณ์ปลายทางรับรู้ว่าต้องการติดต่อกับอุปกรณ์ปลายทางโดยที่ขา DTR นี้จะต้องเชื่อมต่อกับขา DSR ของอุปกรณ์ปลายทางและขา DTR ของอุปกรณ์ปลายทางจะต้องเชื่อมต่อกับขา DSR ของพอร์ตอนุกรมเข้าด้วยกัน และจะต้องเชื่อมต่อเข้ากับขา DCD ด้วยในกรณีที่โปรแกรมสื่อสารที่ใช้ในการตรวจสอบสัญญาณพาหะ
- ขา Signal Ground : GND เป็นขากราวด์ของสัญญาณ
- ขา Data Set Ready : DSR ขานี้ใช้คู่กับขา DTR เพื่อตรวจสอบการเชื่อมต่อระหว่างคอมพิวเตอร์กับอุปกรณ์ปลายทาง ซึ่งขา DSR นี้จะเป็นขาที่รับรู้ข้อมูลจากภายนอก
- Request to send : RTS เป็นขาเอาท์พุทสำหรับส่งสัญญาณร้องขอให้อุปกรณ์ปลายทางส่งข้อมูลมายังคอมพิวเตอร์ โดยขาที่รับสัญญาณขา RTS คือขา CTS ซึ่งในกรณีที่มีการเชื่อมต่อแบบ 3 สาย จะต้องเชื่อมต่อขา RTS และ CTS เข้าด้วยกันเพื่อให้การรับและส่งข้อมูลเกิดขึ้นได้ตลอดเวลา
- ขา Clear To Send : CTS เป็นขาอินพุททำหน้าที่รอรับสัญญาณที่ส่งเข้ามาเมื่อมีการส่งสัญญาณเข้ามา ขานี้ใช้ตรวจสอบอุปกรณ์ต่อพ่วงว่าพร้อมที่จะรับข้อมูลแล้วหรือยัง
- ขา Ring Tnd : cator : RT ใช้แสดงสถานะสัญญาณเรียกจากสายโทรศัพท์ปกติในการสื่อสารทั่วไป สายนี้จะไม่ถูกใช้งาน จะใช้งานก็ต่อเมื่อมีการเชื่อมต่อกับโมเด็มแล้วยังมีความต้องการตรวจสอบสัญญาณเรียกจากสายโทรศัพท์

2.3.2 การส่งสัญญาณเพื่อควบคุมการทำงานของ PLC

ในการส่งสัญญาณเพื่อควบคุม PLC จะส่งสัญญาณผ่านพอร์ตอนุกรมหรือ RS-232 ซึ่งจะอาศัยรูปแบบข้อตกลงในการสื่อสาร (Protocol) ของ PLC โดยทั่วไปจะเป็นการตอบโต้กันระหว่าง เครื่องควบคุมกับอุปกรณ์ภายนอกอุปกรณ์ภายนอกในที่นี้คือ คอมพิวเตอร์ ซึ่งคอมพิวเตอร์จะเป็นตัวส่งคำสั่งไปยัง PLC ในลักษณะตามรูปแบบของ Protocol จากนั้น PLC จะส่งสัญญาณตอบสนองกลับมายังคอมพิวเตอร์ในลักษณะรูปแบบของ Protocol เช่นกัน

ในการติดต่อสื่อสารของ PLC ยี่ห้อ OMRON นี้ รูปแบบข้อตกลงในการสื่อสาร (Protocol)เรียกว่า Host link Protocol ซึ่งมีรูปแบบการสื่อสารดังนี้

รูปแบบของคำสั่ง (Command Format)



@ เป็นเครื่องหมายที่ต้องมีเสมอสำหรับเริ่มต้นคำสั่ง

Node no

จะใช้สำหรับกำหนดว่าจะติดต่อกับ PLC หมายเลขอะไร (Nod) ในกรณีที่ PLC ที่เชื่อมต่ออยู่มีมากกว่า 1 เครื่องขึ้นไป ซึ่งหมายเลขของ PLC นี้จะสามารถตั้งได้ที่ตัว PLC เอง

Header Code

เป็นรหัสของคำสั่งว่าเราต้องการที่จะทำอะไรกับ PLC และทำในพื้นที่ใด เช่น ต้องการที่จะอ่านข้อมูลจากพื้นที่รีเลย์ภายในของ PLC ก็จะใช้รหัสคำสั่ง เป็นตัวอักษรพิมพ์ใหญ่ว่า “RR” หรือถ้าต้องการเขียนข้อมูลลงไปในพื้นที่รีเลย์ภายในของ PLC ก็จะใช้รหัสคำสั่งเป็นตัวอักษรพิมพ์ใหญ่ว่า “WR” เป็นต้น ซึ่งจะสามารถศึกษาได้จากคู่มือของ PLC

Text

เป็นส่วนของข้อมูลหรือกลุ่มคำสั่งที่ต้องการสั่ง เช่น ถ้าเป็นคำสั่งในการอ่านก็จะระบุตำแหน่งเริ่มต้นในการอ่านและจำนวนที่ต้องการอ่าน หรือถ้าเป็นคำสั่งเขียนก็จะระบุตำแหน่งเริ่มต้นในการเขียนข้อมูลและข้อมูลที่ต้องการเขียนลงไป เป็นต้น

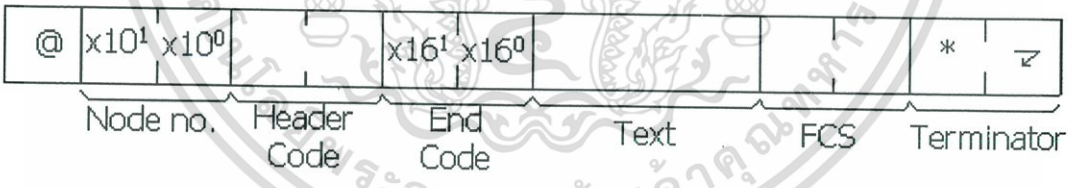
FCS (Frame Check Sequence Code)

เป็นรูปแบบของการตรวจสอบความผิดพลาดในการสื่อสารมี 2 byte โดยที่ FCS นี้จะได้จากการนำค่าเลขฐานสิบหกของรหัส ASCII แต่ละตัวแปลงเป็นรหัส BCD แล้วทำการทางตรรกะโดยการ Exclusive-Or (XOR) โดยเริ่มจากอักษร ๑ จนถึงตัวอักษรสุดท้ายของ Text หลังจากนั้นทำการแปลงกลับให้เป็นเลขฐานสิบหกอีกครั้งหนึ่งก็จะได้รหัสในการตรวจสอบข้อผิดพลาด ของ Host link Protocol ที่เรียกว่า FCS

Terminal

จะประกอบด้วย 2 ตัวอักษร คือ * ใช้สำหรับปิดท้ายคำสั่งเพื่อบอกว่าจบคำสั่งนี้แล้วแล้วก็จะตามด้วยรหัสของ Carriage return (CHR $ (13))

รูปแบบของการตอบสนอง (Response Format)



ในรูปแบบของการตอบสนองนี้จะมีส่วนที่แตกต่างจากรูปแบบคำสั่ง คือ

End Code

เป็นรหัสสถานะของการรับ-การส่งข้อมูล เช่น รหัส 00 เป็นรหัสที่บอกให้ทราบว่าการส่งข้อมูลเรียบร้อยดี หรือรหัส 14 คือรหัสบอกให้ทราบว่ารูปแบบของคำสั่งผิด เป็นต้น

Text

ในรูปแบบของการตอบสนองในส่วนนี้จะมีเฉพาะคำสั่งที่ใช้สำหรับอ่านเท่านั้น ซึ่งจะเป็นส่วนของข้อมูลที่อ่านได้

บทที่ 3
การใช้ Petri Nets ช่วยในการออกแบบวงจรแลดเดอร์ และการออกแบบโปรแกรม

เพื่อให้การตรวจสอบข้อบกพร่อง แก้ไขข้อผิดพลาด และเปลี่ยนแปลงเงื่อนไขการทำงานของวงจรแลดเดอร์ที่ใช้สำหรับควบคุมเครื่องจักรกลต่าง ๆ ง่ายขึ้นนั้น ควรที่จะมีรูปแบบในการเขียนวงจรแลดเดอร์ที่เป็นลำดับขั้นตอนไม่ซับซ้อน ไม่ใช่ว่านึกจะใส่อะไรก็ใส่เข้าไป จากที่ได้ศึกษาเกี่ยวกับทฤษฎีของ Petri Nets. และ Gratect ในบทที่ 2 ที่ผ่านมา ซึ่งจะได้นำความรู้เรื่องนี้มาประยุกต์ใช้ ช่วยในการออกแบบวงจรแลดเดอร์ แต่ในที่นี้ก็ไม่ใช่กฎเกณฑ์ตายตัวที่จะต้องเขียนวงจรแลดเดอร์ด้วยวิธีนี้เท่านั้น เพียงแต่วิธีนี้จะเป็นแนวทางในการออกแบบวงจรแลดเดอร์ที่ทำให้ผู้ออกแบบไม่สับสน และผู้ที่กลับมาแก้ไขก็จะสามารถศึกษาวงจรเก่าที่มีอยู่ได้ง่ายขึ้น และเพื่อให้เห็นผลการทำงานของวงจรที่ใช้ควบคุมเครื่องจักรกลได้ชัดเจนยิ่งขึ้น จึงได้เขียนโปรแกรมเพื่อใช้ในการตรวจสอบโปรแกรมที่ผู้ใช้งานเขียนขึ้นมาเพื่อใช้ในการควบคุมเครื่องจักร และจำลองสภาวะการทำงานต่าง ๆ ก่อนที่จะนำไปใช้งานจริง ซึ่งจะได้เขียนโปรแกรมอยู่บนพื้นฐานของทฤษฎีของ Petri Net. และ Grafcet และควบคุมอุปกรณ์ภายนอกผ่านหน่วยอินพุตและหน่วยเอาท์พุตของ PLC

3.1 การใช้ Petri Nets. ช่วยในการออกแบบวงจรแลดเดอร์

ในการใช้ petri Nets. ช่วยในการออกแบบวงจร แลดเดอร์นั้น พอที่จะสรุปเป็นขั้นตอนต่าง ๆ ได้ดังนี้ คือ

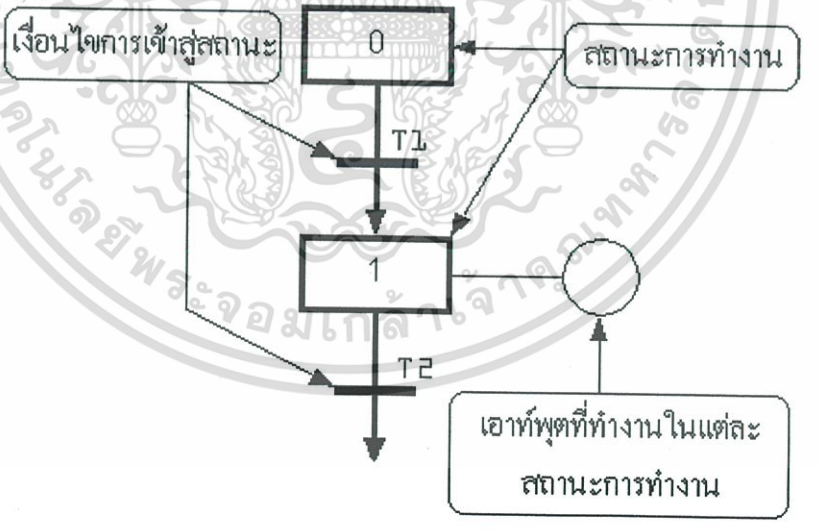
1. วิเคราะห์และทำความเข้าใจกับเงื่อนไขการทำงานของเครื่องจักรกลให้เข้าใจหลักการทำงาน (สมควรเขียนเป็นแผนผังเวลาออกมาเพื่อง่ายในการพิจารณาในขั้นตอนต่อไป)
2. พิจารณาสภาวะการทำงานของเอาท์พุตและอุปกรณ์ช่วยต่าง ๆ ที่ทำงานอยู่ในช่วงเวลาเดียวกัน ให้จัดอยู่ในสถานะการทำงานเดียวกัน
3. พิจารณาว่าเงื่อนไขอะไรที่ทำให้สถานะการทำงานเปลี่ยนแปลงไปจากสถานะของการทำงานใด และไปยังสถานะของการทำงานใด
4. เขียนวงจรการควบคุมออกมาโดยเริ่มจากสถานะการทำงานเริ่มต้นและเขียนตามลำดับการทำงานของสถานะต่าง ๆ โดยที่ระหว่างสถานะของการทำงานในแต่ละสถานะจะถูกขัดขวางโดยเงื่อนไขการเปลี่ยนแปลงสถานะ (ในขั้นตอนที่ 3)

5. ระบุอุปกรณ์ต่าง ๆ ที่จะทำงานในแต่ละสถานะการทำงานนั้นลงไป
6. แปลงจากรูปแบบของ Petri Nets. ให้เป็นวงจรแลดเดอร์

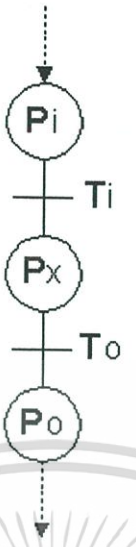
การแปลงจารรูปแบบของ Petri Nets. เป็นวงจรแลดเดอร์

เพื่อให้วงจรควบคุมที่เขียนขึ้นมานั้นสามารถใช้ได้กับ PLC ทั่วไป เราจึงจำเป็นที่จะต้องแปลงวงจรควบคุมที่เขียนขึ้นมาให้อยู่ในรูปแบบของวงจรแลดเดอร์เพื่อให้ PLC รับรู้ได้ โดยที่ในรูปแบบเพททริเนทส์จะเห็นได้ว่ามีส่วนที่สำคัญอยู่ 3 ส่วน คือ

1. เงื่อนไขการเข้าสู่สถานะการทำงาน ซึ่งเงื่อนไขของสถานะการทำงานนี้จะเป็นตัวกำหนดว่าจะให้วงจรนั้นทำงานอยู่ที่สภาวะการทำงานใด
2. สถานะการทำงาน เป็นลำดับการทำงานของแต่ละลำดับขั้นตอนของระบบการทำงาน โดยการทำงานนั้นจะขึ้นอยู่กับเงื่อนไขการเข้าสู่สถานการทำงานเพื่อเข้าสู่การทำงานในแต่ละลำดับขั้นตอน
3. อุปกรณ์ที่จะทำงานในแต่ละสถานะการทำงาน ในส่วนนี้จะเป็นตัวบอกว่าในแต่ละสถานะการทำงานนั้นจะมีเอาท์พุตตัวใดบ้างที่ทำงาน



รูปที่ 3.1 ส่วนประกอบที่สำคัญของวงจรควบคุมในรูปแบบของเพททริเนทส์



รูปที่ 3.2 ส่วนประกอบที่สำคัญของวงจรควบคุมในรูปแบบของเพททริเนทส์

เมื่อ

Px = Place (สถานะการทำงาน) ใดๆ

Pi = Place input ของ Place (สถานะการทำงาน) ใดๆ

Po = Place output ของ Place (สถานะการทำงาน) ใดๆ

Ti = Transition input ของ Place (สถานะการทำงาน) ใดๆ

To = Transition output ของ Place (สถานะการทำงาน) ใดๆ

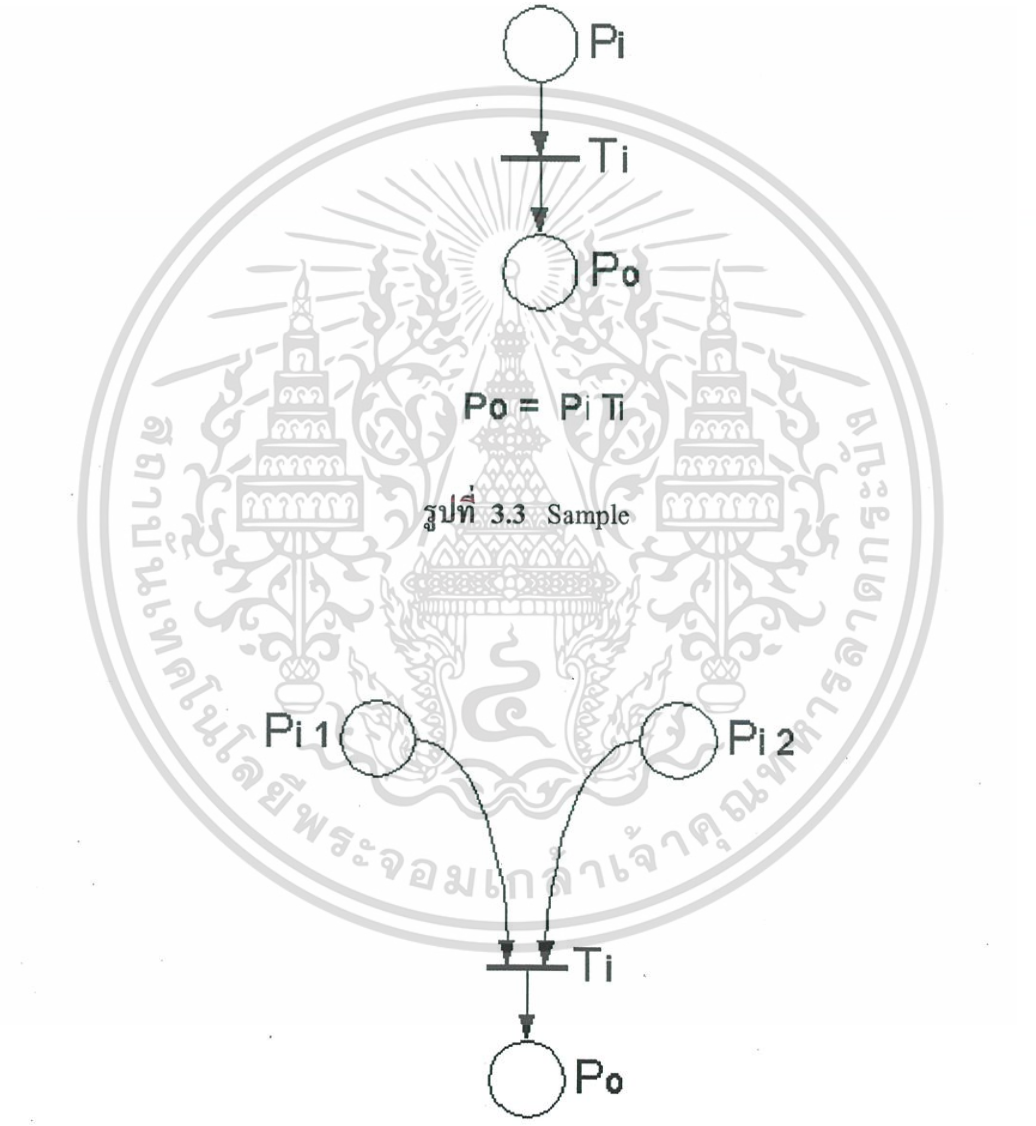
จากเงื่อนไขการส่งผ่าน Token ของ Transition เมื่อนำมาประยุกต์เพื่อใช้ในการออกแบบวงจรแลดเดอร์ พอที่จะสรุปได้ว่า ที่ Place ใดๆก็ตาม จะทำงานได้ก็ต่อเมื่อ Place input และ Transition input ของ Place นั้น มีสภาวะลอจิกเป็น "1" (Place input, Transition input พร้อมที่จะทำงาน และ Transition input ทำการส่งผ่าน Token จาก Place input ไปยัง Place ใดๆ) และที่ Place ใดๆ นี้จะหยุดทำงานเมื่อ Place output และ Transition output ของ Placeนั้น มีสภาวะลอจิกเป็น "1" (Transition output พร้อมที่จะทำงาน และทำการส่งผ่าน Token จาก Place ใดๆ ไปยัง Place output และหลังจากส่งผ่าน Token ไปแล้วจะไม่มีผลกับ Place ใด) ซึ่งสามารถที่จะเขียนเป็นสมการได้ดังนี้

$$Px = (Pi\,Ti + Px)(\overline{Po} + \overline{To})$$

ยกเว้นที่ Place ใดๆ นั้นเป็นสถานะการทำงานเริ่มต้น (P0) ซึ่งจะมีผลของเงื่อนไขที่ว่า Token จะเป็น 1 เมื่อเริ่มทำงานดังนั้นจึงต้องเพิ่มส่วนนี้เข้ามาซึ่งจะได้สมการดังนี้

