วิธีการโปรแกรมโดยใช้การสาธิตตัวควบคุมและแบบจำลองร่วมกัน สำหรับอุปกรณ์พีแอลซี

JOINT CONTROLLER - SIMULATOR PROGRAMMING BY DEMONSTRATION METHOD FOR PLC DEVICE

อิทธิกร ช่างสากล

ITTHIKORN CHANGSAKON

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต

สาขาวิชาเทคโนโลยีสารสนเทศ

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ.2551

KMITL–2008–IT–M-001-004

JOINT CONTROLLER - SIMULATOR PROGRAMMING BY DEMONSTRATION METHOD FOR PLC DEVICE

ITTHIKORN CHANGSAKON

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

MASTER OF SCIENCE

SCHOOL OF GRADUATE STUDIES

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2008

KMITL–2008–IT–M-001-004

หัวข้อวิทยานิพนธ์	วิธีการโปรแกรมโดยใช้การสาธิตตัวควบคุมและแบบจำลองร่วมกันสำหรับอุปกรณ์พีแอลซี
นักศึกษา	นายอิทธิกร ช่างสากล
รหัสนักศึกษา	46066710
ปริญญา	วิทยาศาสตรมหาบัณฑิต
สาขาวิชา	เทคโนโลยีสารสนเทศ
พ.ศ.	2551
อาจารย์ผู้ควบคุมวิทยานิพนธ์	รศ.ดร.นพพร โชติกกำธร

บทคัดย่อ

วิธีการโปรแกรมอุปกรณ์พีแอลซีที่ใช้ในปัจจุบันต้องอาศัยทักษะการเขียนโปรแกรมคอมพิวเตอร์ วิธีการโปรแกรมโดยใช้การสาธิตเป็นแนวทางหนึ่งที่ได้มีการพัฒนาขึ้นเพื่อให้ผู้ใช้งานคอมพิวเตอร์โดยทั่วไปสามารถทำการโปรแกรมรูปแบบการทำงานซ้ำ ๆ กันเมื่อใช้งานโปรแกรมประยุกต์บางประเภท วิทยานิพนธ์ฉบับนี้นำเสนอวิธีการโปรแกรมอุปกรณ์พีแอลซีโดยอาศัยหลักการโปรแกรมโดยใช้การสาธิต เนื่องจากการประยุกต์ใช้หลักการดังกล่าวในการโปรแกรมอุปกรณ์พีแอลซีโดยตรง จำเป็นต้องอาศัยระบบหรือเครื่องจักรจริงหรือแบบจำลองระบบที่สมบูรณ์ ซึ่งอาจไม่สามารถจัดหาได้ด้วยข้อจำกัดด้านเวลา และค่าใช้จ่าย ในงานวิจัยนี้ได้นำเสนอรูปแบบการโปรแกรมโดยใช้การสาธิตซึ่งส่วนของอุปกรณ์ระบบควบคุมและแบบจำลองการทำงานของระบบที่ถูกควบคุมจะถูกโปรแกรมไปด้วยกันในลักษณะสลับไปมา ทำให้ไม่มีความจำเป็นต้องอาศัยระบบจริงหรือแบบจำลองระบบที่สมบูรณ์แล้วในการประยุกต์ใช้หลักการโปรแกรมดังกล่าว นอกเหนือจากการนำเสนอชุดแบบจำลองอุปกรณ์พื้นฐานเพื่อการสร้างแบบจำลองเริ่มต้นแล้ว งานวิจัยนี้ได้นำเสนอวิธีการในการแปลการสาธิตของผู้ใช้ไปเป็นโปรแกรมในภาษา Structure Text และภาษา Mnemonic เพื่อนำไปควบคุมอุปกรณ์พีแอลซีจริงต่อไป นอกจากนี้ในงานวิจัยนี้ได้กล่าวถึงปัญหาที่เกิดขึ้นจากการโปรแกรมโดยใช้การสาธิตในส่วนที่เกี่ยวข้องกับสถานการณ์ที่ยังไม่พิจารณา และได้นำเสนอแนวทางในการแก้ปัญหาดังกล่าวโดยใช้การจำลองสถานการณ์ในกรณีต่าง ๆ ดังกล่าวให้ผู้ใช้พิจารณา จากวิธีการที่พัฒนาขึ้น เครื่องมือเพื่อการพัฒนาได้ถูกสร้างขึ้นและนำไปใช้ทดสอบประสิทธิภาพในการโปรแกรมเทียบเคียงกับการโปรแกรมด้วยภาษาแลดเดอร์ โดยอาศัยอาสาสมัครที่มีทักษะการเขียนโปรแกรมด้วยภาษาแลดเดอร์มาก่อนแล้ว ผลจากการทดลองพบว่าวิธีการและเครื่องมือที่พัฒนาขึ้นสามารถลดเวลาและความผิดพลาดในการโปรแกรมอุปกรณ์พีแอลซีลงได้อย่างมีนัยสำคัญ

Thesis	Joint Controller-Simulator Programming by Demonstration Method for PLC Device
Student	Mr.Itthikorn Changsakon
Student ID.	46066710
Degree	Master of Science
Program	Information Technology
Year	2008
Thesis Advisor	Assoc. Prof. Dr. Nopporn Chotikakamthorn

ABSTRACT

PLCs programs are generally performed by a specialized programming language. Programming by demonstration (PBD) is an alternative strategy for providing a non-programmer tools and technique. Furthermore, PBD can perform a demonstration on a system simulator instead of applied PLC programming with the real system or complete model that would be more costly and time limitation. In this study focused on applying PBD based on three methods such as Model builder, User action recorder, and Controller programming for controlling devices. Moreover, the researcher generated the PBD in Structure text language and Mnemonic language. Besides the research mentioned in PBD with missing cases and obtained this problem by various simulation environments. Experimental tools were developed and compared with prior Ladder Diagram built by volunteers who exerted in Ladder Diagram. The research found that the developed PBD could reduced both time and errors for controlling PLC devices with statistically significant.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จได้อย่างดี ด้วยคำแนะนำ และคำปรึกษาจาก รศ.ดร.นพพร โชติกกำธร ซึ่งเป็นอาจารย์ผู้ควบคุมวิทยานิพนธ์ ข้าพเจ้ารู้สึกซาบซึ้งในความอนุเคราะห์จากท่านอาจารย์ และขอขอบพระคุณเป็นอย่างสูง

ขอกราบขอบพระคุณคณาจารย์คณะวิทยาการสารสนเทศ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ทุก ๆ ท่านที่ได้ประสิทธิ์ประสาทวิชาให้กับข้าพเจ้า

ขอขอบคุณบัณฑิตศึกษาและบัณฑิตวิทยาลัย คณะวิทยาการสารสนเทศที่ให้ความช่วยเหลือในเรื่องต่างๆ

ขอขอบคุณนางสาววรวลัย โชติพานิช ที่เป็นกำลังใจ และให้การสนับสนุนในทุกเรื่องๆ ทำให้ข้าพเจ้าสามารถทำวิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงด้วยดี

ขอขอบคุณ ดร.สมรวย อภิชาติบุตรพงศ์ และนายโอฬาร คำจีน ที่ให้การสนับสนุนในด้านเอกสารทางวิชาการ

ขอขอบคุณแผนกอิเล็กทรอนิกส์ วิทยาลัยเทคนิคราชบุรี และผู้ที่เข้าร่วมทดสอบโปรแกรมทุกท่าน ที่ให้ความช่วยในการทดสอบโปรแกรม

สุดท้ายนี้ข้าพเจ้าขอกราบขอบพระคุณ บิดา มารดา และครอบครัวของข้าพเจ้าที่เป็นกำลังใจ และให้การสนับสนุนในทุกเรื่องๆ ทำให้ข้าพเจ้าสามารถทำวิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงด้วยดี

คุณค่าและประโยชน์อันพึงมาจากวิทยานิพนธ์ฉบับนี้ ข้าพเจ้าขอมอบแด่ผู้มีพระคุณทุกท่าน

อิทธิกร ช่างสากล

สารบัญ

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

สารบัญรูป

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ปัจจุบันในโรงงานอุตสาหกรรมขนาดเล็กของประเทศไทย การควบคุมกระบวนการทำงานของเครื่องจักรและอุปกรณ์ในโรงงานยังคงมีการใช้อุปกรณ์รีเลย์ ซึ่งจำเป็นจะต้องเดินสายไฟฟ้าหรือที่เรียกว่า Hard-Wired ฉะนั้นเมื่อมีความจำเป็นที่ต้องเปลี่ยนกระบวนการทำงาน หรือลำดับการทำงานใหม่ก็ต้องเดินสายไฟฟ้าใหม่ ซึ่งเสียเวลาและเสียค่าใช้จ่ายสูง การใช้โปรแกรมเมเบิลลอจิกคอนโทรลเลอร์ (Programmable Logic Controller; PLC) สำหรับควบคุมเครื่องจักรหรืออุปกรณ์ต่างๆ ในโรงงานจะมีข้อได้เปรียบกว่าการใช้ระบบรีเลย์ ดังนั้นจึงเหมาะกับงานที่มีการเปลี่ยนแปลง หรือแก้ไขฟังก์ชันการควบคุมอยู่ตลอดเวลา มีประสิทธิภาพในการควบคุมและมีขนาดเล็กกว่า เมื่อเทียบกับการใช้รีเลย์ในการควบคุม ส่วนการดูแลรักษาและการซ่อมบำรุงก็ทำได้ง่าย มีค่าใช้จ่ายต่ำ เมื่อเปรียบเทียบกับการใช้รีเลย์

ปัญหาประการสำคัญของการนำเอาอุปกรณ์พีแอลซีมาใช้ทดแทนอุปกรณ์รีเลย์ คือผู้ที่ปฏิบัติงานที่มีความรู้ความเข้าใจในการควบคุมกระบวนการทำงานของเครื่องจักร และอุปกรณ์ในโรงงานนั้น มักจะไม่มีทักษะทางด้านการโปรแกรม ซึ่งเป็นข้อจำกัดที่สำคัญ โดยผู้ที่จะโปรแกรมอุปกรณ์พีแอลซีนั้นจะต้องมีทักษะและความชำนาญสูงในการใช้เครื่องมือเพื่อเขียนชุดคำสั่งเพื่อควบคุมการทำงานของเครื่องจักร ซึ่งในระดับของ End-User นั้นเป็นเรื่องยากที่จะเรียนรู้และทำความเข้าใจ

วิธีการโปรแกรมโดยใช้การสาธิตเป็นแนวทางหนึ่งที่ได้มีการพัฒนาขึ้นเพื่อให้ผู้ใช้งานคอมพิวเตอร์โดยทั่วไปสามารถทำการโปรแกรมรูปแบบการทำงานซ้ำ ๆ กันเมื่อใช้งานโปรแกรมประยุกต์บางประเภท วิทยานิพนธ์ฉบับนี้นำเสนอวิธีการโปรแกรมอุปกรณ์พีแอลซีโดยอาศัยวิธีการโปรแกรมโดยใช้การสาธิต เนื่องจากการประยุกต์ใช้หลักการดังกล่าวในการโปรแกรมอุปกรณ์พีแอลซีโดยตรง จำเป็นต้องอาศัยระบบหรือเครื่องจักรจริงหรือแบบจำลองระบบที่สมบูรณ์ ซึ่งอาจไม่สามารถจัดหาได้ด้วยข้อจำกัดด้านเวลา และค่าใช้จ่าย

งานวิจัยนี้ได้นำเสนอรูปแบบวิธีการโปรแกรมโดยใช้การสาธิตซึ่งส่วนของอุปกรณ์ระบบควบคุมและแบบจำลองการทำงานของระบบที่ถูกควบคุมจะถูกโปรแกรมไปด้วยกันในลักษณะสลับไปมา ทำให้ไม่มีความจำเป็นต้องอาศัยระบบจริงหรือแบบจำลองระบบที่สมบูรณ์แล้วในการประยุกต์ใช้หลักการโปรแกรมดังกล่าว นอกเหนือจากการนำเสนอชุดแบบจำลองอุปกรณ์พื้นฐาน

เพื่อการสร้างแบบจำลองเริ่มต้นแล้ว งานวิจัยนี้ได้นำเสนอวิธีการในการแปลการสาธิตของผู้ใช้ไปเป็นโปรแกรมในภาษา Structure Text และภาษา Mnemonic

1.2 ความมุ่งหมายและวัตถุประสงค์ของการศึกษา

วิทยานิพนธ์ฉบับนี้มุ่งหวังเพื่อพัฒนาเทคนิควิธีการโปรแกรมโดยใช้การสาธิตสำหรับอุปกรณ์พีแอลซีให้สำหรับผู้ที่ปฏิบัติงานที่มีความรู้ความเข้าใจในการควบคุมกระบวนการทำงานของเครื่องจักร และอุปกรณ์ในโรงงานอุตสาหกรรมจะสามารถโปรแกรมอุปกรณ์พีแอลซีได้โดยไม่ต้องใช้ทักษะของการเขียนโปรแกรมคอมพิวเตอร์ ปัญหาประการสำคัญที่เป็นอุปสรรคในการเขียนโปรแกรมด้วยภาษาตามมาตรฐาน IEC 1131-3 ที่ใช้ในปัจจุบันคือการที่ภาษาตามมาตรฐาน IEC 1131-3 นั้นทำให้ผู้โปรแกรมมองภาพอุปกรณ์ต่างๆ ไม่สอดคล้องกับโครงสร้างของเครื่องจักรจริง ทำให้ผู้ทำการโปรแกรมจำเป็นต้องจดจำภาพของเครื่องจักร และเทียบเคียงกับสัญลักษณ์บนภาษาตามมาตรฐาน IEC 1131-3 ซึ่งอุปกรณ์บนเครื่องจักรจริง 1 ตัว อาจเทียบเคียงกับสัญลักษณ์บนภาษาตามมาตรฐาน IEC 1131-3 มากกว่า 1 ตัว ทำให้เกิดการผิดพลาดในการโปรแกรมได้ง่าย อีกทั้งรูปแบบภาษานั้นยังแทนด้วยสัญลักษณ์ที่ผู้ใช้งานเครื่องจักรยากแก่การทำความเข้าใจ ทำให้ผู้ใช้และผู้พัฒนาโปรแกรมนั้นไม่สามารถทำการโปรแกรมไปพร้อมๆ กันได้ แต่ด้วยวิธีการการโปรแกรมที่พัฒนาขึ้นนั้น ทำให้ผู้ใช้และผู้พัฒนาสามารถทำการโปรแกรมไปพร้อมๆ กันได้ ลดความผิดพลาดเนื่องจากความเข้าใจที่คลาดเคลื่อนระหว่างผู้ใช้และผู้พัฒนา

1.3 สมมุติฐานของการศึกษา

ภาษาสำหรับโปรแกรมอุปกรณ์พีแอลซีที่ใช้ในปัจจุบันภาษาหนึ่งคือภาษาแลดเดอร์ซึ่งต้องอาศัยทักษะการเขียนโปรแกรมคอมพิวเตอร์ และจะต้องอาศัยทักษะความรู้ในวิชาตรรกวิทยาโดยสร้างตารางความจริงของตรรกะก่อนลงมือโปรแกรมด้วยภาษาแลดเดอร์ ทำให้เกิดกรณีทำงานไม่เป็นไปตามจุดประสงค์ของผู้โปรแกรม อันเนื่องจากการเขียนตรรกะที่ผิดพลาดได้ง่าย รวมถึงต้องมีความเข้าใจในรูปแบบภาษาแลดเดอร์ เนื่องจากชุดคำสั่งที่เป็นลักษณะชุดคำสั่งทำให้มีการเขียนคำสั่งที่ไม่เป็นไปตามหลักไวยากรณ์ของภาษาแลดเดอร์ อีกทั้งภาษาแลดเดอร์ซึ่งเป็นการเขียนโปรแกรมเชิงนามธรรมนั้น ผู้ใช้มีปัญหามากในการประเมิน เนื่องจากจะต้องตีความหมายจากสถานะของอุปกรณ์ทั้งหมดเปรียบเทียบและทบทวนกับตารางตรรกะเพื่อที่จะประเมินเทียบกับงานว่าเข้าใกล้เป้าหมายหรือยัง

การแก้ปัญหาข้างต้นนี้ เราจะใช้วิธีการโปรแกรมโดยใช้การสาธิตตัวควบคุมและแบบจำลองร่วมกันสำหรับอุปกรณ์พีแอลซีที่ได้มีการพัฒนาขึ้นเพื่อให้ผู้ใช้งานคอมพิวเตอร์

โดยทั่วไปสามารถทำการโปรแกรมโดยไม่ต้องอาศัยทักษะการเขียนโปรแกรมคอมพิวเตอร์ โดยมีลักษณะการแปลงเป้าหมายไปสู่งานคล้ายกับการทำงานแบบ Manual ซึ่งทำให้ผู้เขียนไม่ต้องอาศัยทักษะตรรกวิทยาหรือรูปแบบการเขียนโปรแกรมแต่อย่างใด ผู้ใช้สามารถประเมินและตีความหมายได้โดยตรง อีกทั้งตัวเครื่องมือยังสามารถตรวจสอบความครบถ้วนของการโปรแกรม ซึ่งอยู่ในส่วนของการจัดการเหตุการณ์ที่ไม่ได้ถูกโปรแกรมทำให้ลดข้อผิดพลาดในการโปรแกรมที่ไม่ครบถ้วนลงได้ซึ่งจะสามารถแก้ปัญหาต่างๆ ดังนี้

1.3.1 การจำลองจะลดความผิดพลาดในการแปลงโจทย์มาเป็นโปรแกรม

1.3.2 การจำลองจะลดการทำงานที่ไม่เป็นไปตามจุดประสงค์ของผู้โปรแกรม (Bug) อันเนื่องจากการเขียนตรรกะที่ผิดพลาด (Logic error)

1.3.3 วิธีการโปรแกรมโดยใช้การสาธิตช่วยลดการผิดพลาดตามกฎไวยากรณ์ (Syntax error)

1.3.4 วิธีการโปรแกรมโดยใช้การสาธิตตัวควบคุมและแบบจำลองร่วมกันจะลดเวลาในการโปรแกรม

1.3.5 วิธีการโปรแกรมโดยใช้การสาธิตตัวควบคุมและแบบจำลองร่วมกัน ทำให้สามารถโปรแกรมครอบคลุมเงื่อนไขการทำงานได้ครบถ้วน

1.3.6 วิธีการโปรแกรมโดยใช้การสาธิตตัวควบคุมและแบบจำลองร่วมกันจะลดความผิดพลาดในการตีความหมายจากผลลัพธ์ของการโปรแกรมได้

1.4 ทฤษฎีหรือแนวคิดที่ใช้ในการวิจัย

รูปแบบการโปรแกรมสำหรับอุปกรณ์พีแอลซี (Programming paradigms for PLC device) นั้น มีหลากหลายวิธี วิธีที่นิยมใช้กันในปัจจุบันมีสองวิธี ได้แก่ รูปแบบของวิธีการเขียนโปรแกรมที่มีพื้นฐานมาจากรายการปฏิบัติงาน (Procedures) วิธีการนี้มีส่วนประกอบที่ใช้ภาษาอังกฤษหรือโปรแกรมเชิงตัวอักษร เช่นเดียวกับภาษาระดับสูง และรูปแบบที่สอง ได้แก่ รูปแบบของวิธีการเขียนโปรแกรมที่เน้นให้มีการใช้ภาพประกอบมากขึ้น เพื่อความสะดวกและสื่อความหมายกับผู้ใช้ได้ดีที่สุด การใช้ภาษาแบบนี้จะใช้ เมาส์ ไอคอน และสัญลักษณ์บนหน้าจอ ซึ่งทั้งสองวิธียังเป็นการนำเอาชุดคำสั่งมาจัดเรียงตามกฎเกณฑ์ และรูปแบบของภาษานั้นๆ โดยลักษณะเด่นของวิธีการที่นำเสนอในวิทยานิพนธ์นี้ เป็นวิธีการที่ใกล้เคียงกับปัญหามากที่สุด คือการจำลองเครื่องจักรขึ้นมาแล้วให้ผู้ใช้ทำการควบคุมเครื่องจักรด้วยมือ (Manual Control) จากนั้นระบบจะทำการเรียนรู้วิธีการควบคุมดังกล่าว เพื่อมาสร้างเป็นชุดคำสั่งและสามารถทำงานได้เหมือนการควบคุมโดยมนุษย์ (Human Control) ซึ่งในวิทยานิพนธ์ฉบับนี้ได้แสดงผลการทดสอบการโปรแกรม เพื่อเปรียบเทียบวิธีการโปรแกรมโดยใช้การสาธิตที่นำเสนอกับวิธีการแบบเดิม

วิธีการโปรแกรมโดยใช้การสาธิต เป็นการพัฒนารูปแบบการโปรแกรมที่มุ่งเน้นเพื่อผู้ที่ไม่มีความชำนาญ (Non-experts) เมื่อเทียบกับหลักการในแบบพื้นฐานแล้ว ในส่วนของการโปรแกรมแล้วทำได้ง่ายกว่าการโปรแกรมแบบอุปกรณ์พีแอลซีตามมาตรฐาน IEC 1131-3 ที่ได้กำหนดไว้ห้าภาษา คือ LD (Ladder Diagram), FBD (Function Block Diagram), IL (Instruction List), ST (Structure Text) และ SFC (Sequential Function Chart)

วิธีการโปรแกรมโดยใช้การสาธิตยังสามารถตรวจสอบลำดับการทำงานของเครื่องจักรโดยการแสดงผ่านทางแบบจำลองได้อีกด้วย ทำให้ช่วยลดข้อผิดพลาดในการโปรแกรมลงประหยัดเวลา และค่าใช้จ่ายในการทดสอบการทำงานของเครื่องจักร

1.5 ขอบเขตการวิจัย

ในวิทยานิพนธ์ฉบับนี้ได้นำเสนอการประยุกต์ใช้วิธีการโปรแกรมโดยใช้การสาธิตตัวควบคุมและแบบจำลองร่วมกัน เพื่อให้การโปรแกรมอุปกรณ์พีแอลซีสามารถกระทำได้ง่ายขึ้น ซึ่งมีขอบเขตการวิจัยดังนี้

1.5.1 เพื่อการประยุกต์ใช้พีแอลซีกับเครื่องจักรเดิมที่ใช้อุปกรณ์รีเลย์

1.5.2 แบบจำลองมีลักษณะการเปลี่ยนแปลงเป็นแบบเชิงเส้น (Linear)

1.5.3 ทำการพัฒนาโปรแกรมในลักษณะต้นแบบ (Prototype) ระบบที่ใช้อาจต้องจำกัดให้เป็นระบบที่เป็น Memory less คือทุกอย่างขึ้นกับสถานะปัจจุบันของระบบเท่านั้น

1.5.4 ผลลัพธ์ของการโปรแกรมด้วยวิธีการนี้จะแปลงสู่ภาษาตามมาตรฐาน IEC 1131-3 ในสองภาษา ภาษาแรกคือภาษา Structure Text และภาษาที่สองคือภาษา Instruction List หรือ Mnemonic

1.6 ขั้นตอนของการศึกษา

ในขั้นตอนของการศึกษานี้ ได้แสดงลำดับการทำงานตั้งแต่เริ่มต้นจนถึงสิ้นสุดการทำงานดังรายละเอียดต่อไปนี้

1.6.1 ศึกษาทฤษฎีและงานวิจัยจากเอกสารบทความต่าง ๆ ที่เกี่ยวข้องกับการทำงานวิจัย

1.6.2 กำหนดหัวข้อ วัตถุประสงค์ และขอบเขตการทำงานวิจัย

1.6.3 วิเคราะห์ข้อมูลที่ได้ศึกษารวบรวม และออกแบบเทคนิคการเขียนโปรแกรมใหม่

1.6.4 ทำการพัฒนาโปรแกรม โดยใช้ซอฟท์แวร์ Borland C++ Builder

1.6.5 เตรียมข้อมูลและแบบจำลองเครื่องจักรเพื่อนำมาทดสอบเทคนิคการเขียนโปรแกรมด้วยวิธีการที่ได้นำเสนอนี้

1.6.6 ทดลองกับกลุ่มตัวอย่าง เพื่อวัดประสิทธิภาพของการเขียนโปรแกรมด้วยวิธีการที่ได้นำเสนอนี้ เมื่อเทียบกับวิธีการโปรแกรมด้วยภาษาแลดเดอร์

1.6.7 รวบรวมผลการทดลองจากการสังเกตการทดลองกับกลุ่มตัวอย่าง

1.6.8 วิเคราะห์และสรุปผลการทดลอง

1.6.9 เรียบเรียงเอกสารประกอบวิทยานิพนธ์

บทที่ 2
การโปรแกรมอุปกรณ์พีแอลซีและรูปแบบการสร้างโปรแกรมคอมพิวเตอร์

2.1 อุปกรณ์พีแอลซี (Programmable Logic Control)

พีแอลซีเป็นอุปกรณ์ชนิดโซลิด-สเตท (Solid State) ที่ทำงานแบบลอจิก (Logic Functions) การออกแบบการทำงานของพีแอลซีคล้ายกับหลักการทำงานของคอมพิวเตอร์ทั่วไป หลักการพื้นฐานแล้วพีแอลซีจะประกอบด้วยอุปกรณ์ที่เรียกว่า Solid-State Digital Logic Elements เพื่อให้ทำงานและตัดสินใจแบบลอจิก ใช้สำหรับควบคุมกระบวนการทำงานของเครื่องจักรและอุปกรณ์ในโรงงาน ซึ่งมีลักษณะการทำงานเป็นแบบลอจิก (Logic control system) หรือแบบซีเควนซ์ (Sequence control system) เท่านั้น โดยมีเซนเซอร์และอุปกรณ์ทำงาน (Actuator) ที่ควบคุมการทำงานภายในเครื่องจักร

2.1.1 ความหมายของโปรแกรมเมเบิลลอจิกคอนโทรลเลอร์

ความหมายของโปรแกรมเมเบิลลอจิกคอนโทรลเลอร์ตามมาตรฐานของ IEC 1131 Part 1 หมายถึง ระบบปฏิบัติการทางด้านดิจิตอล ที่ออกแบบมาให้ใช้งานในอุตสาหกรรม ซึ่งใช้หน่วยความจำที่สามารถโปรแกรมได้ในการเก็บคำสั่งที่ผู้ใช้กำหนดขึ้น (User Program) เพื่อเป็นเครื่องมือในการกำหนดฟังก์ชันหรือเงื่อนไขในการทำงานได้ [19, 20]

2.1.2 ประวัติของโปรแกรมเมเบิลลอจิกคอนโทรลเลอร์

ในระยะแรกได้มีการพัฒนาโดยนำเอาอุปกรณ์อิเล็กทรอนิกส์เข้ามาใช้ในการสร้างโปรแกรมเมเบิลลอจิกคอนโทรลเลอร์ หลังจากนั้นในปี ค.ศ.1970 จึงได้มีการพัฒนาโดยนำเอาไมโครโปรเซสเซอร์มาใช้ในการประมวลผล ทำให้โปรแกรมเมเบิลลอจิกคอนโทรลเลอร์มีความสามารถและขอบเขตการใช้งานมากขึ้น เช่น การประมวลผลฟังก์ชันทางคณิตศาสตร์ ซึ่งทำให้โปรแกรมเมเบิลลอจิกคอนโทรลเลอร์สามารถที่จะทำการควบคุมอุปกรณ์ที่มีลักษณะเป็นสัญญาณอนาล็อก (Analog Signal) และสามารถทำการสื่อสารกับระบบคอมพิวเตอร์ต่างๆ ได้ และจากการพัฒนาโปรแกรมเมเบิลลอจิกคอนโทรลเลอร์อย่างต่อเนื่อง ตั้งแต่ปี ค.ศ.1975 ได้มีการนำเทคโนโลยีคอมพิวเตอร์ทั้งทางด้านฮาร์ดแวร์และซอฟท์แวร์มาใช้กับโปรแกรมเมเบิลลอจิกคอนโทรลเลอร์ ทำให้ความสามารถในการทำงานเพิ่มสูงขึ้น เช่น มีหน่วยความจำเพิ่มขึ้น สามารถติดต่อกับอินพุท และเอาท์พุทแบบระยะไกล (Remote input/output) สามารถใช้หน่วยประมวลผลจำนวนหลายตัว (Multi-processor) ร่วมกันประมวลผลโปรแกรม สามารถทำการควบคุมโดยใช้โมดูลแบบพิเศษ

(Intelligent module) และนอกจากนั้นในปัจจุบันโปรแกรมเมเบิลลอจิกคอนโทรลเลอร์ยังสามารถทำการติดต่อสื่อสารข้อมูลเป็นโครงข่ายผ่าน Ethernet Protocol, Profibus และ ASI-bus เป็นต้น

ดังนั้นจะเห็นว่า เราสามารถที่จะนำข้อมูลจากกระบวนการผลิตมาใช้ในการตัดสินใจ และสามารถที่จะควบคุมการผลิตตามแผนการที่กำหนดโดยผู้บริหารได้อย่างรวดเร็วผ่านการสื่อสารแบบต่างๆ และนอกจากนั้นยังทำให้สามารถที่จะติดต่อสื่อสารระหว่างโปรแกรมเมเบิลลอจิกคอนโทรลเลอร์ และอุปกรณ์หรือเครื่องมือที่ผลิตมาจากบริษัทต่างๆ กันได้

โปรแกรมเมเบิลลอจิกคอนโทรลเลอร์หรือชื่อที่ใช้เรียกขานทับศัพท์กันในเชิงพาณิชย์ทั่วๆไปว่า พีแอลซี (PLC, Programmable Logic Controller) ถูกสร้างขึ้นมาครั้งแรกในปี ค.ศ.1968 โดยกลุ่มวิศวกร Hydromantic division ของบริษัท General Motors Corporations เนื่องจากมีความต้องการที่จะสร้างอุปกรณ์ควบคุมมาทดแทนการใช้รีเลย์ในการควบคุมสำหรับโรงงานประกอบรถยนต์ ซึ่งจะต้องสามารถรองรับการประกอบรถยนต์รุ่นใหม่ๆ ได้ตลอดเวลา

2.1.3 คุณสมบัติของโปรแกรมเมเบิลลอจิกคอนโทรลเลอร์

พีแอลซีจะมีข้อได้เปรียบกว่าการใช้ระบบรีเลย์ ดังนั้นจึงเหมาะกับงานที่มีการเปลี่ยนแปลง หรือแก้ไขฟังก์ชันการควบคุมอยู่ตลอดเวลา มีประสิทธิภาพในการควบคุม และมีขนาดเล็กกว่า เมื่อเทียบกับการใช้รีเลย์ในการควบคุม ส่วนการดูแลรักษาและการซ่อมบำรุงก็ทำได้ง่าย และค่าใช้จ่ายต่ำ เมื่อเปรียบเทียบกับการใช้รีเลย์ ดังได้แสดงการเปรียบเทียบความแตกต่างระหว่างการใช้งานพีแอลซีและระบบรีเลย์ในตารางที่ 2.1

ตารางที่ 2.1 การเปรียบเทียบระหว่างการใช้งานพีแอลซีและระบบรีเลย์ในด้านต่างๆ

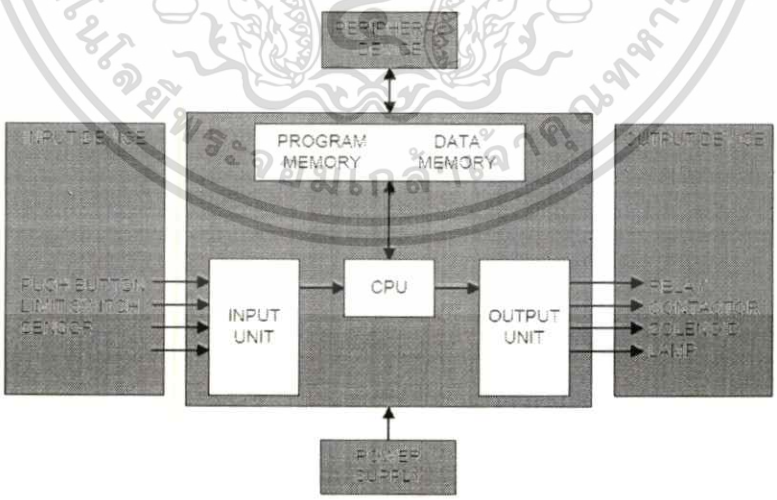
คุณลักษณะ	PLC	Relays
1. ราคาค่าใช้จ่าย(ต่อการทำงานที่มีการใช้รีเลย์มากกว่า 20 ตัวขึ้นไป)	ต่ำกว่า	สูงกว่า
2. ขนาดเมื่อทำการติดตั้ง	กะทัดรัด	มีขนาดใหญ่กว่า
3. ความเร็วในการปฏิบัติการ	มีความเร็วสูงกว่า	ช้ากว่า
4. ความทนทานต่อการรบกวนของสัญญาณไฟฟ้า	ดี	ดีมาก
5. การติดตั้ง	ง่ายในการติดตั้งและโปรแกรม	ใช้เวลามากกว่าในการออกแบบและติดตั้ง

ตารางที่ 2.1 เปรียบเทียบระหว่างการใช้งานพีแอลซีและระบบรีเลย์ (ต่อ)

คุณลักษณะ	PLC	Relays
6. ความสามารถในการปฏิบัติการฟังก์ชันที่ซับซ้อน	สามารถกระทำได้	ไม่สามารถกระทำได้
7. ความสามารถในการเปลี่ยนแปลงลำดับการควบคุม	สามารถกระทำได้ง่าย	สามารถกระทำได้แต่ค่อนข้างยุ่งยาก
8. การซ่อมบำรุง และตรวจสอบแก้ไข	ไม่ต้องการการบำรุงรักษามากและง่ายในการตรวจสอบแก้ไข ในกรณีที่เกิดปัญหาภายในระบบควบคุม	ต้องการการดูแลในส่วนของคอยล์และหน้าสัมผัส และยากในการตรวจสอบและ แก้ไขในกรณีที่เกิดปัญหา

2.1.4 องค์ประกอบของโปรแกรมเมเบิลลอจิกคอนโทรลเลอร์

โปรแกรมเมเบิลลอจิกคอนโทรลเลอร์ มีซีพียูเป็นหน่วยควบคุมกลางที่สำคัญ ซึ่งจะคอยรับข้อมูลจากอินพุทส่งเก็บในหน่วยความจำ หลังจากประมวลผลแล้วจะส่งออกสู่หน่วยเอาท์พุท เพื่อเป็นสัญญาณควบคุมอุปกรณ์ภายนอกอีกทีหนึ่ง โดยมีองค์ประกอบดังรูปที่ 2.1



รูปที่ 2.1 ส่วนประกอบต่างๆ ของอุปกรณ์พีแอลซี

หน้าที่ของหน่วยต่างๆ ภายใน PLC ทำหน้าที่ดังนี้

1) ภาคอินพุท รับสัญญาณจากสวิทช์ต่างๆ และตัวตรวจวัด แปลงชนิดของสัญญาณขาเข้าต่างๆ เช่น AC DC ให้เป็นสัญญาณที่เหมาะสมเพื่อป้อนเข้าไปให้แก่ซีพียู หน่วยนี้เปรียบเหมือนโสตประสาททั้งห้าของคนที่คอยรับข้อมูลจากรอบตัวแล้วกลั่นกรองเพื่อป้อนเข้าสู่สมอง

2) ภาคเอาท์พุท รับข้อมูลจากซีพียูแล้วแปลงเป็นสัญญาณที่มีขนาดใหญ่พอที่จะไปขับอุปกรณ์ภายนอก โดยสัญญาณออกอาจเป็นหน้าสัมผัส ทรานซิสเตอร์ หรือ TRIAC ก็ได้แล้วแต่ชนิด หน่วยนี้ทำงานเหมือนกับมือ เท้า ปากของคนที่รับคำสั่งจากสมองแล้วทำงานตาม

3) ซีพียูเป็นหน่วยควบคุมกลางที่ทำงานประสานกันทุกหน่วย เช่น เริ่มจากอ่านข้อมูลจากอินพุทนำมาเก็บไว้ในหน่วยความจำ นำข้อมูลในหน่วยความจำมาประมวลผล ได้ผลอย่างไรก็ส่งออกไปที่เอาท์พุท การทำงานของซีพียูจะกำหนดโดยโปรแกรมในหน่วยความจำที่เราป้อนเข้าหน่วยนี้ทำงานคล้ายกับสมองแต่สมองนี้จะทำงานตามคำสั่งที่เตรียมไว้เท่านั้น ไม่สามารถทำงานอิสระได้

4) Data memory หน่วยความจำเก็บข้อมูล เป็นบริเวณของหน่วยความจำที่ใช้เก็บสภาวะของอุปกรณ์ภายใน เช่น สภาวะของรีเลย์ทุกตัวว่าทำงานหรือไม่ทำงานอย่างไร เก็บสภาวะของตั้งเวลา ตัวนับทุกตัว นอกจากนั้นยังเก็บสภาวะของสายอินพุท และเอาท์พุทอีกด้วย

5) Program memory หน่วยความจำเก็บโปรแกรม เป็นบริเวณของหน่วยความจำที่ใช้เก็บโปรแกรมที่เป็นวงจร โปรแกรมนี้ผู้ใช้เป็นผู้ป้อนเข้าไป โดยต้องเข้าใจวิธีการแปลงรูปวงจรรีเลย์เป็นโปรแกรมเท่านั้น ซีพียูจะมาอ่านโปรแกรมในหน่วยความจำนี้ทีละคำสั่งและทำงานตามทันที โดยทำงานทุกคำสั่งจนจบโปรแกรมแล้วเริ่มต้นใหม่วนเวียนเช่นนี้ตลอด หน่วยความจำส่วนนี้ยิ่งมีมากเท่าไรก็สามารถบรรจุวงจรซีเควนซ์ที่ใหญ่และซับซ้อนขึ้นได้

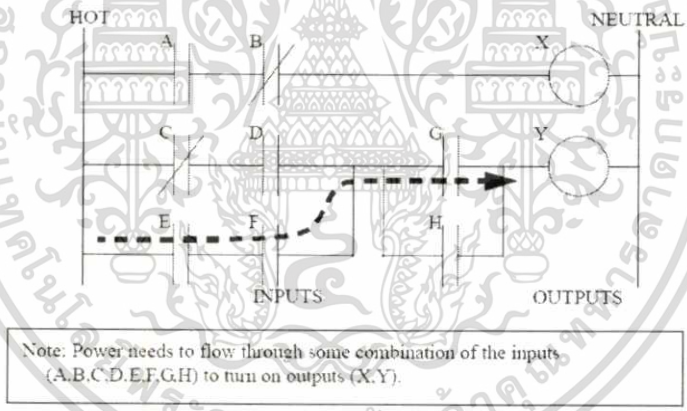
6) Peripheral device อุปกรณ์รายรอบ เป็นเครื่องมือหรืออุปกรณ์เสริมระบบ ทำให้การใช้งานพีแอลซีสะดวกขึ้น เช่น เครื่องป้อนโปรแกรม ทำหน้าที่ตรวจเช็ค และแก้ไขโปรแกรมในหน่วยความจำหรือคัดลอกโปรแกรม และทำการพิมพ์ลงในเครื่องพิมพ์ นอกจากนี้อาจต่อพีแอลซีเข้ากับคอมพิวเตอร์ เพื่อให้สามารถส่งถ่ายข้อมูลและโปรแกรมได้ อุปกรณ์รายรอบพวกนี้มักใช้ในขั้นพัฒนาโปรแกรมสำหรับการควบคุม เมื่อระบบทำงานถูกต้องแล้วอุปกรณ์เหล่านี้จะไม่ใช้ต่อกับพีแอลซีตลอดเวลา จะเหลือเพียงพีแอลซีควบคุมเครื่องจักรเท่านั้น

2.2 ภาษาที่ใช้ในการโปรแกรมอุปกรณ์พีแอลซี ตามมาตรฐาน IEC 1131 Part 3

ภาษาที่ใช้ในการเขียนโปรแกรมตามมาตรฐาน IEC 1131-3 กำหนดไว้ 5 ภาษา คือ LD (Ladder Diagram), FBD (Function Block Diagram), IL(Instruction List), ST(Structure Text) และ SFC(Sequential Function Chart) ถึงแม้ว่าลักษณะโครงสร้างของแต่ละภาษาจะมีความแตกต่างกัน แต่ในแต่ละภาษาจะมีส่วนประกอบต่างๆ ในโปรแกรมที่มีลักษณะเดียวกันตามมาตรฐาน IEC 1131-3 เช่น ลักษณะการประกาศตัวแปร ฟังก์ชัน และฟังก์ชันบล็อก เป็นต้น แต่อย่างไรก็ตาม เราสามารถที่จะเขียนโปรแกรมโดยนำรูปแบบการเขียนในภาษาต่างๆ มารวมกันได้

2.2.1 LD (Ladder Diagram)

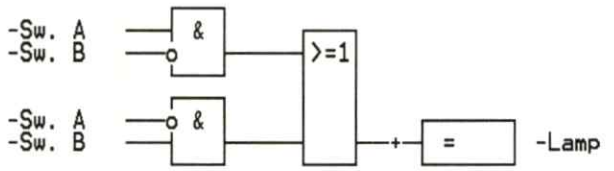
เป็นภาษาที่เขียนอยู่ในรูปของกราฟิก ซึ่งมีพื้นฐานมาจากวงจรควบคุมแบบรีเลย์ และวงจรไฟฟ้า ซึ่ง Ladder Diagram จะประกอบด้วยราง (Rail) ทั้งซ้ายและขวาของไดอะแกรม เพื่อใช้สำหรับเชื่อมต่ออุปกรณ์ที่เป็นสวิตช์หน้าสัมผัส เพื่อเป็นทางผ่านของกระแส และมีขดลวดหรือคอยล์เป็นเอาท์พุท



รูปที่ 2.2 แสดงตัวอย่างภาษา Ladder Diagram

2.2.2 FBD (Function Block Diagram)

เป็นภาษาที่แสดงฟังก์ชันการทำงาน ในรูปของ กราฟิกเช่นเดียวกัน และเชื่อมต่อกันเป็นโครงข่าย โดยการเขียนโปรแกรมในรูปของฟังก์ชันบล็อกไดอะแกรมจะมีพื้นฐานมาจากลอจิกไดอะแกรม



รูปที่ 2.3 แสดงตัวอย่างภาษา Function Block Diagram

2.2.3 IL (Instruction List)

เป็นภาษาที่เขียนอยู่ในรูปของข้อความ และมีลักษณะคล้ายกับภาษาแอสเซมบลี (Assembly) และภาษาเครื่อง (Machine code) ซึ่งภายในหนึ่งคำสั่งควบคุมจะประกอบด้วย ส่วนปฏิบัติการ (Operator) และส่วนที่ถูกดำเนินการ (Operand)

```
00000    LDN     00001
00001    LD      00002
00002    AND
00003    LD      00003
00004    LD      00004
00005    AND
00006    OR
00007    ST      00107
00008    END
```

รูปที่ 2.4 แสดงตัวอย่างภาษา Instruction List

2.2.4 ST (Structure Text)

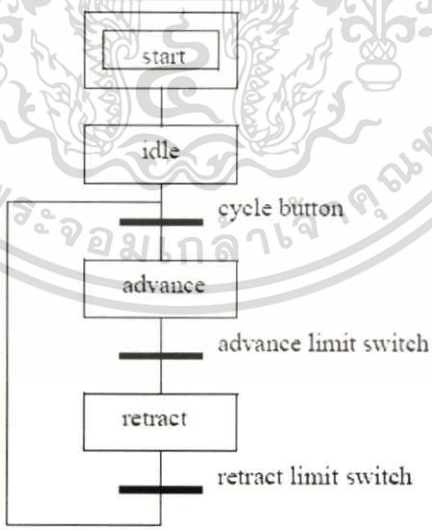
เป็นภาษาในระดับสูง โดยมีพื้นฐานมาจากภาษา Pascal ซึ่งจะประกอบไปด้วย นิพจน์ และคำสั่ง โดยคำสั่งทั่วไปจะอยู่ในรูปของคำสั่งเกี่ยวกับการเลือกทำงาน เช่น IF……..THEN……..ELSE เป็นต้น คำสั่งเกี่ยวกับการทำงานซ้ำ เช่น FOR , WHILE เป็นต้น

```
PROGRAM example7.1
     VAR_INPUT
          MSI   :     BOOL;
          C1    :     BOOL;
          C2    :     BOOL;
          C3    :     BOOL;
          C4    :     BOOL;
     END_VAR
     VAR_OUTPUT
          R1    :     BOOL : FALSE;
          R2    :     BOOL : FALSE;
          R3    :     BOOL : FALSE;
          R4    :     BOOL : FALSE;
     END_VAR
     R1 := MS1 AND (NOT R4);
     R2 := R4 AND (NOT C3) AND (NOT C2);
     R3 := C4 AND (NOT C3);
     R4 := C1;
END_PROGRAM
```

รูปที่ 2.5 แสดงตัวอย่างภาษา Structure Text

2.2.5 SFC (Sequential Function Chart)

เป็นภาษาที่รองรับการเขียนโปรแกรมที่มีโครงสร้างการทำงานเป็นแบบซีเควนซ์ ซึ่งส่วนประกอบของ SFC จะประกอบด้วยขั้นตอน (คำสั่งในการปฏิบัติการในแต่ละขั้นตอน) และ Transition (เงื่อนไขที่กำหนดให้กระทำคำสั่งในแต่ละขั้นตอน) นอกจากนี้ยังสามารถกำหนดลักษณะการทำงาน เช่น Alternative step sequence และ Parallel step sequence เป็นต้น



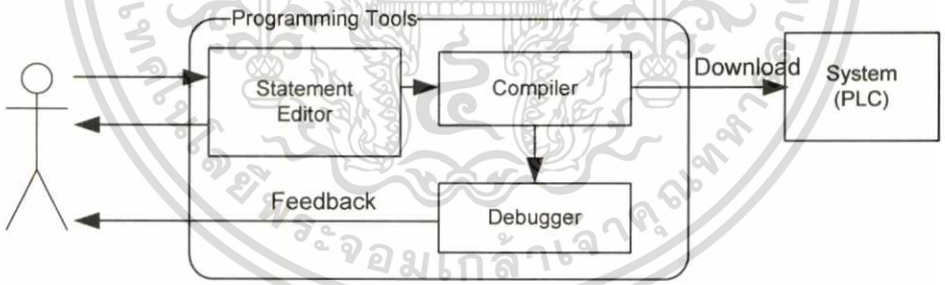
รูปที่ 2.6 แสดงตัวอย่างภาษา Sequential Function Chart

2.3 รูปแบบการโปรแกรมอุปกรณ์พีแอลซี (Programming paradigms for PLC device)

Programming paradigms คือ รูปแบบและสภาพแวดล้อมของการเขียนโปรแกรมมีการจัดเตรียมและกำหนดมุมมองของโปรแกรมเมอร์ที่มีต่อการทำงาน โดยถูกกำหนดจากผู้ผลิตเครื่องมือในการเขียนโปรแกรม หรือองค์กรผู้ทำหน้าที่ดูแลในเรื่องนั้นๆ เช่น IEEE, ISO, UNDP, W3C ฯลฯ ซึ่งรูปแบบและสภาพแวดล้อมของการเขียนโปรแกรมอุปกรณ์พีแอลซีนั้น โดยทั่วไปมีดังต่อไปนี้

2.3.1 รูปแบบการเขียนโปรแกรมแบบรายการปฏิบัติงาน (Conventional programming paradigm)

คือรูปแบบของวิธีการเขียนโปรแกรมที่มีพื้นฐานมาจากรายการปฏิบัติงาน (Procedures) หรืองานประจำและงานย่อยที่ต้องทำซ้ำๆ วิธีการนี้มีส่วนประกอบที่ใช้ภาษาอังกฤษหรือโปรแกรมเชิงตัวอักษร เช่นเดียวกับภาษาระดับสูง แล้วผ่านขบวนการแปลภาษาเพื่อได้เป็นภาษาเครื่องที่สามารถทำงานได้ ภาษาที่ใช้เขียนชุดคำสั่ง เช่น ภาษาที่ใช้ในการโปรแกรมอุปกรณ์ พีแอลซีภายใต้รูปแบบของวิธีการเขียนโปรแกรมนี้ คือ IL (Instruction List) หรือภาษา Mnemonic และ ST (Structure Text) ซึ่งแสดงแผนผังของการโปรแกรมดังรูปที่ 2.7

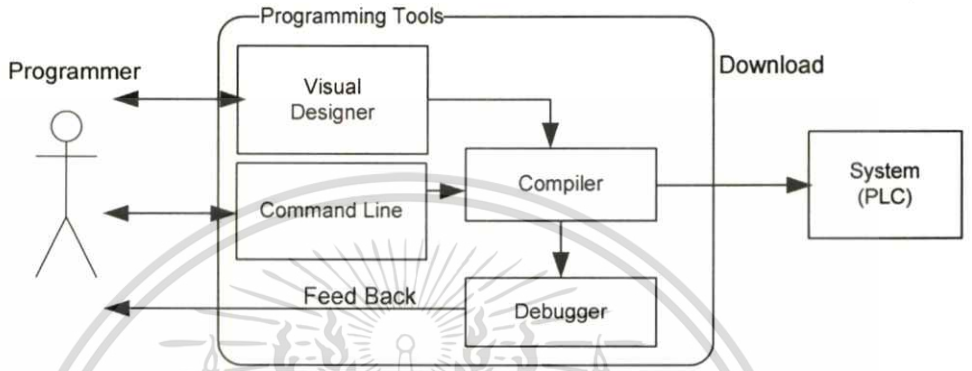


รูปที่ 2.7 แสดงแผนผังของ Conventional programming paradigm

2.3.2 รูปแบบการโปรแกรมที่เน้นให้มีการใช้ภาพประกอบ (Visual programming paradigm)

คือรูปแบบของวิธีการเขียนโปรแกรมที่เน้นให้มีการใช้ภาพประกอบมากขึ้น เพื่อความสะดวก และสื่อความหมายกับผู้ใช้ได้ดีที่สุด การใช้ภาษาแบบนี้จะใช้เมาส์ ไอคอน และสัญลักษณ์บนหน้าจอหรือเมนู โดยการ point-and-click และ drag-and-drop ภาพไอคอนต่างๆ ที่ปรากฏให้เห็นบนหน้าจอนั้นเป็นสัญลักษณ์แทนไฟล์หรือชุดคำสั่งของระบบปฏิบัติการคอมพิวเตอร์ ซึ่งจะถูก

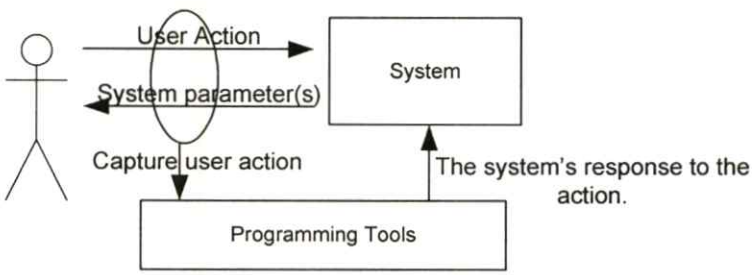
เลือกหรือโยกย้ายไปมาโดยการคลิกเมาส์ลงและกดปุ่มของเมาส์ค้างไว้เพื่อ "จับ" จากนั้นก็ "ลาก" เมาส์เพื่อให้ไอคอนเคลื่อนตามไปจนถึงตำแหน่งหนึ่งตำแหน่งใดบนหน้าจอที่ต้องการแล้ว "ปล่อย" ลงไว้ ณ ที่นั้นด้วยการเลิกกดปุ่มค้างไว้ ทั้งนี้ในส่วนของ GUI จะมีคำสั่งที่ซับซ้อนกว่าการเขียนโปรแกรมในยุคต้น ภาษาที่ใช้สร้างชุดคำสั่ง เช่น Ladder Diagram, Function Block Diagram และ Sequential Function Chart ซึ่งแสดงแผนผังของการโปรแกรมดังรูปที่ 2.8



รูปที่ 2.8 แสดงแผนผังของ Visual programming paradigm

2.3.3 รูปแบบวิธีการโปรแกรมโดยใช้การสาธิต (Programming by Demonstration)

การเขียนโปรแกรมโดยใช้การสาธิตเป็นรูปแบบที่นิยมใช้กับระดับของ End-User โดยที่ Programming Tools จะมีหน้าที่คอยตรวจจับการกระทำของผู้ใช้ (User Action) ที่กระทำต่อระบบ และทำการบันทึกเป็นชุดคำสั่ง ทำให้ Programming Tools ทำงานซ้ำโดยกระทำต่อระบบได้เหมือนกับผู้ใช้ตามสิ่งที่ได้บันทึกไว้ [11] ข้อจำกัดของ Programming by Demonstration คือการโปรแกรมจะเป็นการสาธิตกับตัวระบบที่เสร็จสมบูรณ์แล้ว (Complete System) เท่านั้น คือระบบนั้นจะต้องมีชุดคำสั่งที่ตายตัวอยู่แล้ว ผู้ใช้เพียงจัดลำดับของคำสั่งการทำงานขึ้นมาใหม่เท่านั้น ดังแสดงในรูปที่ 2.9



รูปที่ 2.9 แสดงแผนผังของวิธีการโปรแกรมโดยใช้การสาธิต

2.4 เทคนิคการโปรแกรมด้วยภาพ (Visual Programming Technique)

การโปรแกรมด้วยภาพ (Visual Programming) เป็นการพัฒนารูปแบบในการเขียนชุดคำสั่งให้อยู่ในรูปแบบกราฟิก โดยจะใช้เมาส์ ไอคอน และสัญลักษณ์บนหน้าจอหรือเมนู ช่วยให้โปรแกรมเมอร์สามารถออกแบบจอภาพ เขียนคำสั่งงาน และเรียกคอมไพล์เลอร์มาแปลโปรแกรมแล้วทดสอบโปรแกรมได้ นอกจากนี้ยังมีดีบักเกอร์ (Debugger) เพื่อช่วยในการค้นหาข้อผิดพลาดของโปรแกรม

การโปรแกรมด้วยภาพ การใช้ภาษาแบบนี้จะใช้เมาส์ ไอคอน และสัญลักษณ์บนหน้าจอหรือเมนูโดยการ point-and-click และ drag-and-drop ภาพไอคอนต่างๆ ที่ปรากฏให้เห็นบนหน้าจอซึ่งเป็นสัญลักษณ์แทนไฟล์หรือชุดคำสั่งของระบบปฏิบัติการคอมพิวเตอร์จะถูกเลือกหรือโยกย้ายไปมาก็โดยการคลิกเมาส์ลงและกดปุ่มของเมาส์ค้างไว้เพื่อ "จับ" จากนั้นก็ "ลาก" เมาส์ เพื่อให้ไอคอนเคลื่อนตามไปจนถึงตำแหน่งหนึ่งตำแหน่งใดบนหน้าจอที่ต้องการแล้ว "ปล่อย" ลงไว้ ณ ที่นั้นด้วยการเลิกกดปุ่มค้างไว้

2.5 วิธีการโปรแกรมโดยใช้การสาธิต (Programming by Demonstration)

การเขียนโปรแกรมโดยใช้การสาธิตเป็นรูปแบบที่นิยมใช้กับระดับของ End-User และเป็นทางเลือกหนึ่งที่ถูกนำมาใช้ก็คือ วิธีการโปรแกรมโดยใช้การสาธิตเทคนิคนี้จะง่ายกว่าการโปรแกรมแบบ Procedural Programming เพราะวิธีการโปรแกรมโดยใช้การสาธิตเป็นการพัฒนารูปแบบการโปรแกรมที่มุ่งเน้นเพื่อผู้ที่ไม่มีความชำนาญ (Non-experts)

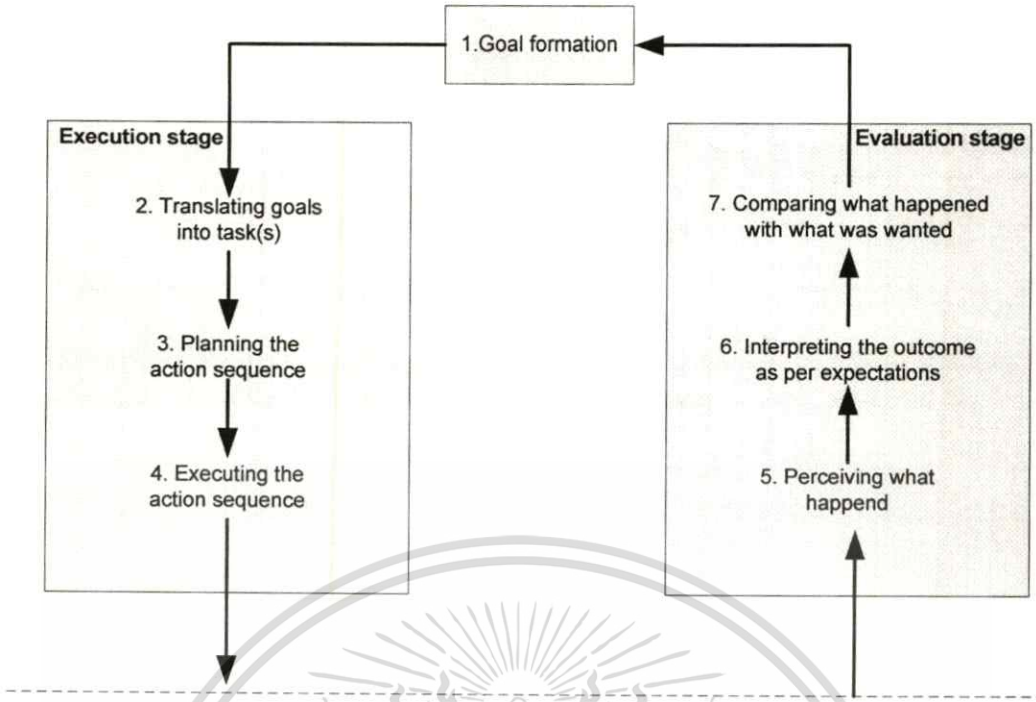
การเขียนโปรแกรมโดยใช้การสาธิต มีแนวความคิดหลักอยู่สองแนวความคิด แนวความคิดแรกคือการเขียนโปรแกรมบนพื้นฐานของตัวอย่าง (Example-based programming) ข้อมูลจากตัวอย่างจะถูกใช้ในการพัฒนาตัวโปรแกรม ด้วยปรัชญาที่ว่าตัวอย่างสร้างด้วยคอนกรีตง่ายสำหรับคนทำงานได้มากกว่าการบอกเล่า แนวความคิดนี้ Programming Tools จะมีหน้าที่คอยตรวจจับค่าของข้อมูลตัวอย่างและบันทึกไว้ เมื่อมีการบันทึกค่าข้อมูลเสร็จแล้วสามารถนำข้อมูลเหล่านั้นมาทำงานซ้ำในรูปแบบเดิมได้ แนวความคิดเหล่านี้เป็นที่รู้จักกันโดยทั่วไปในการโปรแกรมการเคลื่อนไหวของหุ่นยนต์ และระบบ Macro Recorder ซึ่งจดจำการสั่งการของผู้ใช้ในช่วงเวลานั้นๆ เพื่อสร้างชุดคำสั่ง แนวความคิดที่สองคือการเขียนโปรแกรมในส่วนติดต่อกับผู้ใช้ (Programming in the user interface) ด้วยปรัชญาที่ว่าอันไหนใช้บ่อย แนวความคิดนี้การสร้างประโยค (syntax) ในการเขียนโปรแกรมให้เหมือนกับเป็นการใช้งานระบบโดยปกติ ประโยชน์ของมันคือ ผู้ใช้สามารถได้รับการปรับแต่งหน้าจอเป็นที่เรียบร้อยและคุ้นเคย แนวความคิดเหล่านี้เป็นที่รู้จักกันโดยทั่วไปในการเขียนโปรแกรมเพื่อที่จะปรับแต่งส่วนติดต่อกับผู้ใช้แต่ละคนที่แตกต่างกัน

การเขียนโปรแกรมโดยใช้การสาธิต จะเป็นระบบการเขียนโปรแกรมที่มีส่วนติดต่อที่ง่าย สามารถทำให้ผู้ใช้นั้นสามารถสร้างโปรแกรมโดยปราศจากความรู้ในการสร้างประโยค (syntax) ของโปรแกรม หรือแม้แต่ปราศจากการมองเห็นชุดคำสั่งของโปรแกรม นั่นหมายความว่าผู้ใช้ไม่ต้องมีประสบการณ์ในการเขียนโปรแกรมหรือไม่มีการฝึกฝนการเขียนโปรแกรมเลย ข้อมูลที่ใช้ในการเขียนโปรแกรมนั้นคือสิ่งที่มีอยู่และใช้โดยปกติในส่วนติดต่อกับผู้ใช้อยู่แล้ว เพราะ Programming Tools จะมีหน้าที่คอยตรวจจับการกระทำของผู้ใช้ (User Action) ที่กระทำต่อส่วนติดต่อกับผู้ใช้เพื่อที่จะสร้างโปรแกรมขึ้นมา การเขียนโปรแกรมโดยใช้การสาธิตนี้ถูกพัฒนาขึ้นมาเพื่อที่ทำให้ส่วนติดต่อกับผู้ใช้ของระบบการเขียนโปรแกรมง่ายขึ้น และเมื่อเสร็จสิ้นการโปรแกรมแล้วสามารถทำงานซ้ำได้ตามเงื่อนไขโดยปราศจากการมองเห็นชุดคำสั่งของโปรแกรม

2.6 วงจรพฤติกรรมมนุษย์ (Human action cycle)

วงจรพฤติกรรมมนุษย์ คือ โมเดลทางกายภาพ ซึ่งสามารถอธิบายขั้นตอนที่มนุษย์เลือก เมื่อมีปฏิสัมพันธ์กับระบบคอมพิวเตอร์ โมเดลได้ถูกนำเสนอโดย Donald A. Norman [17] ผู้เชี่ยวชาญทางด้านพฤติกรรมของมนุษย์-คอมพิวเตอร์เมื่อปฏิสัมพันธ์ โมเดลนี้สามารถนำไปใช้ช่วยประเมินสมรรถนะของส่วนที่ติดต่อกับผู้ใช้

วงจรพฤติกรรมมนุษย์อธิบายถึงวิธีการที่มนุษย์อาจตั้งเป้าหมายและพัฒนาไปสู่ชุดของขั้นตอนที่ต้องการเพื่อไปสู่ความสำเร็จของเป้าหมายผ่านระบบคอมพิวเตอร์ ดังนั้นผู้ใช้จะปฏิบัติตามขั้นตอนและเป็นโมเดลที่ประกอบไปด้วยทั้งการกระทำทางด้านการรับรู้และด้านกายภาพ



รูปที่ 2.10 แสดงวงจรพฤติกรรมมนุษย์

สิ่งที่ใช้ในการประเมินส่วนเชื่อมต่อผู้ใช้ (Use in evaluation of user interfaces) โดยปกติผู้ประเมินส่วนเชื่อมต่อผู้ใช้จะใช้ชุดของคำถามในแต่ละรอบของขั้นตอน การประเมินคำตอบนั้นพิจารณาจากการแสดงข้อมูลที่เป็นประโยชน์ของส่วนเชื่อมต่อผู้ใช้เพื่อสนับสนุนการใช้งานว่าเพียงพอหรือเหมาะสมกับการใช้งานหรือไม่ ซึ่งคำถามที่ใช้ในการประเมินนั้นจะมีเจ็ดขั้นดังนี้

