ภาควิชาครุศาสตร์วิศวกรรม

คณะครุศาสตร์อุตสาหกรรม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองปริญญานิพนธ์

ชื่อหัวข้อ การควบคุมระดับด้วยไมโครคอมพิวเตอร์

Level Controled by Microcomputer

ชื่อนักศึกษา

1. นายธงฉาน	คำศิริ	รหัสประจำตัว	42035370
2. นายสิทธิชัย	บุญอนันต์	รหัสประจำตัว	42035382
3. นางสาวสุพรรณษา	อินทรปัญญา	รหัสประจำตัว	42035384

หลักสูตร ครุศาสตร์อุตสาหกรรมบัณฑิต **สาขาวิชา** เทคโนโลยีการวัดคุมทางอุตสาหกรรม

อาจารย์ที่ปรึกษา อาจารย์สุระชัย พิมพ์สาลี

อาจารย์ที่ปรึกษาร่วม อาจารย์อำพล ทองระอา

คณะกรรมการสอบปริญญานิพนธ์		ลายมือชื่อ
1. อาจารย์สุระชัย	พิมพ์สาลี	.....
2. อาจารย์อำพล	ทองระอา	.....
3. อาจารย์โกศล	ตราชู	.....
4. อาจารย์พงษ์เกียรติ	เชษฐพิทักษ์สกุล	.....
5. อาจารย์ปิยะ	ศุภวราสุวัฒน์	.....

วัน/เดือน/ปีที่สอบ วันเสาร์ที่ 9 ธันวาคม พ.ศ. 2543 เวลา 14.00 น.

สถานที่สอบ ห้อง ค.301 คณะครุศาสตร์อุตสาหกรรม สจล.

ภาควิชารับรองแล้ว

ลงนาม.....

(ผศ.วิสุทธิ์ อธิพรธรรม)

หัวหน้าภาควิชาครุศาสตร์วิศวกรรม

วันที่ 4 เดือน ธค พ.ศ. 2543

ปริญญานิพนธ์

ระบบควบคุมระดับด้วยโปรแกรมไมโครคอมพิวเตอร์

LEVEL CONTROL SYSTEM BY MICROCOMPUTER

นายธงฉาน คำศิริ
นายสิทธิชัย บุญอนันต์
นางสาวสุพรรณยา อินทรปัญญา

ปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรครุศาสตร์อุตสาหกรรมบัณฑิต
สาขาวิชาเทคโนโลยีการวัดคุมทางอุตสาหกรรม
ภาควิชาครุศาสตร์วิศวกรรม คณะครุศาสตร์อุตสาหกรรม
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2543

ปริญญานิพนธ์

เรื่อง ระบบควบคุมระดับด้วยโปรแกรมไมโครคอมพิวเตอร์

LEVEL CONTROL SYSTEM BY MICROCOMPUTER

วัตถุประสงค์

1. เพื่อศึกษาโครงสร้างและการทำงานของการควบคุมระดับด้วยโปรแกรม
2. เพื่อศึกษาการเขียนโปรแกรมด้วยโปรแกรม Microsoft Visual C++ Version 6.0
3. เพื่อศึกษาเกี่ยวกับกระบวนการควบคุมแบบ PID
4. เพื่อศึกษาการติดต่อสื่อสารระหว่างคอมพิวเตอร์กับชุดควบคุมระดับ
5. เพื่อออกแบบการเขียนโปรแกรมควบคุมชุดควบคุมระดับ
6. เพื่อสร้างชุดประยุกต์ใช้งานควบคุมระดับด้วยอุปกรณ์ที่มีราคาประหยัด

ประโยชน์ที่คาดว่าจะได้รับ

1. ได้ความรู้เกี่ยวกับโครงสร้างและการทำงานของการควบคุมระดับด้วยโปรแกรม
2. ได้ความรู้เกี่ยวกับการเขียนโปรแกรม Microsoft Visual C++ Version 6.0
3. ได้ความรู้เกี่ยวกับกระบวนการควบคุมแบบ PID
4. ได้ความรู้เกี่ยวกับการติดต่อสื่อสารระหว่างคอมพิวเตอร์กับชุดควบคุมระดับ
5. ได้โปรแกรมการควบคุมกระบวนการควบคุมระดับ
6. ได้ชุดประยุกต์ใช้งานควบคุมระดับที่มีราคาประหยัด

ชื่อหัวข้อ	ระบบควบคุมระดับด้วยโปรแกรมไมโครคอมพิวเตอร์	
นักศึกษา	นายธงฉาน	คำศิริ
	นายสิทธิชัย	บุญอนันต์
	นางสาวสุพรรณยา	อินทรปัญญา
อาจารย์ที่ปรึกษา	อาจารย์อำพล	ทองระอา
อาจารย์ที่ปรึกษาร่วม	อาจารย์สุรพงษ์	สิริพงศ์ดี
หลักสูตร	ครุศาสตร์อุตสาหกรรมบัณฑิต	
สาขาวิชา	เทคโนโลยีการวัดคุมทางอุตสาหกรรม	
ปีการศึกษา	2543	

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้เสนอการศึกษาและการออกแบบชุดประยุกต์ใช้งาน ระบบควบคุมระดับด้วยโปรแกรมไมโครคอมพิวเตอร์ โดยใช้โปรแกรม Microsoft Visual C++ Version 6 ควบคุมระดับของน้ำโดยการเขียนโปรแกรม Microsoft Visual C++ Version 6 ผ่านตัวการ์ด ET-PC DIO เป็นการ์ดที่เชื่อมต่อกับเครื่องคอมพิวเตอร์ (PC) ซึ่งสามารถขยายระบบอินพุตและเอาต์พุตได้โดยรับสัญญาณอินพุตแล้วส่งสัญญาณเอาต์พุตออกมาได้ ทั้งในรูปของแอนะลอก (Analog) และดิจิตอล (Digital) ซึ่งการแสดงผลการใช้งานสามารถดูได้จากตัวชุดประยุกต์ใช้งานระบบควบคุมระดับและหน้าจอคอมพิวเตอร์ ทั้งยังควบคุมการทำงานได้ที่ตัวเครื่องคอมพิวเตอร์จากโปรแกรมการทำงาน สามารถควบคุมได้ 2 แบบ คือ แบบ AUTO และ แบบ MANUAL ซึ่งการพัฒนาระบบนี้ทำให้สามารถนำโปรแกรม Microsoft Visual C++ Version 6 ไปพัฒนาใช้งานควบคุมระบบควบคุมอื่นๆ ได้อย่างกว้างขวาง และมีประสิทธิภาพมากยิ่งขึ้น

Thesis Title	Level Control System By Microcomputer
Students	Mr.Tongchan Kumsiri
	Mr. Sittichai Boonanan
	Miss. Supansa Intarapanya
Advisor	Mr. Amphon Tongra-ar
Co-Advisor	Mr. Surapong Siripongdee
Education Level	Bachelor of Science in Industrial Education
Program in	Industrial Instrument Technology
Academic Year	2000

ABSTRACT

This Thsis presents studies and invents Level Control System by Microcomputer with Program Microsoft Visual C++ Version 6.The characteristic of control process level for water and controls by interface to ET-PCDIO card. The project can interfaces to computer (pc) and extend input , out put will receive input signal , output signal for analog and digital. The project can be controls process to program control on computer with system serial transmission and controls both automatically and manually. The development can assit the application of Microsoft Visual C++ Version 6.0 in control system wider and more efficient way.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงไปด้วยดี เนื่องมาจากความร่วมมือของสมาชิกภายในกลุ่มทุกท่าน ขอขอบคุณอาจารย์ที่ปรึกษา อาจารย์ภาควิชาเทคโนโลยีวัดคุมทางอุตสาหกรรม คณะวิศวกรรมศาสตร์ และคณาจารย์รวมทั้งเจ้าหน้าที่ของภาควิชาครุศาสตร์วิศวกรรมทุกท่าน ที่ได้ให้ความอนุเคราะห์เครื่องมือ อุปกรณ์ เอกสารต่างๆ ที่เกี่ยวข้องกับปริญญานิพนธ์ ตลอดจนคำแนะนำ แนวความคิด ความรู้ต่างๆ แนวทางการแก้ไขปัญหา ในการจัดทำปริญญานิพนธ์ขอขอบคุณห้องสมุดคณะครุศาสตร์อุตสาหกรรม ที่ช่วยอำนวยความสะดวกในการค้นคว้าหาข้อมูล และที่สำคัญขอขอบคุณเพื่อนๆ คณะวิศวกรรมศาสตร์ และเพื่อนๆสาขาเทคโนโลยีการวัดคุมทางอุตสาหกรรม ห้อง2/1 ทุกคนที่ร่วมสุขและทุกข์กันมา

คณะผู้จัดทำขอกราบขอบพระคุณท่านบุพการี ที่ให้การสนับสนุนด้านการศึกษาทุกๆ ส่วน และเป็นผู้ให้กำลังใจด้วยดีตลอดมาตั้งแต่อดีตจนถึงปัจจุบัน และขอบคุณอีกครั้งสำหรับกำลังใจจากเพื่อนๆ และความช่วยเหลือในหลายด้านที่มีให้กันในยามทุกข์ยาก

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญตาราง

สารบัญภาพ

สารบัญภาพ (ต่อ)

สารบัญภาพ (ต่อ)

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปริญญานิพนธ์

ในวงการอุตสาหกรรมได้มีการพัฒนาทางด้านระบบควบคุมให้มีความสะดวก และง่ายยิ่งขึ้น และในปัจจุบันโรงงานอุตสาหกรรมส่วนใหญ่ได้ใช้ระบบการควบคุมแบบอัตโนมัติ เพื่อเพิ่มผลผลิตและทำให้การผลิตมีประสิทธิภาพมากยิ่งขึ้น ทั้งในด้านระยะเวลาในการผลิต คุณภาพมาตรฐาน ความแม่นยำ ความน่าเชื่อถือ สะอาดและปลอดภัย ทั้งยังเป็นการลดกำลังคนงานค่าใช้จ่ายซึ่งเกี่ยวกับต้นทุนการผลิตทำให้ผู้ผลิตมีผลกำไรมากขึ้น ซึ่งการนำเอาระบบคอมพิวเตอร์เข้ามาใช้ในกระบวนการควบคุมทำให้การทำมีความสะดวกยิ่งกว่าเดิม และมีความทันสมัยทันกับเทคโนโลยีใหม่ๆ

โปรแกรมไมโครคอมพิวเตอร์ต่าง ๆ ที่สามารถนำมาใช้ควบคุมกระบวนการทำงานมีมากมาย เช่น โปรแกรม LAB VIEW โปรแกรม Microsoft Visual C++ และโปรแกรม Microsoft Visual Basic เป็นต้น ซึ่งปัจจุบันมีการพัฒนาเป็นระดับต่าง ๆ ขึ้นมาและยังมีอีกหลายโปรแกรมที่เหมาะสำหรับการใช้งานในรูปแบบต่างๆ ส่วนใหญ่จะใช้โปรแกรมเหล่านี้จากคอมพิวเตอร์ควบคุมการทำงาน เช่น การควบคุมแรงดันอัตราไหล การควบคุมระดับและอุณหภูมิที่ใช้ในระบบ DCS ในการควบคุมการสั่งงาน

การควบคุมกระบวนการโดยใช้โปรแกรมไมโครคอมพิวเตอร์ที่มีการใช้งานสะดวก และสามารถเชื่อมต่อกับอุปกรณ์ภายนอกได้ดี อย่างเช่น โปรแกรม Microsoft Visual C++ Version 6.0 เป็นโปรแกรมที่กลุ่มผู้จัดทำโครงงานมีความสนใจโดยมีการพัฒนาโปรแกรมมาจากภาษา C และเนื่องจากกลุ่มผู้จัดทำศึกษาอยู่ในสาขาเทคโนโลยีการวัดคุมทางอุตสาหกรรม และได้เรียนวิชาการเชื่อมต่อไมโครคอนโทรลเลอร์และไมโครโปรเซสเซอร์มา จึงเกิดแนวคิดที่จะประยุกต์ใช้การควบคุมชุดประยุกต์ใช้งานด้วยการเขียนโปรแกรมด้วย Microsoft Visual C++ Version 6.0 มาเชื่อมต่อกับตัวการ์ดซึ่งสามารถรับ และส่งสัญญาณติดต่อกันระหว่างตัว PC และอุปกรณ์ต่าง ๆ ทางด้านฮาร์ดแวร์ที่มีอยู่ในชุดประยุกต์ใช้งานได้ เพื่อนำไปควบคุมกระบวนการให้ได้มากและมีประสิทธิภาพดียิ่งขึ้น

1.2 ขีดความสามารถของโครงงาน

1. เขียนโปรแกรมควบคุมการทำงานของชุดประยุกต์ใช้งานกับเครื่องคอมพิวเตอร์ ด้วยโปรแกรม Microsoft Visual C++ Version 6.0 ได้

2. สามารถควบคุม Process ได้ด้วยโปรแกรม Microsoft Visual C++ V6.0 ทั้งแบบ Auto และ Manual

3. แสดงผลการควบคุมออกทางหน้าจอคอมพิวเตอร์ ทั้งรูปแบบ Auto และ Manual

1.3 เนื้อหาโดยสังเขป

บทที่ 2 ทฤษฎีและหลักการ กล่าวถึงความเป็นมาและแนวคิดของโปรแกรม Microsoft Visual C++ Version 6.0 ความรู้เบื้องต้นเกี่ยวกับการควบคุมโปรเซส หลักการของ PID และการทำงาน การเชื่อมต่อสัญญาณข้อมูล การแปลงสัญญาณแอนะลอกเป็นดิจิตอล (A/D) และการแปลงสัญญาณดิจิตอลเป็นแอนะลอก(D / A) เกี่ยวกับการใช้งานในโครงงาน ลักษณะการทำงานและใช้งานของตัวการ์ด ET – PCDIO

บทที่ 3 การออกแบบการสร้างและหลักการทำงาน กล่าวถึงการออกแบบและการสร้างของชุดประยุกต์ใช้งาน ได้แก่ วงจรตรวจจับระดับ (Sensor Level circuit) วงจรขับสัญญาณอินพุตและเอาต์พุต และวงจรInstrument amplifier การออกแบบการควบคุมด้วยโปรแกรม Microsoft Visual C++ version 6.0 เพื่อควบคุมการทำงานส่วนต่างๆ

บทที่ 4 การทดลองและผลการทดลอง กล่าวถึงการทดลองและการตรวจสอบการทำงานของส่วนการทำงานต่างๆ 3 ส่วนใหญ่ คือ ส่วนที่หนึ่งการทดลองตัว Plant ทดลอง ส่วนที่สองเป็นส่วนของการตรวจสอบการทำงานของวงจรขับสัญญาณ วงจรตรวจจับแรงดัน และวงจรInstrument Amplifier ส่วนที่สามเป็นการทดสอบการทำงานของโปรแกรมเพื่อไปควบคุม Plant ทดลอง

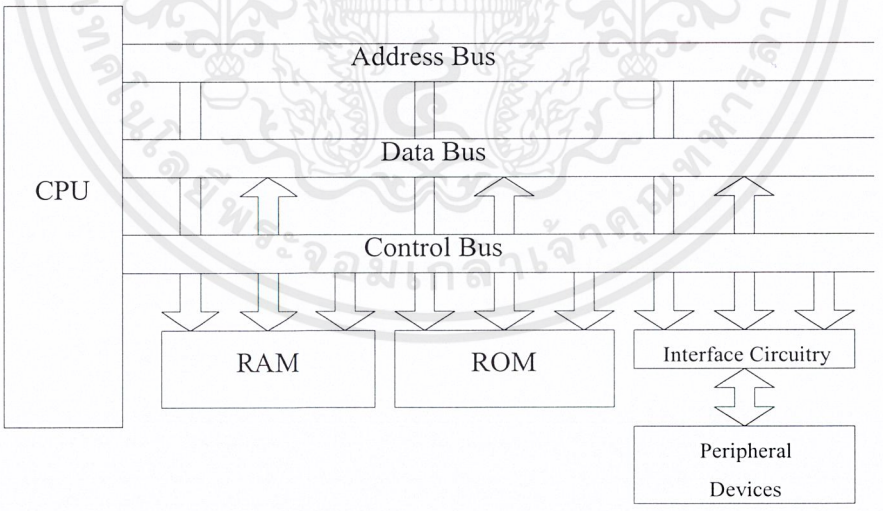
บทที่ 5 กล่าวถึงบทสรุปข้อบกพร่องต่างๆ ของชุดประยุกต์ใช้งาน ปัญหาที่เกิดขึ้นในการจัดสร้างและกล่าวถึงแนวทางการปรับปรุงแก้ไข และการพัฒนาโปรแกรมชุดประยุกต์การใช้งานต่อไป

บทที่ 2
ทฤษฎีและหลักการ

2.1 กล่าวนำ

เครื่องคอมพิวเตอร์ เป็นสมองกลอิเล็กทรอนิกส์ที่ทำงานอย่างหนึ่งได้เร็วและดีกว่าคนมากระดับหนึ่ง สามารถรับและส่งข้อมูลเป็นจำนวนมากมีการแยกแยะได้เร็ว ด้วยเหตุนี้จึงใช้ความของสามารถของคอมพิวเตอร์มาประมวลข้อมูลจำนวนมาก ๆ ในเวลาสั้น ๆ ความสามารถสูงของคอมพิวเตอร์นี้จึงนำมาใช้งานในด้านธุรกิจอุตสาหกรรม และห้องทดลองเพื่อลดรายจ่ายของงานสำหรับการออกแบบด้วยคอมพิวเตอร์ เป็นเครื่องมือที่ใช้ในด้านอุตสาหกรรมคอมพิวเตอร์ได้อย่างดี ซึ่งนำมาช่วยในการควบคุมการผลิตต่างๆ หรือใช้ออกแบบชิ้นส่วนเครื่องจักรและอุปกรณ์ต่างๆ

ไมโครคอมพิวเตอร์ คือ คอมพิวเตอร์ที่มีไมโครโปรเซสเซอร์เป็นตัวประมวลผลกลางหรือ CPU และทำงานร่วมกับหน่วยอื่นอีก 2 หน่วยคือ หน่วยความจำ (Memory Unit) หน่วยรับส่งสัญญาณเข้าออก (Input output unit) เขียนเป็นบล๊อกไดอะแกรมดังรูปที่ 2.1 ได้ดังนี้



รูปที่ 2.1 บล๊อกไดอะแกรมของไมโครคอมพิวเตอร์

ไมโครคอมพิวเตอร์จัดเป็นเครื่องคอมพิวเตอร์ส่วนบุคคล (Prosonal Computer) หรือเรียกย่อๆ ว่า PC นั่นเอง จากองค์ประกอบด้วยหน่วยต่างๆ คือ หน่วยควบคุม หน่วยคำนวณ

หน่วยความจำและหน่วยรับส่งข้อมูล ส่วนวงจรอิเล็กทรอนิกส์ที่ทำหน้าที่ของหน่วยต่างๆ เรียกว่า Hardware วงจรเหล่านี้ทำงานต่างกัน เช่น หน่วยควบคุมจะถูกออกแบบมาให้เข้าใจหรือตีความหมายรหัสที่อยู่ในรูปตัวเลขชุดหนึ่งซึ่งสามารถเก็บไว้ในหน่วยความจำได้ รหัสนี้ถือว่าเป็น คำสั่งสำหรับคอมพิวเตอร์เรียกว่า Instruction คอมพิวเตอร์เครื่องหนึ่งจะมีรหัสที่หน่วยควบคุมเข้าใจได้อยู่ชุดหนึ่ง ซึ่งคำสั่งหลายๆ คำสั่งจะสามารถบังคับให้คอมพิวเตอร์ทำอะไรก่อนหลังเป็นขั้นตอนจนได้คำตอบสุดท้ายที่ต้องการ ชุดคำสั่งนี้เราเรียกว่าโปรแกรม (โปรแกรม) หรือ (Software) ของเครื่อง โปรแกรมสำหรับให้คอมพิวเตอร์ทำงานแต่ละงานย่อมไม่เหมือนกัน เพราะแต่ละงานย่อมมีขั้นตอนปลีกย่อยแตกต่างกัน และในปัจจุบันมีโปรแกรมการใช้งานมากมายและมีการพัฒนาขึ้นมาเรื่อยๆ เพื่อให้มีความสะดวกและเหมาะสมกับงานในแต่ละด้าน

2.2 ทฤษฎีและแนวคิดของ Microsoft Visual C++

โปรแกรม Visual C++ 6.0 ได้รับการพัฒนาขึ้นมาจาก Microsoft C / C++ ที่ทำงานบนระบบปฏิบัติการวินโดวส์ได้อย่างเต็มที่ รองรับการพัฒนาการโปรแกรมบนวินโดวส์โดยมี MFC (Microsoft Foundation Class) เป็นไลบารี่ที่ช่วยอำนวยความสะดวกในการพัฒนาโปรแกรมบนวินโดวส์ เนื่องจากการเขียนโปรแกรม Microsoft Visual C++ V 6.0 เป็นแนวทางการเขียนโปรแกรม ที่ได้แนวคิดมาจาก OOP หรือ Object Oriented Promming ซึ่งเป็นแนวคิดในการเขียนโปรแกรมเชิงวัตถุ หรือการเขียนโปรแกรมแบบอ้างอิงวัตถุ ซึ่งแนวคิด OOP มาใช้ในการเขียนโปรแกรม และการจัดการข้อมูลซึ่งพบว่าโปรแกรมหรือฟังก์ชั่น มีความเป็นอิสระแก่กันอย่างชัดเจน คือโปรแกรมหรือฟังก์ชั่นแต่ละตัวถึงแม้จะมาจากที่เดียวกันแต่มันสามารถทำงานในคนละหน้าที่เก็บข้อมูลคนละค่าได้ โดยจะไม่มายุ่งเกี่ยวกันแต่อย่างใด และการเขียนโปรแกรมแบบ OOP ก็มีลักษณะคล้ายการเขียนโปรแกรมแบบโครงสร้าง โดยใช้หลักการที่คล้ายกันแต่ OOP นั้นจะมีประสิทธิภาพที่สูงกว่า เพราะเป็นการนำเอาคุณลักษณะ (Attribute) และวิธีการ (Method) เช่น ข้อมูล , ตัวแปร หรือฟังก์ชั่นของวัตถุนั้นมารวมไว้ในกลุ่ม ๆ เดียวกัน ที่เราเรียกว่าคลาส (Class)

ข้อแตกต่างระหว่างการเขียนโปรแกรมแบบโครงสร้าง และแบบ OOP คือการเขียนโปรแกรมแบบ OOP เราสามารถเพิ่มฟังก์ชั่นเข้าไปในกลุ่มข้อมูลได้ แต่การเขียนโปรแกรมแบบโครงสร้าง (Struct) จะเป็นการเก็บตัวแปรเพียงอย่างเดียว ซึ่งในภาษา C จะเป็นการเขียนโปรแกรมแบบโครงสร้าง และในโปรแกรม Microsoft Visual C++ V. 6.0 นี้ ก็เป็นการพัฒนามาจาก โปรแกรมภาษา C โดยมีการรวมเอาโครงสร้างของภาษา C กับการเขียนโปรแกรมแบบ Object Oriented Programming มาพัฒนา เป็น Microsolf Visual C++ ซึ่งการใช้งานจะมีประสิทธิภาพสูงและการใช้งานสะดวกยิ่งขึ้นเพราะสามารถเลือกรูปแบบในการสร้างได้ทั้งรูปแบบอัตโนมัติและต้องการสร้าง

โปรแกรมด้วยตนเอง และโปรแกรมที่ใช้แนวโน้มของ OOP ได้รับการยอมรับและพัฒนามาใช้ในระบบต่าง ๆ มากมาย เช่น ระบบปฏิบัติการวินโดวส์

ดังนั้น การใช้งานตัวแปรแบบโครงสร้างในภาษา C และการใช้งานคลาสแบบ OOP จะเห็นว่ามีความคล้ายกันมาก โดยเฉพาะการรวมกันของตัวแปรต่าง ๆ ในโครงสร้าง แต่การใช้งานคลาสจะเห็นได้ชัดว่า ภายในคลาส เราสามารถแทรกฟังก์ชั่นไปได้ ซึ่งจะต่างจากการใช้ตัวแปรแบบโครงสร้างของภาษา C ที่ไม่สามารถกำหนดฟังก์ชั่นลงไปในโครงสร้างได้ และยังไม่สามารถกำหนดขอบเขตต่าง ๆ ลงไปได้อีกด้วย ซึ่งก็สามารถ สรุปได้ว่า

คลาส (Class) ก็คือ การรวมคุณลักษณะและการใช้งานของวัตถุอย่างน้อยหนึ่งอย่าง มาไว้ในกลุ่มเดียวกัน

ออบเจ็กต์ (Object) คือ วัตถุที่เป็นตัวแปรคลาส เป็นรูปแบบของคลาสที่มีตัวตน ที่สามารถนำไปใช้งานได้

ในยุคที่ MFC ยังไม่ถือกำเนิด เมื่อจะพัฒนาโปรแกรมบนระบบปฏิบัติการวินโดวส์จะต้องใช้ SDK (Software Development Kit) และคอมไพลเลอร์ภาษา C เช่น Microsoft C++ , Borland C++ ช่วยในการเขียนโปรแกรมโค้ดโปรแกรมที่เขียนจะขึ้นด้วย SDK นี้จะค่อนข้างซับซ้อน และสร้างความลำบากในการศึกษาทำความเข้าใจโปรแกรม จึงได้มีการสร้างคลาสขึ้นมาหนึ่ง เขียนขึ้นโดยใช้โครงสร้างของ OOP (Objected Oriented โปรแกรมming) ด้วยภาษา C++ ชื่อว่า MFC หรือ Microsoft Foundation Class ใช้สำหรับอำนวยความสะดวกในการเขียนโปรแกรมบนวินโดวส์ โดยเฉพาะ โค้ดโปรแกรม

โปรแกรม Visual C++ 6.0 มีรูปแบบการอินเตอร์เฟสคล้ายกับ Visual C++ 5.0 เวอร์ชั่นก่อน แต่จะมีจุดหนึ่งที่แตกต่างกันคือ ระบบช่วยเหลือของ โปรแกรม Visual C++ 6.0 นั้นจะแยกออกจากตัวโปรแกรม (IDE) ซึ่งระบบช่วยเหลือจะอยู่ในรูปของ HTML Help และในการที่ใช้โปรแกรม Visual C++ 6.0 ได้นั้นจะต้องติดตั้งโปรแกรม Internet Explorer 4.0 (IE4) ลงไปก่อนด้วย หากในระบบของไม่มี IE4 โปรแกรมติดตั้ง Visual C++ จะทำการติดตั้งให้ก่อน

การที่โปรแกรม Visual C++ 6.0 ต้องการให้ติดตั้ง IE4 ลงไปนั้นก็เป็นเพราะ โปรแกรมจะใช้ความสามารถของ IE4 ในการแสดงระบบช่วยเหลือ ซึ่งอยู่ในรูปของ HTML Help แต่ถ้าไม่ต้องการติดตั้ง IE4 ให้ติดตั้งระบบปฏิบัติการ Windows 98 ลงไปก็ได้ เพราะถ้าติดตั้งระบบปฏิบัติการ Windows 98 แล้วก็สามารถ ติดตั้งโปรแกรม Visual C++ 6.0 ลงไปได้เลย โดยไม่ต้องติดตั้ง IE4 อีก (เพราะ IE4 รวมอยู่กับระบบปฏิบัติการ Windows 98 แต่ถ้ายังต้องการใช้ระบบปฏิบัติการ Windows 95 ก็สามารถติดตั้งได้โดยจะต้องติดตั้ง IE4 ก่อน)

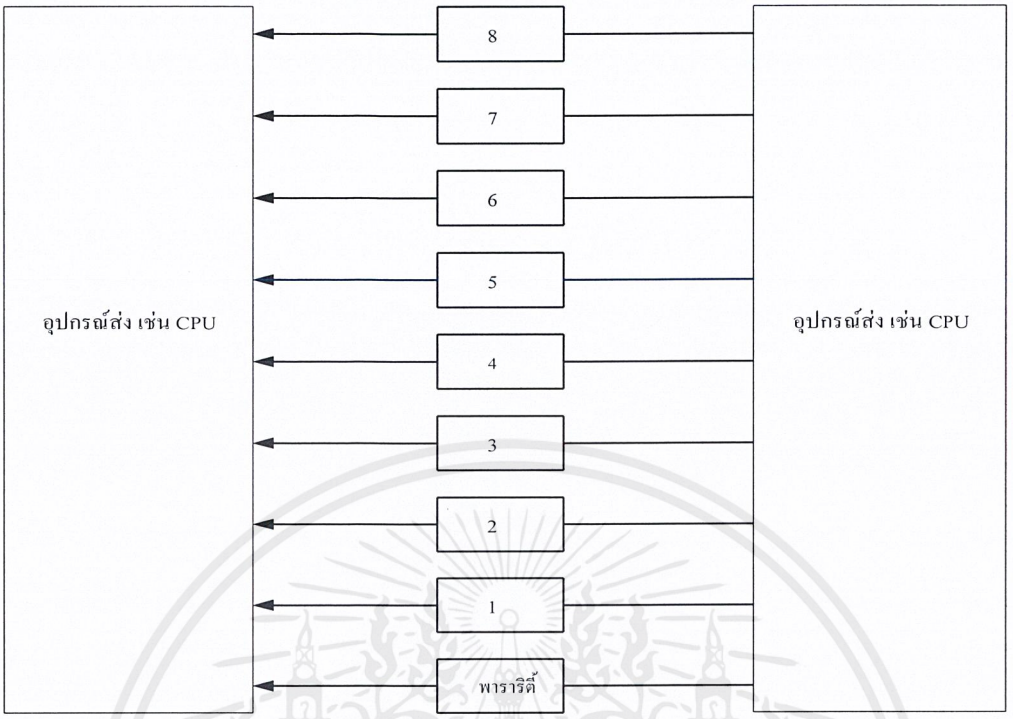
จากหลักการและความสามารถของโปรแกรมจึงนำมาใช้ในการออกแบบใช้งานในด้านการควบคุมการผลิตและกระบวนการต่างๆ เพื่อผลประโยชน์ทางด้านธุรกิจต่างๆหลายด้าน โดยเฉพาะด้านอุตสาหกรรมซึ่งมีการนำโปรแกรม Microsoft visual C++ มาควบคุมเพื่อลดระยะเวลาในการควบคุมและความสะดวกต่างๆ ในกระบวนการ (Process)

จากกระบวนการต่าง ๆ ที่ต้องใช้งานโดยมีการติดต่อกันระหว่างส่วนของโปรแกรมไมโครคอมพิวเตอร์และอุปกรณ์ที่ใช้งานก็ต้องมีการติดต่อสื่อสารกันเพื่อให้สามารถรับคำสั่งการทำงานในส่วนต่างๆ ได้ตามที่เขียนโปรแกรมไว้ ส่วนนี้จะเป็นส่วนสำคัญอย่างหนึ่งที่จะทำให้กระบวนการนั้นทำงานครบตามรูปแบบ

2.3 การสื่อสารข้อมูลแบบขนาน

ลักษณะของการสื่อสารข้อมูลแบบขนานนั้น จะเป็นการสื่อสารข้อมูลที่ข้อมูลจะสื่อสารรับ-ส่งโดยผ่านสาย หรือช่องสัญญาณพร้อมกันหลายๆ เส้น ดังแสดงในรูปที่ 2.2 โดยที่จำนวนของสัญญาณจะมีจำนวนไม่แน่นอน ขึ้นอยู่กับโครงสร้างการประมวลผลข้อมูลของระบบนั้นๆ ข้อดีของการสื่อสารข้อมูลแบบนี้ คือ สามารถสื่อสารข้อมูลกันได้ในระยะเวลาสั้นๆ แต่ ก็มีข้อเสีย คือ จะสิ้นเปลืองสายสัญญาณเป็นจำนวนมาก และถ้ายิ่งใช้ในการสื่อสารข้อมูลในระยะทางไกลๆ นอกจากจะสิ้นเปลืองค่าใช้จ่ายจำนวนมากแล้ว ยังทำให้สัญญาณถูกลดทอนลงไปด้วย

ดังนั้น โดยทั่วไปแล้วการสื่อสารข้อมูลแบบขนานจึงนิยมนำไปใช้กับการสื่อสารข้อมูลในระยะเวลาสั้นๆ ที่ต้องการสื่อสารข้อมูลด้วยอัตราเร็ว เช่น การเชื่อมต่อของสัญญาณระหว่างหน่วยประมวลผลกลางกับอุปกรณ์รอบข้าง หรือการสื่อสารข้อมูลของมือวัดและอุปกรณ์ต่าง ๆ กับเครื่องไมโครคอมพิวเตอร์



รูปที่2.2 โครงสร้างการทำงานของการสื่อสารข้อมูลแบบขนาน

2.4 ลักษณะคุณสมบัติเฉพาะของโปรเซส (Process Characteristic)

2.4.1 ความหมายของโปรเซส

โปรเซส คือ ขบวนการหรืออุปกรณ์ที่ใช้แปลงสภาพวัตถุดิบทางฟิสิกส์หรือทางเคมี

โปรเซสแบ่งออกเป็น 3 ชนิดคือ

- Continous process คือ โปรเซสที่ดำเนินต่อเนื่องตลอดเวลา
- Batch process คือ โปรเซสที่ขาเข้าออก ไม่ต่อเนื่องกัน
- Plant คือ ที่รวมของโปรเซสหลายโปรเซสที่สัมพันธ์กัน

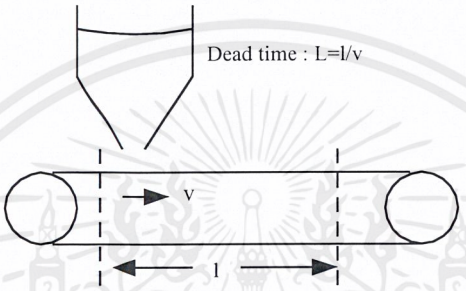
ตัวอย่างของโปรเซส

- การทำวัตถุร้อนหรือเย็น อุปกรณ์ที่ใช้เช่น Heat exchanger
- การทำให้ของเหลวหรือผงเคลื่อนที่ อุปกรณ์ที่ใช้เช่น ท่อ , สายพาน , ปั๊ม
- การเพิ่มหรือลดแรงดัน อุปกรณ์ที่ใช้เช่น ถังแรงดัน
- การทำปฏิกริยาของสารเคมี อุปกรณ์ที่ใช้เช่น ถังทำปฏิกริยา
- การอบให้แห้งหรือทำให้ชื้น อุปกรณ์ที่ใช้เช่น เครื่องอบ , เครื่องทำให้ชื้น
- การผสมหรือแยกสารเคมี อุปกรณ์ที่ใช้เช่น Mixer , Seperator , เครื่องกรอง

2.4.2 คุณสมบัติทั่วไปของโปรเซส

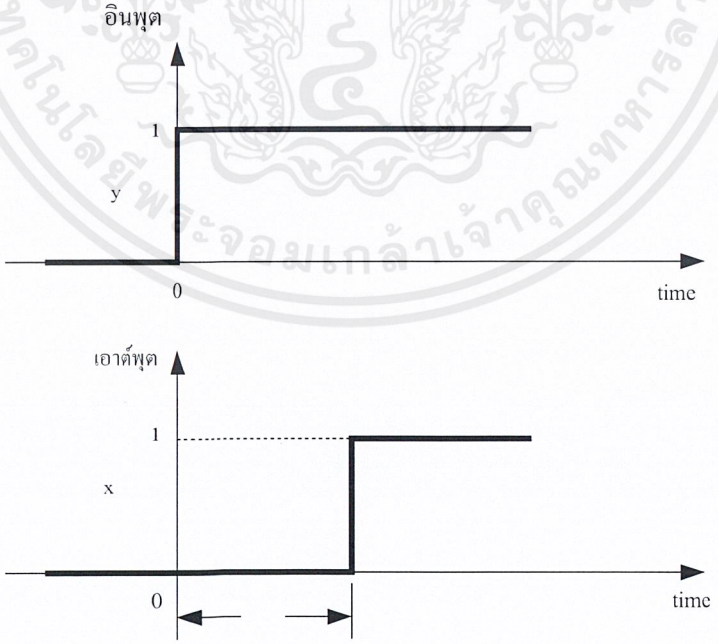
1) Dead time

เมื่อขาเข้าอินพุตของโปรเซสเปลี่ยนแปลงไปขาออกเอาต์พุตจะไม่เปลี่ยนแปลงทันที แต่จะมีเวลาสูญเปล่าช่วงหนึ่ง ก่อนที่ขาออกจะเริ่มเปลี่ยนแปลง เวลาสูญเปล่านี้เรียกว่า Deadtime โปรเซสทั่วๆ ไป จะมี Deadtime สั้นหรือยาวตามชนิดของโปรเซสที่มี Deadtime ยาวจะควบคุมได้ยากว่าโปรเซสที่มี Dead time สั้นดังรูปที่ 2.3



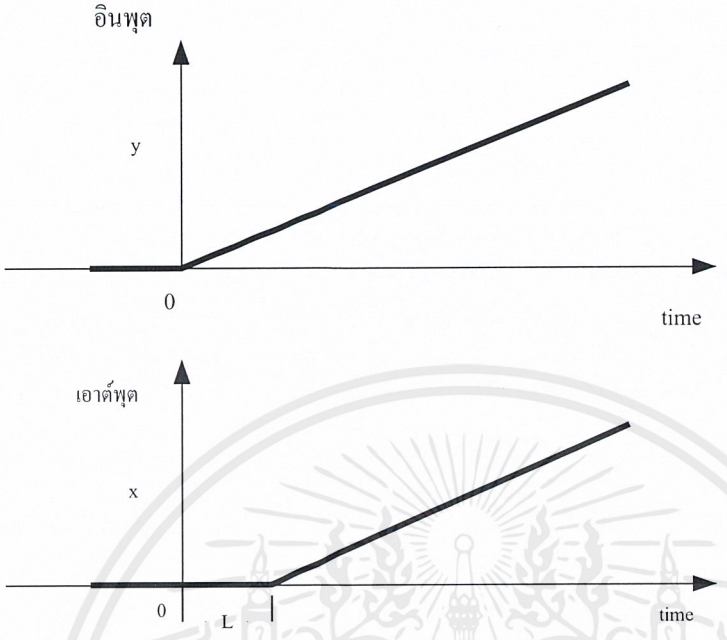
รูปที่ 2.3 ตัวอย่าง Dead time ในระบบสายพาน

Step response : ผลตอบของโปรเซสต่ออินพุตที่เปลี่ยนแปลงแบบขั้นดังรูปที่ 2.4



รูปที่ 2.4 Step respone ของโปรเซสที่มี Dead time

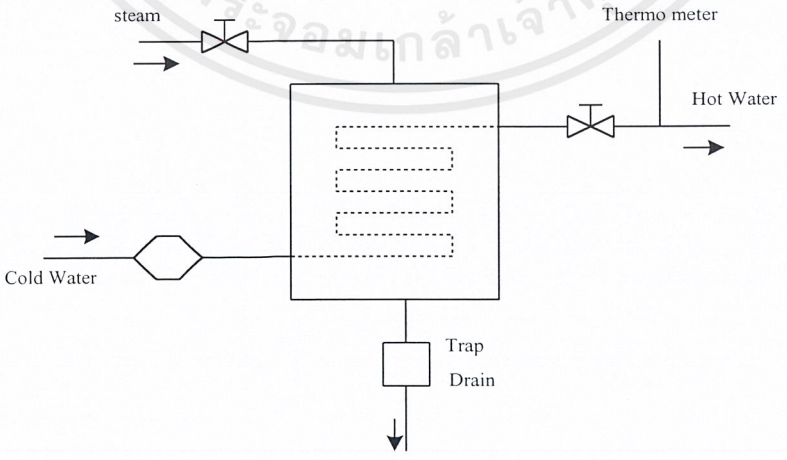
Ramp respone : ผลตอบสนองของโปรเซสเมื่ออินพุตเพิ่มขึ้นตามเวลาด้วยอัตราคงที่ดังรูปที่ 2.5