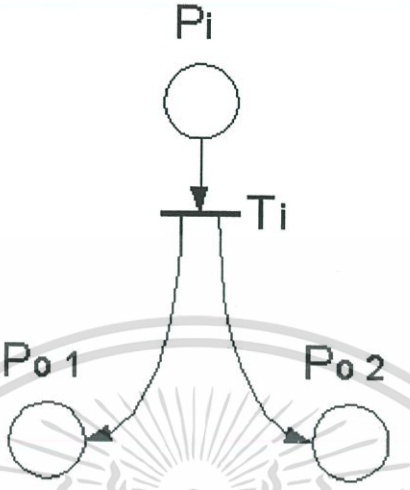
$$P0 = \left(Pi\,Ti + \overline{P}j + P0\right)\left(\overline{P}o + \overline{T}o\right)$$

เมื่อ J = 1 ถึง n (n = จำนวนสถานะการทำงาน)



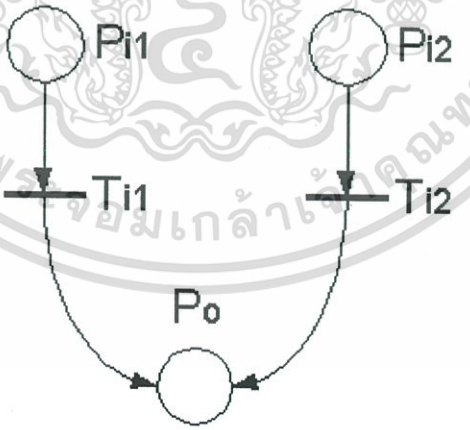
รูปที่ 3.3 Sample

รูปที่ 3.4 JunctionAND



Po1 = Pi Ti
Po2 = P i Ti

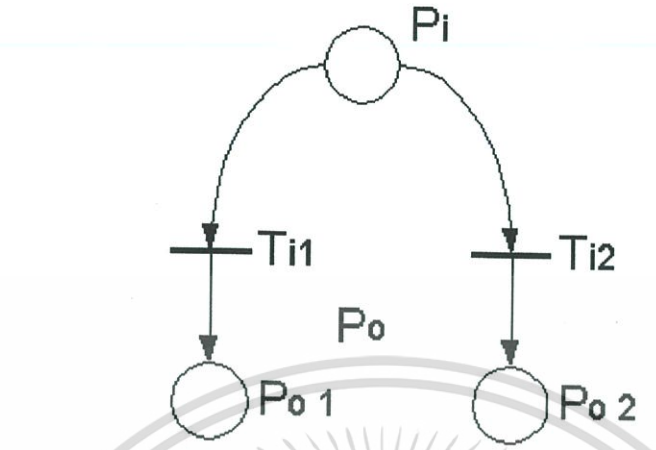
รูปที่ 3.5 DistributionAND



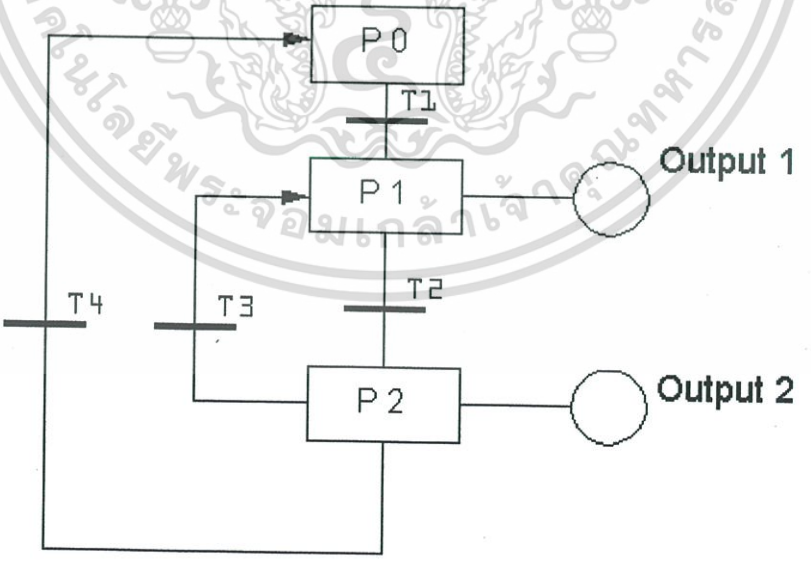
Po = Pi1 Ti1 + P i2 Ti2

รูปที่ 3.6 JunctionOR

ตัวอย่าง



รูปที่ 3.7 DistributionOR



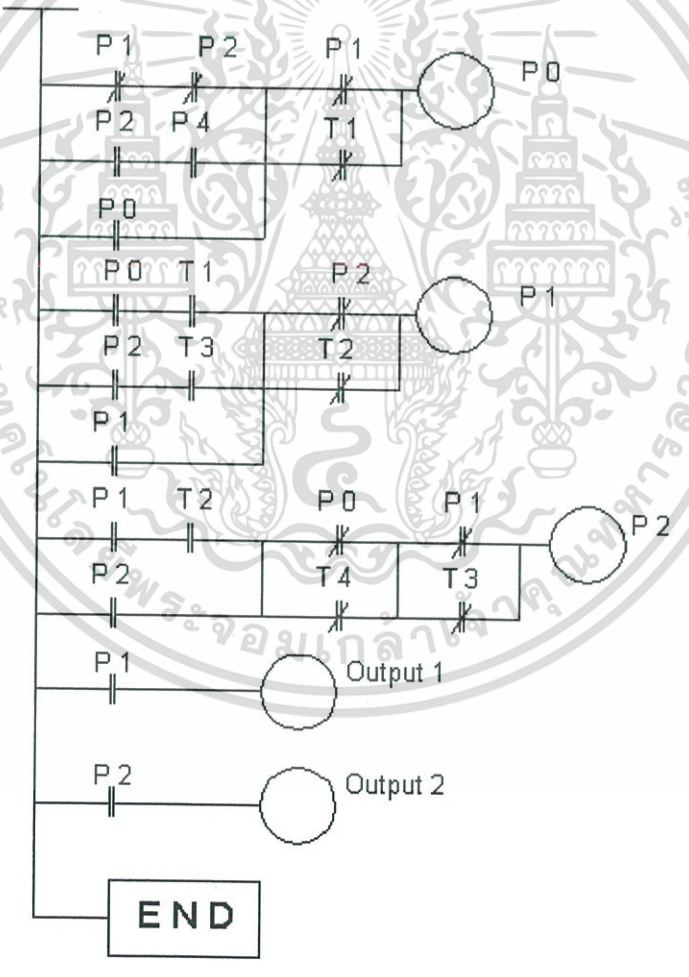
รูปที่ 3.8 ตัวอย่างของรูปแบบ Grafcet

$$P0 = (P2T4 + \overline{P1}\,\overline{P2} + P0)(\overline{P1} + \overline{T1})$$

$$P1 = (P0T1 + P2T3 + P1)(\overline{P2} + \overline{T2})$$

$$P2 = (P1T2 + P2)(\overline{P0} + \overline{T4})(\overline{P1} + \overline{T3})$$

จากสมการข้างต้นเราสามารถที่จะเขียนเป็นวงจรแลดเดอร์ได้ตามการกระทำทางลอจิกของอุปกรณ์หรือหน้าสัมผัสของตัวแปรในสมการข้างต้นนั้นได้เลย และจากสมการข้างต้นนั้นจะเขียนเป็นวงจรแลดเดอร์ได้ดังนี้ คือ



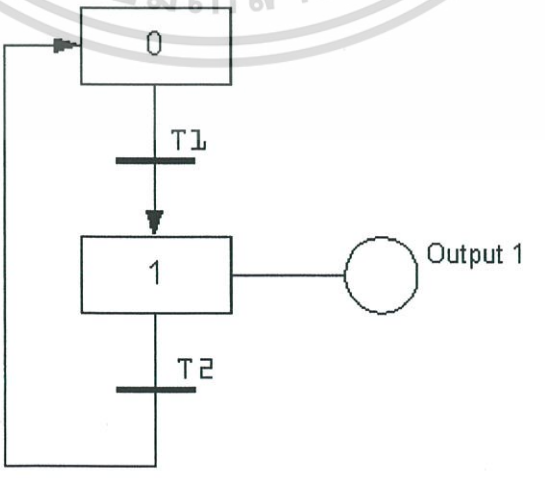
รูปที่ 3.9 ตัวอย่างของการแปลงจาก Petri net เป็นวงจรแลดเดอร์

ถ้าเราใช้วิธีการที่กล่าวมาในข้างต้นวงจรแลดเดอร์ที่ได้มานั้นจะยาวมากดั้งนั้นในกรณีที่เงื่อนไขการทำงานเป็นวงรอบที่แน่นอนเราสามารถที่จะใช้ฟังชั่นก์ของ PLC ที่เกี่ยวกับการเลื่อนสถานะการทำงาน เช่น ชีฟรีจิสเตอร์ (shift register) มาช่วยในการเปลี่ยนสถานะการทำงานได้ซึ่งในเงื่อนไขที่สามารถทำแบบนี้ได้นั้น จะแบ่งการทำงานออกได้เป็น 2 แบบ คือ

1. เงื่อนไขการทำงานแบบอนุกรม หรือ เงื่อนไขที่เป็นลำดับขั้นตอนการทำงานที่แน่นอน เช่น ทำงานจากกระบวนการที่ 1 หลังจากนั้นจะทำงานในกระบวนการที่ 2, 3, 4,... และจบการทำงาน
2. เงื่อนไขการทำงานแบบขนาน หรือ เงื่อนไขการทำงานที่เป็นแบบทางเลือกว่าจะทำงานในกระบวนการใด กระบวนการหนึ่ง ไม่เป็นลำดับขั้นตอนการทำงานที่แน่นอน เช่น เลือกที่จะทำงานในกระบวนการที่ 1 หรือ ทำงานในกระบวนการที่ 2 หลังจากนั้นให้จบการทำงาน

ในการแปลงจากรูปแบบของเพททริเนทส์ เป็นวงจรแลดเดอร์ ฟังชั่นก์ของ PLC ที่เกี่ยวกับการเลื่อนสถานะการทำงานอาจจะมีหลายฟังชั่น ขึ้นอยู่กับ PLC ที่ใช้ แต่ในวิธีที่จะกล่าวถึงต่อไปนี้จะใช้ ชีฟรีจิสเตอร์ (shift register) เป็นเครื่องมือที่ใช้ในการเปลี่ยนสถานะการทำงานในแต่ละสถานะการทำงาน และก็จะขึ้นอยู่กับรูปแบบของเงื่อนไขการทำงานด้วย ในการเขียนวงจรแลดเดอร์เราจะแยกออกเป็น 2 ส่วน ส่วนของการควบคุมสถานะการทำงาน และในส่วนของการควบคุมเอาท์พุตของระบบควบคุม

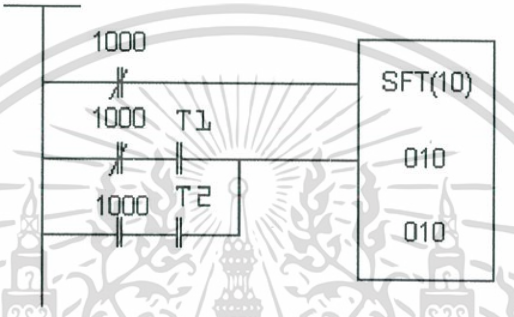
เงื่อนไขการทำงานแบบอนุกรม



รูปที่ 3.10 วงจรควบคุมในรูปแบบของเพททริเนทส์แบบอนุกรม

ส่วนของการควบคุมสถานะการทำงาน

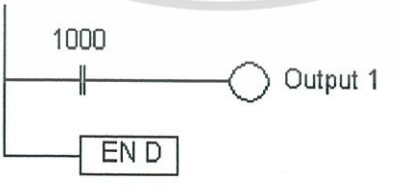
เงื่อนไขของการเข้าสู่สถานะการทำงานจะทำงานได้ก็ต่อเมื่อสถานะการทำงานที่อยู่ก่อนหน้ามันขึ้นไปมีสภาวะลอจิกเป็น 1 จากรูปที่ 3.11 ส่วนควบคุมสถานะการทำงานของวงจรควบคุมในรูปแบบของวงจรแลดเดอร์แบบอนุกรมจะเห็นได้ว่า T2 จะอยู่หลัง สถานะการทำงานที่ 1 ซึ่งจะถูกควบคุมการทำงานโดยพื้นที่ภายในของ PLC ตำแหน่งที่ 1000



รูปที่ 3.11 ส่วนควบคุมสถานะการทำงานของวงจรควบคุมในรูปแบบของวงจรแลดเดอร์แบบอนุกรม

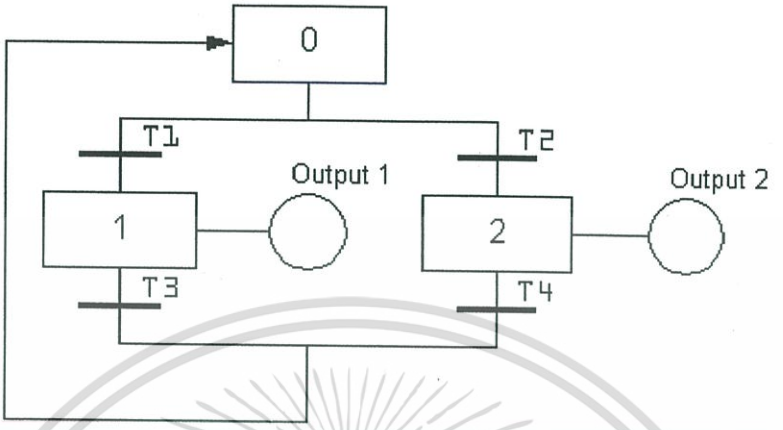
ส่วนของการควบคุมเอาท์พุตของระบบควบคุม

ในส่วนของเอาท์พุตที่จะนำไปใช้ควบคุมอุปกรณ์นั้นจะถูกควบคุมโดยพื้นที่ภายในของ PLC



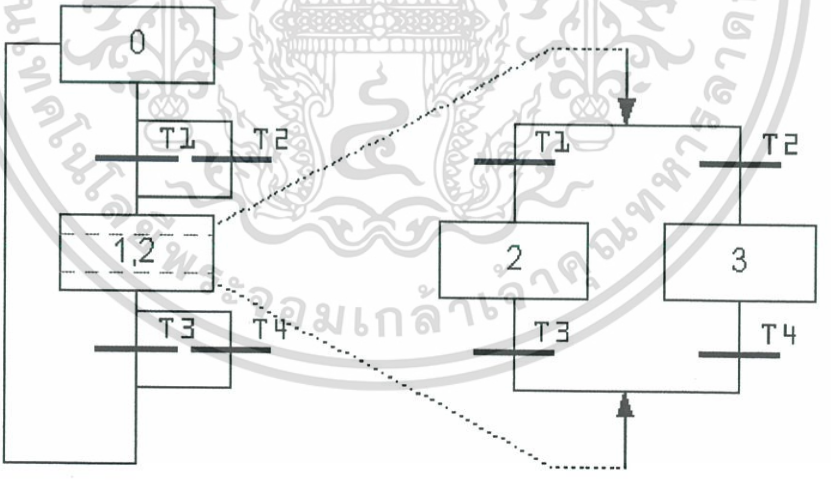
รูปที่ 3.12 ส่วนควบคุมเอาท์พุตของวงจรควบคุมในรูปแบบของวงจรแลดเดอร์แบบอนุกรม

เงื่อนไขการทำงานแบบขนาน



รูปที่ 3.13 วงจรควบคุมในรูปแบบของเพททริเนทส์แบบขนาน

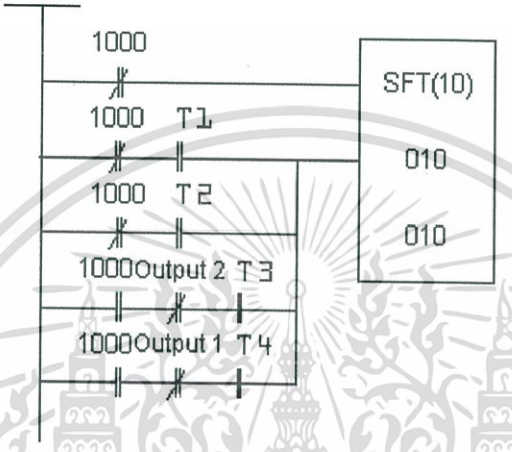
เมื่อรวมสถานะของการทำงานที่ขนานกันอยู่ให้เป็นสถานะการทำงานเดียวกันแล้วเราได้วงจรควบคุมในรูปแบบของเพททริเนทส์แบบอนุกรมดังนี้



รูปที่ 3.14 วงจรควบคุมในรูปแบบของเพททริเนทส์แบบขนานเมื่อรวมให้เป็นแบบอนุกรม

ส่วนของการควบคุมสถานะการทำงาน

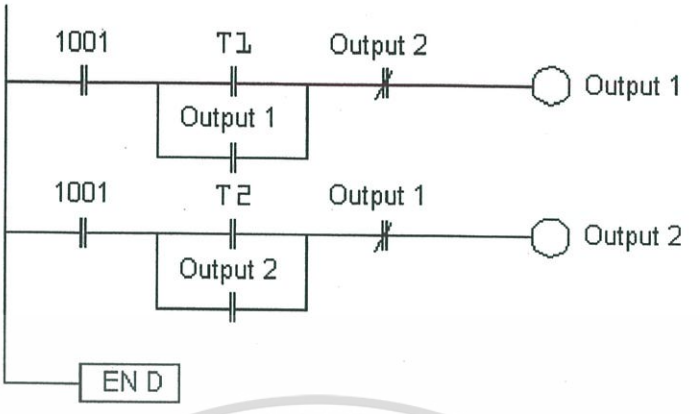
เมื่อรวมสถานะของการทำงานที่ขนานกันอยู่ให้เป็นสถานะการทำงานเดียวกันแล้วเราก็จะสามารถที่จะเขียนส่วนของการควบคุมสถานะของการทำงานได้เหมือนกับส่วนของการควบคุมสถานะของการทำงานในรูปแบบของเพททริเนทส์แบบอนุกรมได้



รูปที่ 3.15 ส่วนของการควบคุมสถานะการทำงานของวงจรควบคุมในรูปแบบของเพททริเนทส์แบบขนาน

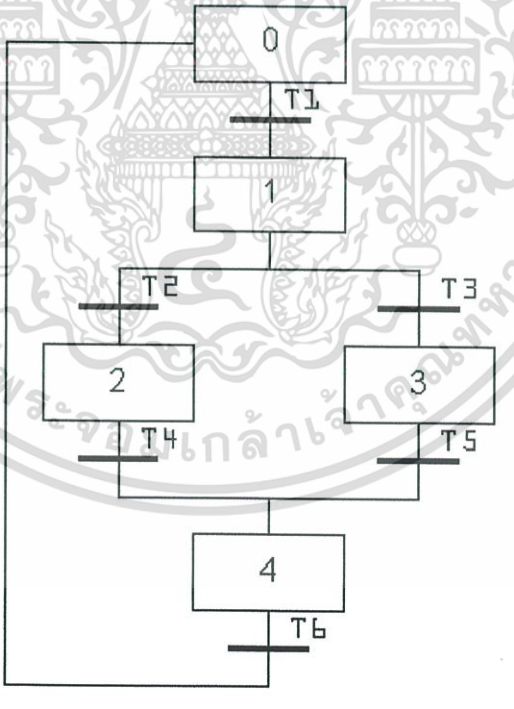
ส่วนของการควบคุมเอาท์พุตของระบบควบคุม

ถึงแม้ว่าเราจะรวมสถานะการทำงานที่ขนานกันอยู่ให้เป็นสถานะการทำงานเดียวกันแล้วแต่นั้นมันก็เป็นแค่การเปรียบเหมือนว่าเป็นสถานะการทำงานเดียวเพื่อใช้ในการเลื่อนสถานะของการทำงานเท่านั้น แต่ในส่วนของการควบคุมเอาท์พุตนั้นเราก็ยังต้องแยกกันอยู่โดยจะแยกกันหลังจากผ่านพื้นที่ภายในของ PLC ที่ใช้เป็นสถานะการทำงานแต่ละสถานะการทำงานแล้ว และในส่วนของเงื่อนไขของการเข้าสู่สถานะการทำงานเราจะต้องขนานด้วยหน้าสัมผัสของตัวเอาท์พุตเพื่อล็อคการทำงานให้อยู่ในสถานะการทำงานนั้นจนกว่าจะมีการเปลี่ยนแปลง ในกรณีที่ไม่มีหน้าสัมผัสของเอาท์พุตที่ใช้ได้ ให้สั่งออกเอาท์พุตที่พื้นที่ภายในของ PLC เพื่อใช้ช่วยในการล็อคสถานะของการทำงาน