ขั้นที่ 1 สร้างเป้าหมาย (Forming a goal) ผู้ใช้มีความเข้าใจและความรู้ในงานเพียงพอ พร้อมทั้งมีความเข้าใจงานเพื่อที่จะสร้างเป้าหมายเพียงพอหรือไม่ และส่วนเชื่อมต่อผู้ใช้ (UI) สามารถช่วยให้ผู้ใช้สร้างเป้าหมายได้หรือไม่

ขั้นที่ 2 แปลงเป้าหมายไปสู่งานหรือชุดงาน (Translating the goal into a task or a set of tasks) ผู้ใช้มีความเข้าใจและความรู้ในงานเพียงพอ พร้อมทั้งมีความเข้าใจงานเพื่อที่จะกำหนดเป็นชุดคำสั่งเพียงพอหรือไม่ และส่วนเชื่อมต่อผู้ใช้ (UI) สามารถช่วยให้ผู้ใช้กำหนดเป็นชุดคำสั่งได้หรือไม่

ขั้นที่ 3 วางแผนงานเพื่อสร้างลำดับการปฏิบัติ (Planning an action sequence) ผู้ใช้มีความเข้าใจและความรู้ในงานเพียงพอ พร้อมทั้งมีความเข้าใจงานเพื่อที่จะสร้างลำดับการปฏิบัติเพียงพอหรือไม่ และส่วนเชื่อมต่อผู้ใช้ (UI) สามารถช่วยให้ผู้ใช้สร้างลำดับการปฏิบัติได้หรือไม่

ขั้นที่ 4 ปฏิบัติตามลำดับการปฏิบัติ (Executing the action sequence) ผู้ใช้ปกติสามารถเรียนรู้และใช้ส่วนเชื่อมต่อผู้ใช้ (UI) ได้โดยง่ายหรือไม่, ผลการปฏิบัติของระบบจะสามารถจับคู่กับ

ความต้องการของผู้ใช้ได้หรือไม่, ตัวบ่งบอกการใช้งานและความชัดเจนของการปฏิบัติส่งผลดีหรือไม่, ผู้ใช้จะสามารถเกิดทักษะความเชี่ยวชาญในการใช้ระบบได้หรือไม่ และระบบจะสามารถสนับสนุนการพัฒนาทักษะความเชี่ยวชาญได้หรือไม่

ขั้นที่ 5 เข้าใจผลภายหลังจากลงมือปฏิบัติ (Perceiving what happened) ผู้ใช้จะสามารถเข้าใจลำดับขั้นตอนของระบบได้หรือไม่ และส่วนเชื่อมต่อผู้ใช้ (UI) จะแสดงการตอบสนองต่อผลของการกระทำของผู้ใช้ให้ผู้ใช้ได้รับรู้อย่างเพียงพอหรือไม่

ขั้นที่ 6 แปลผลลัพธ์ตามความคาดหวังของผู้ใช้ (Interpreting the outcome according to the users' expectations) ผู้ใช้จะสามารถเข้าใจผลการตอบสนองได้หรือไม่ และส่วนเชื่อมต่อผู้ใช้ (UI) จะแสดงการตอบสนองต่อการแปลผลลัพธ์อย่างเพียงพอหรือไม่

ขั้นที่ 7 ประเมินสิ่งที่เกิดขึ้นกับผลที่คาดหวัง (Evaluating what happened against what was intended) ผู้ใช้จะสามารถเปรียบเทียบสิ่งที่เกิดขึ้นกับสิ่งที่คาดหวังไว้ ได้หรือไม่

2.7 งานวิจัยที่เกี่ยวข้อง

จากปัญหาประการสำคัญของการนำเอาอุปกรณ์พีแอลซีมาใช้ทดแทนอุปกรณ์รีเลย์ คือผู้ที่ปฏิบัติงานที่มีความรู้ความเข้าใจในการควบคุมกระบวนการทำงานของเครื่องจักร และอุปกรณ์ในโรงงานนั้น มักจะไม่มีทักษะทางด้านการโปรแกรม เนื่องจากรูปแบบการโปรแกรมสำหรับอุปกรณ์พีแอลซีในปัจจุบันนั้นมีสองรูปแบบคือ รูปแบบแรกคือรูปแบบของวิธีการเขียนโปรแกรมที่มีพื้นฐานมาจากรายการปฏิบัติงาน (Procedures) วิธีการนี้มีส่วนประกอบที่ใช้ภาษาอังกฤษหรือโปรแกรมเชิงตัวอักษร เช่นเดียวกับภาษาระดับสูง รูปแบบที่สองคือรูปแบบของวิธีการเขียนโปรแกรมที่เน้นให้มีการใช้ภาพประกอบมากขึ้น เพื่อความสะดวกและสื่อความหมายกับผู้ใช้ได้ดีที่สุด การใช้ภาษาแบบนี้จะใช้เมาส์ ไอคอน และสัญลักษณ์บนหน้าจอ ซึ่งทั้งสองวิธียังเป็นการนำเอาชุดคำสั่งมาจัดเรียงตามกฎเกณฑ์ และรูปแบบของภาษานั้นๆ ซึ่งทั้งสองรูปแบบยังคงต้องอาศัยทักษะทางด้านการโปรแกรม และการฝึกฝนมากพอสมควร

จากการศึกษาปัญหาข้างต้นวิธีการที่ใกล้เคียงกับปัญหามากที่สุด คือการจำลองเครื่องจักรขึ้นมา แล้วให้ผู้ใช้ทำการควบคุมเครื่องจักรด้วยมือ (Manual Control) จากนั้นระบบจะทำการเรียนรู้วิธีการควบคุมดังกล่าว เพื่อมาสร้างเป็นชุดคำสั่งและสามารถทำงานได้เหมือนการควบคุมโดยมนุษย์ (Human Control) ซึ่งวิธีการที่เหมาะสมกับปัญหาคือวิธีการโปรแกรมโดยใช้การสาธิต (Programming by Demonstration) ดังนั้นจึงขอนำเสนอผลงานวิจัยที่เกี่ยวข้องในสองด้าน ด้านแรกคือ งานวิจัยทางด้านวิธีการโปรแกรมโดยใช้การสาธิต (Programming by Demonstration) ด้านที่สองคือการโปรแกรมอุปกรณ์พีแอลซี (PLC programming)

2.7.1 งานวิจัยทางด้านวิธีการโปรแกรมโดยใช้การสาธิต (Programming by Demonstration)

ในปัจจุบันงานวิจัยทางด้านวิธีการโปรแกรมโดยใช้การสาธิตสามารถจำแนกได้ดังนี้

1) Repetitive task automation software วิธีการโปรแกรมโดยใช้การสาธิตถูกนำมาใช้กับการทำงานที่ซ้ำๆ วิธีการนี้ทำหน้าที่คล้ายกับระบบ Macro Recorder ซึ่งจดจำการกระทำของผู้ใช้ในเวลานั้นเพื่อสร้างชุดคำสั่ง และสามารถทำงานตามรูปแบบเดิมได้โดยอัตโนมัติ [1, 5, 13]

2) User Interface Customization and Personalization วิธีการโปรแกรมโดยใช้การสาธิตนำมาใช้กับการปรับแต่งส่วนเชื่อมต่อผู้ใช้ (User Interface) เพื่อให้เข้ากับลักษณะเฉพาะของการใช้งานสำหรับผู้ใช้แต่ละคน ซึ่งเป็นการพยายามที่จะตรวจสอบคุณลักษณะเฉพาะของส่วนเชื่อมต่อจากการใช้งานของผู้ใช้ และปรับแต่งเปลี่ยนแปลงลักษณะของส่วนเชื่อมต่อให้เหมาะสมกับผู้ใช้ เพื่อที่จะทำให้ระบบมีประสิทธิภาพดีที่สุดเท่าที่จะสามารถทำได้ในการเข้าใช้งานที่ทำบ่อยๆ เช่น การปรับแต่งในการแสดงเมนูของรายการที่ถูกใช้งานบ่อย และซ่อนเมนูที่ไม่ได้ถูกใช้ [6]

3) Programming by Demonstration for Service Robotic tasks เป็นการโปรแกรมการเคลื่อนไหวของหุ่นยนต์ ส่วนใหญ่จะใช้กับแขนของหุ่นยนต์ โดยจะถูกโปรแกรมการเคลื่อนไหวโดยตรงจากการเลียนแบบการเคลื่อนไหวของมนุษย์ โดยใช้อุปกรณ์ตรวจจับ ซึ่งส่วนใหญ่ใช้ในงานอุตสาหกรรมการประกอบชิ้นส่วนต่างๆ เช่น การประกอบรถยนต์ การประกอบเครื่องใช้ไฟฟ้า เป็นต้น [2-4, 12, 14]

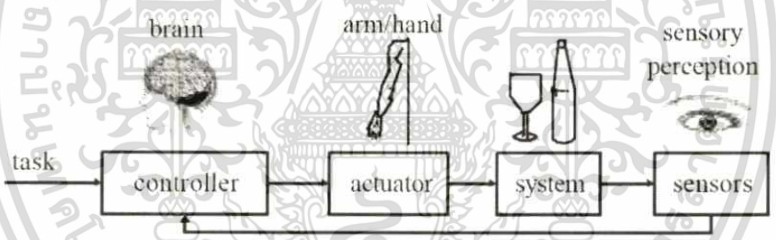
ตัวอย่างผลงานวิจัยทางด้านการโปรแกรมโดยการสาธิตมีดังนี้

Gordon W. Paynter and Ian H. Witten [13] ได้พัฒนา Developing a Programming by Demonstration Tool เพื่อเป็นเครื่องมือภายใต้ระบบปฏิบัติการ Apple Macintosh ช่วยในการทำงานที่ซ้ำๆ ของ Application ที่ไม่มีระบบ Macro record ในการทำงานซ้ำๆ ได้แก่ โปรแกรม Familiar History ซึ่งเป็นเครื่องมือที่คอยดักจับ User Action ที่กระทำต่อ Application ต่างๆ โดยเก็บรายการของ User Action ไว้เป็นทะเบียนประวัติของ User Action และเมื่อผู้ใช้ต้องการทำงานนั้นซ้ำๆ จะมีเครื่องมือชื่อ Familiar prediction ที่สามารถเรียก User Action มาทำงานนั้นซ้ำๆ โดยสามารถกำหนดจำนวนรอบที่ทำงานได้

จากงานวิจัยของ Gordon W. Paynter and Ian H. Witten เป็นการนำหลักการวิธีการโปรแกรมโดยใช้การสาธิตมาพัฒนาเครื่องมือที่คอยดักจับและจดจำลำดับขั้นตอนการทำงานของผู้ใช้ เพื่อให้สามารถนำกลับมาใช้ได้อีกครั้ง ซึ่งจะพบว่าระบบที่เครื่องมือนี้คอยตรวจสอบนั้นมีการทำงานและรูปแบบคำสั่งของระบบที่สามารถทำงานได้อยู่แล้ว ซึ่งระบบในที่นี้คือตัวระบบปฏิบัติการ (Operating system) ของคอมพิวเตอร์นั่นเอง เครื่องมือการโปรแกรมมีหน้าที่

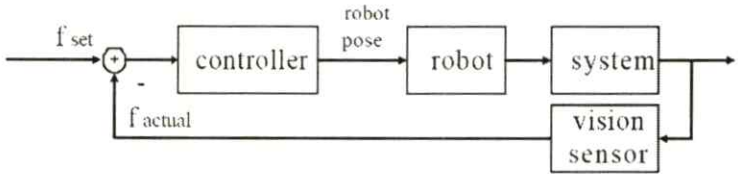
เพียงแต่เป็นตัวจดจำคำสั่งที่ผู้ใช้สั่งงานผ่านเมาส์ และคีย์บอร์ด และการบันทึกรูปแบบคำสั่งนั้นเพื่อนำกลับไปทำงานซ้ำแบบเดิมเท่านั้น ซึ่งแตกต่างจากงานวิจัยที่ได้นำเสนอนี้ เนื่องจากตัวระบบที่ใช้วิธีการโปรแกรมโดยใช้การสาธิตนั้นจะยังไม่มีรูปแบบการทำงานของระบบ หรือการทำงานที่สามารถตอบโต้กับผู้ใช้อยู่ก่อนแล้ว ดังนั้นผู้ใช้จะต้องโปรแกรมการทำงานของระบบขึ้นเสียก่อน ซึ่งระบบนี้หมายถึงการทำงานของแบบจำลองเครื่องจักรที่ถูกจำลองขึ้น

Haiying Sh e, Axel Graeser [2] ได้พัฒนา Closed Loop Control Structures and Automatic Set Point Generation in Programming by Demonstration for Service Robotic Tasks โดยใช้หลักการเดียวกับขบวนการทำงานของมนุษย์ดังแสดงในรูปที่ 2.11 คือมีส่วนควบคุมการทำงาน หมายถึง สมองของคนเรา ส่วนทำงานคือ แขนและมือ ตัวระบบประกอบด้วยวัตถุที่เราต้องกระทำในที่นี้คือ แก้วน้ำ และขวดน้ำ ส่วนตรวจจับการเปลี่ยนแปลงของวัตถุคือ ตา ซึ่งคอยตรวจจับระดับน้ำในแก้ว และป้อนกลับเพื่อให้ส่วนควบคุมสั่งหยุดหรือรินน้ำต่อไป ซึ่งในขั้นตอนการสาธิตนั้นจะมีตัวตรวจจับการกระทำของมนุษย์อยู่สองแบบคือ Vision sensor (Camera) ตรวจจับการเปลี่ยนแปลงของวัตถุในที่นี้คือ ระดับน้ำในแก้ว และ Polhemus sensor (3D digitizer) ตรวจจับการเคลื่อนที่ของแขนที่มีต่อขวดน้ำ โดย Polhemus sensor จะติดอยู่ที่ขวดน้ำ



รูปที่ 2.11 แสดงการสาธิตของมนุษย์โดยป้อนกลับจากการมองด้วยสายตา

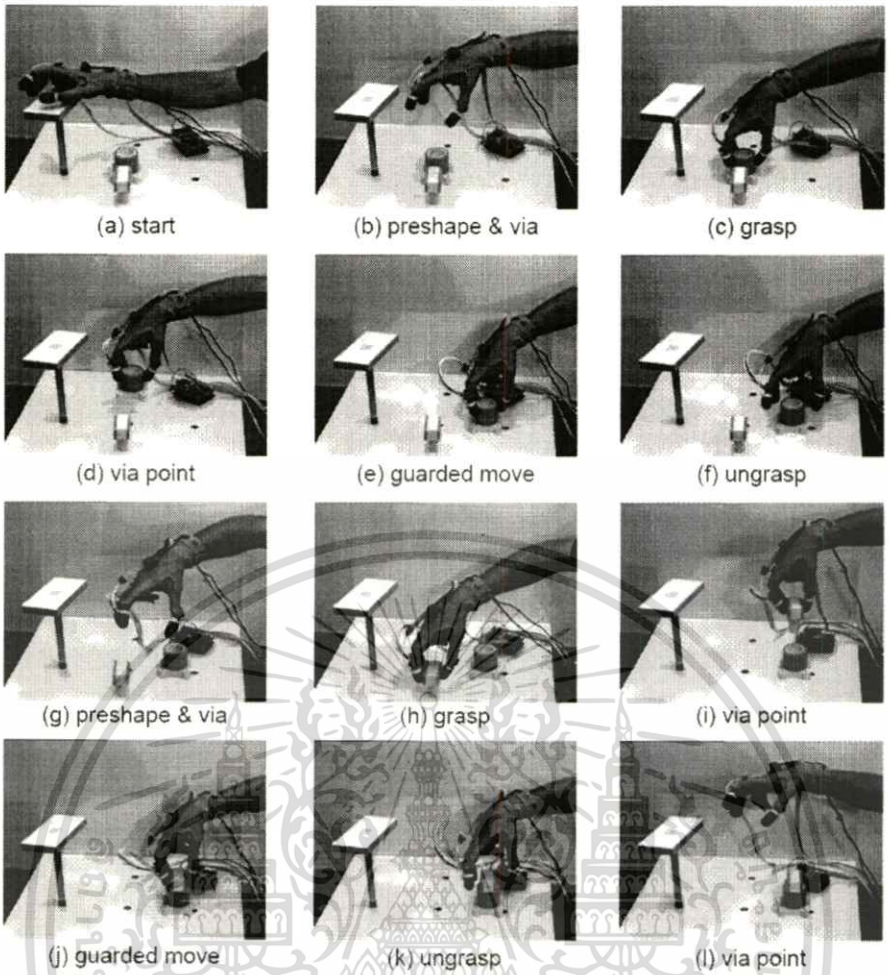
การทำงานของหุ่นยนต์จากการป้อนกลับ การทำงานจะเริ่มจากการจับขวดน้ำซึ่งได้จากการหาตำแหน่งของวัตถุ และการรินน้ำซึ่งได้ข้อมูลจาก Polhemus sensor ซึ่งอ่านค่าของการเคลื่อนที่ของขวดน้ำจากขั้นตอนการสาธิตโดยมนุษย์ ในการรินน้ำนั้นจะถูกควบคุมด้วยข้อมูลของระดับน้ำในแก้วที่ถูกป้อนกลับจาก Vision sensor เมื่อระดับน้ำในแก้วถึงจุดที่กำหนด หุ่นยนต์จะหยุดรินน้ำ ดังแสดงผังการทำงานในรูปที่ 2.12



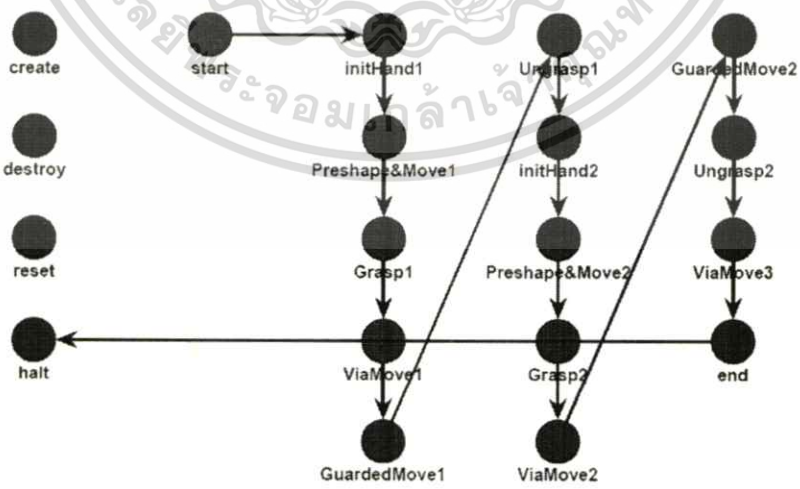
รูปที่ 2.12 แสดงการทำงานของหุ่นยนต์โดยการป้อนกลับจาก Vision sensor

จากงานวิจัยของ Haiying Sh e, Axel Graeser เป็นการนำวิธีการโปรแกรมโดยใช้การสาธิตมาพัฒนาเครื่องมือในการจดจำรูปแบบการทำงานของมนุษย์ โดยรูปแบบการจดจำนั้นจะเป็นการจดจำค่าของตำแหน่งต่าง ๆ ที่อ่านได้จากตัวตรวจจับ (Sensor) และนำค่าข้อมูลตำแหน่งต่างๆ ที่ได้จดจำไว้มาทำงานซ้ำอีกครั้ง วิธีการนี้มีส่วนเหมือนและส่วนต่างกับงานวิจัยที่ได้นำเสนอ ส่วนเหมือนคือเป็นการจำค่าสถานะของอุปกรณ์ต่างๆ ที่มีผลเนื่องมาจากการกระทำของมนุษย์ ส่วนที่ต่างจากงานวิจัยที่ได้นำเสนอคือ ผลงานวิจัยของ Haiying Sh e, Axel Graeser นั้น การตอบสนองกลับมายังระบบเกิดจากอุปกรณ์จริงที่มีคุณสมบัติและรูปแบบของการตอบสนองกลับด้วยตัวเอง เช่น เมื่อเรารินน้ำลงในแก้วระดับน้ำจะเพิ่มสูงขึ้น ซึ่งในงานวิจัยที่ได้นำเสนอนั้น คุณสมบัติในส่วนนี้คือคุณสมบัติที่เราจะต้องโปรแกรมให้กับแบบจำลอง

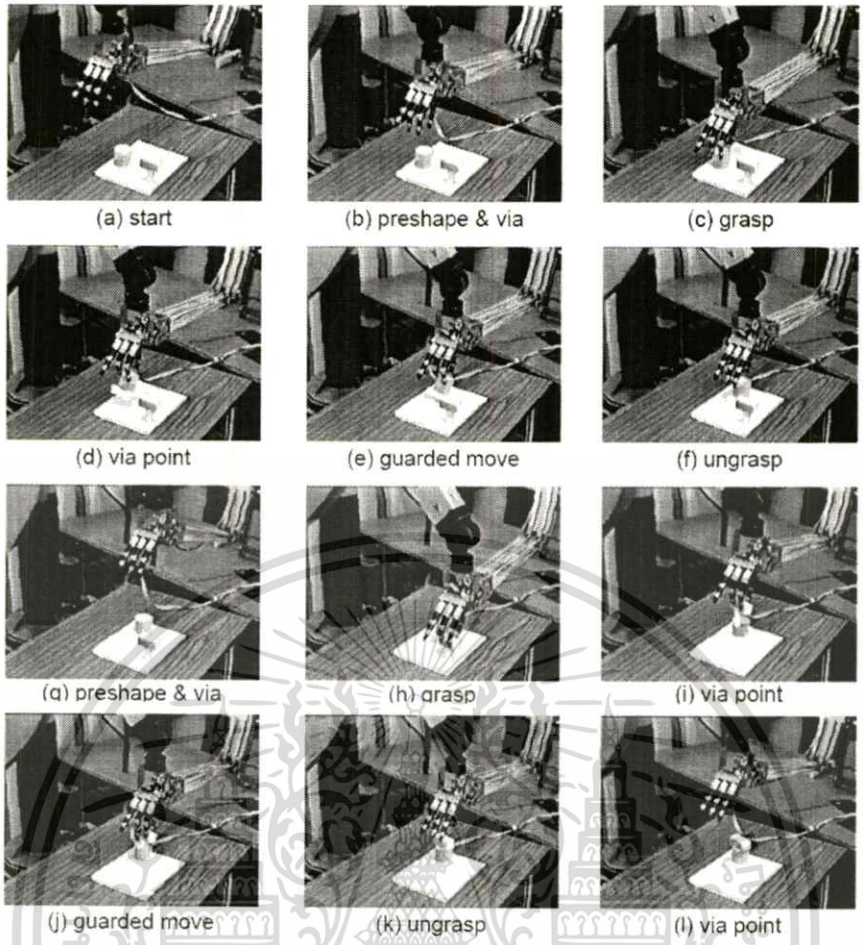
Richard M. Voyles and Pradeep K. Khosla [12] ได้พัฒนา A Multi-Agent system for programming robots by human demonstration ขึ้นมา ซึ่งเป็นการสาธิตวิธีการหยิบและเคลื่อนย้ายสิ่งของโดยตรงจากการเคลื่อนที่และหยิบจับสิ่งของโดยมนุษย์ให้กับแขนหุ่นยนต์ โดยการติดตั้งตัวตรวจจับ (Sensor) ที่มือและนิ้วของมนุษย์เพื่อตรวจวัดการเคลื่อนที่ของมือและการหยิบจับของนิ้ว ซึ่งการสาธิตโดยมนุษย์มีขั้นตอนดังแสดงในรูปที่ 2.13 ผลของการสาธิตดังแสดงในรูปที่ 2.14 และการทำงานของหุ่นยนต์ดังแสดงในรูปที่ 2.15



รูปที่ 2.13 แสดงขั้นตอนการสาธิตการเคลื่อนที่โดยมนุษย์



รูปที่ 2.14 แสดงผลของขั้นตอนการทำงานที่ได้จากสาธิต

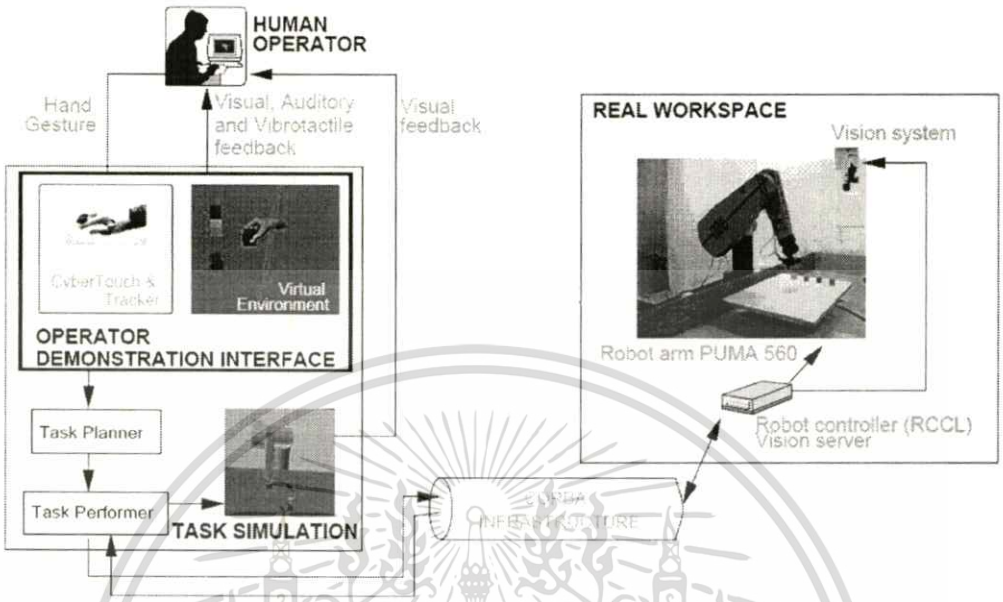


รูปที่ 2.15 แสดงการทำงานของหุ่นยนต์ที่ถูกโปรแกรมตามการสาธิต

จากงานวิจัยของ Richard M. Voyles and Pradeep K. Khosla เป็นการนำวิธีการโปรแกรมโดยใช้การสาธิตมาพัฒนาเครื่องมือในการจดจำรูปแบบการทำงานของมนุษย์ โดยรูปแบบการจดจำนั้นจะเป็นการจดจำตำแหน่งการเคลื่อนไหวของแขนมนุษย์ ซึ่งอ่านได้จากตัวตรวจจับและนำค่าข้อมูลตำแหน่งต่างๆ ที่ได้จดจำไว้มาทำงานซ้ำอีกครั้ง โดยไม่มีปัจจัยอื่นๆ ที่ส่งผลต่อการเปลี่ยนแปลงเงื่อนไขการทำงานของรูปแบบการทำงานที่ได้ถูกบันทึกไว้แล้ว ซึ่งจะแตกต่างกับงานวิจัยที่ได้นำเสนอ เพราะการเปลี่ยนแปลงคุณสมบัติของอุปกรณ์ต่างๆ ในระบบจะส่งผลโดยตรงต่อการเปลี่ยนแปลงเงื่อนไขในการสั่งให้ระบบทำงาน

Jacopo Aleotti, Stefano Caselli, and Giuliano Maccherozzi [14] ได้พัฒนา Trajectory Reconstruction with NURBS Curves for Robot Programming by Demonstration เพื่อใช้ในการโปรแกรมการทำงานของแขนหุ่นยนต์โดยวิธีการสาธิตผ่านแบบจำลอง (3D Model Simulation) ซึ่งจะใช้ถุงมือควบคุมแบบจำลอง (CyberTouch VR glove) เมื่อสร้างแบบจำลองที่จำลองตัววัตถุและตำแหน่งการวางของวัตถุแล้ว จากนั้นผู้ใช้จะใช้ถุงมือควบคุมแบบจำลอง ทำการหยิบและเคลื่อนย้ายวัตถุในแบบจำลอง จากนั้นระบบจะทำการจดจำการเคลื่อนที่ในการใช้ถุงมือควบคุม

แบบจำลอง และแปลงเป็นชุดคำสั่งในการควบคุมแขนหุ่นยนต์ และทำการส่งผ่านการสื่อสารทางไกลโดยใช้ CORBA-based framework เป็นตัวส่งผ่านข้อมูลการควบคุม ดังแสดงในรูปที่ 2.16



รูปที่ 2.16 แสดงผังการทำงานของระบบ Trajectory Reconstruction with NURBS Curves for Robot Programming by Demonstration

จากงานวิจัยของ Jacopo Aleotti, Stefano Caselli, and Giuliano Maccherozzi เป็นการนำวิธีการโปรแกรมโดยใช้การสาธิต มาพัฒนาเครื่องมือที่ทำให้เครื่องจักรสามารถทำตามมนุษย์ผ่านระบบการสื่อสารทางไกล โดยการจดจำรูปแบบการทำงานของมนุษย์นั้นอาศัยการจดจำตำแหน่งการเคลื่อนไหวของแขนมนุษย์ ซึ่งอ่านได้จากตัวตรวจจับและนำค่าข้อมูลในตำแหน่งต่างๆ ให้เครื่องจักรทำตาม โดยปัจจัยที่ส่งผลต่อการเปลี่ยนแปลงเงื่อนไขการทำงานของรูปแบบการทำงานนั้นจะมาจากการตัดสินใจของมนุษย์โดยที่รับรู้ผ่านทางระบบป้อนกลับในลักษณะภาพจำลองจากข้อมูลจริงของเครื่อง ซึ่งจะแตกต่างกับงานวิจัยที่ได้นำเสนอ เพราะการเปลี่ยนแปลงเงื่อนไขการทำงานนั้นเกิดจากเงื่อนไขที่ได้มีการโปรแกรมการทำงานไว้แล้ว ซึ่งเงื่อนไขการทำงานที่เปลี่ยนแปลงนั้นจะเป็นไปตามการเปลี่ยนแปลงของคุณสมบัติของอุปกรณ์ต่างๆ ในระบบ ซึ่งจะส่งผลโดยตรงต่อการเปลี่ยนแปลงเงื่อนไขในการสั่งให้ระบบทำงาน

2.7.2 งานวิจัยทางด้านการโปรแกรมอุปกรณ์พีแอลซี (PLC programming)

ในปัจจุบันงานวิจัยทางด้านการโปรแกรมอุปกรณ์พีแอลซีนั้น จะครอบคลุมเนื้อหาตามตามมาตรฐาน IEC 1131 ทั้งแปดส่วนคือ

IEC 61131-1 ข้อกำหนดทั่วไป การนิยามความหมาย ลักษณะสำคัญ (General information)

IEC 61131-2 การกำหนดความต้องการของอุปกรณ์และการทดสอบ (Equipment requirements and tests)

IEC 61131-3 ภาษาที่ใช้ในการเขียนโปรแกรม (Programming Languages)

IEC 61131-4 เป็น Technical report สำหรับการแนะนำผู้ใช้ (User Guidelines)

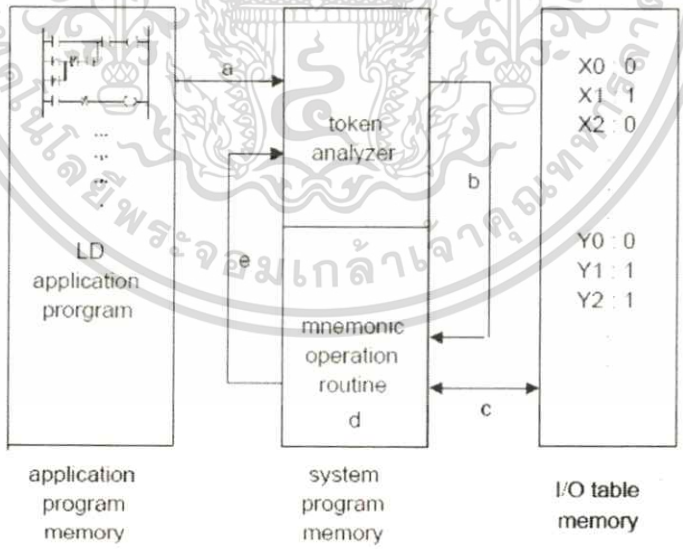
IEC 61131-5 กำหนดรูปแบบการติดต่อสื่อสารข้อมูล (Messaging service specification) ระหว่างอุปกรณ์พีแอลซีกับผู้ควบคุมอุปกรณ์พีแอลซี และอุปกรณ์พีแอลซีกับอุปกรณ์อื่นๆ โดยอาศัยมาตรฐาน ISO/IEC 9506

IEC 61131-7 การเขียนโปรแกรมควบคุมแบบ Fuzzy (Fuzzy control programming)

IEC 61131-8 ข้อแนะนำสำหรับผู้พัฒนาซอฟแวร์เพื่อเป็นเครื่องมือสำหรับภาษาที่ใช้ในการเขียนโปรแกรมอุปกรณ์พีแอลซี ตามที่กำหนดไว้ในIEC 61131 part 3 (Guidelines for the application and implementation of programming languages)

ตัวอย่างผลงานวิจัยทางด้านการโปรแกรมอุปกรณ์พีแอลซีมีดังนี้

Hyung Seok Kimy, Dong Sung Kimy, Naehyuck Changz, Wook Hyun Kwony [15] ได้พัฒนา A Translation Method of Ladder Diagram on PLC with Application to a Manufacturing Process เพื่อพัฒนาการแปลงคำสั่งที่ใช้กับอุปกรณ์พีแอลซี จากภาษาแลดเดอร์ไปสู่ภาษานีโมนิค โดยมีองค์ประกอบดังแสดงในรูปที่ 2.17

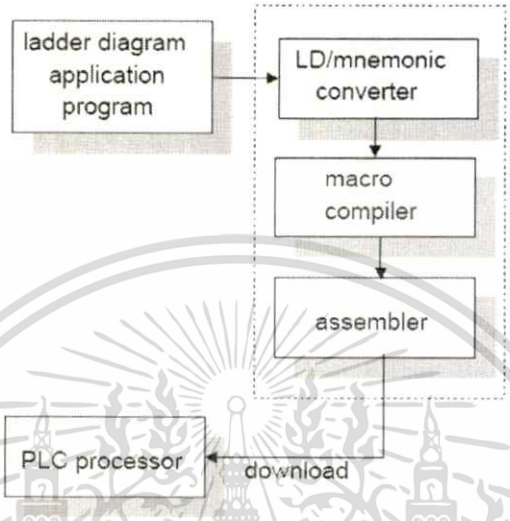


รูปที่ 2.17 แสดง Block diagram ของตัวแปลภาษาแลดเดอร์

โดยมีหลักการทำงานดังนี้ ลำดับแรก (a) อ่านชุดคำสั่งเข้ามาทีละคำสั่ง (b) ไปเรียกส่วนทำงานที่เกี่ยวข้องของแต่ละคำสั่งเพื่อมาวิเคราะห์องค์ประกอบของชุดคำสั่งนั้นๆ (c) แปลงให้อยู่ใน

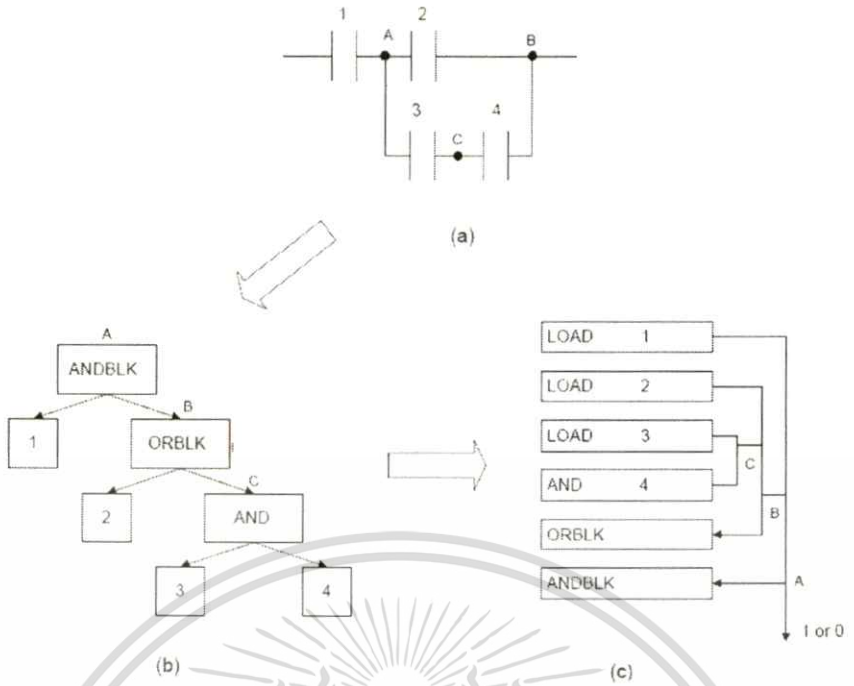
รูปของชุดคำสั่งนีโมนิคและอ่านค่าข้อมูลจากหน่วยความจำอินพุทและเอาท์พุท (d) ทำงานตามคำสั่งนั้น (e) กลับไปทำงานในขั้นตอน (a) และทำงานตามขั้นตอน (a) ถึงขั้นตอน (e) ซ้ำจนจบโปรแกรม

แผนผังหลักของกระแสการแปลภาษาแลดเดอร์ไปสู่ชุดคำสั่งภาษานีโมนิค เป็นดังนี้



รูปที่ 2.18 แสดงแผนผังหลักของกระแสการแปลภาษาแลดเดอร์ไปสู่ชุดคำสั่งภาษานีโมนิค

จากแผนผังหลักของกระแสการแปลภาษาแลดเดอร์ไปสู่ชุดคำสั่งภาษานีโมนิคนั้นจะเห็นได้ว่าสิ่งที่นำเข้าคือภาษาแลดเดอร์ และสิ่งสุดท้ายที่ได้คือ คำสั่งที่เป็นลักษณะไบนารีโค้ดที่สามารถนำเข้าไปในอุปกรณ์พีแอลซีได้ทันที

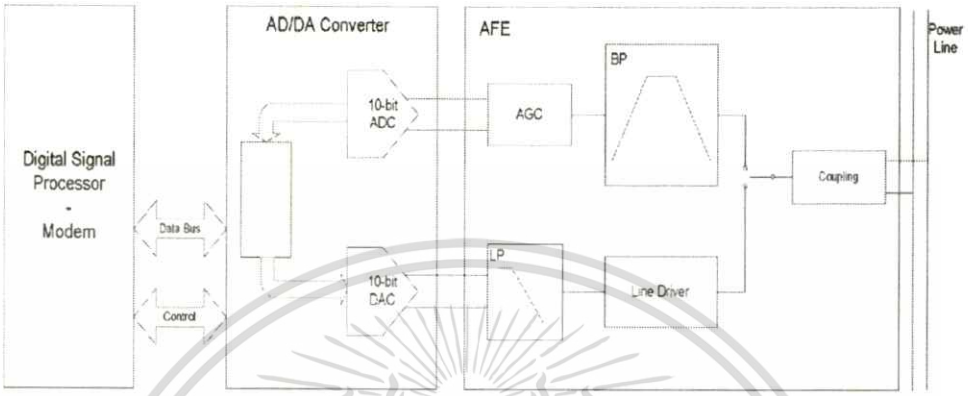


รูปที่ 2.19 แสดงวิธีการแปลงจากภาษาแลดเดอร์ไปสู่ชุดคำสั่งภาษานีโมนิค

จากรูปที่ 2.19 นั้นการแปลงเริ่มจากขั้นตอน (a) เป็นรูปแบบของภาษาแลดเดอร์ก่อนการแปลง โดยในขั้นตอนนี้จะมีการกำหนดจุดต่อของแต่ละส่วนโดยกำหนดให้เป็นจุด A,B และ C จากนั้น (b) เป็นขั้นตอนการแปลงภาษาแลดเดอร์ให้อยู่ในรูปแบบของแผนภูมิต้นไม้โดยให้จุด A,B และ C เป็นรากของแต่ละส่วน จากนั้นทำการท่องไปในแผนภูมิต้นไม้ เราจะได้รูปแบบดังนี้ 1 ^ [2 V[3 ^ 4]] โดยที่ "V" แทนตรรกะ AND และ "^" แทนตรรกะ OR โดยที่หากการเชื่อมด้วยตรรกะนั้นเป็นการเชื่อมระหว่างในวงเล็บ "[]" ตรรกะที่ใช้จะเปลี่ยนไปเป็น "V" แทนตรรกะ ANDBLK (And Block) และ "^" แทนตรรกะ ORBLK (Or Block) ขั้นตอนสุดท้าย ขั้นตอน (c) คือการแปลงจากประโยคที่ "1 ^ [2 V[3 ^ 4]] " ได้จากขั้นตอน (b) ไปเป็นภาษานีโมนิค ซึ่งจะได้ผลลัพธ์ดังรูปที่ 2.19

จากงานวิจัยของ Hyung Seok Kimy, Dong Sung Kimy, Naehyuck Changz, Wook Hyun Kwony จะเป็นการแปลงจากภาษาแลดเดอร์ ซึ่งอยู่ในรูปแบบของแผนภูมิตรรกะ ไปสู่ชุดคำสั่งภาษานีโมนิคซึ่งภาษานีโมนิคนั้นสามารถแปลงสู่คำสั่งที่เป็นลักษณะไบนารีโค้ดที่สามารถนำเข้าไปในอุปกรณ์พีแอลซีได้ทันที ในงานวิจัยที่ได้นำเสนอนี้จะเป็นการแปลงจากการกระทำของผู้ใช้ที่มีต่ออุปกรณ์ต่างๆ มาสร้างเป็นชุดของคำสั่งในรูปแบบของนิพจน์คำสั่งเกี่ยวกับการเลือกทำงาน if…..then……..else และภาษานีโมนิคที่สามารถแปลงสู่คำสั่งที่เป็นลักษณะไบนารีโค้ดที่สามารถนำเข้าไปในอุปกรณ์พีแอลซีได้ทันที

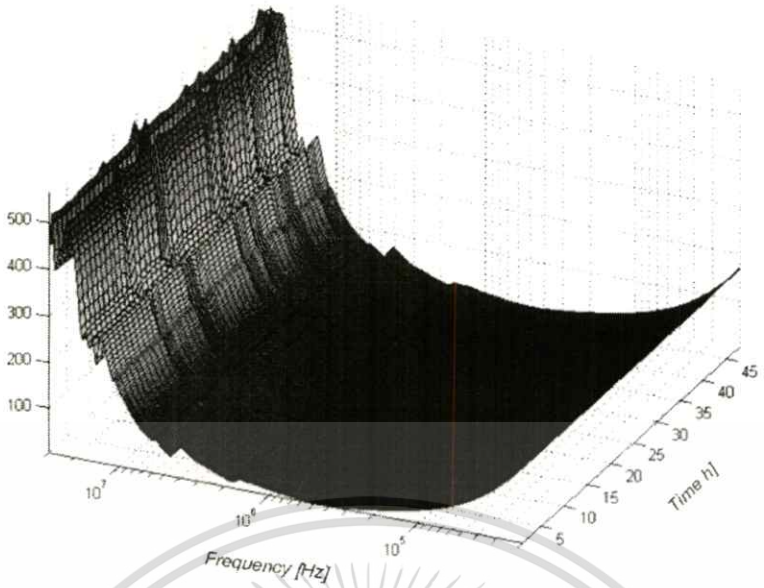
Luis Simões, José A. B. Gerald [16] ได้พัฒนา A Communication System for Power Lines เพื่อพัฒนาการติดต่อสื่อสารข้อมูลระหว่างอุปกรณ์พีแอลซี โดยการส่งข้อมูลนั้นจะอาศัยผ่านทาง Power Lines โดยที่ได้ความเร็วในการส่งข้อมูลสูงถึง 1 Mbit/sec โดยเน้นความสำคัญไปที่วงจรการเชื่อมต่อและวงจรการป้องกัน ดังแสดงโครงสร้างของระบบในรูปที่ 2.20



รูปที่ 2.20 แสดงโครงสร้างของระบบ

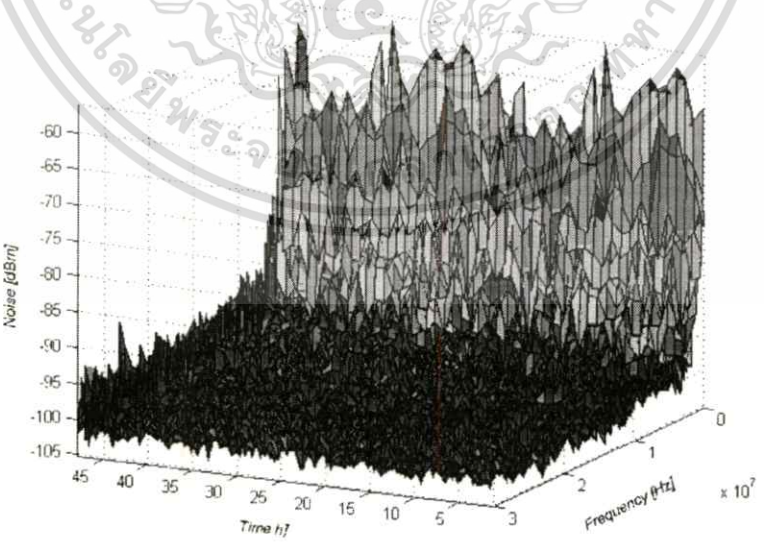
ระบบนี้ถูกออกแบบโดยการเชื่อมโยงระหว่างสัญญาณอนาล็อก และสัญญาณ ดิจิตอลด้วยวงจรประมวลผลสัญญาณ ซึ่งสามารถแบ่งออกเป็นสามส่วน ส่วนแรกคือ AFE (Analog Front End) จะถูกออกแบบให้เป็นวงจรในการประมวลผลสัญญาณอนาล็อก ส่วนที่สองคือส่วนของการแปลงสัญญาณอนาล็อกไปเป็นสัญญาณดิจิตอล และจากสัญญาณดิจิตอลไปเป็นสัญญาณอนาล็อก และส่วนสุดท้ายคือส่วนของโมเด็ม ส่วนนี้จะทำหน้าที่ประมวลผลสัญญาณสำหรับหน่วยควบคุม โดยการ Modulation และ Demodulation

ปัจจัยที่มีผลสำคัญต่อการส่งสัญญาณข้อมูลเข้าไปใน Power Lines ก็คือการเหนี่ยวนำของคลื่นสนามแม่เหล็กไฟฟ้าที่ก่อให้เกิดการลดทอนสัญญาณได้ แต่หากพิจารณาคุณลักษณะพิเศษของ Power Lines ที่มีต่อการเหนี่ยวนำของคลื่นสนามแม่เหล็กไฟฟ้า จะเห็นได้ว่าช่วงความถี่ที่ส่งผ่านสายได้ดีที่สุดคือช่วงของความถี่ 500 kHz ถึง 21MHz ดังแสดงผลการทดลองในรูปที่ 2.21



รูปที่ 2.21 แสดงถึงค่าImpedance throughout เป็นระยะเวลา 48 ชั่วโมง

อีกปัจจัยสำคัญหนึ่งก็คือสัญญาณรบกวนที่เกิดจากสภาพแวดล้อม ซึ่งอาจเกิดจากสัญญาณคลื่นวิทยุกระจายเสียง, เตาไมโครเวฟ, มอเตอร์ไฟฟ้า หรือการเปิดปิดสวิทช์ไฟฟ้า ฯลฯ จากการทดลองระดับความถี่ที่มีผลต่อระดับความแรงของสัญญาณรบกวนในห้องทดลองโดยใช้ความถี่ไปจนถึง 30 MHz จะได้ผลลัพธ์ดังแสดงในรูปที่ 2.22



รูปที่ 2.22 แสดงระดับความถี่ที่มีผลต่อระดับความแรงของสัญญาณรบกวนเป็นระยะเวลา 48 ชั่วโมง

จากผลการทดลองเกี่ยวกับคุณลักษณะพิเศษของ Power Lines จึงนำมาออกแบบวงจรในส่วนของ AFE (Analog Front End) ซึ่งเป็นวงจรในการประมวลผลสัญญาณอนาล็อกนั้นใช้ความถี่ในย่าน 1 MHz to 30 MHz ซึ่งเป็นผลดีต่อการเหนี่ยวนำของคลื่นสนามแม่เหล็กไฟฟ้า และมีผลกระทบน้อยที่สุดจากสัญญาณรบกวน

จากงานวิจัยของ Luis Simões, José A. B. Gerald เป็นการใช้ประโยชน์จากคุณสมบัติของอุปกรณ์ต่างๆ งานวิจัยดังกล่าวเป็นการนำเอาคุณลักษณะพิเศษของ Power Lines ออกแบบวงจรในส่วนของ AFE (Analog Front End) ซึ่งเป็นวงจรในการประมวลผลสัญญาณอนาล็อก ซึ่งจะมีความคล้ายกับงานวิจัยที่ได้นำเสนอนี้ตรงที่เป็นการใช้ประโยชน์จากคุณสมบัติของอุปกรณ์ต่างๆ เช่นกัน ในขั้นการศึกษานั้นได้มีการศึกษาหาคุณสมบัติของอุปกรณ์ ทั้งในส่วนที่คล้ายกันและแตกต่างกัน จากนั้นนำมาแยกหมวดหมู่ โดยพิจารณาถึงความสัมพันธ์ในทางกายภาพระหว่างอุปกรณ์หรือเครื่องจักรต่างๆ สามารถกำหนดให้มีความสัมพันธ์กันเฉพาะอุปกรณ์ที่เชื่อมต่อกันโดยตรง ได้มาเป็นกฎเกณฑ์พื้นฐานของการแบ่งชนิดของอุปกรณ์พื้นฐานชนิดต่างๆ

สรุปผลการศึกษางานวิจัยที่เกี่ยวข้องนั้น เราสามารถมองได้เป็นสองประเด็น ประเด็นแรกคือการนำวิธีการโปรแกรมโดยใช้การสาธิตมาพัฒนาเครื่องมือที่สามารถโปรแกรมการทำงานจนทำให้เครื่องจักรสามารถทำงานได้ตามความต้องการนั้น ปัญหาที่จะต้องหาวิธีการแก้ไขคือ การจำลองการตอบสนองกลับมายังระบบนั้น จะต้องสามารถทำได้เช่นเดียวกับการตอบสนองจากอุปกรณ์จริงที่มีคุณสมบัติและรูปแบบของการตอบสนองกลับด้วยตัวของมันเอง เช่น เมื่อเรารินน้ำลงในแก้วระดับน้ำจะเพิ่มสูงขึ้น ซึ่งในงานวิจัยที่ได้นำเสนอนั้น คุณสมบัติในส่วนนี้คือคุณสมบัติที่เราจะต้องโปรแกรมให้กับแบบจำลองเอง ดังนั้นจึงจำเป็นต้องศึกษาหาคุณสมบัติของอุปกรณ์ ทั้งในส่วนที่คล้ายกันและแตกต่างกันจากนั้นนำมาแยกหมวดหมู่ เพื่อนำมาสร้างอุปกรณ์บนแบบจำลองที่สามารถทำงานได้ในลักษณะเดียวกันกับอุปกรณ์จริง ประเด็นที่สองคือการศึกษาการแปลงจากการกระทำของผู้ใช้ที่มีต่ออุปกรณ์ต่างๆ มาสร้างเป็นชุดของคำสั่งในรูปแบบของนิพจน์คำสั่งเกี่ยวกับการเลือกทำงาน if…..then……..else และภาษานีโมนิคที่สามารถแปลงสู่คำสั่งที่เป็นลักษณะไบนารีโค้ดที่สามารถนำเข้าไปในอุปกรณ์พีแอลซีได้ทันที

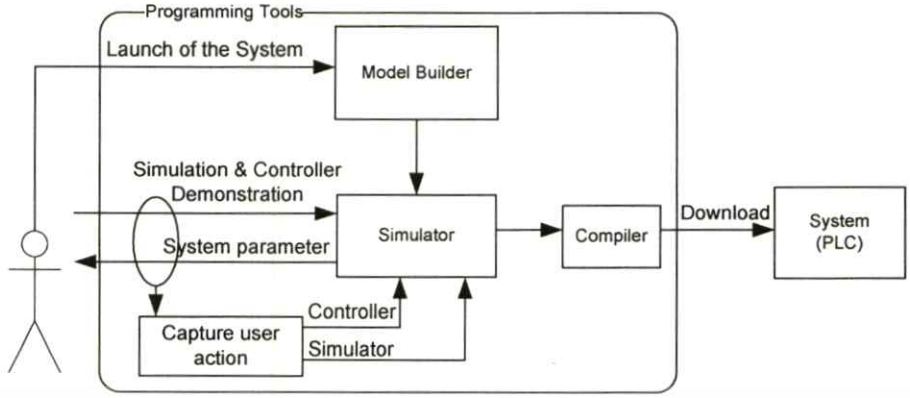
บทที่ 3
วิธีการโปรแกรมโดยใช้การสาธิตสำหรับอุปกรณ์พีแอลซี

เนื่องจากการประยุกต์ใช้หลักการดังกล่าวในการโปรแกรมอุปกรณ์พีแอลซีโดยตรงจำเป็นต้องอาศัยระบบหรือเครื่องจักรจริงหรือแบบจำลองระบบที่สมบูรณ์ ซึ่งอาจไม่สามารถจัดหาได้ด้วยข้อจำกัดด้านเวลา และค่าใช้จ่าย ในงานวิจัยนี้ได้นำเสนอรูปแบบการโปรแกรมโดยใช้การสาธิตซึ่งส่วนของอุปกรณ์ระบบควบคุมและแบบจำลองการทำงานของระบบที่ถูกควบคุมจะถูกโปรแกรมไปด้วยกันในลักษณะสลับไปมา ทำให้ไม่มีความจำเป็นต้องอาศัยระบบจริงหรือแบบจำลองระบบที่สมบูรณ์แล้วในการประยุกต์ใช้หลักการโปรแกรมดังกล่าว นอกเหนือจากการนำเสนอชุดแบบจำลองอุปกรณ์พื้นฐานเพื่อการสร้างแบบจำลองเริ่มต้นแล้ว

ในบทที่ 3 นี้ได้นำเสนอรูปแบบการโปรแกรมโดยใช้การสาธิตซึ่งส่วนของอุปกรณ์ระบบควบคุมและแบบจำลองการทำงานของระบบที่ถูกควบคุมจะถูกโปรแกรมไปด้วยกันในลักษณะสลับไปมา ทำให้ไม่มีความจำเป็นต้องอาศัยระบบจริงหรือแบบจำลองระบบที่สมบูรณ์แล้วในการประยุกต์ใช้หลักการโปรแกรมดังกล่าว นอกเหนือจากการนำเสนอชุดแบบจำลองอุปกรณ์พื้นฐานเพื่อการสร้างแบบจำลองเริ่มต้นแล้ว งานวิจัยนี้ได้นำเสนอวิธีการในการแปลการสาธิตของผู้ใช้ไปเป็นโปรแกรมในภาษาแบบโครงสร้างและภาษานีโมนิค เพื่อนำไปควบคุมอุปกรณ์พีแอลซีจริงต่อไป นอกจากนี้ในงานวิจัยนี้ได้กล่าวถึงปัญหาที่เกิดขึ้นจากการโปรแกรมโดยใช้การสาธิตในส่วนที่เกี่ยวข้องกับสถานการณ์ที่ยังไม่พิจารณา และได้นำเสนอแนวทางในการแก้ปัญหาดังกล่าวโดยใช้การจำลองสถานการณ์ในกรณี ต่าง ๆ ดังกล่าวให้ผู้ใช้พิจารณา

3.1 การประยุกต์ใช้วิธีการโปรแกรมโดยใช้การสาธิตสำหรับอุปกรณ์พีแอลซี

วิธีการโปรแกรมโดยใช้การสาธิตตัวควบคุมและแบบจำลองร่วมกันสำหรับอุปกรณ์พีแอลซี เทคนิคนี้เป็นการพัฒนาและผสมผสานกันระหว่างเทคนิคการเขียนโปรแกรมที่เน้นให้มีการใช้ภาพประกอบกับเทคนิควิธีการโปรแกรมโดยใช้การสาธิตโดยปรับเปลี่ยนการสาธิตระบบจริงมาเป็นการสาธิตกับแบบจำลองแทน ซึ่งขั้นตอนการสร้างแบบจำลองนั้นจะใช้เทคนิคการเขียนโปรแกรมที่เน้นให้มีการใช้ภาพประกอบ เมื่อสร้างแบบจำลองเสร็จเรียบร้อยแล้วจะนำแบบจำลองนั้นมาโปรแกรมโดยใช้เทคนิควิธีการโปรแกรมโดยใช้การสาธิต จุดสำคัญคือการจำลองเครื่องจักรขึ้นมา แล้วให้ผู้ใช้ทำการควบคุมเครื่องจักรด้วยมือ (Manual Control) จากนั้นระบบจะทำการเรียนรู้วิธีการควบคุมดังกล่าว เพื่อมาสร้างเป็นชุดคำสั่งและสามารถทำงานได้เหมือนการควบคุมโดยมนุษย์ (Human Control)



รูปที่ 3.1 แสดงแผนผังวิธีวิธีการโปรแกรมโดยใช้การสาธิตตัวควบคุมและแบบจำลองร่วมกัน สำหรับอุปกรณ์พีแอลซี

จากรูปที่ 3.1 วิธีการโปรแกรมโดยใช้การสาธิตตัวควบคุมและแบบจำลองร่วมกันสำหรับอุปกรณ์พีแอลซีที่นำเสนอนี้ สามารถแบ่งออกได้เป็นสี่ส่วนคือ

1) Model builder : ใช้สร้างแบบจำลอง และกำหนดความสัมพันธ์ระหว่างอุปกรณ์ควบคุมและอุปกรณ์ที่ถูกควบคุมโดยใช้การไกด์ไลน์ ซึ่งในส่วนนี้จะอาศัย Visual Programming Technique สำหรับการสร้างแบบจำลอง

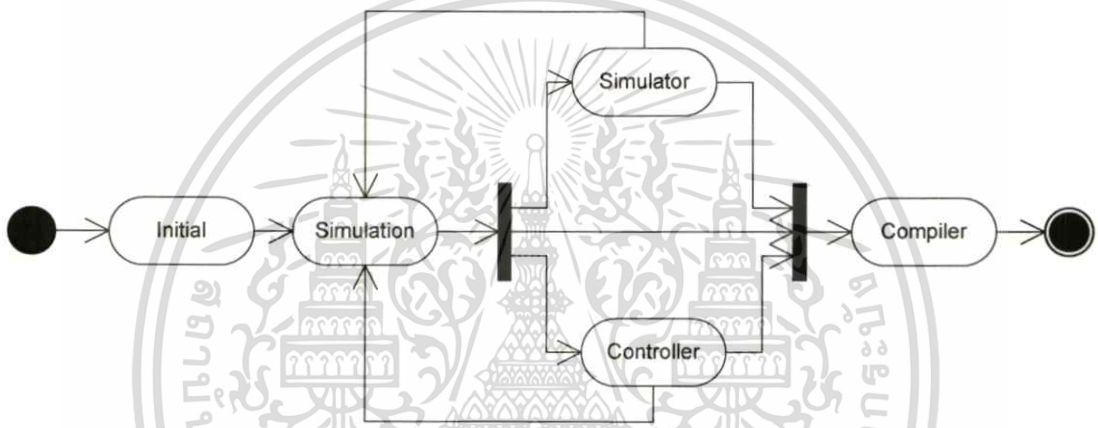
2) System simulator : ใช้แสดงการเปลี่ยนแปลงลักษณะทางกายภาพของอุปกรณ์โดยการแสดงผ่านทางภาพที่แสดงถึงการเคลื่อนไหวของระบบที่มีผลจากการกระตุ้นจากอุปกรณ์อื่นๆ หรือตอบสนองต่อระบบการควบคุม

3) User action recorder : ส่วนนี้ทำหน้าที่บันทึกการกระทำของผู้ใช้ ทั้งวัตถุประสงค์ของการโปรแกรมแบบจำลอง (Simulator programming) เพื่อกำหนดการเปลี่ยนแปลงลักษณะทางกายภาพของอุปกรณ์โดยการแสดงผ่านทางภาพ และการโปรแกรมตัวควบคุม (Controller programming) เพื่อให้ระบบเลือกทำตามเงื่อนไขในแต่ละสถานการณ์ที่เกิดขึ้น

4) Compiler : ทำหน้าที่แปลงบันทึกการกระทำของผู้ใช้ในส่วนของระบบการควบคุมไปสู่ภาษาที่ใช้ในการโปรแกรมอุปกรณ์พีแอลซี ซึ่งจะสามารถเป็นภาษาระดับสูง เช่น ภาษา Structure Text และ ภาษาระดับต่ำ เช่น ภาษา Instruction List หรือภาษา Mnemonic

วงจรวิธีการโปรแกรมโดยใช้การสาธิตตัวควบคุมและแบบจำลองร่วมกันสำหรับอุปกรณ์พีแอลซีนั้น จะเริ่มต้นจากขั้นตอนการสร้างแบบจำลอง (System initialization mode) เมื่อสร้างแบบจำลองเสร็จแล้วเราจะนำแบบจำลองนั้นเข้าสู่ขั้นตอนการจำลองสถานการณ์(Simulation mode) เมื่อเกิดการเปลี่ยนสถานะของอุปกรณ์ในแบบจำลองและยังไม่ได้มีการกำหนดการโปรแกรมเอาไว้

รวมทั้งผู้ใช้สามารถเลือกที่จะเข้าสู่ขั้นตอนการโปรแกรมได้เอง ขั้นตอนของการโปรแกรมนั้นแบ่งออกเป็นสองส่วน ส่วนแรกคือการโปรแกรมแบบจำลอง (Simulator programming mode) ในส่วนนี้เป็นการโปรแกรมคุณสมบัติและการทำงานของระบบที่ถูกจำลองขึ้น ส่วนที่สองคือโปรแกรมตัวควบคุม (Controller programming mode) ในส่วนนี้เป็นการโปรแกรมเพื่อให้ระบบทำตามเงื่อนไขการทำงานในแต่ละสถานการณ์ที่เกิดขึ้น ซึ่งในสองส่วนนี้จะทำสลับกันไปมาจนมีการโปรแกรมครบทุกเหตุการณ์ที่จะเกิดกับระบบที่ถูกจำลองขึ้น เมื่อเราทำการโปรแกรมเสร็จเรียบร้อยจากนั้นขั้นตอนสุดท้ายจะเป็นขั้นตอนของการแปลงโปรแกรมตัวควบคุม (Controller programming mode) ไปสู่ภาษาที่ใช้ในการโปรแกรมอุปกรณ์พีแอลซี (Compiler mode) โดยมีวงจรดังรูปที่ 3.2



รูปที่ 3.2 แสดงผังวงจรวิธีวิธีการโปรแกรมโดยใช้การสาธิตตัวควบคุมและแบบจำลองร่วมกันสำหรับอุปกรณ์พีแอลซี

ผังวงจรการทำงานของการประยุกต์ใช้วิธีการโปรแกรมโดยใช้การสาธิตตัวควบคุมและแบบจำลองร่วมกันสำหรับอุปกรณ์พีแอลซีที่นำเสนอสามารถแบ่งออกได้เป็นสี่สถานะคือ

1) System initialization mode : การสร้างแบบจำลอง และกำหนดความสัมพันธ์ระหว่างอุปกรณ์ควบคุม และอุปกรณ์ที่ถูกควบคุมโดยใช้การไกด์ไลน์ โดยที่การสร้างแบบจำลองนั้นเป็นขั้นตอนการเตรียมการ ก่อนอื่นเราต้องแยกแยะเครื่องจักรออกเป็นอุปกรณ์ย่อยๆ ก่อน เราจะเห็นว่าองค์ประกอบของเครื่องจักรนั้นจะประกอบไปด้วยอุปกรณ์ชนิดต่างๆ รวมกัน โดยที่อุปกรณ์แต่ละตัวจะมีจุดประสงค์ในการใช้งานที่แตกต่างกัน ดังนั้นเราจะต้องพิจารณาอุปกรณ์แต่ละตัวจากจุดประสงค์ในการใช้งาน เพื่อนำมาเทียบกับตัวอุปกรณ์พื้นฐานที่จะถูกนำมาใช้แทนอุปกรณ์ตัวนั้นๆ ซึ่งจะกล่าวโดยละเอียดต่อไป