รูปที่ 2.5 Ramp response ของโปรเซสที่มี Dead time

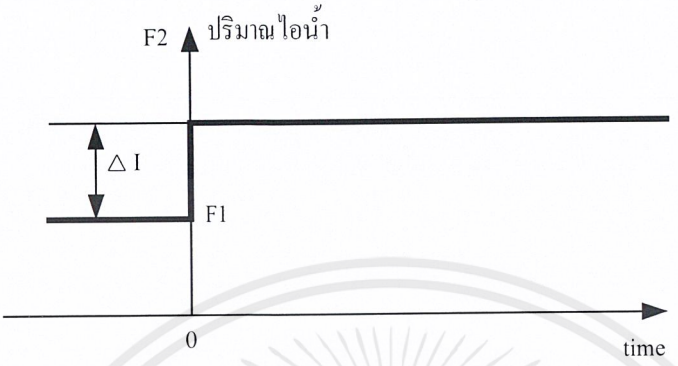
2) Time lag ในโปรเซส

ในขณะที่โปรเซสอยู่ที่จุดสมดุลย์ ถ้าเปลี่ยนค่าอินพุตของโปรเซสไปเอาต์พุตของโปรเซสจะเปลี่ยนไปตามด้วยจนกว่าจะถึงจุดสมดุลย์ใหม่ ลักษณะการเปลี่ยนแปลงของเอาต์พุตจะยังไม่มีการเปลี่ยนแปลงทันทีแต่จะค่อย ๆ เปลี่ยน ทั้งนี้เพราะมี Time lag ในโปรเซสดังรูปที่ 2.6

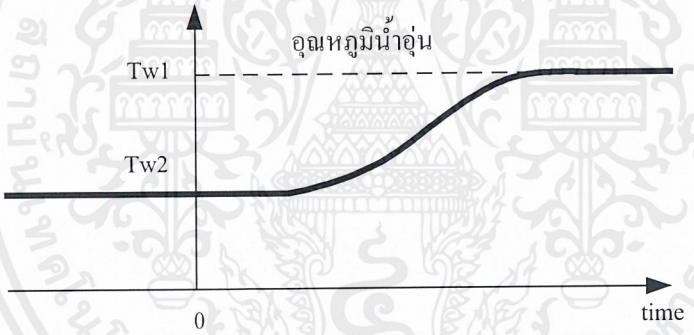


รูปที่ 2.6 Heat exchanger

ถ้าลองปรับขนาดการเปิดวาล์วให้ไอน้ำไหลเข้าใน Heat exchanger มากขึ้น อุณหภูมิของน้ำขาออกจะเปลี่ยนไปดังรูปที่ 2.7



รูปที่ 2.7 Step response ของ Heat exchanger



รูปที่ 2.8 (ต่อ) Step response ของ Heat exchanger

สาเหตุที่มี Lag ใน Heat exchanger

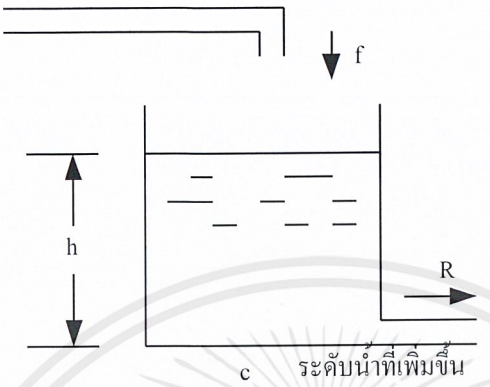
1. Heat capacitance และ Resistance ของท่อ
2. Heat capacitance และ Resistance ของน้ำ
3. Heat capacitance ระหว่าง ไอน้ำกับท่อ และท่อกับน้ำ
4. Heat capacitance และ Resistance ของเทอร์โมมิเตอร์

ชนิดของ Time lag

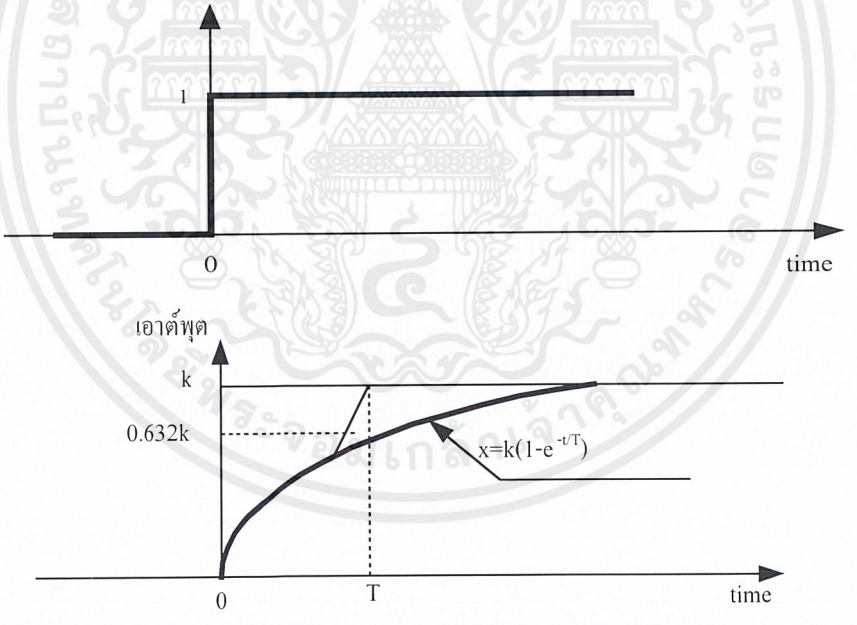
1st order lag	Lag ที่เกิดสาเหตุเดียว
n order lag	Lag ที่เกิดจาก 1st order lag n ชนิด ต่ออนุกรมกัน

High order lag Lag ที่เกิดจาก 1st order lag ต่ออนุกรมกัน

3) **ตัวอย่าง 1st order lag** ดังรูปที่ 2.9 และ 2.10



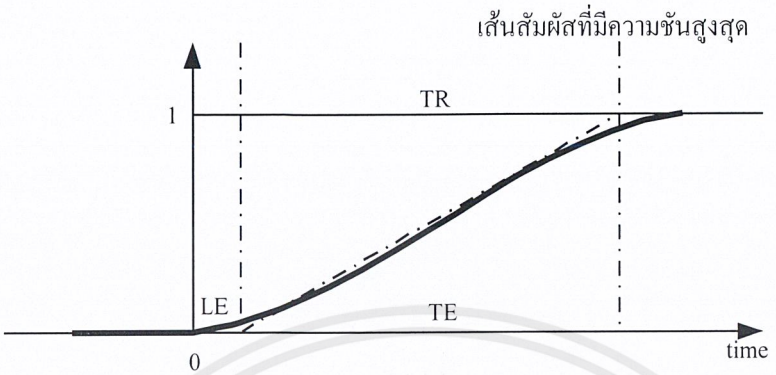
รูปที่ 2.9 ตัวอย่าง 1st order lag อินพุต



รูปที่ 2.10 ผลตอบสนองของ 1st order lag

ค่าคงตัวเวลา (Time Constant) คือ ค่าคงที่ซึ่งใช้เป็นดัชนีแสดงความเร็วในการเปลี่ยนแปลงของผลตอบสนองมีหน่วยเป็นเวลาโปรเซสที่เป็น 1st order lag นั้น เมื่อเวลาผ่านไปเท่ากับค่าคงตัวเวลานับจากอินพุตเริ่มเปลี่ยนแปลงผลตอบจะมีค่าเป็น 63.2% ของค่าสุดท้ายของผลตอบ

4) **High order** ดังรูปที่ 2.11



รูปที่ 2.11 High order lag

ผลตอบของ High order lag สามารถแทนด้วย 1st order lag รวมกับ Dead time

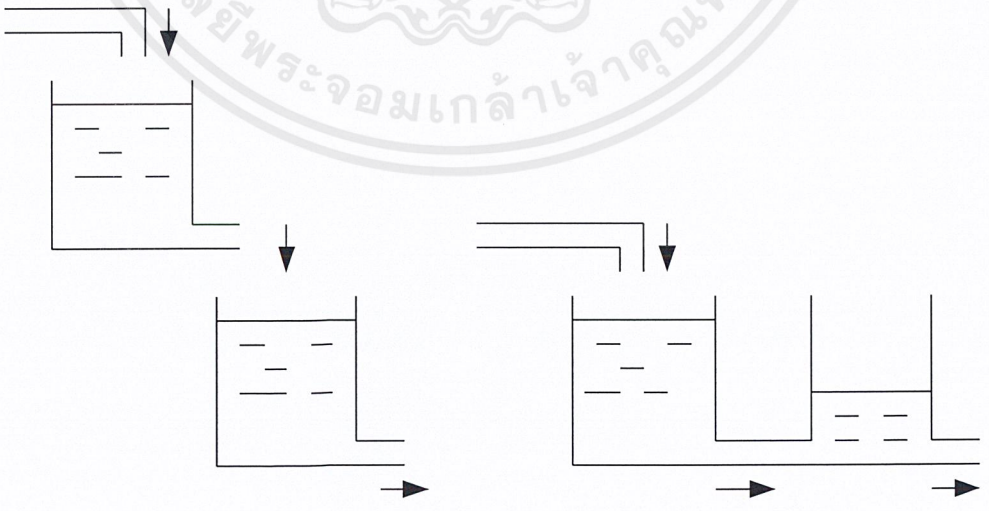
วิธีหา ให้ลากเส้นที่มีความชันสูงสุดแล้วหาช่วงเวลา

LE = Equivalent dead time

TE = Equivalent time constant

TR = Rise time

การรวมกัน (Interfere) ในระบบ High order lag ดังรูปที่ 2.12



รูปที่ 2.12 (ก) ระบบ 2nd order ที่ไม่มีการรบกวน

(ข) ระบบ 2nd แบบที่มีการรบกวนกันซึ่งกันและกัน

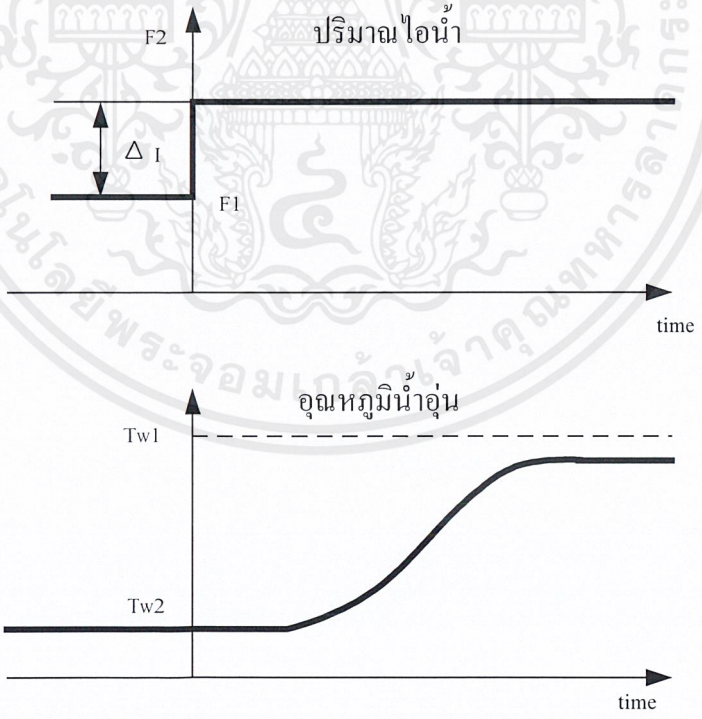
ตามปกติ 1st order lag 3 ระบบ เมื่อมาต่อกันแบบอนุกรมซึ่งไม่มีการรบกวนซึ่งกันและกัน จะได้ระบบทั้งหมด 3rd order lag ซึ่งจะมี Dead time และค่าคงตัวเวลาดังตารางที่ 2.1

ตารางที่ 2.1 LE/TE ใช้แสดงความยากง่ายในการควบคุม

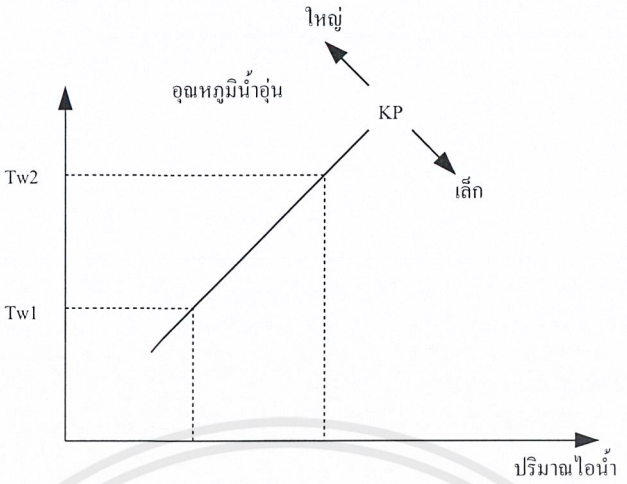
รูปแบบการควบคุม	LE	TE	LE/TE
1st order	0	T	0
2nd order	0.28 T	2.7 T	0.103
3rd order	0.81 T	3.7 T	0.128

4) Process gain

Process gain หมายถึง อัตราขยายของโปรเซส มีค่าเท่ากับอัตราส่วนของการเปลี่ยนแปลงที่เอาต์พุตต่อการเปลี่ยนแปลงที่อินพุตดังรูปที่ 2.13



รูปที่ 2.13 Static characteristic ของ Heat exchanger



รูปที่ 2.13 (ต่อ) Static characteristic ของ Heat exchanger

5) Disturbance

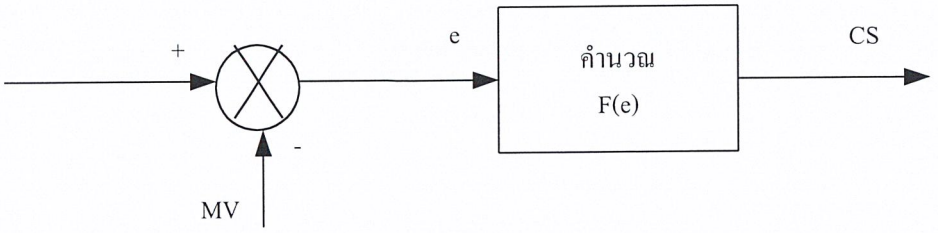
Disturbance คือ สาเหตุภายนอกที่รบกวนโปรเซสให้เปลี่ยนจุดสมดุลย์ Disturbance นี้จะมีผลทำให้ เอาต์พุต ของโปรเซสเปลี่ยนแปลงไป Disturbance ของ Heat exchanger ได้แก่

- การเปลี่ยนแปลงของอุณหภูมิไอน้ำ
- การเปลี่ยนแปลงการไหลของไอน้ำ
- อุณหภูมิรอบ ๆ เปลี่ยนแปลงไป
- การเปลี่ยนแปลงอุณหภูมิของน้ำ
- การเปลี่ยนแปลงปริมาณการใช้น้ำอุ่น (การเปลี่ยนแปลงของโหลด)

ลักษณะของ Disturbance สถานที่เกิด เวลา ขนาดและรูปร่างไม่สม่ำเสมอ อาจเป็นสัญญาณ Step , Ramp Pulse หรือ Sine wave ก็ได้

2.4.3 ตัวควบคุมและแบบการควบคุม (Controller and control action)

ตัวควบคุมจะรับสัญญาณเข้าจากการวัด เพื่อทำการเปรียบเทียบกับค่าเป้าหมาย หรือ Set point ผลต่างของค่าทั้งสองจะถูกส่งไปให้ภาคคำนวณ เพื่อผลิตสัญญาณควบคุมขาออก ซึ่งจะส่งออกไปให้ส่วนขับเคลื่อนอีกทีหนึ่ง โครงสร้างภายในตัวควบคุมมีดังรูปที่ 2.14



รูปที่ 2.14 โครงสร้างตัวควบคุม

ความสัมพันธ์ของผลต่างกับสัญญาณควบคุมขาออก สามารถจะ กำหนดโดยภาคคำนวณ ความสัมพันธ์นี้เรียกว่า Control action จะสามารถแบ่งออกได้เป็น 4 ชนิดคือ

- ON-OFF action
- Proportional ation (P action)
- Intergral หรือ Reset action (I action)
- Derivation action (D action)

การควบคุมจะใช้ action แต่ละชนิดหรือหลายชนิดผสมกัน เช่น ON-OFF P , PI, PD, PID การควบคุมด้วยมือนั้นถ้าคนงานมีความชำนาญงาน การปรับจะมีลักษณะคล้ายคลึง Contrpl action เหล่านี้

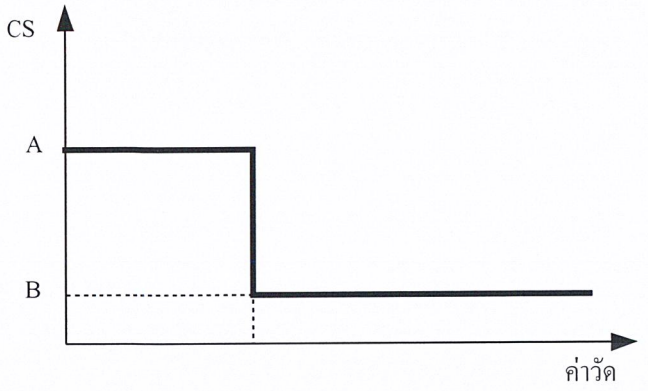
2.4.4 ON-OFF Control (2 – Position action)

ON-OFF action คือ action ที่ทำให้ความสัมพันธ์ของผลต่าง กับสัญญาณควบคุม (CS) ดังนี้

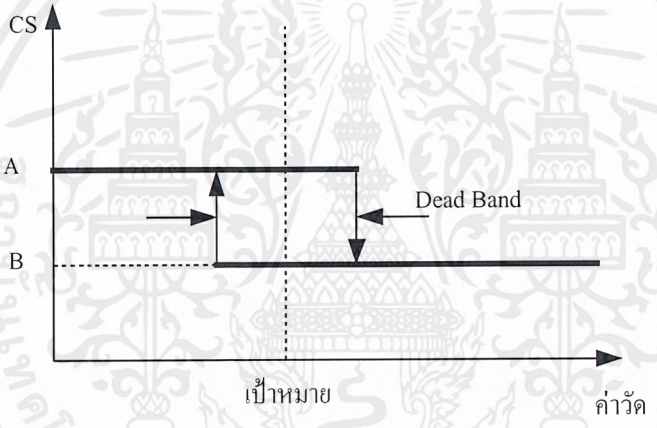
เมื่อ error > 0 , CS = A

เมื่อ error < 0 , CS = B

โดยที่ A และ B คือค่าใด 2 ค่าของ CS หรือสัญญาณควบคุม ดังนั้นการควบคุมแบบนี้ CS จะมีค่าเพียง 2 ค่าเท่านั้น บางครั้งจึงเรียกว่า 2- Position action เช่น ให้วาล์วเปิดเต็มที่หรือปิดเต็มที่ เป็นต้น จากความสัมพันธ์จะเห็นว่าเมื่อค่าวัด (PV) มีขนาดน้อยกว่าค่าเป้าหมาย (SP) ผลต่างจะเป็นบวก สัญญาณควบคุมจะมีค่าหนึ่ง (A หรือ B) ซึ่งจะทำให้โปรเซสเปลี่ยนไปในทางที่ค่าวัดจะสูงขึ้น ยกตัวอย่างเช่น เมื่ออุณหภูมิ ของน้ำร้อนที่ออกจาก Heat exchanger สูงเกิดกว่าค่าเป้าหมาย วาล์วจะถูกควบคุม ให้ปิดเมื่ออุณหภูมิของน้ำร้อนลดลง แต่ถ้าลดกว่าค่าเป้าหมายอีกก็ให้เปิดวาล์ว เพื่อให้อุณหภูมิสูงขึ้นมาดังรูปที่ 2.15



รูปที่ 2.15 การควบคุมแบบ ON-OFF

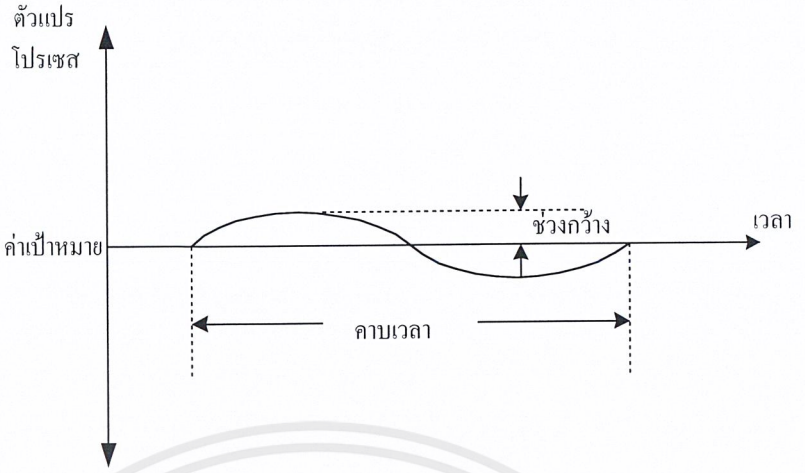


รูปที่ 2.16 (ต่อ) การควบคุมแบบ ON-OFF

ตามปกติการควบคุมแบบ ON-OFF จะมี Dead band ซึ่งเป็นอาณาเขตของค่าเป้าหมายซึ่งจะยอมให้ค่าวัดผิดพลาดไปจากค่าเป้าหมายได้บ้างก่อนที่จะมีการเปลี่ยนแปลงสัญญาณควบคุมดังรูปที่ 2.16

การควบคุมแบบ ON-OFF เป็นการเปิดปิดพลังงานที่เข้าโปรเซสเป็น (Cycling) ขึ้นได้ ช่วงกว้าง (Amplitude) และคาบเวลา (Period) ในการแกว่งจะขึ้นอยู่กับลักษณะคุณสมบัติของโปรเซส เช่น Dead time, ค่า Resistance หรือ Capaitance

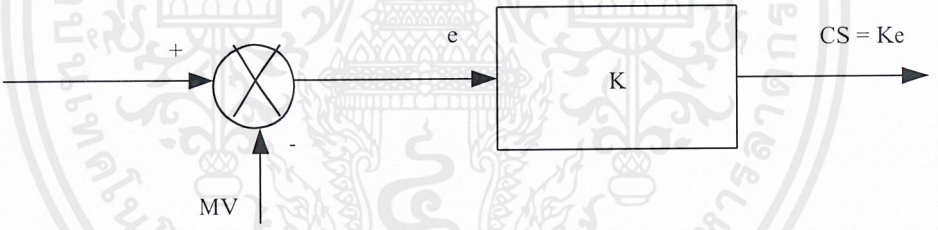
การควบคุมแบบ ON-OFF จะได้ผลต่อเมื่อโปรเซสนั้นมีค่าคงตัวเวลายาวมากแต่มี Dead time สั้นดังรูปที่ 2.17



รูปที่ 2.17 การแกว่งของผลของการควบคุม

2.4.5 Proportional action

Proportional action คือ ช่วงที่ทำให้สัญญาณควบคุมแปรผันโดยตรงกับผลต่างดังรูปที่ 2.18

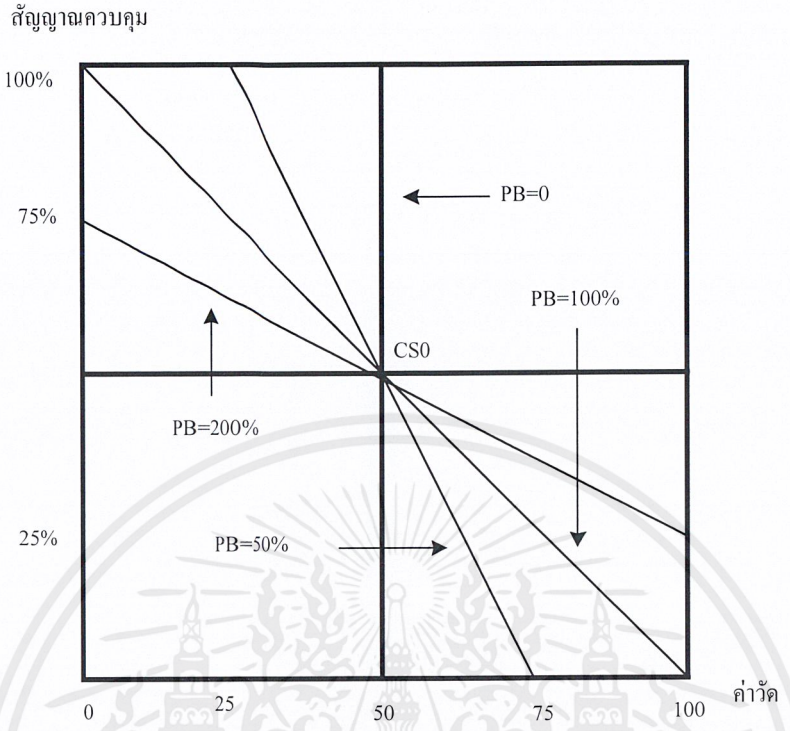


รูปที่ 2.18 โครงสร้างการควบคุมแบบ Proportional Control

K คือ Proportional gain สัญญาณควบคุมก็คือ การขยายผลต่างด้วยอัตราขยาย K นั้นเอง ตามปกติในระบบการควบคุมมักจะไม่นิยมใช้ K แต่จะใช้ Proportional Band (PB) แทนโดยที่

$$PB = 1/K*100\%$$

PB มีความหมายคือ ผลต่าง ( ) จะต้องเปลี่ยนไป PB% จึงจะพอดีทำให้สัญญาณควบคุมเปลี่ยนจาก 0 ถึง 100% ดังรูปที่ 2.19



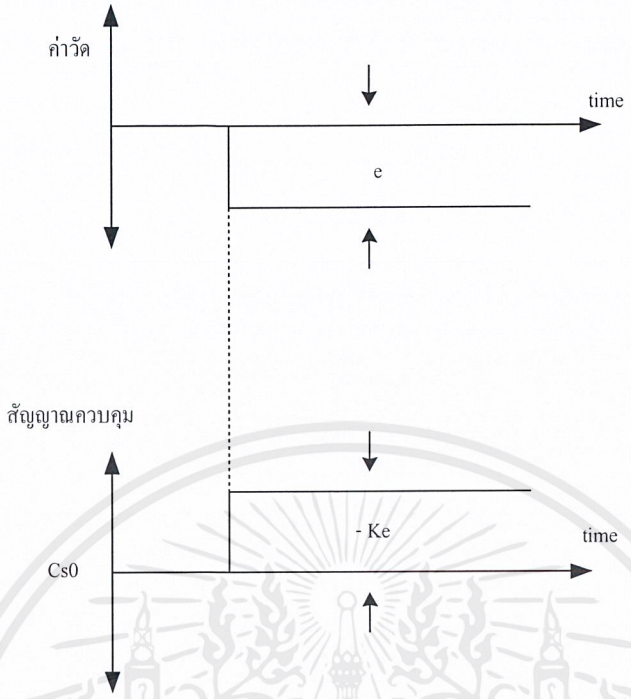
รูปที่ 2.19 ความหมายของ Proportional band

จะเห็นได้ว่าเมื่อ PB มีค่าเล็กลง อัตราขยายสูงขึ้น และเมื่อ PB = 0% อัตราขยายเป็นอนันต์ การทำงานของ Proportional action จะกลายเป็น ON-OFF

ดังในรูปที่ 2.19 จะเห็นได้ว่าเมื่อเป้าหมายอยู่ที่ 50% และค่าวัดได้ 50% ผลต่างจะเท่ากับ 0 ขณะนั้นสัญญาณควบคุมจะมีค่า CS0 ดังนั้น สัญญาณควบคุมจะถูกกำหนดจากความสัมพันธ์ดังนี้

$$CS = -K + CS0$$

การที่จำเป็นต้องมี CS0 หรือที่เรียกว่า ไบแอส นั้น เพื่อให้ผลต่างเท่ากับศูนย์ สัญญาณควบคุม CS จะได้ไม่เท่ากับศูนย์ไปด้วย เพราะจะทำให้ควบคุมโปรเซสไม่ได้ดังรูปที่ 2.20



รูปที่ 2.20 ผลการตอบสนอง Proportional action

คุณสมบัติของ ผลการตอบสนอง Proportional action

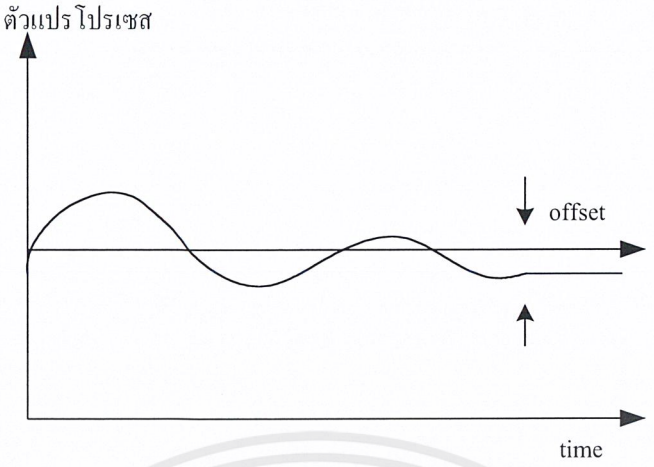
1. เมื่อลด PB ลงไป ทำให้อัตราขยายสูงขึ้น การควบคุมจะไว (Sensitive) ขึ้นทำให้โปรเซสเกิดการแกว่งขึ้นได้

2. เมื่อเพิ่มค่า PB ขึ้น อัตราขยายลดลง จะทำให้ค่าวัดกับค่าเป้าหมายแตกต่างกันมากขึ้น เราเรียกว่าเกิด offset

3. เมื่อโหลดของโปรเซส สภาพแวดล้อมเปลี่ยนไป ซึ่งเราเรียกว่ามี Disturbance ก็จะเกิด offset ขึ้นได้

2.4.6 Offset

Offset เป็นชื่อใช้เรียกปรากฏการณ์ที่ตัวแปรโปรเซสหรือค่าวัด มีค่าไม่เท่ากับค่าเป้าหมาย ทำให้การควบคุมไม่เป็นไปตามที่ต้องการ รูปที่ 2.21 แสดงความหมายของ Offset



รูปที่ 2.21 Offset

ในระบบการควบคุมแบบป้อนกลับนี้เมื่อดูอย่างผิวเผิน Offset ไม่น่าที่จะเกิดขึ้น ได้เพราะตัวแปรโปรเซสจะถูกป้อนกลับมาเปรียบเทียบกับค่าเป้าหมายตลอดเวลา อย่างไรก็ตาม Offset มักจะเกิดในระบบการควบคุมที่ใช้ Proportional Control ที่มีค่า PB ใหญ่ และขณะที่เกิด Disturbance ในโปรเซสซึ่งได้แก่การเปลี่ยนแปลงของโหลด สภาพแวดล้อม เป็นต้น

วิธีการแก้ Offset ให้เก็บลง

- ให้ลด PB ให้เล็กลง
- เปลี่ยนไบแอสของการควบคุมด้วยมือ
- เปลี่ยนค่าเป้าหมาย
- ใช้การควบคุมที่มี Reset action

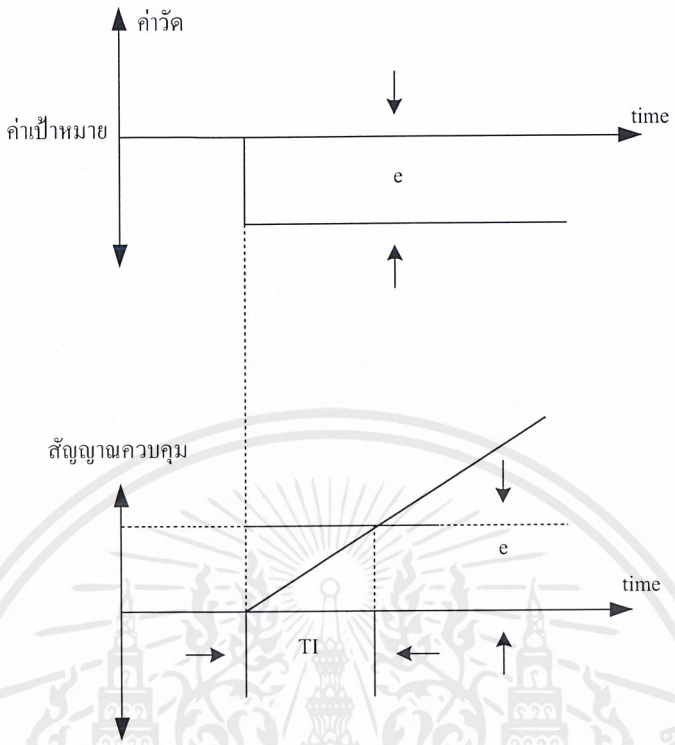
2.4.7 Reset action (Integral action)

Reset action คือ action ที่ทำให้เกิดความสัมพันธ์ดังนี้

$$CS = \frac{1}{T} \bullet \int \in dt$$

โดย T_I คือ Reset time มีหน่วยเป็น นาที

การควบคุมแบบนี้สัญญาณควบคุมจะแปรตามค่าอินทิเกรทต่อเวลาของผลต่าง เมื่อผลต่างเปลี่ยนเป็น Step สัญญาณควบคุมจะค่อย ๆ เพิ่มค่าขึ้นตามเวลา ดังรูปที่ 2.22

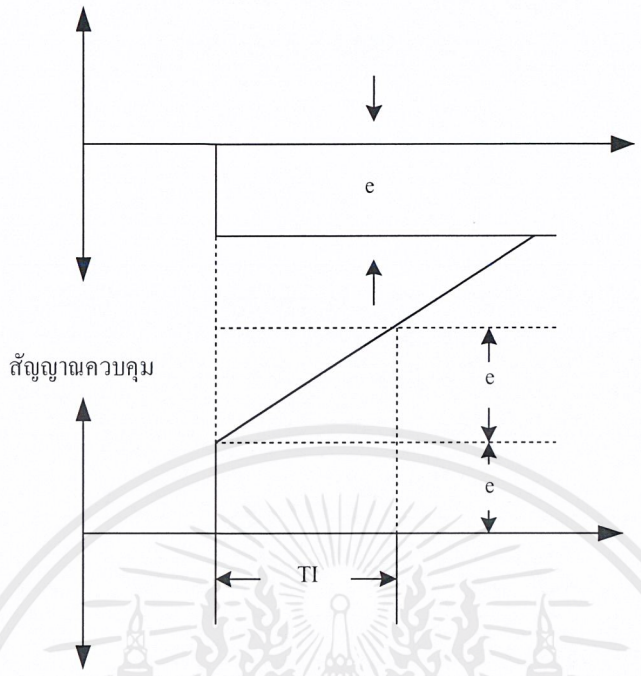


รูปที่ 2.22 ผลตอบของ Reset action

ตามรูปที่ 2.22 ความหมายของ Reset time (T_I) คือเวลาที่ใช้ไปในการเพิ่มค่าของสัญญาณควบคุมจนมีขนาดเท่าผลต่าง ที่เปลี่ยนแปลงไป ดันนั้นเมื่อ T_I มีค่าน้อย ผลของ Reset action จะมากกว่าเมื่อ T_I มีค่ามาก

2.4.8 PI action

ตามปกติ Reset action มักจะไม่ใช้เพียงอย่างเดียว แต่จะใช้ควบคู่กับ P เสมอ ผลตอบของ PI action แสดงในรูปที่ 2.23



รูปที่ 2.23 ผลตอบของ PI action

คุณสมบัติของ Reset action

- offset ที่เกิดจาก P action จะถูก Reset action แก้จนหมดไป
- เมื่อ T_I มีค่าน้อยลง ผลของ Reset action จะมากทำให้เกิดการแกว่งขึ้นได้ซึ่งจะทำให้ระบบขาดเสถียรภาพ

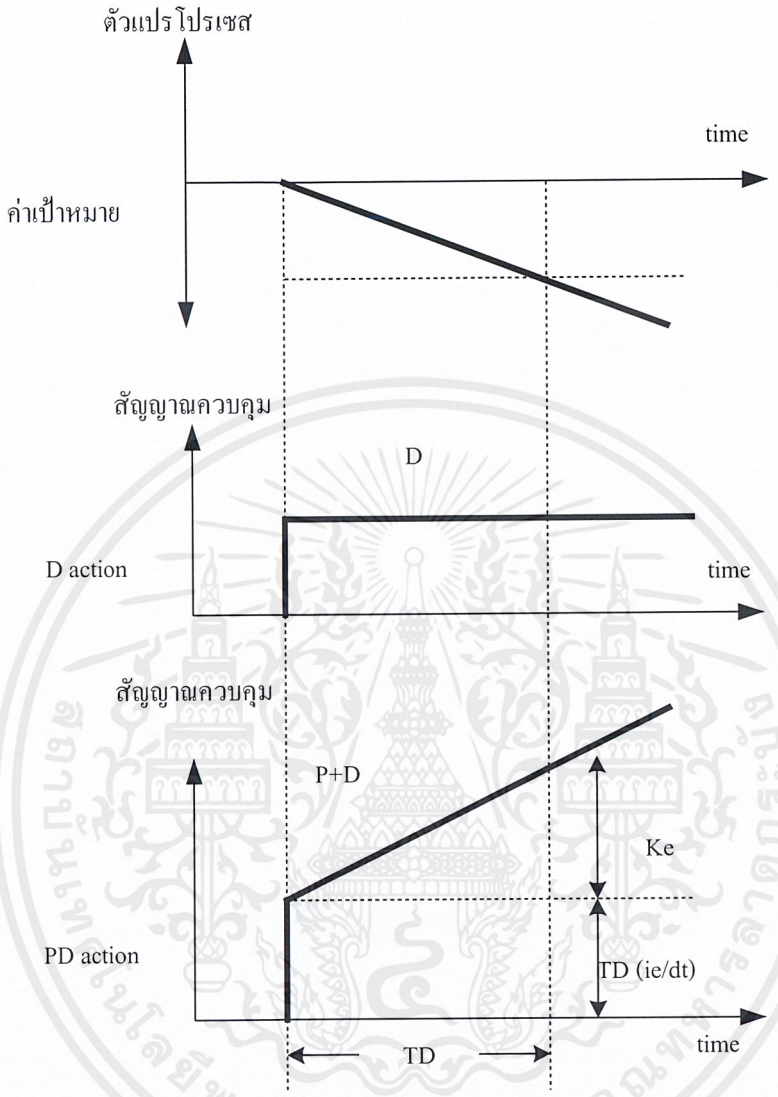
2.4.9 Derivative action (Rate action)