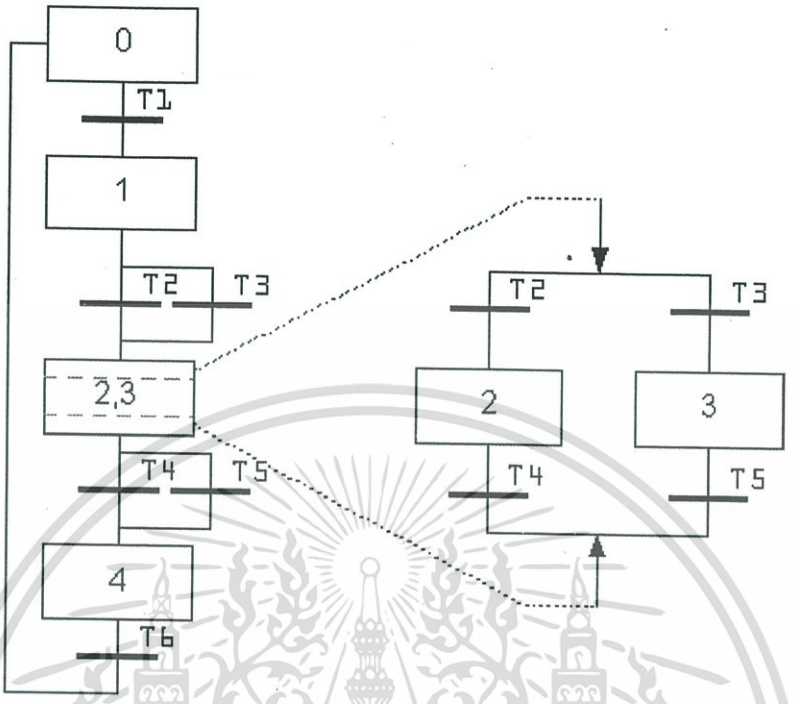


รูปที่ 3.16 ส่วนของการควบคุมเอาท์พุตของวงจรควบคุมในรูปแบบของเพททริเนทส์แบบขนาน

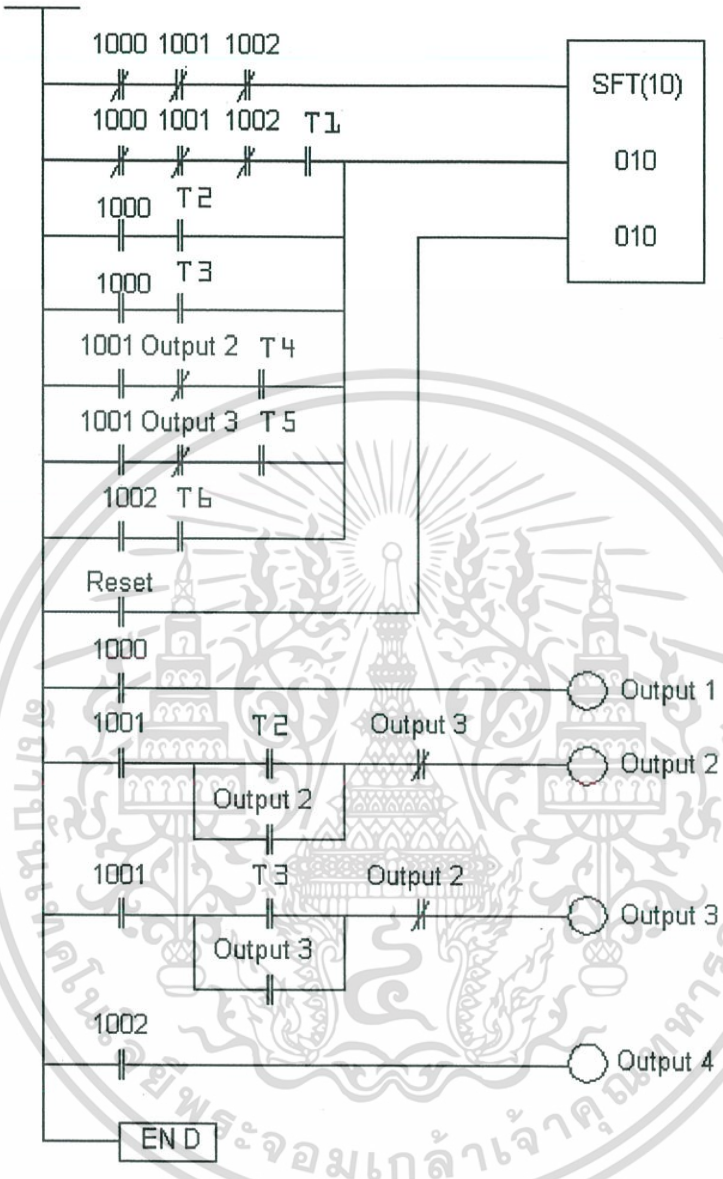
ตัวอย่างที่ 3.1 วงจรควบคุมในรูปแบบของเพททริเนทส์แบบผสม



รูปที่ 3.17 วงจรควบคุมในรูปแบบของเพททริเนทส์แบบผสม



รูปที่ 3.18 วงจรควบคุมในรูปแบบของเพททริเนทส์แบบผสมเมื่อรวมให้เป็นแบบอนุกรม



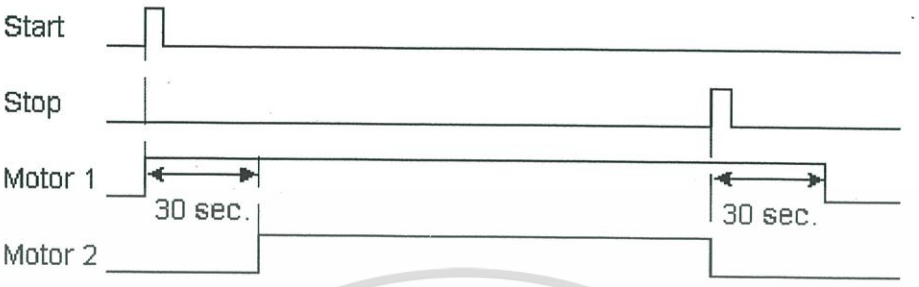
รูปที่ 3.19 วงจรควบคุมในรูปแบบของวงจรแลดเดอร์แบบผสม

ตัวอย่างที่ 3.2 การสตาร์ทมอเตอร์แบบเรียงลำดับจำนวน 2 ตัว

เงื่อนไขการทำงาน

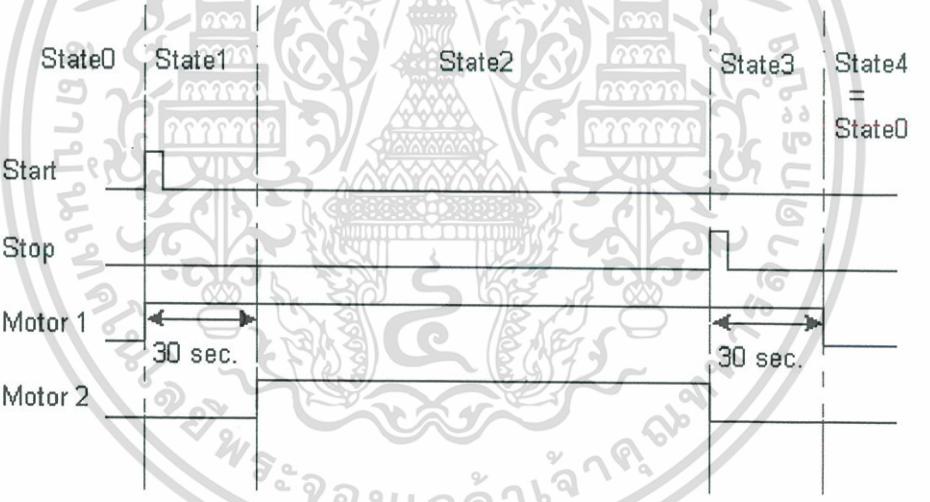
เมื่อกดสวิตช์สตาร์ทแล้วให้มอเตอร์ตัวที่หนึ่งทำงาน หลังจากนั้น 30 วินาทีให้มอเตอร์ ตัวที่สองทำงาน เมื่อกดสวิทช์หยุดการทำงานให้มอเตอร์ตัวที่สองหยุดการทำงานก่อนหลังจากนั้นเป็นเวลา 30 วินาที ให้มอเตอร์ตัวที่หนึ่งหยุดการทำงาน

ขั้นตอนที่ 1 พิจารณาเงื่อนไขการทำงานต่าง ๆ ให้เข้าใจ



รูปที่ 3.20 แผนผังเวลาการทำงาน

ขั้นตอนที่ 2 พิจารณาที่เอาท์พุต เพื่อจัดสถานะการทำงาน



รูปที่ 3.21 การแบ่งสถานะการทำงาน

สถานะที่ 0 คือ สถานะเริ่มต้นไม่มีเอาท์พุตตัวใดทำงาน
สถานะที่ 1 มอเตอร์ตัวที่ 1 ทำงาน และเริ่มนับเวลา
สถานะที่ 2 มอเตอร์ตัวที่ 1 และมอเตอร์ตัวที่ 2 ทำงาน
สถานะที่ 3 มอเตอร์ตัวที่ 1 ทำงานและเริ่มนับเวลา
สถานะที่ 4 หยุดทำงานทั้งหมดกลับเข้าสู่สถานะเริ่มต้น

ขั้นตอนที่ 3 พิจารณาเงื่อนไขที่ทำให้สถานะการทำงานเปลี่ยนแปลง

จากรูปที่ 3.2 เมื่อเราพิจารณาจากเงื่อนไขการเปลี่ยนแปลงสถานะการทำงานต่าง ๆ จะได้สถานะของการทำงานดังนี้ คือ

เริ่มเข้าสถานะที่ 0 จะเป็นสัญญาณเริ่มทำงานโปรแกรม

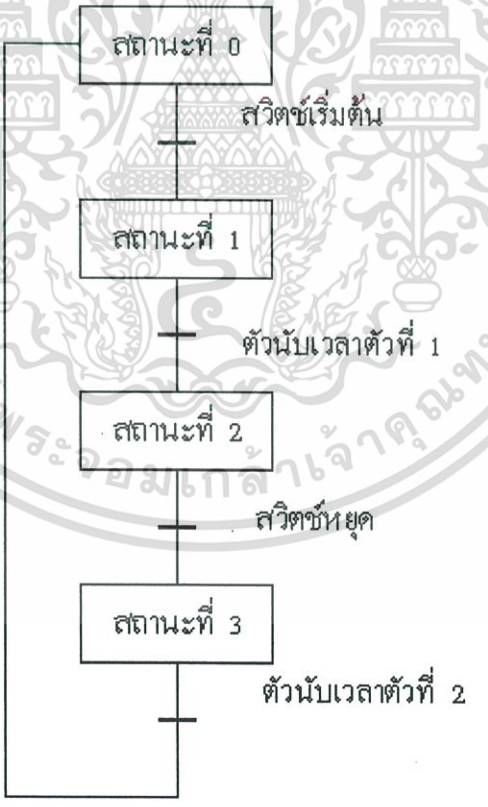
สถานะที่ 0 เป็นสถานะที่ 1 คือ สวิตช์เริ่มต้น

สถานะที่ 1 เป็นสถานะที่ 2 คือ ตัวนับเวลาตัวที่ 1

สถานะที่ 2 เป็นสถานะที่ 3 คือ สวิตช์หยุด

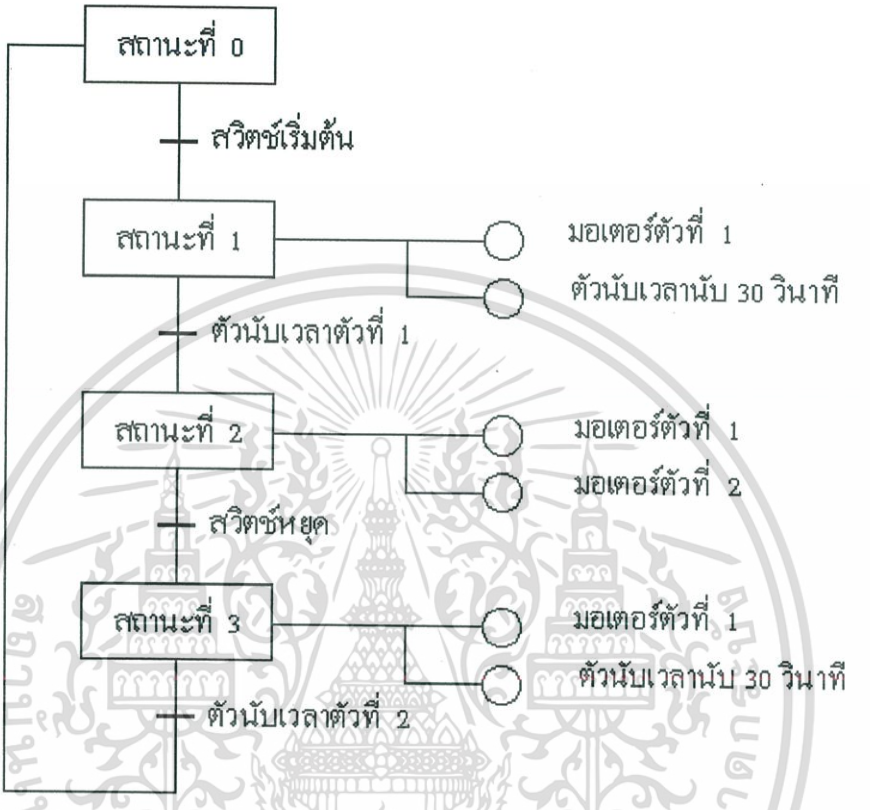
สถานะที่ 3 เป็นสถานะที่ 4 คือ ตัวนับเวลาตัวที่ 2

ขั้นตอนที่ 4 เขียนวงจรควบคุม



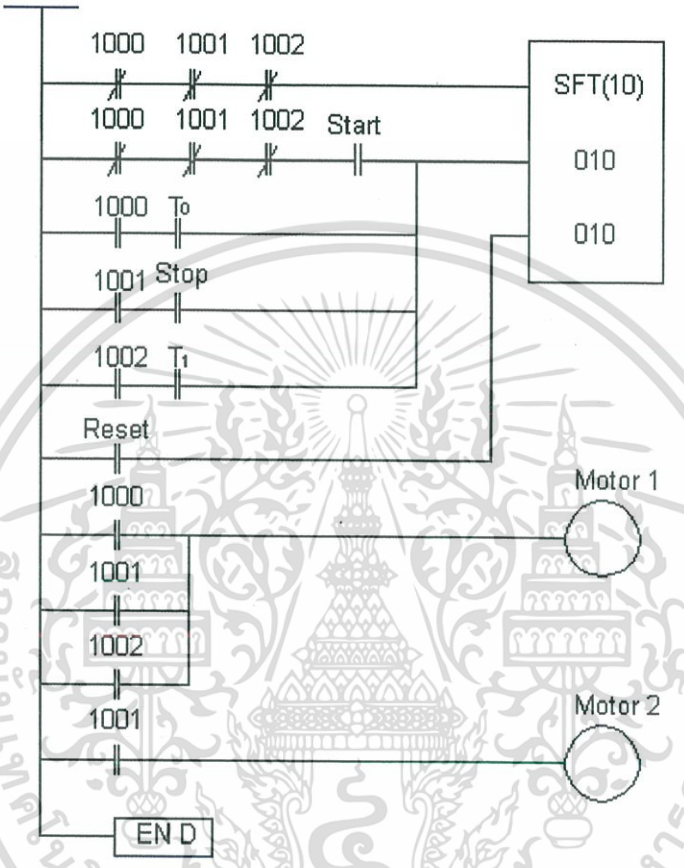
รูปที่ 3.22 วงรอบการควบคุม

ขั้นตอนที่ 5 ระบุ อุปกรณ์ต่าง ๆ ในแต่ละสถานะการทำงาน



รูปที่ 3.23 วงจรการควบคุมในรูปแบบของเพททริเนทส์

ขั้นตอนที่ 6 เปลี่ยนจากรูปแบบของ Petri Nets เป็นวงจรแลดเดอร์



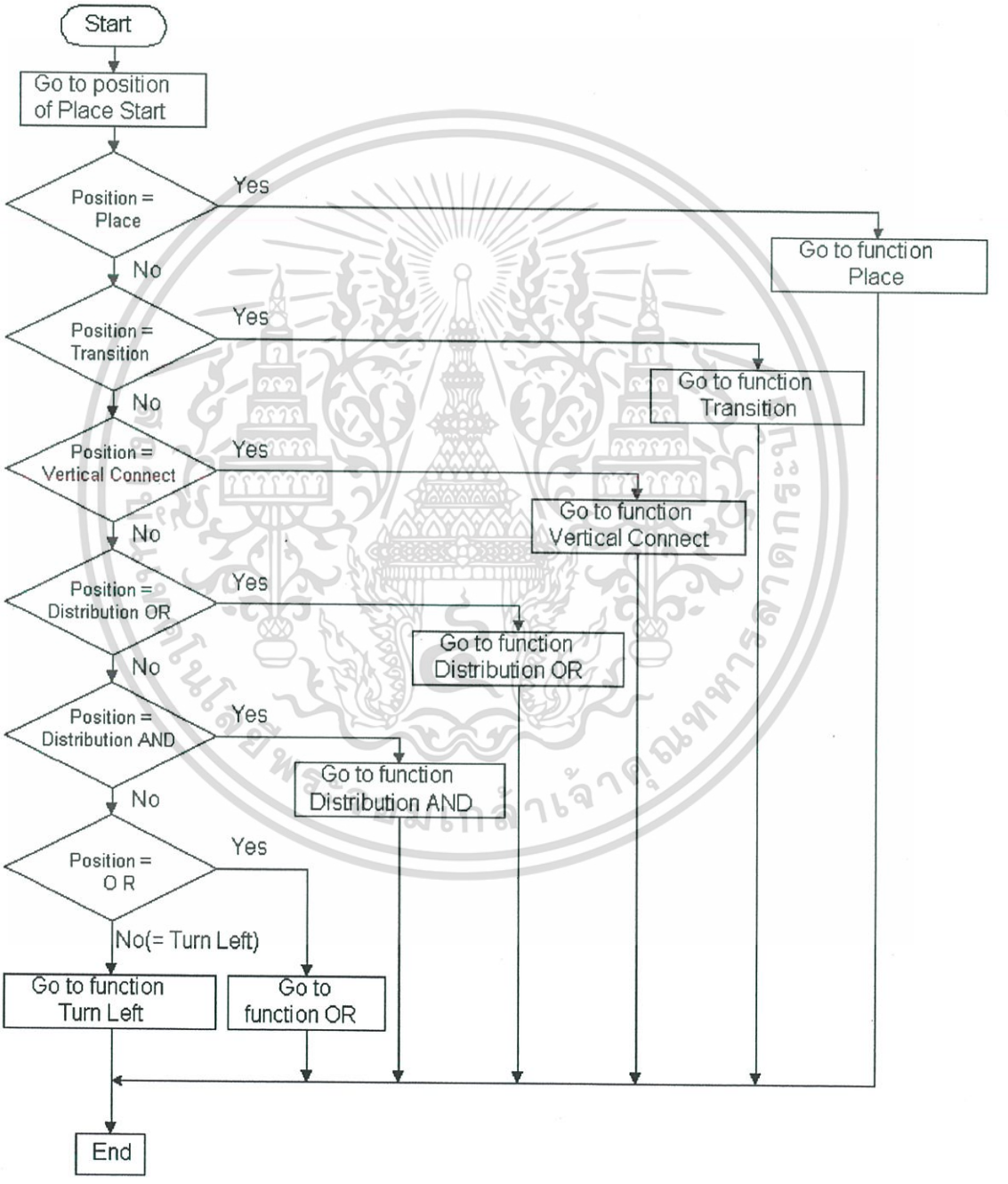
รูปที่ 3.24 วงจรการควบคุมในรูปแบบของวงจรแลดเดอร์