2) Simulation mode : การจำลองสถานการณ์ เป็นการแสดงการทำงานของแบบจำลองตามฐานความรู้ที่ถูกโปรแกรมในกรณีต่างๆ ไว้แล้ว ทั้งในส่วนของการโปรแกรมแบบจำลอง และการโปรแกรมตัวควบคุม คือเมื่อเกิดเหตุการณ์หนึ่งขึ้นกับระบบ ระบบจะทำการเปลี่ยนแปลงคุณสมบัติของอุปกรณ์ที่ถูกควบคุมอย่างไร หรือจะทำการควบคุมสถานะของอุปกรณ์ที่ถูกควบคุมอย่างไร อย่างไรก็ตามหากในกรณีดังกล่าวยังไม่ได้ถูกการโปรแกรมทั้ง 2 ส่วนมาก่อน ระบบจะถามผู้ใช้เพื่อให้ผู้ใช้เลือกที่จะโปรแกรมการทำงานเพิ่มเติมได้

3) Simulator programming mode : การโปรแกรมแบบจำลอง เพื่อกำหนดการเปลี่ยนแปลงลักษณะทางกายภาพของอุปกรณ์โดยการแสดงผ่านทางภาพที่แสดงถึงการเคลื่อนไหวของระบบที่มีผลจากการกระตุ้นจากอุปกรณ์อื่นๆ หรือตอบสนองต่อระบบการควบคุม โดยหลักสำคัญของการโปรแกรมในส่วนนี้ที่จะพิจารณาคือ การกระตุ้นต้องมิได้เกิดจากการกระตุ้นทางไฟฟ้า เช่น ปั๊มน้ำเปิด ส่งผลให้ระดับน้ำที่อยู่ในถังเพิ่มขึ้น หรือมอเตอร์หมุนทำให้สายพานเคลื่อนที่

4) Controller programming mode : การโปรแกรมตัวควบคุมเพื่อให้ระบบเลือกทำตามเงื่อนในแต่ละสถานการณ์ที่เกิดขึ้น โดยผู้ใช้จะแจกแจงการควบคุมเพื่อตอบสนองสถานะต่างๆ ของระบบที่เกิดขึ้น โดยหลักสำคัญของการโปรแกรมในส่วนนี้ที่จะพิจารณาในทางกลับกันกับการโปรแกรมแบบจำลองคือ การกระตุ้นต้องเกิดจากการกระตุ้นทางไฟฟ้า หรือส่งผลทางไฟฟ้าต่ออีกอุปกรณ์หนึ่ง เช่น เปิดสวิทช์ไฟฟ้าทำให้หลอดไฟฟ้าเปิด หรือเมื่อลูกลอยวัดระดับน้ำมีสถานะเป็นเปิดปั๊มน้ำจะปิด

5) Compiler mode : การแปลงบันทึกการกระทำของผู้ใช้ในส่วนของระบบการควบคุมไปสู่ภาษาที่ใช้ในการโปรแกรมอุปกรณ์พีแอลซี ซึ่งเป็นภาษาระดับสูง เช่น ภาษา Structure Text และ ภาษาระดับต่ำ เช่น ภาษา Mnemonic หรือภาษา Instruction List ตามข้อกำหนดของมาตรฐาน IEC1131-3 ซึ่งเป็นมาตรฐานของภาษาที่ใช้ในการเขียนโปรแกรมอุปกรณ์พีแอลซี

3.2 ขั้นตอนการประยุกต์ใช้วิธีการโปรแกรมโดยใช้การสาธิตมาใช้กับแบบจำลองที่ไม่สมบูรณ์

3.2.1 ขั้นตอนการสร้างแบบจำลองระบบ

ขั้นตอนการสร้างแบบจำลองระบบ หมายถึง การแทนที่อุปกรณ์ต่างๆ ของเครื่องจักรจากวัตถุ (Object) ที่แทนด้วยข้อมูลทางคอมพิวเตอร์ที่สามารถแสดงการเปลี่ยนแปลงค่าของข้อมูลของคอมพิวเตอร์ที่สามารถทำให้ผู้ใช้สังเกตเห็นการเปลี่ยนแปลงที่เกิดขึ้นจากแบบจำลองคอมพิวเตอร์ทำให้สามารถตีความกลับไปเป็นพฤติกรรมที่เกิดขึ้นกับระบบจริงได้ การพัฒนาแบบจำลองระบบต้องเตรียมการหลายด้านเพื่อให้ได้แบบจำลองตามจุดประสงค์ และอีกส่วนที่สำคัญคือการสร้างความความสัมพันธ์ระหว่างวัตถุต่างๆ ที่ประกอบเป็นเครื่องจักร โดยเราสามารถแบ่งออกเป็น

1) Based class หมายถึง แม่แบบพื้นฐาน ซึ่งเป็นคลาสที่มีคลาสอื่นนำไปใช้สืบทอดคุณสมบัติ เพื่อใช้เป็นแม่แบบให้กับอุปกรณ์ต่างๆ

2) Relationship หมายถึง ความสัมพันธ์ระหว่างอุปกรณ์ ซึ่งจะเป็นตัวบ่งบอกถึงการที่สถานะหรือค่าตัวแปรของอุปกรณ์ตัวหนึ่งส่งผลต่อสถานะหรือค่าตัวแปรของอุปกรณ์อีกตัวหนึ่ง

3) Simulator federate หมายถึงการจับกลุ่มหรือรวมกลุ่มกันของวัตถุ หรือการที่วัตถุไปเป็นส่วนประกอบของอีกวัตถุหนึ่ง (Dependency) ซึ่งแต่ละ federate จะควบคุมวัตถุภายในแบบจำลอง รวมทั้งส่งและรับการแก้ไขข้อมูล (update) และการโต้ตอบ (interaction) ที่เกี่ยวข้องกับวัตถุของ federate นั้นๆ และของ federate อื่นๆ

3.2.2 ขั้นตอนกำหนดกระบวนการ หรือสถานการณ์

ขั้นตอนกำหนดกระบวนการ หรือสถานการณ์นั้น ได้กล่าวไปแล้วข้างต้นว่าการประยุกต์ใช้ PBD มาใช้กับแบบจำลองที่ไม่สมบูรณ์จะต้องเพิ่มขั้นตอนการโปรแกรมให้แบบจำลองที่ไม่สมบูรณ์สามารถโต้ตอบกับผู้ใช้ได้เหมือนกับอุปกรณ์จริง ดังนั้นการประยุกต์ใช้วิธีการโปรแกรมโดยใช้การสาธิตมาใช้กับแบบจำลองที่ไม่สมบูรณ์ จึงจำเป็นต้องมีการโปรแกรมทั้งในส่วนของการโปรแกรมแบบจำลอง ซึ่งเป็นการโปรแกรมคุณสมบัติและการทำงานของระบบที่ถูกจำลองขึ้น และโปรแกรมตัวควบคุม ซึ่งเป็นการโปรแกรมเพื่อให้ระบบทำตามเงื่อนไขการทำงานในแต่ละสถานการณ์ที่เกิดขึ้น ดังนั้นเราสามารถแบ่งขั้นตอนนี้ได้เป็นสองขั้นตอนย่อยดังนี้

1) ขั้นตอนการสลับไปมาระหว่างการโปรแกรมแบบจำลอง และการโปรแกรมตัวควบคุม

วิธีการโปรแกรมโดยใช้การสาธิตนั้น จำเป็นต้องอาศัยข้อมูล System Simulation Model ซึ่งตรงกับ module 'simulator' ที่ทำหน้าที่สร้างจำลองแบบเพื่อเลียนแบบพฤติกรรมของเครื่องจักร ซึ่งทำให้ในทางปฏิบัติมีข้อยุ่งยากในการพัฒนาแบบจำลองการทำงาน และความสัมพันธ์ในทางกายภาพระหว่างวัตถุที่ประกอบขึ้นเป็นเครื่องจักรก่อนจึงจะสามารถใช้วิธีการโปรแกรมโดยใช้การสาธิตได้

2) ขั้นตอนการจัดการกรณีเหตุการณ์ที่ไม่ได้ถูกโปรแกรม (Missing case)

การโปรแกรมโดยใช้แบบจำลองที่ไม่ได้ทำสำเร็จนั้นจะทำให้เกิดกรณีเหตุการณ์ที่ไม่ได้ถูกโปรแกรมได้ง่ายกว่าการใช้แบบจำลองที่ทำสำเร็จ ดังนั้นการจัดการกับเหตุการณ์ที่ไม่ได้ถูกโปรแกรมจึงเป็นส่วนสำคัญในการขจัดความผิดพลาดของการโปรแกรมให้หมดไป การเกิดเหตุการณ์ในลักษณะนี้อาจส่งผลต่อการทำงานของเครื่องจักรหรืออาจไม่ส่งผลใดๆ ต่อการทำงานของเครื่องจักรเลยก็ได้ ขึ้นอยู่กับว่าเหตุการณ์ดังกล่าวเกิดขึ้นในขณะเครื่องจักรทำงานหรือไม่ เพราะบางเหตุการณ์อาจไม่เกิดขึ้นเลยก็ได้จากกฎของธรรมชาติ

3.2.3 ขั้นตอนการแปลงการกระทำของผู้ใช้ไปสู่ภาษาของอุปกรณ์พีแอลซี

เมื่อแบบจำลองสามารถทำงานได้อย่างถูกต้องแล้วขั้นตอนต่อไปคือการแปลง User Action สู่ภาษาของอุปกรณ์พีแอลซี ภาษาที่ใช้ในการเขียนโปรแกรมตามมาตรฐาน IEC 1131-3 ซึ่งในวิทยานิพนธ์ฉบับนี้ได้นำเสนอในสองภาษา ภาษาแรกคือภาษาแบบโครงสร้าง (Structure Text) นั้นเป็นภาษาในระดับสูง โดยมีพื้นฐานมาจากการเขียนโปรแกรมในลักษณะของการโปรแกรมแบบมีโครงสร้าง (Structure Programming) โดยภาษาในลักษณะนี้มีหลักการสามแบบคือ การทำงานแบบตามลำดับ (Sequence) การเลือกกระทำตามเงื่อนไข (Condition) และการทำซ้ำ (Loop) ภาษาที่สองคือภาษานีโมนิค (Mnemonic) จะเป็นภาษาที่เขียนอยู่ในรูปของข้อความ และมีลักษณะคล้ายกับภาษาแอสเซมบลี (Assembly) และภาษาเครื่อง (Machine code) ซึ่งภายในหนึ่งคำสั่งควบคุมจะประกอบด้วย ส่วนปฏิบัติการ (Operator) และส่วนที่ถูกดำเนินการ (Operand) การแปลง User action สู่ภาษานีโมนิคนั้นจะต้องแปลง User action เสียก่อนจากนั้นอาศัยหลักการของภาษาแบบโครงสร้างแปลงสู่ภาษานีโมนิคอีกครั้ง

3.3 การสร้างแบบจำลองระบบ (Model building)

3.3.1 แม่แบบพื้นฐาน (Based class)

แม่แบบพื้นฐาน ใช้ในการสืบทอดคุณสมบัติให้กับอุปกรณ์ต่างๆ แนวคิดการสร้างอุปกรณ์พื้นฐาน โดยพิจารณาถึงคุณสมบัติพื้นฐานของวัตถุ หรืออุปกรณ์ที่ประกอบขึ้นเป็นเครื่องจักรนั้น ว่าแต่ละวัตถุมีส่วนใดที่เหมือนกันและต่างกันบ้าง รวมถึงการพิจารณาความสัมพันธ์ในทางกายภาพระหว่างอุปกรณ์ต่างๆ ของเครื่องจักร เพื่อให้เราสามารถสร้างอุปกรณ์โดยกำหนดให้อุปกรณ์พื้นฐานเป็นแม่แบบของอุปกรณ์ต่างๆ โดยที่วัตถุต่างๆ สามารถที่จะแสดงการเปลี่ยนแปลงค่าของข้อมูลของคอมพิวเตอร์ที่สามารถทำให้ผู้ใช้สังเกตเห็นการเปลี่ยนแปลงที่เกิดขึ้นจากแบบจำลองคอมพิวเตอร์ ทำให้สามารถตีความกลับไปเป็นพฤติกรรมที่เกิดขึ้นกับระบบจริงได้ รวมทั้งแสดงถึงความสัมพันธ์กันระหว่างวัตถุได้อีกด้วย จากหลักการที่กล่าวมาแล้วนั้น เรานำมากำหนดเป็นกฎเกณฑ์พื้นฐานของการแบ่งชนิดของอุปกรณ์พื้นฐานชนิดต่างๆ ในวิทยานิพนธ์ฉบับนี้ยังได้คำนึงถึงกฎเกณฑ์ตามชนิดของอุปกรณ์ที่มีอยู่ในอุปกรณ์พีแอลซีเป็นหลัก ซึ่งในอุปกรณ์พีแอลซีนั้นจะแบ่งอุปกรณ์ออกเป็นสี่ชนิดหลัก คือ รีเลย์ ตัวจับเวลา ตัวนับจำนวน และตัวจดจำข้อมูล และเพิ่มเติมในส่วนของอุปกรณ์หีบห่อขึ้นมา พร้อมทั้งกำหนดคุณลักษณะเพิ่มเติม เพื่อให้ครอบคลุมลักษณะทางกายภาพของอุปกรณ์ต่างๆ ในการแสดงให้เห็นถึงการเปลี่ยนแปลงลักษณะรูปร่างของอุปกรณ์ เพื่อการสื่อสารกับผู้ใช้ในรูปแบบของภาพเคลื่อนไหวที่มีความใกล้เคียงกับอุปกรณ์จริงมากที่สุด

การออกแบบส่วนการสร้างแบบจำลองนั้น จะต้องอาศัยการแสดงภาพของเครื่องจักร ที่ประกอบด้วยชิ้นส่วนต่างๆ โดยแต่ละชิ้นส่วนนั้นหมายถึงอุปกรณ์หนึ่งตัวนั่นเอง และจะต้องมีการแสดงภาพที่สามารถทำให้ผู้ใช้สังเกตเห็นการเปลี่ยนแปลงที่เกิดขึ้นกับอุปกรณ์ได้ รวมถึงคุณลักษณะเฉพาะของอุปกรณ์ตัวนั้นๆ ฉะนั้นการสร้างอุปกรณ์ในวิธีการที่นำเสนอนี้ เราจะคำนึงถึงสองส่วนคือ ส่วนแรกการแสดงสถานะต่างๆ ของอุปกรณ์โดยใช้ภาพประกอบ ส่วนที่สองคือคุณสมบัติของอุปกรณ์ โดยการกำหนดอุปกรณ์พื้นฐานนั้นยึดประเภทของอุปกรณ์ในพีแอลซีดังนี้

1) อุปกรณ์ทั่วไป (General)

อุปกรณ์ทั่วไป ใช้แทนวัตถุหรืออุปกรณ์ที่เป็นส่วนประกอบของเครื่องจักร ในส่วนของอุปกรณ์ที่เป็นกลไก อุปกรณ์ติดต่อกับผู้ใช้หรืออุปกรณ์แสดงผลต่างๆ โดยอุปกรณ์ทั่วไปนี้เราจะแบ่งส่วนหลักๆ เป็นสองส่วนคือ ส่วนแรกคือส่วนสถานะ (State) ซึ่งส่งผลต่อการควบคุมหรือมีผลต่อเงื่อนไขการทำงานของอุปกรณ์ โดยแบ่งออกได้เป็นสองสถานะคือปิดและเปิด หรือ “0” และ “1” ส่วนที่สองคือข้อมูลค่าการเปลี่ยนแปลงลักษณะทางกายภาพของวัตถุ ซึ่งเป็นค่าตัวแปรที่จะส่งผลต่อการเปลี่ยนแปลงรูปภาพที่ใช้แสดงแทนตัวอุปกรณ์

2) อุปกรณ์หีบห่อ (Package)

ใช้แสดงโครงสร้างของเครื่องจักร หรืออุปกรณ์ที่สามารถบรรจุอุปกรณ์ชนิดอื่นลงไปได้ โดยที่อุปกรณ์ที่นำมาบรรจุในอุปกรณ์ชนิดหีบห่อนั้นจะมีคุณสมบัติเป็นส่วนหนึ่ง (Part of) ของอุปกรณ์ชนิดหีบห่อทันที เช่น ใช้แทนถังน้ำที่สามารถบรรจุอุปกรณ์วัดระดับน้ำ อุณหภูมิ ความดัน ให้เป็นส่วนหนึ่งของถังน้ำได้

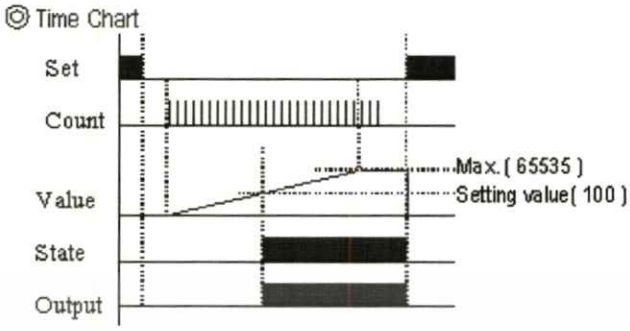
3) อุปกรณ์ตรวจจับ (Sensor)

ใช้ร่วมกับอุปกรณ์ชนิดทั่วไป โดยมีคุณสมบัติในการตรวจสอบค่าตัวแปรของอุปกรณ์ชนิดทั่วไป เพื่อนำมาเปรียบเทียบกับค่าที่กำหนดไว้ (Setting value) ถ้าผลการเปรียบเทียบเป็นไปตามเงื่อนไขที่กำหนด อุปกรณ์ชนิดตรวจจับจะให้ผลลัพธ์เป็นสถานะของตัวอุปกรณ์ (State) เป็น “1” ถ้าหากไม่เป็นไปตามเงื่อนไขที่กำหนดอุปกรณ์ชนิดตรวจจับจะให้ผลลัพธ์เป็น “0”

4) อุปกรณ์นับจำนวน (Counter)

ใช้ในการนับจำนวนโดยใช้ร่วมกับอุปกรณ์ชนิดทั่วไปหรืออุปกรณ์ชนิดตรวจจับ ซึ่งอุปกรณ์ชนิดนับจำนวนจะประกอบไปด้วยค่าต่างๆ ดังนี้ ค่าสถานะ (State) สถานะของการนับจำนวน (Count) ค่าจำนวนนับ (Value) ค่าจำนวนนับที่กำหนด (Setting value) และสภาวะพร้อมที่จะทำงาน (Set) โดยหากสภาวะพร้อมที่จะทำงานเป็น “1” จะทำให้จำนวนนับและค่าสถานะเป็น “0” อุปกรณ์ชนิดนับจำนวนจะเริ่มนับจำนวนต่อเมื่อค่าสภาวะพร้อมที่จะทำงานเป็น “0” จะนับเพิ่ม

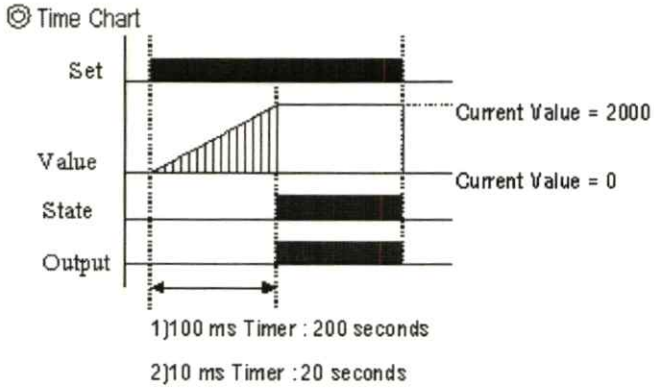
ทุกครั้งที่สถานะของการนับจำนวนเป็น “1” และจะนับเพิ่มขึ้นเรื่อยๆ เมื่อถึงค่าที่กำหนดอุปกรณ์ชนิดนับจำนวนจะให้ผลลัพธ์เป็น “1” ดังแสดงในรูปที่ 3.3



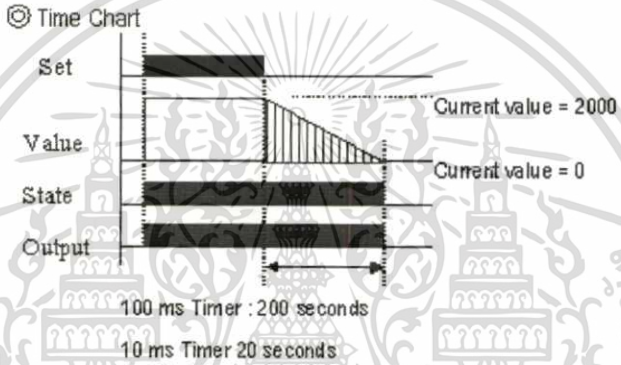
รูปที่ 3.3 แสดงสถานะการทำงานของอุปกรณ์ชนิดนับจำนวน

5) อุปกรณ์จับเวลา (Timer)

ใช้ในการจับเวลาโดยใช้ร่วมกับอุปกรณ์ชนิดทั่วไป หรืออุปกรณ์ชนิดตรวจจับ ซึ่งอุปกรณ์ชนิดจับเวลาจะประกอบไปด้วยค่าต่างๆ ดังนี้ ค่าสถานะ (State) ค่าจำนวนเวลา (Value) ค่าจำนวนเวลาที่กำหนด (Setting value) และสภาวะพร้อมที่จะทำงาน (Set) อุปกรณ์ชนิดจับเวลาจะแบ่งออกเป็นสองชนิดคือ ตั้งเวลาเปิดและตั้งเวลาปิด โดยที่ชนิดตั้งเวลาเปิด (Time On) เมื่อสภาวะพร้อมที่จะทำงานเป็น “0” จะทำให้จำนวนเวลาและค่าสถานะเป็น “0” อุปกรณ์ชนิดจับเวลาจะเริ่มเพิ่มเวลาต่อเมื่อค่าสภาวะพร้อมที่จะทำงานเป็น “1” และจะเพิ่มขึ้นเรื่อยๆ เมื่อถึงค่าที่กำหนด อุปกรณ์ชนิดจับเวลาจะให้ผลลัพธ์เป็น “1” ส่วนชนิดตั้งเวลาปิด (Time Off) หากสภาวะพร้อมที่จะทำงานเป็น “1” จะทำให้จำนวนเวลาเท่ากับค่าเวลาที่กำหนดและค่าสถานะเป็น “0” อุปกรณ์ชนิดจับเวลาจะเริ่มลดจำนวนต่อเมื่อค่าสภาวะพร้อมที่จะทำงานเป็น “0” และจะลดลงเรื่อยๆ จนเท่ากับ “0” อุปกรณ์ชนิดจับเวลาจะให้ผลลัพธ์เป็น “1” ดังแสดงในรูปที่ 3.4 และรูปที่ 3.5 ตามลำดับ



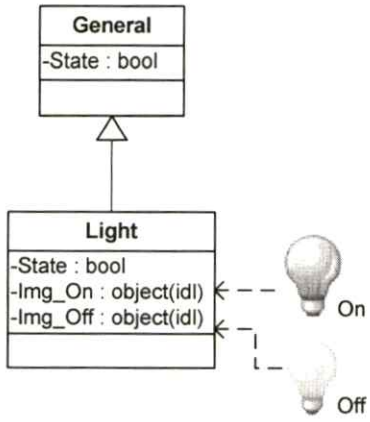
รูปที่ 3.4 แสดงสถานะการทำงานของอุปกรณ์ชนิดจับเวลาชนิดตั้งเวลาเปิด (Time On)



รูปที่ 3.5 แสดงสถานะการทำงานของอุปกรณ์ชนิดจับเวลาชนิดตั้งเวลาปิด (Time Off)

3.3.2 การสร้างอุปกรณ์จากแม่แบบพื้นฐาน

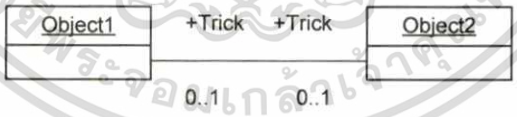
การสร้างอุปกรณ์จากแม่แบบพื้นฐาน คือการสืบทอดคุณสมบัติจากแม่แบบ (Class) เป็นวิธีการสร้างแม่แบบย่อย (Subclass) ซึ่งจะเป็นการกำหนดประเภทของวัตถุให้จำเพาะเจาะจงขึ้น แม่แบบย่อยจะได้รับการถ่ายทอดคุณสมบัติต่างๆ มาจากแม่แบบหลักด้วย จากหลักการดังกล่าวการสร้างวัตถุจากอุปกรณ์พื้นฐานหรือแม่แบบที่มีอยู่ทั้งห้าชนิดให้เป็นวัตถุ หรืออุปกรณ์ต่างๆ ได้ เช่น การสร้างหลอดไฟฟ้าจากแม่แบบอุปกรณ์ทั่วไป (General) โดยสิ่งที่เพิ่มเติมคือรูปภาพของหลอดไฟฟ้าในขณะที่เปิดและปิด



รูปที่ 3.6 แสดงการสร้างหลอดไฟจากอุปกรณ์พื้นฐาน (Inheritance)

3.3.3 ความสัมพันธ์ระหว่างวัตถุแบบเกี่ยวเนื่อง (Association relationship)

หมายถึง ความสัมพันธ์ระหว่างวัตถุแบบเกี่ยวเนื่อง จะทำให้วัตถุที่มีความสัมพันธ์กันสามารถส่งและรับการแก้ไขข้อมูล (Update) และการโต้ตอบ (Interaction) ระหว่างกันได้ หรืออาจกล่าวได้ว่าความสัมพันธ์ระหว่างวัตถุนั้นจะเป็นตัวบ่งบอกถึงการที่สถานะหรือค่าตัวแปรของอุปกรณ์ตัวหนึ่งส่งผลต่อสถานะหรือค่าตัวแปรของอุปกรณ์อีกตัวหนึ่ง เช่น ปั๊มน้ำเปิดส่งผลให้ระดับน้ำในถังเพิ่มขึ้น และอีกประการหนึ่งคือความความสัมพันธ์ระหว่างวัตถุแบบเกี่ยวเนื่องนี้สามารถเป็นความสัมพันธ์ระหว่างวัตถุหนึ่งกับอีกวัตถุหนึ่ง (One to One) หรือเป็นความสัมพันธ์ระหว่างวัตถุหนึ่งกับอีกหลายๆ วัตถุ (One to Many) ก็ได้

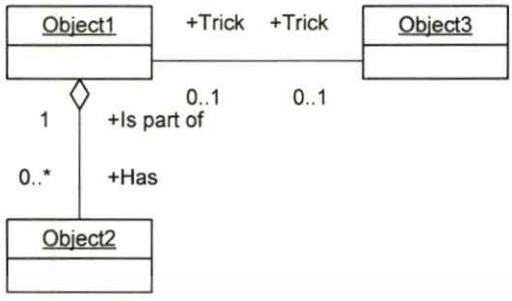


รูปที่ 3.7 แสดงความสัมพันธ์แบบ Association relationship

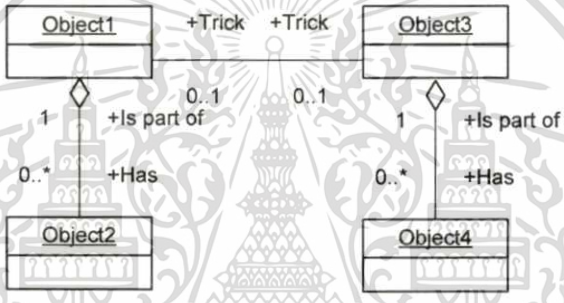
3.3.4 ความสัมพันธ์แบบอาศัย (Dependency relationship)

หมายถึง การจับกลุ่มหรือรวมกลุ่มกัน (Federate) ของวัตถุ หรือการที่วัตถุไปเป็นส่วนประกอบของอีกวัตถุหนึ่ง (Dependency) โดยที่ความสัมพันธ์แบบอาศัยนี้จะต้องมีวัตถุหลัก (Whole) หนึ่งตัวเสียก่อน จึงจะสามารถนำวัตถุอื่นๆ มาประกอบให้เป็นกลุ่มได้ ซึ่งแต่ละกลุ่มจะควบคุมวัตถุภายในแบบจำลอง รวมทั้งส่งและรับการแก้ไขข้อมูล (Update) และการโต้ตอบ (Interaction) ที่เกี่ยวข้องกับวัตถุของกลุ่มนั้น และของกลุ่มอื่นๆ อีกประการหนึ่งคือความสัมพันธ์

ข้อมูล (Update) และการโต้ตอบ (Interaction) จะมีผลต่อวัตถุภายในกลุ่มตนเอง และวัตถุภายในกลุ่มวัตถุอื่น



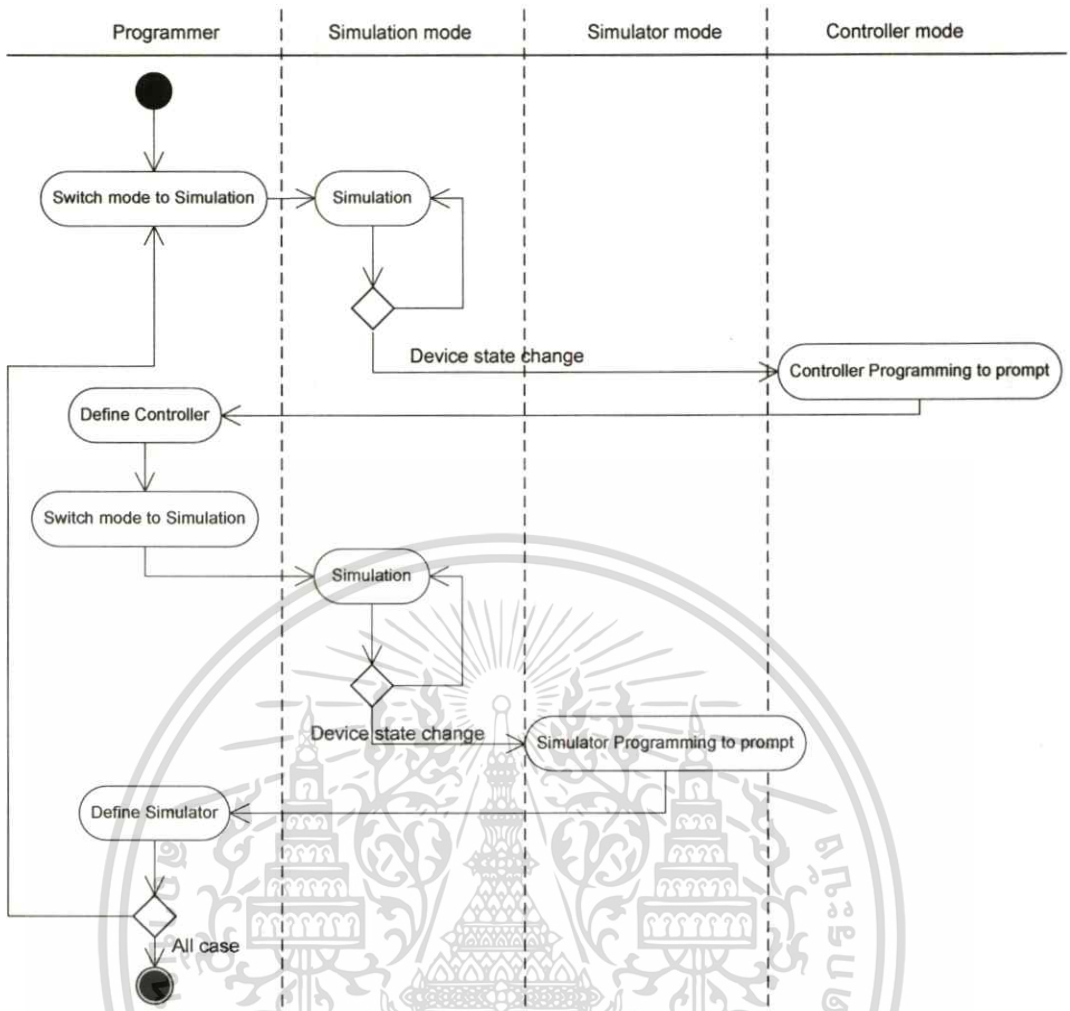
รูปที่ 3.10 แสดงความสัมพันธ์ร่วมระหว่างกลุ่ม Object1 และ Object2 กับ Object3



รูปที่ 3.11 แสดงความสัมพันธ์ร่วมระหว่างกลุ่ม Object1 และ Object2 กับกลุ่ม Object3 และ Object4

3.4 การสลับไปมาระหว่างการโปรแกรมตัวควบคุมและแบบจำลอง (Iterative Programming)

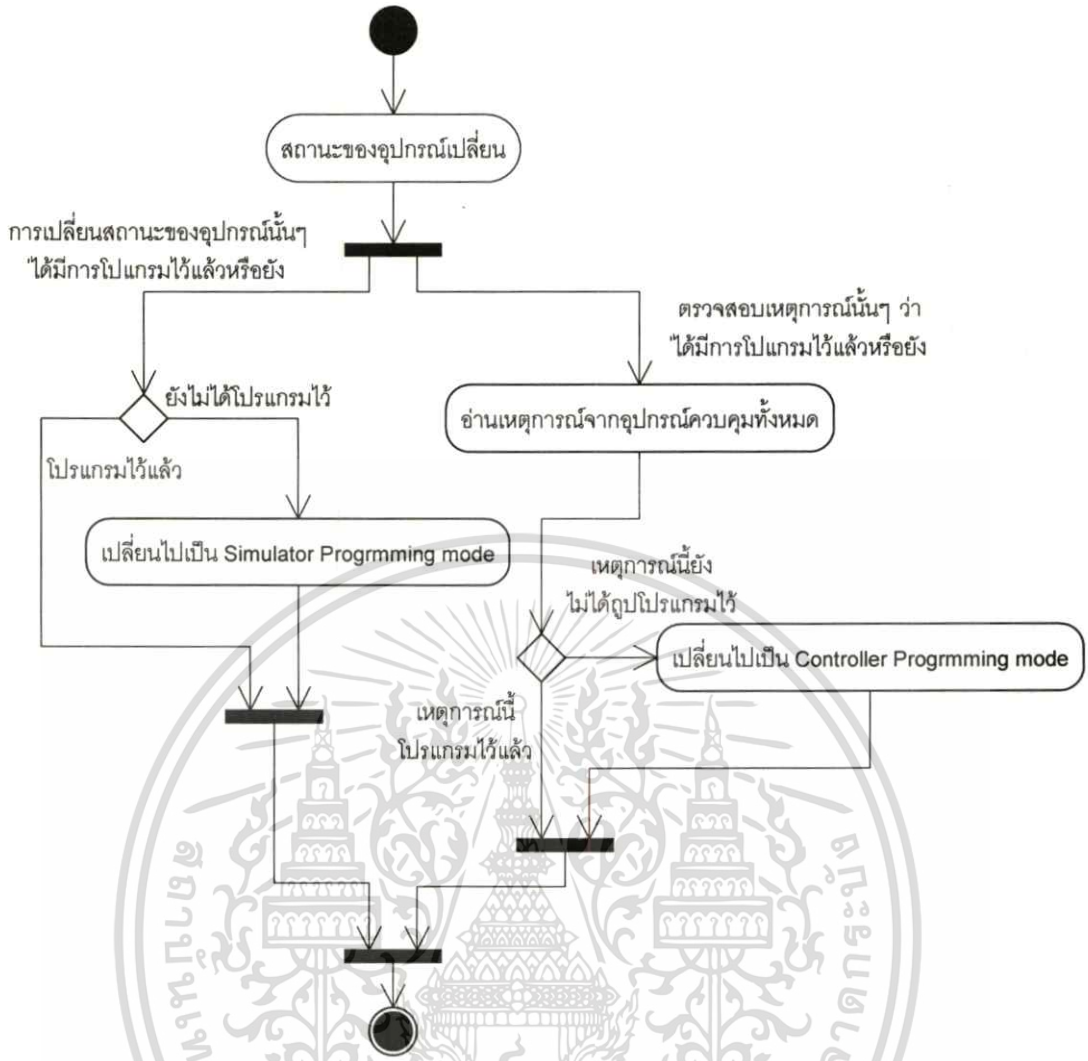
วิธีการสลับไปมาระหว่างการโปรแกรมตัวควบคุมและการโปรแกรมแบบจำลองนั้น จะมีขั้นตอนของการโปรแกรม ซึ่งจะเริ่มจากโหมดการจำลองสถานการณ์เพื่อให้ระบบแสดงการทำงานระบบจะคอยตรวจสอบสถานะของอุปกรณ์ต่างๆ แล้วนำมาวิเคราะห์ว่าจะต้องทำอย่างไรต่อไป หากมีการโปรแกรมไว้แล้วก็จะทำงานไปตามนั้น ส่วนเหตุการณ์ดังกล่าวยังไม่เคยมีการโปรแกรมไว้ ระบบจะวิเคราะห์ว่าการโปรแกรมที่ยังขาดไปนั้นสถานะของการโปรแกรมแบบจำลอง หรือสถานะของการโปรแกรมตัวควบคุม โดยระบบจะหยุดการทำงานแล้วสลับการทำงานไปยังสถานะของการโปรแกรมที่ต้องการให้ผู้ใช้โปรแกรมเพิ่มเติม ซึ่งแสดงแผนผังกิจกรรม (Activity Diagram) ของวิธีการสลับไปมาได้ดังรูปที่ 3.12



รูปที่ 3.12 แสดงแผนผังกิจกรรมของวิธีการสลับไปมา (Iterative Technique)

3.4.1 เงื่อนไขของการสลับไปมา (Condition for Iteration)

เงื่อนไขของการสลับไปมานั้น จะเกิดขึ้นจากการที่ระบบจะทำหน้าที่คอยตรวจสอบสถานะของอุปกรณ์ต่างๆ ว่ามีการเปลี่ยนแปลงหรือไม่ โดยที่การเปลี่ยนแปลงสถานะของอุปกรณ์นั้นเกิดขึ้นได้ในสองลักษณะคือ ลักษะแรกเป็นการเปลี่ยนแปลงที่เกิดจากการกระทำของผู้ใช้งาน ลักษณะที่สองเกิดขึ้นจากเงื่อนไขการควบคุมที่ได้ถูกกำหนดขึ้นโดยผู้ใช้ เมื่อเกิดการเปลี่ยนแปลงสถานะของอุปกรณ์ในแบบจำลอง ระบบจะมีการอ่านเหตุการณ์ที่เกิดขึ้น จากนั้นระบบจะนำมาวิเคราะห์เพื่อเลือกเปลี่ยนโหมดไปเพื่อรอรับการโปรแกรมจากผู้ใช้ในสองลักษณะคือ การโปรแกรมแบบจำลองและการโปรแกรมตัวควบคุม ดังแสดงแผนผังกิจกรรมได้ดังรูปที่ 3.13



รูปที่ 3.13 แสดงแผนผังกิจกรรมเงื่อนไขของการสลับไปมา (Condition for Iteration)

1) การโปรแกรมแบบจำลอง (Simulator Programming Mode)

ซึ่งจะเกิดขึ้นเมื่อมีการเปลี่ยนแปลงสถานะของอุปกรณ์ใดๆ ระบบจะตรวจสอบในสองเงื่อนไข เงื่อนไขแรกคือ อุปกรณ์ตัวที่ผู้ใช้เปลี่ยนสถานะนั้นมีคุณสมบัติที่สามารถส่งผลกระทบทางแรงกลต่ออุปกรณ์อื่น และเงื่อนไขที่สองคือสถานะที่เปลี่ยนไปนั้นยังไม่ได้ถูกกำหนดให้มีการตอบสนองเอาไว้ เมื่อเงื่อนไขครบทั้งสองเงื่อนไขดังนี้แล้ว สถานะของระบบจะเปลี่ยนไปเป็นโหมดโปรแกรมแบบจำลองเพื่อรอรับการกระทำของผู้ใช้ว่ามีการเปลี่ยนแปลงคุณสมบัติของตัวถูกควบคุมอย่างไร เพื่อนำมากำหนดเป็นเงื่อนไขการทำงานของแบบจำลองดังกล่าว

2) การโปรแกรมตัวควบคุม (Controller Programming Mode)

ซึ่งจะเกิดขึ้นเมื่อมีการเปลี่ยนแปลงสถานะของอุปกรณ์ใดๆ ระบบจะตรวจสอบเหตุการณ์โดยการพิจารณาถึงเหตุการณ์ที่สถานะของอุปกรณ์ทั้งหมดที่มีคุณสมบัติของการควบคุมทางไฟฟ้าที่สามารถส่งผลต่ออุปกรณ์ตัวอื่นๆ ได้ ซึ่งเราเรียกว่าอุปกรณ์ควบคุม (Controller Device)

ว่าเหตุการณ์ที่ประกอบไปด้วยสถานะของอุปกรณ์ควบคุมทุกตัวนั้น ได้มีการโปรแกรมตัวควบคุมแล้วหรือยัง หากยังไม่มีระบบจะสลับไปสู่โหมดการโปรแกรมตัวควบคุม (Controller Programming mode) เพื่อให้ผู้ใช้ทำการควบคุมที่เหมาะสมโดยการเปลี่ยนสถานะของอุปกรณ์ที่ถูกควบคุมต่างๆ ตามเงื่อนไขการทำงานของเครื่องจักร

3.4.2 ความเกี่ยวโยงระหว่างอุปกรณ์ควบคุมและอุปกรณ์ที่ถูกควบคุม (Involvement)

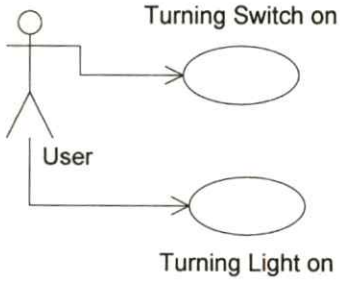
วิธีการที่นำเสนอนี้ จะมีรูปแบบของการโปรแกรมหรือเงื่อนไขของการทำงานที่เกิดจากการเปลี่ยนสถานะของอุปกรณ์ควบคุม ซึ่งรูปแบบของเงื่อนไขนั้นอาจเกิดจากการเปลี่ยนแปลงของอุปกรณ์ควบคุมเพียงตัวเดียว หรืออาจเกิดจากเหตุการณ์ที่อุปกรณ์ควบคุมมากกว่าหนึ่งตัวมีสถานะรวมกันเป็นไปตามเงื่อนไขที่ผู้โปรแกรมกำหนด การโปรแกรมในลักษณะนี้จะอยู่ในรูปแบบประโยคที่ว่า “ถ้าเกิดเหตุการณ์นี้ จะต้องกระทำดังนี้” หรือ “IF (Condition) THEN Statement END” นั่นเอง

ความเกี่ยวโยงระหว่างอุปกรณ์ควบคุมและอุปกรณ์ที่ถูกควบคุม คือการระบุความเป็นไปได้ของการเปลี่ยนสถานะของอุปกรณ์ที่ถูกควบคุมนั้น เกิดจากอุปกรณ์ควบคุมใดบ้าง ที่จะส่งผลต่อการเปลี่ยนสถานะของอุปกรณ์ที่ถูกควบคุม วิธีการนี้จะเป็นการกำหนดความเป็นไปได้เบื้องต้นว่าอุปกรณ์ควบคุมใดบ้างที่จะส่งผลต่อการเปลี่ยนแปลงสถานะของอุปกรณ์ที่ถูกควบคุม ซึ่งข้อมูลดังกล่าวจะเป็นข้อมูลเบื้องต้นให้กับผู้ใช้ จากนั้นจะให้ผู้ใช้ตัดสินใจว่าสิ่งที่ได้กำหนดการควบคุมนั้นเกิดจากการเปลี่ยนแปลงสถานะของอุปกรณ์ควบคุมใดบ้าง โดยระบบจะแสดงรายชื่ออุปกรณ์ควบคุมที่มีความสัมพันธ์กับอุปกรณ์ที่กำลังทำการควบคุม เพื่อถามผู้ใช้ให้ยืนยันความเกี่ยวโยงระหว่างอุปกรณ์ควบคุมและอุปกรณ์ที่ถูกควบคุมอีกครั้ง

1) กฎเหตุและผล (Cause and Effect Rule)

ความเกี่ยวโยงระหว่างอุปกรณ์ควบคุมและอุปกรณ์ที่ถูกควบคุม อันเกิดในขณะเกิดเหตุการณ์การเปลี่ยนแปลงสถานะหนึ่งไปสู่สถานะหนึ่งของอุปกรณ์ที่เป็นสาเหตุ (Cause) แล้วส่งผลให้อุปกรณ์ที่ได้รับผลกระทบ (Effect) เปลี่ยนสถานะไปจากเดิม เราเรียกการโปรแกรมในลักษณะนี้ว่า Event Driven Programming ซึ่งการโปรแกรมลักษณะนี้จะคิดว่า ถ้าเกิดเหตุการณ์ขึ้นมาแต่ละอย่างแล้วจะจัดการกับมันอย่างไร

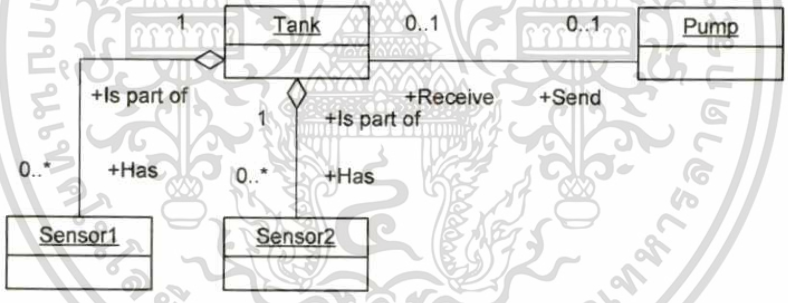
รูปที่ 3.14 แสดงวัตถุสองวัตถุที่ไม่มีความสัมพันธ์กัน



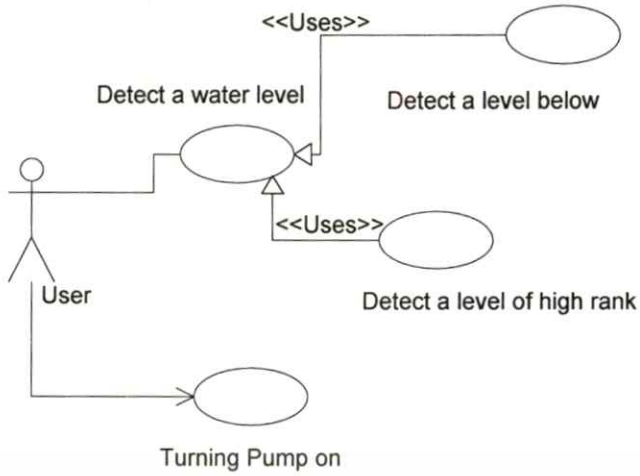
รูปที่ 3.15 แสดงการโปรแกรมของผู้ใช้ เมื่อสวิทช์เปิดผู้ใช้จะโปรแกรมโดยเปิดหลอดไฟ

2) กฎความสัมพันธ์ระหว่างอุปกรณ์ (Relationship Rule)

ความเกี่ยวโยงระหว่างอุปกรณ์ควบคุมและอุปกรณ์ที่ถูกควบคุม อันเกิดจากความสัมพันธ์แบบอาศัย (Dependency relationship) หรือความสัมพันธ์เกี่ยวเนื่อง (Association relationship) โดยที่ความสัมพันธ์เกี่ยวเนื่องนั้นจะส่งผลต่อความความเกี่ยวโยงระหว่างอุปกรณ์ควบคุมและอุปกรณ์ที่ถูกควบคุมที่มีความสัมพันธ์เกี่ยวเนื่อง แต่หากอุปกรณ์ใดมีความสัมพันธ์แบบอาศัยด้วยก็จะส่งผลไปถึงอุปกรณ์ในส่วนย่อยๆ (Part of device) ด้วยเช่นกัน



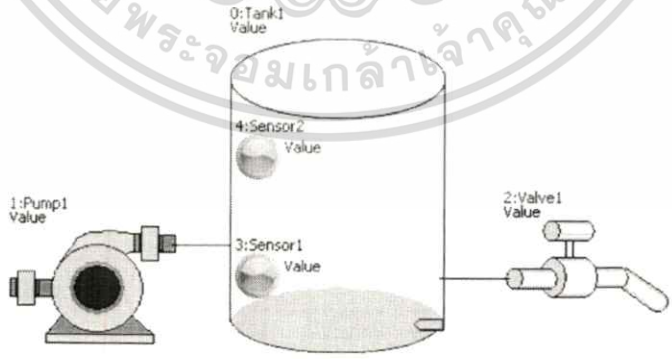
รูปที่ 3.16 แสดงความสัมพันธ์ร่วมระหว่างกลุ่ม Tank และ Sensor1 กับ Sensor2 ร่วมกับความสัมพันธ์แบบ Association relationship ระหว่าง Tank และ Pump



รูปที่ 3.17 แสดงการโปรแกรมของผู้ใช้ซึ่งจะโปรแกรมการเปิดปั๊มโดยมีเงื่อนไขจากระดับน้ำ ซึ่งจะถูกวัดด้วยตัวตรวจจับ 2 ตัวที่ติดตั้งในถังน้ำ

3.4.3 ตัวอย่างวิธีการสลับไปมาระหว่างการโปรแกรมตัวควบคุมและการโปรแกรมแบบจำลอง

เพื่อให้สามารถเข้าใจรูปแบบของการสลับไปมาระหว่างการโปรแกรมตัวควบคุมและการโปรแกรมแบบจำลองได้ง่ายขึ้น จึงขออธิบายโดยใช้กรณีศึกษาเครื่องปั๊มน้ำอัตโนมัติประกอบการอธิบาย โดยการทำงานนั้นจะเป็นการควบคุมการทำงานของปั๊มน้ำอัตโนมัติ ระบบประกอบด้วย ถังน้ำ ปั๊มน้ำ ก๊อกน้ำ ลูกลอยวัดระดับน้ำ 2 ตัว ซึ่งลูกลอยจะติดตั้งในถังน้ำเพื่อวัดระดับน้ำ 2 ระดับด้วยกัน ดังรูปที่ 3.18



รูปที่ 3.18 แสดงองค์ประกอบของปั๊มน้ำอัตโนมัติ

หลักการทำงานคือ เมื่อระดับน้ำต่ำกว่าระดับของลูกลอยตัวที่ 1 (Sensor1) ปั๊มน้ำจะถูกเปิดเพื่อนำน้ำเข้าถัง ปั๊มน้ำจะถูกปิดเมื่อระดับน้ำเข้าถังมากกว่าระดับของลูกลอยตัวที่ 2 (Sensor2) โดยที่

ระดับน้ำจะสัมพันธ์กับปั๊มน้ำ และก๊อกน้ำดังนี้ เมื่อเปิดปั๊มน้ำระดับน้ำในถังจะเพิ่มขึ้น หากเปิดก๊อกน้ำระดับน้ำในถังจะลดลง

การสลับไปมาระหว่างการโปรแกรมตัวควบคุมและการโปรแกรมแบบจำลองของโปรแกรมการทำงานปั๊มน้ำอัตโนมัติ ซึ่งมีลำดับการโปรแกรมตามลำดับดังนี้

1) การโปรแกรมแบบจำลองนั้นเริ่มจากการเข้าสู่โหมดการจำลองสถานการณ์จากนั้นระบบจะทำการตรวจสอบสถานะของอุปกรณ์ที่ควบคุมทั้งหมด ซึ่งในกรณีศึกษานี้คือ Sensor1 และ Sensor2 เพื่อนำมาหาว่าเหตุการณ์ที่เกิดจากสถานะของอุปกรณ์ที่ควบคุมทั้งหมดนี้มีการโปรแกรมตัวควบคุมไว้แล้วหรือยัง เมื่อยังไม่มีการโปรแกรมในเหตุการณ์ดังกล่าวระบบจึงสลับเข้าสู่โหมดการโปรแกรมตัวควบคุม

2) ตามเงื่อนไขการทำงานแล้วผู้ใช้จะกำหนดให้ Pump1 เปิด ระบบจะบันทึกลงในตารางการควบคุม (Controller Statement) ดังนี้ รายการที่ 1 Sensor1 ปิด Sensor2 ปิด ทำให้ Pump1 เปิด และบันทึกสาเหตุลงในตารางสาเหตุ (Input Cause) ว่า Sensor1, Sensor2 เป็นสาเหตุ

3) เข้าสู่โหมดการจำลองสถานการณ์ระบบตรวจสอบสถานะอุปกรณ์พบว่า Pump1 เปลี่ยนแปลงสถานะไปและ Pump1 เป็นอุปกรณ์ที่ส่งผลทางกลกับอุปกรณ์อื่นได้จึงเข้าโหมดโปรแกรมแบบจำลอง

4) การเปิด Pump1 ทำให้ระดับน้ำในถังเพิ่มขึ้น ผู้ใช้จะทำการเพิ่มค่าของ Tank1 ระบบจะบันทึกลงในตารางการทำงานของแบบจำลอง (Simulator statement) ว่า Pump1 เปิด Tank1 จะเพิ่มค่า

5) เข้าสู่โหมดการจำลองสถานการณ์ระบบจะทำการเพิ่มค่าของ Tank1 ไปจนถึงระดับของ Sensor1 ทำให้ Sensor1 เปิด ดังนั้นเมื่อระบบตรวจสอบสถานะอุปกรณ์พบว่า Sensor1 เปิด และตรวจสอบการโปรแกรมว่ามีการโปรแกรมหรือยัง เมื่อยังไม่มีการโปรแกรมในเหตุการณ์ดังกล่าวระบบจึงสลับเข้าสู่โหมดการโปรแกรมตัวควบคุม

6) ตามเงื่อนไขการทำงานแล้วจะไม่เกิดการเปลี่ยนแปลงต่ออุปกรณ์ใดๆ ในระบบจำลองดังนั้นผู้ใช้จะไม่กำหนดใดๆ จากนั้นระบบก็จะบันทึกลงในตารางการควบคุม (Controller Statement) ดังนี้ รายการที่ 1 Sensor1 เปิด Sensor2 ปิด ไม่มีอุปกรณ์ที่ถูกควบคุม และบันทึกสาเหตุลงในตารางสาเหตุ (Input Cause) ว่าไม่มีสาเหตุ

7) เข้าสู่โหมดการจำลองสถานการณ์ระบบจะทำการเพิ่มค่าของ Tank1 ไปจนถึงระดับของ Sensor2 ทำให้ Sensor2 เปิด ดังนั้นเมื่อระบบตรวจสอบสถานะอุปกรณ์พบว่า Sensor2 เปิด และตรวจสอบการโปรแกรมว่ามีการโปรแกรมหรือยัง เมื่อยังไม่มีการโปรแกรมในเหตุการณ์ดังกล่าวระบบจึงสลับเข้าสู่โหมดการโปรแกรมตัวควบคุม

8) ตามเงื่อนไขการทำงานแล้ว ผู้ใช้จะกำหนดให้ Pump1 ปิด ระบบจะบันทึกลงในตารางการควบคุมดังนี้ รายการที่ 1 Sensor1 เปิด Sensor2 เปิด ทำให้ Pump1 ปิด และบันทึกสาเหตุลงในตารางสาเหตุ (Input Cause) ว่า Sensor1, Sensor2 เป็นสาเหตุ

9) เข้าสู่โหมดการจำลองสถานการณ์ระดับน้ำจะไม่เพิ่มขึ้น จากนั้นผู้ใช้จะเปิด Valve1 ระบบตรวจสอบสถานะอุปกรณ์พบว่า Valve1 เปลี่ยนแปลงสถานะไปและ Valve1 เป็นอุปกรณ์ที่ส่งผลทางกลกับอุปกรณ์อื่นได้ จึงเข้าโหมดโปรแกรมแบบจำลอง

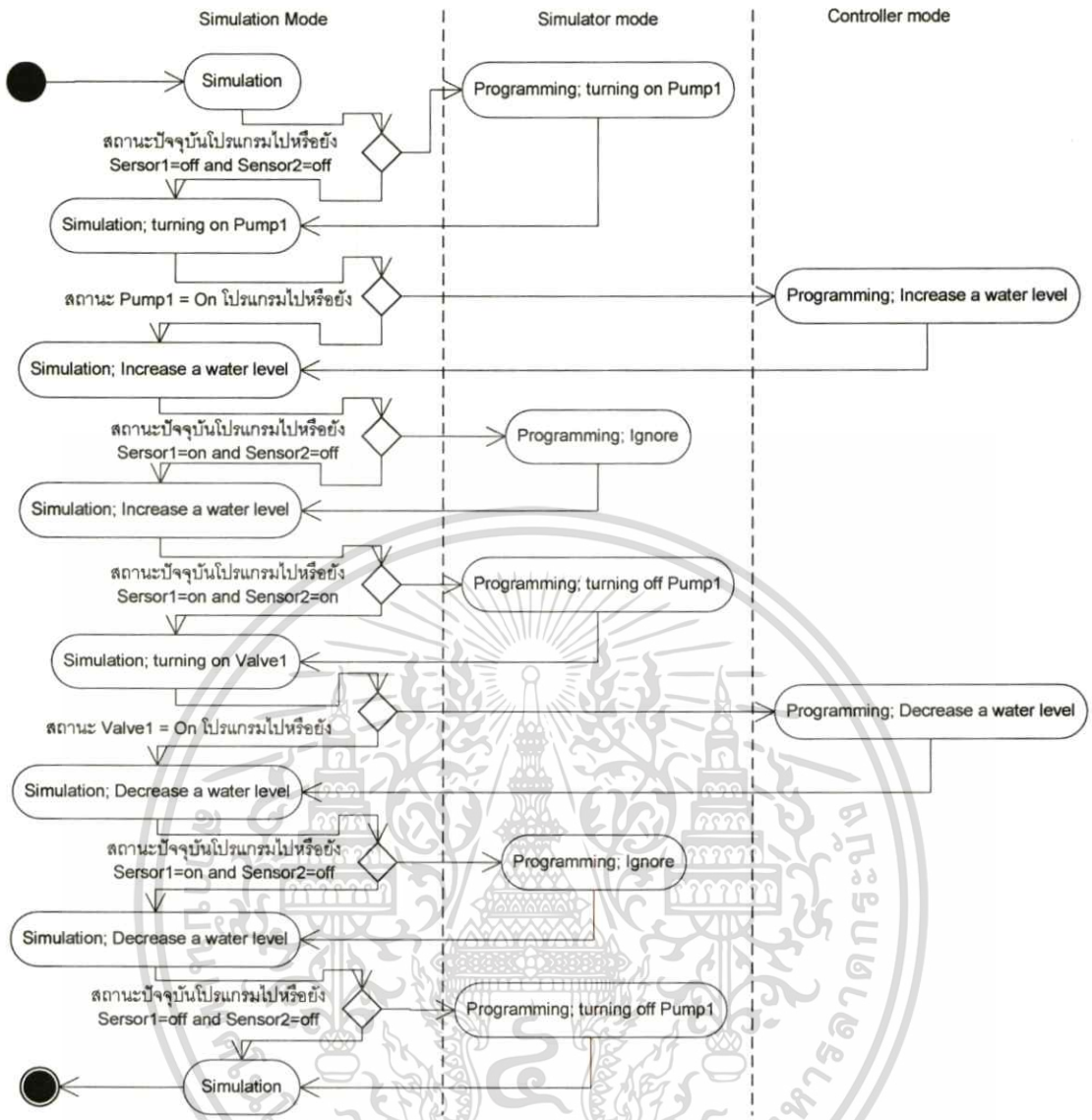
10) การเปิด Valve1 ทำให้ระดับน้ำในถังเพิ่มขึ้น ผู้ใช้จะทำการลดค่าของ Tank1 ระบบจะบันทึกลงในตารางการทำงานของแบบจำลองว่า Valve1 เปิด Tank1 จะลดค่าลง

11) เข้าสู่โหมดการจำลองสถานการณ์ระบบจะทำการลดค่าของ Tank1 ไปจนถึงระดับของ Sensor2 ทำให้ Sensor2 ปิด ดังนั้นเมื่อระบบตรวจสอบสถานะอุปกรณ์พบว่า Sensor2 ปิด และตรวจสอบการโปรแกรมว่ามีการโปรแกรมหรือยัง เมื่อยังไม่มีการโปรแกรมในเหตุการณ์ดังกล่าว ระบบจึงสลับเข้าสู่โหมดการโปรแกรมตัวควบคุม

12) ตามเงื่อนไขการทำงานแล้วจะไม่เกิดการเปลี่ยนแปลงต่ออุปการณ์ใดๆ ในระบบจำลอง ดังนั้นผู้ใช้จะไม่กำหนดใดๆ ระบบจะบันทึกลงในตารางการควบคุมดังนี้ รายการที่ 1 Sensor1 เปิด Sensor2 ปิด ไม่มีอุปกรณ์ที่ถูกควบคุม และบันทึกสาเหตุลงในตารางสาเหตุ (Input Cause) ว่าไม่มีสาเหตุ

13) เข้าสู่โหมดการจำลองสถานการณ์ระบบจะทำการลดค่าของ Tank1 ไปจนถึงระดับของ Sensor1 ทำให้ Sensor1 ปิด ดังนั้นเมื่อระบบตรวจสอบสถานะอุปกรณ์พบว่า Sensor1 ปิด และตรวจสอบการโปรแกรมว่ามีการโปรแกรมหรือยัง ในเหตุการณ์นี้เราได้โปรแกรมไว้แล้ว ระบบจึงทำตามโปรแกรมที่ได้โปรแกรมไว้แล้ว และการโปรแกรมนี้จะสิ้นสุดลง

จากขั้นตอนการโปรแกรม เราสามารถนำมาเขียนเป็นแผนผังกิจกรรมการโปรแกรมได้ดังรูปที่ 3.19



รูปที่ 3.19 แสดงแผนผังกิจกรรมโปรแกรมการทำงานของปั๊มน้ำอัตโนมัติ

3.5 การสร้างชุดคำสั่งจากการสาธิต

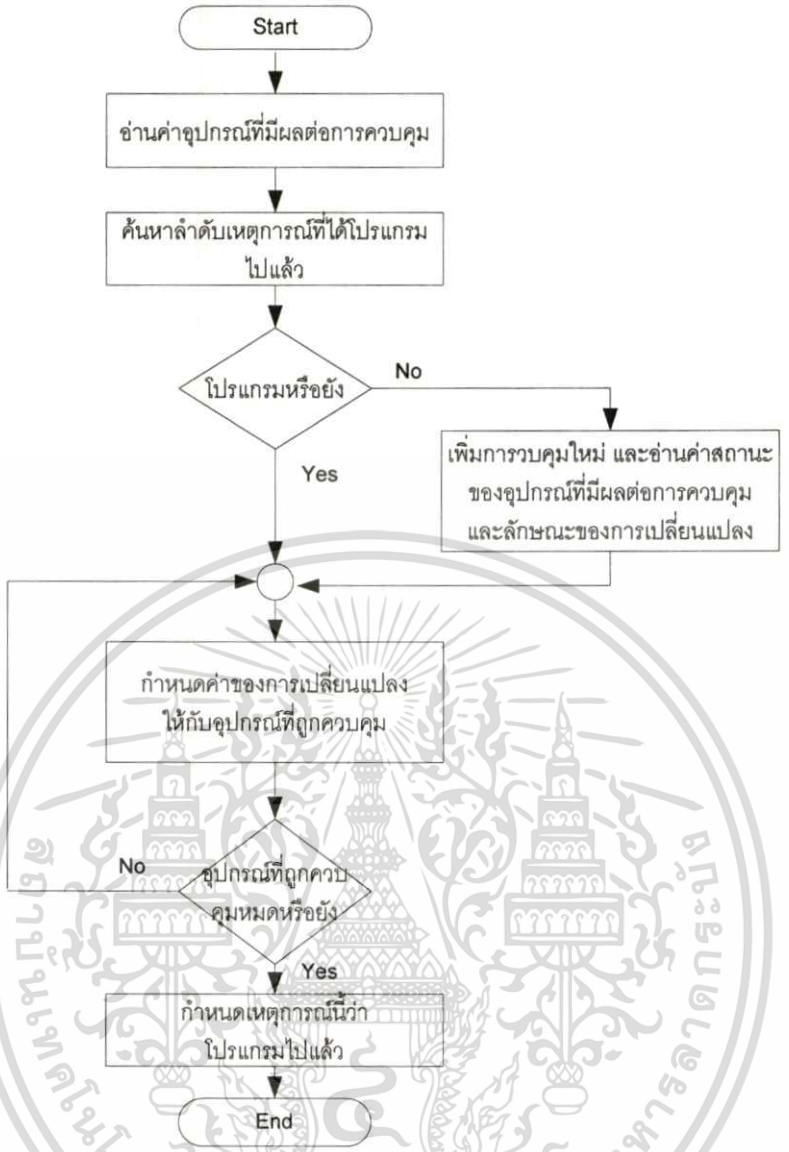
การสร้างชุดคำสั่งจากการสาธิต ระบบจะมีการบันทึกการกระทำของผู้ใช้ที่ทำต่อระบบจำลองการทำงานของเครื่องจักรนั้น สามารถแบ่งออกเป็น 2 ส่วนหลัก ส่วนแรกคือส่วนของการบันทึกการกระทำของผู้ใช้ที่กระทำต่อระบบจำลองในโหมดโปรแกรมแบบจำลองส่วนนี้ระบบจะบันทึกการเปลี่ยนแปลงข้อมูลของอุปกรณ์ที่ถูกผลกระทบ โดยจะเชื่อมโยงกับอุปกรณ์ที่เป็นสาเหตุ (หมายถึงอุปกรณ์ที่เปลี่ยนแปลงตัวสุดท้ายก่อนการเข้าสู่โหมดโปรแกรมแบบจำลอง) ดังนั้นความสัมพันธ์ของอุปกรณ์ที่เป็นสาเหตุกับอุปกรณ์ที่ได้รับผลกระทบจะมีลักษณะแบบหนึ่งต่อหลายๆ ตัว ส่วนที่สองคือส่วนบันทึกการกระทำของผู้ใช้ที่กระทำต่อระบบจำลองในโหมดการ

โปรแกรมตัวควบคุมส่วนนี้ระบบจะบันทึกการเปลี่ยนแปลงข้อมูลของอุปกรณ์ที่ถูกควบคุม โดยอ้างอิงเหตุการณ์กับสถานะของอุปกรณ์ที่ทำหน้าที่ควบคุมทั้งหมด ซึ่งอุปกรณ์ที่ถูกควบคุมนั้นอาจเป็นเพียงบางส่วนของอุปกรณ์ควบคุมที่ส่งผลหรือเป็นสาเหตุให้เกิดการเปลี่ยนแปลงสถานะไป และในเหตุการณ์เดียวกันนี้ ความสัมพันธ์ระหว่างอุปกรณ์ที่ทำหน้าที่ควบคุมและอุปกรณ์ที่ถูกควบคุมแต่ละตัวนั้นอาจไม่เหมือนกันก็ได้

อีกส่วนหนึ่งที่เกิดการกระทำของผู้ใช้ที่มีต่อระบบคือการกระทำของผู้ใช้ที่กระทำต่อระบบจำลองในโหมดการจำลองสถานการณ์ในโหมดนี้การเปลี่ยนแปลงสถานะของอุปกรณ์นั้น ถือว่าเป็นการกระทำจากภายนอกระบบจำลองที่ส่งผลต่อการเปลี่ยนแปลงสถานะของอุปกรณ์ ในขณะที่ระบบจำลองทำงาน โดยที่การกระทำของผู้ใช้ในโหมดนี้นั้นจะไม่มีการบันทึกใดๆ

3.5.1 การบันทึกการกระทำของผู้ใช้โหมดโปรแกรมแบบจำลอง

การบันทึกการกระทำของผู้ใช้ที่กระทำต่อระบบจำลองในโหมดโปรแกรมแบบจำลองเครื่องมือนี้จะบันทึกการเปลี่ยนแปลงข้อมูลของอุปกรณ์ที่ถูกผลกระทบ โดยจะเชื่อมโยงกับอุปกรณ์ที่เป็นสาเหตุ (หมายถึงอุปกรณ์ที่เปลี่ยนแปลงตัวสุดท้ายก่อนการเข้าสู่โหมดโปรแกรมแบบจำลอง ดังนั้นความสัมพันธ์ของอุปกรณ์ที่เป็นสาเหตุกับอุปกรณ์ที่ได้รับผลกระทบจะมีลักษณะแบบหนึ่งต่อหลายๆ ตัวลงในตารางการทำงานของแบบจำลองนั้นเป็นดังนี้



รูปที่ 3.20 แสดงผังลำดับการทำงานในอัลกอริทึมในการบันทึกการกระทำของผู้ใช้ที่กระทำต่อระบบจำลองในโหมดโปรแกรมแบบจำลอง

อัลกอริทึมในการบันทึกการกระทำของผู้ใช้ที่กระทำต่อระบบจำลองในโหมดโปรแกรมแบบจำลองเป็นดังนี้

Algorithm RecordSimulatorAction

Input: รายการการกระทำของผู้ใช้ (L), รายการสถานะของอุปกรณ์ที่ได้รับผลกระทบ (O),
รหัสของอุปกรณ์ที่เป็นสาเหตุ (InputID),
สถานะของอุปกรณ์ที่เป็นสาเหตุ (InputValue)

Output: รายการการกระทำของผู้ใช้ที่ปรับปรุงเพิ่มเติม (L)

Action = L_{Last}

for each item **in** the list L, **do**

if the item = InputID, **then**

Action ⟵ L_{item}

end if

end for each

if It's new action (Action = L_{Last}) , **then**

Action.DeviceID ⟵ InputID

Action.DeviceValue ⟵ InputValue

Last ⟵ Last + 1

end if

for each item **in** the list O, **do**

for each j in the list Action.Output, **do**

if the Action. $Output_j$ .DeviceID = O_{item} .DeviceID, **then**

Output ⟵ Action. $Output_j$

end if

Ouput.Value ⟵ O_{item} .Value;

Ouput.ChangeType ⟵ O_{item} . ChangeType;

if It's new output device (Ouput = Action. $Output_{Last}$) , **then**

Ouput .DeviceID ⟵ O_{item} .DeviceID;

Action.Output.Last = Action. Output.Last + 1

end if

end for each

end for each

รูปที่ 3.21 แสดงอัลกอริทึมในการบันทึกการกระทำของผู้ใช้ที่กระทำต่อระบบจำลองในโหมดโปรแกรมแบบจำลอง

3.5.2 การบันทึกการกระทำของผู้ใช้ในโหมดการโปรแกรมตัวควบคุม

จากกรณีศึกษาเครื่องปั๊มน้ำอัตโนมัติในข้อ 3.4.3 นั้นจะเห็นได้ว่าการเปลี่ยนแปลงในสถานะของอุปกรณ์ควบคุม (Sensor1 และ Sensor2) จะส่งผลต่อสถานะอุปกรณ์ที่ถูกควบคุม (Pump1) โดยสามารถสรุปตารางการเปลี่ยนแปลงสถานะที่บันทึกลงในตารางการควบคุมได้ดังนี้

ตารางที่ 3.1 แสดงการเปลี่ยนแปลงสถานะที่บันทึกลงในตารางการควบคุม

ลำดับที่	*Input device*		*Output device*		
	Sensor1	*Sensor2*	*Pump1*	*Valve1*	*Tank1*
2	0	0	1	X	X
6	1	0	X	X	X
8	1	1	0	X	X
12	1	0	X	X	X

0 = State Off, 1=State On, X=User Ignore.