Rate Action ที่ทำให้ความสัมพันธ์ของสัญญาณควบคุมกับผลต่างเป็นไปตามสูตร

$$CS = Td\frac{d \in}{dt}$$

โดยที่ T_D คือ Derivative time มีหน่วยเป็นนาที

สัญญาณควบคุมจะแปรผันตามอัตราการเปลี่ยนต่อเวลาของผลต่าง เมื่อตัวแปรโปรเซสเปลี่ยนไปด้วยอัตราหนึ่งสัญญาณควบคุมจะถูกผลิตให้แปรผันกับอัตราการเปลี่ยนแปลงนี้ จะถูกสร้างขึ้นเพื่อไปหยุดการเปลี่ยนแปลงนั้น D action เหมาะสำหรับไปใช้กับโปรเซสที่มี Time lag มาก ๆ ผลตอบของ D action และ PD action แสดงในรูปที่ 2.24



รูปที่ 2.24 D action และ PD action

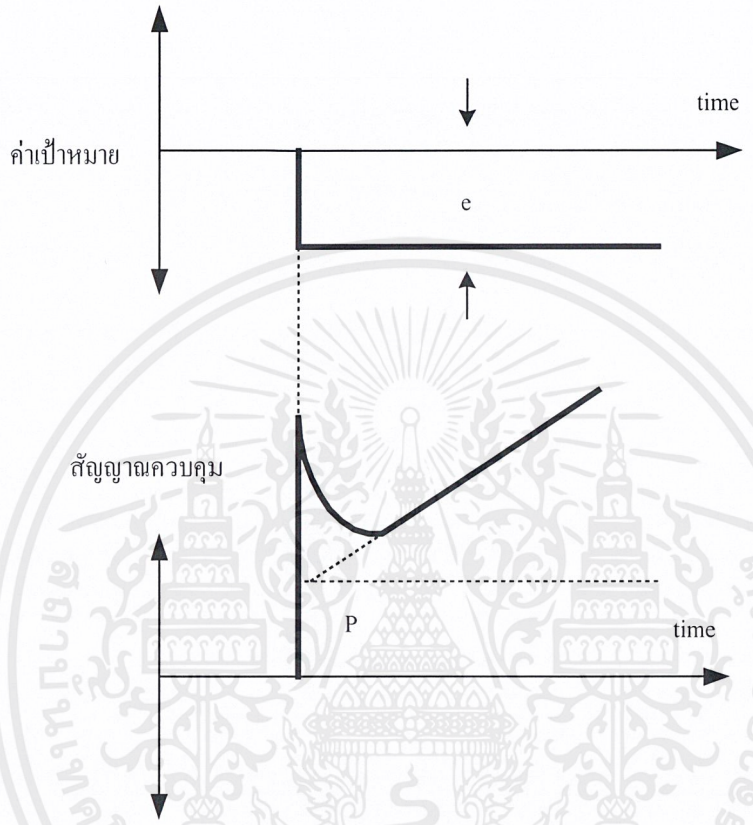
จากรูปที่ 2.24 จะเห็นว่า T_D จะเท่ากับช่วงเวลาที่ผลต่าง $\in$ เปลี่ยนแปลงไปจนมีค่าเท่ากับค่าความชันของการเปลี่ยนแปลงนี้ ถ้า T_D มากจะทำให้ผลของ D action มาก

คุณสมบัติของ D action

1. เหมาะสำหรับโปรเซสที่มี Time lag จะช่วยทำให้การควบคุมถึงจุดที่ต้องการเร็วขึ้น
2. ถ้า TD มากไปผลของ D จะมากทำให้ระบบทั้งระบบไวขึ้น ขาดเสถียรภาพ
3. ไม่เหมาะสมสำหรับโปรเซสที่มี Time lag น้อยและตัวแปรโปรเซสเปลี่ยนแปรได้ง่าย เช่น ระบบควบคุมการไหล ความดัน เป็นต้น

2.4.10 PID action

ได้แก่การรวม P , I, D action เข้าด้วยกันตามรูปที่ 2.25

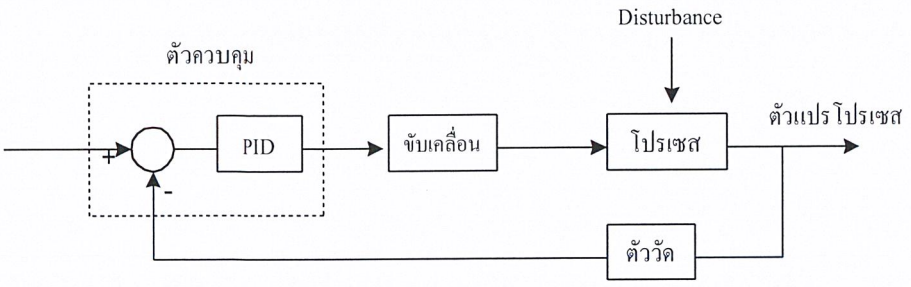


รูปที่ 2.25 PID action

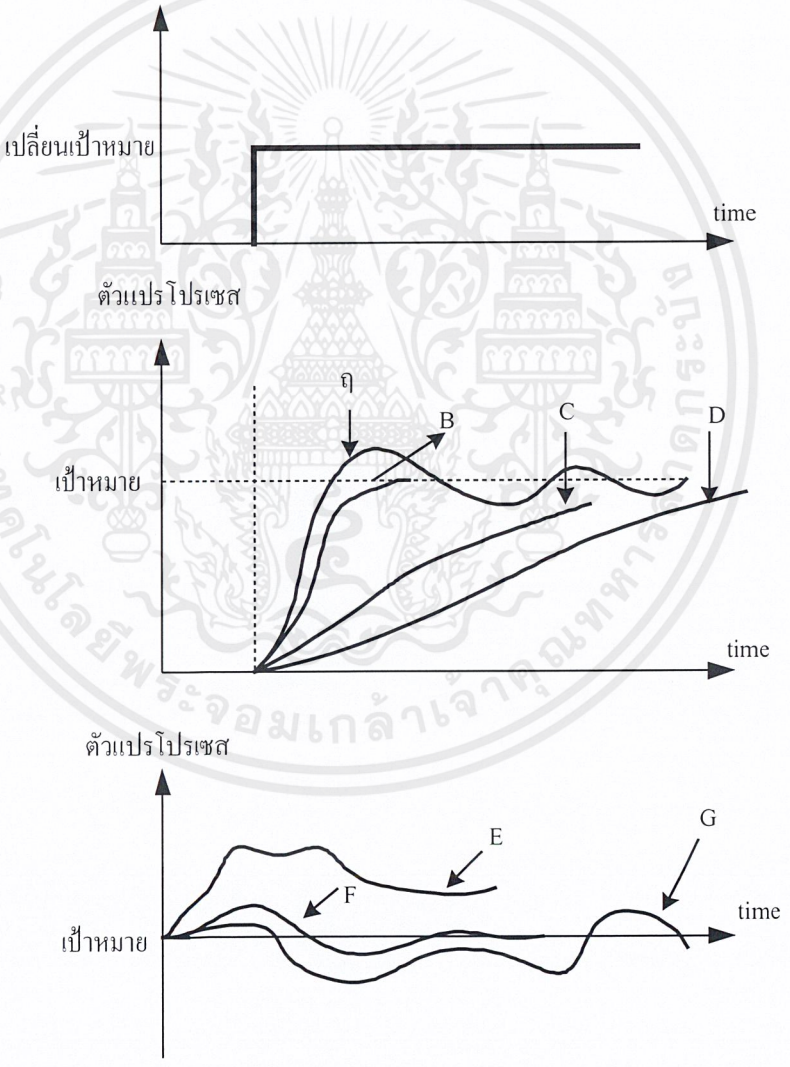
2.5 การปรับ PID

2.5.1 ผลการตอบสนองที่ดี

ในระบบควบคุมแบบป้อนกลับ ตัวควบคุมจะพยายามรักษาให้ค่าตัวแปรโปรเซสมีค่าเท่ากับค่าเป้าหมายอยู่เสมอ ในกรณีที่เกิด Disturbance ในระบบหรือมีการเปลี่ยนค่าเป้าหมายจะทำให้ตัวแปรโปรเซส มีค่าต่างจากค่าเป้าหมายไปขณะหนึ่ง ตัวควบคุมจะพยายามควบคุมให้ตัวแปร โปรเซสมีค่าเท่ากับค่าเป้าหมายนี้ในที่สุด ลักษณะการนำค่าตัวแปรโปรเซสให้เข้าใกล้ค่าเป้าหมายนี้ จะแตกต่างกันตามคุณสมบัติของระบบควบคุม บางระบบควบคุมอาจควบคุมให้ตัวแปรโปรเซสส่งเข้าหาค่าเป้าหมายได้เร็ว แต่บางระบบอาจทำได้ดีกว่า เราสามารถทดสอบความสามารถของระบบควบคุมนี้โดยดูที่ผลตอบของการควบคุม



รูปที่ 2.26 ระบบควบคุมแบบป้อนกลับ



รูปที่ 2.27 ผลตอบของระบบควบคุมต่าง ๆ

A มี Overshoot และการแกว่ง

B ตอบรับเร็ว

C ตัวแปรโปรเซสมีค่าไม่เท่ากับค่าเป้าหมายแม้เวลาผ่านพ้นไปนาน เรียกว่า Offset

D ตอบรับช้ามาก

เมื่อตัวแปรโปรเซสหยุดนิ่งมีค่าเป้าหมาย ในขณะนั้นเกิดมี Disturbance เข้ามารบกวนโปรเซส (Disturbance ได้แก่ การเปลี่ยนของโหลด การเปลี่ยนแปลงของสภาพแวดล้อม เป็นต้น) ผลตอบของโปรเซส จะมีหลายแบบดังแสดงในรูป

E เกิด offset ได้ค่าผิดไปจากค่าเป้าหมายเดิม

F เกิดการแกว่งเล็กน้อย ก่อนกลับเข้าสู่ค่าเป้าหมายเดิม

G ไม่เกิด offset

เมื่อพิจารณาผลตอบของการควบคุมชนิดต่าง ๆ เหล่านี้พอสรุปได้ว่า ระบบการควบคุมที่ดีจะมีคุณสมบัติดังนี้

ก) มีเสถียรภาพ ไม่เกิดการแกว่ง (Oscillation) เมื่อถูกกระตุ้น

ข) ตอบรับการเปลี่ยนค่าเป้าหมาย หรือ Disturbance ได้รวดเร็ว

ค) ไม่เกิด offset

2.5.2 ความยากง่ายในการควบคุมของโปรเซส

" Dead Time " เป็นศัตรูตัวร้ายของการควบคุมลักษณะสมบัติของโปรเซสทั่ว ๆ ไป จะมี Time lag ดังที่ได้กล่าวมาแล้วจากรูปคลื่นผลตอบของโปรเซสต่อ Step อินพุต เราสามารถหาค่า Dead time (LE) และค่าคงตัวเวลา (TE) โดยประมาณได้จะเป็นค่าที่ใช้ประเมินความยากง่ายในการควบคุมและใช้เลือกแบบการควบคุม

ตารางที่ 2.2 การเลือกใช้การควบคุมที่เหมาะสมกับโปรเซส

$LE / _{TE}$	แบบการควบคุมที่เหมาะสมกับโปรเซสนั้น
$LE / _{TE} < 0.2$	ON – OFF , P ,PI
$0.2 < LE / _{TE} < 0.2$	PI ,PID
$1.0 < LE / _{TE}$	Feed forward, Computer Control

$LE / _{TE} < 1.0$ สามารถใช้การควบคุมแบบ PID ได้ตามปกติ $0.5 < LE / _{TE} < 1.0$ การควบคุมแบบ PID ก็ควบคุมได้ยากมาก

1. สาเหตุของการขาดเสถียรภาพในระบบ

- ผลของระบบควบคุมข้างเดียว หรือ อื่น ๆ
- Disturbance เป็น periodic
- Process gain หรือ LOOP gain มีค่าสูงเกินไป
- คุณสมบัติของตัวแปรในระบบไม่คงที่แปรเปลี่ยนตาม Disturbance, ค่าเป้าหมาย, เวลา เป็นต้น

2.5.3 ผลของ PID ต่อเสถียรภาพของระบบ

ในระบบควบคุมแบบป้อนกลับ ซึ่งใช้ตัวควบคุมแบบ PID นั้น ถ้าเราลองแปรค่า PE , T_I และ T_D จะมีผลต่อผลตอบของระบบควบคุมดังนี้

ผลของ P action

เมื่อลดค่า PB ลงทำให้อัตราขยายสูงขึ้น จะมีผลทำให้

ก) Offset ลดลง

ข) Period ของการแกว่งเล็กลง

ค) อัตราส่วนของช่วงกว้างการแกว่งเพิ่มขึ้น ระบบขาดเสถียรภาพมากขึ้น

ผลของ I action

เมื่อให้ P และ D action คงที่แล้ว ลองลด T_I (Reset time) จะมีผลทำให้

ก) Offset จะหายไป

ข) ผลตอบจะเร็วขึ้น (Fast response)

ค) อัตราส่วนของช่วงกว้างการแกว่งเพิ่มชั้น ระบบขาดเสถียรภาพมากขึ้น

ผลของ D action

เมื่อให้ P และ I action คงที่แล้วเพิ่มเวลา T_D (Derivative time) ให้ยาวขึ้นจะมีทำให้

ก) อัตราส่วนของช่วงกว้างการแกว่งลดลง ระบบมีเสถียรภาพมากขึ้น

ข) period ของการแกว่งสั้นลง

ตามปกติการใช้ค่าเวลา T_D ให้ยามมีแนวโน้มที่จะทำให้ระบบมีเสถียรภาพมาก็จริง แต่ก็มีจุดอ่อนตรงตอบรับต่อ Noise ได้ง่าย ทำให้ผลตอบของระบบไวเกินไป ผลตอบของการควบคุมแบบต่าง ๆ ต่อการเปลี่ยนค่าเป้าหมายและการเกิด Disturbance โดยตั้งค่า PB , T_I และ T_D

2.5.4 คุณภาพของการควบคุม

ดังที่ได้กล่าวมาแล้วว่า ระบบควบคุมที่ดีจะต้องมีคุณสมบัติคือ มีเสถียรภาพ ผลตอบเร็ว และไม่เกิด Offset แต่จะใช้อะไรในการตัดสินว่า การควบคุมไหนจะดีที่สุดนั้น จะต้องมีเกณฑ์ตัดสินที่มีหลักเกณฑ์

1. เกณฑ์ในการวัดที่ใช้ตัดสินคุณภาพการควบคุม มีดังนี้

ก) อัตราส่วนช่วงกว้างการแกว่ง (Damping ratio criterion)

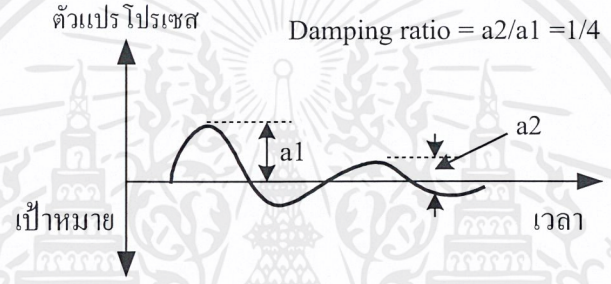
ข) พื้นที่การควบคุมน้อยสุด (Minmum control area criterion)

ค) ความสูงของ Overshoot แรก (Overshoot amplitude criterion)

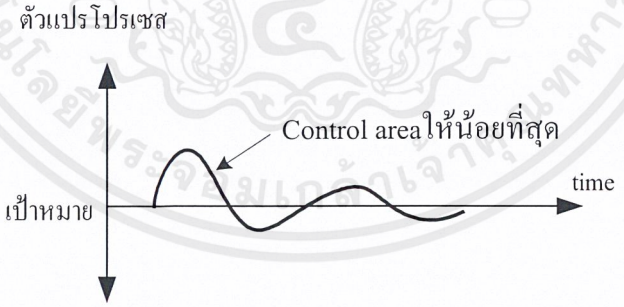
ง) ผลตอบต่อ Disturbance (Minimum disturbance criterion)

จ) ผลตอบเชิงความถี่ของระบบ (Frequency response criterion)

ในการควบคุมโปรเซสเกณฑ์อัตราส่วนช่วงกว้างการแกว่ง เป็นที่นิยมใช้มากที่สุด เพราะสามารถวัดผลได้ง่าย การควบคุมที่ดีจะให้อัตราส่วนช่วงกว้างการแกว่ง 25% หรือ 1 ต่อ 4 ตามรูปที่ 2.28 และ 2.29



รูปที่ 2.28 25% Damping ratio



รูปที่ 2.29 พื้นที่การควบคุมน้อยสุด

2.5.5 วิธีการปรับ

ได้มีนักคณิตศาสตร์คิดวิธีที่จะหาทางตั้งค่า PID เพื่อให้ได้การควบคุมที่มีคุณภาพดีที่สุดในหลายวิธี ดังตัวอย่างที่แสดงในตารางที่ 2.3 อย่างไรก็ตามในทางปฏิบัติก็ยังไม่มีวิธีใดที่ดีพอที่สามารถจะนำไปใช้ในทุกระบบได้

ในทางปฏิบัติวิธีที่นิยมใช้ในการปรับค่า PID สำหรับการควบคุมเพื่อให้ได้ 25% Damping ratio มี 3 วิธี คือ

1. Reaction Curve method
2. Ultimate Sensitivity method
3. Trial and error

ตารางที่ 2.3 การปรับอัตราของ PID

(Trans,ASME, etc.)

ชื่อผู้เสนอ	แบบ	แบบการควบคุม	Control Action			เกณฑ์
			PB(%)	T_D (นาที)	T_D (นาที)	
Zicglor Nichols 1942	A.B.	P	2 PB	-	-	25% Damping Ratio
		PI	2.2 PB^U	0.83 pu	-	
		PID	1.7 PB^U	0.5 pu	0.125 pu	
Zicglor Nichols 1942	A.B.	P	100 KpL / T	-	-	25% Damping Ratio
		PI	110 KpL / T	33 L	-	
		PID	83 KpL / T	2 L	0.5 L	
lakahashi	A	P	101 KpL / T	-	-	Minimum Control area
		PI	111 KpL / T	3.3 L	-	
		PID	77 KpL / T	2.2 L	0.45 L	
Chien Hrones Reswick	A	P	333 KpL / T	-	-	20% Overshoot
		PI	286 KpL / T	1.2 T	-	
		PID	167 KpL / T	T	0.5 L	

ตารางที่ 2.3 (ต่อ) การปรับอัตราของ PID

Chien Hrones Reswick	A	P PI PID	143 KpL / T 167 KpL / T 105 KpL / T	- T 1.35 T	- - 0.47 L	 No Overshoot
Chien Hrones Reswick	B	P PI PID	333 KpL / T 167 KpL / T 105 KpL / T	- 4L 2.4L	- - 0.4 L	 20% Overshoot
Chien Hrones Reswick	B	P PI PID	143 KpL / T 143 KpL / T 83 KpL / T	- 2.3L 2L	- - 0.42 L	Minimum Control area
Fuji Yoshikawa	A	P PI L/T 1 L/T 1 PID L/T = 1 L/T = 1	100 KpL / T 167 KpL / (T+L) 250 KpL / (T+2L)	- T+L 0 L 0.5 (T+L) 0.25 L	- - - 0.125 (T+L)	

1) Reaction Curve method (Transient response method)

<u>วิธีการ</u>

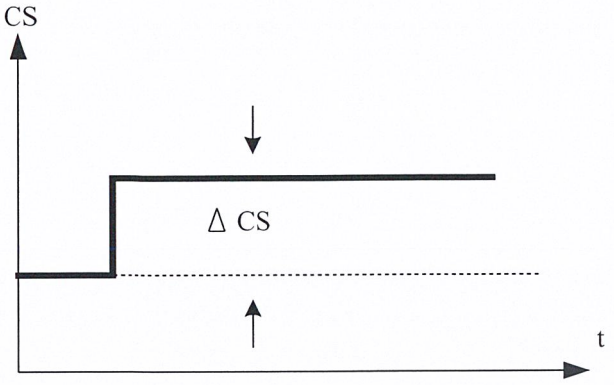
- ให้เปลี่ยนระบบควบคุมเป็นวงรอบเปิด (Open Loop)

- หา Process Characteristic โดยเปลี่ยนค่าสัญญาณควบคุมไป ΔCS แล้วบันทึกรูปคลื่นของตัวแปรโปรเซส

- หา Process gain (Kp), dead time (LE) และค่าคงตัวเวลา (TE) จาก Process Characteristic จากรูปที่ 2.30

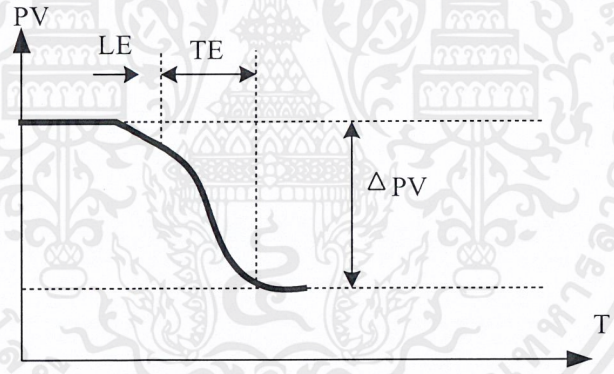
$$KP = \frac{\Delta PV}{\Delta CS}$$

- นำค่า Kp, LE, TE ที่หาได้ไปคำนวณหา PB, T_I และ T_D จากตารางที่ 2.3



(ก)

$$KP = \frac{\Delta PV}{\Delta CS}$$



(ข)

รูปที่ 2.30 (ก) แสดงการหา ΔCS Reaction curve method

(ข) แสดงการหา KP Reaction curve method

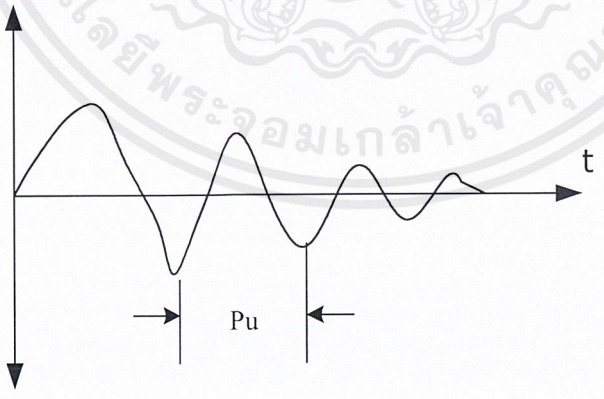
ตารางที่ 2.4 Reaction Curve method

ชนิดการควบคุม	PB(%)	T_I (นาที)	T_D (นาที)
P	100 Kp. LE / TE	α	0
PI	110 Kp. LE / TE	3.3 LE	0
PID	83 Kp. LE / TE	2 LE	0.5 LE

2) Ultimate sensitivity method

<u>วิธีการ</u>

- ให้ระบบควบคุมเป็นแบบวงรอบปิด (Closed loop)
- ตั้ง T_I สูงสุด (max) และ T_D ต่ำสุด (0 หรือ min) ใช้ P action ในการควบคุมอย่างเดียว
- ครั้งแรกตั้งค่า PB ไว้ที่ค่าสูงสุด แล้วลองลดค่า PB ลงมา ลองเปลี่ยนค่าเป้าหมายเพื่อผลตอบ ลดค่า PB ให้ต่ำลงเรื่อย ๆ จนถึงค่าที่เมื่อเปลี่ยนค่าเป้าหมายไปเล็กน้อย จะทำให้โปรเซสเกิดการแกว่งต่อเนื่องไปตลอดค่า PB ในขณะนั้นเรียกว่า PB_u (Ultimate proportional band)



รูปที่ 2.31 การวัด Pu

ตารางที่ 2.5 Uitimate Sensitivity method

ชนิดการควบคุม	PB	T_I	T_D
P	2 PB_u	α	0
PI	2.2 PB_u	0.83 Pu	0
PID	1.7 PB_u	0.5 Pu	0.125 Pu

- หาคาบเวลาในการแกว่ง ให้เท่ากับ Pu (ตามรูปที่ 2.31)
- นำค่า PB_u และ Pu ที่หาได้ไปใช้คำนวณหาค่า PB , T_I และ T_D จากตารางที่ 2.5

3) Trial and error

เป็นวิธีการที่ใช้หลักการของ Ultimate Sensitivity method เพียงแต่ไม่ต้องคำนวณค่าและใช้การทดลองปรับค่าต่าง ๆ เพื่อหาผลตอบที่ดีที่สุด

ก) P Control

1. ปรับตัวควบคุมไปที่ Manual mode
2. ปรับ PB ไปสูงสุด T_I สูงสุด และ T_D ต่ำสุด
3. ปรับค่าเป้าหมาย (Set point) ไปสู่ค่าที่ต้องการ
4. ปรับ Manual Control จนตัวแปรโปรเซสหรือค่าวัดได้เท่ากับค่าเป้นหมาย
5. ปรับตัวควบคุมไปที่ Automatic mode
6. เปลี่ยนค่าเป้าหมายไปเล็กน้อย เมื่อค่าวัดเริ่มเปลี่ยน จึงลดค่าเป้าหมายกลับมาสู่ที่เดิม
7. ลดค่า PB ลงมา และทำขั้น 6 ใหม่ โดยสังเกตผลตอบของค่าวัด
8. ทำขั้น 6 และ 7 หลาย ๆ ครั้งจนได้อัตราส่วนช่วงกว้างการแกว่งของผลตอบเป็น 25% Damping ratio

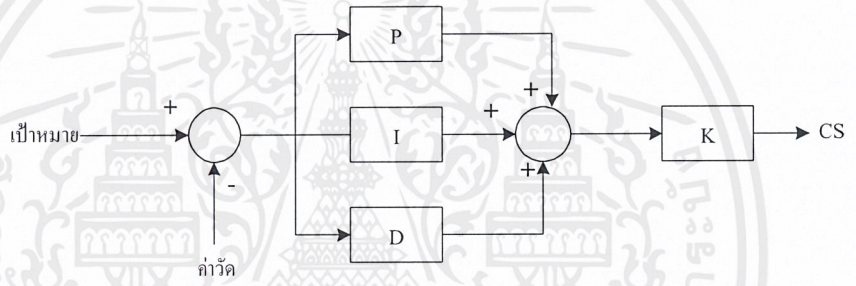
ข) PI Control

1. ทำเหมือน P Control จนขั้น 1 ถึง 8 เพื่อหาค่า PB ที่ดีที่สุด
2. ลดค่า T_I จน offset หายไป
3. เพิ่มค่า T_D ถ้าเกิดการแกว่งขึ้น
4. ทำขั้น 2 และ 3 จนกว่าจะให้ผลตอบ 25% Damping ratio

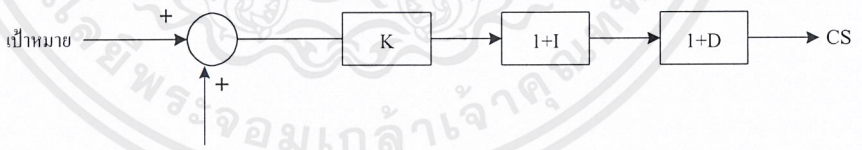
ค) PID Control

1. ทำเหมือน P Control จากขั้น 1 ถึง 6
2. ลดค่า PB ลงมาจนเกิดการแกว่ง
3. เพิ่ม T_D จนการแกว่งหยุด
4. ลดค่า PB จนเกิดการแกว่งใหม่อีก
5. ทำขั้น 2 ถึง 4 หลายครั้ง จน T_D ไม่สามารถหยุดการแกว่งได้
6. เพิ่มค่า PB จนหยุดการแกว่ง
7. ตั้งค่า T_I ให้เท่ากับ T_D ค่าสุดท้าย ($T_I = T_D$ กรณีที่ PID interfere Coefficient เท่ากับ2)

2.5.6 การรบกวนซึ่งกันและกันของค่า PID (PID mutual interference)



รูปที่ 2.32 ตัวควบคุมแบบดีที่สุด



รูปที่ 2.33 ตัวควบคุมทั่ว ๆ ไป

รูปที่ 2.32 แสดง Block diagram ของตัวควบคุมแบบอุดมคติ ซึ่งค่า PID จะสามารถตั้งได้อย่างอิสระไม่มีการรบกวนซึ่งกันและกัน ตัวควบคุมแบบนี้มักจะมีราคาแพง ทั่ว ๆ ไปตัวควบคุมที่มีขายในท้องตลาดมักจะดัดแปลงวงจรให้ง่ายลง โดยมี Block diagram ดังรูป 2.33 จะเห็นว่า I และ D Control ไม่เป็นอย่างอุดมคติ และจะมีการรบกวนซึ่งกันและกัน (Mutual interfere) ซึ่งได้แก่การเปลี่ยนค่า T_I จะมีผลทำให้ PB และ T_D เปลี่ยนแปลงไป หรือถ้าเปลี่ยนค่า T_D จะมีผลทำให้ PB แล T_I เปลี่ยนไปเช่นกัน

ดังนั้น เมื่อทำการปรับค่า PID ที่เหมาะสมได้ทำการคำนวณหาค่า PB , T_I และ T_D จากวิธีการ Reaction Curve หรือ Ultimate Sensitivity method จะใช้ค่าเหล่านี้นำมาตั้งที่ตัวควบคุมไม่ได้ ให้ PB , T_I และ T_D เป็นค่าตั้งที่ตัวควบคุม (Dial Setting)

และ PB' , T'_I และ T'_D เป็นค่าตั้งที่ตัวควบคุม (Dial Setting)

ทั้งสองค่าจะมีความสัมพันธ์กันดังนี้

$$PB = \mu PB' \qquad \text{.....(สูตรที่ 2.1)}$$

$$I = \frac{1}{\mu} T'_I \qquad \text{.....(สูตรที่ 2.2)}$$

$$_D = \mu T'_D \qquad \text{.....(สูตรที่ 2.3)}$$

โดยที่ μ คือ สัมประสิทธิ์การรบกวนซึ่งกันและกัน (Mutual interfere Coefficient)

ค่า μ นี้จะมีค่าไม่คงที่แน่นอน จะขึ้นกับอัตราส่วนของ T_I กับ T_D และขึ้นกับชนิดของตัวควบคุมด้วย μ นี้สัมพันธ์กับค่า $\frac{T'_D}{T'_I}$ (V ค่าคงตัวค่าหนึ่งซึ่งบริษัทผู้ผลิตตัวควบคุมจะต้องกำหนดให้) ซึ่งแสดงตามกราฟในรูป $\frac{T'_D}{T'_I}$ ต้องมีค่าน้อยกว่า 0.25 ถ้ามีค่ามากกว่าจะหาความสัมพันธ์ไม่ได้

2.5.7 วิธีการตั้งค่า PID (Dial Setting) ในตัวควบคุมที่มีการรบกวนกัน

1. คำนวณหาค่า PB' , T'_I และ T'_D ที่ได้จาก Optimum tuning method
2. คำนวณหาค่า $\frac{T'_D}{T'_I}$ โดยที่ V หาจากผู้ผลิตตัวควบคุม
3. หา μ จากกราฟที่ได้จากผู้ผลิตตัวควบคุม
4. คำนวณหาค่า PB , T_I และ T_D จากสูตร 1
5. นำค่า PB , T_I และ T_D ที่คำนวณได้ไปตั้งที่ตัวควบคุม

2.6 การวัดความดัน

2.6.1 บทนำ

กระบวนการทางอุตสาหกรรมส่วนใหญ่หรือเกือบทั้งหมดจะใช้ ของเหลว แก๊ส หรือทั้งสองอย่างการควบคุมกระบวนการเหล่านี้ต้องการการวัดและควบคุมความดันของเหลว และแก๊ส ด้วยเหตุนี้การวัดความดันจึงเป็นสิ่งสำคัญมากที่สุดอันหนึ่งของกระบวนการวัดทั้งหมด

2.6.2 ความดัน

ความดันถูกนิยามเหมือนกับ จำนวนของแรงที่กระทำต่อพื้นผิวอันหนึ่งหรือกระจายทั่วทั้งพื้นผิวและถูกวัดเหมือนกับแรงต่อหนึ่งหน่วยพื้นที่

แรงที่ถูกใช้เพื่อคำนวณความดันจะต้องกระทำเป็นมุมฉากกับพื้นที่ ถ้าแรงกระทำในแนวเอียงแรงที่ถูกแตกออกและอยู่ในแนวตั้งฉากกับพื้นผิวเท่านั้นที่สามารถนำมาใช้คำนวณหาค่าความดัน สำหรับแรงที่ถูกแตกออกและอยู่ในแนวขนานกับพื้นผิวจะไม่มีส่วนร่วมทำให้เกิดความดัน

1) หน่วยวัดความดัน

ความดันอาจจะถูกวัด ในหน่วยของอังกฤษ (fps) หรือหน่วยเมตริค บางทีก็เรียกว่าหน่วย SI ในหน่วยอังกฤษ ความดันถูกวัดในหน่วยของปอนด์ (ของพื้นที่) อย่างไรก็ตามในทางวิศวกรรม ความดันมักแสดงในหน่วยของปอนด์ต่อตารางนิ้ว (psi) ดังนั้นความดันบรรยากาศมีค่าประมาณ 14.696 psi หรือ 1 kg/com ความดันยังสามารถวัดในเทอมของความสูงของของเหลว ซึ่งความดันบรรยากาศจะเท่ากับ 760 mm Hg หรือ 29.92 inches Hg

ความสัมพันธ์ต่อไปนี้ ถูกใช้เพื่อการประมาณค่าความดันสูงและความดันต่ำ

(ก) ความดันสูง 1 นิวตันต่อตารางเมตร

ตารางที่ 2.6 ตารางเปลี่ยนความดัน

เปลี่ยนจาก	คูณด้วย	เป็น Torr
มิลลิเมตรปรอท	1	
นิ้วปรอท	25.4	
นิ้วน้ำ	1.87	
ฟุตน้ำ	22.39	
บรรยากาศ(bars)	760	

ตารางที่ 2.6 (ต่อ) ตารางเปลี่ยนความดัน

Psis	51.7	
Kilopascal	7.2	
Microns	10	

2) ชนิดต่าง ๆ ของความดัน

เมื่อความดันถูกวัดโดยปกติเราต้องการอ่านมันในเทอมของความดันมาตรวัดความสัมบูรณ์ ความดันสูญญากาศหรือความดันแตกต่าง ชนิดต่าง ๆ ของความดันได้อธิบายไว้ข้างล่างนี้

(ก) ความดันมาตรวัด มาตรวัดความดันของเหลวส่วนใหญ่จะใช้ความดันบรรยากาศ (14.7 psi) เหมือนกับจุดศูนย์ นั่นคือ มันจะชี้แสดงความดันที่ศูนย์ psi ที่พื้นผิวของของเหลวอยู่ที่ความดัน 14.7 psi (1 Kg/com)

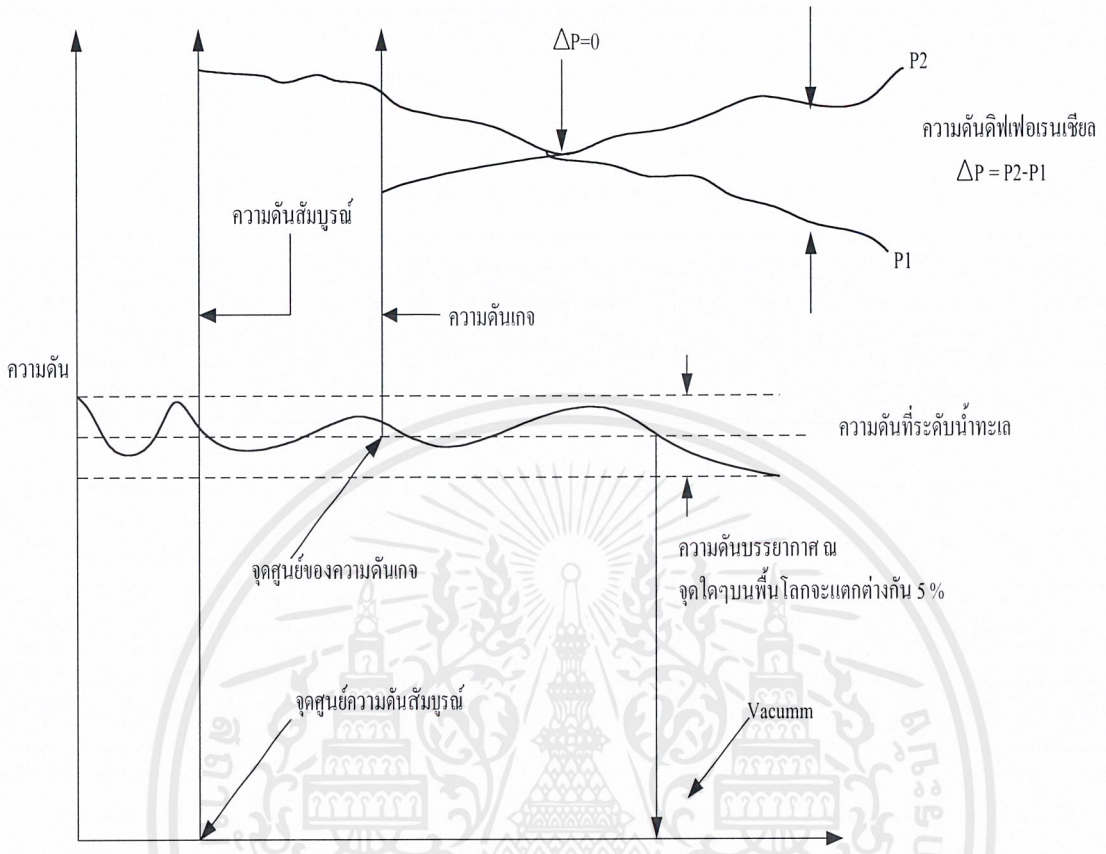
มาตรวัดอันหนึ่งที่ชี้แสดงศูนย์ psi ที่ความดันบรรยากาศจะวัดความแตกต่างระหว่างความดันที่แท้จริงกับความดันบรรยากาศ ความแตกต่างอันนี้ถูกเรียกว่า "ความดันมาตรวัด" มันมีอักษรย่อเป็น psig (pound per square inch gauge)

(ข) ความดันสัมบูรณ์ ความดันสัมบูรณ์ก็คือ ความดันที่แท้จริงทั้งหมด (รวมทั้งความดันบรรยากาศ) ที่กระทำบนพื้นผิวอันหนึ่ง มันมีอักษรย่อเป็น psia (pounds per square inch absolute)

(ค) ความดันสูญญากาศหรือความดันแตกต่าง มาตรวัดที่ใช้ชี้แสดงความดันมาตรวัดอาจถูกออกแบบเพื่อใช้ชี้แสดงความดันที่ต่ำกว่าศูนย์ มาตรวัดเหล่านี้ถูกเรียกว่า "มาตรวัดสูญญากาศ" มาตรวัดที่ใช้ชี้แสดงความดันสัมบูรณ์ไม่สามารถใช้ชี้แสดงความดันที่ต่ำกว่าศูนย์ เพราะว่า ศูนย์คือสูญญากาศที่สมบูรณ์

ในการวัดความดันแตกต่าง ความดันมาตรวัดคือ ความแตกต่างระหว่างความดันสัมบูรณ์ของไหลกับความดันบรรยากาศ

ความสัมพันธ์ระหว่าง ความดันสัมบูรณ์ ความดันมาตรวัด และความกดดันของอากาศได้แสดงไว้ในรูปที่ 2.34