3.2 การออกแบบโปรแกรม

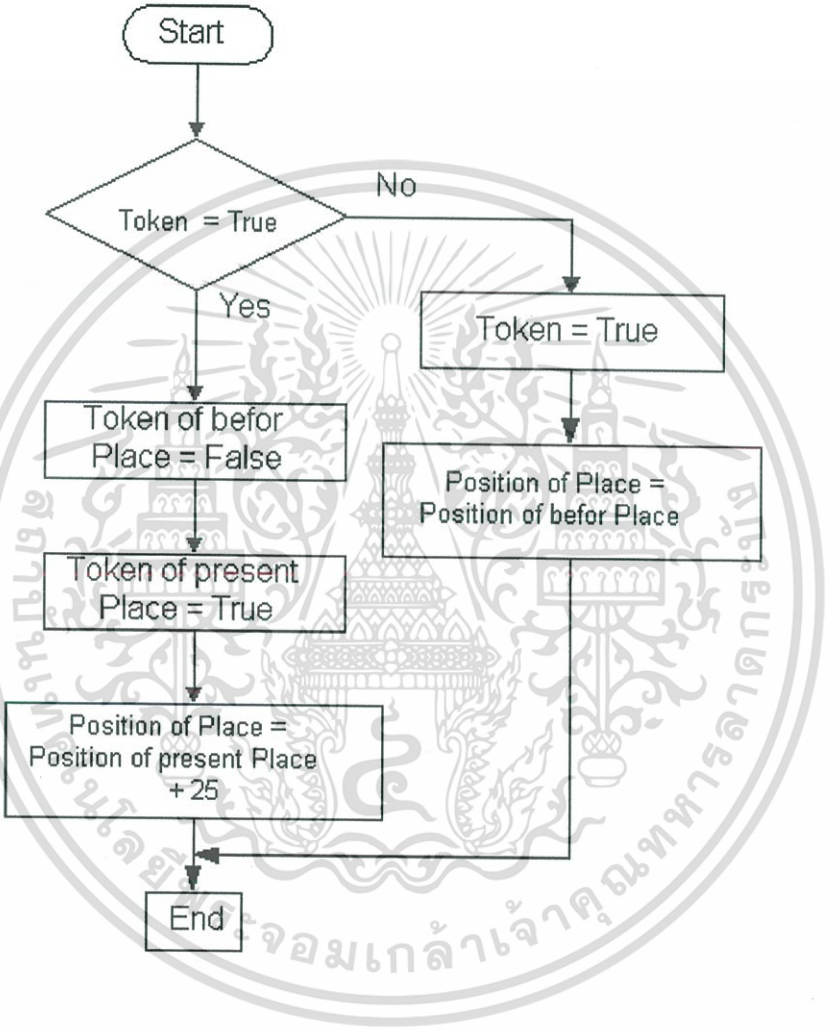
เพื่อที่จะแสดงว่าสามารถที่จะใช้ทฤษฎี Petri Net ในการควบคุมและจำลองการทำงานของเครื่องจักรกลต่าง ๆ ได้จริง ซึ่งได้เขียนโปรแกรมขึ้นมาเพื่อให้ผู้ใช้งานเขียนโปรแกรมควบคุมเครื่องจักรกล ตรวจสอบวงจรควบคุมและจำลองเหตุการณ์และเงื่อนไขต่าง ๆ ของเครื่องจักรกลที่จะเกิดขึ้นบนคอมพิวเตอร์โดยไม่จำเป็นต้องต่อเข้ากับอุปกรณ์จริงก่อนที่จะนำวงจรนี้ไปใช้ในการควบคุมเครื่องจักรกลจริง ๆ หรือจะต่อเข้ากับอุปกรณ์จริงเพื่อให้ควบคุมเครื่องจักรกลเลยก็ได้

ในการเขียนโปรแกรมนั้นได้ออกแบบโปรแกรมให้มีลักษณะเป็นฟังก์ชัน ซึ่งจะถูกเรียกขึ้นมาใช้เมื่อโปรแกรมตรวจสอบเจออุปกรณ์ต่าง ๆ และมีสมมติฐานว่าเวลาในการประมวลผลของโปรแกรมที่เขียนขึ้นมาในแต่ละบรรทัดนั้นน้อยมาก ๆ

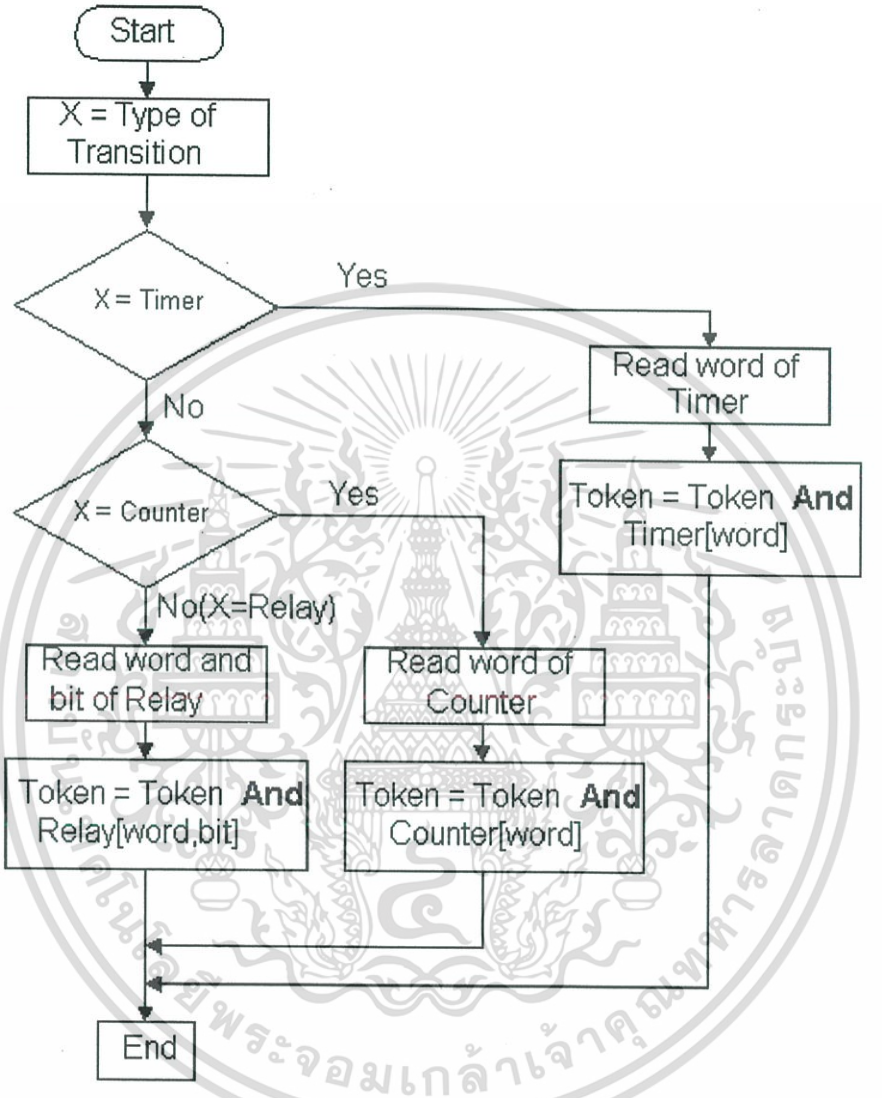
ในหัวข้อนี้จะแสดงโฟว์ชาร์ตที่สำคัญ ๆ ในการเขียนโปรแกรมนี้เท่านั้น



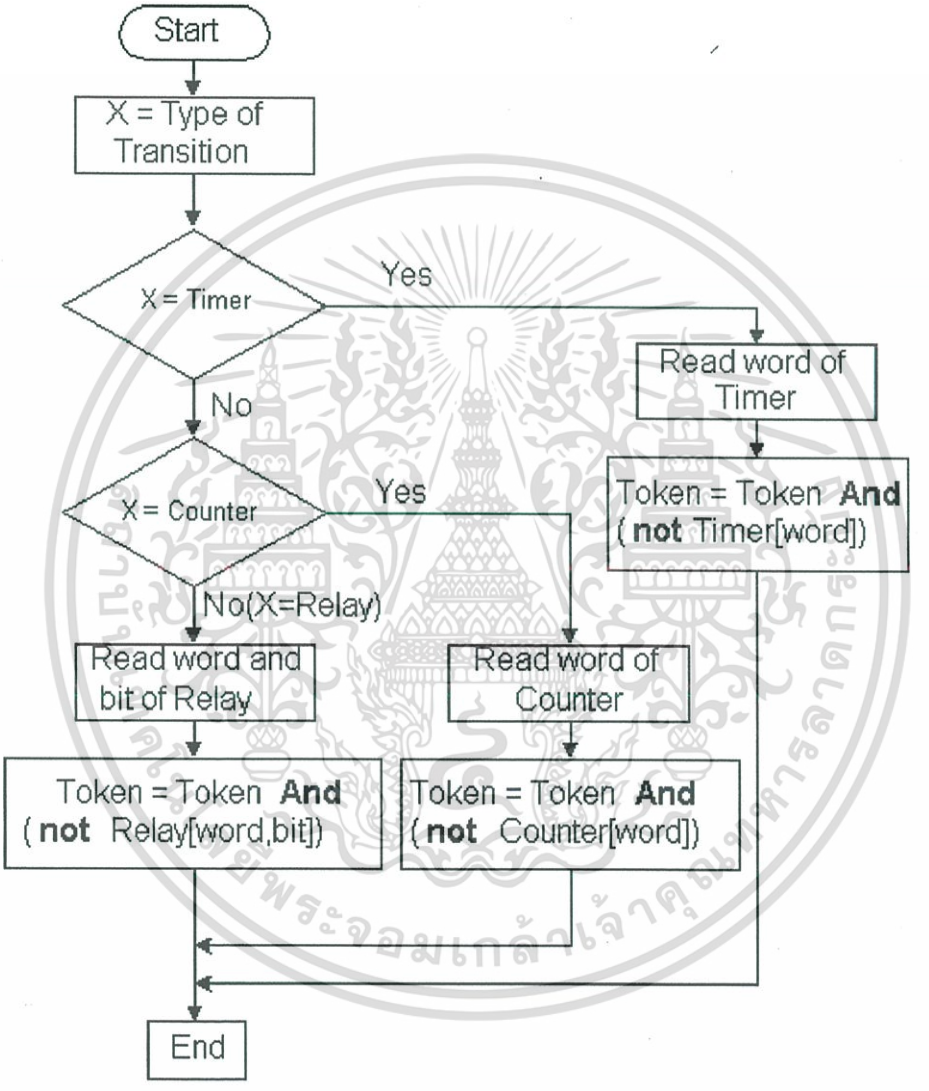
รูปที่ 3.25 โฟว์ชาร์ตแสดงการทำงานของฟังก์ชัน RUN



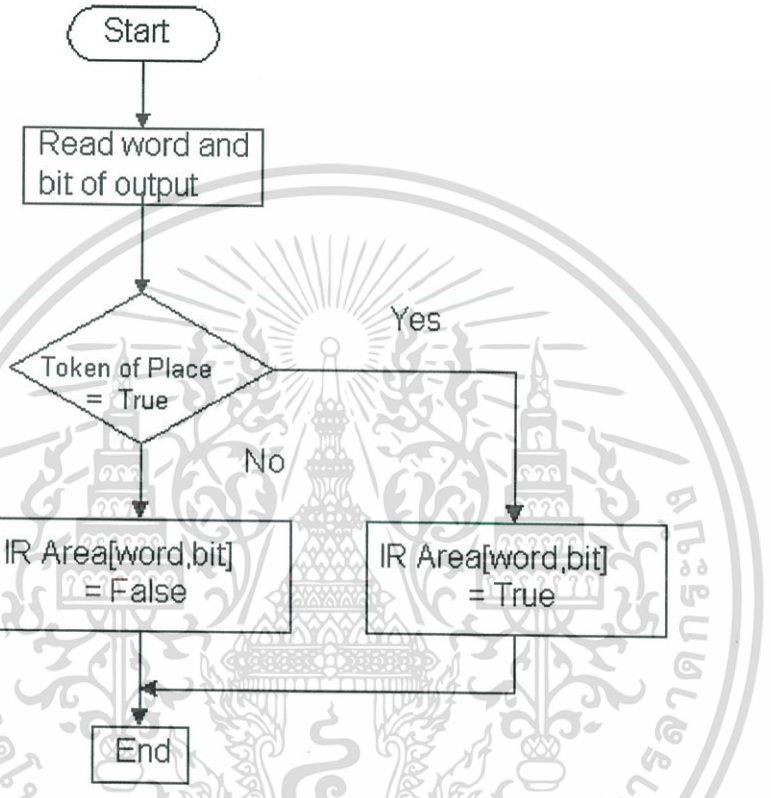
รูปที่ 3.26 โฟว์ชาร์ตแสดงการทำงานของฟังก์ชัน Place



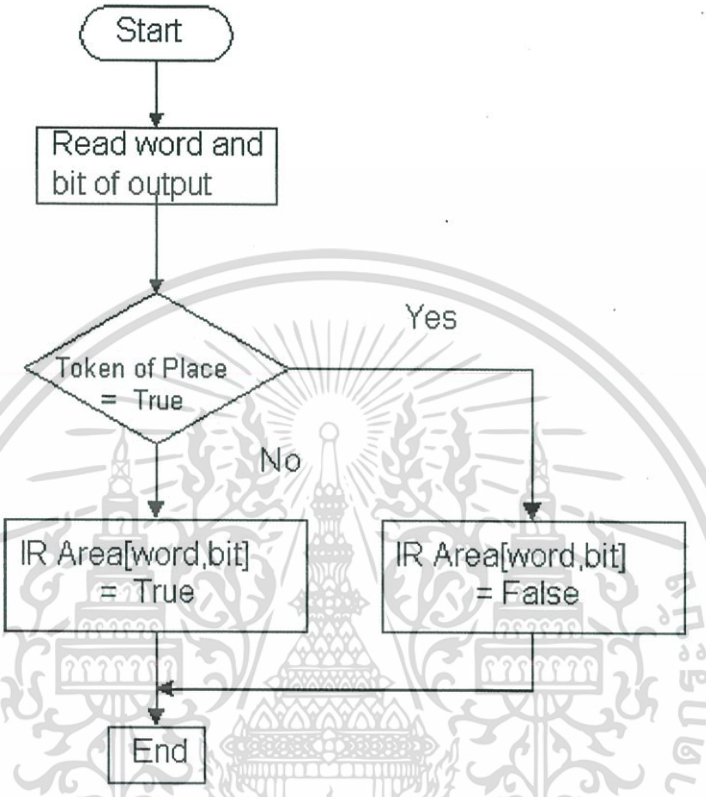
รูปที่ 3.27 โฟว์ชาร์ตแสดงการทำงานของฟังก์ชัน Transition



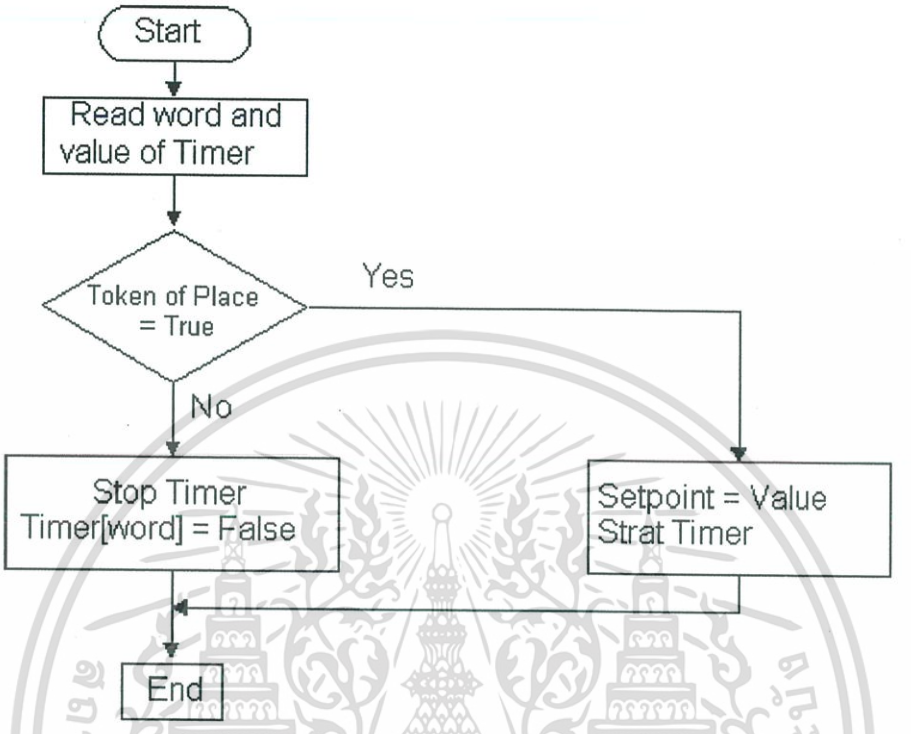
รูปที่ 3.28 โฟว์ชาร์ตแสดงการทำงานของฟังก์ชั่น Transition Not



รูปที่ 3.29 โฟว์ชาร์ตแสดงการทำงานของฟังก์ชั่น Output



รูปที่ 3.30 โฟว์ชาร์ตแสดงการทำงานของฟังก์ชั่น Output Not



รูปที่ 3.31 โฟว์ชาร์ตแสดงการทำงานของฟังก์ชั่น Timer

บทที่ 4
เกี่ยวกับโปรแกรม และการใช้งานโปรแกรม

โปรแกรมที่เราสร้างขึ้นมานี้เป็นโปรแกรมที่ใช้สำหรับประมวลผล เพื่อใช้ในการควบคุมอุปกรณ์ต่างๆ แบบ on – off Control โดยที่โปรแกรมนี้เขียนมาจากโปรแกรม Delphi และเขียนบนพื้นฐานของทฤษฎี Grafcet ซึ่งเป็นทฤษฎีที่ประยุกต์มาจาก ทฤษฎีของ Petri Nets และใช้สำหรับการควบคุมแบบ Logic Control

4.1 ความสามารถของโปรแกรม

โปรแกรมนี้ได้สร้างขึ้นมาบนพื้นฐานของทฤษฎี Petri Nets ซึ่งจะใช้สำหรับเขียนโปรแกรมควบคุมอุปกรณ์แบบ on – off Control โดยที่จะรับค่ามาจากหน่วยอินพุตโมดุล ของ PLC จากนั้นจะทำการประมวลผลตามโปรแกรมที่ผู้ใช้งานเขียนขึ้นมาแล้วส่งค่าไปยังเอาท์พุตโมดุลของ PLC เพื่อนำค่าสถานะทางลอจิกที่ได้นั้นไปควบคุมอุปกรณ์ที่ต้องการ และในโปรแกรมนี้ยังมีความสามารถที่จะช่วยตรวจสอบ deadlock ที่เกิดขึ้นในโปรแกรมที่ผู้ใช้งานเขียนขึ้นมา ซึ่งเป็นสาเหตุของปัญหาในการทำงานของระบบควบคุมต่างๆ และนอกจากนี้โปรแกรมนี้ยังสามารถที่จะจำลองเหตุการณ์ต่าง ๆ ของโปรแกรมที่ผู้ใช้งานเขียนขึ้นก่อนการใช้งานจริงได้บนคอมพิวเตอร์ได้โดยไม่จำเป็นต้องต่อเข้ากับ PLC ก็ได้