จากการเปลี่ยนแปลงสถานะของอุปกรณ์ที่ถูกควบคุม (Pump1) สามารถสรุปการบันทึกสาเหตุลงในตารางสาเหตุ (Input Cause) ได้ดังนี้

ตารางที่ 3.2 แสดงการบันทึกสาเหตุของ Pump1 ในตารางสาเหตุ

ลำดับที่	*Output*	*Cause device*	
	Pump1	*Sensor1*	*Sensor2*
2	1	True	True
6	X	False	False
8	0	True	True
12	X	False	False

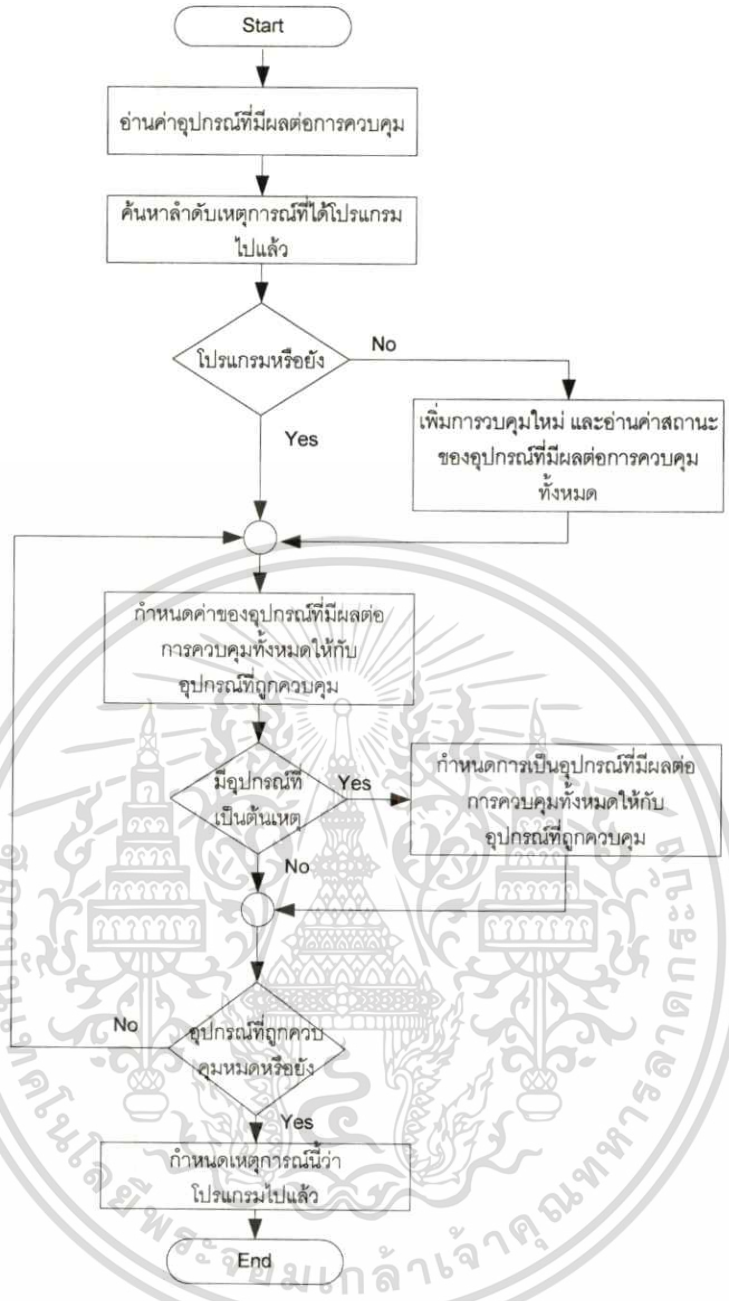
0 = State Off, 1=State On, X=User Ignore.

จากการเปลี่ยนแปลงสถานะของอุปกรณ์ที่ส่งผลต่ออุปกรณ์อันเกิดจากแรงกล จะสามารถสรุปการบันทึกตารางการทำงานของแบบจำลองได้ดังนี้

ตารางที่ 3.3 แสดงการบันทึกตารางการทำงานของแบบจำลอง

ลำดับที่	*Input Device*	*Output Device*	*Value*
4	Pump1	Tank1	+1
10	Valve1	Tank1	-1

การบันทึกการกระทำของผู้ใช้ที่กระทำต่อระบบในโหมดการโปรแกรมตัวควบคุม เครื่องมือนี้จะบันทึกการเปลี่ยนแปลงข้อมูลของอุปกรณ์ที่ถูกควบคุม โดยอ้างอิงเหตุการณ์และสถานะของอุปกรณ์ที่ทำหน้าที่ควบคุมทั้งหมด ซึ่งการเปลี่ยนแปลงสถานะของอุปกรณ์ที่ถูกควบคุมนั้นอาจเกิดจากบางส่วนของอุปกรณ์ควบคุมที่ส่งผลหรือเป็นสาเหตุให้เกิดการเปลี่ยนแปลงสถานะไป และในเหตุการณ์เดียวกันนี้ความสัมพันธ์ระหว่างอุปกรณ์ที่ทำหน้าที่ควบคุมและอุปกรณ์ที่ถูกควบคุมแต่ละตัวนั้นอาจไม่เหมือนกันก็ได้ โดยจะบันทึกการเปลี่ยนแปลงและอุปกรณ์ที่เป็นสาเหตุนี้ลงในตารางการควบคุม ดังนี้



รูปที่ 3.22 แสดงผังลำดับการทำงานในการบันทึกการกระทำของผู้ใช้ที่กระทำต่อระบบในโหมดการโปรแกรมตัวควบคุม

อัลกอริทึมในการบันทึกการกระทำของผู้ใช้ที่กระทำต่อระบบในโหมดการโปรแกรมตัวควบคุมเป็นดังนี้

Algorithm RecordControllerAction

Input: รายการการกระทำของผู้ใช้ (L), รายการสถานะของอุปกรณ์ที่ได้รับผลกระทบ (O), รายการสถานะของอุปกรณ์ทั้งหมด (I)

Output: รายการการกระทำของผู้ใช้ที่ปรับปรุงเพิ่มเติม (L)

Action = L_{Last}

for each item **in** the list L, **do**

  if the L_{item} . Array of input state = I. Array of Sate, **then**

    Action ⟵ L_{item}

  end if

end for each

if It's new action (Action = L_{Last}) , **then**

  for each j **in** the list I, **do**

    Action. $InputState_j$ ⟵ I_j . State

  end for each

  Last ⟵ Last + 1

end if

for each item **in** the list O, **do**

  Action. $Ouput_{O_{item}.DeviceID}$ .State ⟵ O_{item} .State

  for each j **in** the list I, **do**

    Action. $Ouput_{O_{item}.DeviceID}$. $Cause_j$ ⟵ O_{item} . $Cause_j$

  end for each

end for each

if O is empty, **then**

  Action .IgnoreCase ⟵ true

end if

รูปที่ 3.23 แสดงอัลกอริทึมในการบันทึกการกระทำของผู้ใช้ที่กระทำต่อระบบในโหมดการโปรแกรมตัวควบคุม

3.6 เหตุการณ์ที่ไม่ได้ถูกโปรแกรม (Missing case)

เหตุการณ์ที่ไม่ได้ถูกโปรแกรม เป็นส่วนสำคัญในการขจัดความผิดพลาดของการโปรแกรมให้หมดไป การเกิดเหตุการณ์ในลักษณะนี้อาจส่งผลต่อการทำงานของเครื่องจักรหรืออาจไม่ส่งผลใดๆ ต่อการทำงานของเครื่องจักรเลยก็ได้ ขึ้นอยู่กับว่าเหตุการณ์ดังกล่าวเกิดขึ้นในขณะเครื่องจักรทำงานหรือไม่ เพราะบางเหตุการณ์อาจไม่เกิดขึ้นเลยก็ได้จากกฎของธรรมชาติ ตัวอย่างเช่น อุณหภูมิจะต้องผ่าน 20 องศา ก่อนเสมอจึงจะไปถึงจุดที่อุณหภูมิ 80 องศาได้ ดังนั้นหากเราติดตั้งตัววัดอุณหภูมิที่ 20 และ 80 องศา เหตุการณ์ที่จะไม่สามารถเกิดขึ้นได้ตามกฎธรรมชาติคือเหตุการณ์ที่ตัววัดอุณหภูมิที่ 80 องศาทำงาน แต่ตัววัดอุณหภูมิที่ 20 องศาไม่ทำงาน เพื่อป้องกันข้อผิดพลาดในการโปรแกรม เราจึงจำเป็นต้องทำการค้นหาและจัดการเหตุการณ์ที่ไม่ได้ถูกโปรแกรมให้หมดไป

3.6.1 การเกิดเหตุการณ์ที่ไม่ได้ถูกโปรแกรม

สาเหตุการเกิดเหตุการณ์ที่ไม่ได้ถูกโปรแกรมเกิดขึ้นได้สองประการ ประการแรกคือผู้ใช้ยังโปรแกรมเหตุการณ์นั้นไม่ครบถ้วน อีกประการหนึ่งคือเป็นเหตุการณ์ที่มิอาจเกิดขึ้นจริงได้ตามกฎของธรรมชาติ เพื่อให้สามารถทำความเข้าใจได้ง่ายจึงขออธิบายสาเหตุการเกิดเหตุการณ์ที่ไม่ได้ถูกโปรแกรมผ่านทางกรณีศึกษาเครื่องปั๊มน้ำอัตโนมัติ

จากกรณีศึกษานี้เราพอที่จะสรุปเหตุการณ์ได้ดังนี้ จากการเปลี่ยนแปลงสถานะของอุปกรณ์ควบคุม (Sensor1 และ Sensor2) ดังกล่าวที่ส่งผลต่อสถานะอุปกรณ์ที่ถูกควบคุม (Pump1) สามารถสรุปตารางการเปลี่ยนแปลงสถานะ (State table) ดังแสดงในตารางที่ 3.4

ตารางที่ 3.4 แสดงการเปลี่ยนแปลงสถานะของอุปกรณ์ควบคุมและอุปกรณ์ที่ถูกควบคุม (State table)

No	*Sensor1*	*Sensor2*	*Pump1*
1	0	0	1
2	1	0	X
3	1	1	0
4	1	0	X

0 = State Off, 1=State On, X=User Ignore.

จากตารางที่ 3.4 แสดงการเปลี่ยนแปลงสถานะของอุปกรณ์ควบคุม (Sensor1, Sensor2) และอุปกรณ์ที่ถูกควบคุม (Pump1) ซึ่งสามารถเขียนเป็นตารางความจริง (Truth table) เพื่ออธิบาย

ความเป็นไปได้ทั้งหมดเชิงตรรกะของอุปกรณ์ควบคุม ในกรณีนี้เหตุการณ์ของการกระทำของผู้ใช้จะครอบคลุมเหตุการณ์ที่เกิดขึ้นทั้งหมด จากนั้นนำผลลัพธ์ของเหตุการณ์ (State of Pump1) นำมาใส่ในตารางความจริง (Truth table) โดยยึดจากสถานะของอุปกรณ์ควบคุมเป็นหลัก และตัดแถวที่ซ้ำซ้อนออกไป อย่างไรก็ตามมีเหตุการณ์บางส่วนของการกระทำของผู้ใช้ที่ไม่ครอบคลุมความเป็นไปได้ทั้งหมด คือเหตุการณ์ที่สถานะของ Sensor1 เปิด และ Sensoe2 ปิด ระบบจะถามผู้ใช้ถึงข้อมูลเพิ่มเติม โดยการแสดงเหตุการณ์ที่หายไปผ่านทางแบบจำลอง เพื่อให้ผู้ใช้กำหนดการควบคุมเพิ่มเติม ซึ่งผู้ใช้อาจเลือกที่จะไม่สนใจเหตุการณ์ดังกล่าวก็ได้ ดังแสดงในตารางที่ 3.5

ตารางที่ 3.5 แสดงความจริงของสถานะทั้งหมดที่จะเกิดขึ้นได้ตามตรรกะของอุปกรณ์ควบคุม (Truth table)

No	*Sensor1*	*Sensor2*	*Pump1*
1	0	0	0
2	0	1	-
3	1	0	X
4	1	1	1

0 = State Off, 1=State On, X=User Ignore, - = Unknown (Missing case)

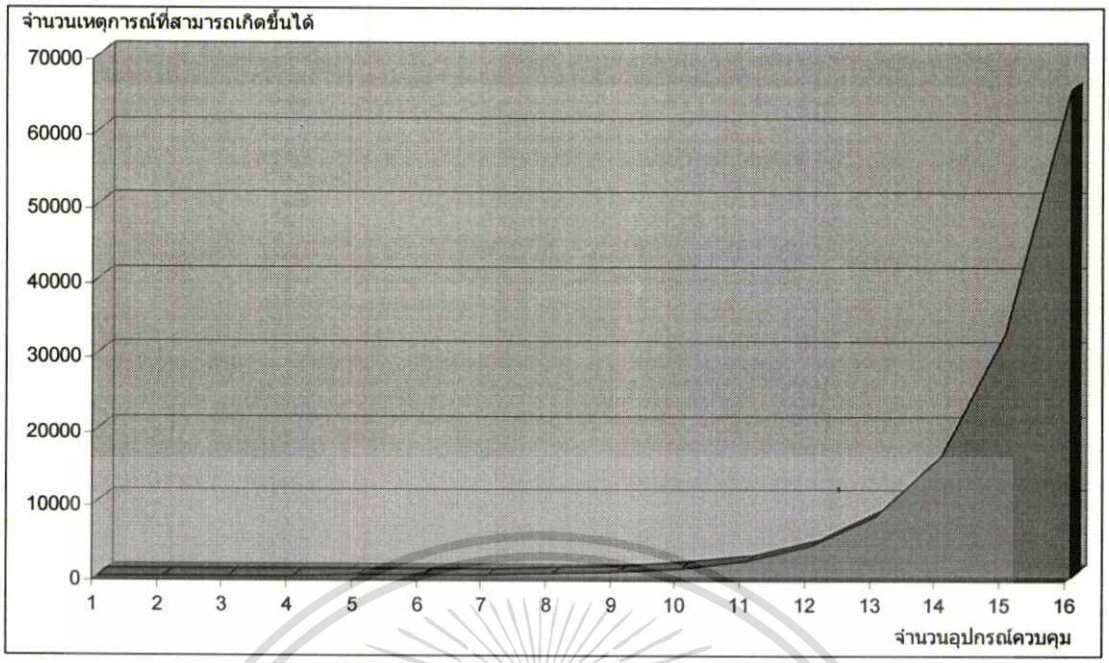
การสร้างตารางความจริง (Truth table) ของจำนวนเหตุการณ์ที่จะเกิดขึ้นได้ทั้งหมด จะขึ้นอยู่กับจำนวนอุปกรณ์ที่มีหน้าที่เป็นอุปกรณ์ควบคุม โดยที่จำนวนเหตุการณ์ที่จะเกิดขึ้นได้ทั้งหมดหาได้จากสมการ (3.1)

$$X = 2^n \tag{3.1}$$

X คือ จำนวนเหตุการณ์ทั้งหมดที่สามารถเกิดขึ้นได้

n คือ จำนวนของอุปกรณ์ควบคุม ซึ่งหมายถึงอุปกรณ์ใดๆ ที่ส่งผลต่อเงื่อนไขการทำงานของอุปกรณ์อื่นๆ

จำนวนเหตุการณ์ที่จะเกิดขึ้นได้ทั้งหมดจะแปรผันตามจำนวนของอุปกรณ์ควบคุม ดังแสดงในรูปที่ 3.24



รูปที่ 3.24 แสดงจำนวนเหตุการณ์ที่สามารถเกิดขึ้น ต่อจำนวนอุปกรณ์ควบคุม

ตารางความจริงของสถานะทั้งหมดในตารางที่ 3.5 สามารถแยกรูปแบบของผลที่เกิดขึ้นในตารางความจริงออกมาได้เป็นสามลักษณะคือ

ลักษณะที่ 1: ผลลัพธ์ที่เกิดจากการควบคุม จากการวิเคราะห์ตารางความจริงเราจะเห็นได้ว่าลำดับที่ 1 และ 4 นั้นผู้ใช้ได้มีการโปรแกรมตัวควบคุมไว้ ซึ่งสามารถเขียนชุดคำสั่งในการควบคุมการเปิดและปิด Pump1 ที่สัมพันธ์กับสถานะของอุปกรณ์ควบคุมคือ Sensor1 และ Sensor2 ได้ดังนี้

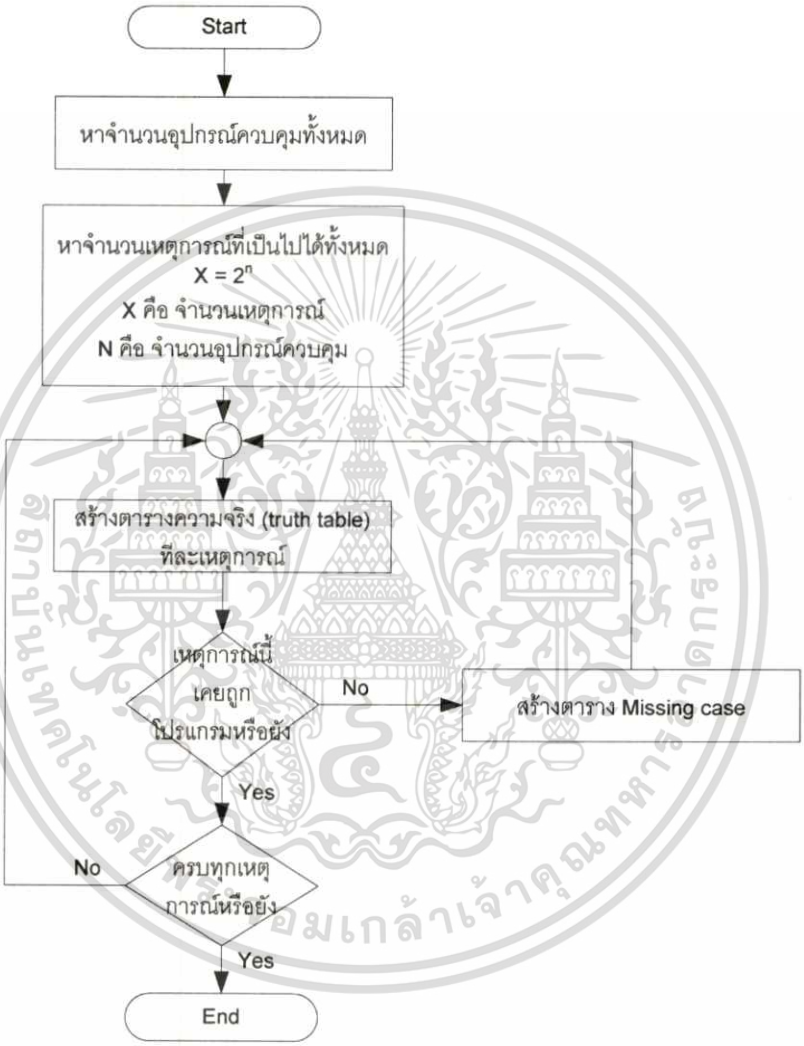
1) ในกรณีที่ปิด Pump1 "IF Sensor1.State = On AND Sensor2.State = On THEN Pump1.State = Off"

2) ในกรณีที่เปิด Pump1 "IF Sensor1.State = Off AND Sensor2.State = Off THEN Pump1.State = On"

ลักษณะที่ 2: ผลลัพธ์ที่เกิดการละทิ้ง (User Ignore) จากการวิเคราะห์ตารางความจริงเราจะเห็นได้ว่าลำดับที่ 2 นั้นผู้ใช้ได้มีการกำหนดการละทิ้งไว้ (Ignore case)

ลักษณะที่ 3: ผลลัพธ์ที่ไม่ได้ถูกกำหนดไว้ (Missing case) จากการวิเคราะห์ตารางความจริงเราจะเห็นได้ว่าลำดับที่ 3 หากเราพิจารณาถึงสาเหตุของการเกิดเหตุการณ์ขึ้น จากกรณีศึกษาดังกล่าวเราสามารถกล่าวได้ว่า เหตุการณ์นี้ไม่สามารถเกิดขึ้นจริงตามกฎของธรรมชาติ นั่นคือระดับน้ำถึงระดับของ Sensor2 โดยที่ไม่ผ่าน Sensor1 ก่อน

การหาเหตุการณ์ที่ไม่ได้ถูกโปรแกรม จะแบ่งออกเป็นสองส่วน ส่วนแรกจะเป็นการสร้างตารางความจริง (Truth table) ของจำนวนเหตุการณ์ที่จะเกิดขึ้นได้ทั้งหมด ส่วนที่สองคือการนำเหตุการณ์ที่ได้โปรแกรมไว้แล้วมาหักออกจากตารางความจริง สิ่งที่เหลือคือเหตุการณ์ที่อยู่นอกเหนือโปรแกรมตัวควบคุม ดังแสดงผังลำดับการทำงานในการหาเหตุการณ์ที่ไม่ได้ถูกโปรแกรมได้ดังรูปที่ 3.25



รูปที่ 3.25 แสดงผังลำดับการทำงานในการหาเหตุการณ์ที่ไม่ได้ถูกโปรแกรม

อัลกอริทึ่มในการหาเหตุการณ์ที่ไม่ได้ถูกโปรแกรมนั้นเป็นดังนี้

Algorithm GetMissingCase

Input: รายการเหตุการณ์ที่ถูกโปรแกรม (L), รายการสถานะของอุปกรณ์ทั้งหมด (I)

Output: รายการเหตุการณ์ที่ไม่ได้ถูกโปรแกรม (M)

ControllerDeviceCount ⟵ 0

for each item **in** the list I, **do**

  for each j **in** the list L, **do**

    for each k **in** the list L.Output, **do**

      if It's cause ($L_{item}.Output_{j}.Cause_{k}$ = true) , **then**

        Cause ⟵ true

  if Cause = true, **then**

    ControllerDeviceCount ⟵InputDeviceCount + 1

end for each

POP ⟵ $2^{\text{ControllerDeviceCount}}$

for each y 0 to POP, **do**

  for each x 0 to ControllerDeviceCount, **do**

    if (Y **mod** 2^{x}) = 0, **then**

      if Y < 1, **then**

        if StateOfBit_{x} = 0, **then**

          StateOfBitTop ⟵ 1

        else

          StateOfBitTop ⟵0

      else

        StateOfBitTop ⟵0

    else

      StateOfBitTop ⟵ StateOfBit_{x}

  end for each

  MissingCase ⟵ false

  for each item **in** the list L, **do**

    for each j 0 to ControllerDeviceCount, **do**

      if L_{item} .InputState $InputState_{DeviceIndex_{j}}$ = StateOfBit_{j} , **then**

        MissingCase ⟵ true

  if MissingCase = true, **then**

    M_{item} ⟵ new M

    for each j 0 to ControllerDeviceCount, **do**

      $M_{item \cdot DeviceIndex_{j}}$ = StateOfBit_{j} ⟵ StateOfBit_{j}

  end if

end for each

รูปที่ 3.26 แสดงอัลกอริทึ่มในการหาเหตุการณ์ที่ไม่ได้ถูกโปรแกรม

3.6.2 การจัดการเหตุการณ์ที่ไม่ได้ถูกโปรแกรม

เหตุการณ์ที่ไม่ได้ถูกโปรแกรม เป็นส่วนสำคัญในการขจัดความผิดพลาดของการโปรแกรมให้หมดไป การจัดการเหตุการณ์ที่ไม่ได้ถูกโปรแกรมนั้น ลำดับแรกจะต้องหาเหตุการณ์ที่ไม่ได้ถูกโปรแกรมทั้งหมดเสียก่อน โดยหาได้จากการสร้างตารางความจริงจำนวนเหตุการณ์ที่จะเกิดขึ้นได้ทั้งหมด แล้วนำเหตุการณ์ที่ได้โปรแกรมไว้แล้วมาหักออกจากตารางความจริง สิ่งที่เหลือคือเหตุการณ์ที่อยู่นอกเหนือโปรแกรมตัวควบคุมดังได้กล่าวมาแล้วข้างต้น จากนั้นนำเหตุการณ์ที่ไม่ได้ถูกโปรแกรมทั้งหมดมาแสดงผ่านทางสถานะต่างๆ ของอุปกรณ์ในแบบจำลอง โดยแสดงทีละเหตุการณ์จนครบทุกเหตุการณ์ ในระหว่างการแสดงไปทีละเหตุการณ์นั้น ผู้ที่ทำการโปรแกรมจะต้องทำการโปรแกรมเพิ่มเติม เพื่อให้เหตุการณ์ที่ไม่ได้ถูกโปรแกรมหมดไป โดยการจัดการนี้สามารถแบ่งออกได้เป็นสองกรณีดังนี้

1) กรณีที่เหตุการณ์ที่เราไม่ประสงค์จะโปรแกรมตัวควบคุม หรือการละทิ้งเหตุการณ์นี้ (User Ignore) ซึ่งจะใช้กับเหตุการณ์ในลักษณะที่ไม่ส่งผลใดๆ ต่อการทำงานของเครื่องจักร หรือเหตุการณ์อาจไม่เกิดขึ้นเลยก็ได้จากกฎของธรรมชาติ

2) กรณีที่เราต้องการจะโปรแกรมตัวควบคุมเพิ่มเติม ซึ่งจะใช้กับเหตุการณ์ในลักษณะที่ส่งผลต่อการทำงานของเครื่องจักร ซึ่งในกรณีนี้ผู้ใช้จะต้องทำการกำหนดการควบคุมเพิ่มเติมว่าเหตุการณ์ดังกล่าวนั้นส่งผลต่อสถานะใดกับอุปกรณ์อื่นๆ บ้าง

3.7 การแปลง User Action สู่ภาษาของอุปกรณ์พีแอลซี (Compiler)

เมื่อแบบจำลองสามารถทำงานได้อย่างถูกต้องแล้ว ขั้นตอนต่อไปคือการแปลง User Action สู่ภาษาของอุปกรณ์พีแอลซี ซึ่งภาษาที่ใช้ในการเขียนโปรแกรมตามมาตรฐาน IEC 1131-3 กำหนดไว้ห้าภาษา คือ Ladder Diagram (LD), Function Block Diagram (FBD), Instruction List (IL) หรือ Mnemonic, Structure Text (ST) และ Sequential Function Chart (SFC) ในวิทยานิพนธ์ฉบับนี้ได้นำเสนอการแปลง User Action สู่ภาษาของอุปกรณ์พีแอลซีในสองภาษาคือ ภาษาแบบโครงสร้าง และ ภาษานีโมนิค

3.7.1 การแปลง User action สู่ภาษา Structure Text

ภาษา Structure Text นั้นเป็นภาษาในระดับสูง โดยมีพื้นฐานมาจากการเขียนโปรแกรมในลักษณะของการโปรแกรมแบบมีโครงสร้าง (Structure Programming) โดยภาษาในลักษณะนี้มีหลักการสามแบบคือ การทำงานแบบตามลำดับ (Sequence) การเลือกกระทำตามเงื่อนไข (Decision) และการทำซ้ำ (Loop)

1) การทำงานแบบตามลำดับ(Sequence): รูปแบบการเขียนโปรแกรมที่ง่ายที่สุดคือ เขียนให้ทำงานจากบนลงล่าง เขียนคำสั่งเป็นบรรทัด และทำทีละบรรทัดจากบรรทัดบนสุดลงไปจนถึงบรรทัดล่างสุด สมมติให้มีการทำงานสามกระบวนการคือ อ่านข้อมูล คำนวณ และส่งออกข้อมูล

2) การเลือกกระทำตามเงื่อนไข (Decision): การตัดสินใจ หรือเลือกเงื่อนไขคือ เขียนโปรแกรมเพื่อนำค่าข้อมูลไปเลือกกระทำ โดยปกติจะมีเหตุการณ์ให้ทำสองกระบวนการ คือ เงื่อนไขเป็นจริงจะกระทำกระบวนการหนึ่ง และเป็นเท็จจะกระทำอีกกระบวนการหนึ่ง แต่ถ้าซับซ้อนมากขึ้น จะต้องใช้เงื่อนไขหลายชั้น

3) การทำซ้ำ (Repeating or Loop): การทำกระบวนการหนึ่งหลายครั้ง โดยมีเงื่อนไขในการควบคุม ซึ่งหมายถึงการทำซ้ำในขณะที่เป็นจริง และเลิกการทำซ้ำเมื่อเงื่อนไขเป็นเท็จ

ภาษาแบบโครงสร้างตามมาตรฐาน IEC 1131-3 นั้นมีพื้นฐานมาจากภาษาปาสคาล ซึ่งจะประกอบไปด้วย นิพจน์ และคำสั่ง โดยคำสั่งทั่วไปจะอยู่ในรูปของคำสั่งเกี่ยวกับการเลือกทำงาน เช่น IF……..THEN……..ELSE เป็นต้น

3.7.1.1 หลักการเขียนภาษาแบบโครงสร้างตามมาตรฐาน IEC 1131-3

1) ชนิดของตัวแปร (Data Types) ในภาษาแบบโครงสร้าง มีดังนี้

ตารางที่ 3.6 แสดงชนิดตัวแปรในภาษาแบบโครงสร้าง

ชนิด	รายละเอียด	ขนาด
SINT	short integer	1 byte
INT	integer	2 bytes
DINT	double integer	4 bytes
LINT	long integer	8 bytes
USINT	unsigned short integer	1 byte
UINT	unsigned integer	2 bytes
UDINT	unsigned double integer	4 bytes
ULINT	unsigned long integer	8 bytes
REAL	real	4 bytes
LREAL	long real	8 bytes
TIME	time duration	
DATE	calendar date	
TOD	time of day	
DT	date and time of day	

ตารางที่ 3.6 แสดงชนิดตัวแปรในภาษาแบบโครงสร้าง (ต่อ)

ชนิด	รายละเอียด	ขนาด
STRING	character strings	
BOOL	boolean	1 bit
BYTE	byte	1 byte
WORD	16 bit bit string	16 bits
DWORD	32 bit bit string	32 bits
LWORD	64 bit bit string	64 bits

2) การกำหนดตัวแปร (Variable Declarations)

รูปแบบ

ชื่อตัวแปร : ชนิดตัวแปร;

ตัวอย่างเช่น

```
input    :    BOOL;
t        :    TIME;
out      :    BOOL;
count    :    INT;
```

3) ขอบเขตของตัวแปร (Local variable)

VAR คือตัวแปรภายในหน่วยความจำ

VAR_INPUT คือตัวแปรที่รับค่าจากช่องรับสัญญาณ (Input port)

VAR_OUTPUT คือตัวแปรที่รับค่าจากช่องส่งสัญญาณ (Output port)

VAR_IN_OUT คือตัวแปรที่รับค่าจากช่องรับและส่งสัญญาณ (Input and Output port)

ตัวอย่างเช่น

```
VAR
    I,j,k    :    INT;
    v        :    REAL;
END_VAR
```

4) ตัวกระทำ (Operators and Expressions)

ตารางที่ 3.7 แสดงตัวกระทำในภาษาแบบโครงสร้าง

สัญลักษณ์	รายละเอียด
-	negation
NOT	Boolean complement
+ - * /	math operators
MOD	modulus operation
=	equal
<>	not equal
AND, &	Boolean AND
XOR	Boolean XOR
OR	Boolean OR

<u>ตัวอย่างเช่น</u>

```
Y := X+1.0;
y := a AND b;
v := (v1 + v2 + v3)/3
```

5) การกำหนดเงื่อนไขการทำงาน (Condition Statements)

<u>รูปแบบ</u>

```
IF เงื่อนไข THEN
    คำสั่ง;
END_IF;

IF เงื่อนไข THEN
    คำสั่งกรณีที่เงื่อนไขเป็นจริง;
ELSE
    คำสั่งกรณีที่เงื่อนไขเป็นเท็จ;
END_IF;
```

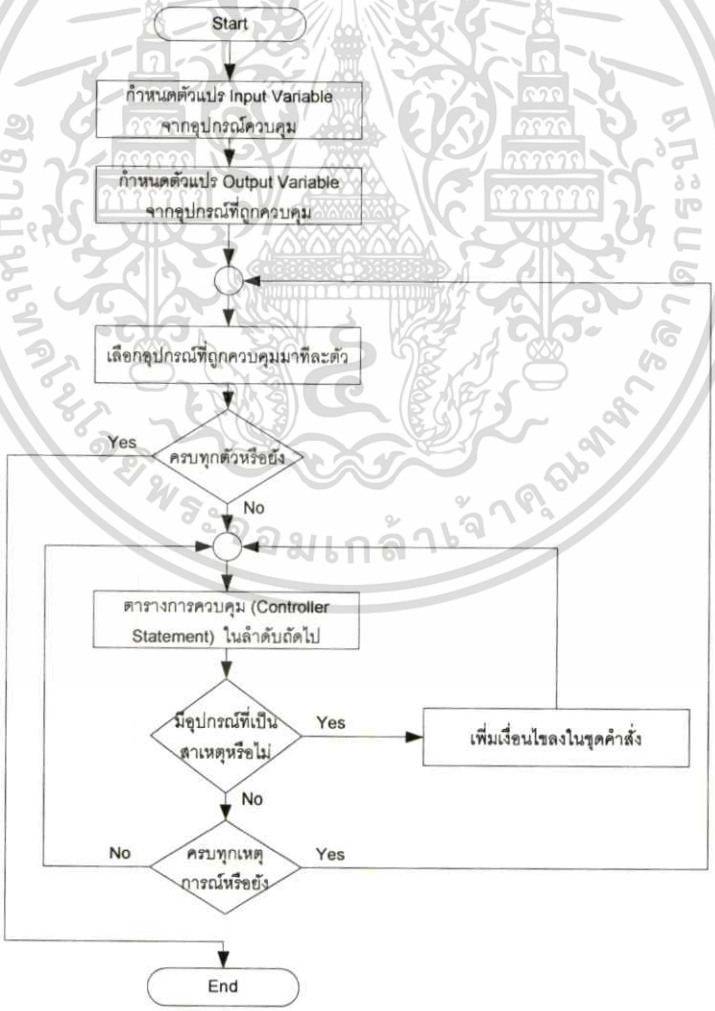
<u>ตัวอย่างเช่น</u>

```
IF  a > 100 THEN
    redlight := on;
ELSEIF  a > 50 THEN
```

```
        yellowlight := on;
    ELSE
        greenlight := on;
    END_IF;
```

3.7.1.2 หลักการแปลงชุดคำสั่งจากการสาธิตการควบคุมไปสู่ภาษาแบบโครงสร้าง

วิธีการแปลงชุดคำสั่งจากการสาธิตการควบคุมไปสู่ภาษาแบบโครงสร้างนั้น จะอาศัยรูปแบบของภาษาแบบโครงสร้างเป็นแนวทางในการแปลภาษา จะมีขั้นตอนในการแปลภาษาจากระบบโดยแบ่งเป็นสองส่วนหลัก ส่วนแรกคือขั้นตอนการกำหนดตัวแปร (Variable) ทั้งในส่วนของตัวแปรทางเข้าของข้อมูล (Input Variable) และตัวแปรทางออกของข้อมูล (Output Variable) ส่วนที่สองคือขั้นตอนการแปลเงื่อนไขการทำงานในส่วนของโปรแกรมตัวควบคุมให้เป็นรูปแบบเงื่อนไขในรูปแบบของภาษาแบบโครงสร้าง ซึ่งแสดงเป็นแผนผังขั้นตอนการแปลภาษาได้ดังนี้



รูปที่ 3.27 แสดงแผนผังขั้นตอนการแปลภาษาแบบโครงสร้าง

เพื่อให้สามารถเข้าใจการแปลภาษารูปแบบของการบันทึกการกระทำของผู้ใช้ให้อยู่ในรูปแบบของภาษาแบบโครงสร้างได้ง่ายขึ้น จึงขออธิบายโดยใช้กรณีศึกษาเครื่องปั๊มน้ำอัตโนมัติที่กล่าวมาแล้วข้างต้นมาประกอบการอธิบายดังนี้

1) การกำหนดตัวแปรให้กับอุปกรณ์ที่ทำหน้าที่ควบคุมนั้น (Input device) เราสามารถกำหนดเป็นตัวแปรทางเข้าของข้อมูล (Input Variable) ในรูปแบบของภาษาแบบโครงสร้างจากอุปกรณ์ที่ทำหน้าที่ควบคุม เช่น Sensor1 และ Sensor2 เป็นตัวลูกลอยที่ใช้วัดระดับน้ำ โดยมีคุณสมบัติในการบอกสถานะของน้ำว่าถึงจุดที่ติดตั้งลูกลอยแล้วหรือยัง ซึ่งจะให้ผลลัพธ์เป็นสถานะปิดและเปิด (State = On/Off) ดังนั้นการกำหนดตัวแปรที่มีคุณลักษณะของอุปกรณ์ที่เป็นตัวบอกสถานะเช่นนี้ เราจะใช้ประเภทของตัวแปร Boolean และสามารถเขียนรูปแบบของภาษาแบบโครงสร้างได้ดังนี้

```
INPUT_VAR
    Sensor1: BOOL;
    Sensor2: BOOL;
END_VAR
```

2) การกำหนดตัวแปรให้กับอุปกรณ์ที่ทำหน้าที่ถูกควบคุมนั้น (Output device) เราสามารถกำหนดเป็นตัวแปรทางออกของข้อมูล (Output Variable) ในรูปแบบของภาษาแบบโครงสร้างจากอุปกรณ์ที่ถูกควบคุม เช่น Pump1 เป็นอุปกรณ์ที่ถูกควบคุมจากอุปกรณ์พีแอลซี ซึ่งทำงานตามเงื่อนไขตามสถานการณ์ต่างๆ โดยที่ Pump1 จะมีคุณสมบัติที่ให้ผลลัพธ์เป็นสถานะปิดและเปิด (State = On/Off) ดังนั้นการกำหนดตัวแปรที่มีคุณลักษณะของอุปกรณ์ที่เป็นตัวบอกสถานะเช่นนี้ เราจะใช้ประเภทของตัวแปร Boolean และสามารถเขียนรูปแบบของภาษาแบบโครงสร้างได้ดังนี้

```
OUTPUT_VAR
    Pump1: BOOL;
END_VAR
```

3) การแปลงโปรแกรมตัวควบคุม (Controller programming) ไปสู่ชุดคำสั่งของภาษาแบบโครงสร้างนั้น จะพิจารณาจากการเปลี่ยนแปลงในสถานะของอุปกรณ์ควบคุม (Sensor1 และ Sensor2) ดังกล่าว ที่ส่งผลต่อสถานะอุปกรณ์ที่ถูกควบคุม (Pump1) สามารถสรุปตารางการเปลี่ยนแปลงในสถานะที่บันทึกลงในตารางการควบคุม (Controller Statement) และตารางสาเหตุ (Input Cause) ได้ดังนี้

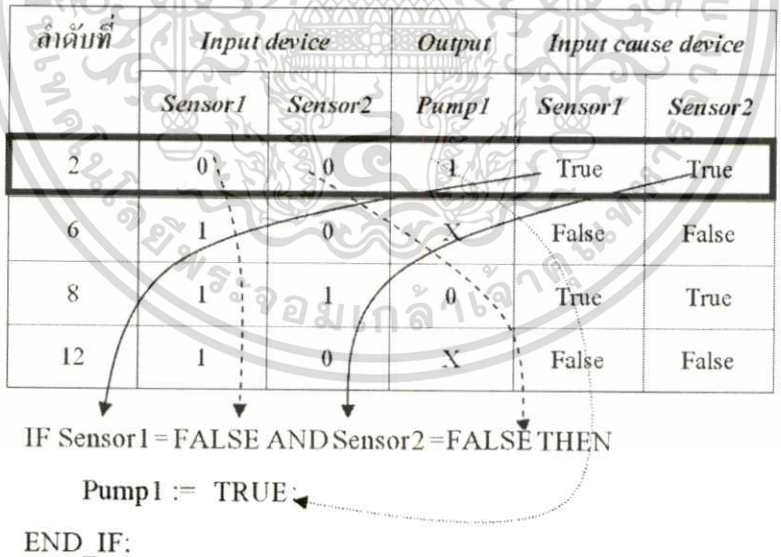
ตารางที่ 3.8 แสดงการบันทึกเหตุการณ์ที่ส่งผลต่อสถานะของ Pump1 ในตารางสาเหตุ

ลำดับที่	*Input device*		*Output*	*Input cause device*	
	Sensor1	*Sensor2*	*Pump1*	*Sensor1*	*Sensor2*
2	0	0	1	True	True
6	1	0	X	False	False
8	1	1	0	True	True
12	1	0	X	False	False

0 = State Off, 1=State On, X=User Ignore.

ตารางความจริงของสถานะทั้งหมดในตารางที่ 3.8 สามารถเขียนชุดคำสั่งในการควบคุมการเปิดและปิด Pump1 ที่สัมพันธ์กับสถานะของอุปกรณ์ควบคุมคือ Sensor1 และ Sensor2 ได้ดังนี้

ก) ในลำดับที่สองเป็นเหตุการณ์ที่ทำให้ Pump1 เปิด โดยอุปกรณ์ที่ส่งผล (Input cause device) ทำให้ Pump1 (Output device) เปิด คือ Sensor1, Sensor2 (Input device) โดยที่ทั้งสองอุปกรณ์มีสถานะปิดทั้งคู่



รูปที่ 3.28 แสดงเหตุการณ์ที่ทำให้ Pump1 เปิดและแสดงอุปกรณ์ที่ส่งผลต่อเหตุการณ์นี้

จากรูปที่ 3.28 สามารถเขียนภาษาแบบโครงสร้างได้ดังนี้

```
IF Sensor1 = FALSE AND Sensor2 = FALSE THEN
    Pump1 :=TRUE;
END_IF;
```

ข) ในลำดับที่แปดเป็นเหตุการณ์ที่ทำให้ Pump1 ปิด โดยอุปกรณ์ที่ส่งผล (Input cause device) ทำให้ Pump1 (Output device) เปิด คือ Sensor1, Sensor2 (Input device) โดยที่ทั้งสองอุปกรณ์มีสถานะเปิดทั้งคู่

ลำดับที่	Input device		Output	Input cause device	
	Sensor1	Sensor2	Pump1	Sensor1	Sensor2
2	0	0	1	True	True
6	1	0	X	False	False
8	1	1	0	True	True
12	1	0	X	False	False

```
IF Sensor1 = TRUE  AND Sensor2 = TRUE  THEN
    Pump1 :=FALSE;
END_IF;
```

รูปที่ 3.29 แสดงเหตุการณ์ที่ทำให้ Pump1 ปิดและแสดงอุปกรณ์ที่ส่งผลต่อเหตุการณ์นี้

จากรูปที่ 3.29 เราสามารถเขียนภาษาแบบโครงสร้างได้ดังนี้

```
IF Sensor1 = TRUE AND Sensor2 = TRUE THEN
    Pump1 :=FALSE;
END_IF;
```

สรุปการแปลงจากชุดคำสั่งในโปรแกรมตัวควบคุมดังกล่าวนั้น สามารถแปลงสู่ภาษาแบบโครงสร้างได้ดังแสดงในตารางที่ 3.9

ตารางที่ 3.9 แสดงการแปลงจากโปรแกรมตัวควบคุมไปสู่ภาษาแบบโครงสร้าง

<table>
<tr><th>Scenario</th><th>ST generated code</th></tr>
<tr><td><u>Input device</u>
Sensor1;
Sensor2;</td><td>INPUT_VAR
Sensor1:BOOL;
Sensor2:BOOL;
END_VAR</td></tr>
<tr><td><u>Output device</u>
Pump1;</td><td>OUTPUT_VAR
Pump1:BOOL;
END_VAR</td></tr>
<tr><td>
<table>
<tr><th rowspan="2">ลำดับที่</th><th colspan="2">Input device</th><th>Output</th><th colspan="2">Input cause device</th></tr>
<tr><th>Sensor1</th><th>Sensor2</th><th>Pump1</th><th>Sensor1</th><th>Sensor2</th></tr>
<tr><td>2</td><td>0</td><td>0</td><td>1</td><td>True</td><td>True</td></tr>
</table>
</td><td>IF Sensor1 = FALSE AND Sensor2 = FALSE THEN
Pump1 :=TRUE;
END_IF;</td></tr>
<tr><td>
<table>
<tr><th rowspan="2">ลำดับที่</th><th colspan="2">Input device</th><th>Output</th><th colspan="2">Input cause device</th></tr>
<tr><th>Sensor1</th><th>Sensor2</th><th>Pump1</th><th>Sensor1</th><th>Sensor2</th></tr>
<tr><td>8</td><td>1</td><td>1</td><td>0</td><td>True</td><td>True</td></tr>
</table>
</td><td>IF Sensor1 = TRUE AND Sensor2 = TRUE THEN
Pump1 :=FALSE;
END_IF;</td></tr>
</table>

จากกรณีศึกษาเครื่องปั๊มน้ำอัตโนมัตินั้น เมื่อแปลงจากการกระทำของผู้ใช้ในส่วนของโปรแกรมตัวควบคุมไปสู่ภาษาแบบโครงสร้างจะได้ดังนี้

```
PROGRAM
VAR
    Pump1   : BOOL;
    Sensor1   : BOOL;
    Sensor2   : BOOL;
END_VAR

IF (Sensor1 = False AND Sensor2 = False) THEN
   Pump1 := TRUE;
END_IF;

IF (Sensor1 = True AND Sensor2 = True) THEN
   Pump1 := FALSE;
END_IF;

END_PROGRAM
```

รูปที่ 3.30 แสดงภาษาแบบโครงสร้างที่ได้จากโปรแกรมตัวควบคุม

3.7.2 การแปลง User action สู่ภาษานีโมนิค

ภาษานีโมนิคจะเป็นภาษาที่เขียนอยู่ในรูปของข้อความ และมีลักษณะคล้ายกับภาษาแอสเซมบลี และภาษาเครื่อง ซึ่งภายในหนึ่งคำสั่งควบคุมจะประกอบด้วย ส่วนปฏิบัติการ (Operator) และส่วนที่ถูกดำเนินการ (Operand) การแปลง User action สู่ภาษานีโมนิคนั้นจะต้องแปลง User action เสียก่อนจากนั้นอาศัยหลักการของภาษาแบบโครงสร้างแปลงสู่ภาษานีโมนิคอีกครั้ง

ตารางที่ 3.10 เปรียบเทียบภาษานีโมนิคกับภาษาแบบโครงสร้าง

Mnemonic	*Structure Text*
LOAD (LD)	IF Variable1 = TRUE THEN
LOAD NOT (LD - NOT)	IF NOT Variable1 = FALSE THEN
AND (AND)	IF Variable1 = TRUE AND Variable2 = TRUE THEN
AND NOT (AND - NOT)	IF Variable1 = TRUE AND NOT Variable2 = FALSE THEN
OR (OR)	IF Variable1 = TRUE OR Variable2 = TRUE THEN
OR NOT (OR - NOT)	IF Variable1 = TRUE OR NOT Variable2 = FALSE THEN
SET (SET)	Variable1 := TRUE;
RESET (RST)	Variable1 := FALSE;
AND LOAD (AND - LD)	IF (Variable1 = TRUE OR Variable2 = TRUE) AND (Variable1 = TRUE OR Variable2 = TRUE) THEN
OR LOAD (OR - LD)	IF (Variable1 = TRUE AND Variable2 = TRUE) OR (Variable1 = TRUE AND Variable2 = TRUE) THEN
END	END_PROGRAM

จากกรณีศึกษาเครื่องปั๊มน้ำอัตโนมัตินั้น เมื่อแปลงจากการกระทำของผู้ใช้ในส่วนของโปรแกรมตัวควบคุมไปสู่ภาษาแบบโครงสร้างและแปลงสู่ภาษานีโมนิคจะได้ดังนี้

ตารางที่ 3.11 กรณีศึกษาเครื่องปั๊มน้ำอัตโนมัติแปลงสู่ภาษานีโมนิค

Structure Text	*Mnemonic*	
	Command	*Variable*
PROGRAM VAR Pump1 : BOOL; Sensor1 : BOOL; Sensor2 : BOOL; END_VAR		
IF (Sensor1 = False AND Sensor2 = False) THEN	LOAD NOT AND NOT	Sensor1 Sensor2
Pump1 := TRUE; END_IF;	SET	Pump1
IF (Sensor1 = True AND Sensor2 = True) THEN	LOAD AND	Sensor1 Sensor2
Pump1 := FALSE; END_IF;	RST	Pump1
END_PROGRAM	END	

ในภาษานีโมนิค ตัวแปร (Variable) นั้นจะต้องแปลงให้เป็นหมายเลขช่องสัญญาณ (Port number) ของอุปกรณ์พีแอลซีเสียก่อน จากกรณีศึกษาเครื่องปั๊มน้ำอัตโนมัตินั้นหากต่ออุปกรณ์ Sensor1 กับช่องสัญญาณหมายเลข P0000 อุปกรณ์ Sensor1 กับช่องสัญญาณหมายเลข P0001 และอุปกรณ์ Pump1 กับช่องสัญญาณหมายเลข P0020 จะสามารเขียนภาษานีโมนิคได้ดังนี้

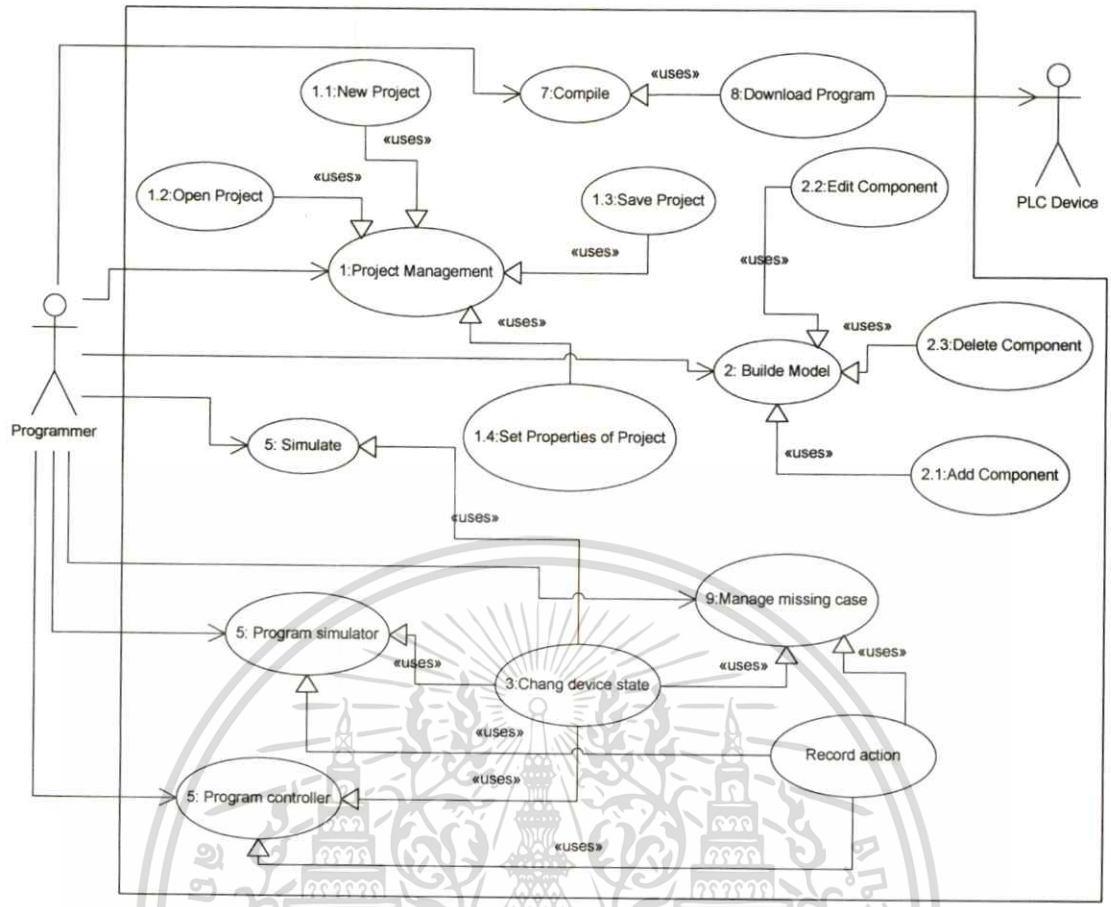
```
LOAD NOT    P0000
AND NOT     P0001
SET         P0020
LOAD        P0000
AND         P0001
RST         P0020
END
```

รูปที่ 3.31 แสดงภาษานีโมนิคที่ได้จากโปรแกรมตัวควบคุม

3.8 เครื่องมือการโปรแกรม (Programming Tools)

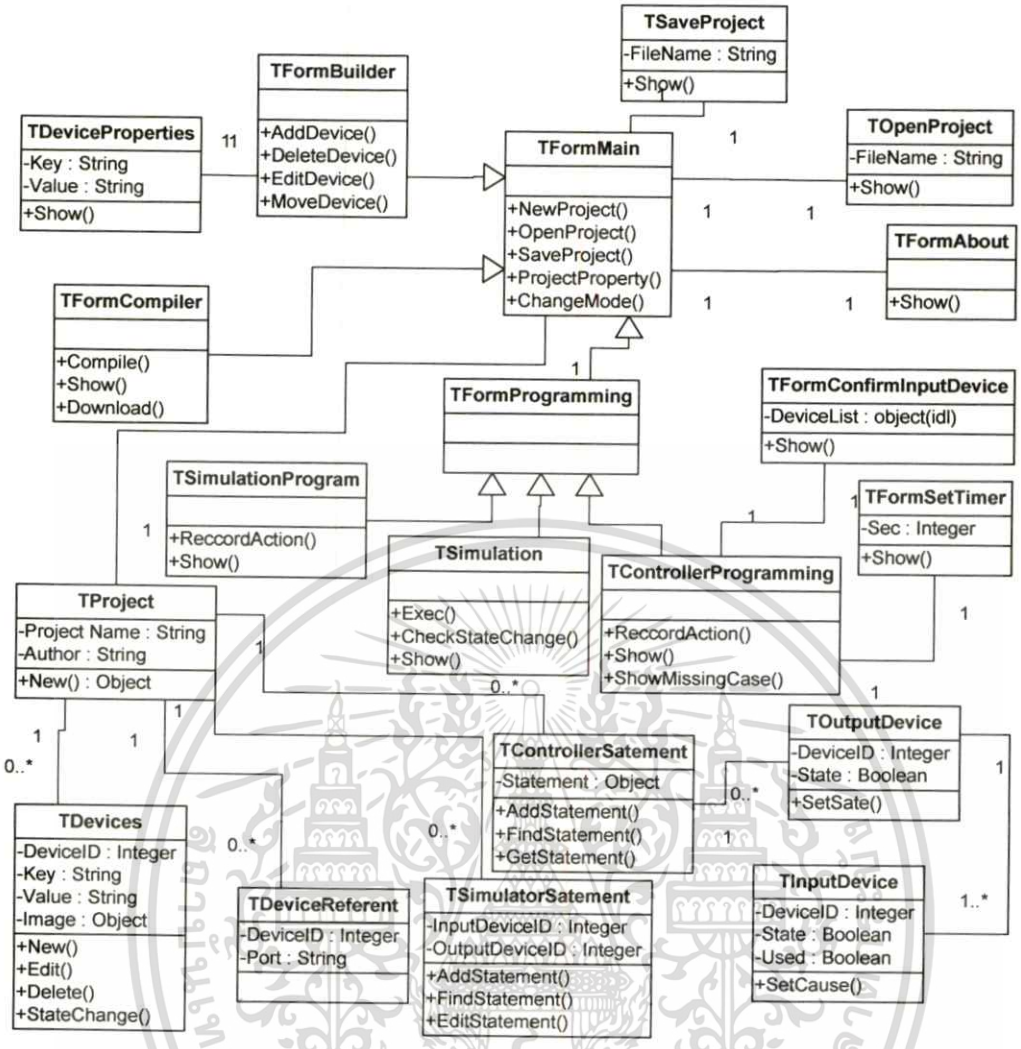
การออกแบบเครื่องมือการโปรแกรมนั้น ได้พิจารณาจากหลักการทำงานของเทคนิคการโปรแกรมอุปกรณ์พีแอลซีโดยใช้การสาธิตการควบคุมและการจำลองร่วมกันนั้น จะเริ่มต้นจากขั้นตอนการสร้างแบบจำลอง (System initialization mode or Model Builder mode) การออกแบบในส่วนนี้อาศัยหลักการของ Visual programming เมื่อสร้างแบบจำลองเสร็จแล้วเราจะนำแบบจำลองนั้นเข้าสู่ขั้นตอนการจำลองสถานการณ์เมื่อเกิดการเปลี่ยนสถานะของอุปกรณ์ในแบบจำลองและยังไม่ได้มีการกำหนดการโปรแกรมเอาไว้ รวมทั้งผู้ใช้สามารถเลือกที่จะเข้าสู่ขั้นตอนการโปรแกรมได้เอง ขั้นตอนของการโปรแกรมนั้นแบ่งออกเป็นสองส่วน ส่วนแรกเป็นการโปรแกรมแบบจำลองส่วนที่สองคือโปรแกรมตัวควบคุม เพื่อให้ระบบเลือกทำตามเงื่อนไขในแต่ละสถานการณ์ที่เกิดขึ้นซึ่งในสามขั้นตอนนี้จะทำสลับกันไปมา โดยในส่วนนี้จะใช้หลักการของ Iterative Programming จนมีการโปรแกรมครบทุกเหตุการณ์ที่จะเกิดขึ้นกับแบบจำลอง เมื่อเราทำการโปรแกรมเสร็จเรียบร้อย ขั้นตอนสุดท้ายจะเป็นขั้นตอนของการแปลงการกระทำของผู้ใช้ในส่วนของระบบการควบคุมไปสู่ภาษาที่ใช้ในการโปรแกรมอุปกรณ์พีแอลซี