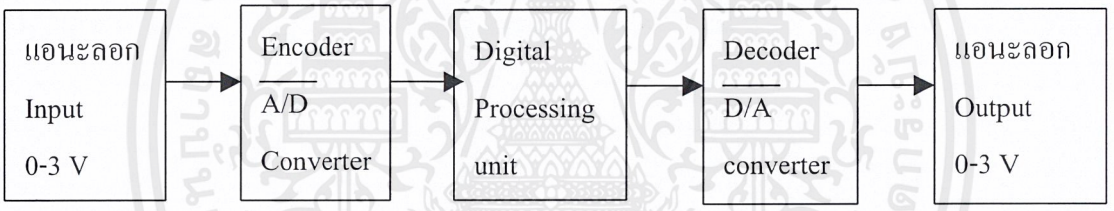
รูปที่ 2.34 ความสัมพันธ์ระหว่างความดันสัมบูรณ์ ความดันมาตรวัด และความกดดันของอากาศ

2.7 การแปลงสัญญาณดิจิตอล- แอนะลอก (D/A) และแอนะลอก-ดิจิตอล (A/D)

2.7.1 บทนำ

วงจรแปลงสัญญาณดิจิตอลเป็นแอนะลอก หรือเรียกย่อ ๆ ว่า วงจร ดี/เอ (Digital to Analog Converter) เป็นวงจรที่ใช้สำหรับเปลี่ยนสัญญาณดิจิตอล ซึ่งอยู่ในรูปของรหัสใบนารีให้เป็นสัญญานแอนะลอก ซึ่งเป็นรูปสัญญานแรงดัน ส่วนวงจรแปลงสัญญาณแอนะลอกเป็นดิจิตอลหรือเรียกว่า วงจร เอ/ดี (แอนะลอก-to-Digital Converter) จะทำงานกลับกันกับวงจร ดี/เอ คือเป็นวงจรที่ใช้สำหรับเปลี่ยนสัญญาณแอนะลอกซึ่งเป็นรูปสัญญาณแรงดันให้เป็นสัญญาณดิจิตอล ซึ่งอยู่ในรูปของรหัสไบนารี วงจรประเภทดี/เอ หรือ เอ/ดี มักพบเสมอในการสื่อสารระบบพีซีเอ็ม (PCM) หรือในวงจรสัญญาณเรียกขาน (Calling Signal) ของโทรศัพท์ระบบเอ็มเอฟพี (MFP) และในเครื่องดิจิตอล-โวลต์-มิเตอร์ (DVOM) เป็นต้น

2.7.2 การแปลงสัญญาณดิจิตอลเป็นแอนะลอก (D/A Converter)



รูปที่ 2.35 ผังไดอะแกรมระบบดิจิตอลซึ่งมีอินพุตและเอาต์พุตเป็นสัญญาณแอนะลอก

จากบล็อกไดอะแกรม ของระบบดิจิตอล ที่มีสัญญาณอินพุตและเอาต์พุตเป็นสัญญาณแอนะลอกดังแสดงในรูปที่ 2.36 ในส่วนของ ภาค ดี/เอ คอนเวอร์เตอร์ สมมติว่าเราต้องการเปลี่ยนสัญญาณไบนารีจากภาคประมวลผลข้อมูล (Digital Processing Unit) ให้เป็นแรงดันแอนะลอกตั้งแต่ 0 ถึง 3 โวลต์ ที่ภาคเอาต์พุต (Analog Out put) โดยมีตารางความจริงแสดงการเปลี่ยนรหัสไบนารีให้เป็นค่าแรงดัน แอนะลอก ค่าต่าง ๆ แสดงตารางที่ 2.7

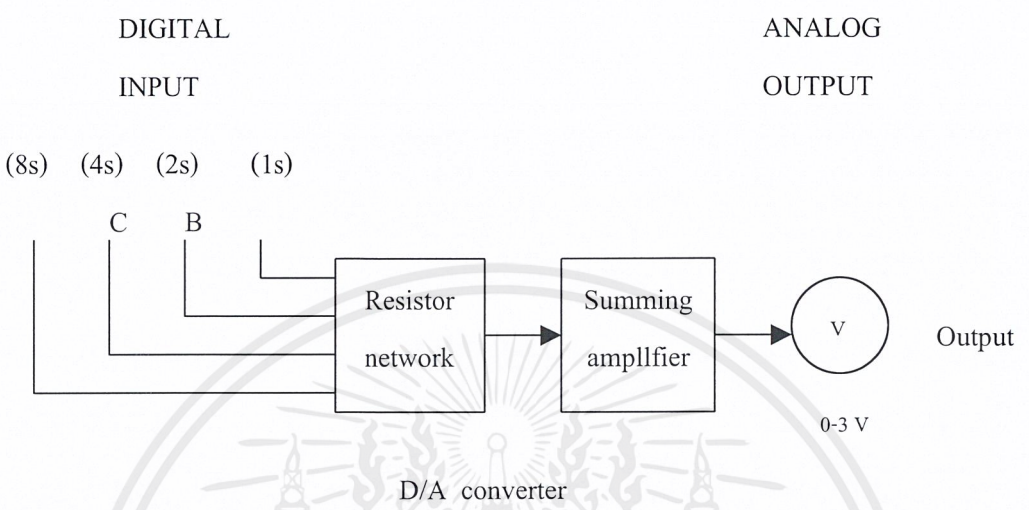
ตารางที่ 2.7 ตารางการเปลี่ยนข้อมูลรหัสไบนารีเป็นค่าแรงดันแอนะลอก

	DIGITAL INPUT				ANALOG OUTPUT
	D	*C*	*B*	*A*	Volts
Row 1	0	0	0	0	0.0
Row 2	0	0	0	1	0.2
Row 3	0	0	1	0	0.4
Row 4	0	0	1	1	0.6
Row 5	0	1	0	0	0.8
Row 6	0	1	0	1	1.0
Row 7	0	1	1	0	1.2
Row 8	0	1	1	1	1.4
Row 9	1	0	0	0	1.6
Row 10	1	0	0	1	1.8
Row 11	1	0	1	0	2.0
Row 12	1	0	1	1	2.2
Row 13	1	1	0	0	2.4
Row 14	1	1	0	1	2.6
Row 15	1	1	1	0	2.8

รหัสไบนารีมีจำนวน 4 บิท (D, C, B, A) เป็นอินพุตของวงจรดี/เอ ซึ่งมีระดับแรงดันลอจิก “1” ที่ประมาณ +3 ถึง 5 โวลต์ และระดับแรงดันลอจิก “0” ที่ประมาณ 0 โวลต์ ตารางค่าแรงดันแอนะลอกเอาต์พุตจะเพิ่มขึ้นครั้งละ 0.2 โวลต์ เริ่มตั้งแต่ 0 โวลต์ ที่รหัส (0000) กล่าวคือ เมื่อรหัสเป็น (0001) เอาต์พุตจะมีแรงดันเท่ากับ 0.2 โวลต์ และเมื่อรหัสไบนารีเป็น (0010) เอาต์พุตจะมีแรงดันเท่ากับ 0.4 โวลต์ เป็นต้น

ผังไดอะแกรมของวงจร ดี/เอ คอนเวอร์เตอร์ แสดงดังรูปที่ 2.36 ดิจิตอลอินพุต (D, C, B, A) จะป้อนเข้าทางซ้ายมือ วงจรถอดรหัสแบ่งออกเป็นสองภาคคือ ภาควงจรความต้านทาน(Resister

Network) และภาคขยายผลรวม (Summing Amplifier) ค่าแรงดันแอนะลอกทางเอาต์พุตที่ได้จะอ่านค่าได้จาก ดีซีโวลต์มิเตอร์ทางขวามือ



รูปที่ 2.36 ผังไดอะแกรมของวงจรดี/เอ

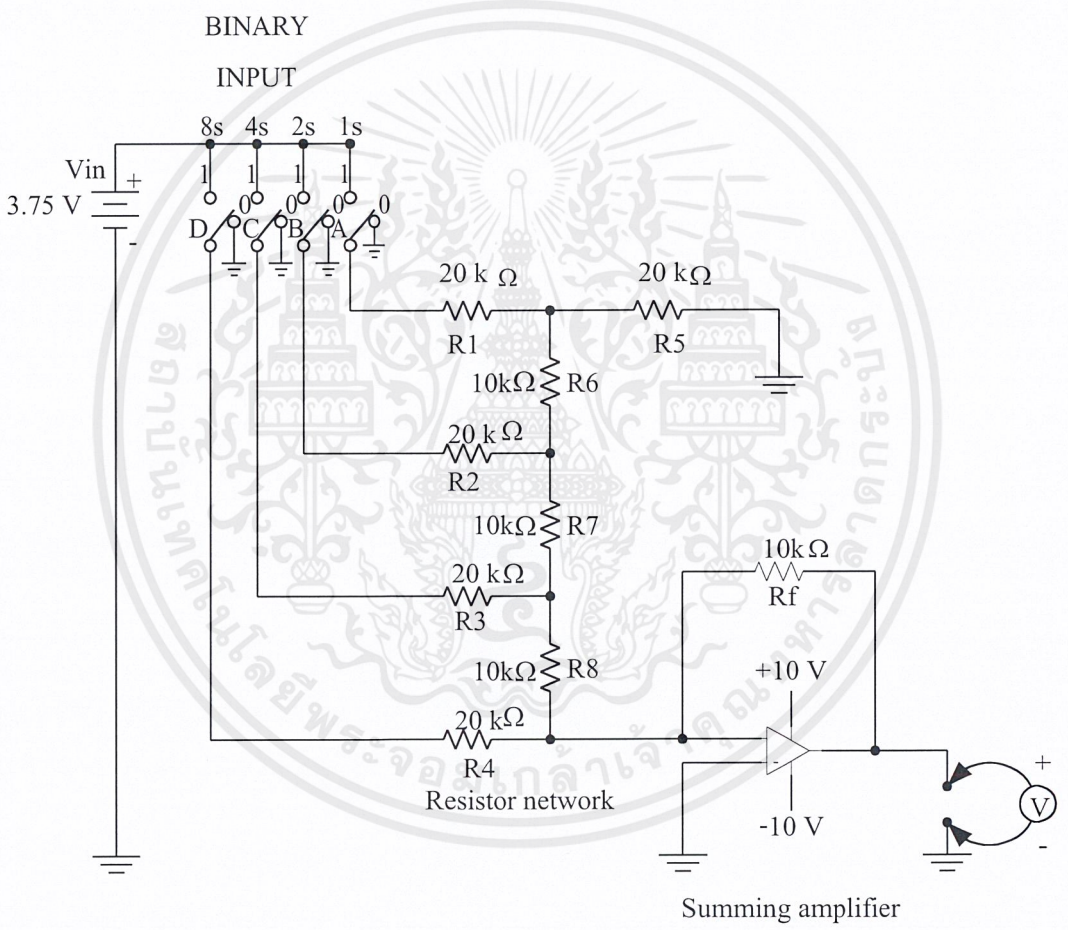
ผังวงจรความต้านทานในรูปที่ 2.36 จะต้องเป็นตัวกำหนดน้ำหนักของรหัสไบนารีที่แต่ละบิท กล่าวคือ รหัส "1" ที่อินพุต B จะมีค่าเป็นสองเท่าของรหัส "1" ที่อินพุต A และรหัส "1" ที่อินพุต C ก็จะมีค่าเป็นสองเท่าของรหัส "1" ที่อินพุต B หรือเป็นสี่เท่าของ รหัส "1" ที่อินพุต C หรือเป็นแปดเท่า ของรหัส 1 ที่อินพุต A วงจรความต้านทานดังกล่าวเรียกว่าวงจรรีซิสเตอร์แลดเดอร์ (Resistor Ladder Network) หรือวงจรแลดเดอร์

วงจรขยายผลรวมโดยทั่วไปมักจะใช้ไอซีออปแอมป์ ซึ่งจะแปลงแรงดันวงจรรีซิสเตอร์แลดเดอร์ให้เป็นค่าแรงดันแอนะลอกที่ต้องการ

2.7.3 อุปกรณ์แปลงสัญญาณดิจิตอลเป็นแอนะลอกชนิด LADDER

วงจร D/A converter ประกอบไปด้วย 2 ส่วนคือ ส่วนตัวต้านทาน และส่วนขยาย ดังรูปที่ 2.37 เรียกว่าวงจร R-2R แลดเดอร์ ดี/เอ ซึ่งเป็นวงจร ดี/เอ 4 หลัก โดยใช้วงจรรีซิสเตอร์แลดเดอร์ (Ladder network) ประกอบด้วย R และ 2R เป็นตัวจัดระดับแรงดันของเอาต์พุต ตามรหัสไบนารี ที่อยู่ในรีจิสเตอร์ถ้าให้ระดับลอจิก " 0 " เท่ากับ 0 โวลต์ และระดับลอจิก "1" เท่ากับ V โวลต์ เอาต์พุตจะให้ระดับแรงดันที่แตกต่างตามรหัสไบนารี 2^4 หรือ 16 ค่า แต่ละระดับจะต่างกัน เท่ากับ 1/16 โวลต์ รหัสไบนารีในริจิสเตอร์จะเรียงหลักมากที่สุด ถึงหลักน้อยที่สุด (ลำดับหลักจากซ้ายไปขวา) จากฟลิบฟลอบ D, C, B และ A ตามลำดับและโดยคุณสมบัติของวงจรแลดเดอร์ ระดับแรงดันที่

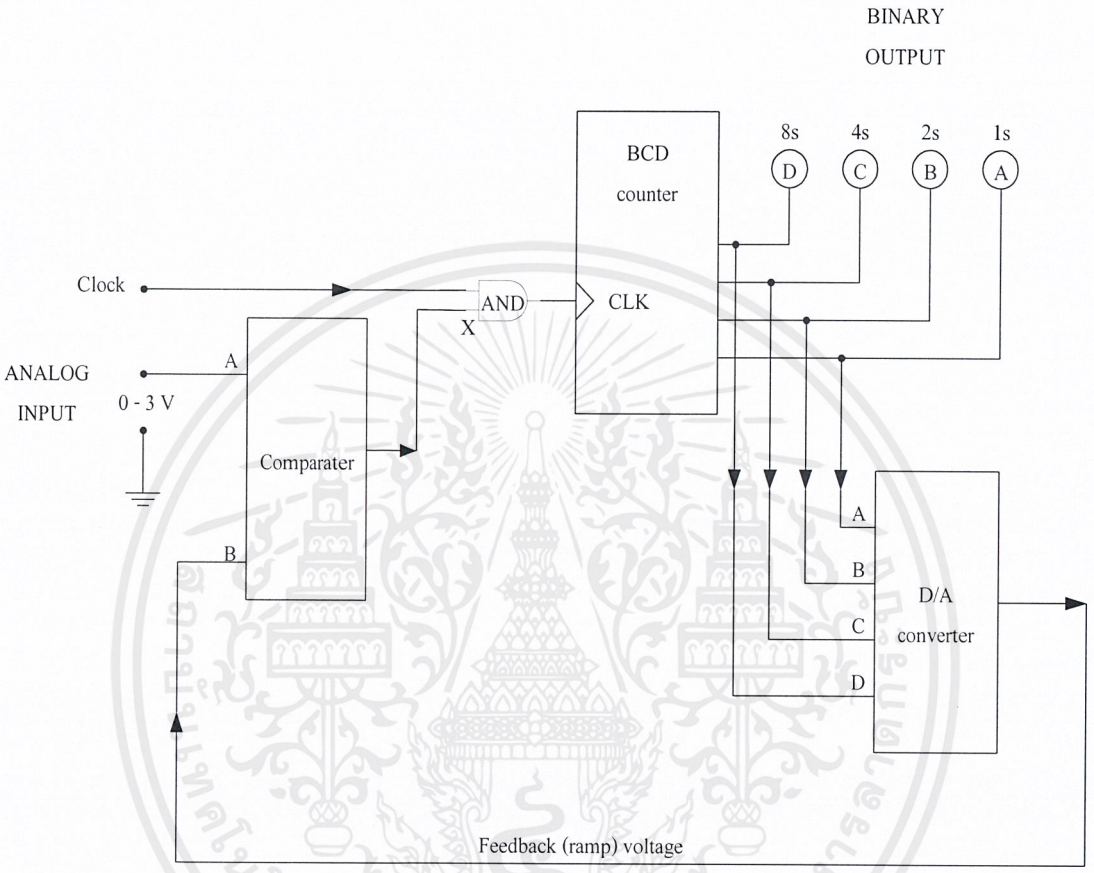
เอาต์พุต จะเท่ากับผลรวมของแรงดันที่เกิดขึ้นจากรหัสไบนารีในหลักที่มีลอจิกเป็น “1” (หรือระดับแรงดัน V โวลต์) ซึ่งแต่ละหลักจะให้แรงดันที่เอาต์พุตในขณะมีลอจิกเป็น “1” เท่ากับ 1/2V, 1/4V, 1/8V และ 1/16 V เรียงลำดับจาก D ถึง A ตัวอย่างเช่น สมมุติรีจิสเตอร์มีรหัสไบนารีอยู่ที่เลข 5 (0101) ซึ่ง Q_D= 0 , Q_C = 1, Q_B = 0 และ Q_A=1 จะได้แรงดัน ที่เอาต์พุตเท่ากับ (0*1/2 V + 1 * 1/4 V + 0 * 1/8 V + 1*1/16V) หรือ 5/16 V เป็นต้นหรือถ้ามีรหัสไบนารีอยู่ที่เลข 15 (1111) ซึ่งเป็นรหัสสูงสุดที่จะมีได้ในวงจรนี้ ก็จะได้แรงดันที่เอาต์พุต เท่ากับ 15/16 โวลต์ เป็นต้น



รูปที่ 2.37 วงจร D/A converter โดยใช้ R-2R Ladder

2.7.4 วงจรแปลงสัญญาณแอนะลอกเป็นสัญญาณดิจิตอล

1) วงจร เอ/ดี แบบเคาน์เตอร์-แร้มพ์ (Conter Ramp A/D)



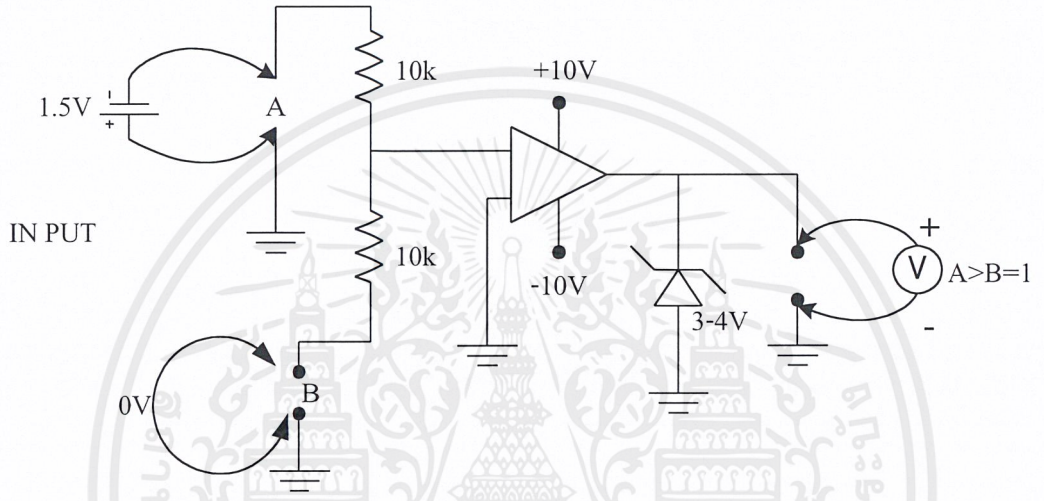
รูปที่ 2.38 วงจร เอ/ดี แบบเคาน์เตอร์แร้มพ์

วงจรในรูปที่ 2.38 เป็นวงจรเบื้องต้นของ เอ/ดี แบบหนึ่ง ซึ่งเรียกว่า แบบเคาน์เตอร์แร้มพ์ (Counter Ramp A/D Converter) อาศัยการทำงานด้วยวงจรดีเอ ที่ประกอบด้วยวงจรนับ (Counter) กับวงจร R-2R แลดเดอร์ ผลิตสัญญาณแบบแร้มพ์ ซึ่งจะเพิ่มระดับขึ้นมารหัสเลขไบนารีของการนับทุก ๆ ครั้งที่มีสัญญาณพัลซ์เวลามากระตุ้นวงจรนับ เอาต์พุตของวงจรเลดเดอร์ จะถูกต่อเข้ากับวงจรเปรียบเทียบแรงดัน โดยอาจจะต่อผ่านวงจรดีซี แอมป์ เพื่อขยายหรือลดขนาดของสัญญาณให้อยู่ในสเกล (Scale) เดียวกับระดับแรงดันอินพุตที่ต้องการแปลง และเมื่อค่าระดับสัญญาณแร้มพ์มีค่าเกินระดับแรงดันอินพุต เอาต์พุตของวงจรเปรียบเทียบแรงดันจะส่งสัญญาณลอจิกไปควบคุมให้วงจรนับหยุดนับ รหัสไบนารี ที่อยู่ในวงจรนับในขณะนี้จะใช้เป็นรหัสไบนารี ที่แทนระดับแรงดันอินพุตที่ต้องการ ในกรณีที่ต้องการทำการแปลงใหม่ จะต้องเริ่มต้นการทำงานของวงจรด้วย การรีเซ็ท

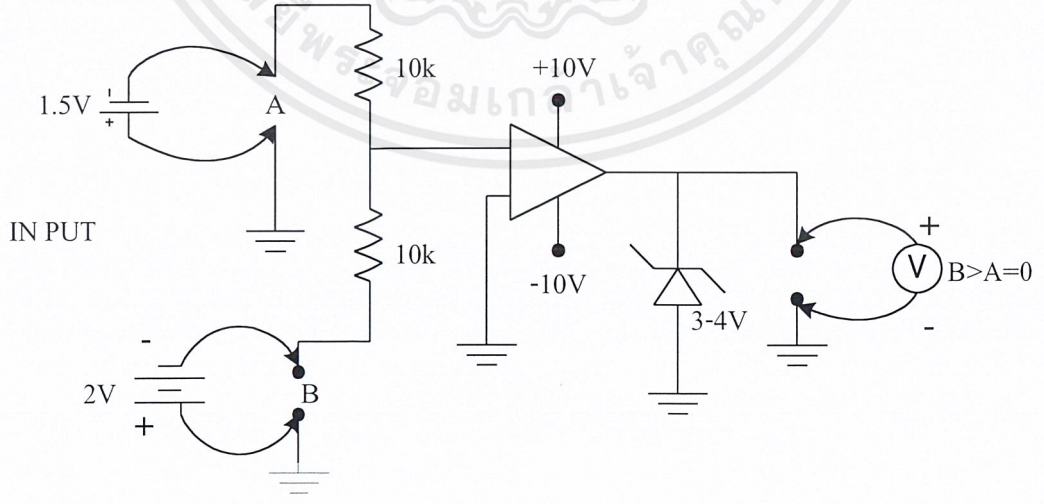
ให้วงจรนับมีรหัสเป็น “0” เสียก่อนเสมอ เกทควบคุมจึงจะได้รับสัญญาณลอจิกที่ทำให้วงจรนับรับสัญญาณพัลซ์เวลาเข้ามากระตุ้นวงจรนับให้เริ่มผลิตสัญญาณแร้มพ์ เพื่อนำไปเปรียบเทียบใหม่ อีกครั้ง

2.7.5 วงจรเปรียบเทียบแรงดัน (Voltage Comparator)

วงจรเปรียบเทียบแรงดันที่ใช้ไอซี ออปแอมป์ แสดงดังรูปที่ 2.39



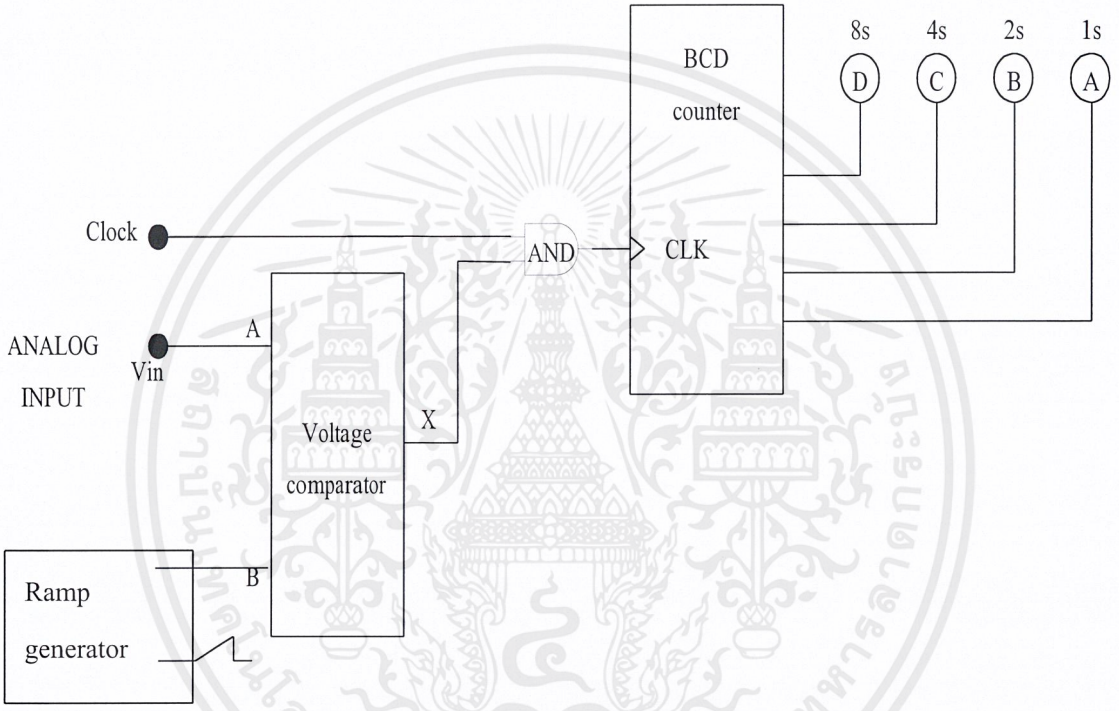
รูปที่ 2.39 (ก) วงจรเปรียบเทียบแรงดันเมื่อค่า A มากกว่า



รูปที่ 2.39 (ข) วงจรเปรียบเทียบแรงดันเมื่อค่า B มากกว่า

จากรูปที่ 2.39 (ก) ถ้าแรงดันที่อินพุต A มากกว่าแรงดันที่อินพุต B จะทำให้เอาต์พุตของออปแอมป์ เป็นลอจิก "1" หรือค่าแรงดันประมาณ 3-4 โวลต์ (จำกัดโดยค่าแรงดันซีเนอร์ของซีเนอร์ไดโอดที่เอาต์พุต) และเมื่อแรงดันที่จุด B มากกว่า แรงดันที่จุด A (ตามขั้วที่กำหนด) จะทำให้เอาต์พุตของออปแอมป์ เป็นลอจิก "0" หรือค่าแรงดันประมาณ –0.6 โวลต์ ซึ่งจำกัดโดยแรงดัน forverse ของซีเนอร์ไดโอดที่เอาต์พุต ดังแสดงในรูปที่ 2.39 (ข)

2.7.6 วงจรลิเนียร์ แร้มพ์ เอ/ดี (Linear Ramp A/D)

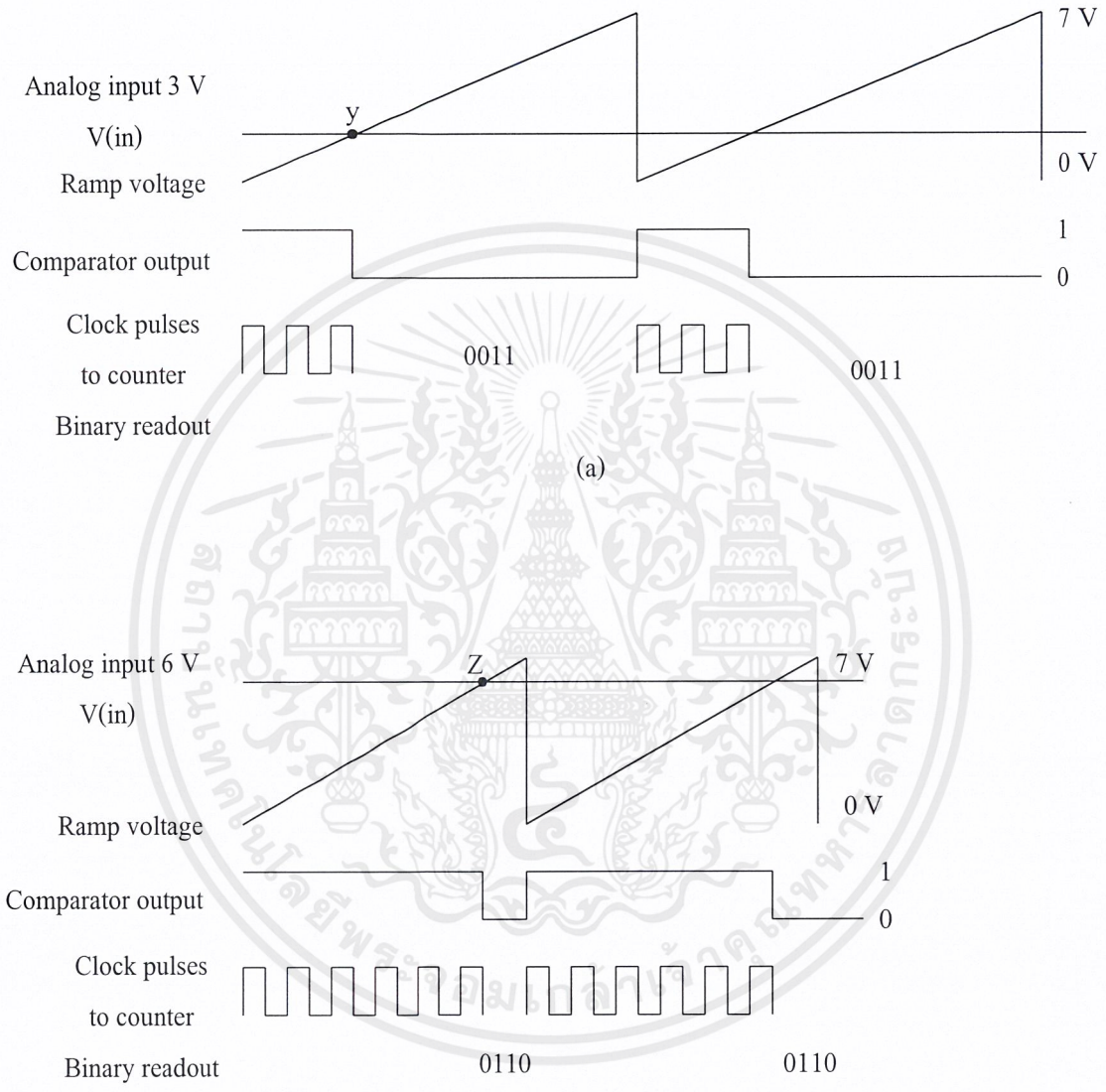


รูปที่ 2.40 ผังไดอะแกรมของวงจรลิเนียร์-แร้มพ์ เอ/ดี

ผังไดอะแกรมของวงจรลิเนียร์-แร้มพ์ เอ/ดี แสดงดังรูปที่ 2.40 การทำงานส่วนของวงจรคล้ายคลึงกันกับวงจรเคาน์เตอร์-แร้มพ์ เอ/ดี มาก โดยจะมีภาคกำเนิดสัญญาณแบบแร้มพ์ ซึ่งจะให้รูปสัญญาณแรงดันเป็นแบบสามเหลี่ยม หรือรูปฟันเลื่อย ป้อนให้กับอินพุตข้างหนึ่งของวงจรเปรียบเทียบแรงดัน

สมมุติว่ามีแรงดันแอนะลอก ขนาด +3 โวลต์ ที่อินพุตของวงจร เอ / ดี ในรูปที่2.40 และสัญญาณแร้มพ์เริ่มต้นเพิ่มค่าแรงดันจาก 0 โวลต์ ขึ้นไปเรื่อย ๆ ที่อินพุต B แต่ยังคงมีระดับแรงดันต่ำกว่าที่อินพุต A จะทำให้เอาต์พุตของวงจรเปรียบเทียบแรงดัน เป็นลอจิก "1" ทำให้แอนด์เกทเปิดให้สัญญาณพัลซ์เวลาผ่านไปยังวงจรนับ (Counter) ได้เป็นจำนวนสามพัลซ์ หลังจากนั้นระดับแรงดันเป็นลอจิก "0" แอนด์เกทจะปิด วงจรนับจึงหยุดอยู่ที่รหัสไบนารี 0011 ซึ่งหมายถึงระดับแรงดัน

3 โวลต์ ที่แอนะลอกอินพุต ดังรูปสัญญาณที่แสดงในรูปที่ 2.41 (a) ในทำนองเดียวกันกับ เมื่อแรงดันที่แอนะลอกอินพุตเป็น 6 โวลต์ จะได้รหัสไบนารีในวงจรนับเป็น 0110 หรือ 6 โวลต์นั่นเอง

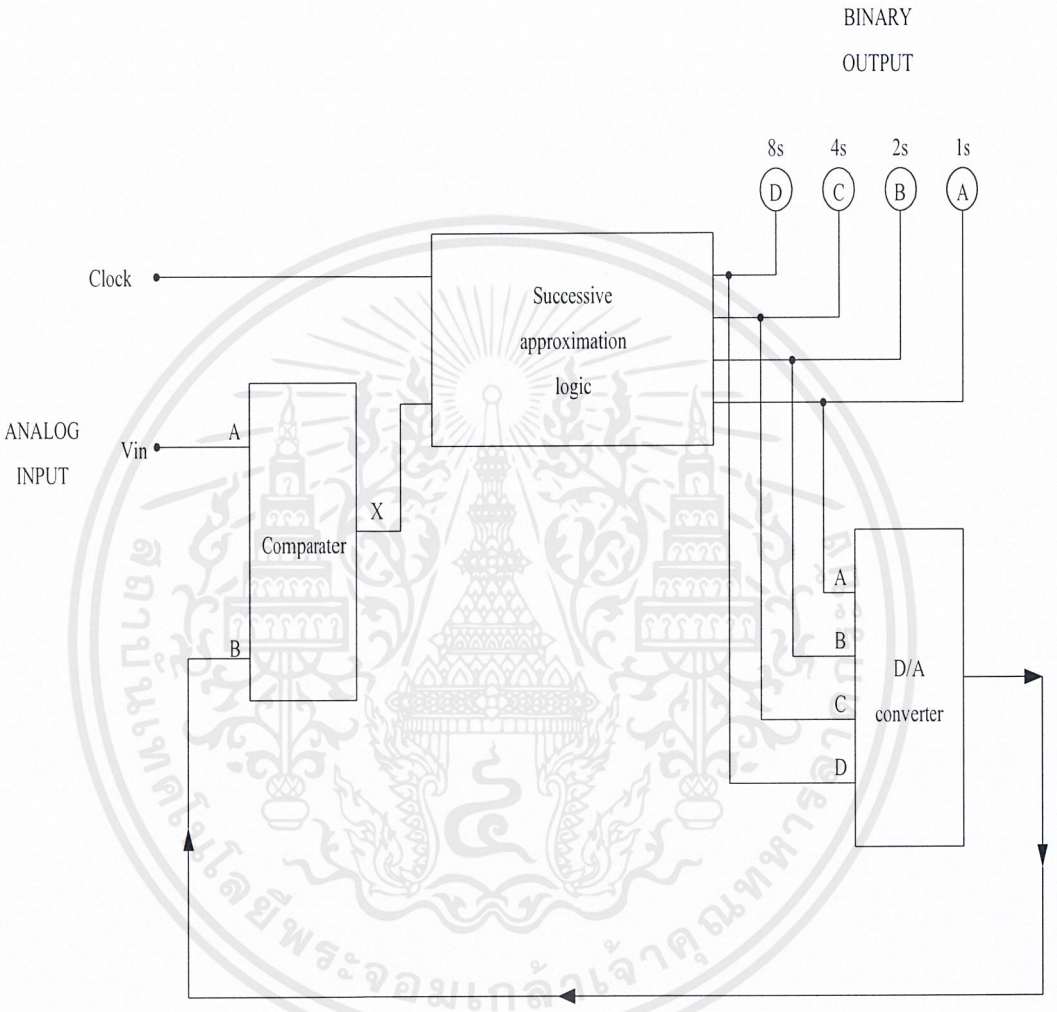


รูปที่ 2.41 รูปสัญญาณในวงจรลิเนียร์แร้มพ์ เอ/ดี

วงจรลิเนียร์ แร้มพ์ เอ / ดี มีความยุ่งยากกว่าจะต้องใช้เวลานานสำหรับการนับให้ได้ค่าแรงดันสูง ๆ ตัวอย่างเช่น ถ้าต้องการไบนารีเอาต์พุตถึงแปดบิท จะต้องใช้พัลซ์นับถึง 255 พัลซ์เป็นต้น

เพื่อลดปัญหาความล่าช้าในการเปลี่ยนรหัสแบบนี้ จึงต้องใช้วงจร เอ / ดี อีกแบบหนึ่งเรียกว่า Successive Approximation เอ/ดี

2.7.7 วงจร Succesive Approximation (A/D)



รูปที่ 2.42 ผังไดอะแกรมของวงจร Succesive Approximation

วงจร เอ/ดี แบบนี้ประกอบด้วยวงจรเปรียบเทียบแรงดัน วงจร ดี / เอ คอนเวอร์เตอร์และวงจร Succesive Approximation ดังแสดงรูปที่ 2.42

สมมุติว่าแรงดันแอนะลอกที่อินพุตนั้นเท่ากับ 7 โวลต์ วงจร Successive Approximation Logic จะตั้งให้บิท MSB เป็นลอจิก "1" ทำให้ได้รหัส ไบนารี (1000) ป้อนผ่านวงจร ดี/เอ ไปยัง อินพุตของวงจรเปรียบเทียบแรงดัน ซึ่งจะเปรียบเทียบว่า แรงดันเปรียบเทียบที่ป้อนกลับสูงหรือต่ำกว่าแรงดันที่อินพุต ถ้าแรงดันเปรียบเทียบสูงกว่า วงจร Successive Approximation ลอจิกก็จะเคลียร์บิท

MSB หรือบิท 2^3 ให้เป็นลอจิก "0" แล้วเซ็ทบิทถัดไปคือบิทที่ 2^2 ให้เป็น ลอจิก "1" ผลที่ได้คือรหัส (0100) แล้วนำไปเปรียบเทียบใหม่ถ้าแรงดันเปรียบเทียบยังต่ำกว่าแรงดันที่อินพุตอีก วงจรลอจิกก็จะเซ็ทบิทถัดไปคือบิทที่ 2^0 ให้เป็นลอจิก "1" ผลที่ได้คือรหัส (0111) เมื่อนำไปเรียบเทียบกับแรงดันที่อินพุตปรากฏว่าเท่ากัน วงจรเปรียบเทียบก็จะควบคุมให้วงจร Successive Approximation หยุดทำงานข้อมูลรหัสไบนารีที่อยู่ในวงจรลอจิกควบคุมก็คือ (0111) ซึ่งจะเป็นรหัสไบนารีแทนค่าแรงดันแอนะลอกที่อินพุต 7 โวลต์นั่นเอง