4.2 การติดตั้งใช้งาน

โปรแกรมนี้ได้ถูกสร้างขึ้นมาจากโปรแกรม Delphi ซึ่งเป็นเครื่องมือในการสร้างและพัฒนาโปรแกรมขึ้นมา อีกทั้งยังสามารถที่จะคอมไพล์โปรแกรมที่ผู้ใช้งาน สร้างขึ้นมาให้เป็นไฟล์นามสกุล .EXE ซึ่งผู้ใช้งานจะสามารถนำไฟล์ดังกล่าวเพียงไฟล์เดียวไปวันที่ PC เครื่องใดก็ได้ แต่ในโปรแกรมที่สร้างขึ้นมานี้จะต้องอ้างถึงไฟล์ข้อมูล ซึ่งเป็นรูปภาพต่าง ๆ ที่จะถูกเรียกขึ้นมาในขณะที่ผู้ใช้งานโปรแกรม เขียนโปรแกรมควบคุมขึ้นมา ดังนั้นจึงต้องนำไฟล์รูปภาพต่าง ๆ นี้ใส่เข้าไปด้วยโดยจะต้องใส่ให้ถูกตำแหน่งด้วยไม่เช่นนั้นโปรแกรมจะไม่สามารถทำงานได้ ซึ่งไฟล์ และตำแหน่ง ดังกล่าวก็คือให้คัดลอกแฟ้มข้อมูล ที่ชื่อว่า picture project ไปไว้ใน Drive C:\ ซึ่งจะเป็นที่สำหรับการอ้างถึงของโปรแกรม

4.3 หน้าจอและส่วนประกอบที่สำคัญ

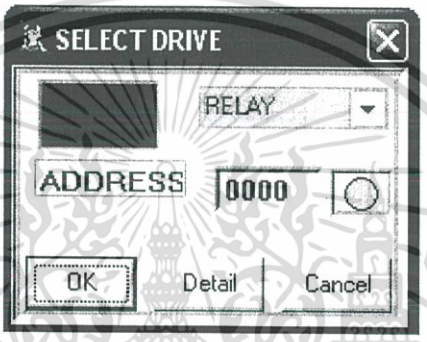
4.3.1 ฟอร์มหลัก

ฟอร์มหลักนี้เป็นฟอร์มเริ่มต้นในการใช้งานโปรแกรมนี้ ซึ่งจะปรากฏขึ้นเมื่อเราเปิดโปรแกรมนี้ขึ้นมาใช้งาน ในขณะนี้จะยังไม่มีพื้นที่ในการเขียนโปรแกรม ซึ่งผู้ใช้งานจะต้องเลือกว่าจะสร้างงานใหม่หรือจะเปิดงานเก่าออกมาแก้ไขหรือใช้งานก็ได้

รูปที่ 4.1 ฟอร์มหลัก

4.3.2 ฟอร์ม Select Drive

ฟอร์ม Select Drive จะปรากฏเมื่อผู้ใช้งานคลิกเลือก Transition ฟอร์มนี้ใช้กำหนดตำแหน่งและชนิดของฟังก์ชั่นการทำงาน โดยจะนำเฉพาะสภาวะเปิด-ปิด มาใช้งานทำหน้าที่เปรียบได้เสมือนสวิตซ์เมื่อฟังก์ชั่นดังกล่าวทำงาน จะขับระบบให้เปลี่ยนลำดับการทำงานไปสู่สภาวะถัดไป ปุ่ม Detail กดเพื่อแสดงตำแหน่งของหน่วยความจำที่ใช้งานได้ซึ่งสอดคล้องกับตำแหน่งอินพุตและ Timer & Counter ของ PLC ที่ใช้งาน



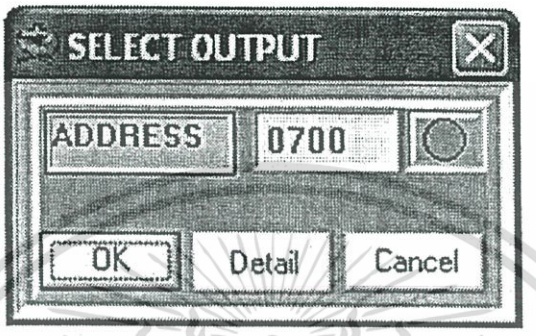
รูปที่ 4.2 ฟอร์ม Select Drive

WORD	INPUT RELAY							
00	0000	0001	0002	0003	0004	0005	0006	0007
01	0100	0101	0102	0103	0104	0105	0106	0107
02	0200	0201	0202	0202	0204	0205	0206	0207

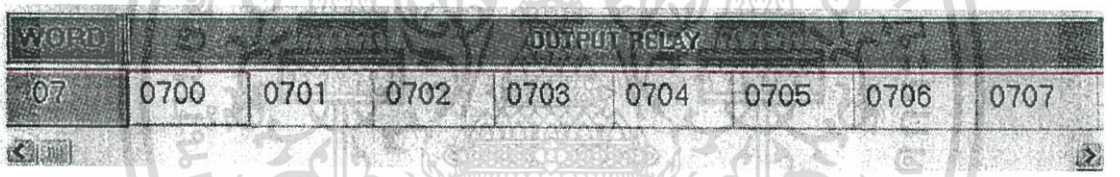
รูปที่ 4.3 ฟอร์มแสดงตำแหน่งหน่วยความจำของรีเลย์ที่สามารถใช้งานได้

4.3.3 ฟอร์ม Select output

ฟอร์ม Select Output นี้จะเป็นฟอร์มที่ใช้สำหรับผู้ใช้งานโปรแกรม เลือกตำแหน่งของเอาท์พุตที่ใช้ในการออกแบบ สำหรับปุ่ม Detail ในที่นี้ใช้แสดงหน่วยความจำของเอาท์พุตซึ่งสอดคล้องกับตำแหน่งเอาท์พุตของ PLC ที่ใช้งาน



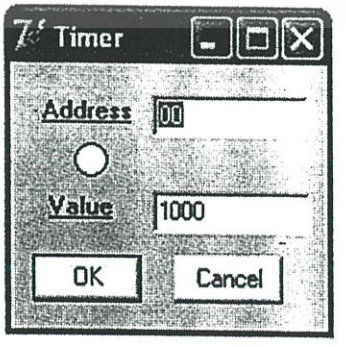
รูปที่ 4.4 ฟอร์ม Select Output



รูปที่ 4.5 ฟอร์มแสดงตำแหน่งหน่วยความจำเอาท์พุตที่สามารถใช้งานได้

4.3.4 ฟอร์ม Select Timer

ฟอร์ม Select Timer ใช้เพื่อกำหนดตำแหน่งหน่วยความจำที่ใช้ในการหน่วงเวลาและค่าเวลาที่ต้องการหน่วงเวลา



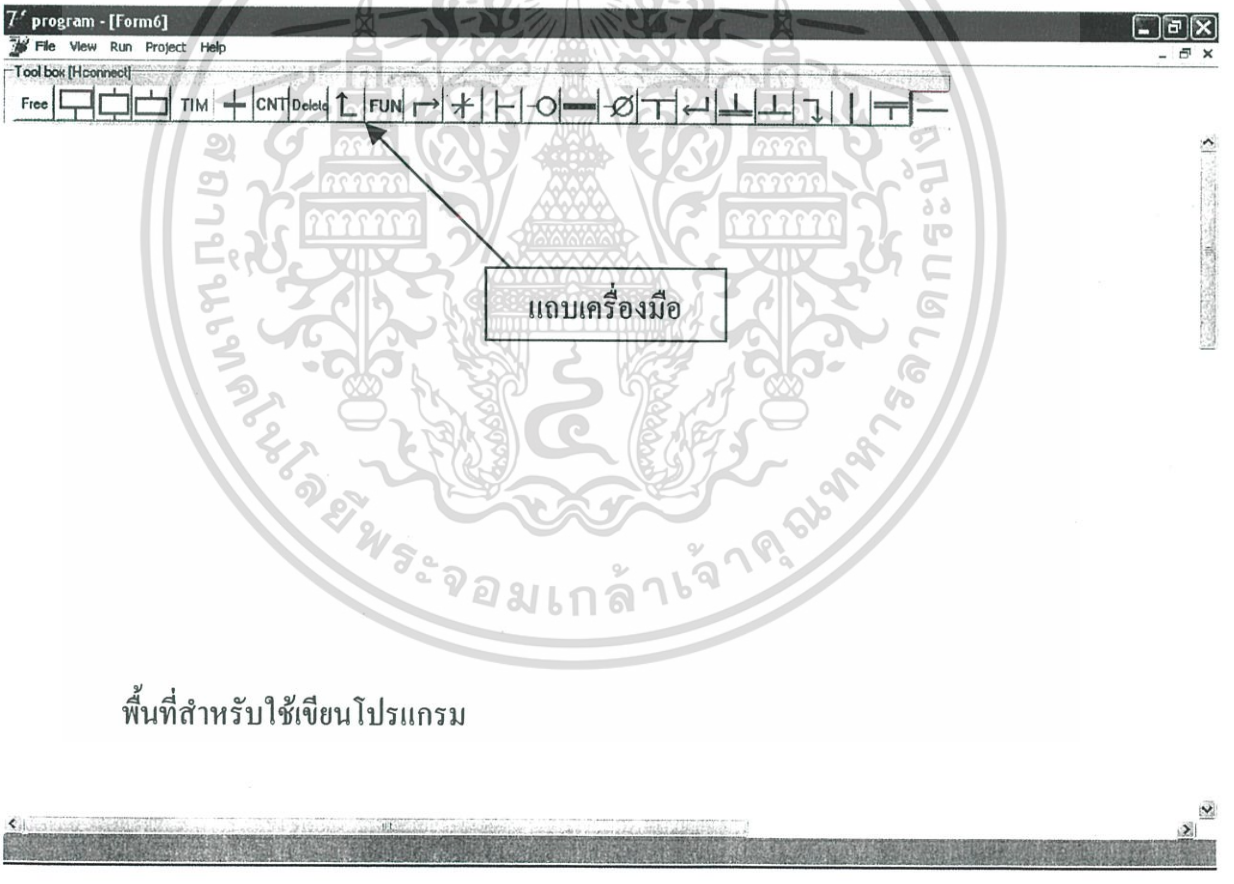
รูปที่ 4.6 ฟอร์ม Timer

4.4 การใช้งานโปรแกรม

หลังจากที่เราได้ทราบถึงหน้าที่การทำงาและส่วนประกอบต่าง ๆ ที่สำคัญต่อการใช้งานโปรแกรมแล้ว ในหัวข้อนี้จะกล่าวถึงการใช้งานโปรแกรม โดยการเขียนโปรแกรมมาควบคุมการทำงานของอุปกรณ์ภายนอกผ่าน PLC และการจำลองเหตุการณ์บนคอมพิวเตอร์

4.4.1 การเรียกใช้โปรแกรม

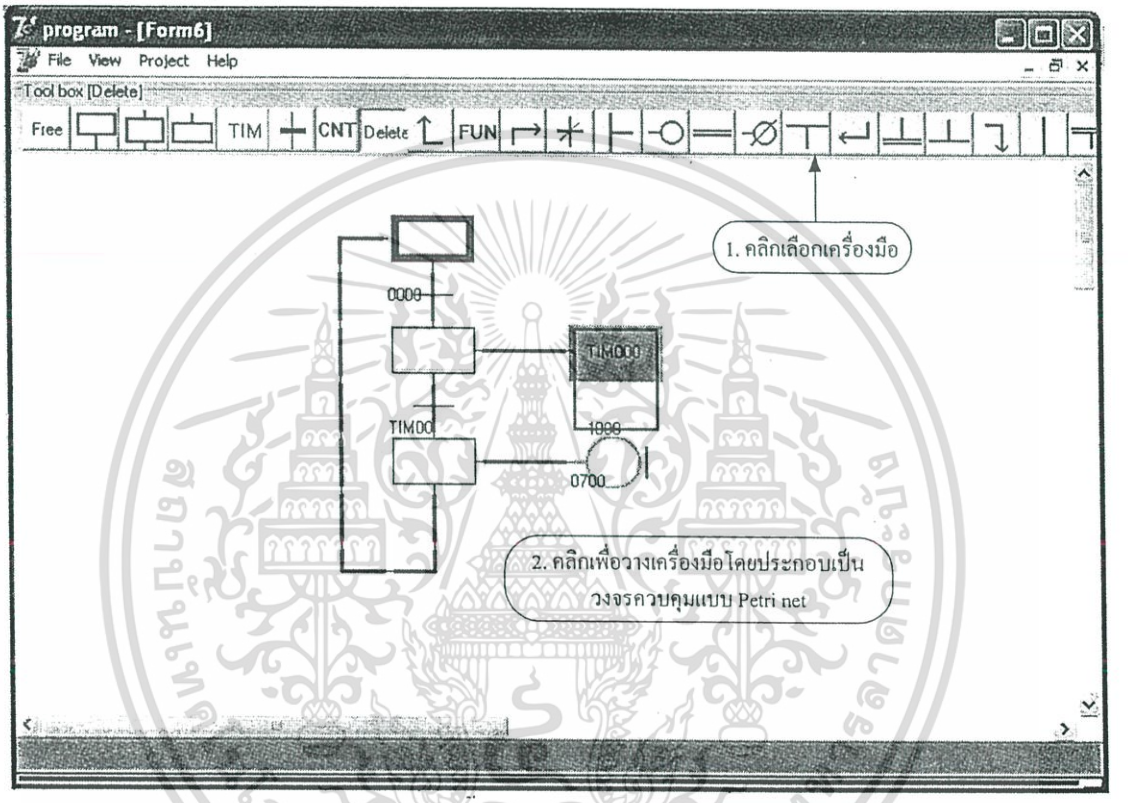
เริ่มต้นเรียกโปรแกรมโดยการดับเบิ้ลคลิกที่ไอคอนของไฟล์โปรแกรม เพื่อเปิดโปรแกรมขึ้นมาใช้งาน โดยฟอร์มเริ่มต้นนั้นจะมีเครื่องมือรูปร่างต่าง ๆ ตามรูปแบบ Petri net โดยใช้ร่วมกับพื้นที่ว่างที่ใช้วางรูปภาพเพื่อออกแบบโปรแกรมควบคุม PLC



รูปที่ 4.7 ฟอร์มเริ่มต้นการใช้งานโปรแกรม

4.4.2 การเขียนโปรแกรมควบคุม

ในการเขียนโปรแกรม มีขั้นตอนการใช้คือ ใช้เมาส์คลิกที่เครื่องมือรูปต่าง ๆ บนแถบเครื่องมือ แล้วคลิกบนพื้นที่สำหรับเขียนโปรแกรม ต่อเครื่องมือต่าง ๆ จนได้เงื่อนไขการทำงานตามที่ต้องการ



รูปที่ 4.8 แสดงขั้นตอนการเขียนโปรแกรมควบคุม

4.4.3 การจำลองสถานการณ์ (Simulation)

หลังจากเขียนวงจรควบคุมเรียบร้อยแล้ว สามารถทำการจำลองเหตุการณ์ต่าง ๆ ที่จะเกิดขึ้นจากโปรแกรมควบคุมที่เขียนมา เพื่อตรวจสอบความถูกต้องของเงื่อนไขต่าง ๆ ในโปรแกรมก่อนการนำไปควบคุม PLC จากการสั่งงานจากคอมพิวเตอร์เพื่อสร้างสัญญาณภายในคอมพิวเตอร์เท่านั้น นอกจากนี้เรายังใช้การจำลองสถานการณ์นี้เพื่อฝึกฝนทักษะในการเขียนโปรแกรมได้ขั้นตอนการจำลองสถานการณ์ทำได้ดังนี้

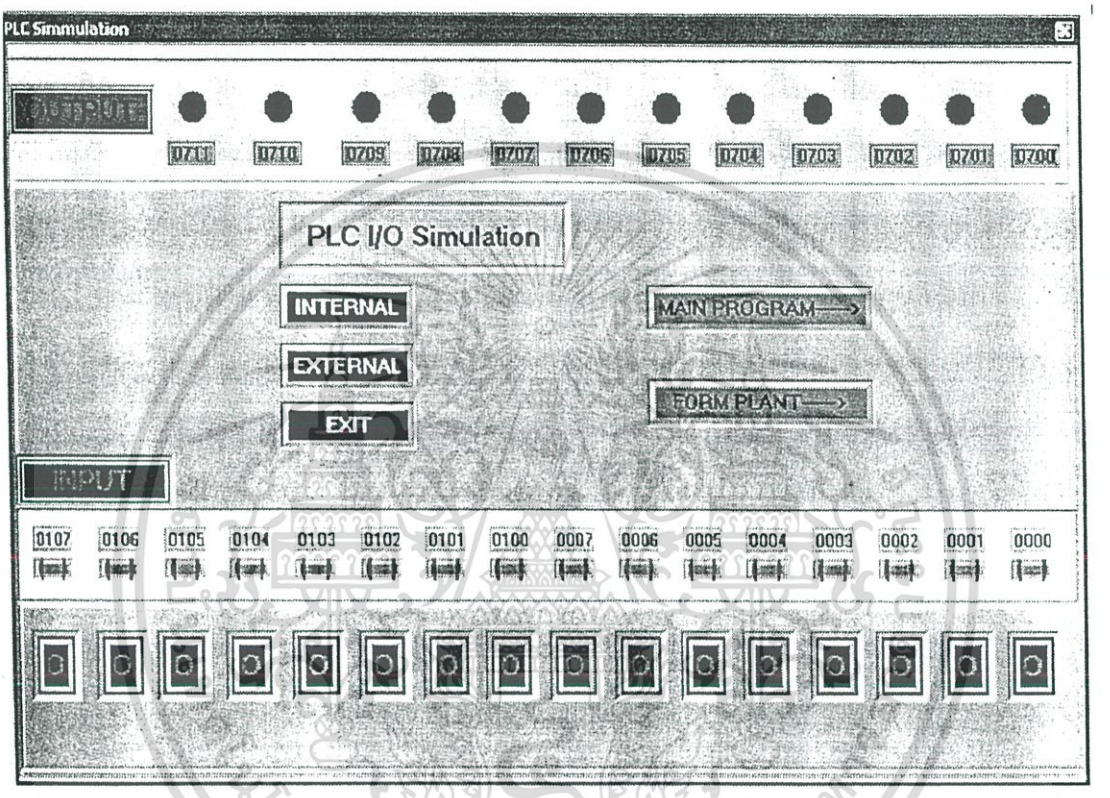
1. การคอมไพล์โปรแกรมควบคุมที่เราเขียนเสร็จและต้องการจะทดสอบโดยการเลือกที่เมนู Project → Compile

2. คลิกเมนู View → PLC Simulation

3. คลิกที่ปุ่ม Internal ในฟอร์ม PLC simulation

4. สั่งโปรแกรมทำงานโดยการคลิกที่เมนู Project → Run

5. ใส่เหตุการณ์ของสวิตซ์ต่าง ๆ บนฟอร์มของ PLC Simulation เพื่อการจำลองเหตุการณ์และตรวจสอบสภาวะของเงื่อนไขต่าง ๆ