จากที่กล่าวข้างต้นนั้น เราสามารถแสดงรายละเอียดโดยใช้ แผนภาพที่แสดงการทำงานของผู้ใช้ระบบ และความสัมพันธ์กับระบบย่อยภายในระบบใหญ่ (Use Case Diagram) ซึ่งเป็นเอกสารที่บรรยายถึงลำดับของเหตุการณ์ (Event) ที่ผู้ใช้ (Actor) ปฏิบัติตามกระบวนการทำงาน (Process) กับเครื่องมือการโปรแกรมเพื่อเป็นการนำเสนอมุมมองการโต้ตอบกันระหว่างผู้ใช้กับเครื่องมือการโปรแกรม ในการทำงานกระบวนการหนึ่งๆ ภายใต้การใช้งานเครื่องมือการโปรแกรมได้ดังนี้



รูปที่ 3.32 แสดง Use Case Diagram ของเครื่องมือการโปรแกรม (Programming Tools)

จากแผนภาพที่แสดงการทำงานของผู้ใช้ระบบ และความสัมพันธ์กับระบบย่อยภายในระบบใหญ่เราสามารถนำมาเขียนแผนภาพแสดงความสัมพันธ์ของแม่แบบของวัตถุ (Class Diagram) ซึ่งเป็นแผนภาพที่ใช้แสดงแม่แบบของวัตถุ และความสัมพันธ์ในแง่ต่างๆ (Relation) ระหว่างแม่แบบของวัตถุเหล่านั้น ซึ่งความสัมพันธ์ที่กล่าวถึงในแผนภาพแสดงความสัมพันธ์ของแม่แบบของวัตถุนี้ถือเป็นความสัมพันธ์เชิงสถิต (Static Relationship) หมายถึง ความสัมพันธ์ที่มีอยู่แล้วเป็นปกติในระหว่างแม่แบบของวัตถุต่างๆ ของระบบ เพื่อใช้เป็นไกด์ไลน์ ของการนำไปใช้ในการสร้าง Application Software ของเครื่องมือการโดยภายในแผนภาพแสดงความสัมพันธ์ของแม่แบบของวัตถุ ดังกล่าวจะมีการอธิบายรายละเอียดของความสัมพันธ์ระหว่างแม่แบบของวัตถุ และรายละเอียดที่อยู่ภายในแม่แบบของวัตถุได้แก่ คุณลักษณะ (Property) และพฤติกรรม (Method) ได้ดังนี้

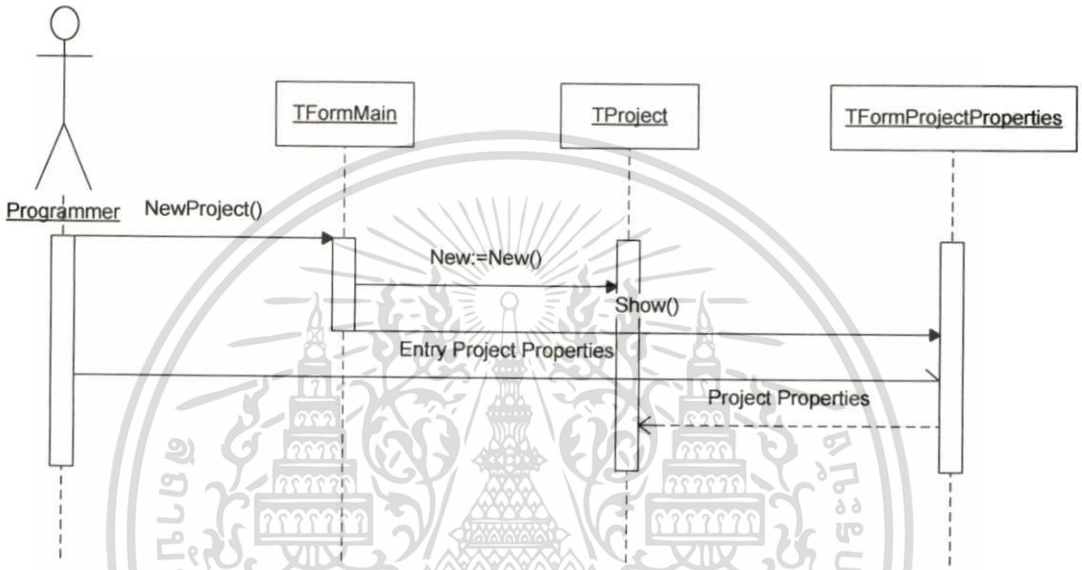


รูปที่ 3.33 แสดง Class Diagram ของเครื่องมือการโปรแกรม (Programming Tools)

แผนภาพที่แสดงการทำงานของผู้ใช้ระบบ และความสัมพันธ์กับระบบย่อยภายในระบบใหญ่และแผนภาพแสดงความสัมพันธ์ของแม่แบบของวัตถุเราสามารถนำมาเขียนเป็นแผนผังจำลองกระบวนการที่ทำให้เกิดกิจกรรมของระบบ (Sequence Diagram) ซึ่งเป็นบทบรรยายรายละเอียดการทำงานภายในแผนภาพที่แสดงการทำงานใดๆ ด้วยภาพ ซึ่งจะแสดงถึงการโต้ตอบกันระหว่างผู้ใช้กับเครื่องมือการโปรแกรมในลักษณะของการร้องขอบริการและการตอบสนองตามลำดับของเหตุการณ์ก่อน-หลัง โดยที่ตัวแผนผังจำลองกระบวนการที่ทำให้เกิดกิจกรรมของระบบจะแสดงถึงกลุ่มของวัตถุต่างๆ ที่ต้องทำงานร่วมกัน (Collaboration) เพื่อให้บรรลุเป้าหมายการทำงานของแผนภาพที่แสดงการทำงานหนึ่งๆ ทั้งนี้การโต้ตอบระหว่างวัตถุ (Object) หรือ ผู้ใช้กับวัตถุจะอยู่ในลักษณะของผู้ส่งข่าว (Sender Object) กับผู้รับข่าวสาร (Received Object) ตามลำดับ ส่วนข่าวสาร (Message) ที่จะส่งหากันนั้นจะใช้เส้นลูกศรและมีการกำหนดชื่อกำกับเส้นลูกศรดังกล่าวด้วย เพื่อแสดงให้เห็นความหมายของข่าวสารดังกล่าว ว่าจะให้วัตถุที่รับข่าวสารนั้นทำหรือปฏิบัติอะไร เพื่อ

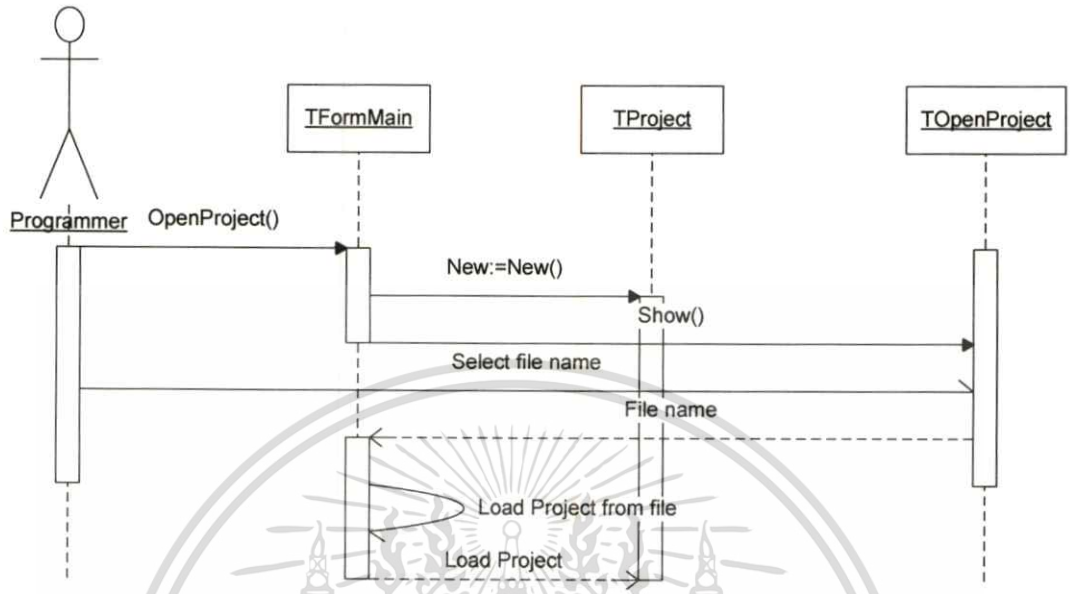
แสดงหรือบรรยายการทำงานโต้ตอบกันระหว่างผู้ใช้ระบบกับเครื่องมือการโปรแกรมในลักษณะของภาพหรือสัญลักษณ์ ในลักษณะตามลำดับเหตุการณ์ก่อน-หลัง ซึ่งจะมีประโยชน์ในการกำหนดหรือสร้างแม่แบบของกลุ่มวัตถุที่มีในแผนผังจำลองกระบวนการที่ทำให้เกิดกิจกรรมของระบบตลอดจนพฤติกรรม (Behavior or Method) ที่กลุ่มของวัตถุหรือแม่แบบโดยแสดงตามแผนภาพที่แสดงการทำงานของผู้ใช้ระบบ และความสัมพันธ์กับระบบย่อยภายในระบบใหญ่ได้ดังนี้

1) Sequence Diagram 1.1: แผนภาพแสดงลำดับการสร้างโปรเจคใหม่ (New Project)



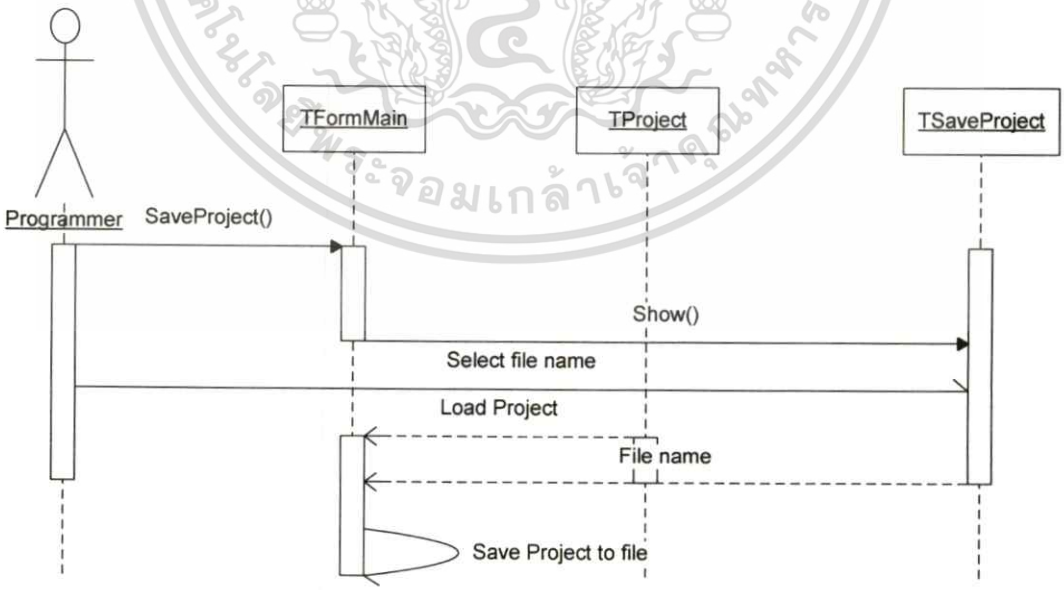
รูปที่ 3.34 แสดงแผนภาพลำดับการสร้างโปรเจคใหม่

2) Sequence Diagram 1.2: แผนภาพแสดงลำดับการเปิดโปรเจคขึ้นมาใช้งาน (Open Project)



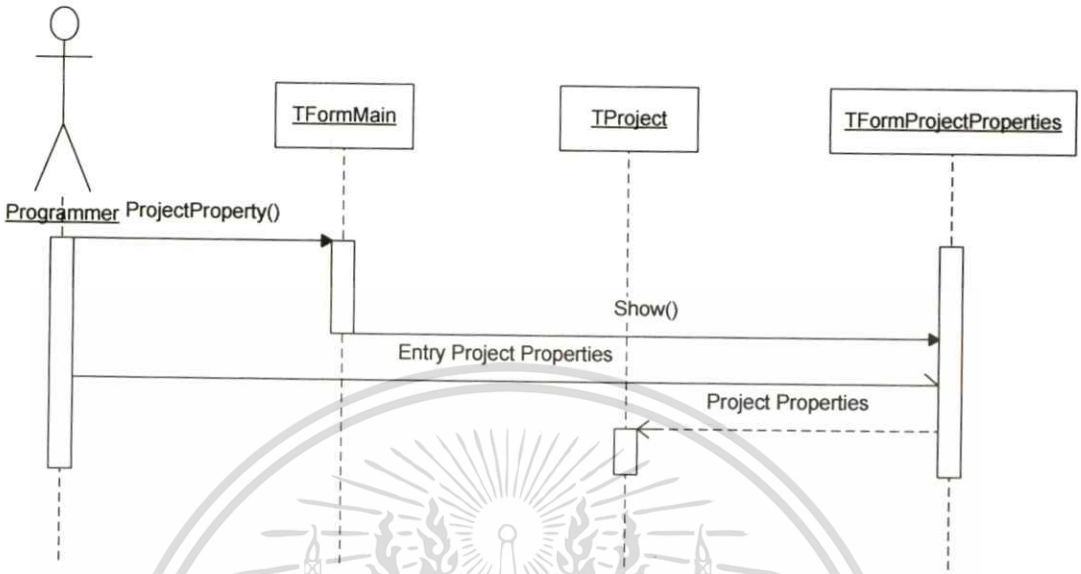
รูปที่ 3.35 แสดงแผนภาพลำดับการเปิดโปรเจคขึ้นมาใช้งาน

3) Sequence Diagram 1.3: แผนภาพแสดงลำดับการจัดเก็บโปรเจค (Save Project)



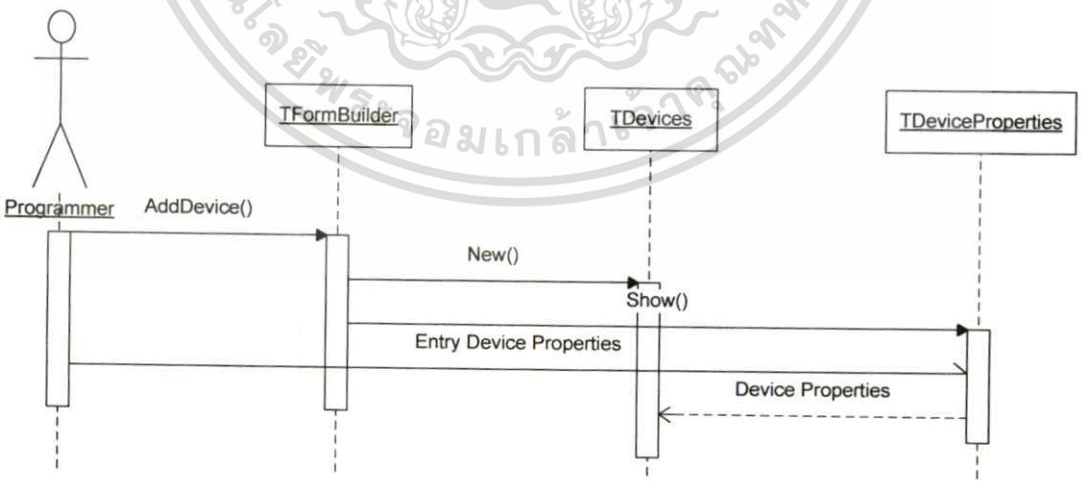
รูปที่ 3.36 แสดงแผนภาพลำดับการจัดเก็บโปรเจค

4) Sequence Diagram 1.4: แผนภาพแสดงลำดับการตั้งค่าข้อมูลของโปรเจค (Set Properties of Project)



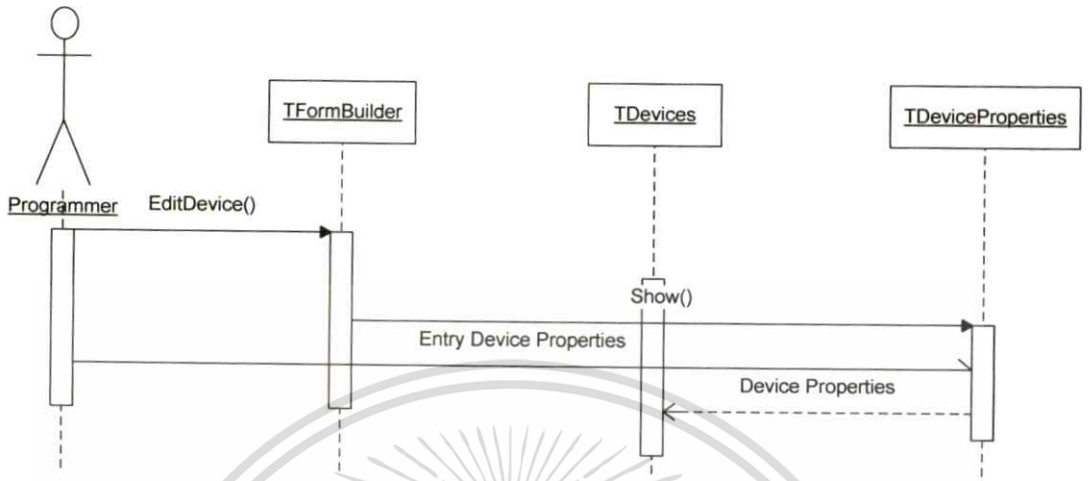
รูปที่ 3.37 แสดงแผนภาพลำดับการตั้งค่าข้อมูลของโปรเจค

5) Sequence Diagram 2.1: แผนภาพแสดงลำดับการเพิ่มอุปกรณ์เพื่อเป็นส่วนประกอบของเครื่องจักร (Add Component)



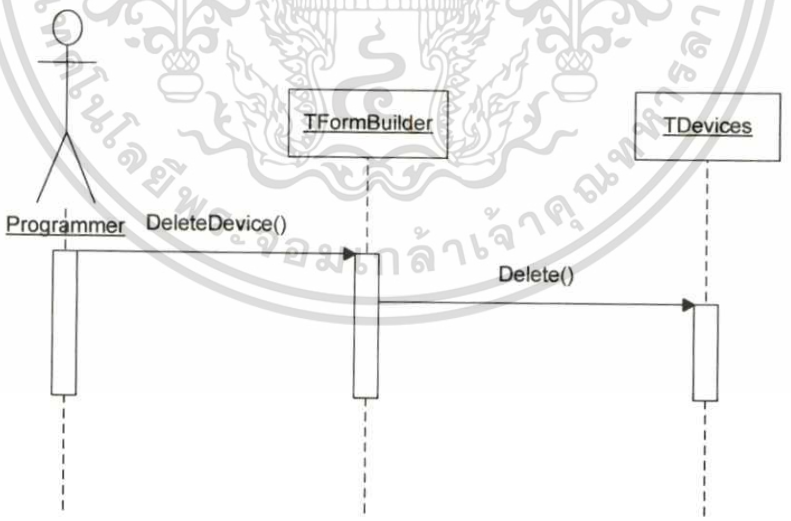
รูปที่ 3.38 แสดงแผนภาพลำดับการเพิ่มอุปกรณ์เพื่อเป็นส่วนประกอบของเครื่องจักร

6) Sequence Diagram 2.2: แผนภาพแสดงลำดับการแก้ไขคุณสมบัติของอุปกรณ์ (Edit Component)



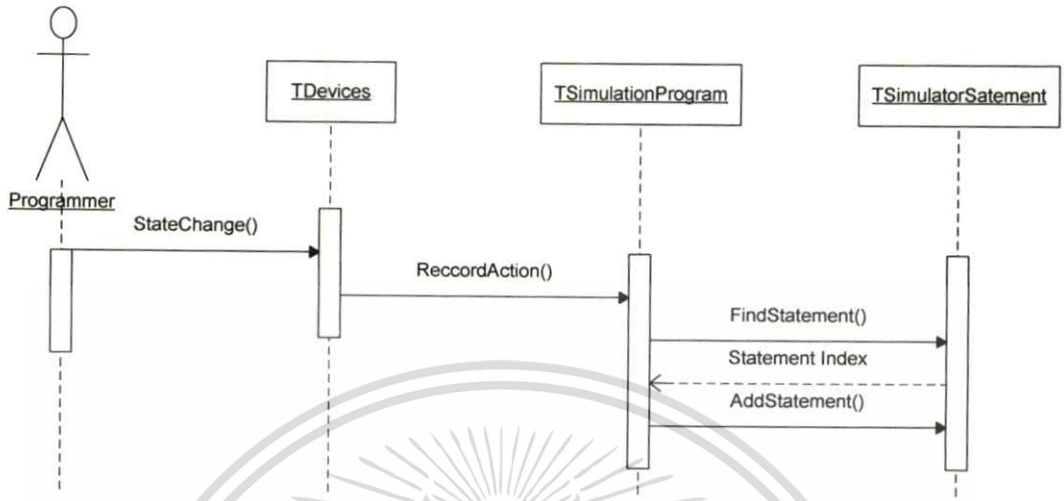
รูปที่ 3.39 แสดงแผนภาพลำดับการแก้ไขคุณสมบัติของอุปกรณ์

7) Sequence Diagram 2.3: แผนภาพแสดงลำดับการลบอุปกรณ์ (Delete Component)



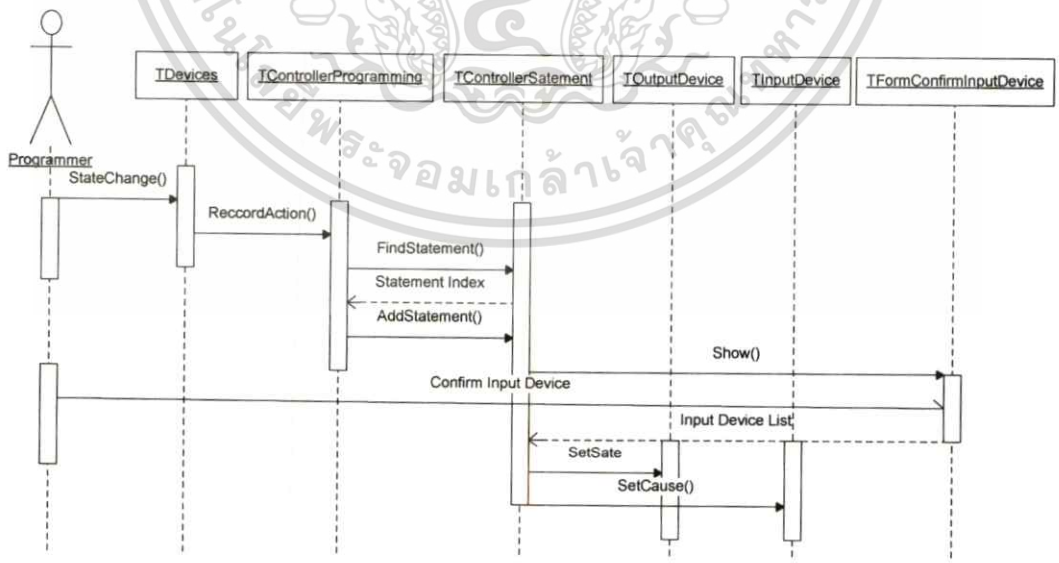
รูปที่ 3.40 แสดงแผนภาพลำดับการลบอุปกรณ์

8) Sequence Diagram 3: แผนภาพแสดงลำดับการเปลี่ยนสถานะของอุปกรณ์ในโหมดการโปรแกรมแบบจำลอง (Change Device State of Simulator Programming)



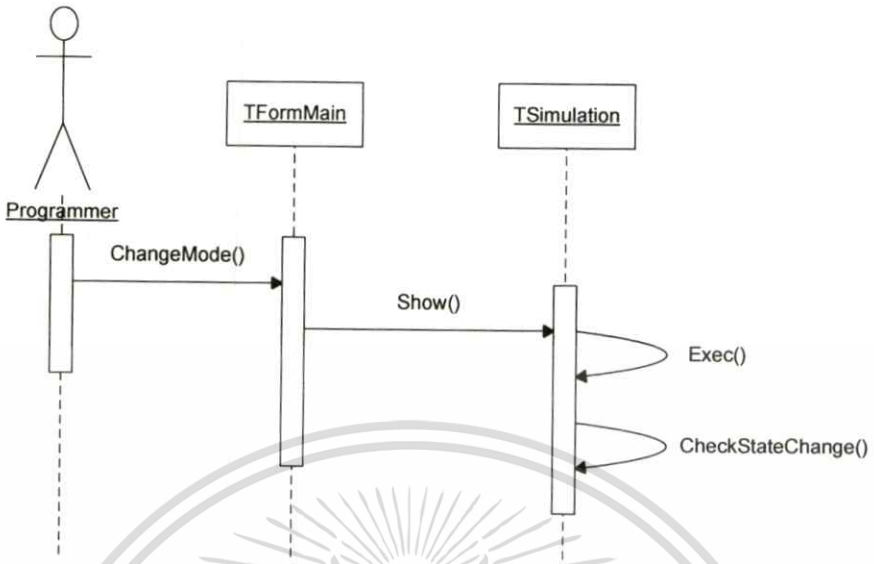
รูปที่ 3.41 แสดงแผนภาพลำดับการเปลี่ยนสถานะของอุปกรณ์ในโหมดการโปรแกรมแบบจำลอง

9) Sequence Diagram 4: แผนภาพแสดงลำดับการเปลี่ยนสถานะของอุปกรณ์ในโหมดการโปรแกรมตัวควบคุม (Change Device State of Controller Programming)



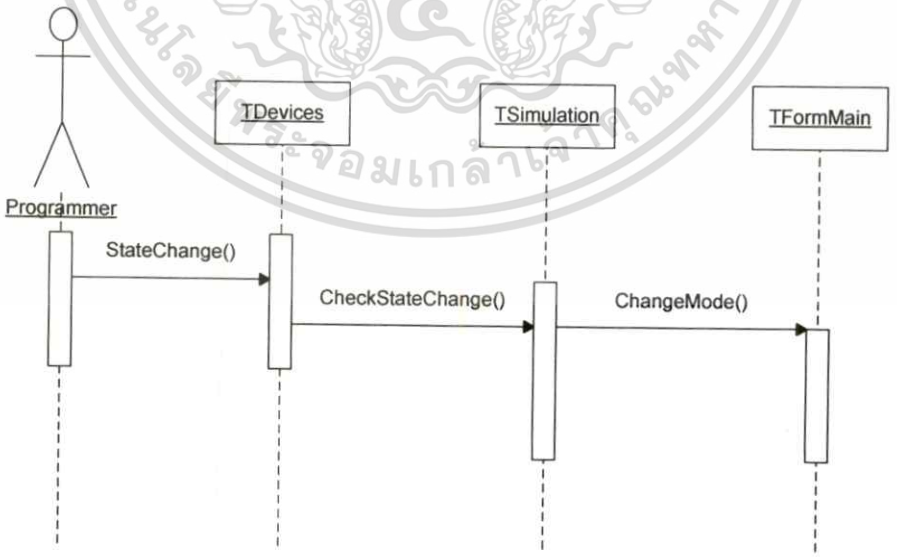
รูปที่ 3.42 แสดงแผนภาพลำดับการเปลี่ยนสถานะของอุปกรณ์ในโหมดการโปรแกรมตัวควบคุม

10) Sequence Diagram 5: แผนภาพแสดงลำดับการจำลองสถานการณ์ (Simulation)



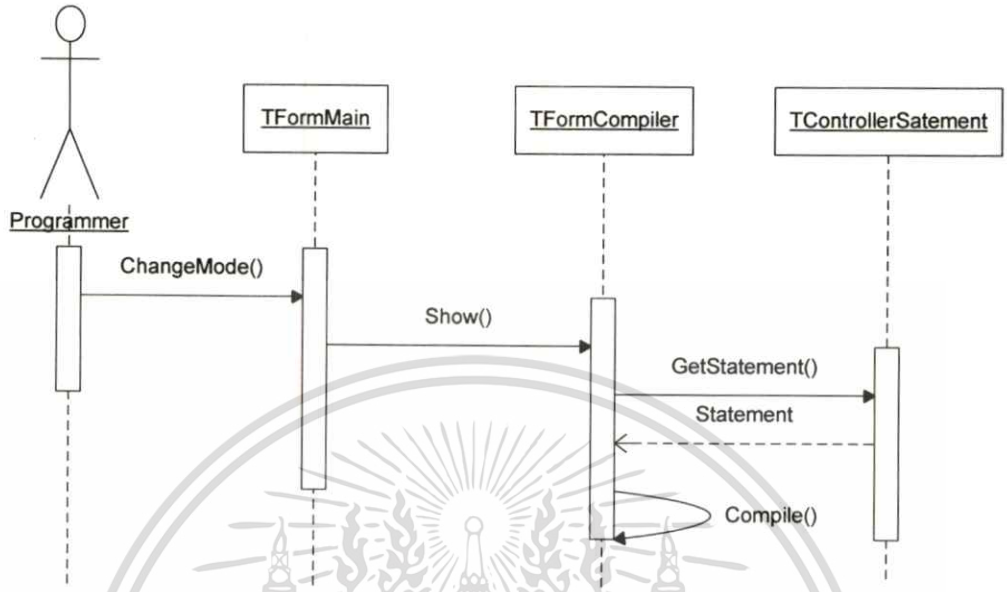
รูปที่ 3.43 แสดงแผนภาพลำดับการจำลองสถานการณ์

11) Sequence Diagram 6: แผนภาพแสดงลำดับการเปลี่ยนสถานะของอุปกรณ์ในโหมดการจำลองสถานการณ์ (Change Device State of Simulation)



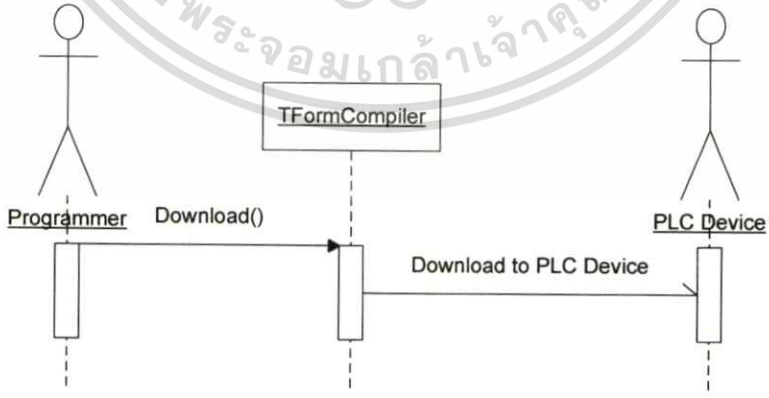
รูปที่ 3.44 แสดงแผนภาพลำดับการเปลี่ยนสถานะของอุปกรณ์ในโหมดการจำลองสถานการณ์

12) Sequence Diagram 7: แผนภาพแสดงลำดับการแปลงการกระทำของผู้ใช้ไปสู่ภาษา Mnemonic (Compiler)



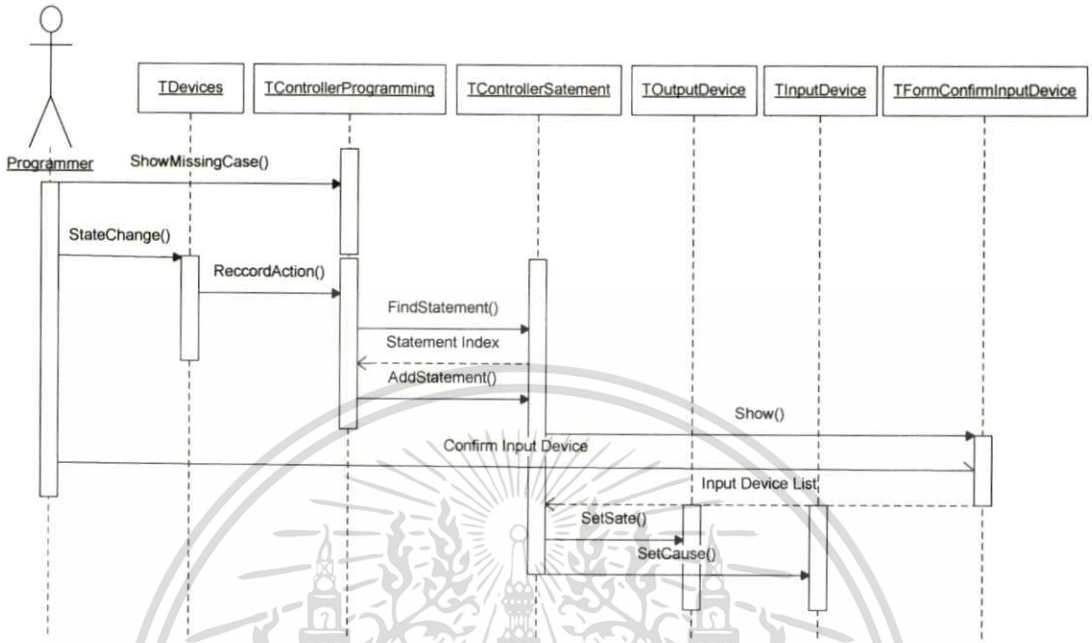
รูปที่ 3.45 แสดงแผนภาพลำดับการแปลงการกระทำของผู้ใช้ไปสู่ภาษานีโมนิค

13) Sequence Diagram 8: แผนภาพแสดงลำดับการส่งชุดคำสั่งที่ผ่านการแปลภาษาแล้วไปยังอุปกรณ์พีแอลซี (Download Program)



รูปที่ 3.46 แสดงแผนภาพลำดับการส่งชุดคำสั่งที่ผ่านการแปลภาษาแล้วไปยังอุปกรณ์พีแอลซี

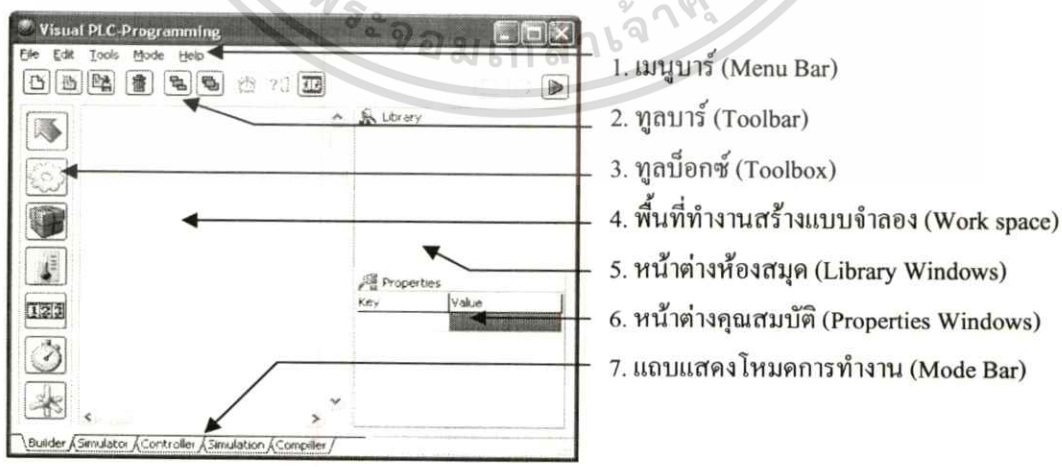
14) Sequence Diagram 9: แผนภาพแสดงลำดับการจัดการกับเหตุการณ์ที่ยังไม่ได้โปรแกรมตัวควบคุม (Missing case managements)



รูปที่ 3.47 แสดงแผนภาพลำดับการจัดการกับเหตุการณ์ที่ยังไม่ได้โปรแกรมตัวควบคุม

3.8.1 การออกแบบเครื่องมือในการโปรแกรม

การออกแบบเครื่องมือในการโปรแกรมนั้นได้นำหลักการของการโปรแกรมด้วยภาพมาใช้ในการออกแบบ ซึ่งมีองค์ประกอบดังนี้



รูปที่ 3.48 แสดงรายละเอียดของส่วนประกอบต่างๆ ของเครื่องมือในการโปรแกรม

จากรูปที่ 3.48 มีรายละเอียดของส่วนประกอบต่างๆ ของสภาวะแวดล้อมในการเขียนโปรแกรมดังต่อไปนี้

1) เมนูบาร์ (Menu Bar) คือแถบที่ใช้เก็บคำสั่งทั้งหมดในโปรแกรม ใช้ควบคุมการทำงานของเครื่องมือการโปรแกรมเพื่อสั่งงานหรือจัดการสิ่งต่างๆ เราสามารถเลือกคำสั่งจากรายการเหล่านี้ได้โดยกดเมาส์ที่รายการใดรายการหนึ่ง เช่น การสร้างโปรเจค การเปิดโปรเจค การกำหนดค่าคุณสมบัติของโปรเจค การเพิ่ม ลบ วัตถุในแบบจำลอง หรือการสั่งให้ทำงานในขั้นตอนต่างๆ เป็นต้น

2) ทูลบาร์ (Toolbar) คือแถบที่ใช้เก็บปุ่มคำสั่ง ซึ่งปุ่มเหล่านี้จะแทนคำสั่งที่ถูกเรียกใช้งานบ่อยๆ เพื่อให้ทำงานได้อย่างรวดเร็ว

3) ทูลบ็อกซ์ (Toolbox) เปรียบเหมือนกล่องเครื่องมือที่เก็บอุปกรณ์พื้นฐาน (Object Based Simulation) เพื่อให้ผู้ใช้งานสามารถเลือกอุปกรณ์ต่างๆ นำไปวางลงบนพื้นที่สร้างแบบจำลอง (Work space) ได้ เป็นการสร้างแบบจำลองของเครื่องจักรต่างๆ โดยที่อุปกรณ์พื้นฐานจะไม่สามารถแสดงถึงอุปกรณ์ใดอุปกรณ์หนึ่งได้เลยทันที ผู้ใช้จำเป็นต้องกำหนดคุณสมบัติจากอุปกรณ์พื้นฐานให้เป็นไปตามอุปกรณ์ที่เราต้องการ เช่น หลอดไฟฟ้า สวิทช์ไฟฟ้า

4) พื้นที่ทำงานสร้างแบบจำลอง (Work space) พื้นที่ทำงานใช้สำหรับการสร้างแบบจำลองและโปรแกรมการทำงานรวมถึงการแสดงการทำงานของแบบจำลองนั้นๆ

5) หน้าต่างห้องสมุด (Library Windows) เปรียบเหมือนกล่องเครื่องมือที่บรรจุอุปกรณ์สำเร็จรูปที่ถูกกำหนดคุณสมบัติให้เป็นอุปกรณ์ใดอุปกรณ์ไว้แล้ว เช่น มอเตอร์ ปั๊มน้ำฯลฯ เพื่อความสะดวกรวดเร็วในการนำไปใช้งาน ผู้ใช้สามารถนำไปใช้สร้างเครื่องจักรได้ทันที

6) หน้าต่างคุณสมบัติ (Properties Windows) ใช้สำหรับแสดงคุณสมบัติและใช้ในการเปลี่ยนแปลงคุณสมบัติของอุปกรณ์ที่ถูกเลือกอยู่ในขณะนั้น

7) แถบแสดงและเลือกโหมดการทำงาน (Mode Bar) คือส่วนที่ใช้แสดงโหมดการทำงานของเครื่องมือการโปรแกรมว่าอยู่ในโหมดใด รวมทั้งผู้ใช้สามารถเลือกที่จะเปลี่ยนโหมดการทำงานได้

3.8.2 การออกแบบการสร้างแบบจำลองโดยอาศัย Visual Programming Technique

การสร้างแบบจำลอง (Modeling or Model building) หมายถึง การแทนที่อุปกรณ์ต่างๆ ของเครื่องจักรจากวัตถุโดยแทนด้วยข้อมูลทางคอมพิวเตอร์ที่สามารถแสดงการเปลี่ยนแปลงค่าของข้อมูลของคอมพิวเตอร์ที่สามารถทำให้ผู้ใช้สังเกตเห็นการเปลี่ยนแปลงที่เกิดขึ้นจากแบบจำลองคอมพิวเตอร์ ทำให้สามารถตีความกลับไปเป็นพฤติกรรมที่เกิดขึ้นกับระบบจริงได้ การพัฒนาแบบจำลองระบบต้องเตรียมการหลายด้านเพื่อให้ได้แบบจำลองตามจุดประสงค์ และอีกส่วนที่สำคัญคือการสร้างความความสัมพันธ์ระหว่างวัตถุต่างๆ

3.8.3 การออกแบบแม่แบบพื้นฐาน

การออกแบบแม่แบบพื้นฐาน โดยพิจารณาถึงคุณสมบัติพื้นฐานของวัตถุ หรืออุปกรณ์ที่ประกอบขึ้นเป็นเครื่องจักรนั้น ว่าแต่วัตถุมีส่วนใดที่เหมือนกันและต่างกันบ้าง รวมถึงการพิจารณาความสัมพันธ์ในทางกายภาพระหว่างอุปกรณ์ต่างๆ ของเครื่องจักร เพื่อให้เราสามารถกำหนดอุปกรณ์พื้นฐาน หรือแม่แบบเพื่อเป็นแม่แบบให้กับอุปกรณ์ต่างๆ โดยที่วัตถุต่างๆ สามารถที่จะแสดงการเปลี่ยนแปลงค่าของข้อมูลของคอมพิวเตอร์ที่สามารถทำให้ผู้ใช้สังเกตเห็นการเปลี่ยนแปลงที่เกิดขึ้นจากแบบจำลองคอมพิวเตอร์ ทำให้สามารถตีความกลับไปเป็นพฤติกรรมที่เกิดขึ้นกับระบบจริงได้ รวมทั้งแสดงถึงความสัมพันธ์กันระหว่างวัตถุได้อีกด้วย จากหลักการที่กล่าวมาแล้วนั้น เรานำมากำหนดเป็นกฎเกณฑ์พื้นฐานของการแบ่งชนิดของอุปกรณ์พื้นฐานชนิดต่างๆ ในวิทยานิพนธ์ฉบับนี้ ยังได้คำนึงถึงกฎเกณฑ์ตามชนิดของอุปกรณ์ที่มีอยู่ในอุปกรณ์พีแอลซีเป็นหลัก ซึ่งในอุปกรณ์พีแอลซีนั้นจะแบ่งอุปกรณ์ออกเป็นสี่ชนิดหลัก คือ รีเลย์ ตัวจับเวลา ตัวนับจำนวน ตัวจดจำข้อมูล และเพิ่มเติมในส่วนของอุปกรณ์หีบห่อขึ้นมา พร้อมทั้งกำหนดคุณลักษณะเพิ่มเติมเพื่อให้ครอบคลุมลักษณะทางกายภาพของอุปกรณ์ต่างๆ ในการแสดงให้เห็นถึงการเปลี่ยนแปลงลักษณะรูปร่างของอุปกรณ์ เพื่อการสื่อสารกับผู้ใช้ในรูปแบบของภาพเคลื่อนไหวที่มีความใกล้เคียงกับอุปกรณ์จริงมากที่สุด

การออกแบบส่วนการสร้างแบบจำลองนั้น จะต้องอาศัยการแสดงภาพของเครื่องจักร ที่ประกอบด้วยชิ้นส่วนต่างๆ โดยแต่ละชิ้นส่วนนั้นหมายถึงอุปกรณ์หนึ่งตัวนั่นเอง และจะต้องมีการแสดงภาพที่สามารถทำให้ผู้ใช้สังเกตเห็นการเปลี่ยนแปลงที่เกิดขึ้นกับอุปกรณ์ได้ รวมถึงคุณลักษณะเฉพาะของอุปกรณ์ตัวนั้นๆ ฉะนั้นการสร้างอุปกรณ์ในวิธีการที่นำเสนอนี้ เราจะคำนึงถึงสองส่วนคือ ส่วนแรกการแสดงสถานะต่างๆ ของอุปกรณ์โดยใช้ภาพประกอบ ส่วนที่สองคือคุณสมบัติของอุปกรณ์ (Device Properties) โดยวิทยานิพนธ์ฉบับนี้ได้แบ่งแม่แบบพื้นฐานออกเป็นห้าชนิดคือ

รูปที่ 3.49 แสดงสัญลักษณ์ของอุปกรณ์ในแถบเครื่องมือ

1) อุปกรณ์ทั่วไป (General) ใช้แทนอุปกรณ์ทั่วไป เราจะแบ่งส่วนหลักๆ เป็นสองส่วนคือ ส่วนแรกคือส่วนที่ส่งผลต่อการควบคุมหรือมีผลต่อเงื่อนไขการทำงานของอุปกรณ์พีแอลซี เช่น สถานะ (State) แบ่งออกได้เป็นสองสถานะคือปิดและเปิด ส่วนที่สองคือส่วนที่มีผลต่อลักษณะทางกายภาพของอุปกรณ์ เช่น ค่าตัวแปร (Value) ภาพที่ใช้แสดงในแต่ละค่าของตัวแปร (Image 1, …, Image N) ฯลฯ ซึ่งค่าตัวแปรนี้จะส่งผลต่อการเปลี่ยนรูปร่างหรือรูปแบบการแสดงผลของอุปกรณ์นั้นๆ รวมทั้งอุปกรณ์ชนิดทั่วไปนี้ยังมีคุณสมบัติในการกระตุ้นหรือส่งผลกระทบกับอุปกรณ์อื่นได้ด้วย ไม่ว่าจะเป็นทางกายภาพกับอุปกรณ์อื่นในการโปรแกรมการทำงานของแบบจำลองหรือส่งผลต่อเงื่อนไขทางการควบคุมของขบวนการทำงานในการโปรแกรมตัวควบคุม การใช้งานอุปกรณ์ชนิดทั่วไปนี้ เช่น หลอดไฟฟ้า สวิทช์ไฟฟ้า สายพาน มอเตอร์ เป็นต้น สัญลักษณ์ที่ใช้ในแถบเครื่องมือคือรูปเฟือง ซึ่งหมายถึงชิ้นส่วนต่าง ๆ ของเครื่องจักร

รายการคุณสมบัติของอุปกรณ์ทั่วไป ดังแสดงในตารางที่ 3.12

ตารางที่ 3.12 รายการคุณสมบัติของอุปกรณ์ทั่วไป

คุณสมบัติ	รายละเอียด
AutoStateOff=False	เมื่อ State = 1 จะกลับเป็น 0 โดยอัตโนมัติหรือไม่ ตัวเลือกเป็น: True / False ตัวอย่างเช่น: สวิทช์แบบกดติดปล่อยดับ
CaptionLeft=0	ตำแหน่งของชื่ออุปกรณ์นับจากด้านซ้ายของอุปกรณ์
CaptionTop=-23	ตำแหน่งของชื่ออุปกรณ์นับจากด้านบนของอุปกรณ์
DefaultState=0	ค่าเริ่มต้นของสถานะ (State) ตัวเลือกเป็น:1 / 0
DefaultValue=0	ค่าเริ่มต้นของค่าตัวแปร (Value) ตัวเลือกเป็น: 0…65,535
DragLength=0,0	ระยะการ Drag mouse เมื่อต้องการเปลี่ยนแปลงค่าตัวแปร (Value) ตัวเลือกเป็น: 10, 200 (หมายถึงเริ่มจากตำแหน่งที่ 10 ไปถึง 200 ซึ่งระบบจะคำนวณหาค่าตัวแปรให้)

ตารางที่ 3.12 รายการคุณสมบัติของอุปกรณ์ทั่วไป (ต่อ)

คุณสมบัติ	รายละเอียด
DragType=Click	วิธีการเปลี่ยนแปลงค่าตัวแปร (Value) ตัวเลือกเป็น: 1. Click (โดยการคลิก จะได้ค่าเป็น 1 / 0) 2. Horizontal (โดยการ Drag mouse ตามแนวนอนจะได้ค่าเท่ากับ 0... ValueMax) 3. Vertical (โดยการ Drag mouse ตามแนวตั้งจะได้ค่าเท่ากับ 0... ValueMax)
ImageMax=2	จำนวนรูปภาพที่ต้องการแสดง ตัวเลือกเป็น: 0…65,535
IsControllerDevice=True	เป็นอุปกรณ์ที่มีผลต่อการควบคุมหรือไม่ ตัวเลือกเป็น: True / False
Left=0	ตำแหน่งด้านซ้ายของอุปกรณ์
Name=General	ชื่อของอุปกรณ์
Parent=	หมายเลขของอุปกรณ์มีความสัมพันธ์แบบ Part of ด้วย ตัวเลือกเป็น: 0…65,535 ตัวเลือกเป็น: หมายเลขของ Package ที่บรรจุอุปกรณ์ตัวนี้อยู่
PhysicalEffect=False	เป็นอุปกรณ์ที่มีผลต่อการเปลี่ยนแปลงลักษณะทางกายภาพของอุปกรณ์หรือไม่ ตัวเลือกเป็น: True / False
State=0	สถานะของอุปกรณ์ ตัวเลือกเป็น:1 / 0
Top=0	ตำแหน่งด้านบนของอุปกรณ์
Type=General	ประเภทของอุปกรณ์ ตัวเลือกเป็น: General, Package, Sensor, Counter, Timer, และGuideline
Value=0	ค่าตัวแปร โดยจะส่งการแสดงภาพของอุปกรณ์ ตัวเลือกเป็น: 0… ValueMax
ValueLeft=0	ตำแหน่งของการแสดงค่าตัวแปรนับจากด้านซ้ายของอุปกรณ์

ตารางที่ 3.12 รายการคุณสมบัติของอุปกรณ์ทั่วไป (ต่อ)

คุณสมบัติ	รายละเอียด
ValueMax=1	ค่าตัวแปรสูงสุด ตัวเลือกเป็น: 0…65,535
ValueRepeating=False	เมื่อค่าตัวแปรเท่ากับค่าตัวแปรสูงสุดเริ่มต้นใหม่หรือไม่ ตัวเลือกเป็น: True / False
ValueTop=-13	ตำแหน่งของการแสดงค่าตัวแปรนับจากด้านบนของอุปกรณ์
Image0= ImageN=	รูปภาพแต่ละสถานะของอุปกรณ์ โดยที่จำนวนของภาพจะขึ้นอยู่กับค่า ImageMax

2) อุปกรณ์หีบห่อ (Package) ใช้ในการบรรจุอุปกรณ์ชนิดอื่นลงไปได้ โดยที่อุปกรณ์ที่นำมาบรรจุในอุปกรณ์ชนิดหีบห่อนั้นจะมีคุณสมบัติเป็นส่วนหนึ่ง (Part of) ของอุปกรณ์ชนิดหีบห่อทันที เช่น ใช้แทนถังน้ำที่สามารถบรรจุอุปกรณ์วัดระดับน้ำ อุณหภูมิ ความดัน ให้เป็นส่วนหนึ่งของถังน้ำได้ สัญลักษณ์ที่ใช้ในแถบเครื่องมือคือรูปกล่อง ซึ่งหมายถึงการบรรจุ

รายการคุณสมบัติของอุปกรณ์หีบห่อ ดังแสดงในตารางที่ 3.13

ตารางที่ 3.13 รายการคุณสมบัติของอุปกรณ์หีบห่อ

คุณสมบัติ	รายละเอียด
CaptionLeft=0	ตำแหน่งของชื่ออุปกรณ์นับจากด้านซ้ายของอุปกรณ์
CaptionTop=-23	ตำแหน่งของชื่ออุปกรณ์นับจากด้านบนของอุปกรณ์
ImageMax=2	จำนวนรูปภาพที่ต้องการแสดง ตัวเลือกเป็น: 0…65,535
IsControllerDevice=True	เป็นอุปกรณ์ที่มีผลต่อการควบคุมหรือไม่ ตัวเลือกเป็น: True / False
Left=0	ตำแหน่งด้านซ้ายของอุปกรณ์
Name=Package1	ชื่อของอุปกรณ์
Parent=	หมายเลขของอุปกรณ์มีความสัมพันธ์แบบ Part of ด้วย ตัวเลือกเป็น: 0…65,535 ตัวเลือกเป็น: หมายเลขของ Package ที่บรรจุอุปกรณ์ตัวนี้อยู่
Top=0	ตำแหน่งด้านบนของอุปกรณ์

ตารางที่ 3.13 รายการคุณสมบัติของอุปกรณ์หีบห่อ (ต่อ)

คุณสมบัติ	รายละเอียด
Type=Package	ประเภทของอุปกรณ์ ตัวเลือกเป็น: General, Package, Sensor, Counter, Timer, และGuideline
Image0= ImageN=	รูปภาพแต่ละสถานะของอุปกรณ์ โดยที่จำนวนของภาพจะขึ้นอยู่กับค่า ImageMax

3) อุปกรณ์ตรวจจับ (Sensor) ใช้ร่วมกับอุปกรณ์ชนิดทั่วไป โดยมีคุณสมบัติในการตรวจสอบค่าตัวแปรของอุปกรณ์ชนิดทั่วไป เพื่อนำมาเปรียบเทียบกับค่าที่กำหนดไว้ (Setting value) ถ้าผลการเปรียบเทียบเป็นไปตามเงื่อนไขที่กำหนด อุปกรณ์ชนิดตรวจจับจะให้ผลลัพธ์ (State) เป็น "1" ถ้าหากไม่เป็นไปตามเงื่อนไขที่กำหนดอุปกรณ์ชนิดตรวจจับจะให้ผลลัพธ์เป็น "0" สัญลักษณ์ที่ใช้ในแถบเครื่องมือคือรูปตัววัดอุณหภูมิ ซึ่งหมายถึงการวัด

รายการคุณสมบัติของอุปกรณ์ตรวจจับ ดังแสดงในตารางที่ 3.14

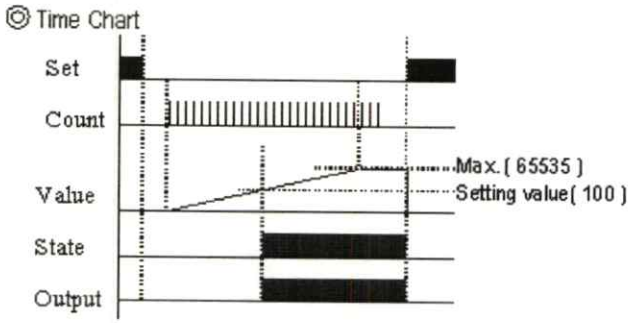
ตารางที่ 3.14 รายการคุณสมบัติของอุปกรณ์ตรวจจับ

คุณสมบัติ	รายละเอียด
CaptionLeft=0	ตำแหน่งของชื่ออุปกรณ์นับจากด้านซ้ายของอุปกรณ์
CaptionTop=-23	ตำแหน่งของชื่ออุปกรณ์นับจากด้านบนของอุปกรณ์
ImageMax=2	จำนวนรูปภาพที่ต้องการแสดง ตัวเลือกเป็น: 0…65,535
Left=0	ตำแหน่งด้านซ้ายของอุปกรณ์
Name= Sensor1	ชื่อของอุปกรณ์
Operator=>=	เครื่องหมายในการเปรียบเทียบค่า ตัวเลือกเป็น: >=, =, <= และ <>
Parent=	หมายเลขของอุปกรณ์มีความสัมพันธ์แบบ Part of ด้วย ตัวเลือกเป็น: 0…65,535 ตัวเลือกเป็น: หมายเลขของ Package ที่บรรจุอุปกรณ์ตัวนี้อยู่
SettingValue=0	เป็นค่าที่ได้กำหนดไว้เพื่อทำการเปรียบเทียบค่าตัวแปร (Value) ตัวเลือกเป็น: 0…65,535

ตารางที่ 3.14 รายการคุณสมบัติของอุปกรณ์ตรวจจับ (ต่อ)

คุณสมบัติ	รายละเอียด
State=0	สถานะของอุปกรณ์ ตัวเลือกเป็น: 0…65,535
Top=0	ตำแหน่งด้านบนของอุปกรณ์
Type=Sensor	ประเภทของอุปกรณ์ ตัวเลือกเป็น: General, Package, Sensor, Counter, Timer, และGuideline
Value=0	ค่าตัวแปร โดยจะส่งการแสดงภาพของอุปกรณ์ ตัวเลือกเป็น: 0… ValueMax
ValueLeft=0	ตำแหน่งของการแสดงค่าตัวแปรนับจากด้านซ้ายของอุปกรณ์
ValueTop=-13	ตำแหน่งของการแสดงค่าตัวแปรนับจากด้านบนของอุปกรณ์
Image0= ImageN=	รูปภาพแต่ละสถานะของอุปกรณ์ โดยที่จำนวนของภาพจะขึ้นอยู่กับค่า ImageMax

4) อุปกรณ์นับจำนวน (Counter) ใช้ในการนับจำนวนโดยใช้ร่วมกับอุปกรณ์ชนิดทั่วไปหรืออุปกรณ์ชนิดตรวจจับ ซึ่งอุปกรณ์ชนิดนับจำนวนจะประกอบไปด้วยค่าต่างๆ ได้แก่ ค่าสถานะ (State) สถานะของการนับจำนวน (Count) ค่าจำนวนนับ (Value) ค่าจำนวนนับที่กำหนด (Setting value) และสภาวะพร้อมที่จะทำงาน (Set) โดยหากสภาวะพร้อมที่จะทำงานเป็น “1” จะทำให้จำนวนนับและค่าสถานะเป็น “0” อุปกรณ์ชนิดนับจำนวนจะเริ่มนับจำนวนต่อเมื่อค่าสภาวะพร้อมที่จะทำงานเป็น “0” จะนับเพิ่มทุกครั้งที่สถานะของการนับจำนวนเป็น “1” และจะนับเพิ่มขึ้นเรื่อยๆ เมื่อถึงค่าที่กำหนดอุปกรณ์ชนิดนับจำนวนจะให้ผลลัพธ์เป็น “1” ดังแสดงในรูปที่ 3.49 สัญลักษณ์ที่ใช้ในแถบเครื่องมือคือรูปตัวเลข ซึ่งหมายถึงการนับ



รูปที่ 3.50 แสดงสถานะการทำงานของอุปกรณ์นับจำนวน

รายการคุณสมบัติของอุปกรณ์นับจำนวน ดังแสดงในตารางที่ 3.15

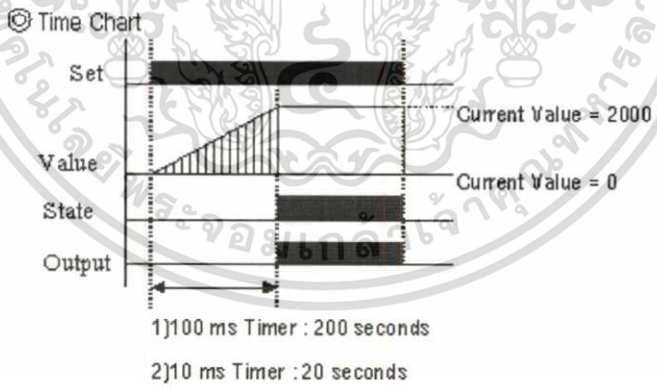
ตารางที่ 3.15 รายการคุณสมบัติของอุปกรณ์นับจำนวน

คุณสมบัติ	รายละเอียด
CaptionLeft=0	ตำแหน่งของชื่ออุปกรณ์นับจากด้านซ้ายของอุปกรณ์
CaptionTop=-23	ตำแหน่งของชื่ออุปกรณ์นับจากด้านบนของอุปกรณ์
ImageMax=2	จำนวนรูปภาพที่ต้องการแสดง ตัวเลือกเป็น: 0…65,535
IsControllerDevice=True	เป็นอุปกรณ์ที่มีผลต่อการควบคุมหรือไม่ ตัวเลือกเป็น: True / False
Left=0	ตำแหน่งด้านซ้ายของอุปกรณ์
Name= Counter1	ชื่อของอุปกรณ์
Parent=	หมายเลขของอุปกรณ์มีความสัมพันธ์แบบ Part of ด้วย ตัวเลือกเป็น: 0…65,535 ตัวเลือกเป็น: หมายเลขของ Package ที่บรรจุอุปกรณ์ตัวนี้อยู่
Set=0	เป็นค่าแสดงสถานะของอุปกรณ์ว่ากำลังทำงานอยู่หรือไม่ ตัวเลือกเป็น:1 / 0
SettingValue=0	เป็นค่าที่ได้กำหนดไว้เพื่อทำการเปรียบเทียบค่าตัวแปร (Value) ตัวเลือกเป็น: 0…65,535
State=0	สถานะของอุปกรณ์ ตัวเลือกเป็น:1 / 0
Top=0	ตำแหน่งด้านบนของอุปกรณ์
Type= Counter	ประเภทของอุปกรณ์ ตัวเลือกเป็น: General, Package, Sensor, Counter, Timer, และGuideline
Value=0	ค่าตัวแปร โดยจะส่งการแสดงภาพของอุปกรณ์ ตัวเลือกเป็น: 0… 65,535
ValueLeft=0	ตำแหน่งของการแสดงค่าตัวแปรนับจากด้านซ้ายของอุปกรณ์
ValueMax=1	ค่าตัวแปรสูงสุด ตัวเลือกเป็น: 0…65,535