ข้อดีของวงจร เอ/ดี แบบนี้คือจะใช้เวลาในการเปลี่ยนสัญญาณเร็วมาก เพราะไม่ต้องใช้วิธีนับเรียงลำดับไปเรื่อย ๆ เหมือนกับวงจร เอ/ดี แบบอื่น ๆ วงจร Successive Approximation เอ/ดี จึงเป็นวงจรที่นิยมใช้กันอย่างมากมายและกว้างขวาง

2.8 คุณสมบัติของ A/D CONVERTER

2.8.1 ชนิดของเอาต์พุต

โดยทั่วไปแล้ว A/D Converter จะถูกแบ่งเป็นชนิดที่ให้เอาต์พุตออกมาเป็นเลขฐานสิบ และเป็นเลขฐานสิบสอง converter ที่ทำหน้าที่เปลี่ยนสัญญาณแอนะลอกเป็นดิจิตอลที่มีเอาต์พุตเป็นเลขฐานสิบ มักจะถูกใช้เป็นดิจิตอลโวลต์มิเตอร์และถูกใช้ใน Digital panal meter และ DMM คอนเวอร์เตอร์ที่เปลี่ยนสัญญาณแอนะลอกเป็นดิจิตอลที่มีเอาต์พุตเป็นเลขฐานสองจะมีจำนวนเอ้าต์พุทตั้งแต่ 4 –16 เอาต์พุต ตัวแปลงสัญญาณอนาลอกเป็นดิจิตอลที่มีเอาต์พุตเป็นเลขฐานสอง จะเป็นอุปกรณ์อินฤตชนิดหนึ่ง ในระบบที่มีไมโครโปรเซสเซอร์เป็นฐานในการควบคุม เรียกว่า A/D converter แบบ μ P – Type

2.8.2 ความละเอียด (Resolution)

ผลที่ได้จากการทำงานของ A/D converter เป็นตัวเลขฐานสอง เอาต์พุตที่ออกมาจึงเป็นเลขแบบหลายบิต สำหรับแบบที่ให้เอาต์พุตออกมาเป็นเลขฐานสิบ (ใช้ใน DMM) ผลที่ได้เป็นตัวเลขในฐานสิบโดยทั่วไปแล้ว A/D converter ที่มีเอาต์พุตเป็นเลขฐานสองจะมีจำนวนบิตเป็น 4 , 6 , 8 , 10 , 12 , 14 และ 16 บิต อาจมีการ error เกิดขึ้นบ้างเล็กน้อยเนื่องจากการใช้ Discrete binary step เพื่อแทนสัญญาณอนาลอกที่มีความต่อเนื่องกันเรียกว่า Quantizing error

A/D converter ขนาด 16 บิต มีความถูกต้องละเอียดแม่นยำมากกว่าแบบ 4 บิต เพราะว่ามันแบ่งอินพุตหรืออ้างอิงโวลต์เตจเป็น Discrete step ที่เล็กๆ เช่น แต่ละ Step ใน A/D converter แบบ 4

บิตจะต้องเป็น หนึ่งในสิบห้าของอินพุตโวลต์เตจ ผลที่ออกมาคือ 6.7 % ส่วนแบบ 8 บิต ควรจะมี discrete step เป็นจำนวน 255 ซึ่งจะเท่ากับ 0.39 % และแบบ 8 บิต จะมีความแม่นยำมากกว่า 4 บิต

2.8.3 ความแม่น (Accuracy)

ผลจากการทำงานของ A/D converter อาจมีข้อผิดพลาดทางตัวเลขอันเนื่องมาจาก Discrete step ที่พบที่เอาต์พุตของไอซี ซึ่ง A/D converter ทุกตัวมีความผิดพลาดอันนี้อยู่ นอกจากนั้นความผิดพลาดที่เกิดขึ้นใน A/D อีกอย่างหนึ่งคือ แอนะลอก Comment เช่น วงจรเปรียบเทียบและความผิดพลาดอื่น ๆ อันเนื่องมาจากโครงข่ายวงจรของตัวต้านทาน ความละเอียดแม่นยำของ A / D converter เรียกว่า Accuracy ของ IC A/D converter

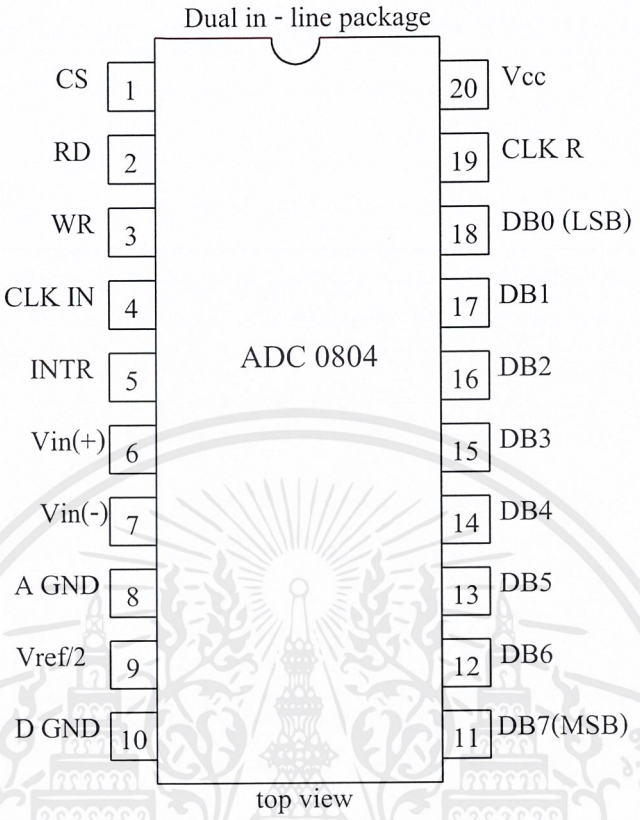
2.8.4 เวลาการแปลงผกผัน (Conversion Time)

Conversion time เป็นเวลาที่ A/D ใช้ในการให้ไอซีเปลี่ยนค่าแอนะลอกทางอินพุต ให้เป็นข้อมูลเอาต์พุตเลขฐานสอง (หรือเลขฐานสิบ) โดยทั่วไปแล้ว conversion time ของ A/D ที่มีเอาต์พุตเป็นเลขฐานสิบมักอยู่ในช่วง 200 – 400 ms ส่วน A / D ที่มีเอาต์พุตเป็นเลขฐานสอง มักมีค่าอยู่ระหว่าง 0.05 – 100,000 μs

ลักษณะเฉพาะอื่นๆ คือค่าแรงดันจากแหล่งจ่ายไฟมักมีค่าประมาณ +5 V แต่ A/D บางตัวก็จะทำงานที่ค่าโวลต์เตจ +5 V ถึง +15 V ระดับแรงดันทางเอาต์พุตเป็นทั้งแบบ TTL , CMOS หรือ tristate (สามสถานะ) ช่วงกว้างของอินพุตโวลต์เตจมักเป็น 5 โวลต์ ค่าการสูญเสียกำลังสูงสุดของ A/D converter มักมีค่าอยู่ระหว่าง 15 ถึง 3000 มิลลิวัตต์

2.8.5ไอซี A/D converter

ไอซี A / D converter มีมากมายและซื้อขายกันทั่วไปตามลักษณะการใช้งาน เนื่องจากภายในโครงงานนี้ได้นำไอซี A /D converter ใช้ในวงจรเป็นไอซี เบอร์ ADC 0804 จึงกล่าวถึงคุณสมบัติและลักษณะต่าง ๆ ของไอซีเบอร์นี้ โดยมีไดอะแกรมรายละเอียดของขาต่าง ๆ ของไอซี และรายละเอียดทั้งหมดของขาไอซีดังรูป2.43



รูปที่ 2.43 แสดงรายละเอียดของไอซีเบอร์ 0804

A/D converter เบอร์ ADC 0804 ถูกออกแบบมาให้สามารถเชื่อมต่อเข้าโดยตรงได้กับไมโครโปรเซสเซอร์หลายๆ เบอร์ บางขาไอซี ADC 0804 อาจเหมือนกับขาของไมโครโปรเซสเซอร์ที่เราใช้งานทั่วไป เช่น ADC 0804 ใช้ชื่อขา INTR WR และ RD ในโมโครโปรเซสเซอร์ไอซีเบอร์นี้สามารถเชื่อมต่อกับไมโครโปรเซสเซอร์แบบ 8 บิต ที่นิยมใช้กันทั่วไปชนิดอื่น ๆ ได้อีกด้วย ขา control input ใช้สำหรับรับสัญญาณ (Chip select) จากวงจรถอดรหัสค่าแอดเดรสในไมโครโปรเซสเซอร์

ADC 0804 เป็น Successive – approximation A/D converter แบบ 8 บิต CMOS มีเอาต์พุต 3สถานะ ดังนั้นจึงสามารถเชื่อมต่อเข้าโดยตรงกับระบบ Microprocessor – base system ทาง data bus ได้ ADC 0804 มีเอาต์พุตเลขฐานสองและมี conversion time เพียง 100 ไมโครเซค เท่านั้น อินพุตและเอาต์พุตของมันเข้ากันได้ทั้ง MOS และ TTL มีตัวกำเนิด clock รวมอยู่ในชิพสำเร็จรูปอยู่แล้ว โดยจะต้องต่ออุปกรณ์ภายนอก (ตัวต้านทาน , ตัวเก็บประจุ) เพิ่มเติมเพื่อให้ทำงานได้

ไอซี ADC 0804 ทำงานด้วยไฟ DC 5 โวลต์ จากเพาเวอร์ซัพพลายและสามารถใส่รหัสความต่างศักย์แอนะลอกทางอินพุตได้ตั้งแต่ 0-5 โวลต์ หน้าที่ของวงจรคือใส่รหัสความแตกต่างกันของศักดาไฟฟ้าระหว่าง Vin (+) และ Vin(-) เปรียบเทียบกับระดับแรงดันอ้างอิงเพื่อให้สัมพันธ์กับค่าตัวเลขฐานสอง

2.8.6 อิเล็กทรอนิกส์สวิตช์

ไอซีเบอร์ CD4066B มีคุณสมบัติที่สามารถเปลี่ยนสัญญาณดิจิตอลเป็นสัญญาณแอนะลอกได้ เป็นไอซีประเภทซีมอสที่ทำหน้าที่เป็นอิเล็กทรอนิกส์สวิตช์ภายในประกอบด้วยสวิตช์ 4 ตัว ซึ่งนำกระแสได้ 2 ทิศทาง สวิตช์แต่ละตัวถูกควบคุม (การเปิดปิดสวิตช์) ด้วยขาควบคุมขาเดียวที่มีอินพุตอิมพีแดนซ์สูงมาก สวิตช์แต่ละตัวมีความต้านทานสูงมาก เมื่อถูกควบคุมให้เปิดวงจร แต่เมื่อถูกควบคุมให้ปิดวงจรจะมีค่าความต้านทานประมาณ 90 โอห์ม

สวิตช์แต่ละตัวสามารถเปิดวงจร (OFF) โดยการต่อขาควบคุมเข้ากับไฟลบ (Vss) หรือกราวนด์ 0 V กรณีใช้แหล่งจ่ายไฟชุดเดียว และสามารถปิดวงจร (ON) โดยการต่อขาควบคุมนี้เข้ากับไฟบวก (Vdd) กรณีของการใช้แหล่งจ่ายไฟ 2 ชุด ให้กับไอซีนี้ค่าแรงดันสูงสุดของแหล่งจ่ายไฟถูกจำกัดไว้ที่ ± 9 V

การใช้อิเล็กทรอนิกส์สวิตช์กับสัญญาณแอนะลอก จะเกิดความผิดเพี้ยนของสัญญาณน้อยกว่าร้อยละ 0.5 สัญญาณที่ใช้กับสวิตช์นี้จะต้องมีแรงดันไม่สูงกว่าแรงดันไฟบวกของแหล่งจ่ายไฟด้วย กรณีที่สวิตช์บางตัวไม่ถูกนำมาใช้งาน จะต้องต่อขาควบคุมเข้ากับไฟบวกหรือไฟลบ หรือจะต่อขาทั้งสามกับไฟลบก็ได้ เพื่อป้องกันไม่ให้สวิตช์เหล่านั้นทำงานเอง

เนื่องจากไอซีเบอร์นี้มีคุณลักษณะสมบัติที่นำมาประยุกต์ใช้งานร่วมกับวงจรขับสัญญาณของโครงงานนี้ เพื่อทำหน้าที่เป็นตัวปิดและเปิดวงจรการทำงานที่จะนำไปควบคุมตัว Plant งานอีกที่ซึ่งจะมีการควบคุมโดยผ่านทางพอร์ตการทำงานของตัวการ์ด ET – PCDIO ที่ใช้ร่วมกับโครงงาน

2.9 ET – DIGITAL INPUT/OUTPUT CARD

2.9.1 ลักษณะทั่ว ๆ ไปของ ET – DIO CARD

ET – DIO CARD เป็นลักษณะของ PC CARD ที่เชื่อมต่อกับเครื่อง PC เพื่อขยายระบบอินพุตและเอาต์พุต ให้ใช้งานได้มากยิ่งขึ้นซึ่ง ET – DIO CARD สามารถที่จะรับสัญญาณอินพุตและให้สัญญาณเอาต์พุตออกมาได้ทั้งในรูปของแอนะลอก และ ดิจิตอล ทำให้มีความอ่อนตัวในการนำไปประยุกต์ใช้งานในด้านต่าง ๆ ได้มากยิ่งขึ้นซึ่ง ET – DIO CARD มีอุปกรณ์ร่วมและมีจุดเด่น ๆ ของตัวมันเองดังนี้คือ

- มี ไอซี 8255 (Programmable Pheripheral Interface) จำนวน 1 ตัว ซึ่งสามารถที่จะใช้โปรแกรมให้เป็นได้ทั้งอินพุตและเอาต์พุตตามความต้องการของผู้ใช้เองได้ทั้งหมด 3 พอร์ต หรือ 24 บิต I/O ในรูปของสัญญาณ Digital นั่นเอง

- มี ไอซี 8253 (Programmable Interval Timer) จำนวน 1 ตัวทำให้ ET – DIO CARD สามารถที่จะประยุกต์ใช้งานเกี่ยวกับระบบฐานเวลาต่าง ๆ ได้มากมาย เช่น

1. โปรแกรมให้ทำงานเป็นวงจรสร้างฐานเวลาแบบต่าง ๆ
2. โปรแกรมให้เป็นวงจรนับแบบต่าง ๆ
3. โปรแกรมให้เป็นวงจรสร้างสัญญาณ Interrupt ให้เครื่อง PC
4. โปรแกรมให้เป็นวงจรสร้างสัญญาณ Square Wave
5. โปรแกรมให้เป็นวงจรสร้างความถี่

ซึ่ง IC 8253 นี้มีโครงสร้างภายในให้ใช้งานถึง 3 แชนแนล และแต่ละแชนแนล ทำงานแยกจากกันอย่างอิสระ

- มี ไอซี ADC (Analog to Digital Converter) จำนวน 1 ตัว ซึ่งสามารถที่จะเลือกใช้ได้ถึง 2 เบอร์ คือ ADCO804 (8 บิต) หรือ ADC1001 (10 บิต) ซึ่งทำให้ ET – DIO CARD สามารถที่จะประยุกต์ใช้งานในการตรวจจับ หรือการวัดสัญญาณหรือรับสัญญาณอินพุต ในรูปของสัญญาณแอนะลอก ได้ 1 channal ซึ่งขนาดของสัญญาณแอนะลอก ที่รับเข้ามาสามารถที่จะรับได้โดยตรงสูงสุดถึง 5 V DC หรือมากกว่าโดยเพิ่มวงจรขยายย่านวัดเข้าไปอีกเล็กน้อย ทำให้ ET – DIO CARD สามารถประยุกต์ใช้เป็นเครื่องมือวัดสัญญาณต่าง ๆ ได้มากมาย เช่น

1. เครื่องวัดแรงดัน
2. เครื่องวัดกระแส
3. เครื่องวัดอุณหภูมิ
4. เครื่องวัดความต้านทาน
5. เครื่องวัดความจุ

ซึ่งผลของการวัดแบบต่าง ๆ สามารถที่จะนำมาเก็บเป็นแฟ้มข้อมูลหรือแสดงผลในรูปแบบอื่น ๆ ได้ตามต้องการโดยผู้ใช้เขียนโปรแกรมควบคุมเอง

- มี ไอซี DAC (Digital to Analog Converter) จำนวน 1 ตัว ซึ่งสามารถเลือกใช้ได้ถึง 2 เบอร์ คือ DACO832 (8 บิต) หรือ DAC1232 (12 บิต) ซึ่งเลือกได้โดยการ Set Jumper JP1 ทำให้ ET – DIO CARD สามารถที่จะประยุกต์ใช้ในงานควบคุมต่างๆ โดยส่งสัญญาณในรูปของแอนะลอกออกไปควบคุมอุปกรณ์ภายนอก ซึ่งขนาดของสัญญาณแอนะลอก มีวงจร OP – AMP เพื่อขยายขนาดของสัญญาณอยู่แล้วซึ่งผู้ใช้สามารถปรับขนาดของสัญญาณได้ตั้งแต่ OV – 10.66 VDC

- มีวงจรถอดรหัสตำแหน่งของพอร์ตที่จะใช้งาน ทำให้สะดวกในการเปลี่ยนแปลงตำแหน่งของพอร์ต ที่จะใช้งานได้ง่ายโดยการ SET DIP-SWITCH ทำให้มีความอ่อนตัวในการใช้งาน และสามารถที่จะนำ ET – DIO CARD ต่อร่วมกับเครื่อง PC ได้มากว่า 1 CARD โดยกำหนดพอร์ตใช้งานที่แตกต่างกัน

- มี Working Arear มากถึง 5 CM x 9 CM ทำให้มีพื้นที่ว่างใช้งานมากยิ่งขึ้นจึงทำให้ผู้ใช้ สามารถที่จะประยุกต์ใช้งานต่อวงจรหรือเพิ่มเติมอุปกรณ์ต่าง ๆ ได้มากมายและสะดวกยิ่งขึ้น

2.9.2 การ DECODE PORT

ตำแหน่งของพอร์ต บน ET – DIO CARD จะใช้ IC TTL 74LS688 (U3) 74LS139 (U4) 74LS32 (U7) SWITCH1 เป็นตัวกำหนดเบอร์ พอร์ต ตามความต้องการของผู้ใช้ โดยใน ET – DIO CARD จะใช้ตำแหน่งของพอร์ต ทั้งหมด 12 พอร์ต คือ

XX0H = Port PA ของ 8255

XX1H = Port PB ของ 8255

XX2H = Port PC ของ 8255

XX3H = Port Control ของ 8255

XX4H = Port Counter0 ของ 8253

XX5H = Port Counter1 ของ 8253

XX6H = Port Counter2 ของ 8253

XX7H = Port Control ของ 8253

XX8H = Port Control ของ DAC

XX9H = Port Control ของ DAC

XXAH = Port Control ของ ADC

XXBH = Port Control ของ ADC

เราสามารถที่จะกำหนดเบอร์พอร์ต ได้โดยการกำหนดระดับ Logic ให้กับตำแหน่งทำงาน ADDRESS นั้น ๆ ตามความต้องการ ซึ่งบน ET-DIO CARD สามารถที่จะกำหนดระดับของ Logic ให้กับตำแหน่ง ADDRESS ใด ๆ โดยใช้ SWITCH1 ซึ่งหาก ON SWITCH จะได้ระดับ Logic “0” หาก OFF SWITCH จะได้ระดับ Logic “1”

ซึ่งเมื่อเรากำหนดให้ ET-DIO CARD มีค่าตำแหน่งเท่าได้แล้วการใช้งานอุปกรณ์ใด ๆ บน ET-DIO CARD ต้องอ้างตำแหน่งนั้น ๆ เสมอ จะมีผลทำให้เบอร์ พอร์ตเป็นดังนี้คือ

300H = Port PA ของ 8255

301H = Port PB ของ 8255

302H = Port PC ของ 8255

303H = Port Control ของ 8255

304H = Port Counter0 ของ 8253

305H = Port Counter1 ของ 8253

306H = Port Counter2 ของ 8253

307H = Port Control ของ 8253

308H = Port Control ของ DAC

309H = Port Control ของ DAC

30AH = Port Control ของ ADC

30BH = Port Control ของ ADC

2.9.3 การใช้งาน 8255 (Programmbale Pheripheral Interface)

ไอซี 8255 (Programmbale Pheripheral Interface) เป็น ไอซี ประกอบด้วยพอร์ตใช้งานถึง 3 พอร์ต และพอร์ตควบคุม (CONTROL PORT) อีก 1 พอร์ตรวมเป็น 4 พอร์ต ซึ่งไอซี 8255 สามารถที่จะโปรแกรมให้เป็นได้ทั้งอินพุตและเอาต์พุตได้ทั้ง 3 พอร์ต หรือ 24 บิต I/O โดยการกำหนดที่พอร์ตควบคุม (CONTROL PORT) ซึ่งการโปรแกรมเพียงแต่ส่งค่า Control Word Code ไปให้พอร์ตควบคุมเพื่อกำหนดการทำงานของ 8255 ซึ่งมีรายละเอียดและตัวอย่างการโปรแกรมดังนี้คือ

DO ใช้สำหรับกำหนดการทำงานของพอร์ต C ล่าง (PCO – PC3) คือ

ถ้าเป็น 1 หมายถึงให้เป็น Input

ถ้าเป็น 0 หมายถึงให้เป็น Output

D1 ใช้สำหรับกำหนดการทำงานของพอร์ต B (PBO 0 PB7) คือ

ถ้าเป็น 1 หมายถึงให้เป็น Input

ถ้าเป็น 0 หมายถึงให้เป็น Output

D2 ใช้สำหรับกำหนดโหมดการทำงานของพอร์ต C ล่าง และพอร์ต B คือ

ถ้าเป็น 0 หมายถึงให้พอร์ต C ล่าง และพอร์ต B ทำงานในโหมด 0

ถ้าเป็น 1 หมายถึงให้พอร์ต C ล่าง และพอร์ต B ทำงานในโหมด 1

D3 ใช้สำหรับกำหนดการทำงานของพอร์ต C บน (PC4 –PC7) คือ

ถ้าเป็น 1 หมายถึงให้เป็น Input

ถ้าเป็น 0 หมายถึงให้เป็น Output

D4 ใช้สำหรับกำหนดการทำงานของพอร์ต A บน (PCO –PC7) คือ

ถ้าเป็น 1 หมายถึงให้เป็น Input

ถ้าเป็น 0 หมายถึงให้เป็น Output

D6, D5 ใช้สำหรับกำหนดโหมดการทำงานของพอร์ต C บนและพอร์ต A คือ

ถ้าเป็น 00 หมายถึง ให้พอร์ต C บนและพอร์ต A ทำงานในโหมด 0

ถ้าเป็น 01 หมายถึง ให้พอร์ต C บนและพอร์ต A ทำงานในโหมด 1

ถ้าเป็น 1X หมายถึง ให้พอร์ต C บนและพอร์ต A ทำงานในโหมด 2

D7 ใช้สำหรับกำหนด MODE SET FLAG คือ

ถ้าเป็น 1 หมายถึงACTIVE ซึ่งต้องกำหนดให้บิตนี้เป็น 1 เสมอ

ถ้าเป็น 0 หมายถึง NON-ACTIVE

2.9.4 การใช้งาน 8253 (Programmable Interval Timer)

ไอซี 8253 (Programmable Interval Timer) เป็นไอซีซึ่งประกอบด้วย พอร์ตใช้งาน 3 พอร์ต และพอร์ตควบคุมการทำงาน (Control Port) อีก 1 พอร์ต รวมเป็น 4 พอร์ต ซึ่ง 8253 เหมาะสำหรับการใช้งานในด้านฐานเวลาต่าง ๆ ซึ่งมีอินพุต 2 อินพุต (CLK และ GATE) และเอาต์พุต 1 เอาต์พุต (OUT) ต่อ 1 แชนแนล ซึ่งใน 8253 มีให้ใช้งานถึง 2 แชนแนล และแต่ละแชนแนลยังแยกการทำงานกันอย่างอิสระ 8253 แต่ละแชนแนลสามารถเลือกการทำงานได้ 6 โหมด การโปรแกรมให้ 8253 ทำงานในโหมดใดนั้นทำได้โดยการส่งค่า Control Word ให้กับ Register Mode Control (Port Control 8253) ซึ่งมีรายละเอียดการโปรแกรมดังนี้คือ

ตารางที่ 2.8 แสดงรายละเอียดของบิตต่างๆ

บิต	D7	D6	D5	D4	D3	D2	D1	D0
หน้าที่	SC1	SCO	RL1	RL0	M2	M1	M0	BCD

บิต D4 ,D6 (SC1, SC0) :

Select Counter ใช้สำหรับเลือกแชนแนลที่ต้องการ

ตารางที่ 2.9 แสดงการเลือกแชนแนล

SC1	SC0	แชนแนลที่ถูกเลือก
0	0	แชนแนล 0
0	1	แชนแนล 1
1	0	แชนแนล 2
1	1	-

บิต D5 , D4 (RL1 , RL0) :

Read/load ใช้สำหรับกำหนดไบท์ในการอ่าน / เขียนข้อมูล

ตารางที่ 2.10 แสดงการกำหนดพอร์ตในการอ่าน/เขียนข้อมูล

RL1	RL0	หน้าที่
0	0	ทำการแลทซ์ค่าในรีจิสเตอร์-เค้าน์เตอร์
0	1	อ่าน /เขียน เฉพาะข้อมูลใน 8 บิต ล่าง (LSB)
1	0	อ่าน /เขียน เฉพาะข้อมูลใน 8 บิต บน (MSB)
1	1	อ่าน /เขียน ข้อมูลทั้ง 16 บิต โดยเริ่มจาก 8 บิตล่าง ก่อน จากนั้นจึงอ่าน/เขียนข้อมูลใน 8 บิตบน

บิต D3 ,D2, D1 (M2, M1, M0) :

Mode ใช้สำหรับเลือกโหมดการทำงานของ 8253 คือ

ตารางที่ 2.11 การเลือกโหมดการทำงานของ 8253

M2	M1	M0	โหมดการทำงาน
0	0	0	โหมด 0 : Interrupt On Terminal
0	0	1	โหมด 1 : โปรแกรมmable One-Shot
X	1	0	โหมด 2 : Rate Generator
X	1	1	โหมด 3 : Square Wave Generator
1	0	0	โหมด 4 : Software Trigger Stobe
1	0	0	โหมด 5 : Hardware Trigger Stobe

ตารางที่ 2.11 (ต่อ) การเลือกการทำงานของ 8253

M2	M1	M0	โหมดการทำงาน
1	0	1	โหมด 5 : Hardware Trigger Stobe

BCD : ใช้กำหนดการลดค่าข้อมูลในรีจิสเตอร์เคาน์เตอร์ กล่าวคือ

ถ้าเป็น 1 ค่าของข้อมูลในรีจิสเตอร์เค้าน์เตอร์จะถูกลดลงแบบ BDC

ถ้าเป็น 0 ค่าของข้อมูลในรีจิสเตอร์เคาน์เตอร์จะถูกลดลงแบบ Binary

2.9.5 การใช้งาน ADC (Analog to Digital Converter)

ไอซี ADC0804 เป็นไอซี ทำหน้าที่รับสัญญาณอินพุตในรูปของแอนะลอก แล้วเปลี่ยนเป็น Digital เพื่อส่งให้ CPU ประมวลผล ซึ่งสามารถที่จะประยุกต์ใช้งานในการเชื่อมต่อ (Interface) กับอุปกรณ์ภายนอกที่ให้สัญญาณเป็น แอนะลอก ซึ่ง ET-DIO CARD สามารถรับสัญญาณ แอนะลอกได้โดยตรงถึง 5 VDC หรือมากกว่า โดยใช้เพียงเพิ่มเติมวงจรขยายย่านวัด เช่นเดียวกับเครื่องวัดทั่ว ๆ ไป ซึ่งขนาดของสัญญาณแอนะลอก ที่รับเข้ามานั้น สามารถกำหนดแถบความกว้าง (Range) ของสัญญาณได้โดยการควบคุมขนาดของ V1 (–) และ Vref/2 ซึ่ง V1(-) จะเป็นตัวกำหนดจุดเริ่มต้นหรือค่าต่ำสุดของสัญญาณที่รับเข้ามาโดยบน ET-DIO CARD สามารถเลือกได้โดย JP4 คือถ้า Short JP4 ที่ตำแหน่ง 1-2 (Ground) จะทำให้สัญญาณเริ่มต้นจาก OV จนถึง Maximum (ไม่ควรเกิน 5V) แต่ถ้าShort JP4 ที่ตำแหน่ง 2-3 จุดเริ่มต้นของสัญญาณจะขึ้นอยู่กับขนาดของสัญญาณ V1(-) จากภายนอกที่ต่อมาจาก Connector CN2 ซึ่งต้องกำหนดเองซึ่งโดยปรกติแล้วต้องมีค่ามากกว่า OV แต่ต่ำกว่า Vref/2 เสมอ ส่วนสัญญาณ Vref/2 จะเป็นตัวกำหนดขนาดสูงสุดของสัญญาณที่รับเข้ามา (Maximum) นั่นคือ

- ถ้าสัญญาณ แอนะลอก ที่รับเข้ามามีค่าเป็น 2 เท่าของ Vref/2 จะได้ DATA = FFH
- ถ้าสัญญาณ แอนะลอก ที่รับเข้ามามีค่าเป็น 1 เท่าของ Vref/2 จะได้ DATA = 7FH
- ถ้าสัญญาณ แอนะลอก ที่รับเข้ามามีค่าเท่ากับ V1 จะได้ DATA = OOH

ซึ่งอัตราการเปลี่ยนแปลงของ DATA ที่มีต่อขนาดของสัญญาณนั้นจะเป็นเชิงเส้นตลอด (Linear) ซึ่งขนาดของสัญญาณ Vref/2 สามารถเลือกได้โดย JP5 กล่าวคือถ้าผู้ใช้ Short ที่ตำแหน่ง 1-2 จะได้ Vref/2 = 2.50V ถ้า Short ที่ตำแหน่ง 2-3 ขนาดของสัญญาณ Vref/2 จะขึ้นอยู่กับขนาดที่ผู้ใช้กำหนดเองจากภายนอกที่ต่อมาจาก Connecter CN2 ซึ่งขนาดของ Vref/2 ที่ป้อนเข้ามาต้องมีค่ามากกว่า OV และไม่ควรเกิน 2.50V ด้วยเช่นกัน

ตัวอย่างเช่นเลือก V1(-) เท่ากับ OV (JP4 Short ที่ตำแหน่ง 1-2) และเลือก Vref/2 เท่ากับ 2.50V (JP5 Short) ที่ตำแหน่ง 1-2 แล้ว จะได้ว่าช่วงของสัญญาณ แอนะลอกอินพุตจะต้องอยู่ระหว่าง OV – 5.0V เท่านั้นซึ่งความละเอียดของแต่ละช่วงสัญญาณที่รับเข้ามามีความละเอียดถึง 256 ระดับ นั่นคือ

- ถ้าสัญญาณ แอนะลอก Input มีค่าเท่ากับ OV จะได้ DATA เท่ากับ OOH
- ถ้าสัญญาณ แอนะลอก Input มีค่าเท่ากับ 2.50V จะได้ DATA เท่ากับ 7FH
- ถ้าสัญญาณ แอนะลอก Input มีค่าเท่ากับ 5.0V จะได้ DATA เท่ากับ FFH

ดังนั้นจะได้ความละเอียดของสัญญาณ = (0V – OV) / 256

= 0.0195 V

หรืออาจกล่าวได้ว่า สามารถรับสัญญาณได้ตั้งแต่ OV –5.0V โดยมีความแตกต่างของแต่ละช่วง (Step) เป็น 0.0195V

หรืออีกกรณีหนึ่งคือ หากเลือก V1- เท่ากับ 0.50 V จากภายนอกซึ่งต่อเข้ามาทาง Connector CN2 (JP4 Short ที่ตำแหน่ง 2-3) และเลือก Vref/2 เท่ากับ 1.50V (JP5 Short ที่ตำแหน่ง 2-3) แล้ว จะได้ว่าช่วงของสัญญาณแอนะลอก Input จะต้องอยู่ระหว่าง 0.50V – 3.50V นั่นเอง

สำหรับ ไอซี A to D บน ET-DIO CARD สามารถเลือกโหมดการทำงานได้ทั้งหมด 2 โหมดการทำงาน โดยการเลือกที่ Jumper JP2 คือ

- โหมดทำงานปรกติ (Free Run Mode)
- โหมดอินเทอร์รัพต์ (Interrupt Mode)

2.9.6 การใช้งาน DAC (Digital To Analog Converter)

ไอซี DAC มีลักษณะการทำงานที่ตรงกันข้ามกับไอซี ADC กล่าวคือมันจะทำหน้าที่เปลี่ยนสัญญาณดิจิตอล ให้เป็นสัญญาณแอนะลอก ซึ่งนิยมใช้งานในด้านการควบคุมต่าง ๆ ซึ่งบน ET-DIO CARD ไอซี DAC ให้ใช้งาน 1 ตัว โดยสามารถที่จะเลือกใช้ได้ 2 เบอร์ คือ DAC0832 (8 Bits) หรือ DAC1232 (12 Bits) ตัวใดตัวหนึ่งโดยการเลือกที่ Jumper JP1

ซึ่งการควบคุมขนาดของสัญญาณ แอนะลอกเอาต์พุต นั้นทำได้โดยการส่งค่า DATA ออกไปยังพอร์ตควบคุมของไอซี DAC (Port Control DAC) ซึ่งอัตราส่วนของสัญญาณแอนะลอกเอาต์พุตที่ได้นั้นจะเปลี่ยนแปลงอย่างเป็นเชิงเส้น (Linear) นั่นคือ

- ถ้าส่งค่า DATA ค่า FFH จะได้ขนาดสัญญาณ Output สูงสุด
- ถ้าส่งค่า DATA ค่า 7FH จะได้ขนาดสัญญาณ Output ครึ่งหนึ่ง
- ถ้าส่งค่า DATA ค่า 00H จะได้ขนาดสัญญาณ Output ต่ำสุด

ซึ่งขนาดของสัญญาณ แอนะลอกเอาต์พุต นั้นมีความละเอียดถึง 256 ค่าจากย่านความกว้างของสัญญาณทั้งหมด (Range) หรือคำนวณได้จากสูตร

ความละเอียดของช่วงสัญญาณ = (Vmax-Vmin) / 256

กลุ่มของสัญญาณ GATE ใช้เลือกว่าจะให้สัญญาณ GATE ของ 8253 ซึ่งมีเลือกทั้งหมด 3 Channal (GATE O , GATE 1 และ GATE 2) ทำงานตลอดเวลาตามสัญญาณนาฬิกาที่ป้อนเข้ามาเพียงอย่างเดียว (เหมาะที่จะใช้กับกรณีที่โปรแกรมให้ 8253 ทำงานสร้างสัญญาณ Interrupt หรือโหมดอื่น ๆ ที่ไม่ต้องการควบคุมจากภายนอก) หรือจะให้สัญญาณ GATE ถูกควบคุมจากสัญญาณภายนอก การ SET ทำได้ดังนี้คือ

ตารางที่ 2.12 การควบคุมสัญญาณ GATE

Channal Output	GATE = VCC. Signal Jumper JP7 SET		GATE = EXT. Signal Jumper JP7 SET	
Channal 0 (GATE0)	SHORT 10-11	OPEN 11-12	SHORT 11-12	OPEN 10-11
Channal 1 (GATE1)	SHORT 13-14	OPEN 14-15	SHORT 14-15	OPEN 13-14
Channal 2 (GATE2)	SHORT 16-17	OPEN 17-18	SHORT 17-18	OPEN 16-17

CN 1 เป็น Connector ขนาด 34 Pin มาตรฐาน ETT ใช้เป็นจุดเชื่อมของสัญญาณ อินพุต / เอาต์พุต ระหว่าง 8255 และ อุปกรณ์ภายนอก

CN2 เป็น Connector ขนาด 20 Pin ใช้เป็นจุดเชื่อมต่อของสัญญาณระหว่าง 8253 , ADC และ DAC กับอุปกรณ์ภายนอก

2.9.7 การปรับแต่งแรงดันอ้างอิง (2.50V บน ET-DIO CARD)

ET-DIO CARD มีวงจรควบคุมแรงดันอ้างอิงซึ่งประกอบอยู่ในตัวเรียบร้อยแล้วซึ่งใช้ไอซี LM336 เป็นอุปกรณ์หลักโดยมีตัวต้านทานปรับค่าได้ (VR1) เป็นตัวปรับแต่งค่าแรงดันอ้างอิง ซึ่งขนาดของแรงดันอ้างอิงนี้กำหนดให้มีขนาดเท่ากับ 2.50 V โดยใช้เป็นค่าแรงดันอ้างอิง ของไอซี

DAC และไอซี ADC ซึ่งการปรับแต่งค่าแรงดันอ้างอิงนี้ทำได้โดยการใช้โวลต์มิเตอร์วัดเทียบระหว่าง GND และจุดกำเนิดแรงดันอ้างอิง 2.50V (ขา 8 ของ DAC0832 หรือ ขา 1 ของ JP5) แล้วทำการปรับ VR1 จนได้แรงดัน 2.50 VDC พอดี

** หมายเหตุ** ถ้าค่าแรงดันอ้างอิงนี้ไม่เที่ยงตรงจะมีผลต่อค่าพารามิเตอร์ต่าง ๆ ที่เกี่ยวกับไอซี DAC และไอซี ADC อาจทำให้ผลการวัดต่าง ๆ ได้ค่าที่ผิดพลาดตามไปด้วย

2.9.8 การปรับแต่งขนาดของสัญญาณแอนะลอก

สัญญาณแอนะลอกที่ได้จาก ไอซี D / A มีวงจร OP - AMP เพื่อทำการขยายให้มีขนาดสัญญาณสูงขึ้นเพื่อให้เกิดความสะดวกในการใช้งานด้านต่าง ๆ ได้ดีขึ้น ซึ่งขนาดของสัญญาณนี้สามารถกำหนดและปรับแต่งได้ตั้งแต่ OV = 10.6V โดยประมาณ การปรับแต่งขนาดของสัญญาณแอนะลอก ของ D / A ทำได้โดย

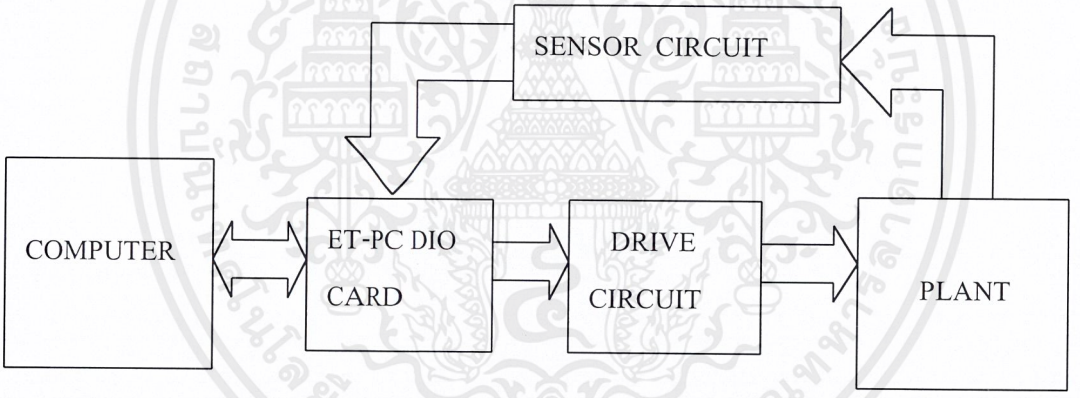
1. กำหนดค่าของสัญญาณสูงสุดที่ต้องการใช้
2. ส่งค่า DATA = FFH ให้ Port Control ของ DAC (XX8 OR XX9)
3. ปรับ VR2 ให้ได้ขนาดของสัญญาณตามต้องการ