รูปที่ 4.9 ฟอร์ม PLC Simulation

หมายเหตุ สวิตซ์ในฟอร์ม PLC Simulation มี 2 ลักษณะ คือ

1. กดติด ปล่อยดับ
2. กดติด กดดับ

การเปลี่ยนคุณลักษณะการทำงานของสวิตซ์ทำได้ดังนี้

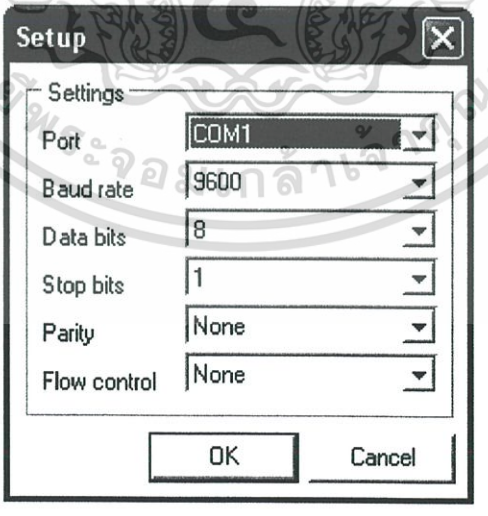
1. คลิกขวาที่ปุ่มสวิตซ์ที่ต้องการเปลี่ยนลักษณะการทำงาน
2. เลือกลักษณะการทำงานในฟอร์มที่แสดงขึ้นมา
3. เลือกกด ปุ่ม OK เพื่อยืนยันการเปลี่ยนแปลง

4.4.4 การใช้โปรแกรมควบคุมอุปกรณ์ภายนอก

เมื่อนำโปรแกรมที่จำลองเหตุการณ์ในฟอร์ม PLC Simulation ตรงตามเงื่อนไขและการทำการป้อนลง PLC แล้ว ก่อนนำไปใช้งานจริงต้องทำการทดสอบว่าเมื่อนำไปควบคุม PLC ที่ต่อร่วมกับอุปกรณ์ภายนอก โปรแกรมยังมีประสิทธิภาพเหมือนกับการจำลองเหตุการณ์ในเครื่องคอมพิวเตอร์หรือไม่

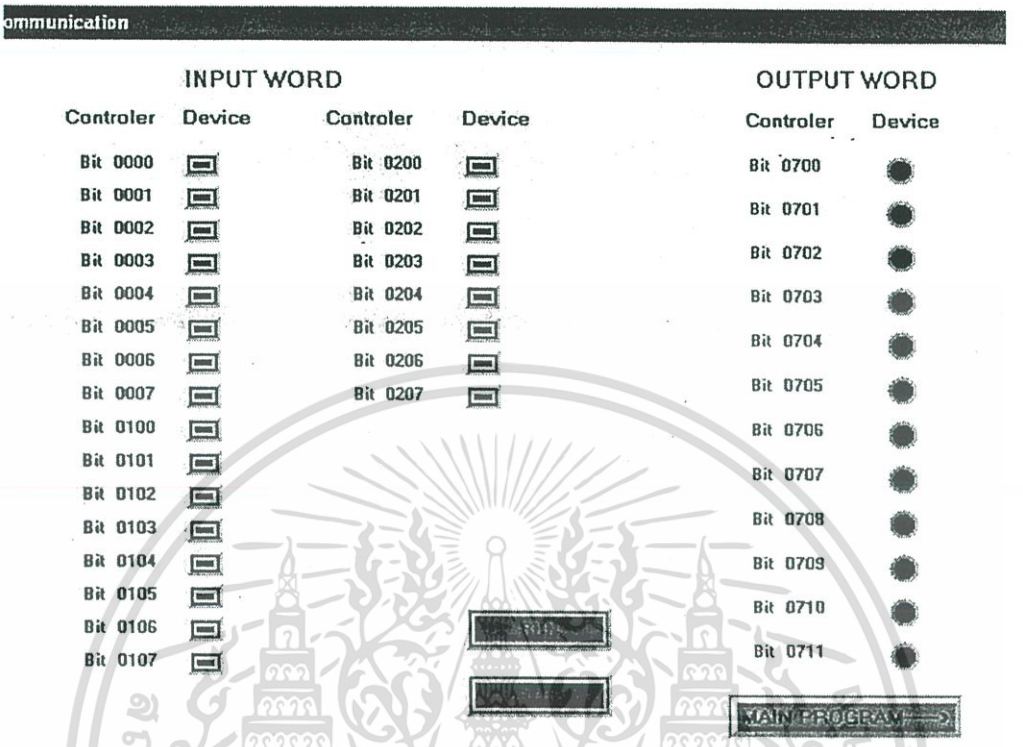
ขั้นตอนการทดสอบโปรแกรม

1. ต่อสายสัญญาณระหว่าง PLC และคอมพิวเตอร์เพื่อใช้สื่อสารข้อมูล ซึ่งรายละเอียดของการเชื่อมต่อนั้นได้กล่าวไว้แล้วในบทที่ 2
2. หลังจากต่อสายสัญญาณเรียบร้อยจึงเปิดเครื่อง PLC และเลือกสวิตซ์ให้อยู่ในตำแหน่ง Program
3. ทำการคอมไพล์โปรแกรมที่เขียนขึ้นมาโดยการเลือกที่เมนู Project → Compile
4. เลือกฟอร์มที่จะใช้แสดงผลการทำงานของโปรแกรม โดยการเลือกที่เมนู View → PLC Simulation
5. เลือก External ในฟอร์ม PLC Simulation เพื่อใช้ในการแสดงผลการควบคุม (ในกรณีนี้จะไม่สามารถกำหนดสถานะอินพุตของ PLC ได้)
6. ตั้งค่าในการสื่อสารข้อมูล โดยการเลือกที่เมนู View → Com Setup



รูปที่ 4.10 ฟอร์มการตั้งค่าในการสื่อสารข้อมูล

7. สั่งให้โปรแกรมทำงานโดยการเลือกที่เมนู Project → Run



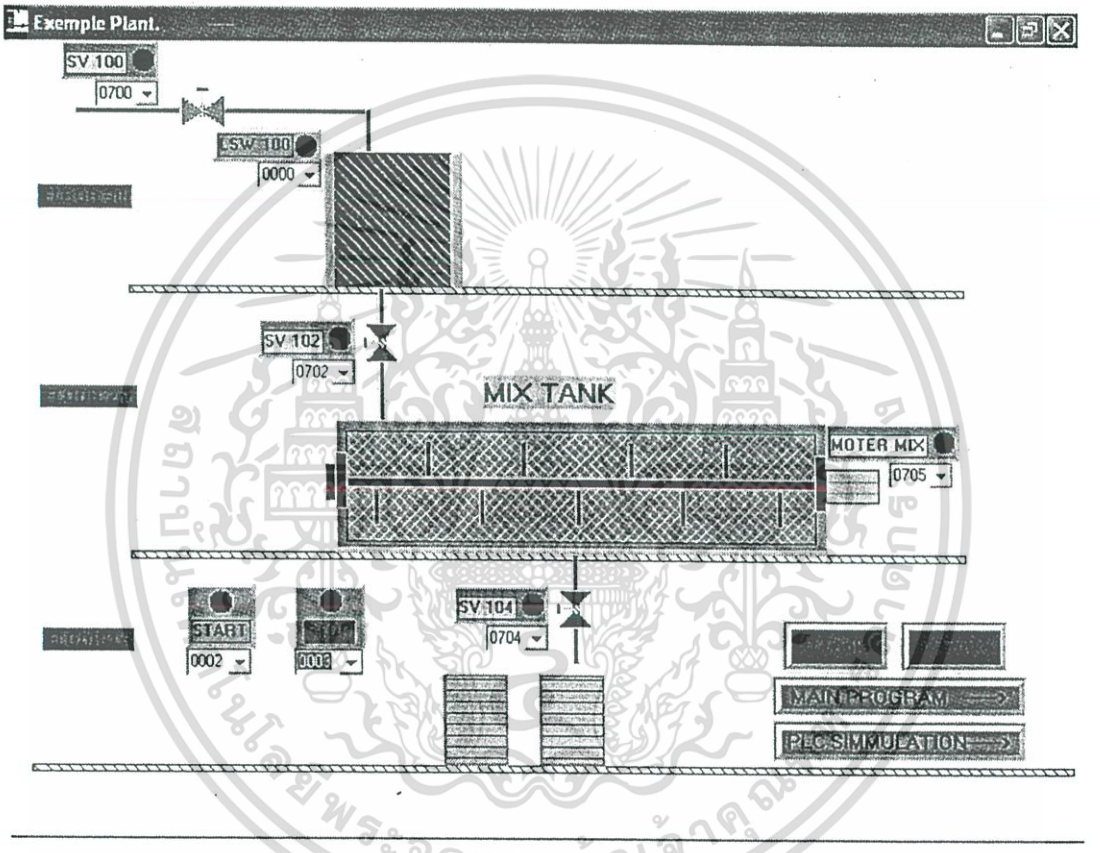
รูปที่ 4.11 ฟอร์มการติดต่อสื่อสาร

หมายเหตุ เราสามารถใช้ฟอร์มการติดต่อสื่อสารเพื่อดูสถานะการทำงานของโปรแกรมได้

บทที่ 5

การทดลอง

5.1 แบบจำลองกระบวนการผสมสารเคมี 1 ถัง



รูปที่ 5.1 แบบจำลองกระบวนการผสมสารเคมี 1 ถัง

5.1.1 เงื่อนไขการทดลอง

1. เริ่มต้นการทำงานของแบบจำลองกระบวนการผสมสารเคมี เมื่อกด Switch Star Input 002 วาล์ว SV 100 จะทำงาน เพื่อเติมสารเคมีลงใน TANK 1 ระดับของสารเคมีในถังต้องได้ระดับตามที่ Limit Switch LSW 0000 ทำงาน และวาล์ว SV 100 จะหยุดการทำงาน จึงจะทำให้ SV102 ทำงาน

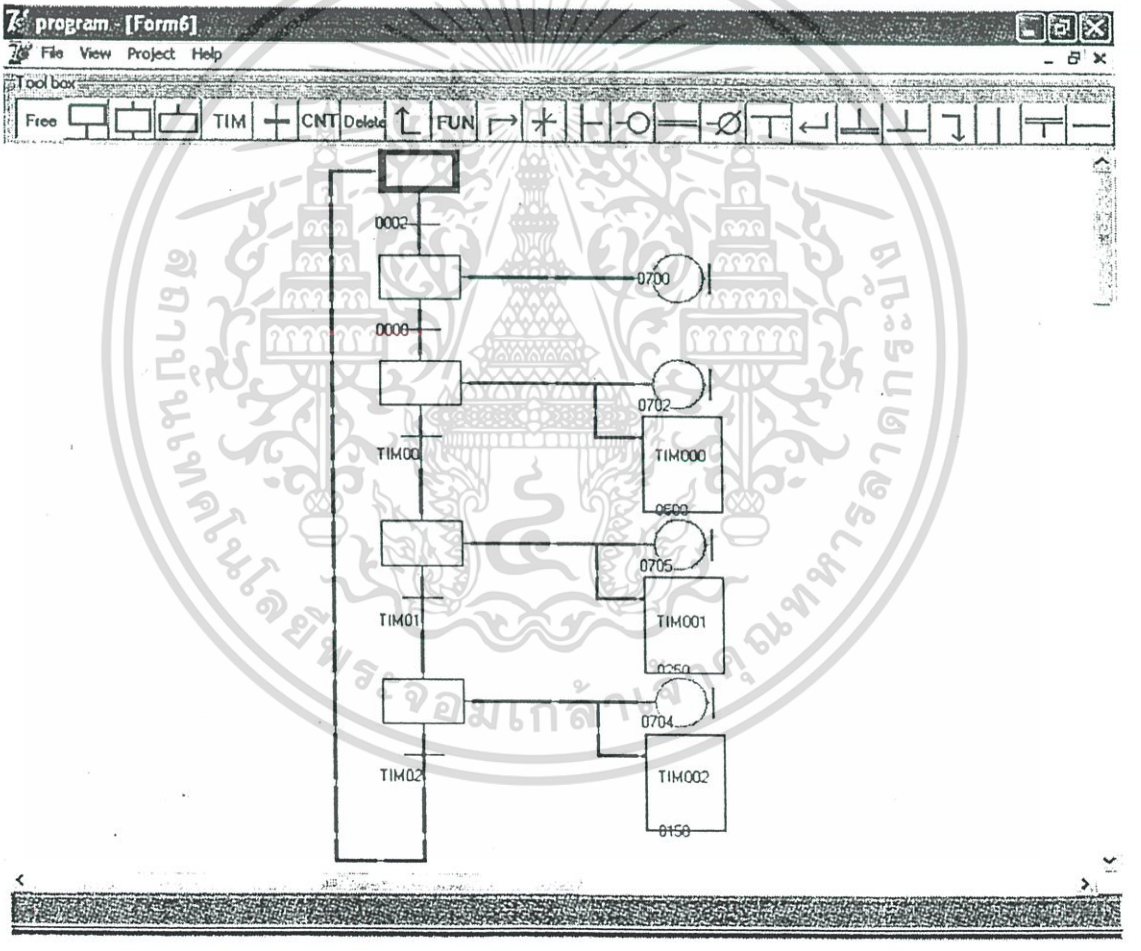
2. เมื่อ วาล์ว SV 102 ทำงาน เพื่อเติมสารเคมีลงใน MIX TANK ตั้งเวลา TIM 00 นาน 50 วินาที เมื่อเวลาครบ 50 วินาที วาล์ว SV 102 จะหยุดทำงาน

3. หลังจากนั้น MOTER MIX จะทำงานเพื่อผสมสารเคมี ตั้งเวลา TIM 01 นาน 25 วินาที เมื่อเวลาครบ 25 วินาที MOTER MIX จะหยุดการทำงาน

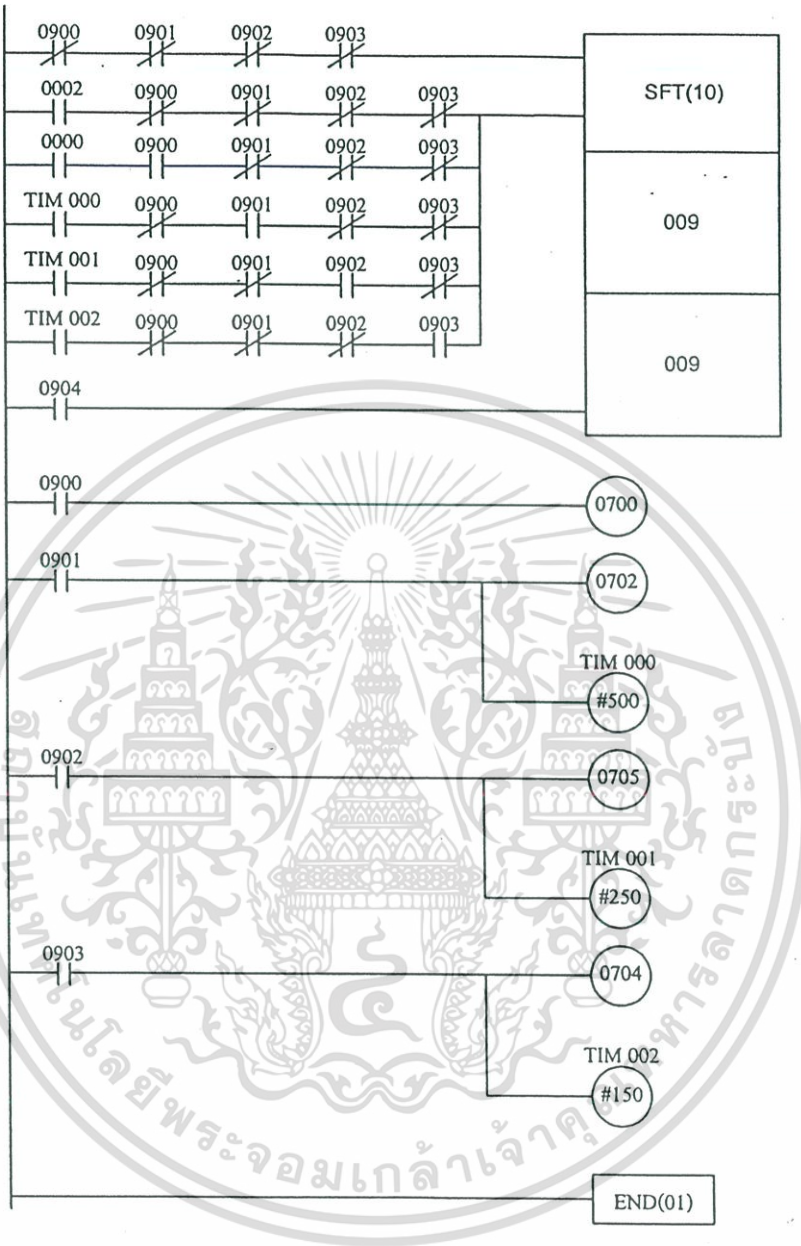
4. หลังจากนั้น วาล์ว SV 104 จะทำงาน ตั้งเวลา TIM 02 นาน 15 วินาที เมื่อเวลาครบ 15 วินาที วาล์ว SV 104 จะหยุดทำงาน

5. เมื่อต้องการเริ่มการทำงานใหม่ต้องกด Switch Start Input 0002

5.1.2 การออกแบบด้วยเพททริเน็ท



รูปที่ 5.2 วงจรควบคุมแบบจำลองกระบวนการผสมสารเคมี 1 ถัง ในรูปแบบของเพททริเน็ท



รูปที่ 5.3 วงจรควบคุมแบบจำลองกระบวนการผสมสารเคมี 1 ถัง ในรูปแบบของวงจรแลดเดอร์

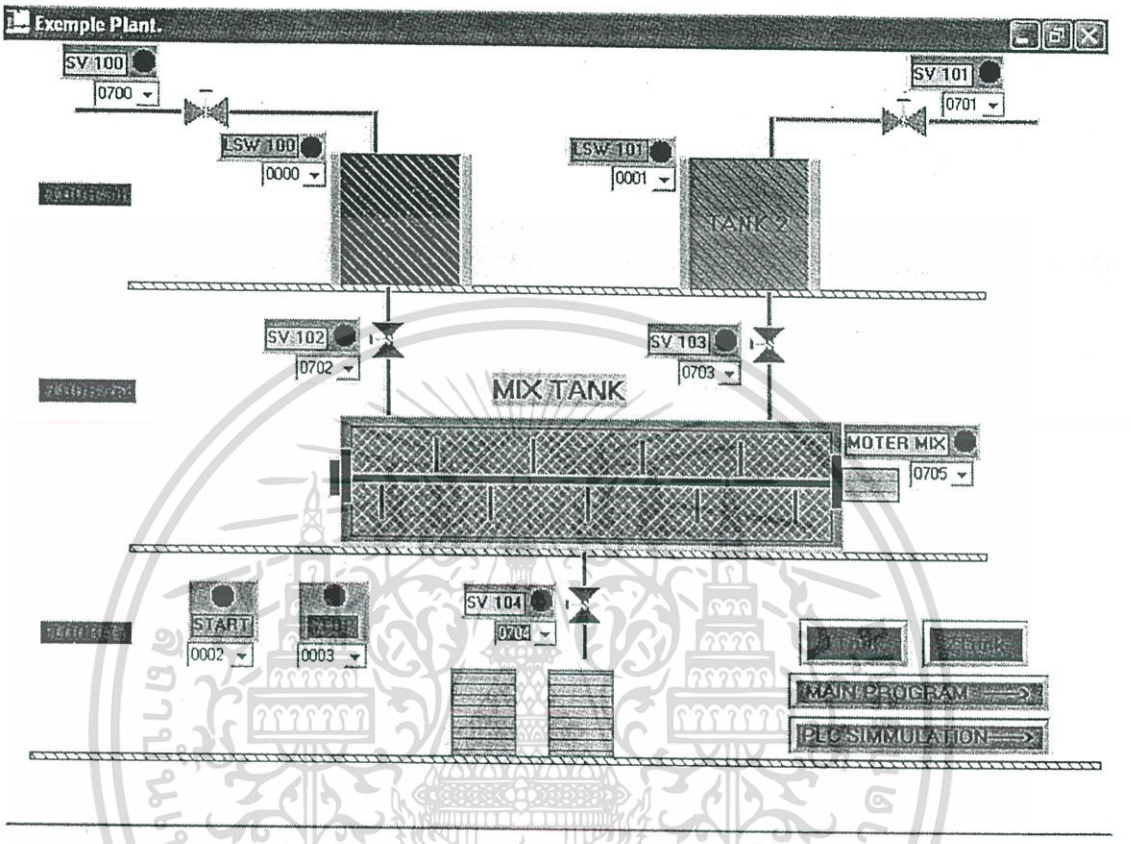
ตารางที่ 5.1 โปรแกรมควบคุมกระบวนการผสมสารเคมี 1 ถัง ในรูปแบบของภาษาบูลีน