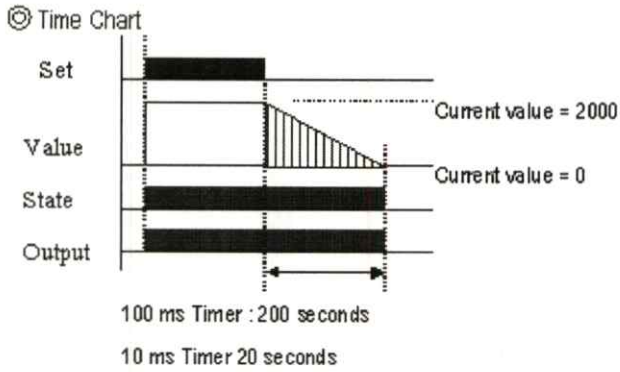
ตารางที่ 3.15 รายการคุณสมบัติของอุปกรณ์นับจำนวน (ต่อ)

คุณสมบัติ	รายละเอียด
ValueTop=-13	ตำแหน่งของการแสดงค่าตัวแปรนับจากด้านบนของอุปกรณ์
Image0= ImageN=	รูปภาพแต่ละสถานะของอุปกรณ์ โดยที่จำนวนของภาพจะขึ้นอยู่กับค่า ImageMax

5) อุปกรณ์จับเวลา (Timer) ใช้ในการจับเวลาโดยใช้ร่วมกับอุปกรณ์ชนิดทั่วไปหรืออุปกรณ์ชนิดตรวจจับ ซึ่งอุปกรณ์ชนิดจับเวลาจะประกอบไปด้วยค่าต่างๆ ได้แก่ ค่าสถานะ (State) ค่าจำนวนเวลา (Value) ค่าจำนวนเวลาที่กำหนด (Setting value) และสภาวะพร้อมที่จะทำงาน (Set) อุปกรณ์ชนิดจับเวลาจะแบ่งออกเป็นสองชนิดคือ ตั้งเวลาเปิดและตั้งเวลาปิด โดยที่ชนิดตั้งเวลาเปิด (Time On) เมื่อสภาวะพร้อมที่จะทำงานเป็น “0” จะทำให้จำนวนเวลาและค่าสถานะเป็น “0” อุปกรณ์ชนิดจับเวลาจะเริ่มเพิ่มเวลาต่อเมื่อค่าสภาวะพร้อมที่จะทำงานเป็น “1” และจะเพิ่มขึ้นเรื่อยๆ เมื่อถึงค่าที่กำหนดอุปกรณ์ชนิดจับเวลาจะให้ผลลัพธ์เป็น “1” ส่วนชนิดตั้งเวลาปิด (Time Off) หากสภาวะพร้อมที่จะทำงานเป็น “1” จะทำให้จำนวนเวลาเท่ากับค่าเวลาที่กำหนดและค่าสถานะเป็น “0” อุปกรณ์ชนิดจับเวลาจะเริ่มลดจำนวนต่อเมื่อค่าสภาวะพร้อมที่จะทำงานเป็น “0” และจะลดลงเรื่อยๆ จนเท่ากับ “0” อุปกรณ์ชนิดจับเวลาจะให้ผลลัพธ์เป็น “1” ดังแสดงในรูปที่ 3.51 และรูปที่ 3.52 ตามลำดับ สัญลักษณ์ที่ใช้ในแถบเครื่องมือคือรูปนาฬิกา ซึ่งหมายถึงการจับเวลา



รูปที่ 3.51 แสดงสถานะการทำงานของอุปกรณ์จับเวลาชนิดตั้งเวลาเปิด (Time On)



รูปที่ 3.52 แสดงสถานะการทำงานของอุปกรณ์จับเวลาชนิดตั้งเวลาปิด (Time Off)

รายการคุณสมบัติของอุปกรณ์จับเวลา ดังแสดงในตารางที่ 3.16

ตารางที่ 3.16 รายการคุณสมบัติของอุปกรณ์จับเวลา

คุณสมบัติ	รายละเอียด
CaptionLeft=0	ตำแหน่งของชื่ออุปกรณ์นับจากด้านซ้ายของอุปกรณ์
CaptionTop=-23	ตำแหน่งของชื่ออุปกรณ์นับจากด้านบนของอุปกรณ์
ImageMax=2	จำนวนรูปภาพที่ต้องการแสดง ตัวเลือกเป็น: 0…65,535
IsControllerDevice=True	เป็นอุปกรณ์ที่มีผลต่อการควบคุมหรือไม่ ตัวเลือกเป็น: True / False
Left=0	ตำแหน่งด้านซ้ายของอุปกรณ์
Name= Timer 1	ชื่อของอุปกรณ์
Parent=	หมายเลขของอุปกรณ์มีความสัมพันธ์แบบ Part of ด้วย ตัวเลือกเป็น: 0…65,535 ตัวเลือกเป็น: หมายเลขของ Package ที่บรรจุอุปกรณ์ตัวนี้อยู่
Set=0	เป็นค่าแสดงสถานะของอุปกรณ์ว่ากำลังทำงานอยู่หรือไม่ ตัวเลือกเป็น:1 / 0
SettingValue=0	เป็นค่าที่ได้กำหนดไว้เพื่อทำการเปรียบเทียบค่าตัวแปร (Value) ตัวเลือกเป็น: 0…65,535
State=0	สถานะของอุปกรณ์ ตัวเลือกเป็น:1 / 0

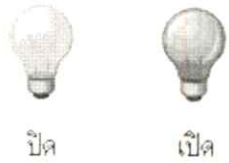
ตารางที่ 3.16 รายการคุณสมบัติของอุปกรณ์จับเวลา (ต่อ)

คุณสมบัติ	รายละเอียด
TimerType=TOFF	ประเภทของอุปกรณ์จับเวลา ตัวเลือกเป็น: TON และ TOFF
Top=0	ตำแหน่งด้านบนของอุปกรณ์
Type= Timer	ประเภทของอุปกรณ์ ตัวเลือกเป็น: General, Package, Sensor, Counter, Timer, และGuideline
Value=0	ค่าตัวแปร โดยจะส่งการแสดงภาพของอุปกรณ์ ตัวเลือกเป็น: 0… 65535
ValueLeft=0	ตำแหน่งของการแสดงค่าตัวแปรนับจากด้านซ้ายของอุปกรณ์
ValueMax=1	ค่าตัวแปรสูงสุด ตัวเลือกเป็น: 0…65,535
ValueTop=-13	ตำแหน่งของการแสดงค่าตัวแปรนับจากด้านบนของอุปกรณ์
Image0= ImageN=	รูปภาพแต่ละสถานะของอุปกรณ์ โดยที่จำนวนของภาพจะขึ้นอยู่กับค่า ImageMax

3.8.4 การออกแบบการกำหนดคุณสมบัติของอุปกรณ์ (Device Properties)

การสร้างแบบจำลองนั้นจะต้องอาศัยการแสดงภาพของเครื่องจักรซึ่งมีองค์ประกอบของชิ้นส่วนอุปกรณ์ต่างๆ โดยแต่ละชิ้นส่วนนั้นหมายถึงอุปกรณ์หนึ่งอุปกรณ์นั่นเอง โดยที่แต่ละอุปกรณ์จะมีการแสดงออกถึงรูปร่างลักษณะที่บ่งบอกถึงสถานะของอุปกรณ์ (Device Image) อีกส่วนหนึ่งคือคุณสมบัติของอุปกรณ์ (Device Properties) ที่ตอบสนองกับการกระตุ้นเงื่อนไขการทำงานของอุปกรณ์ค่าตัวแปรที่บ่งบอกคุณสมบัติแตกต่างกัน ฉะนั้นการสร้างอุปกรณ์ในวิธีการนี้เราจะคำนึงถึงสองส่วนคือ ส่วนแรกการใช้รูปภาพแสดงสถานะต่างๆ ของอุปกรณ์ ส่วนที่สองคือการกำหนดคุณสมบัติให้กับอุปกรณ์

สิ่งแรกที่เราจะต้องจัดเตรียมคือรูปภาพของชิ้นส่วนของอุปกรณ์นั้นๆ โดยรูปภาพที่จะนำมาใช้แทนอุปกรณ์ต้องสามารถแสดงถึงรูปร่างหรือสถานะที่จะเกิดขึ้นได้ของอุปกรณ์ชิ้นนั้นๆ ให้ครบถ้วนด้วย เช่น หลอดไฟฟ้า จะมีสถานะของตัวเองอยู่สองสถานะคือสถานะปิดและสถานะเปิด ดังนั้นเราจะมีภาพที่จะแทนหลอดไฟฟ้า 2 ภาพเช่นกันดังรูปที่ 3.53

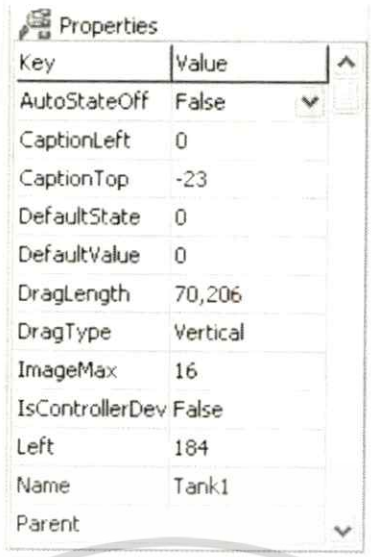


รูปที่ 3.53 แสดงภาพของหลอดไฟฟ้าทั้งสองสถานะ

นอกเหนือจากอุปกรณ์ที่มีสถานะเพียงสองสถานะแล้วยังมีอุปกรณ์อื่น ที่มีสถานะมากกว่าสองสถานะ ตัวอย่างเช่น ระดับน้ำในถัง การเคลื่อนที่ของสายพานลำเลียง เราจะต้องจัดเตรียมรูปภาพตามจำนวนที่เหมาะสม ซึ่งอาจไม่ได้เป็นไปตามเหตุการณ์จริงที่เกิดขึ้นได้ก็ตาม จากตัวอย่างของระดับน้ำในถังเราอาจจะสร้างภาพในการแสดงผลเพียง 10 ภาพ โดยที่ค่าตัวแปร (Value) ที่จะเป็นได้อาจมีถึง 100 ค่า โดยที่เราจะนำภาพมาเฉลี่ยให้กับค่าตัวแปรให้โดยอัตโนมัติ ดังแสดงการเตรียมภาพของระดับน้ำในถังในรูปที่ 3.54

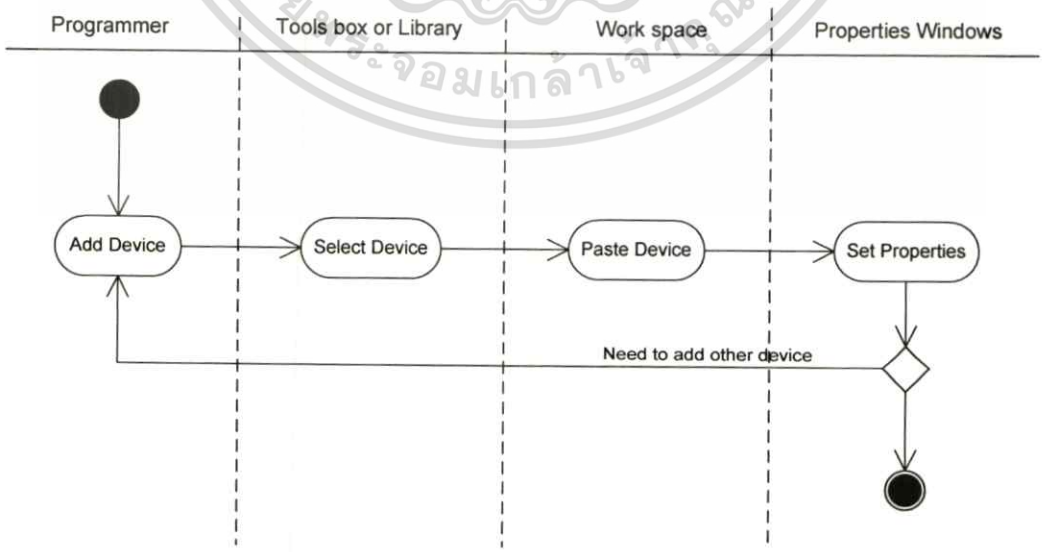
รูปที่ 3.54 แสดงการเตรียมภาพของระดับน้ำในถัง

เมื่อเราได้ทำการเตรียมภาพที่ใช้แสดงแทนสถานะต่างๆ ให้กับอุปกรณ์ครบทุกสถานะ ขั้นตอนต่อไปคือการนำภาพเหล่านั้นมากำหนดให้อุปกรณ์หนึ่งไปเป็นคุณสมบัติของอีกอุปกรณ์หนึ่ง คุณสมบัติของอุปกรณ์ต่างๆ ที่เราสามารถกำหนดได้ เช่น รูปภาพที่แสดงแทนสถานะต่างๆ ของอุปกรณ์ ชื่อของตัวอุปกรณ์ ค่าตัวแปร (Value) สถานะ (State) เป็นต้น โดยที่จะแสดงผ่านทางหน้าต่างคุณลักษณะ (Properties windows) สำหรับรายละเอียดของคุณสมบัติต่างๆ ของอุปกรณ์นั้นจะขึ้นอยู่กับชนิดของอุปกรณ์ ซึ่งได้กล่าวไว้แล้วในหัวข้ออุปกรณ์พื้นฐาน (Standard Devices)



รูปที่ 3.55 แสดงหน้าต่างคุณลักษณะ (Properties windows)

ในการสร้างแบบจำลองนั้น เราสามารถเลือกใช้อุปกรณ์ได้ทั้งสองแบบ แบบแรกคืออุปกรณ์พื้นฐาน (Standard Devices) และแบบที่สองคืออุปกรณ์สำเร็จรูปจากหน้าต่างห้องสมุดอุปกรณ์ แล้วมาวางลงบนพื้นที่ทำงาน (Workspace) โดยที่เมื่อวางแล้วจะตั้งชื่อให้กับอุปกรณ์ ซึ่งจะกำหนดชื่อให้เบื้องต้นก่อน (Default Name) โดยใช้กฎในการตั้งชื่อของอุปกรณ์คือใช้ชื่อของชนิดอุปกรณ์ตามด้วยหมายเลขลำดับของชนิดอุปกรณ์นั้น เช่น อุปกรณ์ชนิดทั่วไปตัวแรก จะมีชื่อว่า General1 ตัวต่อไปจะมีชื่อว่า General2 ฯลฯ ซึ่งสามารถแสดงแผนภาพขั้นตอนการสร้างแบบจำลองได้ดังนี้



รูปที่ 3.56 แสดงแผนภาพขั้นตอนการสร้างแบบจำลอง

บทที่ 4

ผลการทดลอง

ในบทนี้จะกล่าวถึงผลการทดลองเพื่อวัดประสิทธิภาพของวิธีการโปรแกรมโดยใช้การสาธิตตัวควบคุมและแบบจำลองร่วมกันสำหรับอุปกรณ์พีแอลซีโดยเปรียบเทียบกับวิธีการเขียนโปรแกรมด้วยภาษาแลดเดอร์ ซึ่งเป็นภาษาที่เขียนอยู่ในรูปของกราฟิก โดยมีพื้นฐานมาจากวงจรควบคุมแบบรีเลย์ และวงจรไฟฟ้า ซึ่งภาษาแลดเดอร์จะประกอบด้วยราง (Rail) ทั้งซ้ายและขวาของไดอะแกรม เพื่อใช้สำหรับเชื่อมต่ออุปกรณ์ที่เป็นสวิตช์หน้าสัมผัส เพื่อเป็นทางผ่านของกระแสและมีขดลวดหรือคอยล์เป็นเอาท์พุท ภาษาแลดเดอร์นี้เป็นหนึ่งในภาษาที่ใช้ในการเขียนโปรแกรมตามมาตรฐาน IEC 1131-3 การประเมินส่วนเชื่อมต่อผู้ใช้จะเป็นการประเมินโดยใช้วงจรพฤติกรรมมนุษย์ (Human action cycle) ซึ่งเป็นโมเดลทางกายภาพที่สามารถอธิบายขั้นตอนที่มนุษย์เลือกเมื่อมีปฏิสัมพันธ์กับระบบคอมพิวเตอร์ โมเดลได้ถูกนำเสนอโดย Donald A. Norman [17] ผู้เชี่ยวชาญทางด้านพฤติกรรมของมนุษย์-คอมพิวเตอร์เมื่อปฏิสัมพันธ์

4.1 รายละเอียดการทดลองกับกลุ่มตัวอย่าง

4.1.1 กลุ่มตัวอย่าง

กลุ่มตัวอย่างแบ่งเป็นสองกลุ่มตามคุณสมบัติของผู้เข้าร่วมทดสอบ กลุ่มแรกเป็นผู้ที่มีประสบการณ์การทำงานเกี่ยวกับการควบคุมเครื่องจักรโดยใช้อุปกรณ์พีแอลซี สามารถเขียนโปรแกรมภาษาคอมพิวเตอร์ได้ สามารถบำรุงรักษา แก้ไข ออกแบบวงจรไฟฟ้าเล็กทรอนิกส์ได้ สามารถออกแบบระบบควบคุมเครื่องจักรอัตโนมัติได้ และเขียนโปรแกรมภาษาที่ใช้กับอุปกรณ์พีแอลซีได้ดี กลุ่มที่สองเป็นผู้ที่ไม่มีประสบการณ์การทำงานเกี่ยวกับการควบคุมเครื่องจักรไม่มีความสามารถด้านเขียนโปรแกรมภาษาคอมพิวเตอร์ ไม่มีความรู้ด้านไฟฟ้าเล็กทรอนิกส์ และระบบควบคุมเครื่องจักรอัตโนมัติ

ตารางที่ 4.1 แสดงคุณสมบัติของกลุ่มตัวอย่างที่ 1

คุณสมบัติ	ผู้เข้าร่วมทดสอบคนที่				
	1	2	3	4	5
1. ระดับการศึกษา	ปริญญาตรี	ปริญญาตรี	ปริญญาตรี	ปริญญาตรี	ปริญญาตรี
2. จบสาขาที่เกี่ยวข้อง	ไม่ตรงสาขา	ตรงสาขา	ตรงสาขา	ตรงสาขา	ตรงสาขา
3. ใช้พีแอลซียี่ห้อ	Siemen, Kience, Koyo, Misubishi	Omron	Simulation PLC	Misubishi, Omron	Misubishi
4. ประสบการณ์การทำงานอุปกรณ์พีแอลซี	7 ปี	3 ปี	ฝึกงาน	ฝึกงาน	3 ปี
5. เขียนโปรแกรมภาษาคอมพิวเตอร์	สามารถเขียนได้	สามารถเขียนได้	สามารถเขียนได้	สามารถเขียนได้	สามารถเขียนได้
6. ความรู้ด้านวงจรไฟฟ้าอิเล็กทรอนิกส์	ออกแบบแก้ไขได้	ออกแบบแก้ไขได้	ออกแบบแก้ไขได้	ออกแบบแก้ไขได้	ออกแบบแก้ไขได้
7. ความรู้ด้านระบบควบคุมเครื่องจักรอัตโนมัติ	ออกแบบระบบได้	ออกแบบระบบได้	ออกแบบระบบได้	ออกแบบระบบได้	ออกแบบระบบได้
8. เขียนโปรแกรมภาษาที่ใช้กับอุปกรณ์พีแอลซี	Mnemonic, Ladder, Structure text, Function block	Mnemonic, Ladder, Function block	Ladder	Mnemonic, Ladder	Mnemonic, Ladder

ตารางที่ 4.2 แสดงคุณสมบัติของกลุ่มตัวอย่างที่ 2

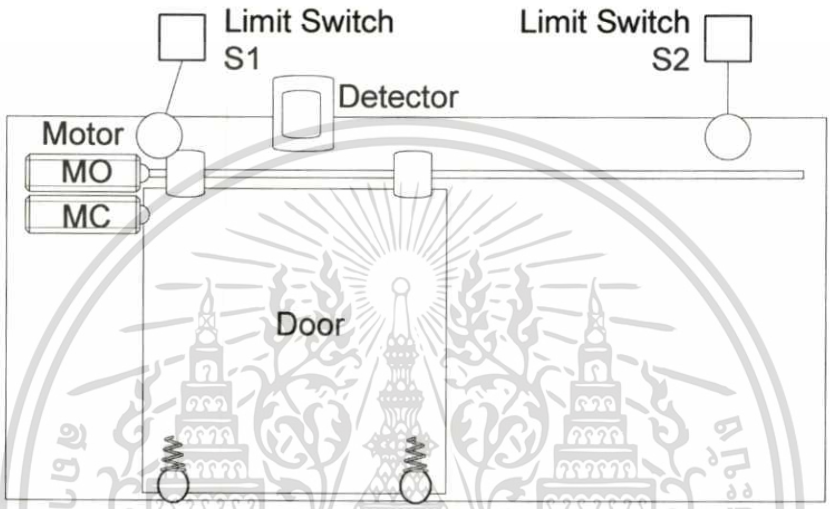
คุณสมบัติ	ผู้เข้าร่วมทดสอบคนที่				
	1	2	3	4	5
1. ระดับการศึกษา	ปริญญาตรี	มัธยมศึกษา	ปริญญาตรี	ปริญญาตรี	ปริญญาตรี
2. จบสาขา	การตลาด	-	สถิติ	สถิติ	สถิติ
3. ความรู้ด้านอุปกรณ์พีแอลซี ด้านการเขียนโปรแกรม ด้านไฟฟ้าอิเล็กทรอนิกส์ และด้านระบบควบคุมเครื่องจักรอัตโนมัติ	ไม่มี	ไม่มี	ไม่มี	ไม่มี	ไม่มี

4.1.2 วิธีการทดสอบ

เนื่องจากกลุ่มตัวอย่างทั้งหมดมีประสบการณ์การทำงานเกี่ยวกับการควบคุมเครื่องจักรโดยใช้อุปกรณ์พีแอลซี สามารถเขียนโปรแกรมภาษาคอมพิวเตอร์ได้ สามารถบำรุงรักษา แก้ไขออกแบบวงจรไฟฟ้าเล็กทรอนิกส์ได้ สามารถออกแบบระบบควบคุมเครื่องจักรอัตโนมัติได้ และเขียนโปรแกรมภาษาที่ใช้กับอุปกรณ์พีแอลซีได้ดี อีกทั้งได้มีการอธิบายและสร้างความเข้าใจในแบบทดสอบก่อนลงมือปฏิบัติ ดังนั้นจึงถือได้ว่ากลุ่มตัวอย่างมีความเข้าใจและความรู้ในงาน พร้อมทั้งมีความเข้าใจงานเพื่อที่จะกำหนดเป็นชุดคำสั่งเพียงพอ โดยปกติการเขียนโปรแกรมด้วยภาษาแลดเดอร์นั้นจะต้องใช้ระยะเวลาในการเรียนรู้มาก เริ่มตั้งแต่การเรียนรู้วิชาตรรกวิทยา (Logical) และวิชาที่ว่าด้วยรูปแบบและวิธีการโปรแกรมด้วยภาษาแลดเดอร์ ซึ่งในสถานศึกษาจะใช้เวลา 1 ถึง 2 ภาคเรียน การทดสอบครั้งนี้จำเป็นต้องใช้ผู้ที่สามารถเขียนโปรแกรมภาษาแลดเดอร์ได้ดี เพื่อลดระยะเวลาในการเก็บข้อมูลลง แต่กลุ่มตัวอย่างทั้งหมดยังไม่เคยใช้เครื่องมือการเขียนโปรแกรมของเคจีแอลซึ่งเป็นเครื่องมือในการเขียนโปรแกรมภาษาแลดเดอร์ของบริษัทแอลจีมาก่อน จึงต้องมีการอบรมการใช้เครื่องมือการเขียนโปรแกรมของเคจีแอลก่อนทดสอบเป็นระยะเวลา 1 ชั่วโมง ส่วนการอบรมการใช้เครื่องมือวิธีการโปรแกรมโดยใช้การสาธิตที่ได้นำเสนอนั้นได้มีการอบรมการใช้งานเครื่องมือวิธีการโปรแกรมโดยใช้การสาธิตเป็นระยะเวลา 2 ชั่วโมงก่อนทดสอบ สาเหตุที่ใช้เวลามากกว่านั้นเพราะกลุ่มตัวอย่างทั้งหมดยังไม่เคยเรียนรู้ในวิธีการนี้มาก่อนเลย จึงต้องเพิ่มเติมในส่วนของรูปแบบและวิธีการโปรแกรมด้วย ซึ่งต่างจากการเขียนโปรแกรมภาษาแลดเดอร์ที่กลุ่มตัวอย่างทั้งหมดได้ผ่านการเรียนและใช้งานมาเป็นระยะเวลาอย่างน้อย 1 ปีแล้ว

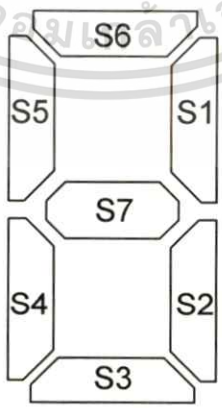
การทดสอบในครั้งนี้ได้อธิบายและสร้างความเข้าใจในแบบทดสอบก่อนลงมือปฏิบัติ ดังนั้นจึงถือได้ว่ากลุ่มตัวอย่างมีความเข้าใจและความรู้ในงานเพียงพอ พร้อมทั้งมีความเข้าใจงานเพื่อที่จะกำหนดเป็นชุดคำสั่งเพียงพอ จากนั้นจะเริ่มทำการทดสอบโดยใช้แบบทดสอบที่แตกต่างกันสองแบบทดสอบ โดยมีระยะเวลาในการทำแบบทดสอบ 45 นาทีต่อ 1 แบบทดสอบ วิธีแรกการเขียนโปรแกรมภาษาแลดเดอร์และวิธีที่สองคือวิธีการโปรแกรมโดยใช้การสาธิตตามลำดับ

แบบทดสอบที่ 1 ควบคุมการทำงานของประตูอัตโนมัติ



รูปที่ 4.1 แสดงองค์ประกอบของประตูอัตโนมัติ

แบบทดสอบที่ 2 ควบคุมการเปิดปิดหลอดไฟฟ้าจำนวน 7 หลอด เพื่อแสดงเป็นเลข 1 ถึง เลข 5 โดยแต่ละเลขมีระยะเวลาห่างกัน 5 นาที



รูปที่ 4.2 แสดงตำแหน่งหลอดไฟฟ้าจำนวน 7 หลอด

4.2 สรุปวิเคราะห์ผลการสังเกตการทดลองกับกลุ่มตัวอย่าง

จากผลการสังเกตการทดลองกับกลุ่มตัวอย่างสามารถวิเคราะห์ผลการสังเกตได้ว่า วิธีการที่ได้นำเสนอนี้สามารถแก้ปัญหาต่างๆ ดังนี้

4.2.1 การจำลองจะลดความผิดพลาดในการแปลงโจทย์มาเป็นโปรแกรม การใช้แบบจำลองของเครื่องจักรที่สามารถแสดงผลการทำงานได้ใกล้เคียงกับเครื่องจักรจริงนั้น ทำให้ผู้ใช้สามารถประเมินและตีความหมายในขั้นที่ 7 ได้โดยตรง อีกทั้งการจำลองแบบยังส่งผลให้ในขั้นที่ 3 การแปลงงานสู่ลำดับของการปฏิบัตินั้นคล้ายกับการทำงานแบบ Manual จึงสามารลดความผิดพลาดลงได้ ส่วนวิธีการเขียนโปรแกรมภาษาแลดเดอร์นั้นมีหลักการแปลงเป้าหมายไปสู่งานโดยการสร้างตารางความจริงของตรรกะเสียก่อน จากนั้นจึงนำตรรกะที่ได้มาเขียนโปรแกรมภาษาแลดเดอร์ทำให้เกิดข้อผิดพลาดได้ง่าย รวมถึงขั้นที่ 7 ที่จะต้องตีความหมายจากสถานะของอุปกรณ์ทั้งหมดเปรียบเทียบและทบทวนกับ Logic Table เพื่อที่จะประเมินเทียบกับงานนั้นก็ส่งผลต่อความผิดพลาดเช่นกัน

4.2.2 การจำลองจะลดการทำงานที่ไม่เป็นไปตามจุดประสงค์ของผู้โปรแกรม (Bug) อันเนื่องจากการเขียนตรรกะที่ผิดพลาด (Logic error) ซึ่งการโปรแกรมการควบคุมในวิธีการโปรแกรมโดยใช้การสาธิตนั้น ในขั้นที่ 2 หลักการแปลงเป้าหมายไปสู่งาน, ขั้นที่ 3 การแปลงงานสู่ลำดับของการปฏิบัติ และขั้นที่ 4 ลงมือโปรแกรมนั้น จะเป็นไปตามเหตุการณ์ที่เกิดขึ้นคล้ายกับการทำงานแบบ Manual ส่วนวิธีการเขียนโปรแกรมภาษาแลดเดอร์ ทั้งสามขั้นตอนนั้นจะเริ่มจากการสร้างตารางความจริงของตรรกะเสียก่อน จากนั้นจึงนำตรรกะที่ได้มาโปรแกรม โดยในการทดลองพบว่าในแบบทดสอบที่ 1 พบการทำงานที่ไม่เป็นไปตามจุดประสงค์ของผู้โปรแกรมเฉลี่ยอยู่ที่ 4.2 ครั้ง แก้ไขชุดคำสั่ง 8.6 ครั้ง แบบทดสอบที่ 2 พบการทำงานที่ไม่เป็นไปตามจุดประสงค์ของผู้โปรแกรมเฉลี่ยอยู่ที่ 7.2 ครั้ง แก้ไขชุดคำสั่ง 12 ครั้ง

4.2.3 วิธีการโปรแกรมโดยใช้การสาธิตช่วยลดการผิดพลาดตามกฎไวยากรณ์ (Syntax error) เนื่องจากวิธีการนี้จะไม่มีการใช้ชุดคำสั่ง จึงไม่พบการผิดพลาดตามกฎไวยากรณ์ ส่วนวิธีการเขียนโปรแกรมภาษาแลดเดอร์ชุดคำสั่งจะเป็นลักษณะ Command Base จากการสังเกตกลุ่มตัวอย่างในขั้นที่ 4 เมื่อลงมือโปรแกรมพบว่ามีการเขียนคำสั่งที่ไม่เป็นไปตามหลักไวยากรณ์ของภาษาแลดเดอร์ โดยในการทดลองพบว่าในแบบทดสอบที่ 1 พบการผิดพลาดตามกฎไวยากรณ์เฉลี่ยอยู่ที่ 1.8 ครั้ง แบบทดสอบที่ 2 พบการผิดพลาดตามกฎไวยากรณ์เฉลี่ยอยู่ที่ 5.8 ครั้ง

4.2.4 วิธีการโปรแกรมโดยใช้การสาธิตการควบคุมและการจำลองร่วมกันจะลดเวลาในการโปรแกรม เนื่องจากขั้นที่ 2 หลักการแปลงเป้าหมายไปสู่งาน, ขั้นที่ 3 การแปลงงานสู่ลำดับของการปฏิบัติ, ขั้นที่ 4 ลงมือโปรแกรมซึ่งคล้ายกับการทำงานแบบ Manual และขั้นที่ 7 การตีความหมายจากผลลัพธ์ของการโปรแกรมได้โดยตรงนั้น ทำให้วิธีการโปรแกรมโดยใช้การสาธิต

ช่วยลดระยะเวลาการโปรแกรมลงได้ ส่วนวิธีการเขียนโปรแกรมภาษาแลดเดอร์นั้น การโปรแกรมจะเป็นลักษณะตรรกะรวมถึงขั้นที่ 7 ซึ่งจะต้องตีความหมายจากสถานะของอุปกรณ์ทั้งหมดเปรียบเทียบและทบทวนกับ Logic Table ทำให้วิธีการนี้ใช้ระยะเวลาในการโปรแกรมมากกว่า จากการทดสอบกับกลุ่มตัวอย่างได้ผลดังนี้

1) แบบทดสอบที่ 1 การโปรแกรมภาษาแลดเดอร์ใช้เวลาไป 19.47 การโปรแกรมโดยการสาธิตใช้เวลาไป 2.404

2) แบบทดสอบที่ 2 การโปรแกรมภาษาแลดเดอร์ใช้เวลาไป 37.878 การโปรแกรมโดยการสาธิตใช้เวลาไป 3.072

4.2.5 วิธีการโปรแกรมโดยใช้การสาธิตตัวควบคุมและแบบจำลองร่วมกัน ทำให้สามารถโปรแกรมครอบคลุมเงื่อนไขการทำงานได้ครบถ้วน เนื่องจากตัวเครื่องมือการโปรแกรมนั้นสามารถตรวจสอบความครบถ้วนของการโปรแกรม ซึ่งอยู่ในส่วนของการจัดการเหตุการณ์ที่ไม่ได้ถูกโปรแกรม (Missing case) ทำให้ลดข้อผิดพลาดในการโปรแกรมที่ไม่ครบถ้วนลงได้

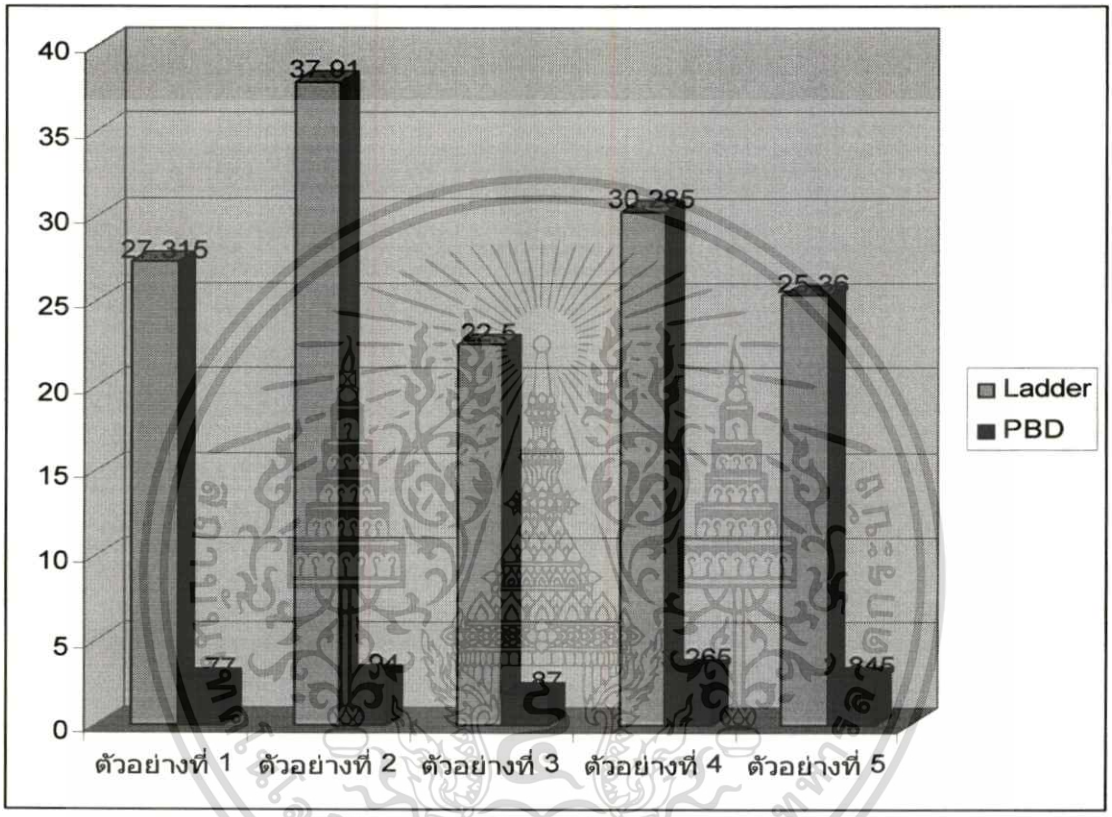
4.2.6 วิธีการโปรแกรมโดยใช้การสาธิตตัวควบคุมและแบบจำลองร่วมกันจะลดความผิดพลาดในการตีความหมายจากผลลัพธ์ของการโปรแกรมได้ เนื่องจากการใช้แบบจำลองเครื่องจักรที่เหมือนจริงและการแสดงผลลัพธ์ด้วยภาพเคลื่อนไหว ในวิธีการนี้ทำให้ผู้ใช้สามารถประเมินและตีความหมายในขั้นที่ 7 ได้โดยตรง ส่วนภาษาแลดเดอร์ซึ่งเป็นการเขียนโปรแกรมเชิงนามธรรม (Abstract) นั้น ผู้ใช้มีปัญหามากในการประเมิน เนื่องจากจะต้องตีความหมายจากสถานะของอุปกรณ์ทั้งหมดเปรียบเทียบและทบทวนกับ Logic Table เพื่อที่จะประเมินเทียบกับงานว่าเข้าใกล้เป้าหมายหรือยัง จากการสังเกตการทดลองกับกลุ่มตัวอย่างพบว่าใช้เวลานานมากในการวิเคราะห์ว่างานทั้งหมดเข้าใกล้เป้าหมายหรือยัง และบางกลุ่มตัวอย่างกลับไปแก้ไขโปรแกรมทั้งๆ ที่ผลลัพธ์ถูกต้องแล้ว

4.3 สรุประยะเวลาและจำนวนเหตุการณ์ในระหว่างการโปรแกรม

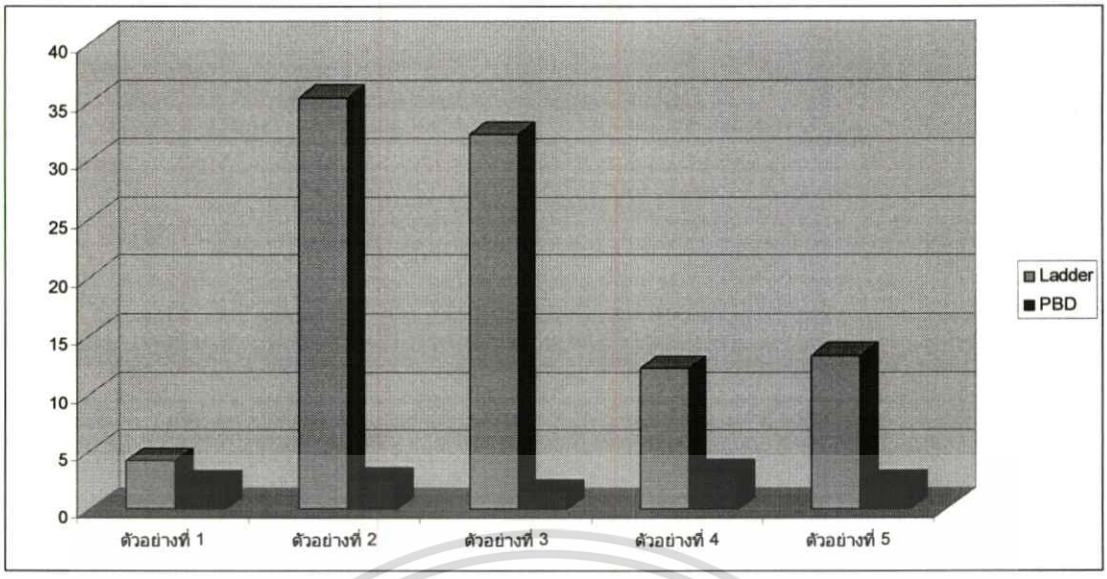
จากการสังเกตการทดลองกับกลุ่มตัวอย่างในครั้งนี้ได้บันทึกระยะเวลา และจำนวนเหตุการณ์ต่างๆ ที่เกิดขึ้นในระหว่างทดสอบการโปรแกรมเทียบกันทั้งสองวิธี ผลลัพธ์ที่ได้จากการทดสอบครั้งนี้จะเห็นได้ชัดเจนว่า วิธีการโปรแกรมโดยใช้การสาธิตตัวควบคุมและแบบจำลองร่วมกันนั้น ระยะเวลาของการโปรแกรมและข้อผิดพลาดลดลง เมื่อเทียบกับการโปรแกรมด้วยวิธีการเดิมที่เขียนด้วยภาษาแลดเดอร์ตามมาตรฐาน IEC 1131-3 ซึ่งเป็นภาษาที่นิยมใช้ในปัจจุบันอย่างมาก

ผลการบันทึกระยะเวลาโดยรวมของการโปรแกรมโดยใช้แบบทดสอบที่ 1 และแบบทดสอบที่ 2 นั้นจะพบว่าในวิธีการเขียนโปรแกรมภาษาแลดเดอร์นั้น จะใช้เวลาในการ

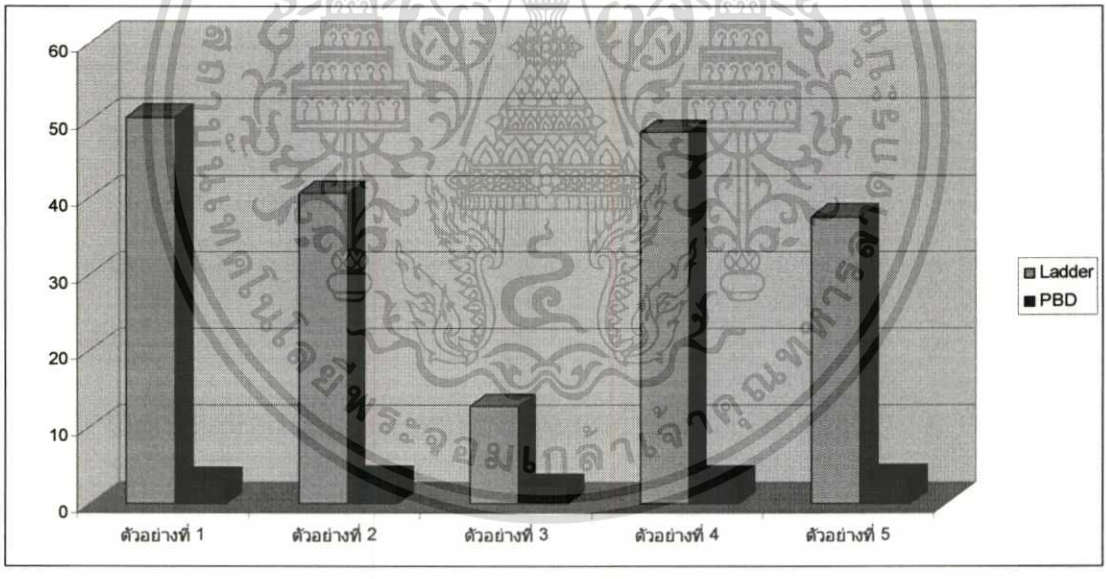
โปรแกรมมากกว่าวิธีการโปรแกรมโดยใช้การสาธิต ดังในรูปที่ 4.3 จากการทดสอบในครั้งนี้มีข้อสังเกตระยะเวลาของการโปรแกรมที่ใช้จำแนกตามกลุ่มตัวอย่างดังแสดงในรูปที่ 4.4 และรูปที่ 4.5 ได้ว่าบางกลุ่มตัวอย่างทำเวลาของแบบทดสอบที่ 1 ได้ดีกว่าแบบทดสอบที่ 2 แต่ในทางกลับกันบางกลุ่มตัวอย่างทำเวลาของแบบทดสอบที่ 2 ได้ดีกว่าแบบทดสอบที่ 1 ซึ่งนั่นขึ้นกับทักษะประสบการณ์หรือความถนัดของแต่ละบุคคลนั่นเอง



รูปที่ 4.3 แสดงเวลาเฉลี่ยที่ใช้จำแนกตามกลุ่มตัวอย่าง (หน่วยเป็นนาที)



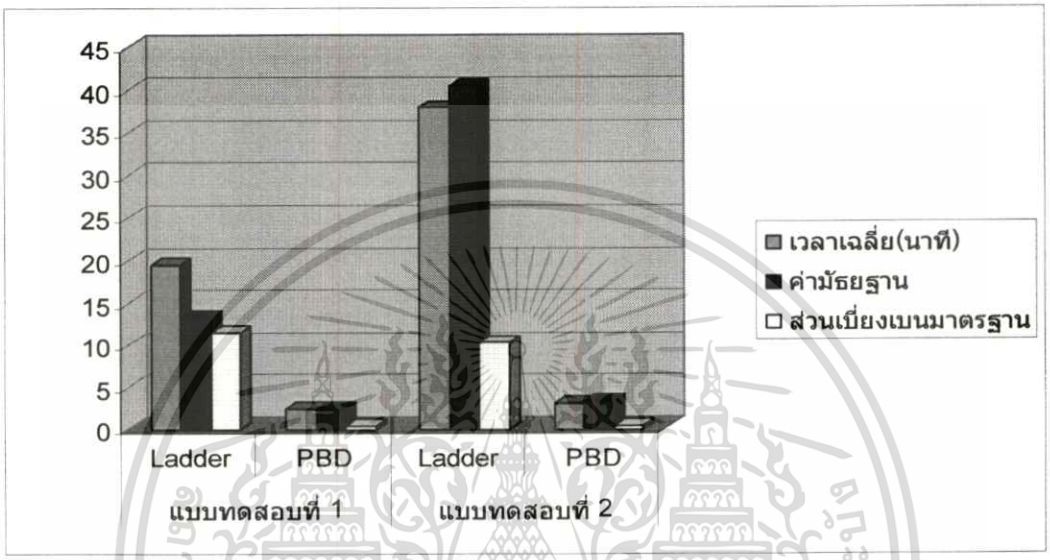
รูปที่ 4.4 แสดงเวลาที่ใช้ในแบบทดสอบที่ 1 จำแนกตามกลุ่มตัวอย่างและวิธีโปรแกรม (หน่วยเป็นนาที)



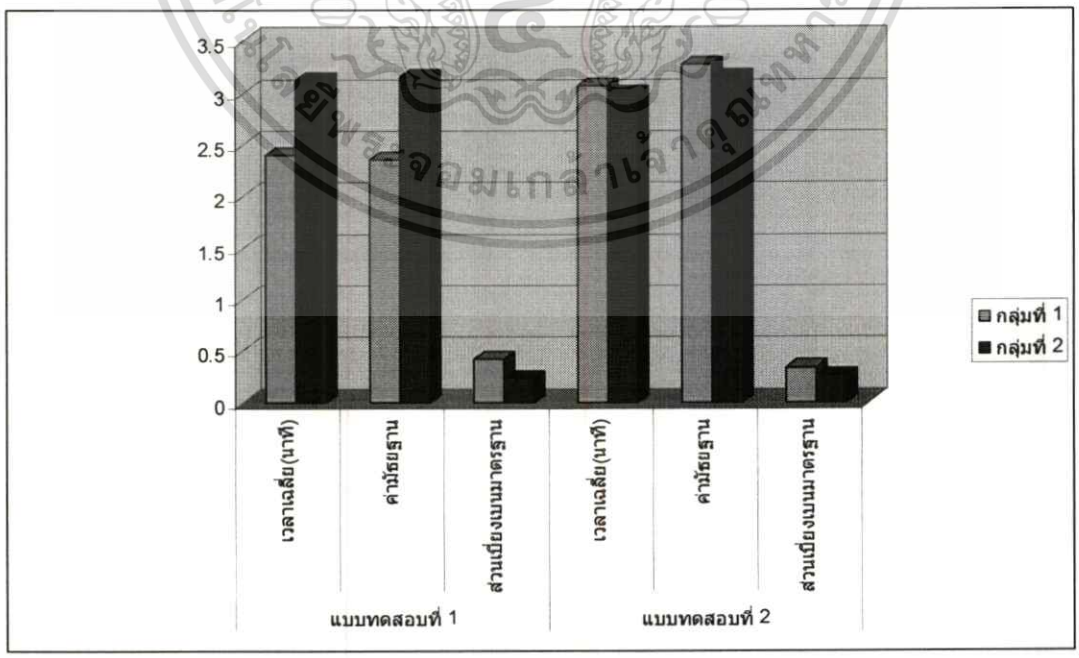
รูปที่ 4.5 แสดงเวลาที่ใช้ในแบบทดสอบที่ 2 จำแนกตามกลุ่มตัวอย่าง และวิธีโปรแกรม (หน่วยเป็นนาที)

ระยะเวลาที่ใช้ไปในการเขียนโปรแกรมด้วยภาษาแลดเดอร์ของแต่ละกลุ่มตัวอย่างมีความแตกต่างกันมาก ซึ่งขึ้นอยู่กับประสบการณ์หรือความถนัดของแต่ละคนโดยสังเกตได้จากค่าส่วนเบี่ยงเบนมาตรฐานของการเขียนโปรแกรมด้วยภาษาแลดเดอร์นั้นสูงถึง 11.496 และ 10.254 ส่วนวิธีการโปรแกรมโดยใช้การสาธิตจะใช้เวลาที่ไม่แตกต่างกันมากส่วนเบี่ยงเบนมาตรฐานอยู่ที่ 0.417

และ 0.337 นั่นหมายความว่าทักษะประสบการณ์หรือความถนัดนั้นมีผลน้อยเมื่อเทียบกับการเขียนโปรแกรมด้วยภาษาแลดเดอร์ดังแสดงในรูปที่ 4.6 อีกทั้งผลการทดสอบระหว่างกลุ่มที่ 1 ซึ่งเป็นผู้ที่มีความรู้ด้านอุปกรณ์พีแอลซี ด้านการเขียนโปรแกรม ด้านไฟฟ้าอิเล็กทรอนิกส์ และด้านระบบควบคุมเครื่องจักรอัตโนมัติ เมื่อเปรียบเทียบกับกลุ่มที่ 2 ซึ่งเป็นผู้ที่ไม่มีความรู้ด้านดังกล่าวพบว่าไม่มีความแตกต่างกันดังแสดงในรูปที่ 4.7



รูปที่ 4.6 แสดงเวลาเฉลี่ย/ ค่ามัธยฐาน/ ส่วนเบี่ยงเบนมาตรฐาน



รูปที่ 4.7 แสดงเวลาเฉลี่ย/ ค่ามัธยฐาน/ ส่วนเบี่ยงเบนมาตรฐาน ระหว่างกลุ่มที่ 1 และ 2

การบันทึกจำนวนเหตุการณ์ต่างๆ ในระหว่างการโปรแกรมนั้น เพื่อเป็นข้อสรุปว่าวิธีการโปรแกรมโดยใช้การสาธิตนั้นจะช่วยลดข้อผิดพลาดและระยะเวลาในการโปรแกรมลงได้หรือไม่ โดยมีรายละเอียดการบันทึกดังนี้

1) Syntax หมายถึง ใช้รูปแบบคำสั่งผิดพลาด (Syntax error)

2) Renew หมายถึง มีการเริ่มต้นเขียนโปรแกรมใหม่ทั้งหมดหรือเฉพาะกลุ่มของงาน

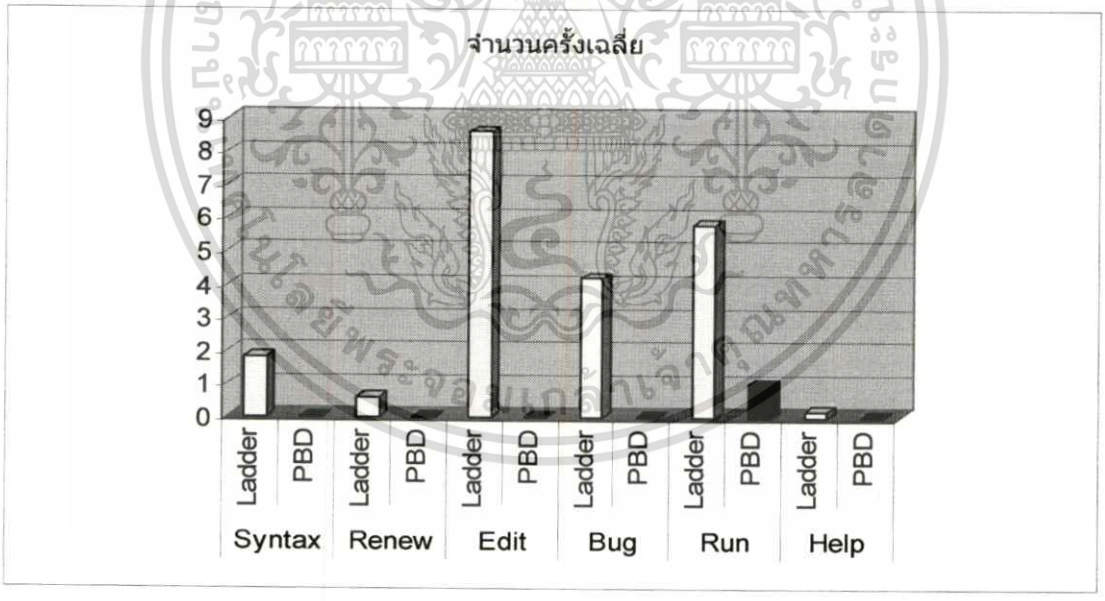
3) Edit หมายถึง มีการแก้ไขชุดคำสั่งเดิม

4) Bug หมายถึง เมื่อสั่งทำงานตามโปรแกรมแล้วได้ผลลัพธ์ไม่ตรงตามเงื่อนไข

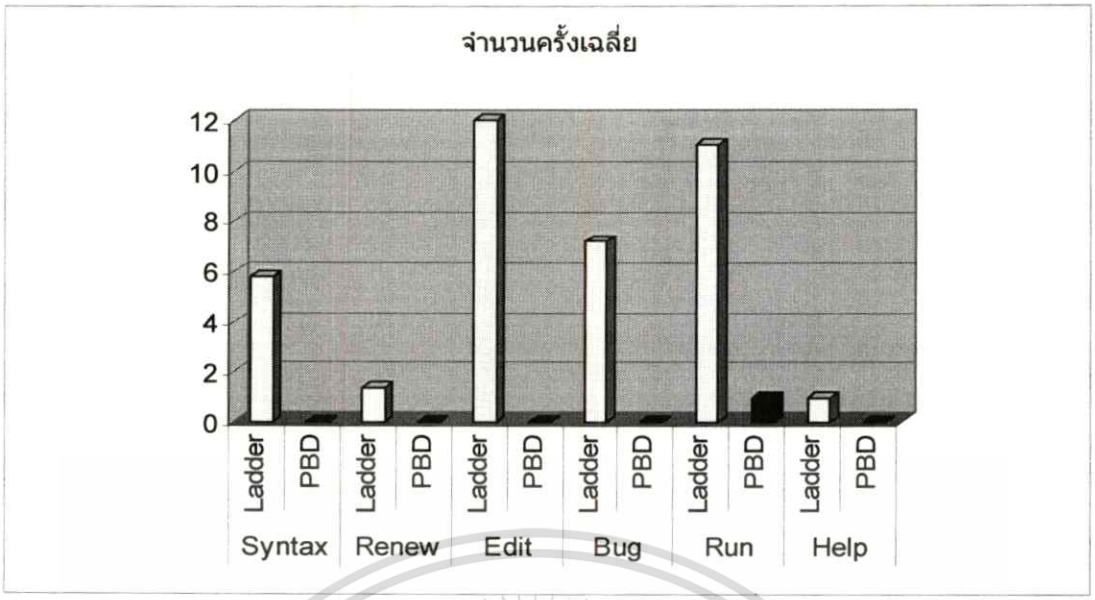
5) Run หมายถึง สั่งทำงานตามโปรแกรม

6) Help หมายถึง การเปิดตัวช่วยในการใช้ชุดคำสั่ง

ผลการบันทึกจำนวนเหตุการณ์นั้นชี้ให้เห็นว่าจากแบบทดสอบที่ 1 และแบบทดสอบที่ 2 วิธีการเขียนโปรแกรมภาษาแลดเดอร์นั้นกลุ่มตัวอย่างที่เข้าร่วมทดสอบมีการใช้รูปแบบคำสั่งผิดพลาด (Syntax error) มีการเริ่มต้นเขียนโปรแกรมใหม่ การแก้ไขชุดคำสั่งเดิม และเมื่อสั่งทำงานตามโปรแกรมแล้วได้ผลลัพธ์ไม่ตรงตามเงื่อนไขมีจำนวนสูงกว่าวิธีการโปรแกรมโดยใช้การสาธิต ดังแสดงรายละเอียดในรูปที่ 4.8 และรูปที่ 4.9



รูปที่ 4.8 แสดงจำนวนเหตุการณ์เฉลี่ยระหว่างการโปรแกรมของแบบทดสอบที่ 1



รูปที่ 4.9 แสดงจำนวนเหตุการณ์เฉลี่ยระหว่างการโปรแกรมของแบบทดสอบที่ 2

4.4 ผลการตอบแบบสอบถามจากกลุ่มตัวอย่างจำนวน 5 ตัวอย่าง

4.4.1 แบบสอบถามก่อนการทดสอบจำนวน 13 ข้อ

1) ระดับการศึกษา กลุ่มตัวอย่างทั้งหมดจบการศึกษาระดับปริญญาตรี

2) การสำเร็จการศึกษาในสาขาที่เกี่ยวข้องกับการใช้งานพีแอลซีโดยตรง กลุ่มตัวอย่างสำเร็จการศึกษาในสาขาที่เกี่ยวข้องกับการใช้งานพีแอลซีโดยตรง 4 คนและไม่ตรง 1 คน

3) การทำงานเกี่ยวกับเครื่องจักรอัตโนมัติ กลุ่มตัวอย่างเคยทำงานเกี่ยวกับเครื่องจักรอัตโนมัติ 4 คน และไม่เคย 1 คน

4) การควบคุมการทำงานของเครื่องจักรเองด้วยมือ กลุ่มตัวอย่างเคยควบคุมการทำงานของเครื่องจักรเองด้วยมือ 4 คน และไม่เคย 1 คน

5) การใช้คอมพิวเตอร์ในการควบคุมการทำงานของเครื่องจักร กลุ่มตัวอย่างเคยใช้คอมพิวเตอร์ในการควบคุมการทำงานของเครื่องจักร 3 คน และไม่เคย 2 คน

6) ความรู้ด้านไฟฟ้าอิเล็กทรอนิกส์ กลุ่มตัวอย่างมีความรู้ด้านไฟฟ้าอิเล็กทรอนิกส์ในระดับที่บำรุงรักษาและแก้ไขได้ 2 คน และสามารถออกแบบวงจรไฟฟ้าอิเล็กทรอนิกส์ได้ 4 คน

7) ประเภทการใช้งานโปรแกรมคอมพิวเตอร์ กลุ่มตัวอย่างใช้โปรแกรมคอมพิวเตอร์ในงานทั่วไป (Office, Internet ฯลฯ) 4 คน ใช้งานออกแบบ (CAD/CAM) 1 คน ใช้เขียนโปรแกรมภาษา (Pascal ฯลฯ) ทั้ง 5 คน และใช้โปรแกรมเครื่องจักรอัตโนมัติ 4 คน

8) ภาษาที่ใช้เขียนโปรแกรมภาษา กลุ่มตัวอย่างเคยเขียนโปรแกรมโดยใช้ภาษา ASM 4 คน ใช้ภาษา Pascal 2 คน ใช้ภาษา C ทั้ง 5 คน ใช้ภาษา Basic 4 คน ใช้ภาษา Java 1 คน และใช้ภาษาอื่นๆ 2 คน

9) ความรู้เกี่ยวกับระบบควบคุมการทำงานเครื่องจักรอัตโนมัติด้วยการสั่งงานแบบลำดับ (Sequence Control) กลุ่มตัวอย่างทั้งหมดมีเกี่ยวกับระบบควบคุมการทำงานเครื่องจักรอัตโนมัติด้วยการสั่งงานแบบลำดับในระดับที่สามารถออกแบบและโปรแกรมขั้นตอนการทำงานได้

10) ความสนใจในการนำอุปกรณ์พีแอลซีไปประยุกต์ใช้ในการทำงาน กลุ่มตัวอย่างมีความสนใจมากในการนำอุปกรณ์พีแอลซีไปประยุกต์ใช้ในการทำงาน 4 คน และสนใจปานกลาง 1 คน

11) ความรู้เกี่ยวกับอุปกรณ์พีแอลซี กลุ่มตัวอย่างทั้งหมดมีความรู้เกี่ยวกับอุปกรณ์พีแอลซี

12) ระยะเวลาใช้อุปกรณ์พีแอลซี กลุ่มตัวอย่างเคยใช้อุปกรณ์พีแอลซีเป็นระยะเวลา 2 ถึง 6 เดือน 1 คน และมากกว่า 1 ปี 4 คน

13) ภาษาที่ใช้เขียนโปรแกรมอุปกรณ์พีแอลซี กลุ่มตัวอย่างเคยเขียนโปรแกรมอุปกรณ์พีแอลซีด้วยภาษานีโมนิค 4 คน ภาษาแลดเดอร์ทั้ง 5 คน ภาษา Structure text 1 คน ภาษา Function block 2 คน และภาษาอื่นๆ 1 คน

4.4.2 แบบสอบถามหลังการทดสอบจำนวน 13 ข้อ

1) วิธีการโปรแกรมโดยใช้การสาธิตช่วยลดเวลาในการเรียนรู้หรือไม่ เมื่อเทียบกับภาษานีโมนิค และภาษาแลดเดอร์ กลุ่มตัวอย่างทั้งหมดคิดว่าการโปรแกรมด้วยวิธีการนี้ จะช่วยลดเวลาในการเรียนรู้ เมื่อเทียบกับภาษานีโมนิค และภาษาแลดเดอร์

2) วิธีการโปรแกรมโดยใช้การสาธิตช่วยลดเวลาในการโปรแกรมหรือไม่ เมื่อเทียบกับภาษานีโมนิค และภาษาแลดเดอร์ กลุ่มตัวอย่างทั้งหมดคิดว่าการโปรแกรมด้วยวิธีการนี้ จะช่วยลดเวลาในการโปรแกรม เมื่อเทียบกับภาษานีโมนิค และภาษาแลดเดอร์

3) วิธีการโปรแกรมโดยใช้การสาธิตช่วยลดความผิดพลาดในการโปรแกรมหรือไม่ เมื่อเทียบกับภาษานีโมนิค และภาษาแลดเดอร์ กลุ่มตัวอย่างคิดว่าการโปรแกรมด้วยวิธีการนี้ จะช่วยลดความผิดพลาดในการโปรแกรมเมื่อเทียบกับภาษานีโมนิค และภาษาแลดเดอร์4 คน และคิดว่าอาจจะเพิ่มข้อผิดพลาด 1 คน

4) วิธีการโปรแกรมโดยใช้การสาธิตมีผลต่อค่าใช้จ่ายในการทดสอบการทำงานของโปรแกรมกับเครื่องต้นแบบหรือไม่ เมื่อเทียบกับภาษานีโมนิค และภาษาแลดเดอร์ กลุ่มตัวอย่าง

ทั้งหมดคิดว่าการโปรแกรมด้วยวิธีการนี้ จะช่วยลดค่าใช้จ่ายในการทดสอบการทำงานของโปรแกรมกับเครื่องต้นแบบ เมื่อเทียบกับภาษานีโมนิค และภาษาแลดเดอร์

5) ความยากง่ายในการศึกษาเรียนรู้ เมื่อเทียบกับภาษานีโมนิค และภาษาแลดเดอร์ กลุ่มตัวอย่างทั้งหมดคิดว่าการโปรแกรมด้วยวิธีการนี้ง่ายในการศึกษาเรียนรู้ เมื่อเทียบกับภาษานีโมนิค และภาษาแลดเดอร์

6) การสร้างแบบจำลองมีความยากง่ายเพียงใด กลุ่มตัวอย่างคิดว่าเครื่องมือที่ใช้ในการโปรแกรมด้วยวิธีการนี้ ในส่วนของการสร้างแบบจำลองง่าย 4 คน และยาก 1 คน

7) การโปรแกรมการทำงานของแบบจำลองมีความยากง่ายเพียงใด กลุ่มตัวอย่างทั้งหมดคิดว่าเครื่องมือที่ใช้ในการโปรแกรมด้วยวิธีการนี้ ในส่วนของการโปรแกรมการทำงานของแบบจำลองมีความง่าย