** หมายเหตุ** การปรับขนาดของสัญญาณแอนะลอก ต้องทำหลังการจากปรับแต่งแรงดันอ้างอิง Vref 2.50VDC เรียบร้อยแล้วจึงจะได้ผลที่แน่นอน

บทที่3
การออกแบบ การสร้างและการทำงาน

3.1 กล่าวนำ

ในการออกแบบระบบควบคุมระดับด้วยโปรแกรมไมโครคอมพิวเตอร์ได้แบ่งออกเป็นสามส่วนที่สำคัญ คือ ส่วนของโปรแกรม ส่วนของวงจรขับสัญญาณ และส่วนที่เป็นตัว Plant ของงาน ซึ่งส่วนของโปรแกรมจะประกอบไปด้วยส่วนของการเขียนโปรแกรมควบคุมการทำงานทั้งหมด ที่มีตัวการ์ด ET - PCDIO เพื่อเป็นตัวรับและส่งสัญญาณไปควบคุมในส่วนของวงจรขับสัญญาณ และส่วนของตัว Plant ชุดประยุกต์ใช้งานควบคุมระดับน้ำอีกที ซึ่งได้แสดงโครงสร้างการทำงานของระบบควบคุมระดับด้วยโปรแกรมไมโครคอมพิวเตอร์ ดังรูปที่ 3.1 ดังนี้



รูปที่ 3.1 แสดงโครงสร้างการทำงานของชุดประยุกต์ใช้งานการควบคุมระดับ

3.2 ทฤษฎี Signal Condition

เนื่องจากสัญญาณที่ส่งมาจากอุปกรณ์ทรานสดิวเซอร์หรือสัญญาณที่ส่งมาให้กับอุปกรณ์ที่จะควบคุมเป็นสัญญาณแอนะลอก ดังนั้นเมื่อมีการควบคุมด้วยไมโครคอนโทรลเลอร์ซึ่งมีอยู่ภายในตัวการ์ด จึงต้องทำสัญญาณให้อยู่ในรูปแบบของสัญญาณดิจิตอลเพื่อนำไปสั่งให้ตัวการ์ดทำงาน ซึ่งการทำสัญญาณแอนะลอกให้เป็นสัญญาณดิจิตอล ทำได้โดยการใช้ IC Analog to Digital เบอร์ ADC 0804 ซึ่งสัญญาณแอนะลอกที่เข้ามาทางอินพุตของ A to D เบอร์นี้จะอยู่ในรูปแบบของแรงดันที่มีค่าตั้งแต่

0 – 5 V ดังนั้นจึงต้องมีวงจรที่ทำหน้าที่กำหนดเงื่อนไขของสัญญาณที่รับเข้ามาจากอุปกรณ์ทรานสดิวเซอร์และส่งสัญญาณเอาต์พุตให้กับ A / D ของวงจรนั้นคือ Signal Condition Circuit ซึ่งภายในวงจรนี้มีวงจรที่ทำงานร่วมกันอยู่ 3 ชนิดคือ

- วงจรขับสัญญาณโซลินอยด์วาล์ว (DRIVER SOLINOID VALVE CIRCUIT)
- วงจรตรวจจับสัญญาณ (SENSOR CIRCUIT)
- วงจร Instrument Amplifier

3.2.1วงจรขับสัญญาณโซลินอยด์วาล์ว (DRIVER SOLINOID VALVE CIRCUIT)

วงจรขับสัญญาณ เป็นวงจรที่ทำหน้าที่ขยายสัญญาณค่าแรงดันและค่ากระแส โดยในโครงงานนี้จะนำไปใช้ในการขับสัญญาณให้กับตัว Solinoid Valve ที่ทำหน้าที่เป็นตัวเปิดและตัวปิดน้ำที่ไหลผ่านจากแทงก์น้ำหนึ่งไปยังอีกแทงก์น้ำหนึ่งทำงาน โดยในวงจรนี้จะรับค่าคำสั่งและข้อมูลจากคอมพิวเตอร์โดยเฉพาะ เป็นตัวควบคุมวงจรขับสัญญาณอีกที

การออกแบบของวงจรนี้จะใช้ตัว RELAY ที่รับแรงดันขนาด 5 V และทนกระแสได้ขนาด 10 A เป็นตัวช่วยขับสัญญาณที่หน้าสัมผัสของตัว RELAY เพื่อเป็นสวิตซ์เปิดและปิดให้แรงดัน 220 โวลต์ผ่านไปสั่งให้ตัว Solinoid Valve ทำงานและใช้ตัว IC เบอร์ ULN 2803 เป็นตัวขยายกระแสที่ส่งมาจากตัววงจรคอนโทรลเลอร์จ่ายให้กับตัว RELAY ทำงานอีกที โดยที่ตัว IC เบอร์นี้จะป้องกันการไหลย้อนกลับของกระแสให้กับตัว RELAY อีกด้วย

การป้อนสัญญาณให้กับตัว RELAY เพื่อสั่งให้ Solinoid valve ทำงานนั้น จะรับสัญญาณที่ส่งมาจากตัวไมโครคอนโทรลเลอร์ เบอร์ AT 89H8252 ซึ่งจะส่งสัญญาณ Logic 0 เป็นสภาวะ ON และส่งสัญญาณ Logic 1 ให้เป็นสภาวะ OFF สั่งผ่าน IC เบอร์ ULN 2803 ให้กับ RELAY ไปสั่งให้ตัว Soliniod Valve ทำงานในสภาวะเปิดหรือสภาวะปิด ซึ่งสภาวะเปิดจะทำงานที่แรงดัน 5 V และสภาวะปิดจะทำงานที่แรงดัน 0 V ดังรูปที่ 3.1

นอกจากจะขับสัญญาณให้โซลินอยด์วาล์วทั้งหมดภายในตัว Plant แล้วยังช่วยขับให้ตัวปั๊มน้ำทำงานอีกซึ่งจะถูกควบคุมด้วยคอมพิวเตอร์เช่นกัน

3.2.2 วงจรตรวจจับสัญญาณ (SENSOR CIRCUIT)

วงจรตรวจจับสัญญาณ เป็นวงจรที่ทำหน้าที่ตรวจจับสัญญาณจากตัวเซนเซอร์ระดับเบอร์ MPX10AP ซึ่งเป็นเซนเซอร์ตรวจจับระดับน้ำ (D - P Cell) ในโครงงานนี้เซนเซอร์จะตรวจจับสัญญาณภายในแทงก์น้ำที่ 1 (T1) และแทงก์น้ำที่ 2 (T2) ในสถานะที่มีน้ำไหลเข้าไปภายในแทงก์ทั้งสองเพื่อตรวจเชคน้ำภายในแทงก์อยู่ในระดับที่เท่าใด แล้วทำการส่งสัญญาณไปให้ตัววงจร Instrument Amplifier ขยายสัญญาณต่อและทำการส่งสัญญาณเป็นสัญญาณแอนะลอกไปให้ตัววงจรแปลงสัญญาณแอนะลอกเป็นสัญญาณดิจิตอล ส่งผ่านตัวการ์ดไปประมวลผลที่คอมพิวเตอร์

ส่งเพื่อแสดงผลต่อไป นอกจากจะส่งสัญญาณแอนะลอกไปแปลงสัญญาณเป็นดิจิตอลแล้ว ยังต้องส่งสัญญาณแอนะลอกผ่านตัววงจร Instrument Amplifier ให้กับตัวการ์ด ET - DIO CARD เพื่อไปประมวลผลเช่นกัน

การออกแบบในวงจรส่วนนี้จะใช้ตัวเซนเซอร์ทำงานร่วมกับวงจร Instrument Amplifier โดยตัวเซนเซอร์จะตรวจจับภายใน T1 และ T2 ในส่วนที่น้ำผ่านเข้ามาให้อยู่ในระดับที่ตั้งค่าไว้ตามต้องการ และจะตรวจจับระดับน้ำที่ส่งมาใน T1 ในกรณีที่มีน้ำจ่ายเข้ามามากเกินไปหรือมีระดับน้ำที่เข้ามาน้อยเกินไป เพื่อส่งสัญญาณไปควบคุมการทำงานของ VALVE ต่าง ๆ ให้อยู่ในสภาวะ ON หรือ OFF เนื่องจากตัวเซนเซอร์ที่ใช้ในโครงงานนี้จะทำหน้าที่ดังต่อไปนี้คือ

- ตรวจสอบระดับน้ำเพื่อรักษาระดับน้ำให้คงที่ภายในตัวแทงก์น้ำ
- ส่งสัญญาณให้กับวงจรคอนโทรลเลอร์เพื่อนำไปแสดงผล
- ตรวจสอบระดับน้ำที่ต้องการภายในแทงก์

3.2.3 วงจร Instrument Amplifier

Instrument Amplifier เป็นวงจรขยายสัญญาณแบบหนึ่งที่ใช้วัดสัญญาณจากทรานสดิวเซอร์ แบบต่าง ๆ โดยสัญญาณที่รับมาขยายนั้นจะมีความถูกต้องสูงมีความเพี้ยนต่ำ

ดังนั้น Instrument Amplifier จึงต้องมีคุณสมบัติเป็นวงจรขยายในอุดมคติเมื่อพิจารณาระบบควบคุมแล้วพบว่าใช้ในความถี่ต่ำ คือ ตั้งแต่ 0 Hz ถึง 500 KHz ดังนั้นการนำเอาออปแอมป์มาใช้จึงเป็นอุปกรณ์ขยายสัญญาณที่เหมาะสม ดังนั้นการพัฒนา Instrument Amplifier ให้ใช้งานได้จริงต้องมีคุณสมบัติคือ

1. อิมพีแดนซ์ที่ขาเข้าเป็นอนันต์
2. กระแสออฟเซตทางเข้า และแรงดันออฟเซตขาเข้า
3. อัตราการขยายเป็นอนันต์ พิจารณา Instrument Amplifier ประกอบด้วย 2 ส่วนคือส่วนแรกเป็นส่วนอินพุตอิมพีแดนซ์ที่มีค่าสูง และสามารถกำหนดอัตราการขยายของวงจรได้โดยใช้การปรับ VR เพียงตัวเดียว จะได้เอาต์พุตระหว่างออปแอมป์ทั้งสองตัวตามสูตร

$$Vo = (Via - Vib) [2R / R4 + 1] Rf / Ri$$

ในวงจรที่ออกแบบนั้นจะได้

$$Rf = Ri * Rin = (Via-Vib)R4 = VR$$

$$Vo = Vin [2R / VR + 1]$$

การกำหนดอัตราการขยายของวงจรนี้จะเป็นการกำหนดช่วงของความดันที่ต้องการควบคุม (กำหนด Span)

ส่วนที่ 2 ของ Instrument Amplifier เป็นวงจรขยายความแตกต่างที่มี อัตราการขยายเท่ากับหนึ่ง มีป้อนแบบลบ มีการกำหนดจุดการกำหนดจุดอ้างอิงโดยการปรับระดับไฟตรงที่ป้อนทางด้านขา Non-Inverting ของออปแอมป์เพื่อการกำหนดจุด ZERO ของวงจรดังรูปที่ 3.2

เรากำหนดสัญญาณ Common Mode โดยใช้ Vcom เป็น Via และ Vib ที่ ออปแอมป์ A จะทำให้ V2a = Vcom เพราะฉะนั้น จะไม่มีค่าแรงดันไฟฟ้าตกคร่อม R4 หมายความว่า จะไม่มีกระแสไหลผ่าน R3 และจะทำให้ Va = V2a ดังนั้นจะได้

$$\frac{Va - Vb}{R3 + R4 + R5} = \frac{V2a - V2b}{R4}$$

จากคุณสมบัติของออปแอมป์เราจะได้ว่า V2 = V3

$$\frac{Va - Vb}{R3+R4+R5} = \frac{Via - Vib}{R4}$$

หรือ

$$Va - Vb = \frac{(Via - Vib)R3+R4+R5}{R4}$$

$$I1 = If$$

$$Va - V2 / R1 = V2 - V0 / R1$$

$$VaRfV2Rf = V2R1 - V0R1$$

$$VoR1 = V2R1 - VaRf + V2Rf$$

$$VoR1 = V2(R1 + Rf) - VaRf$$

หา V3

$$V3 = Vb\,(R2 / R1 + R2) \qquad \text{สมการที่ 3.11}$$

กำหนดให้

$$V2 = V3$$

$$VoRi = Vb\,R2\,\frac{(Ri + Rf)}{R1 + R2} - VaRf$$

ให้ $R1 = Ri$

$R2 = Rf$

ได้สมการ

$$VoRi = VbRfVaRf$$

ได้

$$Vo = (Vb - Va)\, Rf / Ri \quad \text{สมการที่ 3.12}$$

จากนั้นกำหนดให้ Va กับ Vb ต่อสลับกันจะได้สมการใหม่

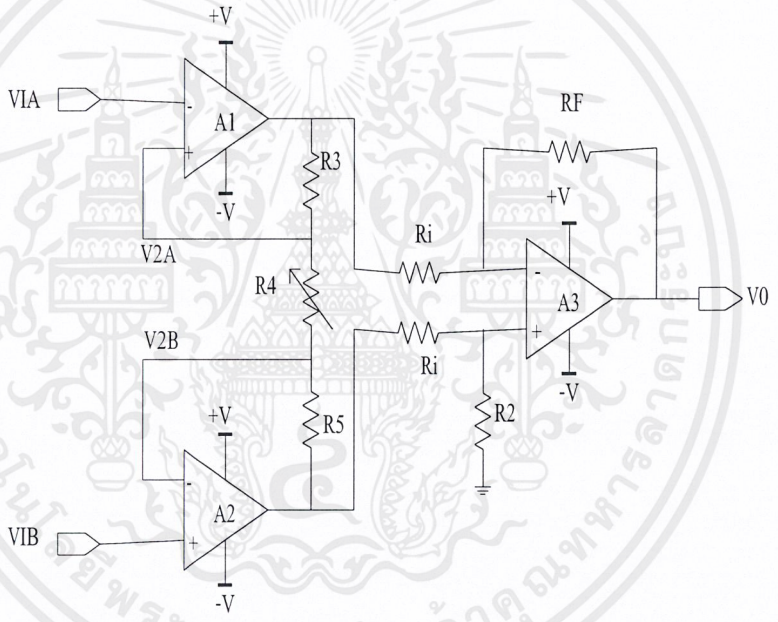
$$Vo = (Va - Vb)\, Rf / Ri \quad \text{สมการที่ 3.13}$$

จะได้ว่า

$$Vo = \frac{(Via - Vib)\,(R3 + R4 + R5)}{R4*Rf/Rin}$$

กำหนดให้ R3 = R4 =R5 จะได้

$$Vo = (Via - Vib)[2R+1]/R4*Rf/Rin$$



รูปที่ 3.2 วงจร Instrument Amplifier

3.3 การออกแบบการเขียนโปรแกรม

3.3.1 ทฤษฎีการสร้างโปรแกรมการทำงาน

เนื่องจากว่าโปรแกรมการทำงานที่ใช้ในโครงงานนี้เป็น โปรแกรมสำเร็จรูป Microsoft Visual C++ V 6.0 มีเครื่องมือในการเขียนโปรแกรมอยู่ในตัวก็คือ AppWizard และ ClassWizard ซึ่งจะใช้ AppWizard ช่วยในการสร้างซอร์สโค้ดต้นแบบขึ้นมา เพื่อให้พัฒนาโปรแกรมได้ง่ายขึ้น ซอร์สโค้ดที่ AppWizard สร้างขึ้น จะใช้คลาสจาก MFC เป็นหลัก ซึ่งอุปกรณ์ตัวนี้มีความหมายดังนี้

AppWizard เป็นอุปกรณ์ตัวหนึ่งที่ใช้สำหรับสร้างซอร์สโค้ดโปรแกรมต้นแบบ ช่วยอำนวนความสะดวกในการเขียนโค้ดโปรแกรมด้วยตนเอง ใช้ง่าย ๆคือ AppWizard จะถามเกี่ยวกับรูปแบบของโปรแกรม , ตัวเลือกที่ต้องการ จากนั้น AppWizard จะสร้างซอร์สโค้ดอัตโนมัติขึ้นมาให้และสามารถใช้งานได้ทันที ตัวเลือกที่ต้องกำหนดให้ AppWizard คือ รูปแบบของโปรแกรมซึ่งมีให้เลือกอยู่ 3 รูปแบบคือ

1.โปรแกรมแบบ SDI (Single Document Interface) เป็นการสร้างโปรแกรมในลักษณะของ Notepad

2.โปรแกรมแบบ MDI (Mutiple Document Interface) เป็นการสร้างโปรแกรมที่มีหลายหน้าต่างคล้ายๆกับ Microsoft Word

3.โปรแกรมแบบ Dialog – Based เป็นการสร้างโปรแกรมแบบไดอะล๊อก เช่น Scandisk , Defrag

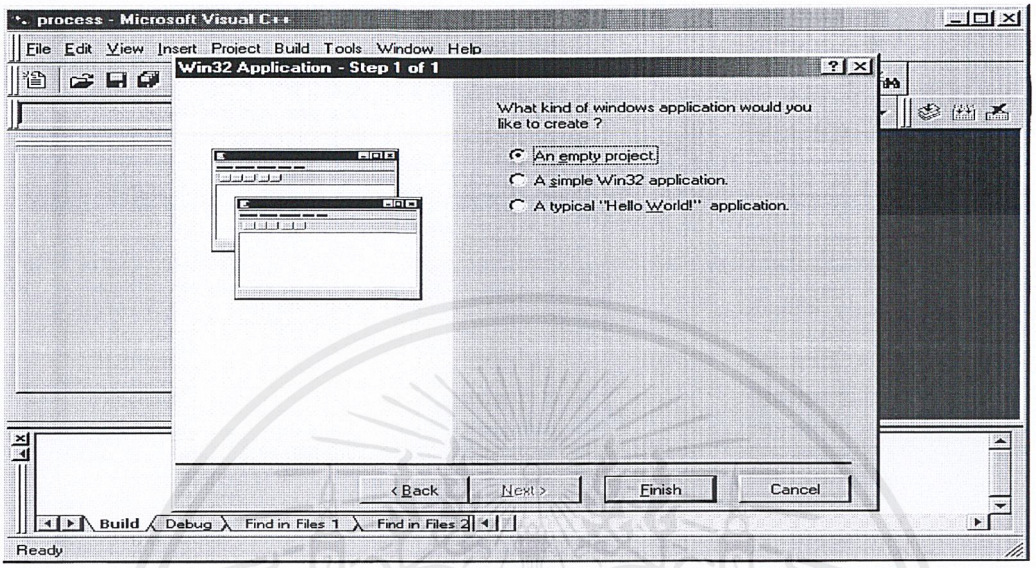
การเลือกโปรแกรมจะต้องพิจารณาการทำงานให้ดี เมื่อเลือกรูปแบบของโปรแกรมที่จะต้องการสร้าง และกำหนดรายละเอียดให้กับ AppWizard เรียบร้อยแล้ว AppWizard ก็จะสร้างซอร์สโค้ดให้ทันที สามารถคอมไพล์และรันโปรแกรมได้ โปรแกรมที่ได้จะเหมือนที่เรากำหนดใน AppWizard ทุกประการ โดยมีการกำหนดให้ส่วนที่เป็นการประกาศคลาสและตัวแปรต่างๆ จะถูกเก็บอยู่ในไฟล์ส่วนหัว (.H) และส่วนที่เป็นเนื้อหาของฟังก์ชั่นจะเก็บไว้ในไฟล์โปรแกรม (.CPP) ซึ่งจะทำให้ง่ายต่อการแก้ไขและดัดแปลงซอร์สโค้ด

3.3.2 การสร้างโปรแกรมการทำงาน

1) ขั้นตอนการสร้างโปรแกรม

1.1) ขั้นตอนที่ 1 สร้างโปรเจ็กต์เวิร์กสเปซใหม่ เลือกเมนู File >New พบไดอะล๊อก New เลือกแท็บ Project ดังรูปที่ 3.3 กำหนดชื่อโปรเจ็กต์และกำหนดไดร์ฟ และไดเร็กทอรี่ที่

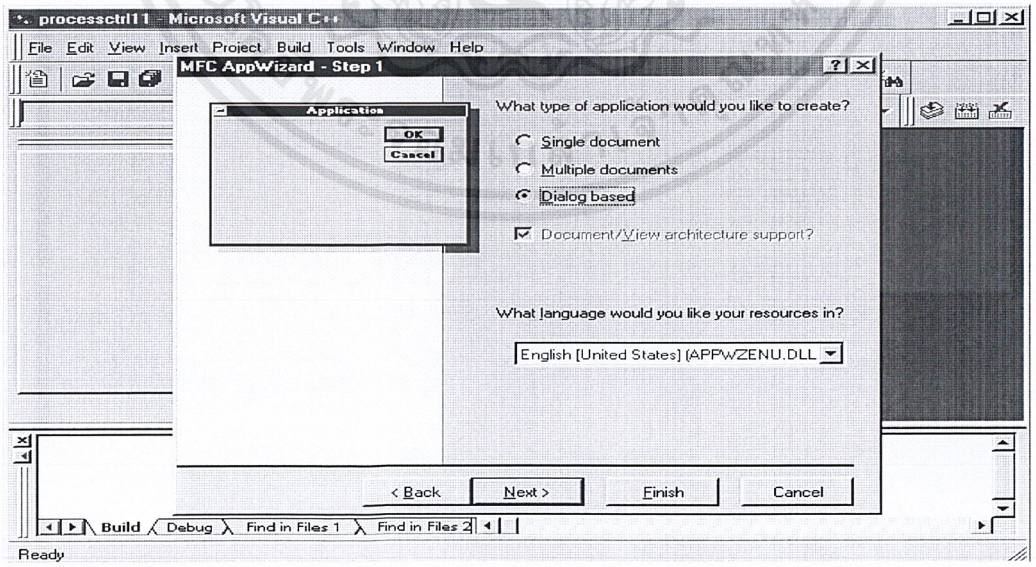
เก็บโปรเจ็กต์ในช่อง Location ซึ่งจะได้ชื่อโปรเจ็กต์ใหม่และสร้างไดเร็กทอรี่ให้กับโปรเจ็กต์โดยอัตโนมัติ



รูปที่ 3.3 แสดงการเลือกสร้างโปรแกรมเข้าสู่การทำงานของ AppWizard

1.2) ขั้นตอนที่ 2 เลือกรูปแบบของโปรแกรม

เลือกรูปแบบการทำงานแบบง่ายๆ เพื่อทดสอบการทำงาน และสามารถทดสอบส่วนโปรแกรมได้ง่าย แล้วกดปุ่ม next เพื่อเข้าสู่กระบวนการต่อไป ดังรูปที่ 3.4



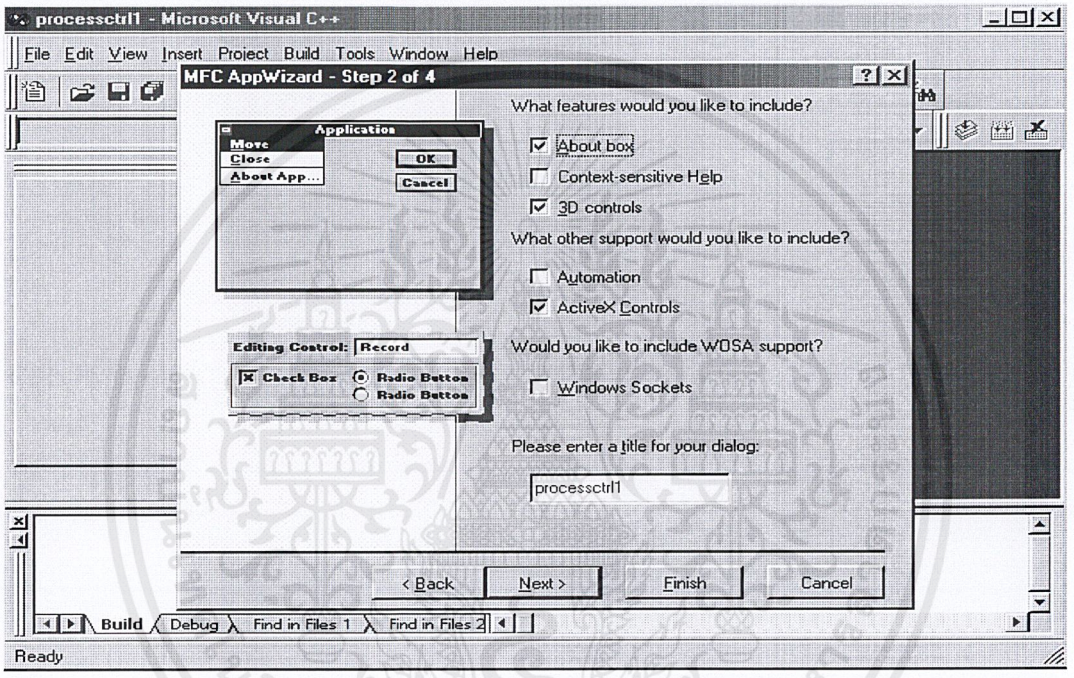
รูปที่ 3.4 แสดงการเลือกรูปแบบโปรแกรมการทำงาน

1.3) ขั้นตอนที่ 3 กำหนดรายละเอียดที่ต้องการให้กับโปรแกรม

เมื่อพบไดอะล๊อก MFC AppWizard จะพบขั้นตอนการสร้างซึ่งกำหนดรายละเอียดของโปรเจ็กต์ที่ต้องการ จากไดอะล๊อก MFC AppWizard กำหนดค่าดังนี้

1. ยกเลิกเครื่องหมายในช่อง About Box Context - sensitive Help ให้เป็นช่องว่าง

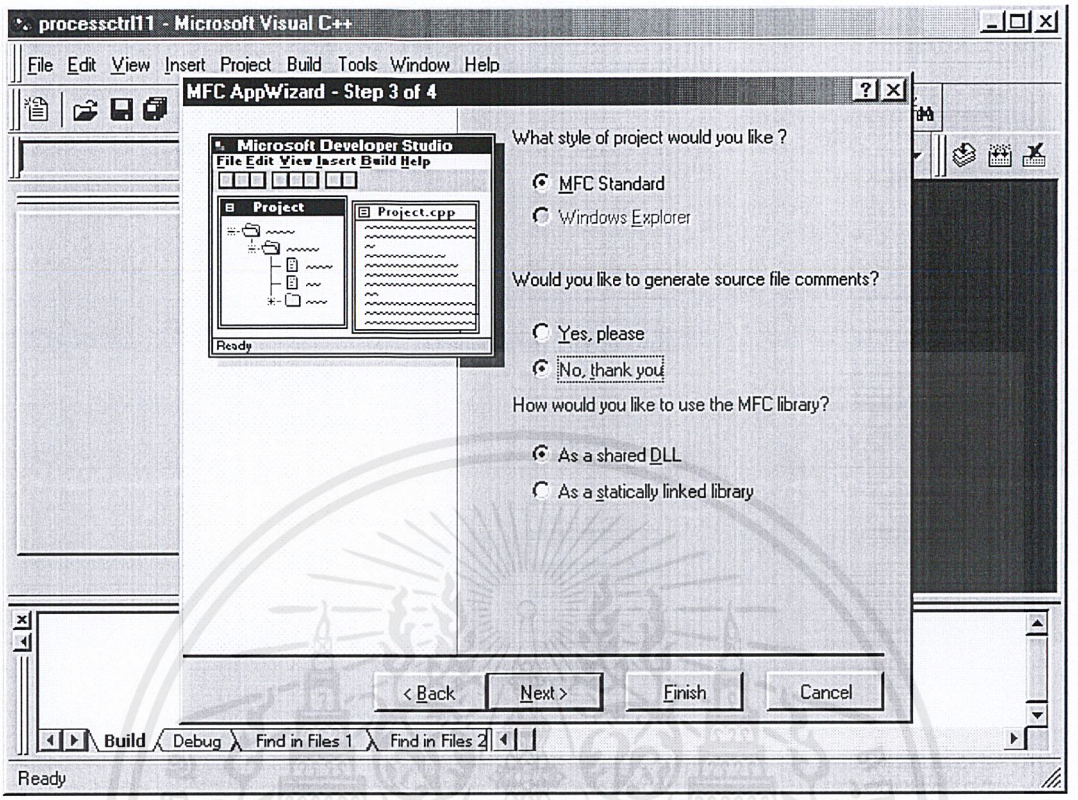
2. กาเครื่องหมายในช่อง 3D Control เพื่อการแสดงผลบนวินโดวส์แบบ 3 มิติ แสดงขั้นตอนการกำหนดดังรูปที่ 3.5



รูปที่ 3.5 การกำหนดรายละเอียดให้กับโปรแกรม

1.4) ขั้นตอนที่ 4 กำหนดตัวเลือกของโปรเจ็กต์

กำหนดตัวเลือก 2 ตัวเลือก คือ ตัวเลือกของการสร้างคำอธิบายโปรแกรมและรูปแบบการ Link ของโปรแกรมดังรูปที่ 3.6 การลิงค์กับไลบารี เลือกการลิงค์แบบ Shared DLL เมื่อนำโปรแกรมไปรันบนวินโดวส์ในเครื่องอื่น ในระบบต้องมีไฟล์ MFC42.DLL และ MSVCRT.DLL อยู่ด้วย ถ้าไม่ม่ไฟล์ .DLL ทั้งสองตัวนี้โปรแกรมก็จะไม่สามารถทำงานได้ ข้อดีของการลิงค์แบบนี้คือ ไฟล์ .EXE จะมีขนาดเล็ก



รูปที่ 3.6 แสดงกำหนดตัวเลือกของโปรเจ็กต์

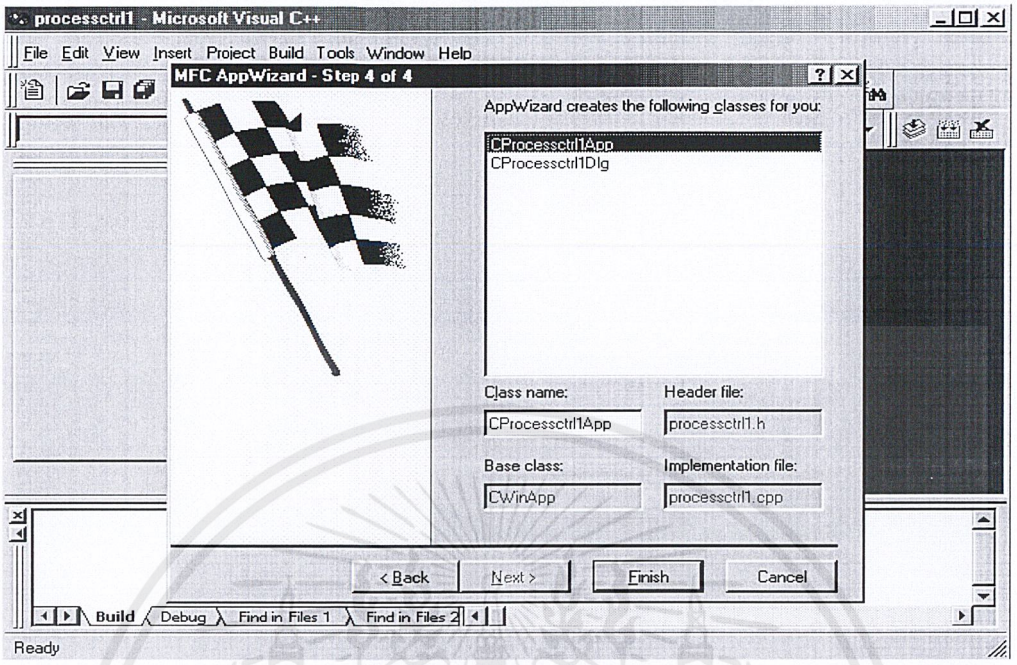
1.5) ขั้นตอนที่ 5 สิ้นสุดการสร้างโปรเจ็กต์

AppWizard จะแสดงรายละเอียดของซอร์สไฟล์ที่ AppWizard จะสร้างขึ้นมาให้ ดังรูปที่ 3.8 ซึ่งจะสร้างคลาสขึ้นมาให้เรา 2 คลาสด้วยกัน คือ

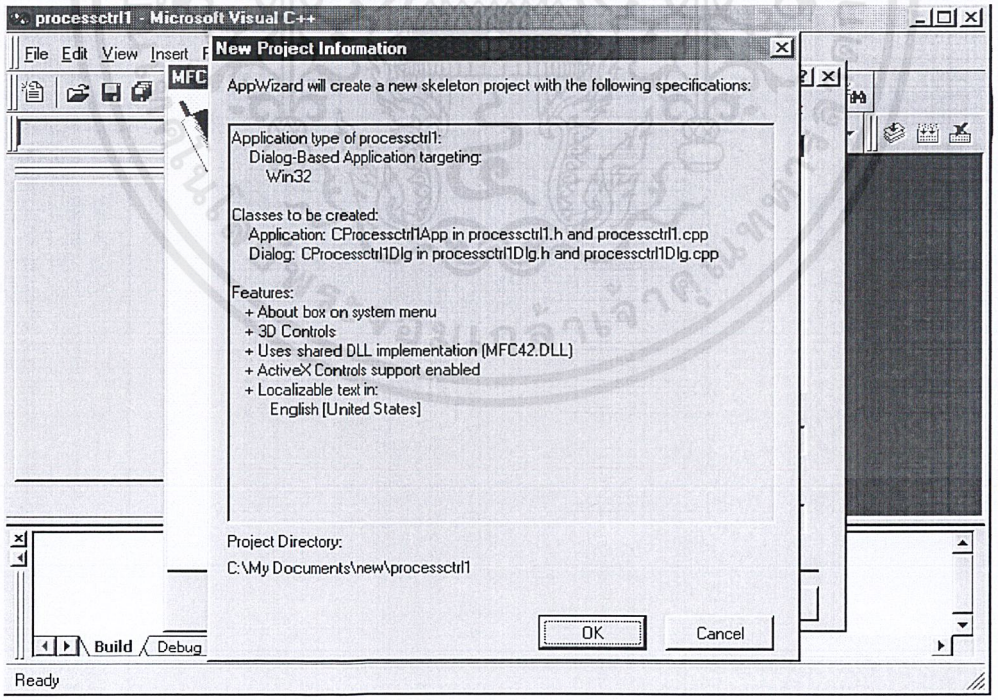
- คลาส CBasic App คลาสนี้จะสืบทอดจากคลาส CwinApp ของ MFC ใช้ในการควบคุมการทำงานของโปรแกรมทั้งหมด เป็นคลาสที่สำคัญที่สุดในโปรแกรม

- คลาส CbasicDlg คลาสนี้จะสืบทอดมาจากคลาส CDialog ของ MFC ทำหน้าที่ควบคุมการแสดงผลและการทำงานของอุปกรณ์ส่วนประกอบต่างๆ ภานในไดอะล็อก

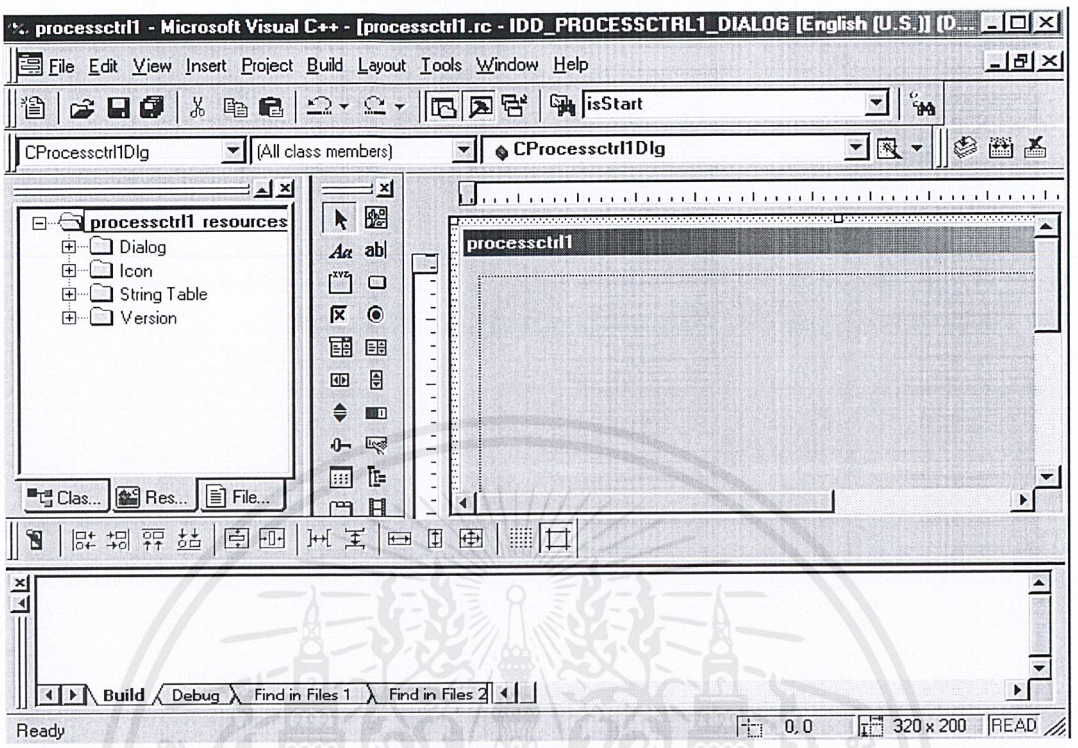
กดปุ่ม OK เพื่อยืนยันความต้องการในการสร้างโปรเจ็กต์ ถ้าตัวเลือกไม่ครบก็ให้กดปุ่ม Cancel และกด Back กลับไปแก้ไขได้ดังรูปที่ 3.9