Address	Instruction		Operation
0000	LD	NOT	0900
0001	AND	NOT	0901
0002	AND	NOT	0902
0003	AND	NOT	0903
0004	LD		0002
0005	AND	NOT	0900
0006	AND	NOT	0901
0007	AND	NOT	0902
0008	AND	NOT	0903
0009	LD		0000
0010	AND		0900
0011	AND	NOT	0901
0012	AND	NOT	0902
0013	AND	NOT	0903
0014	ORLD		
0015	LD	TIM	000
0016	AND	NOT	0900
0017	AND		0901
0018	AND	NOT	0902
0019	AND	NOT	0903
0020	ORLD		
0021	LD	TIM	001
0022	AND	NOT	0900
0023	AND	NOT	0901
0024	AND		0902
0025	AND	NOT	0903
0026	ORLD		

ตารางที่5.1 (ต่อ)

Address	Instruction		Operation
0027	LD	TIM	002
0028	AND	NOT	0900
0029	AND	NOT	0901
0030	AND	NOT	0902
0031	AND		0903
0032	ORLD		
0033	LD		0904
0034	SET (10) A		009
	DATA B		009
0035	LD		0900
0036	OUT		0700
0037	LD		0901
0038	OUT		0702
0039	TIM		000
	TIM	DATA	#500
0040	LD		0902
0041	OUT		0705
0042	TIM		001
	TIM	DATA	#250
0043	LD		0902
0044	OUT		0704
0045	TIM		002
	TIM	DATA	#150
0046	END	(01)	

5.2 แบบจำลองกระบวนการผสมสารเคมี 2 ถัง



รูปที่ 5.4 แบบจำลองกระบวนการผสมสารเคมี 2 ถัง

5.2.1 เงื่อนไขการทำงาน

1. เริ่มต้นการทำงานของแบบจำลองกระบวนการผสมสารเคมี เมื่อกด Switch Start Input 002 วาล์ว SV 100 และ วาล์ว SV 101 จะทำงานเพื่อเติมสารเคมีลงใน TANK 1 และ TANK 2 ตั้งเวลา TIM 00 นาน 20 นาที เมื่อเวลาครบ 20 วินาที วาล์ว SV 100 และ วาล์ว SV 101 จะหยุดการทำงาน

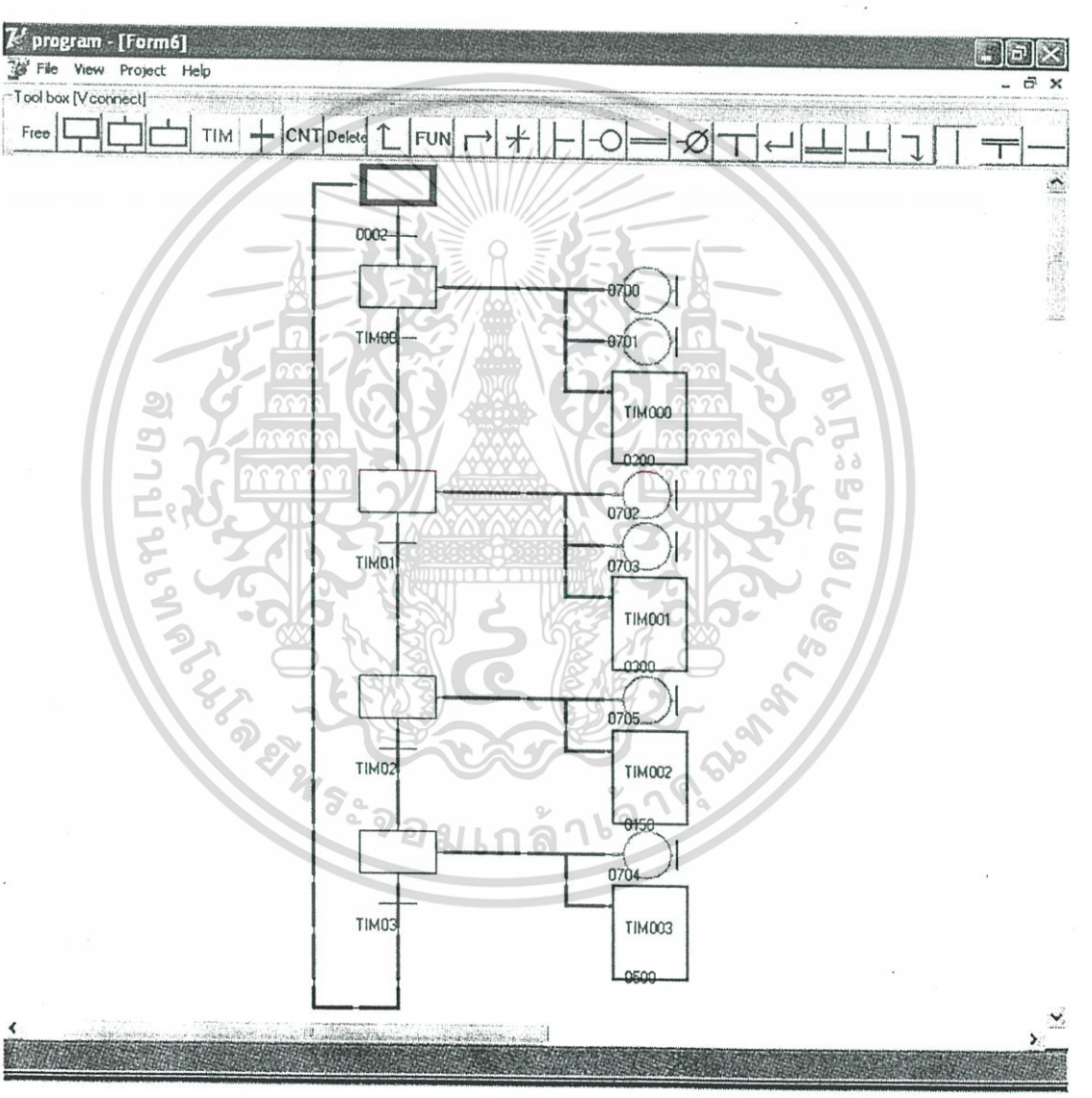
2. หลังจากนั้น วาล์ว SV 102 และ วาล์ว SV 103 จะทำงานเพื่อเติมสารเคมีลงใน MIX TANK ตั้งเวลา TM 01 นาน 30 วินาที เมื่อเวลาครบ 30 วินาที วาล์ว SV 102 และ วาล์ว SV 103 จะหยุดการทำงาน

3. หลังจากนั้น MOTER MIX จะทำงานเพื่อผสมสารเคมีทั้ง 2 ชนิด ตั้งเวลา TIM 02 นาน 15 วินาที เมื่อเวลาครบ 15 วินาที MOTOR MIX จะหยุดการทำงาน

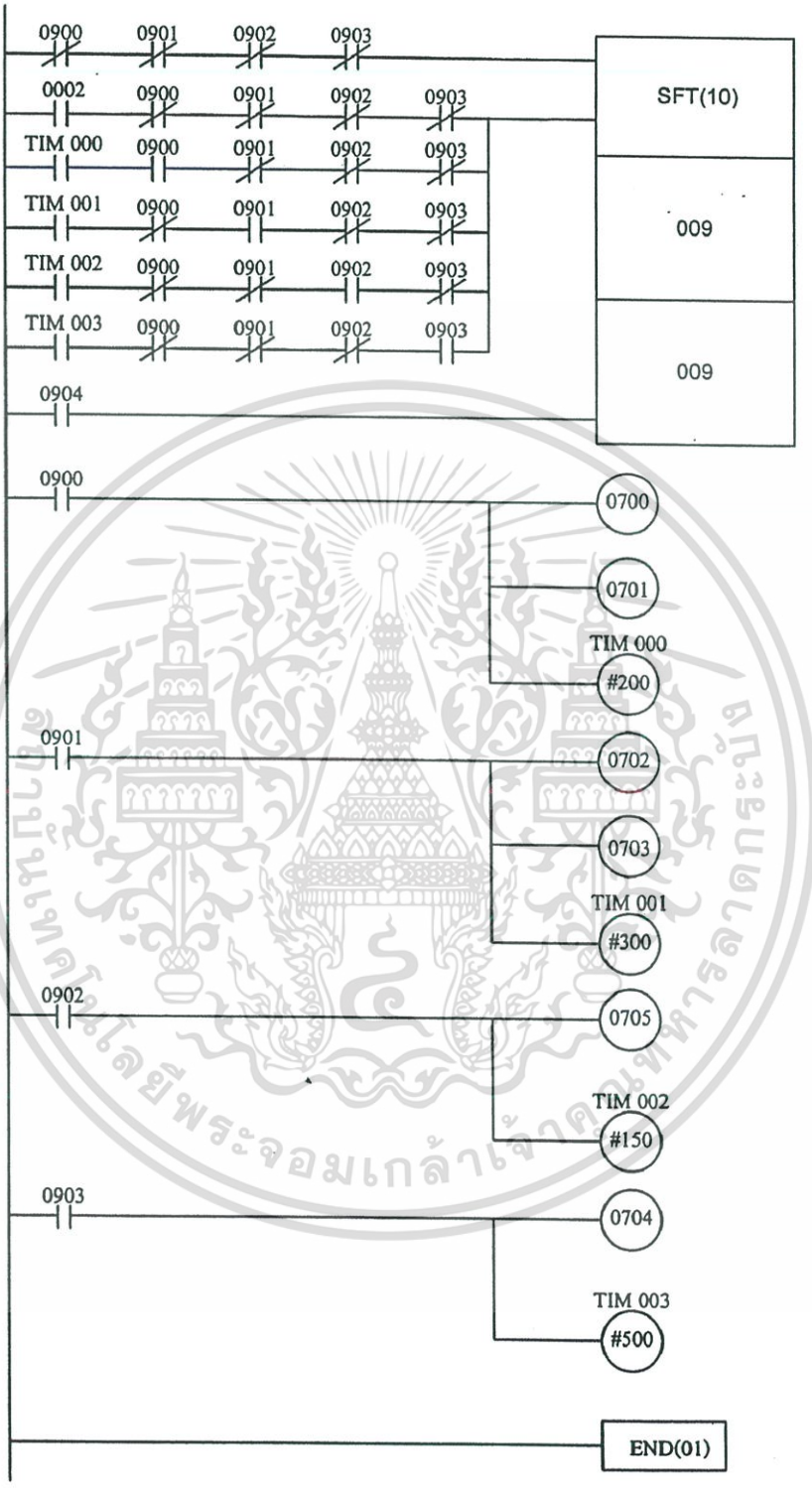
4. หลังจากนั้น วาล์ว SV 104 จะทำงาน ตั้งเวลา TIM 03 นาน 50 วินาที เมื่อเวลาครบ 50 วินาที วาล์ว SV 104 จะหยุดทำงาน

5. เมื่อต้องการเริ่มการทำงานใหม่ต้องกด Switch Start Input 0002

5.2.2 การออกแบบด้วยเพททริเน็ท



รูปที่ 5.5 วงจรควบคุมแบบจำลองกระบวนการผสมสารเคมี 2 ถัง ในรูปแบบของเพททริเน็ท



รูปที่ 5.6 วงจรควบคุมแบบจำลองกระบวนการผสมสารเคมี 2 ถัง ในรูปแบบของวงจรแลดเดอร์

ตารางที่ 5.2 โปรแกรมควบคุมกระบวนการผสมสารเคมี 2 ถัง ในรูปแบบของภาษาบูลีน

Address	Instruction		Operation
0000	LD	NOT	0900
0001	AND	NOT	0901
0002	AND	NOT	0902
0003	AND	NOT	0903
0004	LD		0002
0005	AND	NOT	0900
0006	AND	NOT	0901
0007	AND	NOT	0902
0008	AND	NOT	0903
0009	LD	TIM	0000
0010	AND		0900
0011	AND	NOT	0901
0012	AND	NOT	0902
0013	AND	NOT	0903
0014	ORLD		
0015	LD	TIM	001
0016	AND	NOT	0900
0017	AND		0901
0018	AND	NOT	0902
0019	AND	NOT	0903
0020	ORLD		
0021	LD	TIM	002
0022	AND	NOT	0900
0023	AND	NOT	0901
0024	AND		0902
0025	AND	NOT	0903
0026	ORLD		

ตารางที่ 5.2 (ต่อ)

Address	Instruction		Operation
0027	LD	TIM	003
0028	AND	NOT	0900
0029	AND	NOT	0901
0030	AND	NOT	0902
0031	AND		0903
0032	ORLD		
0033	LD		0904
0034	SET (10) A		009
	DATA B		009
0035	LD		0900
0036	OUT		0700
0037	OUT		0701
0038	TIM		000
	TIM	DATA	#200
0039	LD		0901
0040	OUT		0702
0041	OUT		0703
0042	TIM		001
	TIM	DATA	#300
0043	LD		0902
0044	OUT		0705
0045	TIM		002
	TIM	DATA	#150
0046	LD	(01)	0903
0047	OUT		0704
0048	TIM		003
	TIM		#500
0049	END	(01)	

บทที่ 6
บทวิจารณ์และสรุป

6.1 สรุปผลการทดลอง

นับจาการเริ่มต้นศึกษาทฤษฎีของ Petri Net ตลอดจนพัฒนาโปรแกรมเรื่อยมา จนได้โปรแกรมสำเร็จรูปที่มีความสามารถพอที่จะสรุปได้ดังนี้

1. เขียนเงื่อนไขการทำงานของระบบควบคุมการทำงานต่าง ๆ ให้อยู่ในรูปแบบของ Petri Net โดยที่เงื่อนไขการทำงานดังกล่าว สามารถแปลงให้อยู่ในรูปแบบภาษาบูลีน แล้วนำโปรแกรมดังกล่าวไปป้อนให้ PLC ยี่ห้อต่าง ๆ เพื่อควบคุมเครื่องจักรกลและอุปกรณ์ภายนอกที่ต่อร่วมกับ PLC

2. ขั้นตอนหลังการออกแบบโปรแกรมเป็นที่เรียบร้อยแล้ว เราจะสามารถทดสอบขั้นตอนการทำงานของโปรแกรมก่อนการนำไปใช้งานจริงกับ PLC การตรวจสอบดังกล่าวสามารถป้องกันการเกิด Dead lock ของโปรแกรมได้ ซึ่งเป็นข้อดีของการออกแบบโดยใช้ทฤษฎีของ Petri Net

3. สามารถใช้ฟอร์ม Communication ในการต่อร่วมกับ PLC เพื่อทำการตรวจสอบสถานะและลักษณะการทำงานของโปรแกรมที่อยู่ภายใน อีกทั้ง ยังช่วยให้เห็นผลตอบสนองของ PLC เมื่อต่อร่วมกับอุปกรณ์ภายนอก (หากการทำงานของระบบไม่ถูกต้อง จะช่วยให้รู้ถึงความผิดพลาดของอุปกรณ์นั้น ๆ

4. สามารถเก็บบันทึกโปรแกรมที่เขียนไว้ได้ ดังนั้นการแก้ไขโปรแกรมทำได้โดยง่าย

6.2 แนวทางการพัฒนาต่อ

เนื่องจากความหลากหลายในการใช้งานของโปรแกรม ณ เวลานี้ยังไม่มากนัก ดังนั้นการพัฒนาโปรแกรมเพิ่มเติมเพื่อเพิ่มประสิทธิภาพและความน่าเชื่อถือจึงเป็นสิ่งที่ดีสำหรับผู้ที่สนใจเป็นอย่างยิ่ง

1. โปรแกรมนี้สามารถทำงานในการควบคุมได้ในระดับ Basic Control เท่านั้น ยังไม่สามารถนำเรื่องความน่าจะเป็นของแต่ละ arc เข้ามาเกี่ยวข้องได้ ซึ่งถ้าพัฒนาเกี่ยวกับความน่าจะเป็นของแต่ละ arc ที่ใช้ในการส่งผ่าน Token ได้ก็จะนำไปสู่การควบคุมแบบอนาลอกได้ในที่สุด

2. เมื่อผู้ใช้งานเขียนโปรแกรมและทำการจำลองหรือรันโปรแกรมที่ได้เขียนขึ้นเรียบร้อย สมควรที่จะมี Timing diagram เพื่อใช้ในการตรวจสอบและวิเคราะห์การทำงานของโปรแกรมการทำงานที่ผู้ใช้งานเขียนขึ้นมาได้

3. คำสั่งการใช้ร่วมกับ PLC ยังไม่เพียงพอ เช่น คำสั่ง Counter เป็นต้น จึงควรมีการพัฒนาคำสั่งเหล่านี้ ให้ครอบคลุมการใช้งานจริงมากขึ้น

4. พัฒนาโปรแกรมให้สามารถใช้กับฟังก์ชั่นที่มีความซับซ้อนภายใน PLC

6.3 สรุปผลงานโดยรวม

ตั้งแต่เริ่มทำงานจนถึงปัจจุบันที่ได้สร้างแอพพลิเคชั่นในรูปแบบของโปรแกรมทำให้เราได้ความรู้ความเข้าใจในเรื่องต่าง ๆ ดังนี้

1. ความรู้ในทฤษฎี Petri net โดยได้เรียนรู้จากการจำลองระบบด้วยโมเดลของ Petri net ทฤษฎีการทำงานศึกษาพฤติกรรมของระบบจากโมเดลที่จำลองขึ้นมา รวมถึงแนวทางในการนำ Petri net ไปใช้ในการวิเคราะห์ตรวจสอบระบบการทำงานหรือระบบการผลิตของเครื่องจักรต่าง ๆ

2. ความรู้ในด้านการเขียนโปรแกรมจากโปรแกรม Delphi ทักษะการเขียนโปรแกรมแบบ visual การเขียนโปรแกรมเชิงวัตถุ (Object Oriented Program) ซึ่งมาจากพื้นฐานของภาษาปาสคาล

3. ได้ตระหนักและเรียนรู้ถึงการวางแผนงาน วิเคราะห์ปัญหาก่อนที่จะลงมือทำงานจริง ซึ่งเป็นสิ่งสำคัญมากในการทำงานให้มีประสิทธิภาพ

4. ความรู้ด้านการติดต่อสื่อสารระหว่างเครื่องควบคุมแบบตรรกะที่โปรแกรมได้กับคอมพิวเตอร์

5. ได้ฝึกทักษะในการแก้ไขปัญหาเฉพาะหน้าระหว่างการทำงาน

เอกสารอ้างอิง

1. ก่อกิจ วิระอาชากุล, บุญชัย เอี่ยมเมตตา *"แพททริเน็ท Simulation Tool"* ปริญญานิพนธ์ ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
2. นิพนธ์ เรืองวิริยะนันท์ *"เอกสารประกอบการสอน วิชาโปรแกรมเมเบิ้ลคอนโทรลเลอร์"* แผนกวิชาช่างไฟฟ้ากำลัง วิทยาเขตตาก
3. สัจจะ จรัสรุ่งรวีวร, จักรพงษ์ สุขประเสริฐ *"เริ่มต้นอย่างมืออาชีพด้วย Delphi"* สำนักพิมพ์อินโฟเพรส 200 หมู่ 4 ห้อง 508 อาคารจัสมินอินเตอร์เนชั่นแนลทาวเวอร์ ชั้น 5 ถ.แจ้งวัฒนะ อ.ปากเกร็ด จ.นนทบุรี, 2543
4. ณัฐวุฒิ ศิริธร, วิชิต โวกสูงเนิน และ สิริวิช ประดิษฐ์เทียมผล, *"รายการเขียนโปรแกรมแพททริเน็ทสำหรับ PLC"* ปริญญานิพนธ์ ภาควิชาวิศวกรรมวิศวกรรมวัดคุม คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหารลาดกระบัง
4. Alan J. Crispin, *"Programmable Logic Controller and their Engineering Application"*, Second Edition, the McGraw-Hill Companies. 1997.
5. Alan A. Desrochers, Robert Y. Al-Jaar *"Application of Petri nets in Manufacturing System"* The Institute of Electrical Engineers, Inc, New York Rene David, Hassane Alla "Petri nets and Grafcet" University Press, Cambridge, 1922
6. Rene' David and Hassane Alla, *"Petri Nets and Grafcet-Tools for modeling decrete event systems"*, Prentice Hall 1992.