8) การโปรแกรมการควบคุมขั้นตอนการทำงานของเครื่องจักรมีความยากง่ายเพียงใด กลุ่มตัวอย่างทั้งหมดคิดว่าเครื่องมือที่ใช้ในการโปรแกรมด้วยวิธีการนี้ ในส่วนของการโปรแกรมการควบคุมขั้นตอนการทำงานของเครื่องจักรมีความง่าย

9) การจัดการเหตุการณ์ที่ไม่ได้ควบคุมไว้ (Missing case) มีความยากง่ายเพียงใด กลุ่มตัวอย่างทั้งหมดคิดว่าเครื่องมือที่ใช้ในการโปรแกรมด้วยวิธีการนี้ ในส่วนของจัดการเหตุการณ์ที่ไม่ได้ควบคุมไว้ (Missing case) ง่าย

10) การแสดงภาพจำลองการทำงานของเครื่องจักรในขณะโปรแกรม ช่วยให้ท่านเข้าใจขบวนการทำงานได้มากกว่าภาษานีโมนิค และภาษาแลดเดอร์หรือไม่ กลุ่มตัวอย่างทั้งหมดคิดว่าการแสดงภาพจำลองการทำงานของเครื่องจักรในขณะโปรแกรมมีส่วนช่วยอย่างมากที่จะทำให้เข้าใจขบวนการทำงานได้มากกว่าภาษานีโมนิค และภาษาแลดเดอร์

11) ท่านจะใช้การโปรแกรมด้วยวิธีการนี้หรือไม่ กลุ่มตัวอย่างทั้งหมดคิดว่าจะใช้การโปรแกรมด้วยวิธีการนี้

12) การโปรแกรมด้วยวิธีการนี้ ควรนำมาสอนในสถานศึกษาหรือไม่ กลุ่มตัวอย่างทั้งหมดคิดว่าการโปรแกรมด้วยวิธีการนี้ ควรนำมาสอนในสถานศึกษา

13) ข้อเสนอแนะในการพัฒนาโปรแกรม กลุ่มตัวอย่างมีข้อเสนอแนะในการพัฒนาโปรแกรมดังนี้

13.1) ควรมีตัวหนังสือขั้นตอนการทำงานหรือสัญลักษณ์

13.2) ต้องการให้เพิ่มฟังก์ชันระดับสูงของพีแอลซี เช่น Analog I/O PID Control

13.3) เพิ่มตัวแปลงจากภาษาแลดเดอร์ที่ได้จากโปรแกรมให้สามารถโหลดลงพีแอลซีได้โดยตรง

13.4) การโหลดห้องสมุดอุปกรณ์ควรจะมีการแบ่งกลุ่มให้ชัดเจน

4.5 สรุปผลจากการสังเกตการทดลองกับกลุ่มตัวอย่าง

การสังเกตการทดลองกับกลุ่มตัวอย่างนั้นจะใช้วงจรพฤติกรรมมนุษย์ซึ่งเป็นโมเดลที่ใช้ในการประเมินสมรรถนะของส่วนเชื่อมต่อผู้ใช้ โดยปกติการประเมินส่วนเชื่อมต่อผู้ใช้จะใช้ชุดของคำถามในแต่ละรอบของขั้นตอน การประเมินคำตอบนั้นพิจารณาจากการแสดงข้อมูลที่เป็นประโยชน์ของส่วนเชื่อมต่อผู้ใช้ เพื่อสนับสนุนการใช้งานว่าเพียงพอหรือเหมาะสมกับการใช้งานหรือไม่ดังนี้

4.5.1 ขั้นที่ 1 สร้างเป้าหมาย (Forming a goal)

กลุ่มตัวอย่างที่เข้าร่วมทดสอบในครั้งนี้ เป็นผู้ที่มีความสนใจงานด้านระบบควบคุมเครื่องจักรอัตโนมัติ โดยมีทั้งผู้ที่ทำงานในด้านนี้ และเป็นผู้ที่สำเร็จการศึกษาในสาขาวิชาที่เกี่ยวข้องโดยตรง อีกทั้งกลุ่มตัวอย่างที่เข้าร่วมทดสอบยังมีความสามารถบำรุงรักษา แก้ไข ออกแบบวงจรไฟฟ้าเล็กทรอนิกส์ได้ สามารถออกแบบระบบควบคุมเครื่องจักรอัตโนมัติได้ รวมทั้งได้มีการอธิบายและสร้างความเข้าใจในแบบทดสอบก่อนลงมือปฏิบัติ ดังนั้นจึงถือได้ว่ากลุ่มตัวอย่างมีความเข้าใจและความรู้ในงาน พร้อมทั้งมีความเข้าใจงานเพียงพอในการที่จะสร้างเป้าหมายได้

4.5.2 ขั้นที่ 2 แปลงเป้าหมายไปสู่งานหรือชุดงาน (Translating the goal into a task or a set of tasks)

เป็นขั้นที่ผู้ใช้จะต้องกำหนดว่าจะต้องทำอะไรบ้างเพื่อบรรลุเป้าหมาย ซึ่งงาน (Task) ที่จะนำไปสู่เป้าหมาย (Goal) คือต้องการเขียนโปรแกรมควบคุมการทำงานตามข้อกำหนด ซึ่งวิธีการเขียนโปรแกรมภาษาแลดเดอร์และวิธีการโปรแกรมโดยใช้การสาธิตจะมีความแตกต่างกันดังนี้

1) วิธีการโปรแกรมโดยใช้การสาธิต

จากแบบทดสอบที่ 1 เป้าหมายคือการควบคุมการทำงานของประตูอัตโนมัติ โดยมีหลักการแปลงเป้าหมายไปสู่งานที่คล้ายกับการทำงานแบบ Manual ดังนี้

1.1) สร้างแบบจำลองของเครื่องจักร

1.2) จำลองเหตุการณ์คนเข้าและคนออก (Detect a person) ซึ่งผู้โปรแกรมจะกำหนดจากสถานะของอุปกรณ์ตรวจสอบ (Detector) สถานะเปิดหมายถึงมีคนเข้ามา และสถานะปิดหมายถึงไม่มีคน

1.3) สาธิตเปิดประตู (Opening a door demonstration) ผู้โปรแกรมจะเปิดมอเตอร์ที่ใช้เปิดประตู (Motor: MO) จนประตูเปิดจนสุดจึงหยุดมอเตอร์ที่ใช้เปิดประตู

1.4) สาธิตปิดประตู (Closing a door demonstration) ผู้โปรแกรมจะเปิดมอเตอร์ที่ใช้ปิดประตู (Motor: MC) จนประตูปิดจนสุดจึงหยุดมอเตอร์ที่ใช้ปิดประตู

2) วิธีการเขียนโปรแกรมภาษาแลดเดอร์

จากแบบทดสอบที่ 1 เป้าหมายคือการควบคุมการทำงานของประตูอัตโนมัติ โดยมีหลักการแปลงเป้าหมายไปสู่งานที่แตกต่างไปโดยสิ้นเชิง ดังนี้

2.1) สร้างตารางความจริง (Truth Table) ของตรรกะ (Logic) สำหรับการเปิด และปิดประตู โดยต้องทำด้วยมือในกระดาษ หรือการคิดไว้ในใจ

2.1.1) ตารางความจริงสำหรับการเปิดประตูเป็นดังนี้

ตารางที่ 4.3 แสดงตารางความจริงสำหรับการเปิดประตู

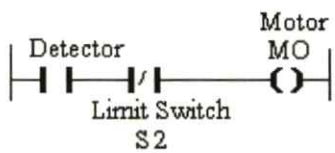
Detector	Limit Switch S2	Motor: MO
0	0	0
0	1	0
1	0	1
1	1	0

2.1.2) ตารางความจริงสำหรับการปิดประตูเป็นดังนี้

ตารางที่ 4.4 แสดงตารางความจริงสำหรับการปิดประตู

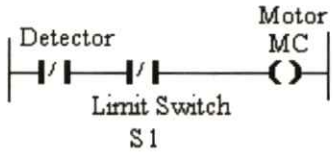
Detector	Limit Switch S1	Motor: MC
0	0	1
0	1	0
1	0	0
1	1	0

2.2) เขียนภาษาแลดเดอร์ในการเปิดประตู (Opening a door)



รูปที่ 4.10 แสดงการเขียนภาษาแลดเดอร์สำหรับการเปิดประตู

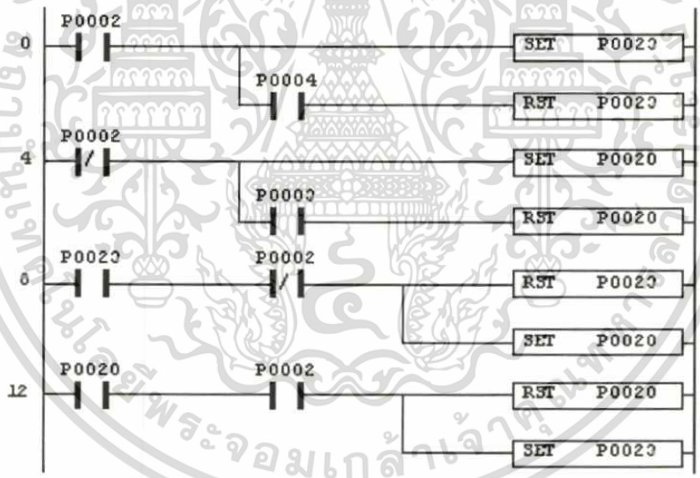
2.3) เขียนภาษาแลดเดอร์ในการปิดประตู (Closing a door)



รูปที่ 4.11 แสดงการเขียนภาษาแลดเดอร์สำหรับการปิดประตู

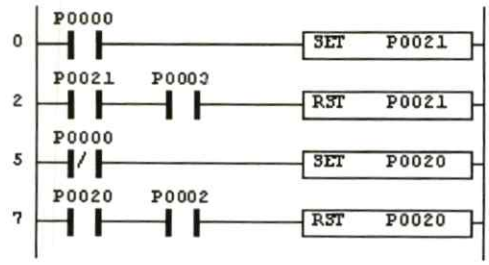
วิธีการเขียนโปรแกรมภาษาแลดเดอร์นั้น งานที่จะนำไปสู่เป้าหมายผู้เขียนสามารถแปลงงานสู่ลำดับของการปฏิบัติได้หลายรูปแบบ จากกลุ่มตัวอย่างจะมีรูปแบบของการเขียนที่แตกต่างกัน เช่น

รูปแบบแรก แยกงานออกเป็น เปิดประตู, ปิดประตู, เปิดประตูขณะประตูกำลังปิด และปิดประตูขณะประตูกำลังเปิด นำมาเขียนภาษาแลดเดอร์ได้ดังนี้



รูปที่ 4.12 แสดงการเขียนภาษาแลดเดอร์ในรูปแบบ ก

รูปแบบที่สอง แยกงานออกเป็น มอเตอร์เปิดประตูทำงาน มอเตอร์เปิดประตูหยุดทำงาน มอเตอร์ปิดประตูทำงาน และมอเตอร์เปิดประตูหยุดทำงาน นำมาเขียนภาษาแลดเดอร์ ได้ดังนี้



รูปที่ 4.13 แสดงการเขียนภาษาแลดเดอร์ในรูปแบบ ข

ผลจากการวิเคราะห์สังเกตการแปลงเป้าหมายไปสู่งานของทั้งสองวิธีพบว่าวิธีการเขียนโปรแกรมภาษาแลดเดอร์จะต้องอาศัยทักษะความรู้ในวิชาตรรกวิทยาและวิชาที่ว่าด้วยรูปแบบการเขียนโปรแกรมภาษาแลดเดอร์เป็นสำคัญ จึงจะสามารถแปลงเป้าหมายไปสู่งานได้ ซึ่งต่างจากวิธีการโปรแกรมโดยใช้การสาธิตที่การแปลงเป้าหมายไปสู่งานนั้นคล้ายกับการทำงานแบบ Manual ซึ่งทำให้ผู้เขียนไม่ต้องอาศัยทักษะตรรกวิทยาหรือรูปแบบการเขียนโปรแกรมแต่อย่างใด

4.5.3 ขั้นที่ 3 วางแผนงานเพื่อสร้างลำดับการปฏิบัติ (Planning an action sequence)

ขั้นที่ 3 นี้จะเป็นการวางแผนว่าจะทำแต่ละงานนั้นอย่างไร ในการแปลงแต่ละงานสู่ลำดับของการปฏิบัติ (Action Sequence) โดยที่เครื่องมือที่ใช้ในการโปรแกรมทั้งสองวิธีนั้นจะมีความแตกต่างกันดังนี้

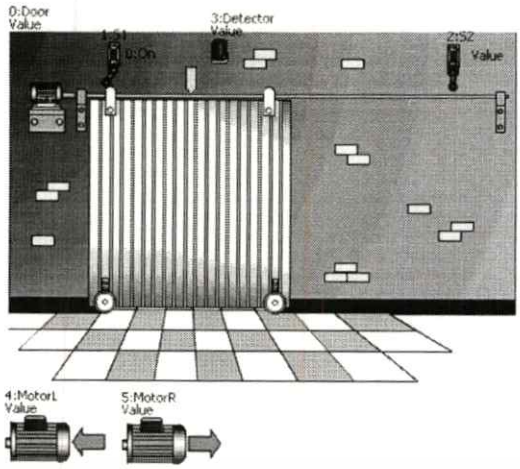
1) วิธีการโปรแกรมโดยใช้การสาธิต

วิธีการโปรแกรมโดยใช้การสาธิต มีขั้นตอนการแปลงแต่ละงานสู่ลำดับของการปฏิบัติ (Action Sequence) ดังนี้

1.1) งานที่ 1 สร้างแบบจำลอง วิธีการโปรแกรมโดยใช้การสาธิตนั้นจะต้องมีการสร้างแบบจำลองขึ้นก่อน ซึ่งแบบจำลองนั้นสามารถสร้างด้วยอุปกรณ์สำเร็จรูปจากห้องสมุดอุปกรณ์โดยมีขั้นตอนการสร้างดังนี้

1.1.1) เลือกใช้อุปกรณ์ที่ใช้ในการสร้างแบบจำลอง จากห้องสมุดอุปกรณ์

1.1.2) วางอุปกรณ์บนพื้นที่ทำงานตามโครงสร้างเครื่องจักรจริง



รูปที่ 4.14 แสดงการวางอุปกรณ์ในการสร้างแบบจำลอง

1.1.3) กำหนดค่าเริ่มต้นของอุปกรณ์โดยการปรับแต่งค่าคุณสมบัติ

1.2) งานที่ 2 จำลองเหตุการณ์คนเข้าและคนออกซึ่งจะกำหนดสถานะของอุปกรณ์ตรวจสอบ

1.2.1) คลิกที่อุปกรณ์เพื่อปรับเปลี่ยนสถานะ หากสถานะเปิดหมายถึงมีคนเข้ามา และสถานะปิดหมายถึงไม่มีคน

1.2.2) เมื่อมีคนเข้าหรือออก เครื่องมือจะตรวจสอบว่ามีการโปรแกรมไว้หรือไม่ หากไม่มีเครื่องมือจะหยุดถามเพื่อให้ผู้ใช้สาธิตการเปิดประตูและปิดประตูต่อไป

1.3) งานที่ 3 สาธิตการเปิดประตูการโปรแกรมการควบคุมลงบนเครื่องมือการเขียนโปรแกรมจากกลุ่มตัวอย่างไม่มีความแตกต่างกันมาก เนื่องจากการโปรแกรมนั้นจะเป็นไปตามเหตุการณ์ที่เกิดขึ้นคล้ายกับการทำงานแบบ Manual โดยมีรายละเอียดดังนี้

1.3.1) การเปิดประตู เริ่มจากผู้โปรแกรมเปิดมอเตอร์ที่ใช้เปิดประตู คือมอเตอร์ MO โดยการคลิกที่มอเตอร์ MO จากนั้นผู้ใช้เปลี่ยนโหมดจำลองเหตุการณ์

1.3.2) เครื่องมือจะหยุดถามว่ามอเตอร์ MO เปิดจะเกิดอะไรขึ้น ผู้โปรแกรมจะต้องทำการเลื่อนบานประตูเพื่อเปิดออก จากนั้นสั่งให้เครื่องมือจำลองเหตุการณ์ เครื่องมือจะเปิดประตูจนสุดทำให้ Limit Switch S2 มีสถานะเปิด

1.3.3) เครื่องมือจะหยุดถามว่า Limit Switch S2 เปิดจะเกิดอะไรขึ้น ผู้โปรแกรมจะต้องจะต้องปิดมอเตอร์ MO โดยการคลิกที่มอเตอร์ MO เป็นอันจบงานที่ 3

1.4) งานที่ 4 สาธิตการปิดประตูการลงมือโปรแกรมการควบคุมลงบนเครื่องมือการเขียนโปรแกรมมีรายละเอียดดังนี้

1.4.1) การปิดประตู เริ่มจากผู้โปรแกรมเปิดมอเตอร์ที่ใช้ปิดประตู คือมอเตอร์ MC โดยการคลิกที่มอเตอร์ MC จากนั้นผู้ใช้เปลี่ยนโหมดจำลองเหตุการณ์

1.4.2) เครื่องมือจะหยุดถามว่ามอเตอร์ MC เปิดจะเกิดอะไรขึ้น ผู้โปรแกรมจะต้องทำการเลื่อนบานประตูเพื่อปิดจากนั้นสั่งให้เครื่องมือจำลองเหตุการณ์ เครื่องมือจะปิดประตูจนสุดทำให้ Limit Switch S1 มีสถานนะเปิด

1.4.4) เครื่องมือจะหยุดถามว่า Limit Switch S1 เปิดจะเกิดอะไรขึ้น ผู้โปรแกรมจะต้องจะต้องปิดมอเตอร์ MC โดยการคลิกที่มอเตอร์ MC เป็นอันจบงานที่ 4

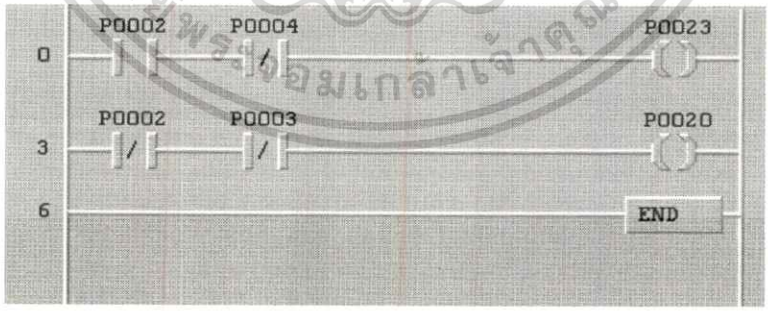
2) วิธีการเขียนโปรแกรมภาษาแลดเดอร์

วิธีการเขียนโปรแกรมภาษาแลดเดอร์ มีขั้นตอนการแปลงแต่ละงานสู่ลำดับของการปฏิบัติดังนี้

2.1) งานที่ 1 กำหนดหมายเลขช่องสัญญาณ (Port number) แทนอุปกรณ์ต่างๆ ดังนี้ Detector หมายเลขช่องสัญญาณคือ P0002, Limit Switch S2 หมายเลขช่องสัญญาณคือ P0004 และ Motor: MO หมายเลขช่องสัญญาณคือ P0023 ซึ่งหมายเลขดังกล่าวจะถูกกำหนดจากการต่ออุปกรณ์นั้นๆ เข้ากับอุปกรณ์พีแอลซี

2.2) จากนั้นจะเริ่มการโปรแกรมโดยเริ่มจากงานที่ 2 เขียนภาษาแลดเดอร์ในการเปิดประตูโดยการคลิกเลือกอุปกรณ์จากแถบเครื่องมือหรือจากแถบเมนูมาวางลงบนตาราง (Grid) จนครบ

2.3) จากนั้นจะเริ่มงานที่ 3 คือเขียนภาษาแลดเดอร์ในการปิดประตูโดยที่งานใหม่นี้จะสร้างในบรรทัดถัดไปจากงานชุดแรก ซึ่งแต่ละงานนี้เราเรียกว่า Rung และจบด้วยคำสั่ง End เสมอ ซึ่งชุดคำสั่งที่ได้จะเป็นไปตามขั้นตอนที่ 2 การสร้างงานที่จะนำไปสู่เป้าหมายซึ่งผู้เขียนสามารถแปลงงานสู่ลำดับของการปฏิบัติได้หลายรูปแบบ



รูปที่ 4.15 แสดงชุดคำสั่งในการเปิดและปิดประตูในภาษาแลดเดอร์

ผลจากการวิเคราะห์สังเกตพบว่า วิธีการโปรแกรมโดยใช้การสาธิตนั้นจะเริ่มจากการสร้างแบบจำลองของเครื่องจักรที่สามารถวางอุปกรณ์ในรูปแบบเช่นเดียวกันกับเครื่องจักรจริง จึงทำให้ผู้ใช้สามารถมองเห็นถึงความสัมพันธ์ทางกายภาพระหว่างอุปกรณ์แต่ละตัว ซึ่งช่วยให้ผู้ใช้ทำความ

เข้าใจกับรูปแบบการทำงานได้ง่ายขึ้น และการโปรแกรมการควบคุมนั้นจะเป็นไปตามเหตุการณ์ที่เกิดขึ้นคล้ายกับการทำงานแบบ Manual ทำให้ผู้ใช้ไม่จำเป็นต้องมีทักษะการเขียนโปรแกรมแต่อย่างใด ส่วนวิธีการเขียนโปรแกรมภาษาแลดเดอร์จะเริ่มจากการกำหนดหมายเลขช่องสัญญาณแทนอุปกรณ์ต่างๆ ซึ่งหมายเลขดังกล่าวจะถูกกำหนดจากการต่ออุปกรณ์นั้นๆ เข้ากับอุปกรณ์พีแอลซี จากนั้นจะเป็นการวางอุปกรณ์ที่จะต้องวางตามรูปแบบของตรรกะที่ได้จากการสร้างตาราง ความจริงอีกอย่างที่ต้องคำนึงถึงคือ การวางอุปกรณ์จะต้องเป็นไปตามกฎเกณฑ์ที่เครื่องมือสามารถตีความได้ รวมไปถึงอุปกรณ์ตัวเดียวกันอาจมีการวางไว้หลายๆ จุดตามความเกี่ยวพันกับงานต่างๆ เช่น P002 จะมีการวาง 2 จุด ทั้งที่เป็นอุปกรณ์ตัวเดียวกัน ทำให้อุปกรณ์ที่วางมีจำนวนมากกว่าจำนวนอุปกรณ์ตามความเป็นจริง

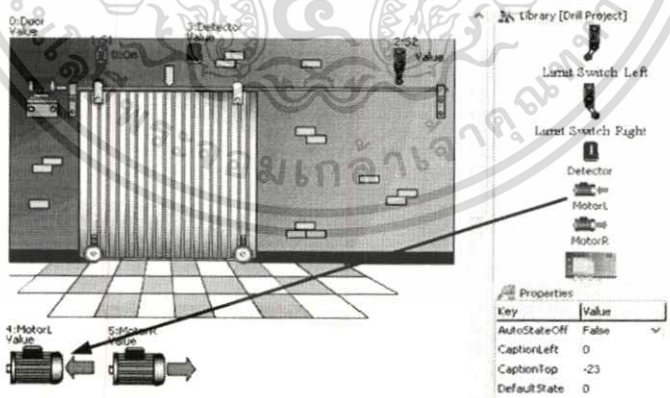
4.5.4 ขั้นที่ 4 ปฏิบัติตามลำดับการปฏิบัติ (Executing the action sequence)

ในขั้นที่ 4 นี้จะลงมือเขียนโปรแกรมตามลำดับของการปฏิบัติซึ่งเป็นผลลัพธ์จากขั้นที่ 3 โดยที่เครื่องมือที่ใช้ในการโปรแกรมทั้งสองวิธีนั้นมีความแตกต่างกันดังนี้

1) วิธีการโปรแกรมโดยใช้การสาธิต

1.1) สร้างแบบจำลอง โดยมีขั้นตอนการสร้างดังนี้

1.1.1) การวางอุปกรณ์ทำได้โดยคลิกที่อุปกรณ์ แล้วลากมาวางบนพื้นที่ทำงาน โดยจัดวางตามโครงสร้างของเครื่องจักรจริง ทำให้สามารถมองเห็นถึงความเชื่อมโยงทางกายภาพของอุปกรณ์ต่างๆ ที่นำมาจัดวางได้อย่างชัดเจน

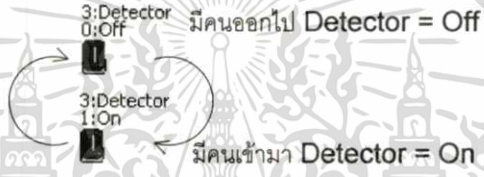


รูปที่ 4.16 แสดงการเลือกอุปกรณ์เพื่อสร้างแบบจำลอง

1.1.2) เมื่อวางอุปกรณ์ ในบางครั้งอาจจะต้องมีการปรับแต่งค่าเริ่มต้นของอุปกรณ์โดยการกำหนดในส่วนของค่าคุณสมบัติ (Properties)

รูปที่ 4.17 แสดงหน้าต่างการกำหนดค่าคุณสมบัติ

1.2) จำลองเหตุการณ์คนเข้าและคนออกโดยการคลิกที่อุปกรณ์เพื่อปรับเปลี่ยนสถานะ โดยเป็นลักษณะสลับสถานะไปมา (Toggle switch)



รูปที่ 4.18 แสดงจำลองเหตุการณ์คนเข้าและคนออก

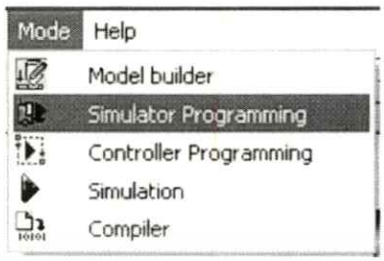
1.3) การสาธิตการเปิดประตูและปิดประตูนั้น การโปรแกรมการควบคุมลงบนเครื่องมือการเขียนโปรแกรมจะเป็นไปตามเหตุการณ์ที่เกิดขึ้นคล้ายกับการทำงานแบบ Manual โดยมีรายละเอียดดังนี้

1.3.1) การโปรแกรมเริ่มจากผู้ใช้สั่งให้เครื่องมือจำลองเหตุการณ์โดยการคลิกที่ปุ่มเริ่มการจำลอง (Run) ซึ่งใช้สัญลักษณ์สามเหลี่ยมสีเขียว การหยุดการจำลองเหตุการณ์ใช้สัญลักษณ์สี่เหลี่ยมสีน้ำเงิน และการบันทึกการกระทำของผู้ใช้ใช้สัญลักษณ์วงกลมสีแดง ซึ่งเป็นสัญลักษณ์ที่ใช้กันโดยทั่วไป

รูปที่ 4.19 แสดงสัญลักษณ์ปุ่มควบคุมการจำลองเหตุการณ์

1.3.2) การเปลี่ยนโหมดการทำงานจะเกิดขึ้นในสองกรณีคือ กรณีที่หนึ่งเครื่องมือตรวจสอบการเปลี่ยนสถานะของอุปกรณ์พบว่าการเปลี่ยนแปลงนั้นยังไม่ได้มีการ

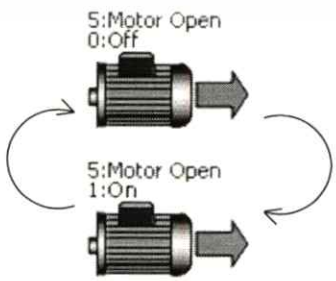
โปรแกรมไว้ เครื่องมือจะหยุดการจำลองเหตุการณ์ และเปลี่ยนโหมดเข้าสู่การโปรแกรมโดยอัตโนมัติ กรณีที่สองผู้ใช้ต้องการโปรแกรมเพิ่มเติม ผู้ใช้สามารถเปลี่ยนโหมดได้เองโดยการเลือกจากเมนู “Mode” หรือ เลือกจากแถบการเปลี่ยนโหมดได้โดยสะดวก



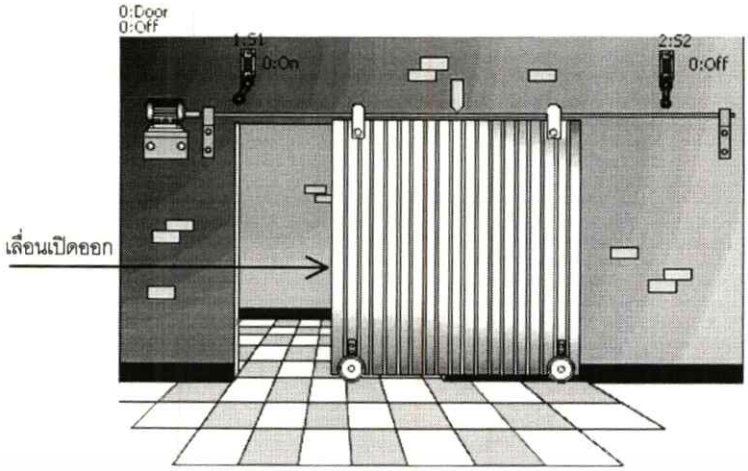
รูปที่ 4.20 แสดงการเปลี่ยนโหมดโดยใช้เมนู

รูปที่ 4.21 แสดงการเปลี่ยนโหมดโดยใช้แถบการเปลี่ยนโหมด

1.3.3) การโปรแกรมการควบคุมจะกระทำต่อเมื่อเครื่องมือหยุดถาม หรือผู้ใช้ต้องการโปรแกรมการควบคุมเพิ่มเติม ซึ่งการโปรแกรมการควบคุมนั้นเป็นการเปลี่ยนสถานะของอุปกรณ์ โดยสามารถกระทำได้สองลักษณะ คือลักษณะแรกเปลี่ยนสถานะเปิดและปิด (Switch on / Switch off) โดยการคลิกที่อุปกรณ์เพื่อเปลี่ยนสถานะ การเปลี่ยนสถานะนั้นจะเป็นลักษณะสลับไปมา เช่น เปิดและปิดมอเตอร์ ลักษณะที่สองเปลี่ยนแปลงลักษณะทางกายภาพของอุปกรณ์ โดยการคลิกและลากตัวชี้ตำแหน่งของเมาส์ (Drag) ไปบนตัวอุปกรณ์ เช่น การเปิดและปิดประตู



รูปที่ 4.22 แสดงการเปลี่ยนสถานะเปิด/ปิดโดยการคลิกที่อุปกรณ์



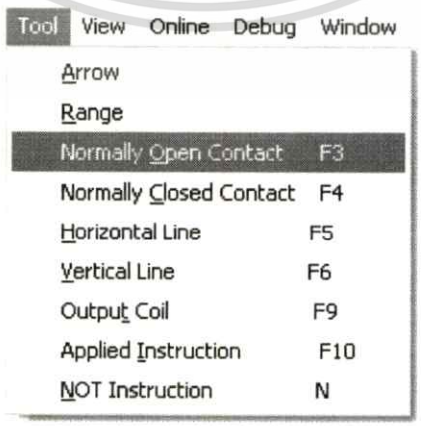
รูปที่ 4.23 แสดงการเปลี่ยนแปลงลักษณะทางกายภาพของอุปกรณ์ โดยการคลิกและลากตัวชี้ตำแหน่งของเมาส์ (Drag) ไปบนตัวอุปกรณ์

2) วิธีการเขียนโปรแกรมภาษาแลดเดอร์

เขียนภาษาแลดเดอร์ลงบนเครื่องมือการเขียนโปรแกรมมีรายละเอียดดังนี้

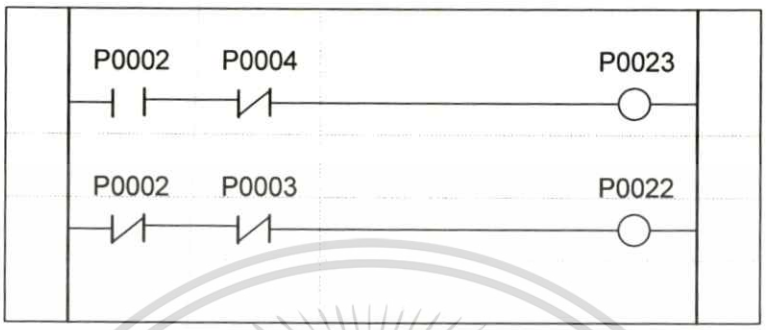
2.1) อุปกรณ์จะเป็นลักษณะของตรรกะ การเลือกนั้นกระทำได้สองวิธี วิธีแรกคือเลือกไอคอน (Icon) จากแถบเครื่องมือ (Tools bar) วิธีที่สองคือเลือกรายการจากแถบเมนู (Menu bar)

รูปที่ 4.24 แสดงการเลือกอุปกรณ์จากแถบเครื่องมือ



รูปที่ 4.25 แสดงการเลือกอุปกรณ์จากแถบเมนู

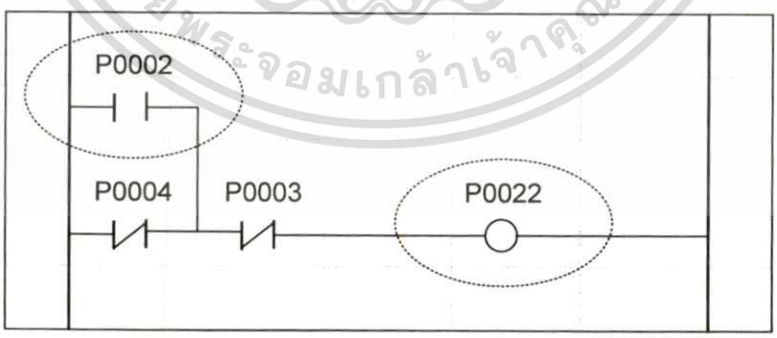
2.2) พื้นที่วางอุปกรณ์นั้นจะเป็นตาราง (Grid) โดยจะมีเส้นแนวเริ่มต้นและเส้นแนวจบ และใน 1 ช่องจะวางอุปกรณ์ได้ 1 ตัวโดยที่เส้นเชื่อมระหว่างอุปกรณ์ก็ถือว่าเป็นอุปกรณ์ 1 ตัวด้วยเช่นกัน และจะต้องเป็นไปตามกฎเกณฑ์ที่เครื่องมือสามารถตีความได้ เช่น อุปกรณ์ที่เป็นเอาท์พุทจะต้องอยู่ติดกับเส้นจบเท่านั้น ห้ามมีอุปกรณ์ใดๆ อยู่ก่อนแนวเส้นหลัก



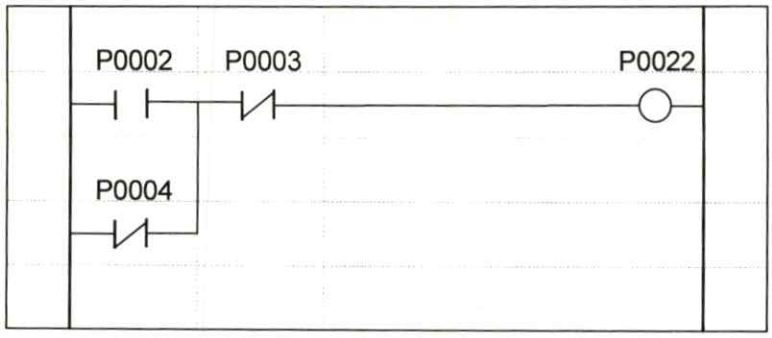
รูปที่ 4.26 แสดงการวางอุปกรณ์ลงบนตาราง (Grid)

2.3) เมื่อวางอุปกรณ์แล้วจะต้องกำหนดคุณสมบัติของอุปกรณ์ หรือการเขียนชุดคำสั่งในกรณีที่อุปกรณ์ที่วางนั้นเป็นชนิด Applied instruction ซึ่งเป็นอุปกรณ์ลักษณะปลายเปิดให้สามารถกำหนดคำสั่งที่ต้องการได้เอง

2.4) ลักษณะการวางอุปกรณ์นั้นจะต้องเป็นไปตามกฎเกณฑ์ที่เครื่องมือสามารถตีความได้ จากกลุ่มตัวอย่างมีการวางรูปแบบผิดบ่อยครั้ง จากรูปที่ 4.27 จะเห็นได้ว่า P0002 นั้นอยู่เหนือแนวเส้นหลัก และ P0022 นั้นเป็นอุปกรณ์ที่เป็นเอาท์พุทแต่ไม่อยู่ชิดแนวเส้นจบ

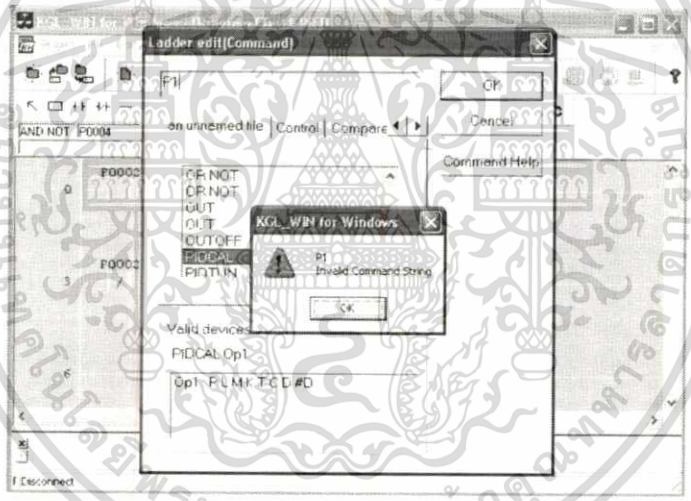


รูปที่ 4.27 แสดงการวางอุปกรณ์ที่ไม่เป็นไปตามกฎเกณฑ์



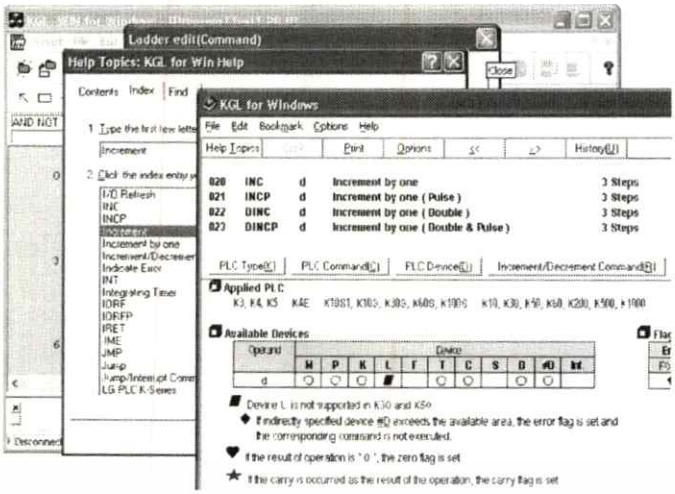
รูปที่ 4.28 แสดงการวางอุปกรณ์ที่เป็นไปตามกฎเกณฑ์

2.5) เมื่อวางอุปกรณ์เครื่องมือจะให้กำหนดคุณสมบัติของอุปกรณ์ หรือชุดคำสั่งที่เป็นลักษณะ Command Base จากการสังเกตกลุ่มตัวอย่างพบว่ามีการเขียนคำสั่งที่ไม่เป็นไปตามหลักไวยากรณ์ของภาษาแลดเดอร์ (Syntax error)



รูปที่ 4.29 แสดงการใช้คำสั่งที่ผิดหลักไวยากรณ์

2.6) อีกประการหนึ่งที่พบกับกลุ่มตัวอย่างคือการจดจำชุดคำสั่งไม่ได้ เนื่องจากชุดคำสั่งในภาษาแลดเดอร์มีจำนวนมาก ผู้ใช้ไม่สามารถจดจำรูปแบบและเงื่อนไขการทำงานว่าตรงกับเงื่อนไขการทำงานที่ต้องการใช้หรือไม่ รวมถึงรูปแบบการทำงานและการกำหนดค่าตัวแปรเสริม (Parameter) ของคำสั่งได้ทั้งหมด ทำให้ผู้ใช้จำเป็นต้องเรียกใช้ตัวช่วย (Help) เพื่อเข้าไปอ่านรายละเอียดของคำสั่งต่างๆ ทีละคำสั่งจนกว่าจะพบคำสั่งที่สามารถเข้ากับเงื่อนไขที่ต้องการได้ ทำให้เสียเวลาในการโปรแกรมมาก



รูปที่ 4.30 แสดงการใช้ตัวช่วยเพื่อศึกษารูปแบบของคำสั่ง

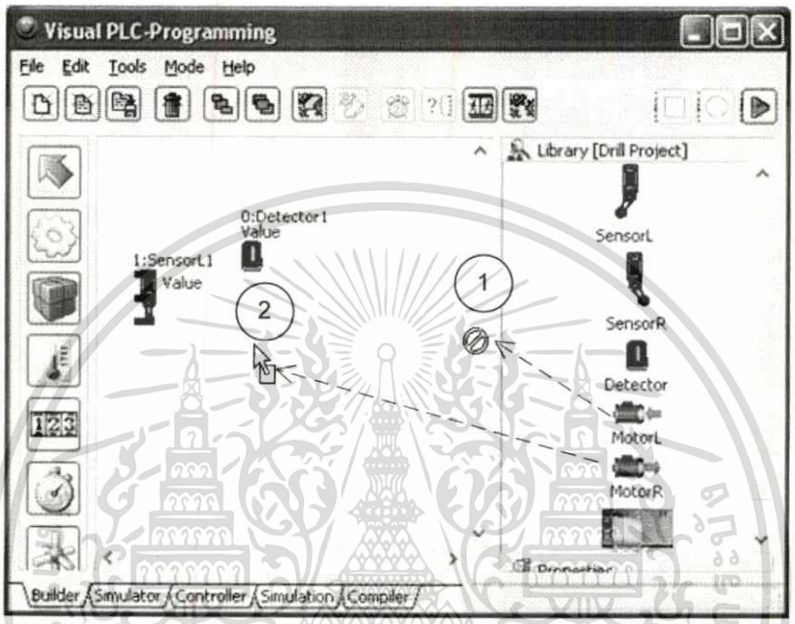
สรุปวิเคราะห์ผลการสังเกต จากการลงมือเขียนโปรแกรมตามลำดับของการปฏิบัตินั้น วิธีการโปรแกรมโดยใช้การสาธิตจะต้องมีการสร้างแบบจำลองขึ้นมาก่อนที่จะเริ่มลงมือโปรแกรมการควบคุม การเลือกใช้อุปกรณ์ในการสร้างแบบจำลองนั้นจะใช้จากแฟ้มห้องสมุดอุปกรณ์ซึ่งสัญลักษณ์ที่ใช้จะเป็นรูปภาพของอุปกรณ์จริง ช่วยให้เลือกใช้ได้อย่างถูกต้อง รวมถึงการจัดวางอุปกรณ์ซึ่งเป็นไปตามโครงสร้างของเครื่องจักรจริง ส่วนวิธีการเขียนโปรแกรมภาษาแลดเดอร์ชุดคำสั่งที่เป็นลักษณะ Command Base จากการสังเกตกลุ่มตัวอย่างพบว่ามีการเขียนคำสั่งที่ไม่เป็นไปตามหลักไวยากรณ์ของภาษาแลดเดอร์อีกทั้งผู้ใช้จะต้องจดจำชุดคำสั่งที่มีจำนวนมาก และการเลือกใช้ชุดคำสั่งให้เหมาะสมกับงานส่งผลให้ใช้เวลาในการโปรแกรมมาก นอกจากนี้การวางอุปกรณ์ตามกฎเกณฑ์ที่เครื่องมือสามารถตีความได้ และการที่พื้นที่วางอุปกรณ์นั้นเป็นลักษณะตารางทำให้มีข้อจำกัดในเรื่องของตำแหน่งการวางอุปกรณ์เป็นอย่างมาก ดังนั้นผู้เขียนโปรแกรมจำเป็นต้องมีทักษะมากกว่าวิธีการโปรแกรมโดยใช้การสาธิต

4.5.5 ขั้นที่ 5 รับรู้ผลภายหลังจากลงมือปฏิบัติ (Perceiving what happened)

การป้อนกลับผลลัพธ์ (Feedback) การได้รับรู้ผลของผู้ใช้นั้น เนื่องจากทั้งสองโปรแกรมออกแบบส่วนเชื่อมต่อผู้ใช้ (User Interface) ด้วยหลักการจับเอาสิ่งที่ผู้ใช้ต้องการและสิ่งที่เครื่องมือต้องการนำเสนอมาไว้รวมกัน (Direct Manipulation Interaction Style) จึงไม่มีความแตกต่างกันมากนัก ซึ่งมีรายละเอียดดังนี้

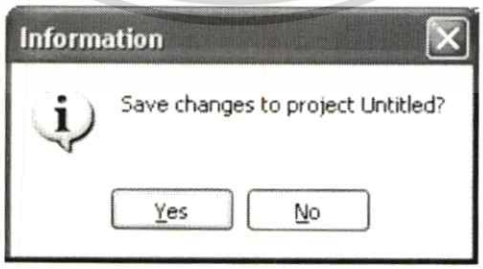
1) วิธีการโปรแกรมโดยใช้การสาธิต

1.1) วิธีการโปรแกรมโดยใช้การสาธิตจะมีการแสดงสัญลักษณ์เพื่อแจ้งให้ผู้ใช้ทราบว่าเมื่อลากอุปกรณ์จากส่วนห้องสมุดอุปกรณ์สัญลักษณ์ของตัวชี้ตำแหน่งของเมาส์ (Cursor) นั้น จะเป็นตัวบ่งบอกว่าสามารถวางอุปกรณ์ได้หรือไม่ ดังรูปที่ 4.31 สัญลักษณ์ที่ 1 หมายถึงวางไม่ได้ และสัญลักษณ์ที่ 2 หมายถึงวางได้



รูปที่ 4.31 แสดงการแจ้งเตือนถึงพื้นที่ที่สามารถวางอุปกรณ์ได้และวางไม่ได้ในวิธีการโปรแกรมโดยใช้การสาธิต

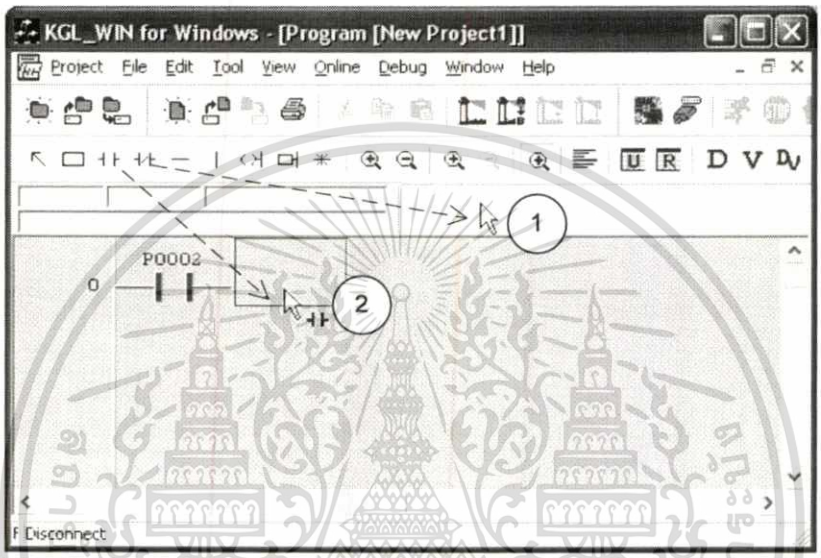
1.2) มีการแสดงข้อความเตือน เช่น ไม่ได้จัดเก็บแฟ้มข้อมูล



รูปที่ 4.32 แสดงข้อความเตือนผู้ใช้ในกรณีต่างๆ

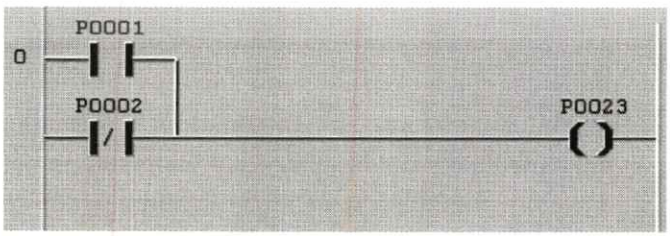
2) วิธีการเขียนโปรแกรมภาษาแลดเดอร์

2.1) การเลือกไอคอนมาวางนั้น เมื่อเลือกไอคอนในแถบเครื่องมือเสร็จแล้ว เครื่องมือจะยังคงไม่แสดงการเปลี่ยนแปลงหรือมีตัวบ่งบอกว่าเราเลือกไอคอนใด แต่จะแสดงอุปกรณ์ที่เลือกต่อเมื่อเป็นตำแหน่งที่สามารถวางอุปกรณ์ได้เท่านั้น โดยใช้สัญลักษณ์ของอุปกรณ์นั้นต่อท้ายตัวชี้ตำแหน่งของเมาส์ดังรูปที่ 4.33 สัญลักษณ์ที่ 1 หมายถึงวางไม่ได้ และสัญลักษณ์ที่ 2 หมายถึงวางได้



รูปที่ 4.33 แสดงสัญลักษณ์ต่อท้ายตัวชี้ตำแหน่งของเมาส์ในกรณีที่วางได้ และวางไม่ได้ในวิธีการเขียนโปรแกรมภาษาแลดเดอร์

2.2) หากลักษณะการวางอุปกรณ์นั้นไม่เป็นไปตามกฎเกณฑ์ที่เครื่องมือสามารถตีความได้ เครื่องมือจะมีการเตือนโดยเปลี่ยนสีเป็นสีเทาแกมเขียว



รูปที่ 4.34 แสดงการเตือนเมื่อการวางอุปกรณ์ไม่เป็นไปตามกฎเกณฑ์

2.3) มีการแสดงข้อความเตือน เช่น เมื่อใช้คำสั่งที่ไม่เป็นไปตามหลักไวยากรณ์การวางอุปกรณ์นอกตารางและไม่ได้จัดเก็บแฟ้มข้อมูล

รูปที่ 4.35 แสดงข้อความเตือนผู้ใช้เมื่อเกิดข้อผิดพลาด

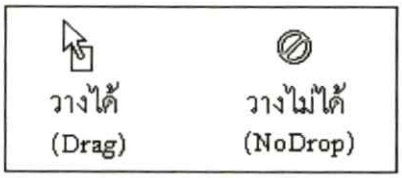
สรุปวิเคราะห์ผลการสังเกต เครื่องมือการโปรแกรมทั้งสองวิธีมีการแสดงให้ผู้ใช้ทราบถึงสถานะการอนุญาตวางตัวอุปกรณ์ และข้อความแจ้งเตือนต่างๆ ส่วนลักษณะการวางอุปกรณ์นั้นไม่เป็นไปตามกฎเกณฑ์และการใช้คำสั่งไม่เป็นไปตามหลักไวยากรณ์นั้น ในวิธีการโปรแกรมโดยใช้การสาธิตจะไม่มีการใช้ชุดคำสั่งและการกำหนดรูปแบบการวางอุปกรณ์ จึงไม่มีการเตือนในส่วนนี้ จากการสังเกตกลุ่มตัวอย่างนั้น การแจ้งเตือนช่วยให้ผู้ใช้แก้ไขการโปรแกรมให้ถูกต้องทันที

4.5.6 ขั้นที่ 6 แปลผลลัพธ์ตามความคาดหวังของผู้ใช้ (Interpreting the outcome according to the users' expectations)

การสื่อให้ผู้ใช้เข้าใจในความหมายของผลลัพธ์นั้น มีความแตกต่างกันในรายละเอียดดังนี้

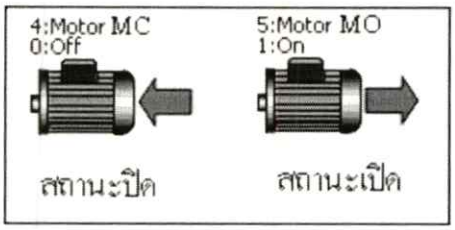
1) วิธีการโปรแกรมโดยใช้การสาธิต

1.1) การแสดงสัญลักษณ์ของตัวชี้ตำแหน่งของเมาส์เพื่อบอกถึงสถานะการอนุญาตวางตัวอุปกรณ์ โดยในกรณีที่วางได้จะใช้รูป "Drag" และวางไม่ได้จะใช้รูป "NoDrop" ซึ่งสัญลักษณ์ที่ใช้นั้นเป็นมาตรฐานของไมโครซอฟท์วินโดว์ ทำให้ผู้ใช้สามารถเข้าใจความหมายได้ง่าย



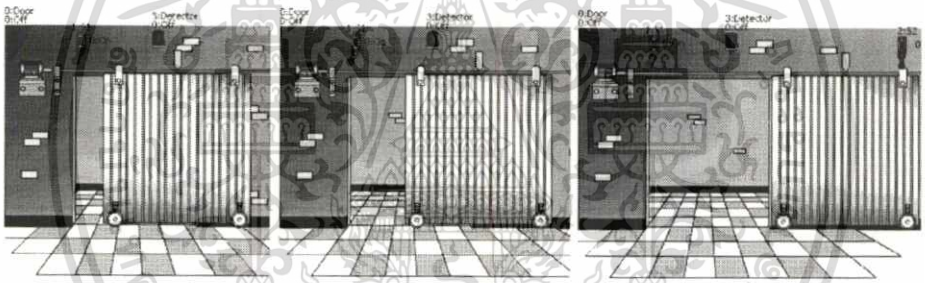
รูปที่ 4.36 แสดงสัญลักษณ์ของตัวชี้ตำแหน่งของเมาส์ในกรณีที่วางอุปกรณ์ได้และวางไม่ได้ในวิธีการโปรแกรมโดยใช้การสาธิต

1.2) การแสดงสถานะของตัวอุปกรณ์ด้วยสีเขียวและสีแดงซึ่งเป็นสีที่ใช้บอกสถานะเปิดและปิดที่เป็นสากล รวมทั้งยังบอกถึงสถานะด้วยตัวอักษร “On” หรือ “Off” เพื่อความชัดเจนยิ่งขึ้น



รูปที่ 4.37 แสดงสถานะอุปกรณ์ในวิธีการโปรแกรมโดยใช้การสาธิต

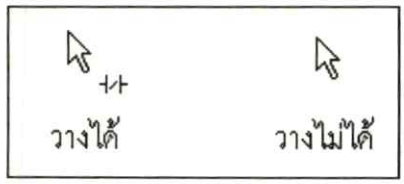
1.3) การแสดงสถานะของอุปกรณ์ด้วยภาพเคลื่อนไหว ทำให้ผู้ใช้มีความเข้าใจผลลัพธ์ที่เกิดจากการโปรแกรมการควบคุมได้มากขึ้น เช่น แสดงภาพเคลื่อนไหวของการเปิดประตู



รูปที่ 4.38 แสดงสถานะของอุปกรณ์ด้วยภาพเคลื่อนไหว

2) วิธีการเขียนโปรแกรมภาษาแลดเดอร์

2.1) การแสดงสัญลักษณ์ของตัวชี้ตำแหน่งของเมาส์เพื่อบอกถึงสถานะการอนุญาตวางตัวอุปกรณ์ โดยในกรณีที่วางได้จะใช้รูปลูกศรที่มีสัญลักษณ์ของอุปกรณ์นั้นต่อท้าย และวางไม่ได้จะใช้รูปลูกศรปกติ



รูปที่ 4.39 แสดงสัญลักษณ์ของตัวชี้ตำแหน่งของเมาส์ในกรณีที่วางอุปกรณ์ได้และวางไม่ได้ในวิธีการเขียนโปรแกรมภาษาแลดเดอร์

2.2) การแสดงผลลัพธ์ของภาษาแลดเดอร์จะแสดงสถานะเปิดหรือปิดด้วยแถบสีน้ำเงิน โดยสีในสถานะปิดกับสีในขณะที่ไม่ได้ตรวจสอบสถานะของอุปกรณ์นั้นใช้สีเดียวกัน ซึ่งอาจทำให้ผู้ใช้ตีความหมายของแถบสีน้ำเงินผิดได้ แต่ก็พอจะเทียบกับอุปกรณ์อื่นๆ ทำให้พอที่จะทราบได้ว่าอุปกรณ์ตัวนั้นอยู่ในสถานะใด

รูปที่ 4.40 แสดงสถานะอุปกรณ์ในวิธีการเขียนโปรแกรมภาษาแลดเดอร์

2.3) การใช้สีในการเตือนหากลักษณะการวางอุปกรณ์นั้นไม่เป็นไปตามกฎเกณฑ์ที่เครื่องมือสามารถตีความได้ เครื่องมือจะมีการเตือนโดยเปลี่ยนสีให้เป็นสีเทาแกมเขียว ซึ่งมาตรฐานโดยทั่วไปการผิดพลาดจะเตือนด้วยสีเหลืองหรือแดง

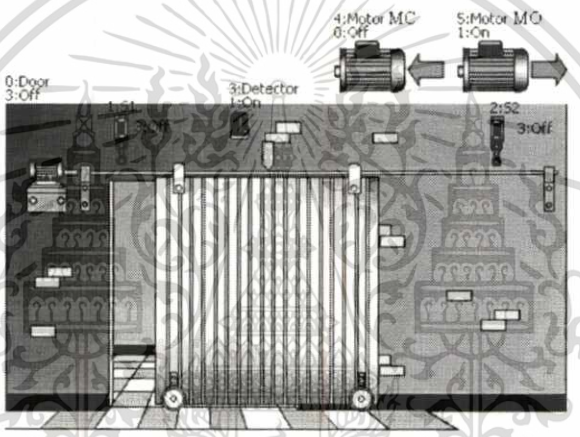
สรุปวิเคราะห์ผลการสังเกต เครื่องมือการโปรแกรมทั้งสองวิธีมีการแสดงสัญลักษณ์ของตัวชี้ตำแหน่งของเมาส์เพื่อบอกถึงสถานะการอนุญาตวางตัวอุปกรณ์ มีการแสดงสถานะปิดและเปิดของอุปกรณ์ด้วยแถบสีเหมือนกัน แต่วิธีการโปรแกรมโดยใช้การสาธิตมีการเพิ่มเติมข้อความบอกสถานะอุปกรณ์ด้วย อีกทั้งวิธีการโปรแกรมโดยใช้การสาธิตยังสามารถแสดงสถานะของอุปกรณ์ด้วยภาพเคลื่อนไหว ทำให้ผู้ใช้มีความเข้าใจผลลัพธ์ที่เกิดจากการโปรแกรมการควบคุมได้มากขึ้น ถึงแม้ว่าเครื่องมือในวิธีการเขียนโปรแกรมภาษาแลดเดอร์จะมีการใช้สีที่แตกต่างจากมาตรฐานโดยทั่วไป แต่จากการสังเกตกลุ่มตัวอย่างก็ไม่พบปัญหาในการใช้งานแต่อย่างใด

4.5.7 ขั้นที่ 7 ประเมินสิ่งที่เกิดขึ้นกับผลที่คาดหวัง (Evaluating what happened against what was intended)

ในขั้นที่ 7 นี้เป็นการประเมินผลลัพธ์เทียบกับงานและเชื่อมโยงงานต่างๆ ว่าเข้าใกล้เป้าหมายแล้วหรือยัง

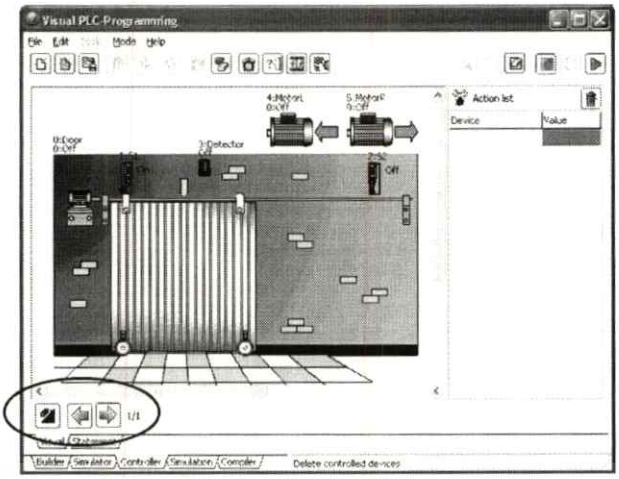
1) วิธีการโปรแกรมโดยใช้การสาธิต

1.1) การแสดงผลลัพธ์ของวิธีการโปรแกรมโดยใช้การสาธิตนั้น จะแสดงในสองลักษณะ ลักษณะแรกคือ การแสดงสถานะของตัวอุปกรณ์ด้วยสีเขียวหรือแดง และตัวอักษร "On" หรือ "Off" ลักษณะที่สองคือ การแสดงด้วยภาพเคลื่อนไหว เมื่อประเมินเทียบกับงานหรือเป้าหมาย ทำให้ผู้ใช้เข้าใจได้อย่างชัดเจนทันทีว่าทำงานตรงตามเงื่อนไขหรือไม่



รูปที่ 4.41 แสดงการเปิดประตูในวิธีการโปรแกรมโดยใช้การสาธิต

1.2) เครื่องมือสามารถตรวจสอบความครบถ้วนของการโปรแกรม ซึ่งอยู่ในส่วนของการจัดการเหตุการณ์ที่ไม่ได้ถูกโปรแกรม (Missing case) จากแบบทดสอบที่ 1 ในรูปที่ 4.42 จะเห็นได้ว่ามีอยู่ 1 เหตุการณ์ที่ยังไม่ได้ถูกโปรแกรมคือเหตุการณ์ที่ Limit Switch S1 มีสถานะเปิด และอุปกรณ์ตัวอื่นมีสถานะปิด ซึ่งในเหตุการณ์นี้ผู้ใช้จะไม่ต้องการโปรแกรมก็จะสามารถกำหนดให้ข้ามไปได้ (Ignore case)



รูปที่ 4.42 แสดงการจัดการเหตุการณ์ที่ไม่ได้ถูกโปรแกรม

2) วิธีการเขียนโปรแกรมภาษาแลดเดอร์

2.1) การแสดงผลลัพธ์ของภาษาแลดเดอร์จะแสดงเพียงสถานะเปิด หรือปิด เท่านั้น โดยแสดงด้วยแถบสีน้ำเงินที่หมายเลขช่องสัญญาณ ผู้ใช้จะต้องเทียบกลับไปเป็นอุปกรณ์จริงอีกครั้งว่าเป็นอุปกรณ์ใด รวมถึงผู้ใช้จะต้องอาศัยการคิดเปรียบเทียบและทบทวนกับ Logic Table เพื่อที่จะประเมินเทียบกับงานหรือเป้าหมาย

รูปที่ 4.43 แสดงสถานะเปิดประตูในวิธีการเขียนโปรแกรมภาษาแลดเดอร์

ตารางที่ 4.5 แสดงการเปรียบเทียบกับตาราง Logic Table สำหรับการเปิดประตู

Detector	Limit Switch S2	Motor: MO
0	0	0
0	1	0
1	**0**	**1**
1	1	0

สรุปวิเคราะห์ผลการสังเกต การประเมินผลลัพธ์เทียบกับงานและเชื่อมโยงงานต่างๆ ว่าเข้าใกล้เป้าหมายแล้วหรือยังนั้น ภาษาแลดเดอร์ซึ่งเป็นการเขียนโปรแกรมเชิงนามธรรม (Abstract) นั้น ผู้ใช้มีปัญหามากในการประเมิน เนื่องจากจะต้องตีความหมายจากสถานะของอุปกรณ์ทั้งหมด เปรียบเทียบและทบทวนกับ Logic Table เพื่อที่จะประเมินเทียบกับงานว่าเข้าใกล้เป้าหมายหรือยัง จากการสังเกตการทดลองกับกลุ่มตัวอย่างพบว่า ผู้ที่ร่วมทำการทดสอบใช้เวลานานมากในการวิเคราะห์ว่างานทั้งหมดเข้าใกล้เป้าหมายหรือยัง และบางกลุ่มตัวอย่างกลับไปแก้ไขโปรแกรมทั้งๆ ที่ผลลัพธ์ถูกต้องแล้ว ในส่วนของวิธีการโปรแกรมโดยใช้การสาธิตนั้น ผู้ใช้สามารถประเมินและตีความหมายได้โดยตรง อีกทั้งตัวเครื่องมือยังสามารถตรวจสอบความครบถ้วนของการโปรแกรม ซึ่งอยู่ในส่วนของการจัดการเหตุการณ์ที่ไม่ได้ถูกโปรแกรมทำให้ลดข้อผิดพลาดในการโปรแกรมที่ไม่ครบถ้วนลงได้

บทที่ 5

สรุปผลการวิจัย และข้อเสนอแนะ

ความต้องการในการใช้เทคโนโลยีของอุปกรณ์พีแอลซีมาใช้ควบคุมกระบวนการทำงานของเครื่องจักรและอุปกรณ์ในโรงงานยังคงมีการเพิ่มขึ้นอย่างรวดเร็ว ดังจะเห็นได้จากการขยายตัวของตลาดด้านอุปกรณ์การควบคุมอัตโนมัติที่มีการเพิ่มขึ้นจากบริษัทผู้ผลิตรายใหม่ๆ รวมไปถึงการแข่งขันในด้านการผลิตของโรงงานอุตสาหกรรมขนาดเล็กของประเทศไทยมีสูงขึ้น การลดต้นทุนการผลิต การสร้างความมั่นใจให้กับลูกค้าจึงเป็นประเด็นสำคัญที่จะทำให้โรงงานอุตสาหกรรมยืนหยัดอยู่ได้อย่างมั่นคงในตลาดต่อไป

ปัญหาประการสำคัญของการนำเอาอุปกรณ์พีแอลซีมาใช้ทดแทนอุปกรณ์รีเลย์คือ ผู้ที่ปฏิบัติงานที่มีความรู้ความเข้าใจในการควบคุมกระบวนการทำงานของเครื่องจักร และอุปกรณ์ในโรงงานนั้น มักจะไม่มีทักษะทางด้านการโปรแกรมซึ่งเป็นข้อจำกัดที่สำคัญ โดยผู้ที่จะโปรแกรมอุปกรณ์พีแอลซีนั้น จะต้องมีทักษะและความชำนาญสูงในการใช้เครื่องมือเพื่อเขียนชุดคำสั่งเพื่อควบคุมการทำงานเครื่องจักร ซึ่งในระดับของ End-User นั้นเป็นเรื่องยากที่จะเรียนรู้และทำความเข้าใจ

ในวิทยานิพนธ์ฉบับนี้ได้นำเสนอวิธีการโปรแกรมโดยใช้การสาธิตตัวควบคุมและแบบจำลองร่วมกันสำหรับอุปกรณ์พีแอลซี เทคนิคนี้เป็นการพัฒนาและผสมผสานกันระหว่างเทคนิคการเขียนโปรแกรมที่เน้นให้มีการใช้ภาพประกอบกับเทคนิควิธีการโปรแกรมโดยใช้การสาธิตโดยปรับเปลี่ยนการสาธิตจากระบบจริงมาเป็นการสาธิตกับแบบจำลองแทน ซึ่งขั้นตอนการสร้างแบบจำลองนั้นจะใช้เทคนิคการเขียนโปรแกรมที่เน้นให้มีการใช้ภาพประกอบ เมื่อสร้างแบบจำลองเสร็จเรียบร้อยแล้วจะนำแบบจำลองนั้นมาโปรแกรมโดยใช้เทคนิควิธีการโปรแกรมโดยใช้การสาธิต จุดสำคัญคือการจำลองเครื่องจักรขึ้นมาแล้วให้ผู้ใช้ทำการควบคุมเครื่องจักรด้วยมือ (Manual Control) จากนั้นระบบจะทำการเรียนรู้วิธีการควบคุมดังกล่าว เพื่อมาสร้างเป็นชุดคำสั่งและสามารถทำงานได้เหมือนการควบคุมโดยมนุษย์ (Human Control)

ทำการพัฒนาโปรแกรมในลักษณะต้นแบบ (Prototype) เพื่อใช้ในการทดลอง ซึ่งผลลัพธ์ของการโปรแกรมแปลงสู่ภาษา Structure Text และภาษา Mnemonic จากผลการทดลองพบว่าวิธีการนำเสนอนี้ใช้เวลาในการโปรแกรมลดลง และข้อผิดพลาดจากการโปรแกรมลดลง

ข้อเสนอแนะในการพัฒนาคือ ปรับปรุงการจำลองแบบให้มีประสิทธิภาพเพิ่มมากขึ้น ลดเหตุการณ์ที่ไม่ได้ถูกโปรแกรมให้มากขึ้น และปรับปรุงการใช้ชุดคำสั่งสำหรับอุปกรณ์พีแอลซีให้กระชับและมีประสิทธิภาพมากขึ้น (Optimize code)

บรรณานุกรม

[1] Gordon W. Paynter and Ian H. Witten, **"Applying machine learning to programming by demonstration"**, University of California, Riverside and University of Waikato New Zealand, 2004.