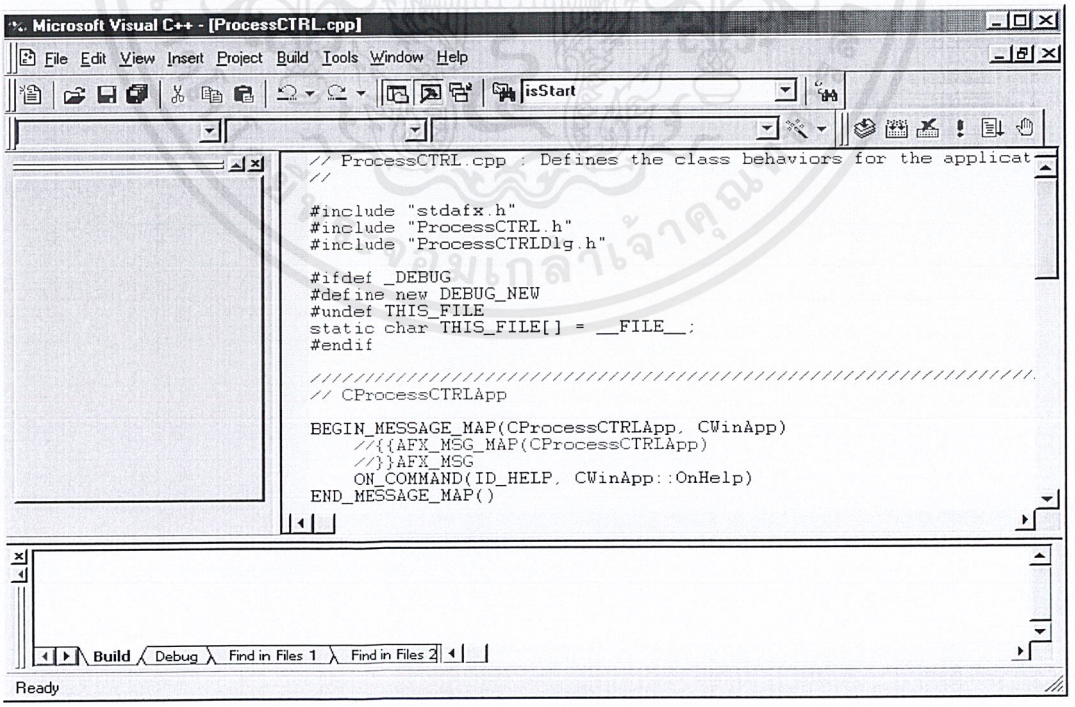
รูปที่ 3.7 แสดงรายละเอียดของซอร์สไฟล์ที่สร้างมาทั้งหมด



รูปที่ 3.8 แสดงการยืนยันความต้องการสร้างโปรเจ็กต์



รูปที่ 3.9 แสดงหน้าต่างการทำงานพร้อมเขียนโปรแกรม

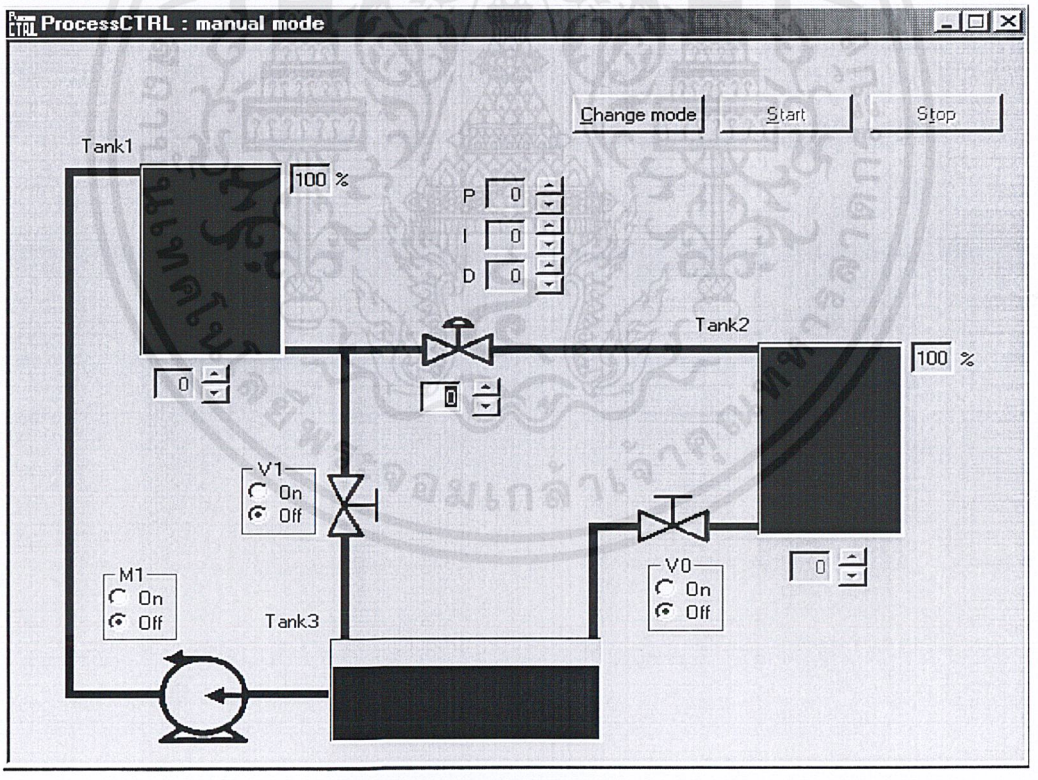


รูปที่ 3.10 แสดงการประกาศไฟล์เนื้อหาฟังก์ชั่น

2) การกำหนดอุปกรณ์การทำงานของโปรแกรม

การทำงานของโครงงานนี้มีการกำหนดอุปกรณ์การใช้งานต่างๆ เมื่อการทำงานของโปรแกรมแสดงผลทางหน้าจอคอมพิวเตอร์และที่ตัว PLANT การทำงานใช้อุปกรณ์การทำงานดังนี้

- M1 คือ ปั๊มน้ำเพื่อจ่ายน้ำให้กับแทงก์น้ำที่ต้องการ
- V1 คือ วาล์วเปิด-ปิดน้ำภายในตัว PLANT
- SN1 คือ สวิตซ์เปิด-ปิดน้ำตามคำสั่งของโปรแกรมการทำงานแบบต่างๆ
- T1 คือ แทงก์น้ำสำหรับพักน้ำมีทั้งหมดจำนวน 3 แทงก์ โดยแทงก์ที่ 1 จะเป็นแทงก์สำหรับพักน้ำเพื่อรักษาระดับน้ำให้คงที่รอการเรียกใช้งาน แทงก์ที่ 2 เป็นแทงก์ที่เรียกน้ำที่จะใช้งานมาพักไว้และปล่อยน้ำใช้งาน แทงก์ที่ 3 เป็นแทงก์สำหรับพักน้ำเพื่อจ่ายให้กับแทงก์ต่างๆ และเป็นแทงก์สำหรับรองรับน้ำที่ถูกปล่อยออกมาจากการตรวจจับน้ำเกินระดับ เพื่อรอจ่ายน้ำให้กับแทงก์น้ำต่อไป ซึ่งสร้างเป็นหน้าจอแสดงผลการทำงานได้ดังรูปที่ 3.11



รูปที่ 3.11 แสดงหน้าจอการทำงานทางคอมพิวเตอร์แบบ MANUAL

3.3.3 การกำหนดการติดต่อพอร์ตกับตัวการ์ด ET-PCDIO

1) การทำงานของพอร์ตติดต่อกับอุปกรณ์การทำงาน

เนื่องจากตัวการ์ด ET-PCDIO มีการรับและส่งสัญญาณติดต่อกับตัวอุปกรณ์ทั้งนี้เนื่องจากว่าการอ่านสัญญาณจากวงจร Instrument Amplifier จะส่งสัญญาณมาติดต่อกับตัวการ์ด ซึ่งพอร์ตการติดต่อในการเขียนโปรแกรมมี 2 รูปแบบทั้งแบบ AUTO และ MANUAL ด้วยกันการทำงานของพอร์ตมีดังนี้คือ

PA0 จะเป็นการทำงานของ M1

PA1 จะเป็นการทำงานของ V1

PA2 จะเป็นการทำงานของ V0

PA3 จะเป็นการทำงานของ SN1

PA4 จะเป็นการทำงานของ SN2

PA5 จะเป็นการทำงานของ SN3

PB1 จะเป็นการทำงานของ T1

PB2 จะเป็นการทำงานของ T2

ทุกพอร์ตการทำงานจะมีสภาวะการทำงานที่ลอจิก “0” หากต้องการให้การทำงานของพอร์ตใดหยุดการทำงานต้องป้อนลอจิก “1” ให้กับตัวโปรแกรม

2) การทำงานของตัวโซลินอยด์วาล์ว

โซลินอยด์วาล์ว จะทำหน้าที่คล้ายๆกับตัวคอนโทรลวาล์ว ซึ่งในโครงงานนี้จะกำหนดการทำงานของโซลินอยด์วาล์วให้เปิด - ปิดการทำงานโดยมีการตั้งค่าของการทำงานเป็นเปอร์เซนต์ของระดับน้ำที่วงจรเซนเซอร์ระดับตรวจจับมา การทำงานของโซลินอยด์วาล์วแต่ละตัวกำหนดค่าเป็นดังนี้คือ

- ที่ระดับน้ำ 0 % ให้ PA3 , PA4 ,PA5 อยู่ในสภาวะลอจิก “1” คือโซลินอยด์วาล์วทุกตัวยังไม่ทำงาน

- ที่ระดับน้ำ 1- 33 % ให้ PA3 อยู่ในสภาวะลอจิก “ 0 ” คือโซลินอยด์วาล์วตัวที่ 1 ทำงาน ส่วน PA4 ,PA5 อยู่ในสภาวะลอจิก “1” คือโซลินอยด์วาล์วทั้งสองจะไม่ทำงาน

- ที่ระดับน้ำ 34 -66 % ให้ PA3 , PA4 อยู่ในสภาวะลอจิก “ 0 ” คือโซลินอยด์วาล์วตัวที่ 1 และ 2 จะทำงาน ส่วน PA5 อยู่ในสภาวะลอจิก “1” คือโซลินอยด์วาล์วยังไม่ทำงาน

- ที่ระดับน้ำ 67-100 % ให้ PA3 , PA4 ,PA5 อยู่ในสภาวะลอจิก “ 0 ” คือโซลินอยด์วาล์วทุกตัวทำงานจะเปิดน้ำให้ไหลผ่านมากขึ้น

แบบ MANUAL และ AUTO จะต่างกันที่ แบบ AUTO จะนำสูตรคำนวณแบบ PID มาคิดซึ่ง กำหนดสูตรการคำนวณเป็นดังนี้

$$M = P(A-B) + I((A-B) + C)) \ / 2 \ + D((A-B)-C)$$

- เมื่อ P คือ ค่า Propertional

I คือ ค่า Integral

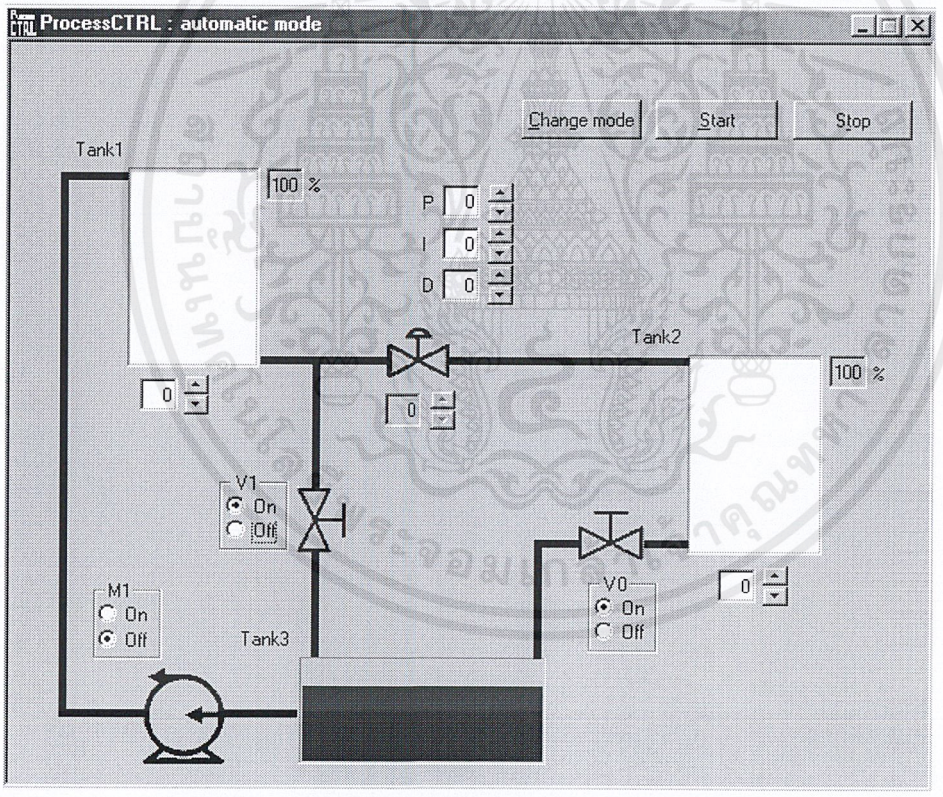
D คือ ค่า Derivative

A คือ ค่าที่ตั้งไว้ของ T2 = (SV)

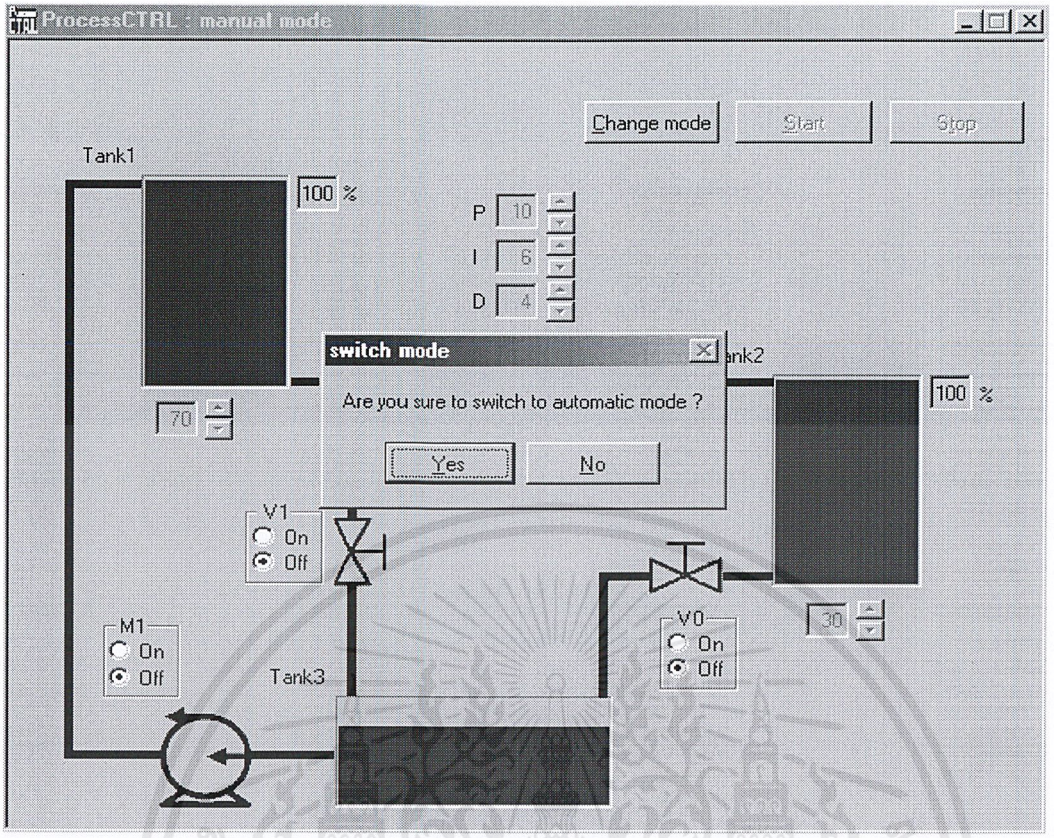
B คือ ค่าที่อ่านมาได้จาก T2 = (PV)

C คือ ค่าที่คำนวณเสร็จ

ดังนั้น ค่าที่คำนวณเสร็จเท่ากับ SV-PV การทำงานของแบบ AUTO จะรับค่าสัญญาณที่ต้องการมาคำนวณจากสูตรอย่างเดียว ซึ่งหลักการทำงานการควบคุม PID ได้กล่าวไว้แล้วในบทที่ 2 การทำงานของโหมด AUTO แสดงดังรูปที่ 3.13



รูปที่ 3.12 แสดงลักษณะการทำงานของโหมด AUTO



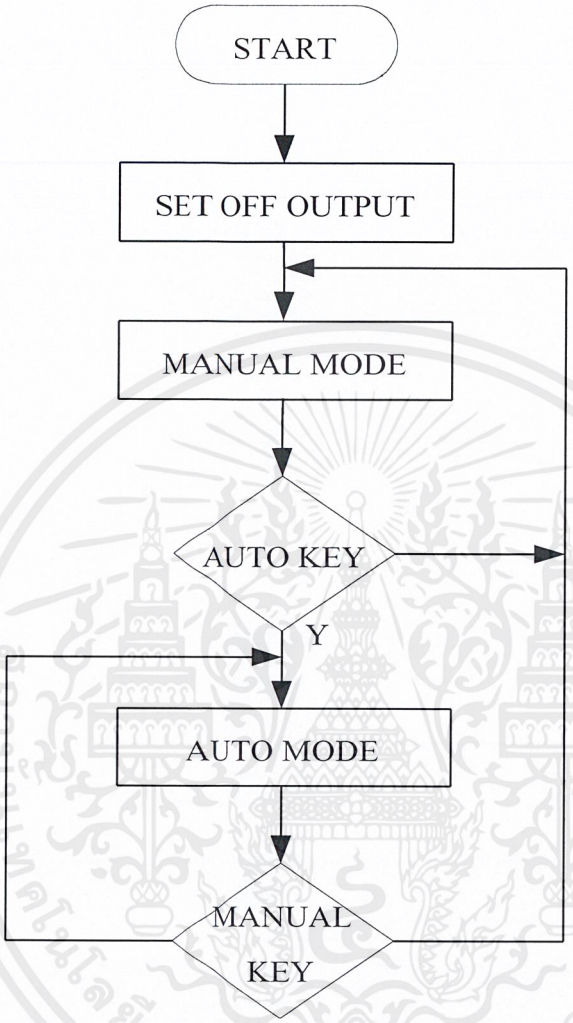
รูปที่ 3.13 แสดงการสับเปลี่ยนโหมดการทำงาน

3) การอ่านสัญญาณของอุปกรณ์ภายนอก

การอ่านสัญญาณอินพุทของ TANK 1 (PB1) และ TANK 2 (PB 2) จะส่งสัญญาณการอ่านค่าจากวงจร Instrument Amplifier โดยการอ่านค่าจะหน่วงเวลาไว้ 100 ms อ่านค่าจาก T1 โดย Set PB1 เป็น " 1" และ PB2 เป็น " 0 " และจะเริ่มอ่านค่าจาก T1 ที่พอร์ต ADC หลังจากนั้นหน่วงเวลาไว้ 100 ms ให้อ่านค่า T2 โดนเซ็ตค่า PB2 เป็น "1" และ PB1 เป็น "0" และเริ่มอ่านค่าจาก T2 ที่พอร์ต ADC การอ่านค่า T1 และ T2 ใช้พอร์ต ADC ร่วมกัน โดยใช้ PB1 และ PB2 เป็นตัวแยกสัญญาณ

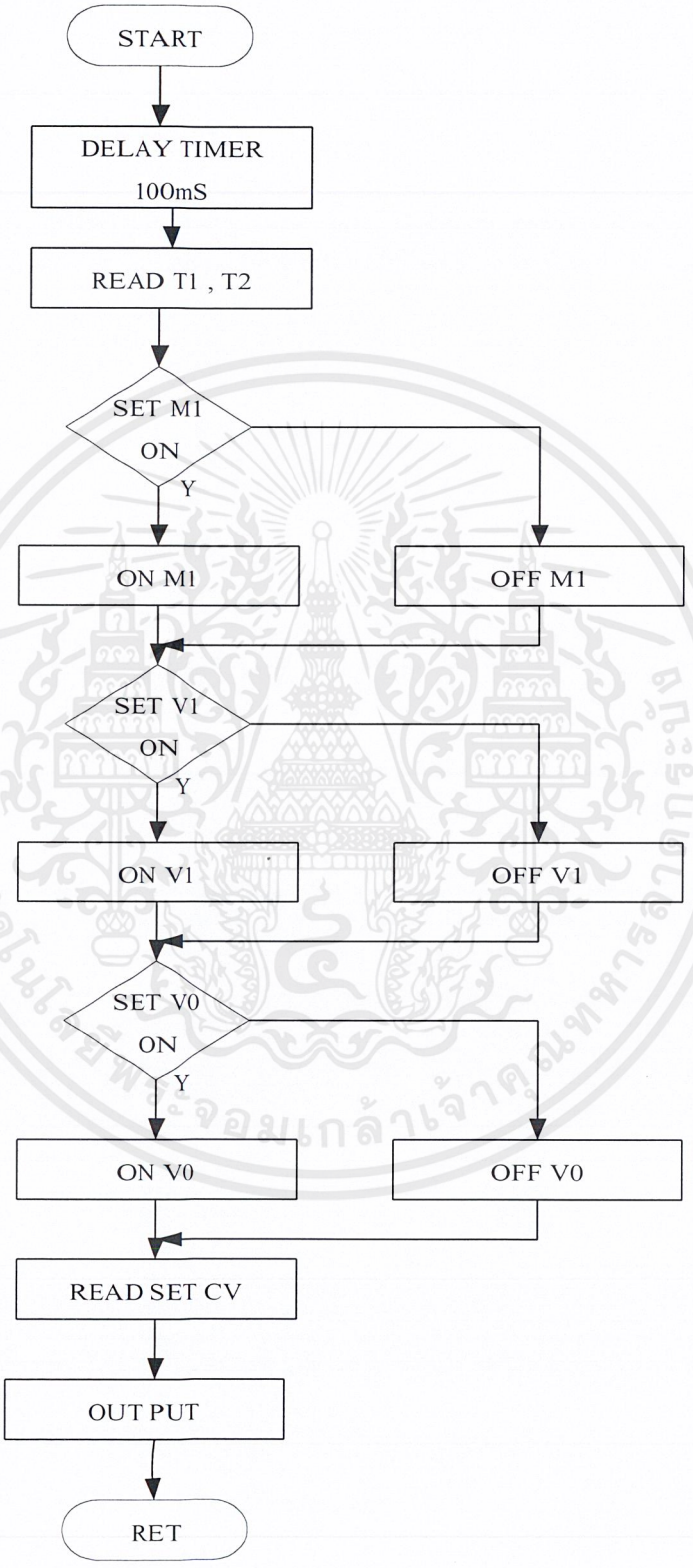
3.4 ผังการทำงานของโปรแกรม

3.4.1 ผังการทำงานของระบบการควบคุม



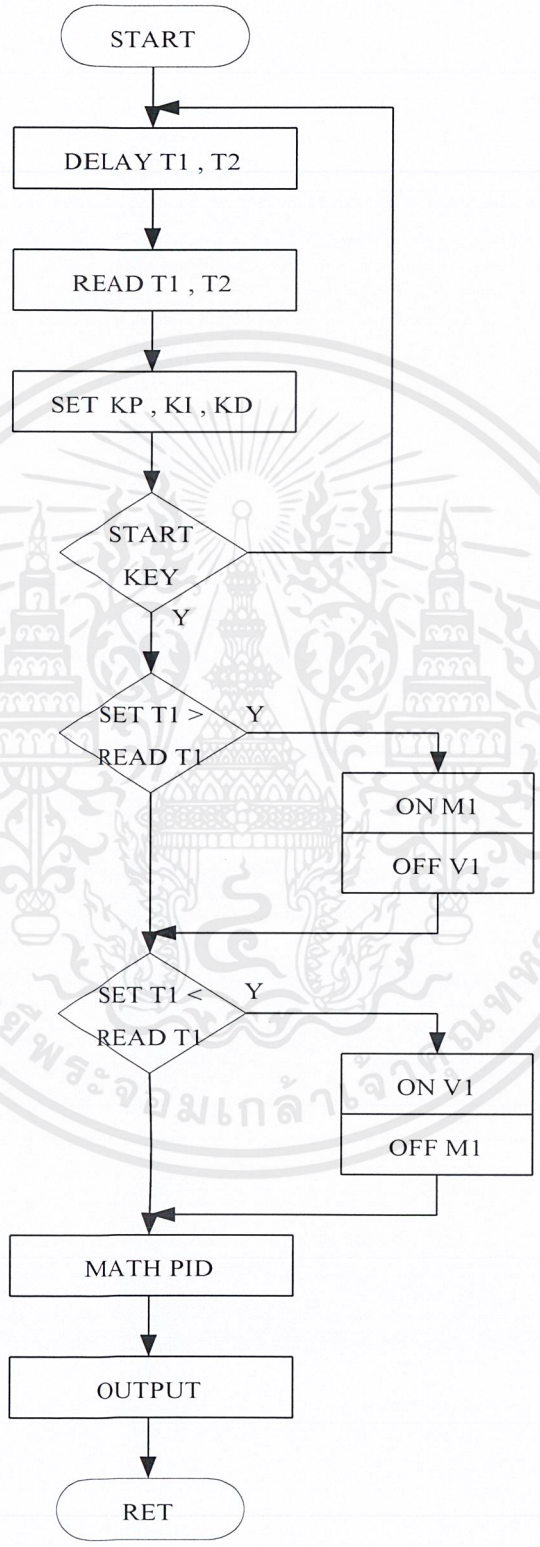
รูปที่3.14 แสดงลำดับขั้นการทำงานของระบบควบคุม

3.4.2 ผังการทำงานของโปรแกรมแบบ MANUAL



รูปที่3.15 แสดงการทำงานของโปรแกรมแบบ MANUAL

3.4.3 ผังการทำงานของระบบควบคุมแบบ AUTO



รูปที่ 3.16 แสดงการทำงานโปรแกรมแบบ AUTO

บทที่ 4

การทดลองและผลการทดลอง

จากการทดลองและมีการตรวจสอบการทำงานของระบบควบคุมระดับได้แบ่งการทดลองออกเป็น 4 ส่วนใหญ่ๆ คือ ส่วนที่หนึ่งเป็นการทดลองส่วนของตัว Plant ชุดประยุกต์ใช้งาน ส่วนที่สองเป็นการทดลองส่วนที่เป็นวงจรทางด้าน Hardware ทั้งหมดได้แก่ วงจรขับสัญญาณอินพุตและเอาต์พุต วงจรตรวจจับระดับ และ วงจร Instrument Amplifier ส่วนที่สามเป็นการทดสอบโปรแกรมการทำงาน ส่วนที่สี่เป็นการทดลองโปรแกรมการทำงานโดยสั่งควบคุมทางคอมพิวเตอร์ไปควบคุม Plant การทดลอง

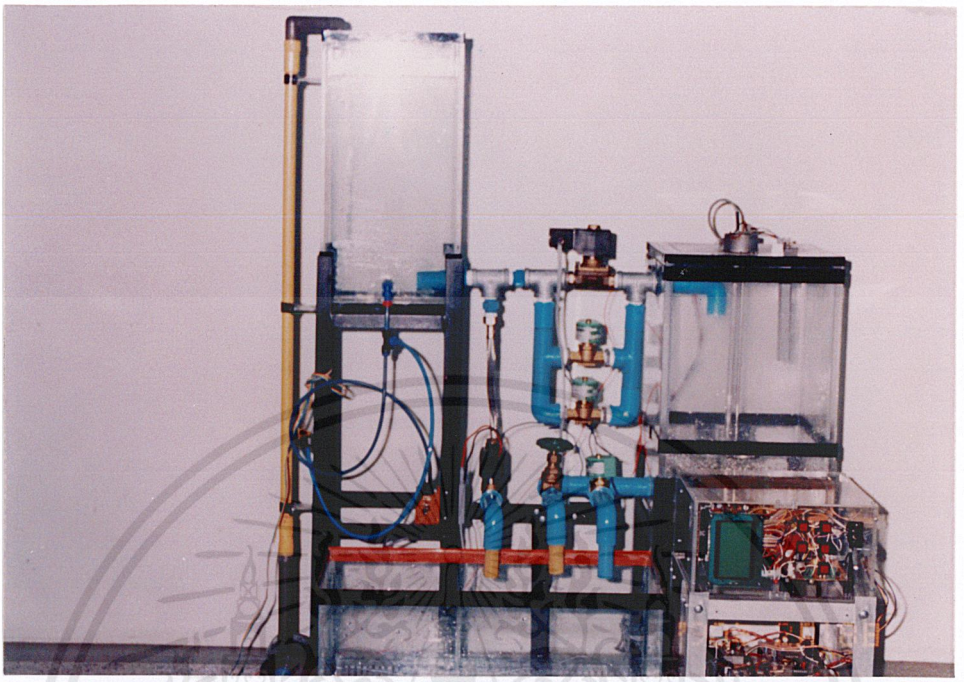
4.1 การทดลองส่วนของ plant การทดลอง

ลำดับขั้นการทดลอง

1. ประกอบ Plant ตามรูปที่ 4.1
2. ตรวจสอบความเรียบร้อยของตัว Plant แล้วทำการจ่ายไฟให้ปั๊มและตัวโซลีนอยด์วาล์ว
3. จ่ายน้ำให้กับแทงก์น้ำโดยให้ปั๊มน้ำดูดน้ำขึ้นมาจากแทงก์พักน้ำให้ได้ระดับที่ต้องการมาไว้ในแทงก์น้ำที่จะต้องจ่ายให้กับแทงก์ส่วนอื่น
4. ทำให้โซลีนอยด์วาล์วแต่ละตัวที่ตำแหน่งต่างๆ เปิด และปิด เพื่อสังเกตการไหลของน้ำ

ผลการทดลอง

จากการทดลองส่วนของ plant ผลปรากฏว่าเมื่อจ่ายไฟให้กับตัวโซลีนอยด์วาล์วแล้ว ตัวโซลีนอยด์วาล์วจะทำหน้าที่เปิดและปิดน้ำ โดยจะมีการทำงานที่ตัวหน้าสัมผัสของโซลีนอยด์วาล์วในแต่ละตัว ในการทดลองจะได้ยินเสียงดังขึ้นจากตัวโซลีนอยด์เพื่อแสดงว่าได้ทำงานในสภาวะเปิดหรือ ปิด แล้ว ทั้งนี้ตัว Plant ต้องถูกสั่งผ่านจากส่วนอื่นๆ อีก แต่การทดลองทำให้ทราบว่าตัว Plant สามารถเปิด และปิดการไหลของน้ำได้ พร้อมที่จะได้รับการควบคุมตามคำสั่งการทำงานต่อไป



รูปที่ 4.1 แสดง PLANT การทดลอง

4.2 การทดลองส่วนของวงจรทางด้านฮาร์ดแวร์

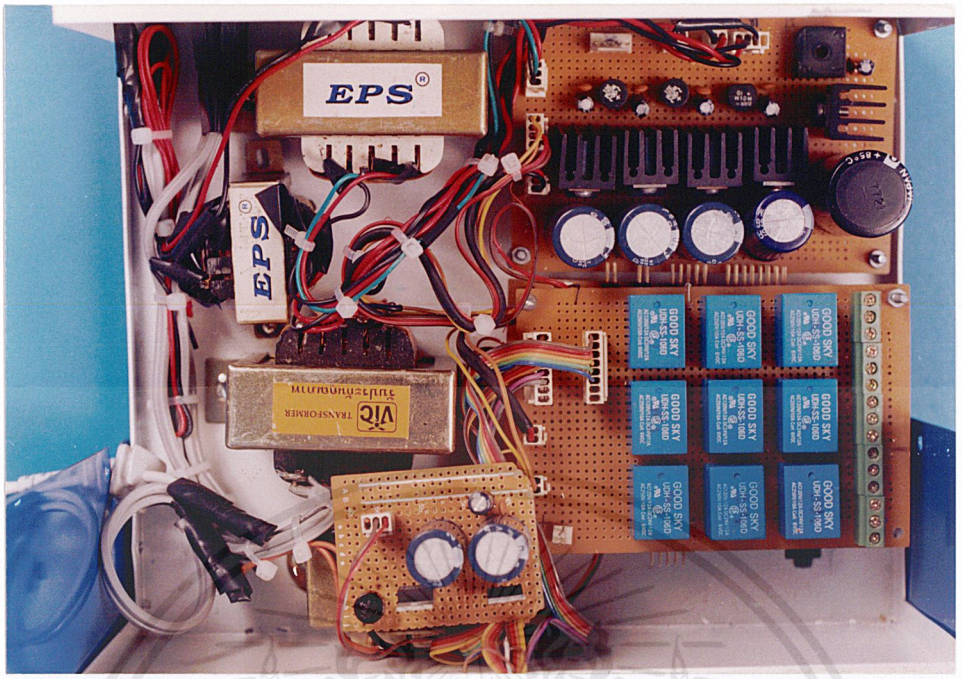
4.2.1 การทดลองวงจรขับสัญญาณอินพุตและเอาต์พุต

ลำดับขั้นการทดลอง

1. ประกอบวงจรขับสัญญาณ ตามรูปที่ 4.2
2. ตรวจสอบความเรียบร้อยของวงจร
3. ทำการจ่ายไฟให้กับวงจร
4. ทำการโยกสวิตซ์ทดสอบการทำงานของรีเลย์แต่ละตัว
5. สังเกตผลการทำงานที่หลอด LED ที่ติดกับตัววงจร

ผลการทดลอง

จากการทดลองในส่วนของวงจรขับสัญญาณ เมื่อทำการทดลองตามขั้นตอนการทำงานของวงจรขับสัญญาณจะแสดงผลออกทางหลอด LED ให้หลอดสว่างขึ้นมา เมื่อเราโยกสวิตซ์การทำงานในแต่ละตัว ซึ่งเป็นหน้าสัมผัสการใช้งานของตัวรีเลย์ภายในวงจรและกำหนดตัวไว้แล้วว่าแต่ละสวิตซ์คือการทำงานของส่วนใด ซึ่งแสดงว่าวงจรขับสัญญาณสามารถควบคุมการทำงานของ plant ได้



รูปที่ 4.2 แสดงวงจรขับสัญญาณ

4.2.2 การทดลองวงจรตรวจจับระดับ

ลำดับขั้นการทดลอง

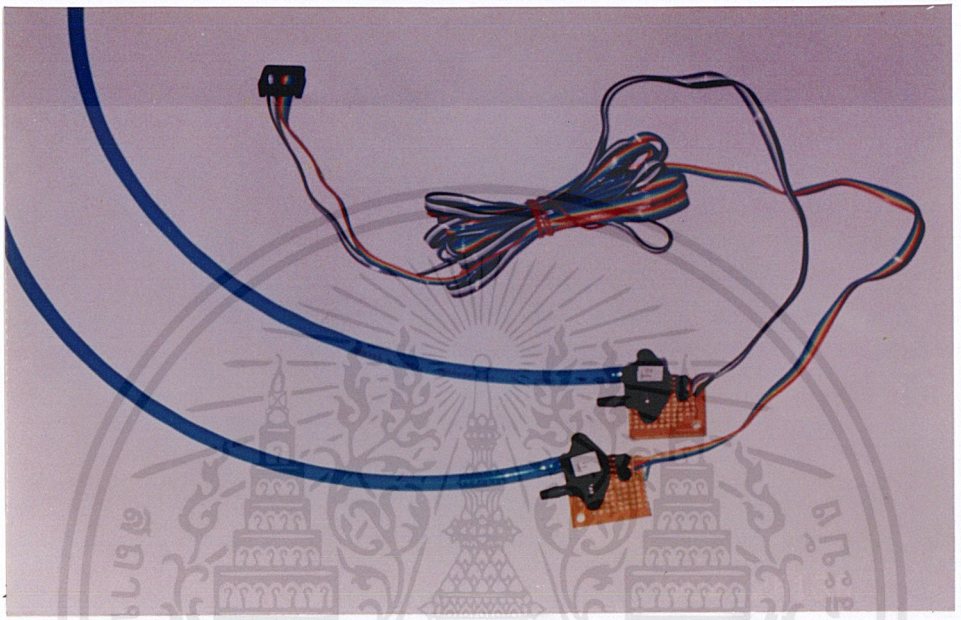
1. ประกอบวงจรตรวจจับแรงดันตามรูปที่ 4.3
2. ทำการจ่ายไฟ +5 V เข้าที่ขา 3 และต่อขา 1 ลงกราวด์
3. นำมิเตอร์มาทำการวัดโดยที่ให้ขั้วบวกวัดที่ขา 2 ขั้วลบวัดที่ขา 4
4. ทำการจ่ายลมให้กับตัว Sensor โดยใช้แรงดันตั้งแต่ 0-10 Kpa
5. สังเกตการเปลี่ยนแปลงของมิเตอร์นำค่าที่ได้บันทึกดูการเปลี่ยนแปลง

ตารางผลการทดลองที่4.1 วงจรเซนเซอร์วัดระดับ

แรงดัน (Kpa)	ค่าจากการคำนวณ (mV)	ค่าที่ได้จากการวัด (mV)
10	70	69
8	63	62
6	56	55
4	49	48
2	42	41
0	35	34.2

ผลการทดลอง

จากการทดลองวงจรแล้วทำการบันทึกค่าไว้ในตารางจะเห็นได้ว่าการทำงานของวงจรจะมีค่าความผิดพลาดและการคลาดเคลื่อนไป แต่ในความเป็นจริงนั้นเราสามารถที่จะคาลิเบรทสัญญาณได้ตามต้องการ



รูปที่ 4.3 ตัวตรวจจับสัญญาณวัดระดับ (D / P)

4.2.3 วงจร Instrument Amplifier

ลำดับขั้นการทดลอง

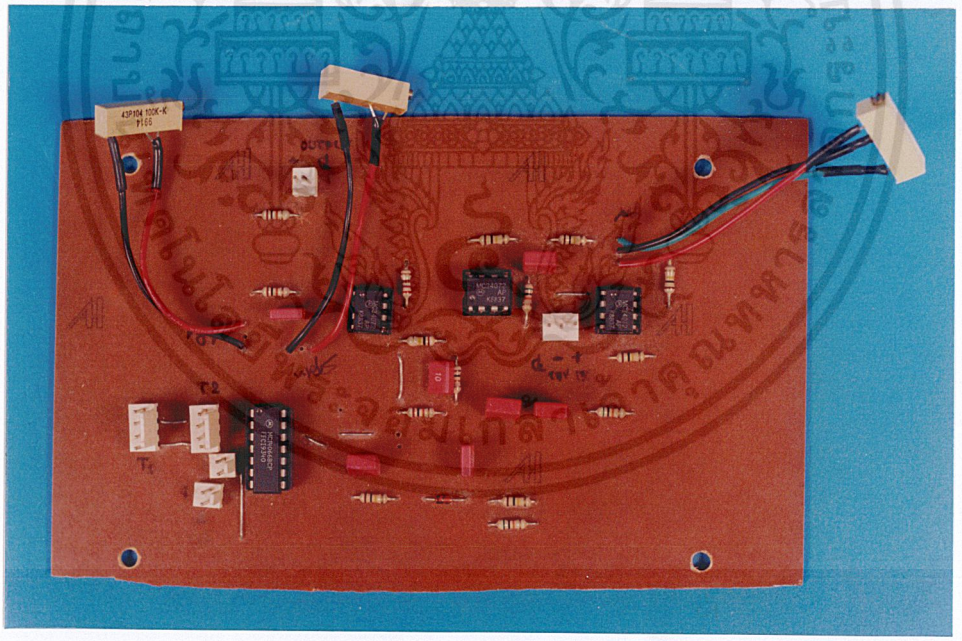
1. ประกอบวงจร Instrument Amplifier ตามรูปที่ 4.4
2. ตรวจสอบความเรียบร้อยทั้งหมดของวงจร
3. ทำการจ่ายอินพุตให้กับวงจรที่ 100 % แล้วทำการปรับค่า RG (Span)ให้ได้แรงดันเอาต์พุตที่ 5 โวลต์
4. ทำการจ่ายอินพุตให้กับวงจรที่ 0 % แล้วทำการปรับค่า RG(Zero) ให้ได้แรงดันเอาต์พุต 0 โวลต์
5. ทำการปรับตามขั้นตอนที่ 2 และ 3 จนทำให้ได้ค่าผิดพลาด 1%