ภาคผนวก

บทความที่ได้รับการตีพิมพ์

Proceedings of the 2004 IEEE
Conference on Robotics, Automation and Mechatronics
Singapore, 1-3 December, 2004

Interpretation Petri Net model to IEC 1131-3: LD For Programmable Logic Controller

T. Suesut P. Inban P. Nilas P. Rerngreun and S. Gulphanich

Department of Instrumentation Engineering, Faculty of Engineering
King Mongkut's Institute of Technology Ladkrabang, Bangkok 10520, Thailand
(Tel: 66-2739-2407: Ext. 102; E-mail: kstaweep@kmitl.ac.th)

***Abstract*-The Programmable logic Controllers (PLCs) are widely used in industrial. Thus PLCs usually used the IEC 1131-3 standard programming language. The ladder diagram is the most popular language that similar to the electrical schematic diagrams which does not require a lot of the steps and sequences of the operation. Therefore it is difficult to understand the program structure and developing the system. The Petri nets is one of graphical models that can be explained the system operating easily and clearly. This paper presents the transformation structures from Petri Nets to Ladder diagram and possible implementation to control the process through PLC's input and output. In this article can be applied to other applications on any PLCs as well.**

I. INTRODUCTION

Generally, The PLC programming languages have standardized by IEC 1131-3[1]. These are two types of programming structure, textual structure and graphical structure. The Ladder programming is the one of graphical structure that employed in both small-scale and medium scale PLC system and it is the most popular programming language from the past till the present also. The Ladder Diagram (LD) has been invented from relay and logic circuit schematic diagram. Thus the programming algorithm does not have a clearly uniform; it concerns logic conditions, timing diagram and process sequence of the machines. Whereas the LD could not be illustrated the process sequences and the control status clearly. Therefore, the drawbacks of LD are the difficulty to understand and find out or diagnosis fault in the ladder program. Such problems, the PLC programmer must have a programming experience to design the complicate system.

The Petri Nets model is the graphical modeling that has been used for discrete event system. It can be applied to model many systems such as manufacturing system, communications protocol, and also real time controller. This modeling tool can illustrate the step and sequence of discrete process including the performance analysis for the malfunctions condition such as conflict condition, and deadlock condition. In this paper, the advantages of Petri Nets are used to develop computer software to transform Petri Nets to LDs. This computer program was developed in Delphi on Microsoft Windows Operating System.

II. PETRI NET THEORY

Carl A. Petri invented a net-theoretic approach to model and analyzes communication systems [2]. Petri nets have proven to be very useful in the modeling, analysis, Simulation, and control of manufacturing systems. They provide very useful models for the following reasons:

- Petri nets capture the precedence relations and structural interactions of stochastic, concurrent, and asynchronous events. In addition, their graphical nature helps to visualize such complex systems.
- Conflicts and buffer sizes can be modeled easily and efficiently.
- Deadlocks in the system can be detected.
- Petri net models give a structured framework for carrying out a systematic analysis of complex systems. Various software packages have been developed for this purpose.
- Finally, Petri net models can also be used to implement real-time control system for a flexible manufacturing system. Hence, they serve as a replacement for PLC.

The Petri net model can also be used as the basis model for real-time controller in manufacturing systems. The flow of tokens through the net establishes the sequence of event to carry out a specific task, such as the manufacturing of a particular part type. Petri net controllers have been used in factories in Japan and Europe. French research workers have widely contributed to this development. They used this model to describe logic controllers, thereby accounting for the influence on the Grafcet. The Petri net model enables qualitative analyses and its numerous applications have been added to by a number of researchers to enable more condensed descriptions and/or ones where the time factor intervenes: timed, interpreted, stochastic, coloured, continuous, etc Petri nets.

2.1 Basic concepts

2.1.1 Places, transitions and arcs

A Petri net (PN) has two types of node, namely places and transitions A place is represented by a circle and a transition by a bar. Places and transitions are connected by arcs. The number of places is finite and not zero. The number of transitions is also finite and not zero. An arc is directed and connects either a place to a transition or a transition to a place. Fig. 1a represents a PN having seven places, six transitions and fifteen directed arcs. The set of places of a PN will be called P. The set of places is shown as $P=\{P_1,P_2,P_3,P_4,P_5,P_6,P_7\}$ and the set of transitions is $T=\{T_1,T_2,T_3,T_4,T_5,T_6\}$. Place P_3 is said to be upstream or an input of transition T_3 because there is a directed arc from P_3 to T_3. Place P_5 is said to be downstream or an output of transition T_3 because there is a directed arc from T_3 to P_5. In similar way, a transition is said to be an input (upstream) or an output (downstream) of a place. A transition without an input place is a source transition. A transition without an output place is a sink transition.

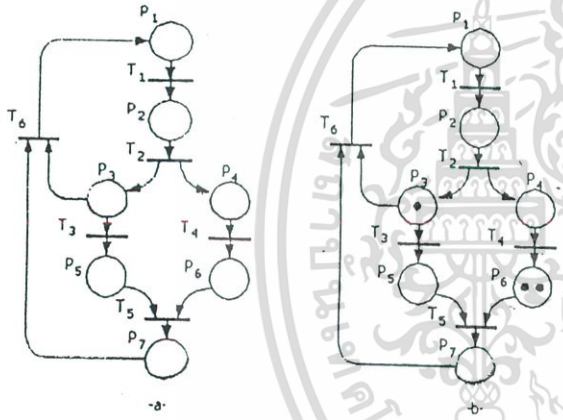


Fig.1 Petri nets. (a) Not marked. (b) Marked.

2.1.2 Marking

Fig. 1b represents a marked Petri net. Each place contains an integer (positive or zero) of tokens or marks. The number of tokens contained in a place P_i will be called either $M(P_i)$ or m_i, The PN marking of Fig.1b is thus $M=(1,0,1,0,0,2,0)$. The marking at a certain moment defines the state of the PN, or more precisely the state of the system described by the PN. The evolution of the state therefore corresponds to an evolution of the marking, caused by firing of transitions.

2.1.3 Firing of a transition

A transition can only fired if each of the input places of this transition contains at least one token. The transition is then said to be fireable, or enabled. A source transition is thus always enabled. When a transition is enabled, this does not imply that is will be immediately fired. This only remains a possibility. In Fig. 1b, two transitions are enabled, T_1 and T_3. Although we do not know when these transitions will be fired, we do know that the next evolution of the marking will correspond either to a firing of T_1 or to a firing of T_3.

2.1.4 Live ness and deadlock

The marking of a PN evolves by the firing of transitions. When certain transitions are no longer fireable and when all or part of the PN no longer 'functions' , there is most probably a problem in the design of the system described. Fig.2a illustrates a quasi-live PN. For each transition, a firing sequence exists which contains it, from the initial indicated marking. M_0 $[T_1>(0,1,0,0)$ and M_0 $[T_2T_3T_4>(0,0,1,0)$. However, there is a conflict $< P_1, \{T_1, T_2\}>$. If the first transition fired is T_1, then the remaining transitions will never be enabled, nor will the first transition fired is T_2, then T_3 will never be enabled. A deadlock is a marking such that no transition is enabled. A deadlock-free PN is shown in Fig.2b. If T1 is the first transition fired, the marking M1= (0,1,0,0,0) is obtained, and transitions T_3 and T_5 are live from this marking. If T_2 is the first transition fired, there is symmetrical behavior. In other words, a deadlock-free PN is such that there will always be a part 'which functions'.

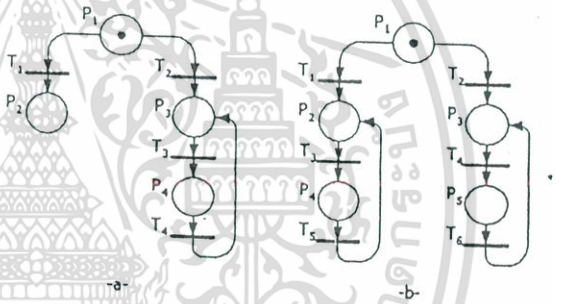
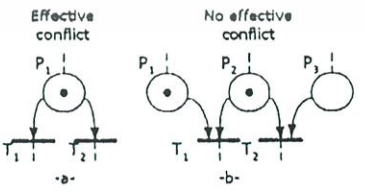


Fig.2 Quasi-live Petri nets. (a) With deadlock, (b) Deadlock-free.

2.1.5 Conflicts

A structural conflict corresponds to a set of at least two transitions T_1 and T_2 which have an input place in common. An effective conflict is the existence of a structural conflict, K, and of a marking, M, such that the number of tokens in P_i is smaller than the number of output transitions of P_i which are enabled by M. An effective and no effect conflict are illustrated in Fig.3



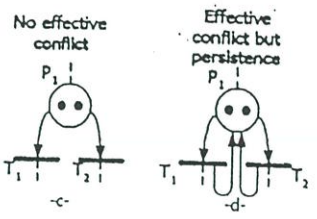


Fig.3 Conflicts

2.2 Grafcet

Grafcet is designed to represent logic controllers, i.e. systems in which the information has an 'all or nothing' character [3]. In this design, Boolean algebra is used. A Boolean variable, for example *a*, can only assume two values, 0 or 1. A Grafcet is a graph having two types of nodes, i.e. steps and transitions. Directed arcs either connect a step to a transition or a transition to a step. The Grafcet structural is similarly as the SFC in IEC 1131-3 as show in Fig.4 .

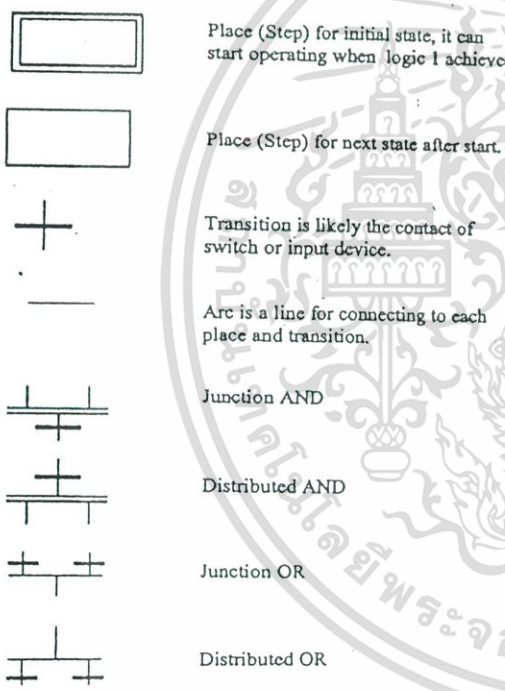


Fig.4. Grafcet

III.TRANSFORMATION ALGORITHMS

In this research, we provided the Boolean equations from the Grafcet which is the one format of Petri net models. The Grafcet is the discrete event model that developed and modified from the traditional Petri net. The Petri net can be applied to general discrete even system, whereas the Grafcet is designed for a logic real-time controller. The proposed research provide the Boolean equation which transformed by Petri net (Grafcet) model. Such these equations can be provided the Ladder Diagram by developed computer software. However, this scheme does not fix rules of transformation algorithms. This method can reduce the confusion to design Ladder Diagram and easily to fault diagnosis also.

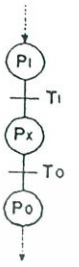


Fig.5 Petri Net composition

The Petri net composition as shows in Fig.5 can be provided the Boolean expression below.

$$P_x=(P_iT_i+P_x)(\overline{P_o}+\overline{T_o}) \quad (1)$$

Where P_x is Place
P_i is Place input
P_o is Place output
T_i is Transition input
T_o is Transition output

According to the token firing of transition can be applied to design Ladder Diagram. Conclusively, any place can be enabled when Place input and Transition input are logic "1". In the other hand, any place will be stopped when Place output and Transition output are logic "1". Fig.6 shows the relation of Place and Transition in each form.

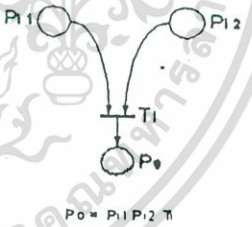


Fig. 6a Junction AND

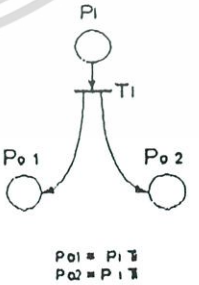
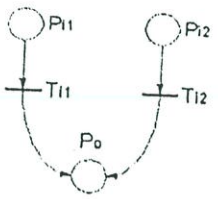
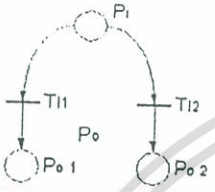


Fig. 6b Distribution AND



$P_O = P_{i1} T_{i1} + P_{i2} T_{i2}$

Fig. 6c Junction OR



$P_{O1} = P_i T_{i1}$
$P_{O2} = P_i T_{i2}$

Fig. 6d Distribution OR

3.1 Test Case 1

This case consists of three places and four transitions.

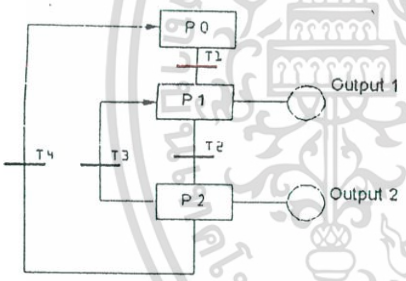


Fig. 7 Test Case 1

The interpreted Ladder Diagram can be provided from the equations as follow:

$$P_0=(P_2T_4+\overline{P_1}\,\overline{P_2}+P_0)(\overline{P_1}+\overline{T_1}) \quad (2)$$

$$P_1=(P_0T_1+\overline{P_2}\,\overline{T_3}+P_1)(\overline{P_2}+\overline{T_2}) \quad (3)$$

$$P_2=(P_1T_2+P_2)\,(P_2+T_3)(P_2+T_4) \quad (4)$$

From the equations number (2),(3),(4), the place and transition can be converted to Ladder diagram as shown in Fig.8.

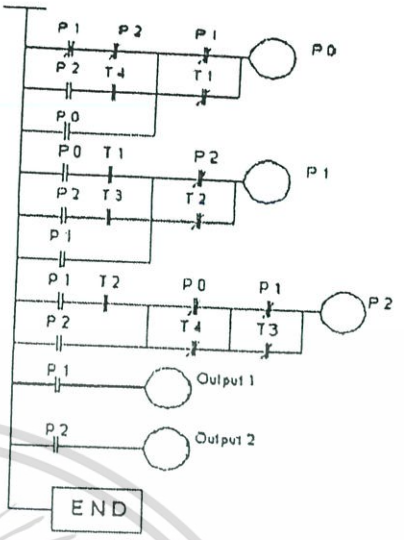


Fig. 8 The result on test Case 1

Such this method, the Ladder Diagram in test case1 can be interpreted from the Petri net in Fig.7. However, the excessive Ladder Diagram is the disadvantage for programming, especially, when implements to the complicate system. In the case of the system structure with certain steps and sequences in operations loop, we can apply the shift register function in such case.

In this article, the Ladder Diagram can be reduced as illustrates in test case 2.

3.2 Test Case 2

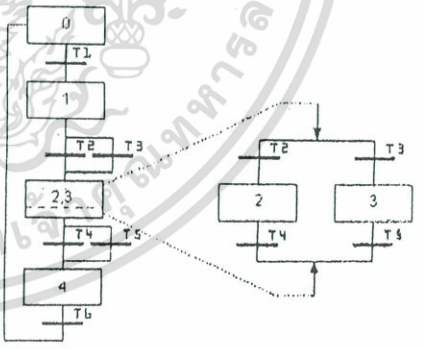


Fig. 9 Test Case 2

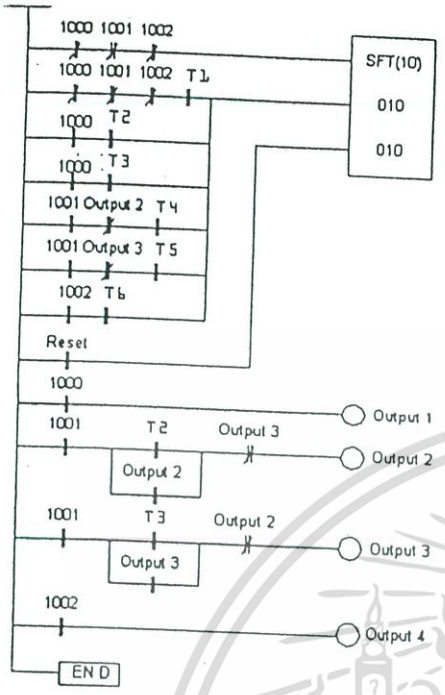


Fig. 11 The result on test Case 2

IV. MPLEMENTATIONS

The transformation algorithm is applied to develop the computer programming based window operating system. The user can design Ladder Program by Petri net graphical editor in this software. The program plat form has shown in Fig 11.

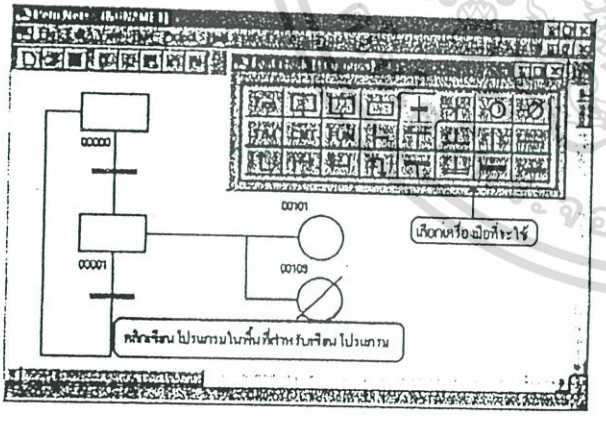


Fig. 11 The Petri net to Ladder software.

V. CONCLUSIONS

The transformation structures from Petri Nets to Ladder diagram in this paper can be implemented to control the process through PLC's input and output. Moreover, the developed software provides checking the conflict and deadlock condition on the designed Petri net. Our scheme, the users can be applied to control applications on any PLC as well.

VII. REFERENCES

[1] Alan J. Crispin, "*Programmable Logic Controller and their Engineering Application*", Second Edition, the McGraw-Hill Companies. 1997.

[2] Alan A. Desrochers, "*Applications of Petri Nets in Manufacturing Systems- Modeling, Control, and Performance Analysis*" , IEEE Press 1995.

[3] Rene' David and Hassane Alla, "*Petri Nets and Grafcet-Tools for modeling decrete event systems*", Prentice Hall 1992.

ประวัติผู้เขียน

นาย ประยุทธ อินแบน บ้านเลขที่ 26/28 หมู่ 1 ถ. สุวินทวงศ์ ต. คลองอุดมชลจร อำเภอเมือง จังหวัดฉะเชิงเทรา 24000

มือถือ (06) 3925387

ประวัติการศึกษา

พ.ศ.2548 สำเร็จการศึกษาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชา วิศวกรรมการวัดคุม คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปัจจุบัน อาจารย์ระดับ 5 คณะเทคโนโลยีอุตสาหกรรม
สาขาวิชาเทคโนโลยีไฟฟ้าอุตสาหกรรม มหาวิทยาลัยราชภัฏราชนครินทร์

งานวิจัยที่ได้รับการตีพิมพ์ต่างประเทศ

Proceedings of the 2004 IEEE
Conference on Robotics, Automation and Mechatronics
Singapore, 1-3 December, 2004