[2] Haiying She and Axel Graeser, **"Closed Loop Control and Automatic Set Point Generation in Programming by Demonstration for Service Robotic Tasks"**, University of Bremen, Germany, 2004.

[3] Richard M. Voyles, **"Toward Gesture-Based Programming: Agent-Based Haptic Skill Acquisition and Interpretation"**, University of Minnesota, 1997.

[4] S. Munch, J. Kreuziger, M. Kaiser, R. Dillmann , **"Robot Programming by Demonstration (RPD) Using Machine Learning and User Interaction Methods for the Development of Easy and Comfortable Robot Programming System"**, University of Karlsruhe, Germany, 1994.

[5] Tessa Lau, **"Programming by Demonstration: a Machine Learning Approach"**, University of Washington, 2001.

[6] Lawrence D.Bergman, Tessa A. Lau, Vittrio Castelli, Daniel Oblinger, **"Programming-by-Demonstration for Behavior-based User Interface Customization"**, IBM T.J. Watson research Center, USA, 2004.

[7] Henry Lieberman, **"Your Wish is My Command: Giving Users the Power to Instruct their Software"**, Massachusetts Institute of Technology, 2000.

[8] Richard G. McDaniel, **"Creating Complete User Interfaces by Demonstration"**, Carnegie Mellon University, 1993.

[9] Ben Shneiderman, **"Foreword to Your Wish is My Command"**, University of Maryland, 2000.

[10] Karl-Heinz john and Michael Tiegelkamp, **"ICE 61131-3 Programming Industrial Automation Systems"**, Springer, january 2001.

[11] Cypher, Allen, Introduction, In Cypher, Allen, ed., **"Watch What I Do: Programming by Demonstration"**, Cambridge: The MIT Press, 1993.

[12] Richard M. Voyles and Pradeep K. Khosla, **"A Multi-Agent system for programming robots by human demonstration"**, University of Minnesota and Carnegie Mellon University, 1998.

[13] Gordon W. Paynter and Ian H. Witten, **"Developing a Programming by Demonstration Tool"**, Department of Computer Science The University of Waikato,New Zealand, 2000.

[14] Jacopo Aleotti, Stefano Caselli, Giuliano Maccherozzi, **"Trajectory Reconstruction with NURBS Curves for Robot Programming by Demonstration"**, RIMLab - Robotics and Intelligent Machines Laboratory Dipartimento di Ingegneria dell'Informazione, University of Parma, Italy, 2005.

[15] Hyung Seok Kimy, Dong Sung Kimy, Naehyuck Changz, Wook Hyun Kwony,"A Translation Method of Ladder Diagram on PLC with Application to a Manufacturing Process", School of Electrical Eng, Seoul National University, Seoul, Korea, 1999.

[16] Luis Simões, José A. B. Gerald, **"A Communication System for Power Lines"**, Dept. of Electrical and Computer Engineering Instituto Superior Técnico, Lisbon, Portugal, 2005.

[17] Donald A. Norman, "Human action cycle", Wikipedia, New York, 1988

[18] ชัยวุฒิ จันมา, "การเขียนโปรแกรมภาษา Visual Basic 6.0", Siam computer

[19] ชำนาญ เฉลิมยุทธ,"โปรแกรมเมเบิลลอจิกคอนโทรลเลอร์",www.9engineer.com

[20] Hugh Jack,"Automating Manufacturing Systems with PLCs", Grand Valley State University,Michigan,2004

ภาคผนวก

ภาคผนวก ก.
ผลงานวิจัยที่ได้รับการตีพิมพ์เผยแพร่

1. Phissanu Changsakol, Nopporn Chotikakamthorn, Olarn Wongwirat, "**A Joint Controller-Simulator Programming by Demonstration Method for PLC Devices**", 2nd IEEE International Conference on Cybernetics & Intelligent Systems (CIS) Robotics, Automation & Mechatronics (RAM), Bangkok, Thailand, June 7-9, 2006
2. Phissanu Changsakol, Nopporn Chotikakamthorn, "**An Iterative Approach to Simulator-Controller Programming by Demonstration of PLC Devices**", 2006 ECTI International Conference (ECTI-CON2006), Ubon Ratchathani Province, Thailand, May 10-13, 2006

2nd IEEE International Conference on

Cybernetics & Intelligent Systems (CIS) Robotics, Automation & Mechatronics (RAM)

7—9 June, 2006
The Twin Towers Hotel
Bangkok, Thailand

Final Program

http://www.ntu.edu.sg/cis-ram/

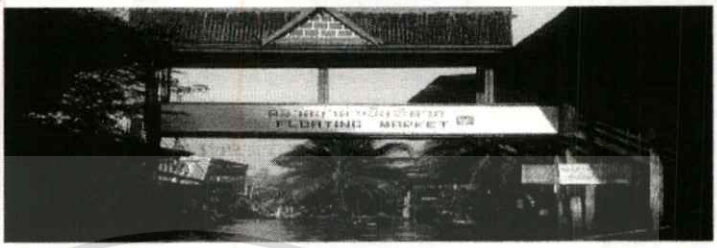
Organizers
IEEE SMC Society Singapore Chapter
IEEE R&A Society Singapore Chapter

Technical Sponsor
IEEE Thailand Section

Final Call For Papers
Second IEEE International Conference On Cybernetics and Intelligent Systems (IEEE CIS 2006)

7 - 9 JUNE 2006, Bangkok, Thailand

[WWW.NTU.EDU.SG/CIS-RAM/INDEX.HTM]



Objectives: The goal of the CIS 2006 is to bring together experts from the field of cybernetics and intelligent systems to discuss on the state-of-the-art and to present new research findings and perspectives of future developments with respect to the conference themes. The CIS 2006 is organized by the IEEE SMC Singapore Chapter, and is held together with the IEEE International Conference on Robotics, Automation and Mechatronics (RAM 2006). The conference welcomes paper submissions from researchers, practitioners, and students worldwide in but not limited to the following areas:

Computational Intelligence, Soft Computing, Fuzzy Systems, Neuro-Fuzzy Systems, Neural Networks (NN), Genetic Algorithm (GA), Evolutionary Computation (EC), Hybrid CI Algorithms, DNA Computing, Evolutionary Logistics, Evolutionary Systems, Adaptive Computing Systems, Data Mining and Management, Decision Support Systems, Informatics, Environmental Systems, Expert and Knowledge Base Systems, Human/Computer Interaction, Human/Machine Systems, Image Processing, Computer Vision, Information Assurance and Security, Intelligent Communications, Intelligent Systems, Intelligent Transportation Systems, Internet/Electronic Commerce, Knowledge Acquisition and Engineering, Manufacturing Systems, Optimization, Pattern Recognition, Quality/Reliability & Systems Engineering, Service Systems and Organizations, Socio-Technical Systems Design, etc.

Invited Sessions: The conferences will feature invited sessions on specialized topics of interests. The invited sessions are intended to usher in in-depth discussions in special areas relevant to the conference themes. The session organizers will coordinate the associated review process. All accepted papers from the invited sessions will be included in the conference proceedings. For instructions on organizing an invited session, please go to http://www.ntu.edu.sg/cis-ram/callsessions.htm

Paper Submission: Deadline for initial submission is *15 January 2006* and final submission of camera-ready manuscript is *1 April 2006*. Papers must be written in English and should describe original research work. The length of initial paper submission is limited to 6 pages, and up to 8 pages maximum, including figures, tables and references. The paper must be in US Letter size, two column format, single spacing and in Times-Roman font of size 10. Paper should be submitted on-line in IEEE Xplore compliant PDF file format. For detailed instructions on paper submissions, please go to http://www.ntu.edu.sg/cis-ram/callpapers.htm. Upon acceptance, the authors are required to register and present their papers in the conference. Papers will only be published in the conference proceedings if at least one of the authors is officially registered. The conference proceedings will be included in the EI Compendex Database.

Best Paper Award: Selection of the best paper will be made at the Conference based on both the technical content and presentation. The winner will be chosen by an Awards Committee in consultation with the International Advisory Committee.

Important Dates:

Invited Session Summary Submission	: 1 January 2006
Conference / Invited Session Paper Submission	: 15 January 2006
Notification of Acceptance	: 1 March 2006
Final Camera-Ready Manuscript	: 1 April 2006
Advance Registration Deadline	: 1 April 2006

Cybernetics & Intelligent Systems

Wed 7 June 10:15–12:15

W1A Bumrang

Informatics
Chair(s): Ali Siadat
Prasong Praneetpolgrang

11:15 – 11:35

The Performance Assessment on Universities' Informatics using Balanced Scorecard

*Prasong Praneetpolgrang, *Ubonsin Poprom, *Preang Kitratporn,* Sripatum University , *Phranakhon Rajabhat University, THAILAND

This research was conducted with three objectives; 1)evaluated the effectiveness of Informatics management or ICT management of public universities in Thailand which based on Balanced Scorecard(BSC). 2)created Key Performance Indicators (KPIs) of informatics developing, and 3)developed a new informatics strategic model (or ICT strategic model) for informatics management. This was to confirm that informatics could give an extreme benefit to universities and stakeholders. This proposed model provided some guidelines to improve investment and worthwhile informatics governance. In order to reach these objectives, qualitative and quantitative research methodologies were approached. The researchers investigated on 40 public universities in Thailand as a case study. The findings of this ...

11:35 – 11:55

A Framework for using a Case Based Reasoning System Applied to Cost Estimation

Alexandre Thibault, Ali Siadat, Patrick Martin, Ecole Nationale Superieure d'Arts et Metiers, FRANCE

In the context of the cost estimation, the use of a case based reasoning system is an interesting solution. However the creation of case corresponding to a mechanical part to estimate a more general problem resolution doesn't require the opinion of only one expert but a consensus of several experts and for extended enterprises it may be difficult since the experts aren't necessary at the same place. Traditionally the filling of a case base is done with several interviews of experts. We thus defined a case creation method by obtaining consensus of a group of experts, as well as the procedures which would make it possible to associate it with a case based reasoning system.

11:55 – 12:15

Management and Selection of Visual Metaphors for Courseware Development in Web Based Learning

*Ramadoss Balakrishnan, *S. Balasundaram Ramakrishnan,* National Institute of Technology, *National Institute of Technology, INDIA

Metaphors are widely seen in the user interfaces of todays web based applications. Some of the major features of metaphors are their ability to organize information related to any concept and to provide better understanding of the information. The traditional teaching and learning activities of educational systems have enjoyed the benefits of metaphors in several ways. This includes representing metaphors in various forms such as text, graphics, animated images etc. One of the challenges for Web Based Learning (WBL) courseware developer is to design the appropriate metaphors relevant to the context of information presented in the courseware. This paper focuses on the method of selecting one or more metaphors constructed already by the courseware developers that will help in understanding as well ...

W1B Rong

Vehicle Systems Control (*RAM*)
Chair(s): Chidchanok Lursinsap
Dikai Liu

Research on Fault Diagnosis and Protect Model for DC Traction Power Supply System based on Traveling Wave Theory

*X Li, S Guo, Y Zhou, *Y Li, **S Fan,* Beijing Inst of Petrochemical Tech., *Beijing Univ. of Aeron. & Astron. **Beihang Univ., CHINA

The paper analyses the limitation of the differentiation (Delta I protection method. It puts forward traveling wave protection method for the DC power supply system that is based on interrelated characteristics of the traveling wave and the unit impedance of the transmission line. The traveling wave protection method overcomes the disadvantage of the traditional method in which the protection action is affected by the overall impedance of the circuit, and thus improves the reliability and sensibility of the relay protection system. A great number of field tests are carried out and the research results are satisfactory.

Design of High Quality Controller to be Advantage over PI with Low-Pass Controllers

Zhibing Shu, Caizhong Yan, Nanjing University of Technology, CHINA

In this paper, an advanced PI control method was analyzed. The analysis of Bode plot in frequency domain and the experiment show that advanced controller is better than the existing PI controllers. They can achieve higher bandwidth, lower setting time and better disturbance rejection ability. The increased performance costs little in sensor noise amplification. We show using true-life design examples that advanced control algorithms improve equally well both speed and position controllers.

W1C Charu

Human/Computer Interaction
Chair(s): Wei Tech Ang
Kun-Chieh Wang

A Study of Emotional Motion Description by Motion Modification and Adjectival Expressions

Atsushi Yamaguchi, Yoshikazu Yano, Shinji Doki, Shigeru Okuma, Nagoya University, JAPAN

Emotion-expression through motion is meaningful communication process between humans and robots. Automatic emotion expression generation is desired to robots which communicate with humans. Proposed technique generate the emotional motion by modifying the base motion pattern with the combination of adjectival expressions. In order to generate emotional motion, emotion modification rule must be prepared which represents intensity and corresponding the relationship between emotion expression and the adjectival expressions. In this paper, emotion modification rule is obtained by organoleptic tests and generates the suitable emotional motion pattern from base motion pattern.

Adaptive, Language Independent Spell Checking using Intelligent Traverse on a Tree

*Behrang Qasemizadeh, *Ali Ilkhani, Amir Ganjeh,* Iran University of Science and Technology, IRAN, *Digital Clone Corp., IRAN

This paper introduces an adaptive, language independent, and 'built-in error pattern free' spell checker. Proposed system suggests proper form of misspelled words using nondeterministic traverse of 'Ternary Search Tree' data structure. In other words the problem of spell checking is addressed by traverse a tree with variable weighted edges. The proposed system uses interaction with user to learn error pattern of media. In this way, system improves its suggestions as time goes by.

A Joint Controller-Simulator Programming by Demonstration Method for PLC Devices

*Plassanu Changsakol, *Napporn Chotikakamthorn, Olarn Wongwirat,* King Mongkut's Institute of Technology, Ladkrabang, THAILAND

In this paper, a problem of visual programming for PLC (programmable logic controller) devices is described. A programming by demonstration method is developed so that a problem-domain expert can program such devices without programming knowledge. The method is novel in the sense that, unlike other programming-by-demonstration methods, it requires little a priori knowledge of the control system simulation model, nor it needs to interact with a real system. A simulation model, which captures the laws of nature that govern the system behavior, is jointly programmed with a control device (PLC), under the joint controller-simulator programming by demonstration paradigm. A PLC programming tool developed based on the proposed method can generate a high-level ST (Structure Text) program code, from ...

A Joint Controller-Simulator Programming by Demonstration Method for PLC Devices

Phissanu Changsakol, Nopporn Chotikakamthorn, Olarn Wongwirat
King Mongkut's Institute of Technology Ladkrabang
Faculty of Information Technology and Research Center for Communication and Information Technology
Bangkok, Thailand
Phissanu@hotmail.com

Abstract— **In this paper, a problem of visual programming of PLC (programmable logic controller) devices is described. A programming by demonstration method is developed so that a problem-domain expert can program such devices without computer programming skill. The method is novel in the sense that, unlike other programming-by-demonstration methods, it requires little a priori knowledge of the controlled system model, nor it needs to interact with a real system. A simulation model, which captures the laws of nature governing the system behavior, is jointly programmed with a control device (PLC). A PLC programming tool, developed based on the proposed joint controller-simulator programming by demonstration paradigm, can generate a high-level structure text program code, from the demonstrated actions as performed by a user.**

Keywords—**Programming by Demonstration, Programmable Logic Controller, Programming Paradigm, End-User Programming**

I. INTRODUCTION

Programmable Logic Controller (PLC) is a digitally operating electronic apparatus which uses a programmable memory for the internal storage of instructions. It includes functions such as logic sequencing, timing, counting, and arithmetic operations, to control various types of machines or processes. PLCs programs are generally written by a specialized programming language as specified by the IEC 1131-3 standard. It is thus not possible for an end user to program the devices. Programming by demonstration (PBD) concerns with providing a non-programmer tools and techniques to control, automate, and customize the software applications or intelligent physical devices. PBD technique is simpler than procedural programming because there is no need to learn a language syntax and rules. This concept has been applied for various purposes such as those listed below.

- Automation of repetitive tasks using software functions. For example, a macro recorder learns patterns of user's activity and can then execute those patterns autonomously. [1, 5]
- User interface customization and personalization. Here, PBD is applied to change interface appearance to optimize access to frequently used functions. [6]
- PBD is also applied to robotic manipulation. In [2, 4], a PBD technique is proposed so that a robot arm movement can be programmed through demonstration.

One of the most important limitations of those existing programming-by-demonstration methods, when applied to a problem of PLC programming, is the need for an existence of a real system to accompany a demonstration. Alternatively, without a real system, a complete simulation model of the system must be available so that a user can perform a demonstration on a system simulator instead. In this paper, a programming-by-demonstration method is developed to avoid such limitation. Without the need for a priori knowledge on the system simulation model, the method shares a similar generality property of standard programming languages.

The paper is divided into two parts. First, PLC programming paradigms are described. Next, the proposed joint controller-simulator programming by demonstration method is detailed in Sections 3 and 4. Use of the PLC programming tool, developed based on the proposed programming paradigm, is described using a simple water tank system as a case study. Finally, concluding remarks are given.

II. PROGRAMMING PARADIGMS FOR PLC DEVICES

Programming paradigm is a paradigmatic style of programming. It provides and determines the view that the programmer has of the execution of the program. For example, in object-oriented programming, programmers can think of a program as a collection of interacting objects, while in functional programming it can be thought of as a sequence of stateless function evaluations. The common PLC programming paradigms are described below.

1) Conventional programming paradigm: This is a programming paradigm based upon the concept of procedures, also known as routines, subroutines, methods, or functions, that contain a series of computational steps. Examples of PLC languages under this conventional programming paradigm category are mnemonic and ST (Structure text) language.

1-4244-0023-6/06/$20.00 ©2006 IEEE

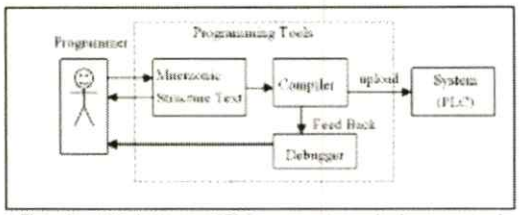


Figure 1. A block diagram of PLC programming under the conventional programming paradigm

2) Visual programming paradigm. This is a programming language that uses a visual representation such as graphics, drawing, animation or icons, partially or completely. It views program construction as a visual task similar to the solving of jigsaw puzzles. Example of the language that uses this programming paradigm includes the ladder diagram, the function block, and the sequential function chart.

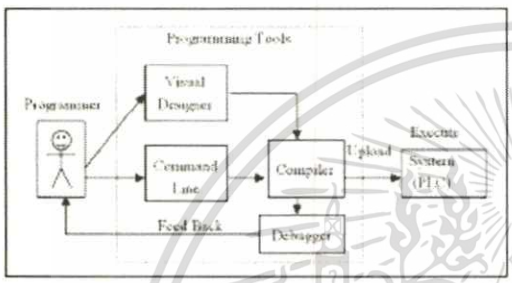


Figure 2. A block diagram of PLC programming under the visual programming paradigm

Programming by Demonstrations (PBD) is the most recent approach to programming and is largely used at present for end-user programming. The essential notion is that the user demonstrates a program to the system by working through one or more examples. The system then generalizes the user's sequence of actions to produce a general program. One of the best known such systems is Cypher's EAGER system which can automatically detect repetitions in user's actions and build a program to perform the use task. [11]

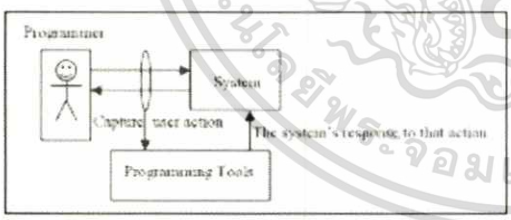


Figure 3. A conceptual view of the programming by demonstration paradigm

The programming by demonstration paradigm is generally applicable only when the full functional system (to be programmed) is available. Therefore, applicability of PBD to program a new system that has not yet existed is questionable. Similarly, applying PBD concept to PLC programming requires the existence of a controlled system to be interacted with during the demonstration.

In this paper, the above problem is solved by a joint controller-simulator programming by demonstration method. This technique combines visual programming and PBD for jointly programming of controller devices and a simulation model of the controlled devices. A joint controller-simulator programming by demonstration method is described in the next section.

III. A JOINT CONTROLLER-SIMULATOR PROGRAMMING BY DEMONSTRATION METHOD FOR PLC DEVICES

A joint controller-simulator programming by demonstration method is composed of four parts as follows (see Figure 4):

- Model builder: This module is responsible for building relationships among controller devices and controlled system.
- System simulator: This module provides an animated display of the controlled system dynamics in response to a controller action.
- User action recorder: This module records a user action for the purpose of both programming the controlled system simulation model and the controller.
- Compiler: This module is used to transform a recorded user action into an actual PLC programming code. The code can be in a high level language such as a structure text, or a low-level one such as a mnemonic code.

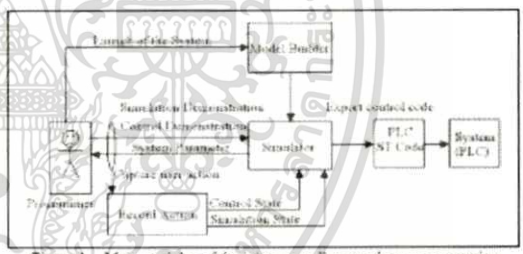


Figure 4. Main modules of the joint controller-simulator programming method

Conceptually, the system as shown in Figure 4 operates in one of five modes (states) as explained below (see Figure 5).

1) System initialization mode. In this mode, a controlled system is created by a user through visual interface. Each controlled device and the corresponding properties can be defined. In addition, through visual programming, general

relationship among controlled devices, and between controlled devices and the controller devices can be declared (visually).

2) Simulator programming mode: Under this mode, a system simulation model, which captures the relationship among controlling and controlled devices, can be programmed through user demonstration. A user programs the model by demonstrating how controlled device's properties change in response to a controller action, as well as to changes in other controlled devices' properties.

3) Controller programming mode: In this mode, a user is allowed to define a controller action in response to a particular system state. As for the case of the simulator programming mode, a controller is programmed through user demonstration of actions.

4) Simulation mode: In this mode, a programming tool demonstrates to a user how a system state changes in response to a controller action and vice versa. The simulation is provided based on a system simulation model, constructed from previous user actions under Mode 2. Therefore, the model may be incomplete and the simulation may be inaccurate. System model and simulation accuracy can be corrected, however, through iteratively re-programming.

5) Compiler mode: When in this mode, the tool transforms user actions into an actual PLC programming code. For example, the tool developed in this study chooses ST (Structure Text) as specified by the IEC 1131-3 standard as a targeted PLC programming language.

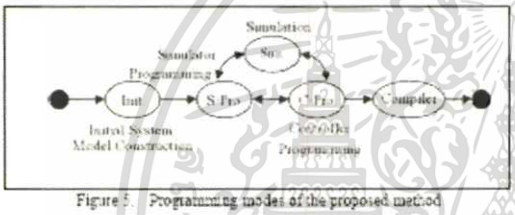


Figure 5. Programming modes of the proposed method

IV. PBD-BASED PLC PROGRAMMING TOOL

In this section, design of the tool developed based on the proposed method as described in Section 3 is described below.

1) Model builder: This module is active when the tool is in the system initialization mode. It is used for system model construction. Using a visual programming paradigm, a controlled and/or controller device is defined by the selection of icons from the tool palette. Each device's properties are defined through the object inspector (see Figure6).

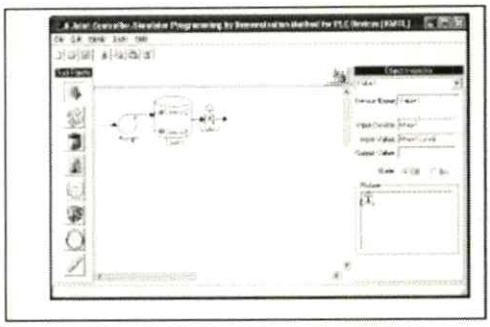


Figure 6. Screen shot of the tool's model builder

TABLE I STANDARD OBJECT CLASSES SUPPORTED BY THE TOOL

Device	*Description*
1. Activator	A generic controller's output device
2.Sensor	A generic controller's input device
3.Container	A generic controlled device
4.Counter	A specialized counter device
5.Timer	A specialized timer device
6.Generic	A general purpose device class for extension
7. Pipeline	A connector component

TABLE I summarizes standard device classes as supported by the developed tool.

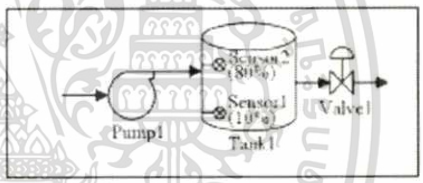


Figure 7. A simple water tank controlling system

To illustrate how the tool is used for PLC programming, a simple water tank controlling system shown in Figure 7 is used as a case study. To create a simulator model, relationship between controlling device and controlled device as well as the relation between controlled device and controlled device need to be defined. Through the language of visual programming, two types of relationship can be defined.

a) Association relationship: Once defined, allows the associated objects to exchange message (parameters, commands, etc.) with each other. For examples, in our case study, when Valve1 is connected with Tank1 through the connector device, opening or closing of Valve1 can have an

effect on the properties (such as a water level) of Tank1. The direction of the connector's arrow head is used to identify the direction of message flow.

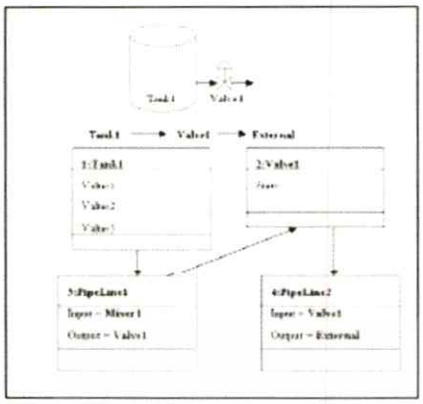


Figure 8. Defining the association relationship

b) Dependency relationship: Is applied to define a part-of relationship between two or more controlled devices, or between a controlled device and one or more controller devices. For example, in our case study, to install a controller sensor to measure a tank's water level, the sensor is placed inside the container Tank1 to establish the required relationship. In Figure 9, two such sensors, L1 and L2, are defined with this type of relationship with Tank1, so that two (low and high) water levels can be sensed.

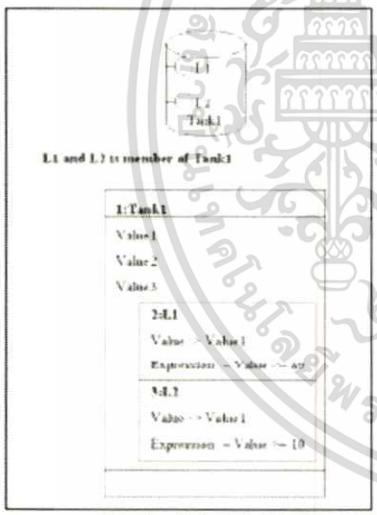


Figure 9. Defining the dependency relationship

2) User action recorder: This module is responsible to recording a user action. It operates in one of the two modes, the simulator and controller programming modes. The two modes, along with the simulation mode, can be inter-changed. Generally, the simulator programming mode is activated when a user changes the state of a controlling device of which the corresponding system response is not known yet. Once a system state change in response to the controller action is defined during the simulation programming mode, the tool should be switched to the simulation mode. From then, when a particular system state occurs, a user can switch to the controller programming mode so that an appropriate controller action can be defined.

To illustrate how to program a system simulation mode, consider our case when Valve1 or Pump1 is turned on/off (Figure 10).

a) In response to the controller state change when the status of Pump1 is turned to "ON", a user switches to the mode. Based on the association relationship between Pump1 and Tank1 as established through visual programming during system model initialization, the tool prompts for a change to any of Tank1's properties. A user can then program the simulation model by raising the tank's water level. The tool records the action, with a dialogue for the user to define the rate of water level increase.

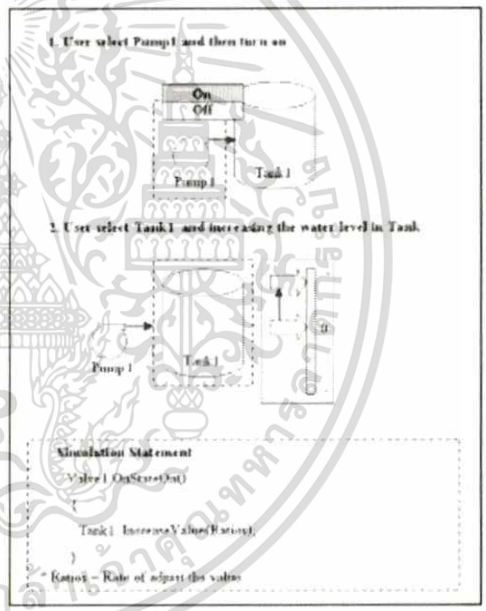


Figure 10. Using the tool to program the system model when Pump1 is turned on

b) Consider then the case where Valve1 is turned on. When switched to the simulator programming mode, Tank1 is highlighted as a result of the established relationship between Valve1 and Tank1.To drain the water level in the tank, a user select the tank's property that is affected by this controller action and then lowering the water level. The tool then records the action (see Figure 11).

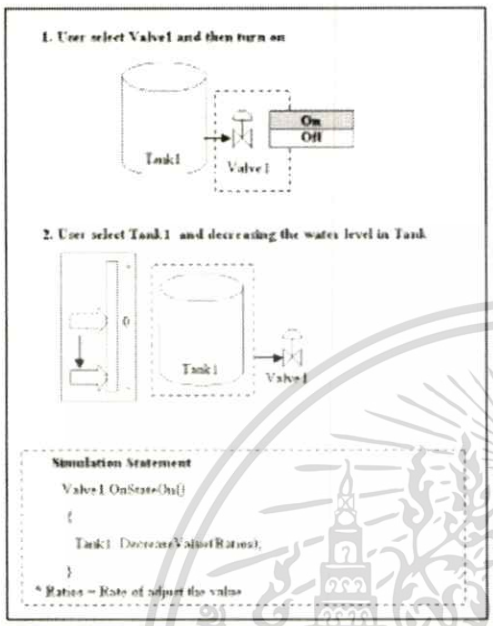


Figure 11. Using the tool to program the system mode, when Valve1 is turned on.

3) Controller Programming: To illustrate how to program a controller through demonstration, consider the following scenarios.

a) While in a simulation mode, a user notices that a water level is increasing. Once the level arrives at the point specified by Sensor 1, the simulator then prompts a user for a controller action by switching to a controller programming mode. The user informs the tool to continue the simulation. When the level arrives at the point specified by Sensor 2, the simulator prompts for the action again. This time, however, the user defines the action by visually turning Pump1 off. The tool then records that action (and if no other action, the tool switches to the simulator programming mode).

b) While in a simulation mode, the water level is decreasing. When it passes the level below that specified by Sensor1, a user can proactively turns the tool into a controller programming mode himself. He then can program the controller by visually turning Pump1 on (see Figure 12).

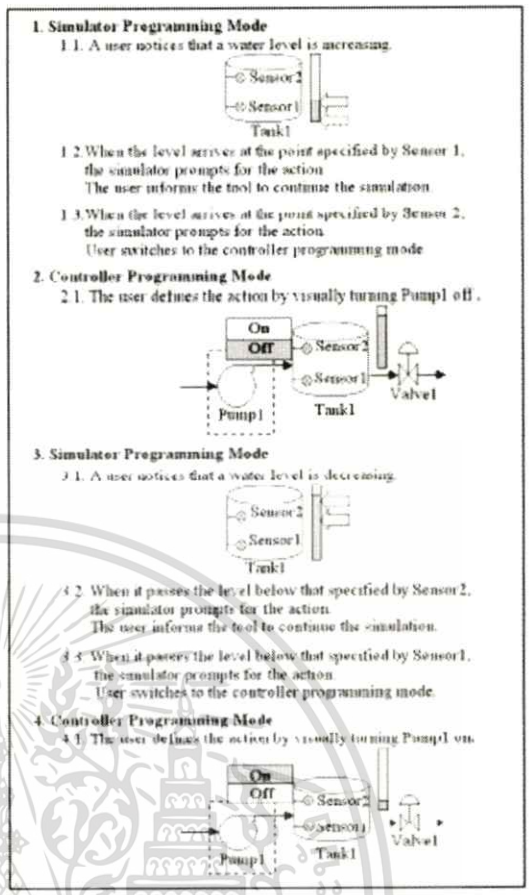


Figure 12. Programming the controller during the controller programming mode.

Figure 13 describes the sequence of user actions to program both the simulator and the controller. From the figure, note the iterative nature of the joint-programming process.

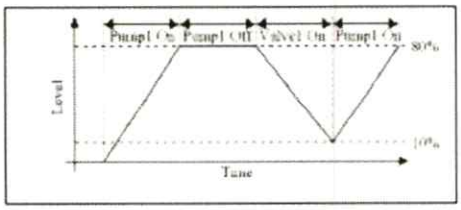


Figure 13. A sequence of user actions during the joint programming-by-demonstration process

From the changes in controller devices' states as described, those involved devices' states can be summarized as shown in TABLE II.

TABLE II. CONTROLLER AND CONTROLLED DEVICES' STATE CHANGES

No	*Sensor1*	*Sensor2*	*Pump1*
1	0	0	1
2	1	0	X
3	1	1	0
4	1	0	X
5	0	0	1

0 = State Off, 1 = State On, X = Don't Ignore

From the above state table, a truth table describing all possible logic combinations of the controller devices' states can be derived as shown in TABLE III. In this case, user action scenarios cover all possible combinations. Therefore, the truth table is simply the state table with redundant row being removed. When user action scenarios do not cover all possible combinations, the tool must ask the user for more information by displaying the missing scenarios. The topic of generating a missing scenario is discussed elsewhere, however.

TABLE III. A TRUTH TABLE AS GENERATED FROM THE STATE TABLE

Sensor1	*Sensor2*	*Pump1*
0	0	0
0	1	
1	0	X
1	1	1

0 = State Off, 1 = State On, X = Don't Ignore, - = Unknown

From the truth table (see TABLE III), an actual controller program can be generated when the tool is switched to the compiler mode. For the case of our simple water tank system, the generated code for each scenario as explained so far is summarized in TABLE IV.

TABLE IV. CODES GENERATED CORRESPONDING TO VISITED SCENARIOS AND USER DEMONSTRATIONS

Scenario	*ST generated code*
Sensor Sensor1; Sensor Sensor2;	INPUT_VAR Sensor1:BOOL; Sensor2:BOOL; END_VAR;
Activator Pump1;	OUTPUT_VAR Pump1:BOOL; END_VAR;
IF(Sensor1 State = Off & Sensor2 State = Off) Pump1 State = On;	IF Sensor1 = FALSE AND Sensor2 = FALSE THEN Pump1:=TRUE; END_IF;
IF(Sensor1 State = On & Sensor2 State = On) Pump1 State = Off;	IF Sensor1 = TRUE AND Sensor2 = TRUE THEN Pump1 :=FALSE; END_IF;

V. CONCLUDING REMARKS

In this paper, the problem of PLC programming by the programming by demonstration paradigm has been considered. A new joint controller-simulator programming-by-demonstration method has been proposed. Details of the design and implementation of the PLC programming tool based on the proposed method has been given. Use of the tool for PLC programming has been demonstrated by using a simple case of a water tank system. Based on user demonstration, the tool can generate a PLC programming code in a form of a structure text as defined by IEC 1131-3 standard.

REFERENCES

[1] Gordon W. Paynter and Ian H. Witten, "Applying machine learning to programming by demonstration", University of California, Riverside and University of Waikato New Zealand, 2004.

[2] Hanying She and Arul Gnesier, "Closed Loop Control and Automatic Set Point Generation in Programming by Demonstration for Service Robotic Tasks", University of Bremen, Germany, 2004.

[3] Richard M. Voyles, "Toward Gesture-Based Programming: Agent-Based Haptic Skill Acquisition and Interpretation", University of Minnesota, 1997.

[4] S. Munch, J. Kreuziger, M. Kaiser, R. Dillmann, "Robot Programming by Demonstration (RPD) Using Machine Learning and User Interaction Methods for the Development of Easy and Comfortable Robot Programming System", University of Karlsruhe, Germany, 1994.

[5] Tessa Lau, "Programming by Demonstration: a Machine Learning Approach", University of Washington, 2001.

[6] Lawrence D.Bergman, Tessa A. Lau, Vittorio Castelli, Daniel Oblinger, "Programming-by-Demonstration for Behavior-based User Interface Customization", IBM T.J. Watson research Center, USA, 2004.

[7] Henry Lieberman, "Your Wish is My Command: Giving Users the Power to Instruct their Software", Massachusetts Institute of Technology, 2000.

[8] Richard G. McDaniel, "Creating Complete User Interfaces by Demonstration", Carnegie Mellon University, 1993.

[9] Ben Shneiderman, "Foreword to Your Wish is My Command", University of Maryland, 2000.

[10] Karl-Heinz John and Michael Tiegelkamp, "IEC 61131-3 Programming Industrial Automation Systems", Springer, january 2001.

[11] Cypher, Allen. Introduction. In Cypher, Allen, ed., "Watch What I Do: Programming by Demonstration", Cambridge: The MIT Press, 1993.

ECTI – CON 2006

Ubon Ratchathani, Thailand, May 10-13.

Second Call For Paper

ECTI-CON 2006 is the third annual international conference organized by Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI) association of Thailand. The conference aims to be the premier international forum for presentation of technological advances and research results in the field of electrical engineering, electronics, computer, telecommunications and information technology.

Steering Committee
Panaak Sirikuchatapong, Chair
Wanlop Surakampontorn
Sawasd Tantaratana
Sawat Tantiphanwadi
Akachai Sang-in
Sarchai Kanoiphiwong
Weerapant Muvigaram
Chidchanok Lursinsap
Kobchai Dejhan
Somchai Chatratana
Somsak Choomchaum, Secretary

Advisory Committee
Akinori Nishihara, (TIT, Japan)
Hara Shinji (U. of Tokyo, Japan)
Luigi Benedicenti, (U. of Regina, Canada)
Rolf H. Jansen (Aachen U., Germany)
Yong-Hwan Lee (SNU, Korea)
Narong Yoothanom (SPU)

Organizing Committee General
Wanlop Surakampontorn (KMITL)

Vice chairs
Monai Krairiksh (KMITL)

International Coordination Chair
Saykhong Saynasine (NUOL, Laos)
Hong Chan (RUOP, Cambodia)
Nicholas Shuley (UQ, Australia)
Pung Kong (NUS, Singapore)

Technical Program Co-Chairs
Athikom Roekrabut (MUT)
Charwut Chat-uthai (KMITL)
Bundit Thipakorn (KMUTT)
Jun-ichi Takada (TIT, Japan)
Prabhas Chongstitvattana (CU)
Vutipong Areekul (KU)
Waree Kongprawechnon (SIIT)
Jitkasem Ngamwiwit (MUT)

Special session chairs
Kazushi Nakano (UEC, Japan)
Prayoot Akkaraekthalin (KMITNB)
Chawong Phongcharoenpanich (KMITL)

Local arrangement chairs
Werachet Khan-ngern (KMITL)
Surajate On-rit (UBRU)
Aprat Siritaratiwat (KKU)
Phaophak Sirisuk (MUT)
Yingrak Auttarasutak (RTU)

Publication chairs
Apirat Siritaratiwat (KKU)
Roungrong Sukaewathira (KMUTT)
Damri Torrungrueng (AUST)

Publicity chairs
Pinit Kumhom (KMUTT)
Anantawat Kunakorn (KMITL)

Exhibition chairs
Keattisak Sripimanwat (NECTEC)
Supachai Tret (UTCC)
Denchai Worasawate (KU)

Finance chairs
Banlue Srisuchinwong (SIIT)
Chalie Charoenlarpnopparut (SIIT)

General secretary
Kosin Chamnongthai (KMUTT)
Apanuit Thanachayanont (KMITL)

ecti-con2006@ecti.or.th
Contact Address
Kosin Chamnongthai
King Mongkut's University of Technology Thonburi
91 Pracha-uthit Rd., Bangmod, Tungkru, Bangkok, Thailand
Tel. (+66)2470-9064, Fax (+66)2470-9070

Keynote Speaker:
Nicholas Shuley, University of Queensland, Australia, Title: "Electromagnetic"
Saykhong Saynasine, National University of Laos, Title: "Trend of IT in Laos"

The program of ECTI-CON 2006 will consist of plenary sessions, invited sessions, special sessions, and regular technical sessions. Topics of interest for submissions include, but are not restricted to:

1. Circuit and Systems: Linear and nonlinear systems, analog circuits, filters, and data conversion, RF and mixed-signal circuits, circuits and systems for communications, low power design & VLSI physical design, high level synthesis and logic synthesis, devices and modeling, MEMS and nano electronic devices

2. Computer and Information: Computer systems and applications, internet technology and applications, computer networks, storage systems and techniques, natural language processing, artificial Intelligence and applications, system software, modeling and simulation, multimedia services and technology

3. Communication Systems: Communication theory, high speed networks, wireless/mobile communications, optical communications, modulation and coding, networks and network protocols, network management and design, microwave technology, antenna and propagation

4. Control Engineering: Control theory and applications, adaptive and learning control systems, fuzzy and neural systems, H-infinity control systems, model and system identification, robust, optimal and nonlinear control, manufacturing control systems, process control systems, control devices and instruments, and robotics

5. Electrical Power: Energy and power systems, electromagnetic compatibility, electric machinery, high voltage and insulation, system design and illumination

6. Signal Processing: Adaptive signal processing, multirate signal processing, array processing, image processing, filter design and realizations, speech processing and coding, video processing and coding, signal processing for communications

7. Other related fields: Educations, IT foresight, IT policy and law, IT/telecommunication management and others

PAPER SUBMISSION Prospective authors are invited to submit original papers, in English, of not more than four (4) pages in standard IEEE two-column format, reporting their research and development results as well as technical application and implementation in the described scopes.
Papers will be accepted only by electronic submission through the conference web site, located at http://www.ecti.or.th/~ecti-con2006/. Accepted papers are expected to be presented at the conference.
All papers are peer reviewed.
At least one author of each accepted paper MUST register for the conference for the paper to be included in the program.

Important Dates:
- Special Session and Tutorial Proposals December 31, 2005
- Notification of Special Session January 15, 2006
- Full paper submission due January 31, 2006
- Notification of Acceptance March 15, 2006
- Conference advance registration due April 8, 2006
- Camera-ready paper submission due April 8, 2006
- Conference date May 10-13, 2006

PROPOSALS FOR TUTORIALS AND SPECIAL SESSIONS Proposal for tutorials must include a title, an outline and its motivation, contact information for the presenter, and a short description of the material to be covered. The special session proposal should include a brief description of the proposed session including (1) Title, (2) Abstract, (3) Full Organizer Name, (4) Short biography of the session organizer, (5) Complete contact information with E-mail address, and (6) List of potential authors who would contribute to the session. Please submit your proposal to the conference secretariat via e-mail at ecti-con2006@ecti.or.th Selected papers will be considered for publication in the special issue of the ECTI Transactions. For further information: http://www.ecti.or.th/~ecti-con2006/

IEEE CAS Society — IEEE-LEOS THAILAND — IEEE Communications Society Thailand Chapter — Seagate — NECTEC — IEEE MTT/AP/ED THAILAND CHAPTER

An Iterative Approach to Simulator-Controller Programming by Demonstration of PLC Devices

Phissanu Changsakol, and Nopporn Chotikakamthorn

Faculty of Information Technology and Research Center for Communication and Information Technology, King Mongkut's Institute of Technology Ladkrabang, Thailand, phissanu@hotmail.com

ABSTRACT

In this paper, a problem of visual programming for PLC (programmable logic controller) devices is addressed. A method based on the programming by demonstration approach is developed so that a non-programmer can program such devices without programming training. The method requires neither priori knowledge of the control system simulation model, nor it needs a real controlled system to interact with. A simulation model, which captures the laws of nature that govern the system behavior, is jointly and iteratively programmed with a control device. A cycle of this iterative joint programming is described. A problem of missing scenarios, not known to the user during the demonstration, is addressed in this paper. Solutions to the problem are provided.

Keywords: Programming by Demonstration, Programmable Logic Controller, Programming Paradigm, End-User Programming

1. INTRODUCTION

A PLC (Programmable Logic Controller) is normally programmed using a structured programming language as specified by IEC 1131-3 standard. Like other computer programming language, to program the device, programming skill needs to be developed. This prevents a domain-expert with no programming background from defining instructions for the device. Programming by demonstration (PBD) concerns with providing a non-programmer tools and techniques to control, automate, and customize the software applications or intelligent physical devices. PBD technique is designed for a non-programmer so that he/she can use such technique for various program customizations and automations as listed below.

- Automation of repetitive tasks through a macro recorder. [1, 5]
- User interface customization and personalization. Here, PBD is applied to change interface appearance to optimize access to frequently used functions. [6]
- Robotic programming. In [2, 4], a PBD technique is applied to program a movement of a robot arm.

One important drawback of these existing programming-by-demonstration methods, when applied to a problem of PLC programming, is the need for a real system to be controlled, with which a user can perform a demonstration. Alternatively, without a real system, a complete simulation model of the system must be available. In this paper, a programming-by-demonstration method is developed to avoid such limitation. Without the need for a priori knowledge on the system simulation model, the method shares similar generality properties of standard programming languages.

The paper is organized as follows: In Section 2, PLC programming paradigms are briefly described. The proposed iterative controller-simulator programming by demonstration method is detailed in Section 3. In Section 4, use of the tool developed by the proposed method is demonstrated through a chosen case study. The situation where user's demonstrated actions are incomplete, is addressed. Concluding remarks are given in Section 5.

2. PROGRAMMING PARADIGMS FOR PLC DEVICES

In thi section, brief description of common PLC programming paradigms are described as follows.

1) Conventional programming paradigm: This is a programming paradigm based upon the concept of procedures. Examples of PLC languages under this conventional programming paradigm category are mnemonic and ST (Structure text) language.

2) Visual programming paradigm: This is a programming language that uses a visual representation such as graphics, drawing, icons, partially or completely. Example of the language that uses this programming paradigm includes the ladder diagram, the function block, and the sequential function chart.

The programming by demonstration paradigm is generally applicable only when the full functional system (to be programmed) is available. Therefore, applicability of PBD to program a new system that has not yet existed is questionable. Similarly, applying PBD concept to PLC programming requires the existence of a controlled system to be interacted with during the demonstration.

In this paper, the above problem is solved by an iterative approach to joint simulator-controller programming by demonstration method. This technique combines visual programming and PBD for jointly programming of controller devices and a simulation model of the controlled devices. Iterative nature of the method allows the initially incomplete knowledge on the system simulation model to be refined with the progress

of the iteration. Detail of the method is described in the next section.

3. An Iterative Joint Simulator-Controller PLC Programming by Demonstration Method

An iterative approach to joint simulator-controller programming by demonstration method is composed of four parts as follows (see Fig. 1):

- Model builder: This module is responsible for building relationships among controller devices and controlled system.
- System simulator: This module provides an animated display of the controlled system dynamics in response to a controller action.
- User action recorder: This module records a user action for the purpose of both programming the controlled system simulation model and the controller.
- Compiler: This module is used to transform a recorded user action into an actual PLC programming code.

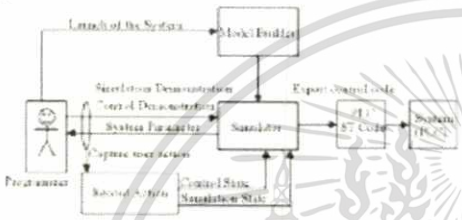


Fig. 1: Main modules of the proposed joint simulator-controller programming system

Conceptually, the system as shown in Figure 1 operates in one of five modes (states) as explained below (see Fig. 2).

1) System initialization mode: In this mode, a controlled system is created by a user through visual interface. Each controlled device and the corresponding properties can be defined. In addition, through visual programming, general relationship among controlled devices, and between controlled devices and the controller devices can be visually declared.

2) Simulator programming mode: Under this mode, a system simulation model, which captures the relationship among controlling and controlled devices, can be programmed through user demonstration. A user programs the model by demonstrating how controlled device's properties change in response to a controller action, as well as to changes in other controlled devices' properties.

3) Controller programming mode: In this mode, a user is allowed to define a controller action in response to a particular system state. As for the case of the simulator programming mode, a controller is programmed through user demonstration of actions.

4) Simulation mode: In this mode, a programming tool demonstrates to a user how a system state changes in response to a controller action and vice versa. The simulation is provided based on a system simulation model, constructed from previous user actions under Mode 2. Therefore, the model may be incomplete and the simulation may be inaccurate. System model and simulation accuracy can be corrected, however, through iteratively re-programming.

5) Compiler mode: When in this mode, the tool transforms user actions into an actual PLC programming code. For example, the tool developed in this study chooses ST (Structure Text) as specified by the IEC1131-3 standard as a targeted PLC programming language.

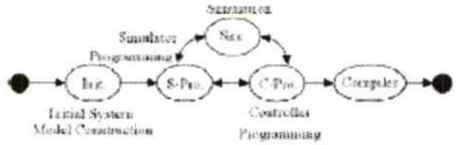


Fig. 2: Programming modes of the proposed method

4. The Tool and Its Usage

To explain how the tool developed using the above method is applied, in the following we will consider the case of an automatic pump system as shown in Fig. 3.

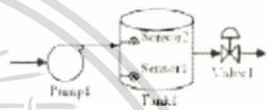


Fig. 3: An automatic pump system

4.1 Model builder

This module is active when the tool is in the system initialization mode. It is used for system model construction. Using a visual programming paradigm, a controlled and/or controller device is defined by the selection of icons from the tool palette. Each device's properties are defined through the object inspector (see Fig. 4).

Fig. 4: Screen shot of the tool's model builder

Table. 1: Standard object classes supported by the tool

Device	Description
1. Activator	A generic controller's output device
2. Sensor	A generic controller's input device
3. Container	A generic controlled device
4. Counter	A specialized counter device
5. Timer	A specialized timer device
6. General	A general purpose device class for extension
7. PipeLine	A connector component

Table 1 summarizes standard device classes as supported by the developed tool.

To illustrate how the tool is used for PLC programming, an automatic pump system as shown in Fig. 3 is used as a case study. To create a simulator model, relationship between controlling device and controlled device as well as the relation between controlled device and controlled device need to be defined. Through the language of visual programming, two types of relationship can be defined.

1) Association relationship: Once defined, allows the associated objects to exchange message (parameters, commands, etc.) with each other. For examples, in our case study, when Valve1 is connected with Tank1 through the connector device, opening or closing of Valve1 can have an effect on the properties (such as a water level) of Tank1. The direction of the connector's arrow head is used to identify the direction of message flow.

2) Dependency relationship: It applied to define a part-of relationship between two or more controlled devices, or between a controlled device and one or more controller devices. For example, in our case study, to install a controller sensor to measure a tank's water level, the sensor is placed inside the container Tank1 to establish the required relationship. In Figure 9, two such sensors, L1 and L2, are defined with this type of relationship with Tank1, so that two (low and high) water levels can be sensed.

4.2 User action recorder

This module is responsible to recording a user action. It operates in one of the two modes, the simulator and controller programming modes. The two modes, along with the simulation mode, can be inter-changed. Generally, the simulator programming mode is activated when a user changes the state of a controlling device of which the corresponding system response is not known yet. Once a system state change in response to the controller action is defined during the simulation programming mode, the tool should be switched to the simulation mode. From then, when a particular system state occurs, a user can switch to the controller programming mode so that an appropriate controller action can be defined.

To illustrate how to iteratively program a system simulation model and a controller, consider our case study of an automatic pump system when the system is initialized such as Pump1 is turned on (Fig. 7).

1) From the initialization mode, where the status of Pump1 is turned to "ON", the tool switches to the simulator programming mode. Based on the association relationship between Pump1 and Tank1 as established through visual programming during system model initialization, the tool prompts for a change to any of Tank1's properties. A user can then program the simulator model by raising the tank's water level. The tool records the action, with a dialogue for the user to define the rate of water level increase.

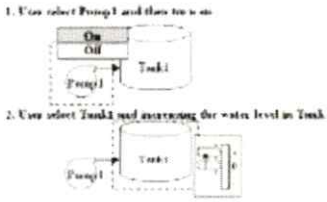


Fig. 7: Using the tool to program the system model when Pump1 is turned on

2) Once a simulator model is (partly) defined by user demonstration in Step 1, the tool is switched to the simulation mode. In this mode, the controlled system's properties are dynamically changed in response to the initial condition. In this case, the water level is gradually increased. When the water reaches the level specified by Sensor2, the tool prompts for an action. In response, a user switches to the controller programming mode.

3) The user then programs the controller by visually turning Pump1 off. The tool then records that action (see Fig. 8). Then, the tool is switched to the simulator programming model.

Fig. 8: Using the tool to program the system model when the water level arrives at the point specified by Sensor 2

4) In the simulator programming model, the user should program the simulation model by stopping the water level from further increase.

5) When the tool is switched to the simulation mode, the user informs the tool to switch to the controller programming mode when he sees the water level stops changing. Then, Valve1 is instructed to be open (turned on) (see Fig. 9).

6) The tool then turns into the simulator programming mode. In this mode, Tank1 is highlighted as a result of the established relationship between Valve1 and Tank1.To drain the water level in the tank, a user select the tank's property that is affected by this controller action and then lowering the water level. The tool then records the action (see Fig. 9).

7) While in the simulation mode, the water level is decreasing. When it passes the level below that specified by Sensor1, a user can proactively turns the tool into a controller programming mode himself. He then can

program the controller by visually turning Pump1 on (see Fig. 10).

8)

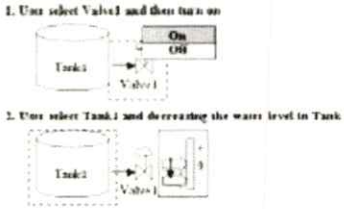


Fig. 9: Using the tool to jointly program the system when Valve1 is turned on

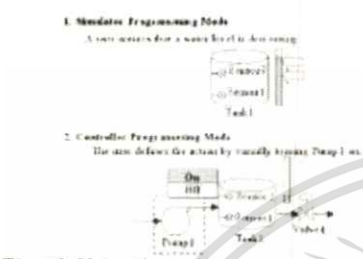


Fig. 10: Using the tool to program the system model when it passes the level below that specified by Sensor1

From the above user demonstrations, a truth table describing all possible logic combinations of the controller devices' states can be derived as shown in Table 2

Table. 2: A truth table as generated from user demonstrated actions

Sensor1	Sensor2	Pump1
0	0	0
0	1	-
1	0	X
1	1	1

From the truth table (see Table 2), an actual controller program can be generated when the tool is switched to the compiler mode. For the case of our automatic pump system, the generated code for each scenario as explained so far is summarized in Table 3.

Table. 3: Code generated corresponding to visited scenarios and demonstrations

Scenario	ST generated code
Sensor Sensor1; Sensor Sensor2;	INPUT_VAR Sensor1, Sensor2:BOOL; END_VAR
Activator Pump1;	OUTPUT_VAR Pump1:BOOL; END_VAR
IF(Sensor1.State = Off & Sensor2.State = Off) Pump1.State = On;	IF Sensor1=FALSE AND Sensor2= FALSE THEN Pump1:=TRUE; END_IF;
IF(Sensor1.State = On & Sensor2.State = On) Pump1.State = Off;	IF Sensor1 = TRUE AND Sensor2 = TRUE THEN Pump1 :=FALSE; END_IF;

4.3 Missing cases

From the truth table, it reveals that there is a case not known to the user. It is the case where Sensor1 is off and Sensor2 is on. This case, fortunately, can be automatically removed as a unfeasible case. However, in practice, there can be a case not shown to a user where it is not clear whether the case is feasible or not. When this occurs, the tool will collect all these missing cases and display them in the simulator mode. The tool then asks the user to provide an appropriate action in response to each scenario corresponding to a missing case.

5. CONCLUSION

In this paper, the problem of PLC programming by the programming by demonstration paradigm has been considered. A new iterative joint controller-simulator programming-by-demonstration method has been proposed. Details of the design and implementation of the PLC programming tool based on the proposed method has been given. Use of the tool in an iterative manner, for joint programming of both controlled system and controller, has been described through the case study of an automatic pump system. Based on a user demonstrated action, the tool can generate a PLC programming code in a form of a structure text as defined by IEC 1131-3 standard. All missing cases are collected and analyzed by the tool. Those that cannot be rejected as feasible, the tool asks for a user action by creating a scenario corresponding to each missing case and displaying it in the simulator mode.

6. REFERENCES

[1] Gordon W. Paynter and Ian H. Witten, "Applying machine learning to programming by demonstration", University of California, Riverside and University of Waikato New Zealand, 2004

[2] Haiying She and Axel Graeser, "Closed Loop Control and Automatic Set Point Generation in Programming by Demonstration for Service Robotic Tasks", University of Bremen, Germany, 2004.

[3] Richard M. Voyles, "Toward Gesture-Based Programming: Agent-Based Haptic Skill Acquisition and Interpretation", University of Minnesota, 1997.

[4] S. Munch, J. Kreuziger, M. Kaiser, R. Dillmann , "Robot Programming by Demonstration (RPD) Using Machine Learning and User Interaction Methods for the Development of Easy and Comfortable Robot Programming System", University of Karlsruhe, Germany, 1994.

[5] Tessa Lau, "Programming by Demonstration: a Machine Learning Approach", University of Washington, 2001.

[6] Lawrence D.Bergman, Tessa A. Lau, Vittrio Castelli, Daniel Oblinger, "Programming-by-Demonstration for Behavior-based User Interface Customization", IBM T.J. Watson research Center, USA, 2004.

ประวัติผู้เขียน

ชื่อ-นามสกุล	นายอิทธิกร ช่างสากล
วัน เดือน ปีเกิด	28 มกราคม พ.ศ.2513 ที่จังหวัดลพบุรี
ที่อยู่	1 ถ.เทศบาลรังสฤษฎ์เหนือ แขวงลาดยาว เขตจตุจักร กรุงเทพมหานคร 10900 โทร.0-2954-2095
ประวัติการศึกษา	2541 วิทยาศาสตรบัณฑิต สาขาวิทยาการคอมพิวเตอร์ สถาบันราชภัฏสวนสุนันทา
ความชำนาญเฉพาะด้าน	1) ระบบเครือข่ายคอมพิวเตอร์ 2) ระบบงานคอมพิวเตอร์
ประสบการณ์การทำงาน	
พ.ศ.2533-2544	ตำแหน่งเจ้าหน้าที่เครื่องคอมพิวเตอร์ ศูนย์สารสนเทศ กระทรวงศึกษาธิการ
พ.ศ.2544-ปัจจุบัน	ตำแหน่งนักวิชาการคอมพิวเตอร์ ศูนย์เทคโนโลยีสารสนเทศและการสื่อสาร กระทรวงศึกษาธิการ