ตารางผลการทดลองที่ 4.2 วงจร Instrument Amplifier

อินพุต (%)	ค่าจากการคำนวณ (V)	ค่าที่ได้จากการวัด (V)
100	5	5
80	4	4
60	3	3
40	2	2
20	1	1
0	0	0

ผลการทดลอง

จากการทดลองและบันทึกค่าจากตารางจะเห็นได้ว่าการทำงานของ Instrument Amplifier จากค่าที่คำนวณและค่าที่ทำการวัดนั้นได้มาตรฐานตามต้องการ และสามารถทำงานได้จริง



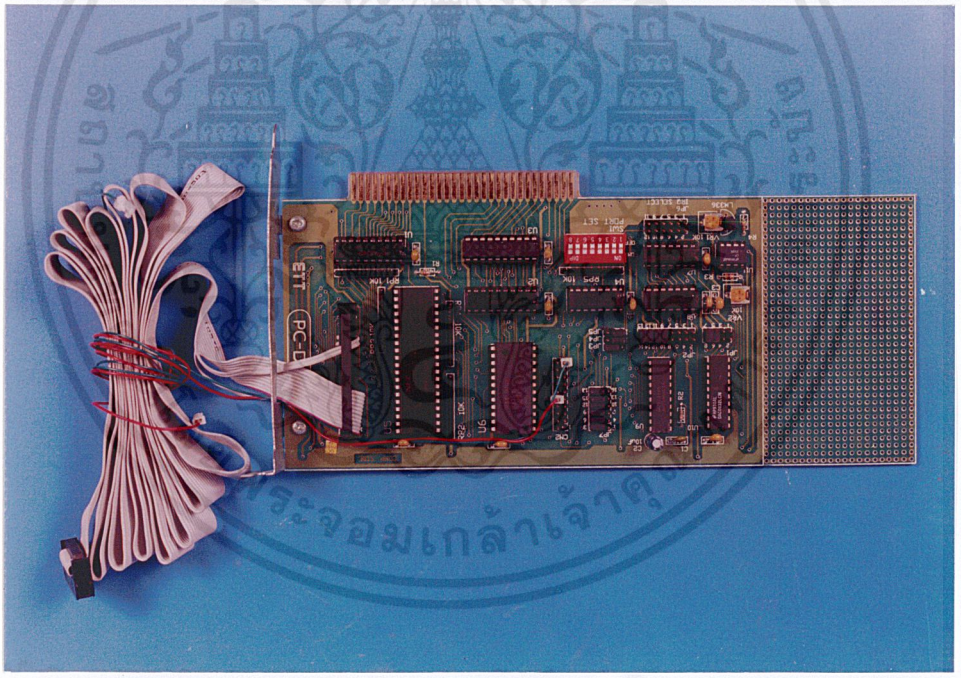
รูปที่ 4.4 วงจร Instrument Amplifier

4.3 การทดลองส่วนของโปรแกรมการทำงาน

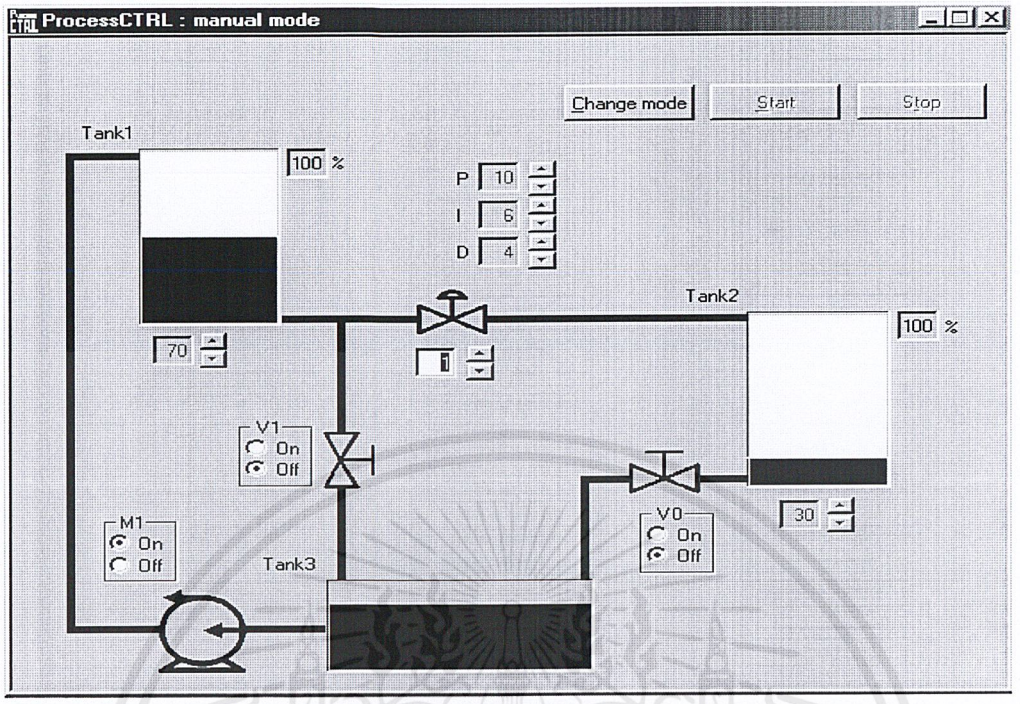
4.3.1 การทดลองส่วนของโปรแกรมในรูปแบบการทำงานแบบ MANUAL

ลำดับขั้นการทดลอง

1. เขียนโปรแกรมการทำงานแบบ MANUAL ตามผังงานของโปรแกรม
2. ต่อการ์ด ET- PCDIO เข้ากับตัวเครื่องคอมพิวเตอร์
3. ตรวจสอบความเรียบร้อยของวงจร
4. ทำการรันโปรแกรมและทำการจ่ายแรงดันให้กับบอร์ดเอาต์พุต 0-5 โวลต์ที่ขา VI+ และ VI-
5. ทดลองป้อนค่าให้กับแทงก์น้ำที่ 1 และแทงก์น้ำที่ 2 ตามลำดับ
6. ทดลองกดปุ่ม ON และ OFF ของวาล์วต่างๆ ที่แสดงผลบนหน้าจอเครื่องคอมพิวเตอร์ สังเกตุผลที่เกิดขึ้นบนหน้าจอคอมพิวเตอร์ดังรูปที่ 4.6



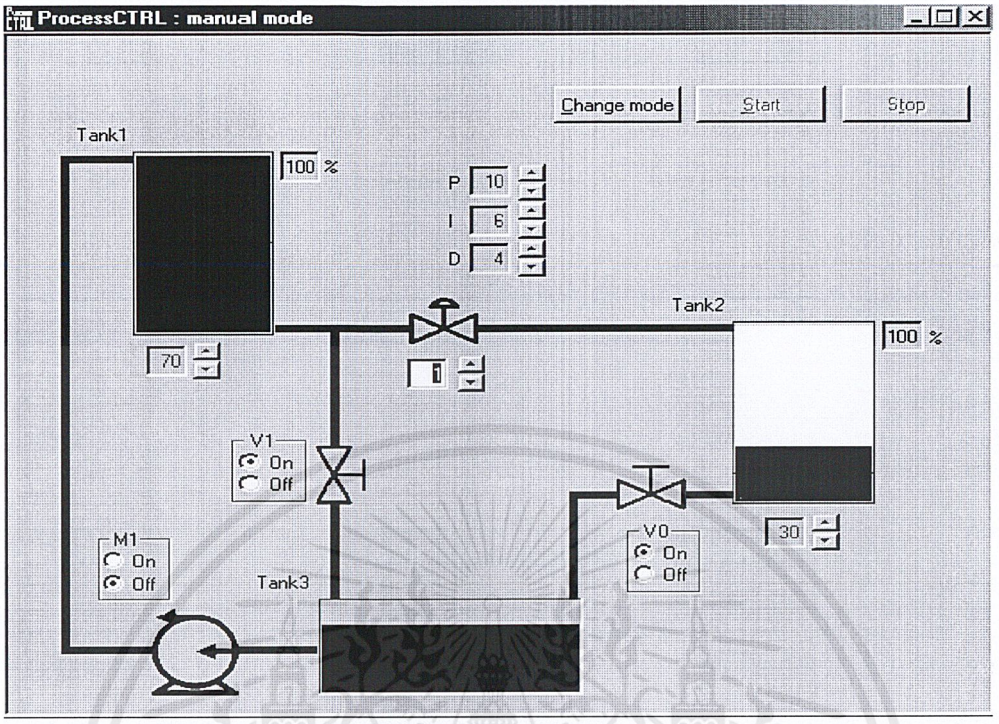
รูปที่ 4.5 แสดงตัวการ์ด ET – PC DIO



รูปที่ 4.6 แสดงผลที่เกิดขึ้นบนหน้าจอคอมพิวเตอร์ตามการทดลอง

ผลการทดลอง

จากการทดลองโปรแกรมการควบคุมแบบ Manual ผลปรากฏว่าเมื่อทำการป้อนค่าให้กับตัวแทงก์น้ำที่1 และ 2 ตามลำดับนั้นจะเห็นผลได้ตามรูปที่ 4.5 โดยการควบคุมแบบ Manual จะสามารถควบคุมได้ด้วยการนำ Mouse ไปคลิกปุ่มตามตัวอุปกรณ์ที่แสดงทางหน้าจอเพื่อต้องการให้ทำงานในสภาวะ ON หรือ OFF การทำงาน สามารถเลือกการทำงานได้ว่าต้องการให้อุปกรณ์ตัวไหนทำงานก่อนหรือหลังได้ และสามารถรับสัญญาณจากวงจรตรวจจับสัญญาณมาประมวลผลในคอมพิวเตอร์อีกด้วย และในกรณีที่น้ำที่จ่ายให้กับแทงก์น้ำที่ 1 และ แทงก์น้ำที่ 2 จ่ายให้มากเกินไป โซลินอยด์วาล์วก็จะเปิดให้น้ำไหลผ่านทิ้งไปที่แทงก์รับน้ำทันทีดังรูปที่ 4.6

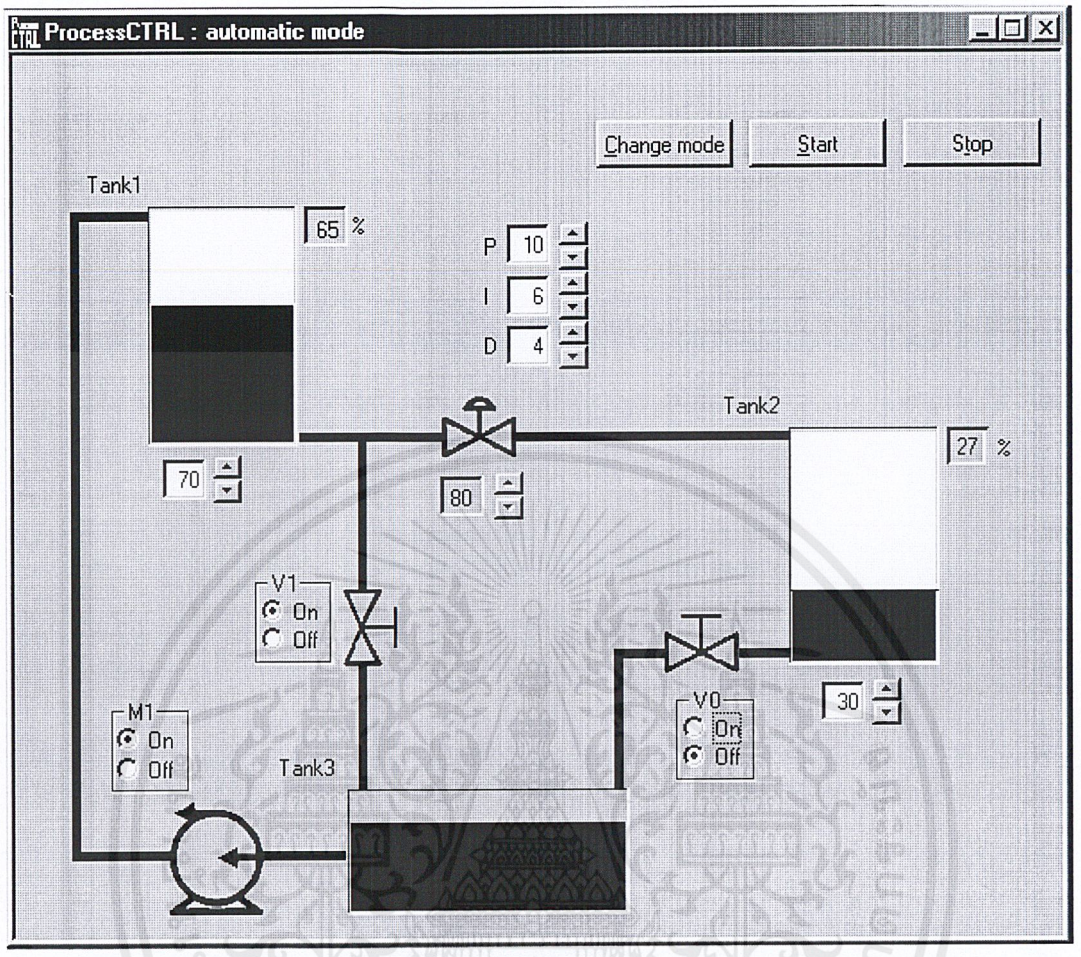


รูปที่ 4.7 แสดงผลกรณีจ่ายน้ำเกินค่าที่ต้องการทางจอคอมพิวเตอร์

4.3.2 การทดลองส่วนของโปรแกรมในรูปแบบการทำงานแบบ AUTO

ลำดับขั้นการทดลอง

1. เขียนโปรแกรมการทำงานแบบ AUTO ตามผังงานของโปรแกรม
2. ต่อการ์ด ET- PCDIO เข้ากับตัวเครื่องคอมพิวเตอร์
3. ตรวจสอบความเรียบร้อยของวงจร
4. ทำการรันโปรแกรมและทำการจ่ายแรงดันให้กับบอร์ดเอาต์พุต 0-5 โวลต์ที่ขา VI+ และ VI-
5. ทดลองป้อนค่าให้กับแทงก์น้ำที่ 1 และแทงก์น้ำที่ 2 ตามลำดับ
6. ทดลองป้อนค่าให้กับ P , I และ D
7. ทดลองกดปุ่ม START บนหน้าจอเครื่องคอมพิวเตอร์
8. สังเกตผลที่เกิดขึ้นบนหน้าจอคอมพิวเตอร์ดังรูปที่ 4.7



รูปที่ 4.8 แสดงหน้าจอการทำงานทางคอมพิวเตอร์แบบAUTO ตามการทดลอง

ผลการทดลอง

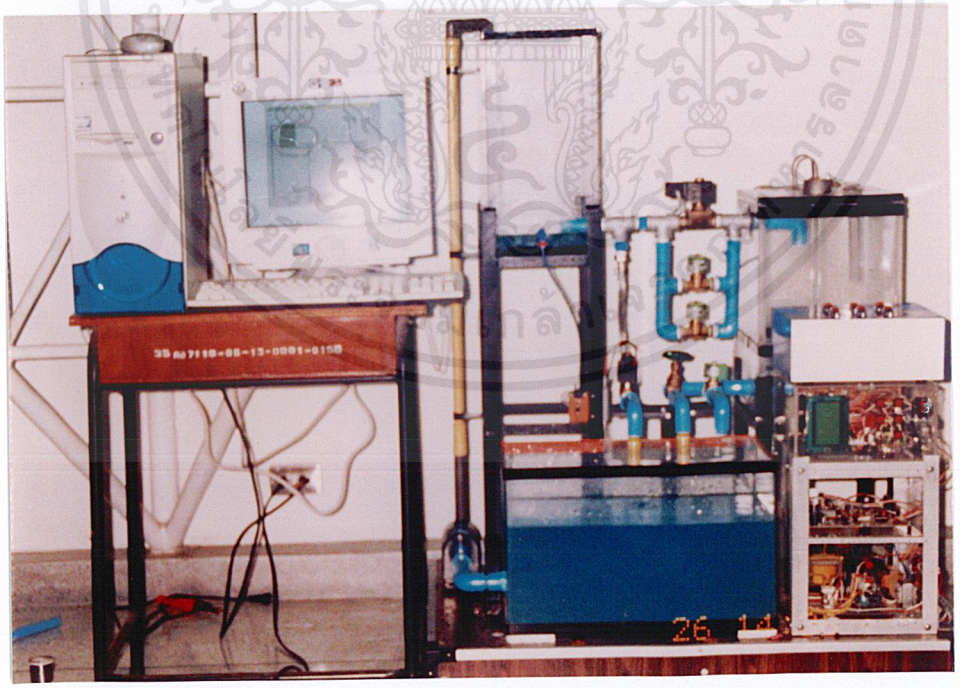
จากการทดลองโปรแกรมการควบคุมการทำงานแบบ AUTO จะเห็นได้จากรูปที่4.8 แสดงการทำงานในการทดลอง ซึ่งจะเห็นว่าในการรันโปรแกรมการทำงานแบบนี้จะควบคุมเพียงปุ่มการทำงานเดียวเท่านั้นคือ ปุ่ม START หลังจากนั้นก็รอดูผลการเปลี่ยนแปลงอย่างเดียว โดยจะสังเกตจากหน้าจอบนคอมพิวเตอร์ที่แสดงการเปลี่ยนแปลงของระดับน้ำ และการเปลี่ยนแปลงของค่าเปอร์เซนต์ที่แสดงทาง Text box ที่เปลี่ยนแปลงตามระดับน้ำที่ขึ้นลง การทำงานแบบอัตโนมัติจะทำงานจนกว่าจะครบวงจรการทำงาน การทำงานทั้งหมดในการเซตค่าไว้ในแต่ละครั้ง แล้วจึงป้อนค่าใหม่เข้าไปเพื่อสั่งให้ทำงานอีก ซึ่งเมื่อครบวงจรการทำงานแล้วจะหยุดการทำงานโดยอัตโนมัติ

4.4 การทดลองส่วนของโปรแกรมควบคุมการทำงานที่ตัว plant

4.4.1 การทดลองรวมส่วนของโปรแกรมควบคุม plant แบบ Manual

ลำดับขั้นการทดลอง

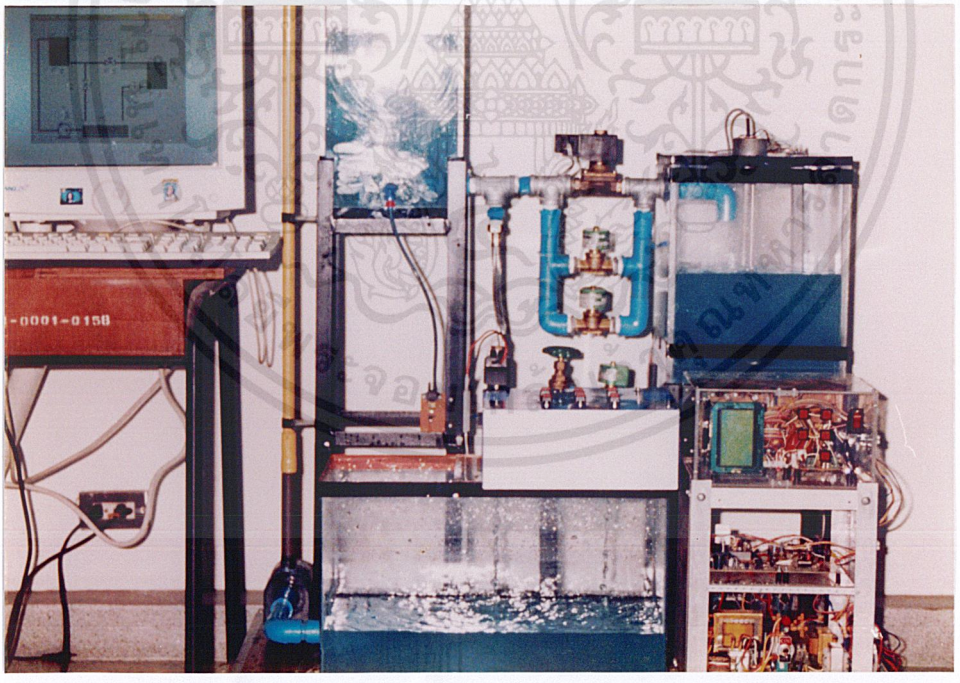
1. ประกอบวงจรรวมทั้งหมดเข้ากับชุดวงจรขับสัญญาณทุกส่วนตามรูปที่4.7
2. ต่อการ์ดET-PCDIO เข้าที่ตัวคอมพิวเตอร์
3. ต่อวงจรระหว่างตัวการ์ด ET-PCDIO เข้ากับชุดวงจรรวมและต่อเข้ากับอุปกรณ์ที่ต้องควบคุมภายใน Plant การทดลอง
4. ตรวจสอบความเรียบร้อยของวงจรและการต่อวงจรทุกส่วนให้ถูกต้อง
5. เข้าสู่โปรแกรมการทำงานที่แสดงออกมาบนคอมพิวเตอร์
6. ทำการรันโปรแกรมการทำงานแบบ Manual
7. ป้อนค่าให้กับแทงก์น้ำที่ 1 และ 2
8. กดปุ่มการทำงานของตัวอุปกรณ์ให้ ON หรือ OFF ตาม Step การทำงาน
9. สังเกตผลการทำงานบนหน้าจอคอมพิวเตอร์และการทำงานที่ตัว plant ทดลอง



รูปที่ 4.9 แสดงการติดตั้งระหว่างเครื่องคอมพิวเตอร์และตัว plant ทดลอง

ผลการทดลอง

จากการทดลองเห็นว่าเมื่อเราป้อนค่าให้กับตัวแทงก์น้ำที่ 1และ 2 แล้วกดปุ่ม ON ที่ตัวปั๊มน้ำจะเห็นว่าปั๊มน้ำจะคอยปั๊มน้ำขึ้นไปให้กับแทงก์ที่ 1 ตามที่เรากำหนดค่าให้กับแทงก์ที่ 1 เพื่อต้องการให้แทงก์ที่ 1 รักษาระดับน้ำที่ระดับตามต้องการ หลังจากนั้นเราป้อนค่าให้กับแทงก์น้ำที่ 2 เพื่อแสดงว่าต้องการน้ำไปใช้ที่แทงก์ 2 แล้วจึงไปใส่ค่าที่ตัวคอนโทรลวาล์ว เพื่อเรียกเพื่อเรียกระดับน้ำใช้งาน เมื่อเรากดปุ่มเลือกค่าที่ตัวคอนโทรลวาล์ว ตัวคอนโทรลวาล์วจะ ON ซึ่งภายในตัวคอนโทรลวาล์วที่ตัว plant เราใช้ตัวโซลีนอยด์วาล์วแทนเมื่อต้องการน้ำที่จำนวนกี่เปอร์เซนต์จะกำหนดระดับน้ำเริ่มที่ 1-33 % โ ซลีนอยด์จะทำงาน 1 ตัว ที่ 34-66 % ทำงาน 2 ตัว และที่ 67 - 100 % น้ำจะไหลผ่านตัวโซลีนอยด์ทั้งหมด 3 ตัวด้วยกัน ดังนั้นการทำงานในแต่ละส่วนของรูปแบบ Manual นั้นไม่จำเป็นที่จะรอการทำงาน หากต้องการให้ส่วนใดทำงานก็สามารถเข้าไปกดปุ่มได้



รูปที่ 4.10 แสดงผลการทดลองบนหน้าจอคอมพิวเตอร์และตัว plant ทดลอง

4.4.2 การทดลองในส่วนของโปรแกรมควบคุม ตัวplant ในแบบของ AUTO

ลำดับขั้นการทดลอง

1. ประกอบวงจรรวมทั้งหมดเข้ากับชุดวงจรขับสัญญาณทุกส่วน
2. ต่อการ์ดET-PCDIO เข้าที่ตัวคอมพิวเตอร์
3. ต่อวงจรระหว่างตัวการ์ด ET-PCDIO เข้ากับชุดวงจรรวมและต่อเข้ากับอุปกรณ์ที่ต้องควบคุมภายใน Plant การทดลอง
4. ตรวจสอบความเรียบร้อยของวงจรและการต่อวงจรทุกส่วนให้ถูกต้อง
5. เข้าสู่โปรแกรมการทำงานที่แสดงออกมาบนคอมพิวเตอร์
6. ทำการรันโปรแกรมการทำงานแบบ AUTO
7. ป้อนค่าให้กับแทงก์น้ำที่ 1 , 2 และ ค่า P I D
8. กดปุ่ม START โปรแกรมเริ่ม Step การทำงาน
9. สังเกตผลการทำงานบนหน้าจอคอมพิวเตอร์และการทำงานที่ตัว plant ทดลอง

ผลการทดลอง

จากการทดลองผลปรากฏว่าการควบคุมการทำงานชุดประยุกต์แบบ AUTO คือค่าที่จะสั่งให้ตัวโซลีนอยด์ทั้งสามตัวทำงาน คือ ค่าของ P I D ซึ่งมีการคำนวนสูตรไว้ในโปรแกรมแล้ว ผลคือค่า P I D คำนวณเปอร์เซนต์ได้ค่าตามที่กำหนดเปอร์เซนต์ของตัวคอนโทรลวาล์วไว้ ตัวโซลีนอยด์ก็จะทำการเปิด – ปิด ให้น้ำไหลผ่านมาก หรือน้อย ตามเปอร์เซนต์ที่คำนวณได้ การทำงานแบบ AUTO สามารถทำงานได้ง่ายโดยไม่ต้องควบคุมมาก เพียงกำหนดค่าที่ต้องการตามส่วนของแทงก์น้ำและ ค่า P I D เท่านั้นก็จะสามารถทำงานได้ รูปลักษณะการทำงานเหมือนกับรูปที่ 4.9 และมีการกำหนดเปอร์เซนต์ระดับน้ำเหมือนกัน

บทที่ 5
บทสรุป ปัญหาแนวทางแก้ไขและพัฒนา

5.1 บทสรุป

ชุดประยุกต์ใช้งานควบคุมระดับน้ำด้วยเครื่องไมโครคอมพิวเตอร์ ซึ่งการสร้างได้แบ่งออกเป็น 2 ส่วนใหญ่ๆ คือ ส่วนของ Soft ware และ Hard ware

1. การเขียนโปรแกรมควบคุมกระบวนการ สามารถควบคุมกระบวนการของระดับน้ำได้ในรูปแบบของ AUTO และ MANUAL และแสดงสภาวะผลการทำงานได้ที่หน้าจอของตัว PC ด้วยโปรแกรม Microsoft Visual C++ Version 6.0

2. ชุดประยุกต์ใช้งานการควบคุมระดับน้ำ สามารถรับค่าสัญญาณและส่งสัญญาณผ่านชุดขับสัญญาณ เข้าตัวการ์ด ET – PC DIO เพื่อรับคำสั่งจากโปรแกรมการทำงานมาควบคุมตัวชุดประยุกต์ใช้งานให้ทำงานตามต้องการ

5.2 ปัญหาที่เกิดขึ้นในการจัดทำโครงงาน

ในการจัดทำโครงงานชุดนี้ สามารถสรุปปัญหาที่เกิดขึ้นได้ดังนี้

1.ปัญหาในการเขียนโปรแกรม คือโปรแกรมที่ใช้ในการเขียนควบคุมทางด้าน PC นั้นเป็นการใช้โปรแกรม Visual C++ V 6.0 ซึ่งต้องใช้เวลาเป็นอย่างมากและโปรแกรมเมื่อเขียนแล้วก็ยังต้องมีการแก้ไขอยู่เรื่อยๆ

2. ปัญหาการเขียนโปรแกรม คือ ผู้จัดทำโครงงานยังขาดประสบการณ์และข้อมูลในการเขียนโปรแกรมและมีความรู้พื้นฐานในโปรแกรมที่เขียนน้อยมาก

3. ปัญหาการแบ่งงานภายในกลุ่ม ผู้จัดทำโครงงานยังขาดการวางแผนงานให้เป็นระบบ จึงทำให้เกิดการดำเนินงานล่าช้า

4. ปัญหาด้านอุปกรณ์ที่ใช้ในการทดลองเกิดการขัดข้อง และอุปกรณ์ที่ต้องการบางอย่างมีราคาแพงและหาอุปกรณ์ยาก

5. ปัญหาในการทดลองส่วนรวมโปรแกรมเพื่อควบคุมตัว PLANT ยังไม่สัมพันธ์กันเท่าที่ควร เกิดค่าความผิดพลาดขึ้นพอสมควร จึงทำให้การทดลองได้ผลคลาดเคลื่อนไปบ้าง

5.3 แนวทางการแก้ไขและพัฒนา

แนวทางการแก้ไขและพัฒนาชุดประยุกต์การใช้งานนี้ คือ

1. ในกรณีของอุปกรณ์ในการทำโครงงานนี้ ได้ทำการใช้อุปกรณ์ที่มีราคาถูกกว่าและมีคุณสมบัติใกล้เคียงกันทดแทนอุปกรณ์ที่มีราคาแพง

2.ในด้านระยะเวลาในการทำโครงงานนั้นตอนแรกไม่ค่อยมีเวลาทำ เพราะว่าต้องเรียนด้วยและการจัดแบ่งเวลาในการทำมีไม่มากนัก จึงต้องมาช่วยกันในส่วนที่ทำค่อนข้างยากก่อน

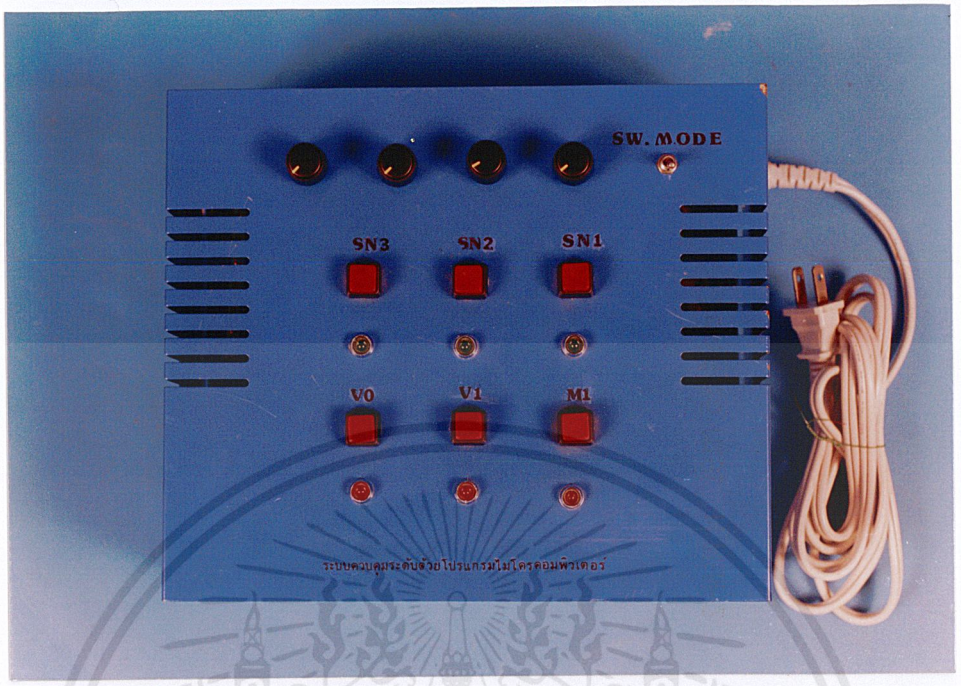
3. เนื่องจากโปรแกรมที่ใช้เขียนมีความซับซ้อนมากพอสมควร จึงได้ขอคำปรึกษาจากคณาจารย์ที่มีความเชี่ยวชาญในด้านโปรแกรมที่เขียน และการค้นหาฟังก์ชั่นที่ใช้งานจริงในการทำโครงงานทำให้เข้าใจง่ายขึ้น

4. การออกแบบและการทำงานเชื่อมต่อกับอุปกรณ์และโปรแกรมภายในโครงงาน ใช้คู่มือการทำงานและการออกแบบโดยใช้ ผังการทำงาน ทำให้ออกแบบได้ถูกต้องและเข้าใจหลักการทำงานไปด้วย

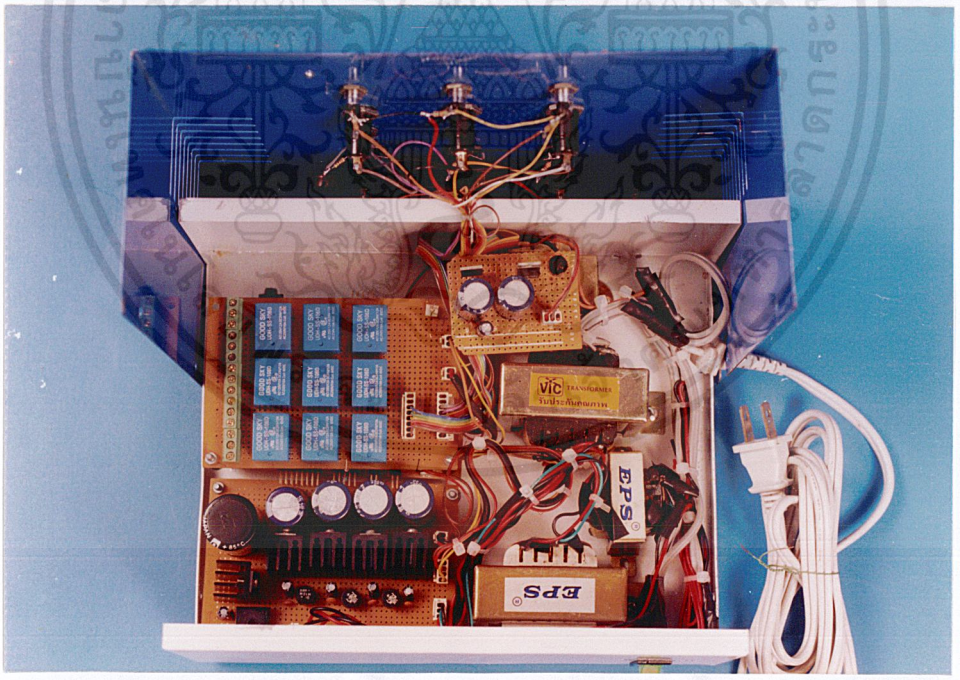
5. การทดลองในส่วนการเชื่อมต่อกับอุปกรณ์การทดลองควรมีการเพิ่มเติมวงจรรองรับการทำงานของชุดประยุกต์ใช้งานให้มีประสิทธิภาพดีขึ้น

6. ควรพัฒนาหลักสูตรการเรียนการสอนให้มีวิชาทางด้าน Instrumentenation And Process Control ทั้งยังหลักการเขียนโปรแกรมและโปรแกรมใหม่ๆที่น่าสนใจและประยุกต์ใช้ให้มากขึ้น

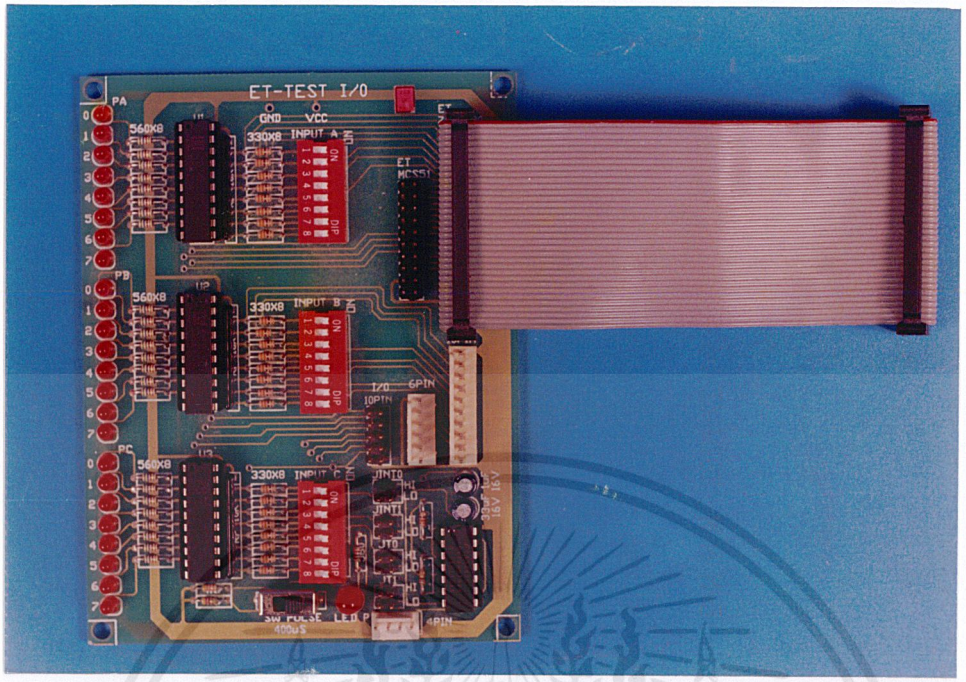
ภาคผนวก ก
รูปอุปกรณ์ที่ใช้ในโครงงาน



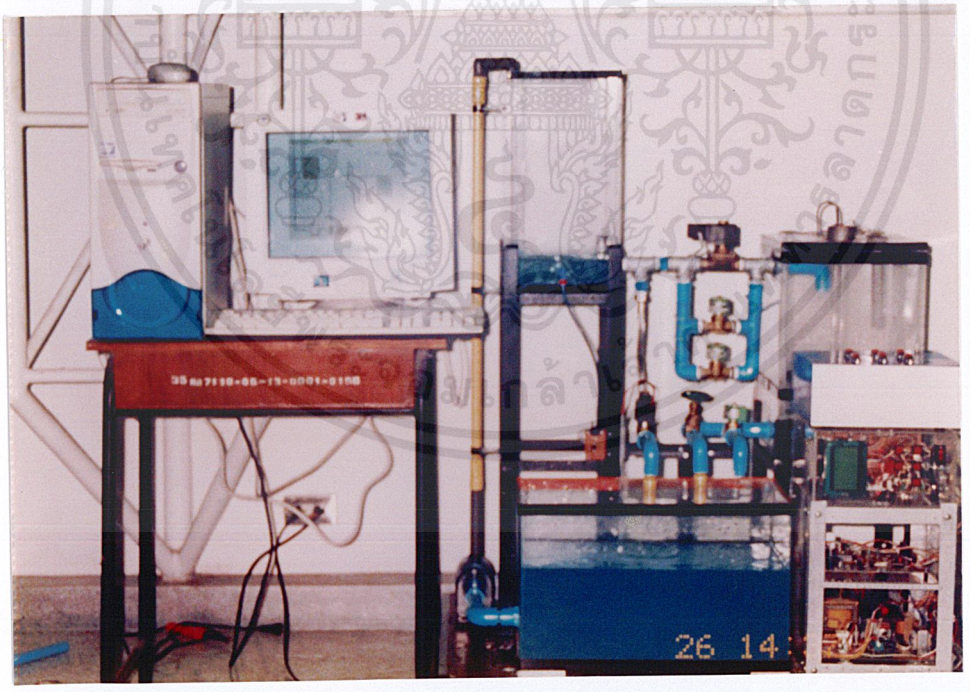
รูปที่ ก.1 ชุดประยุกต์ใช้งานควบคุมระดับ



รูปที่ ก.2 ภายในชุดขับสัญญาณโซลีนอยด์วาล์ว

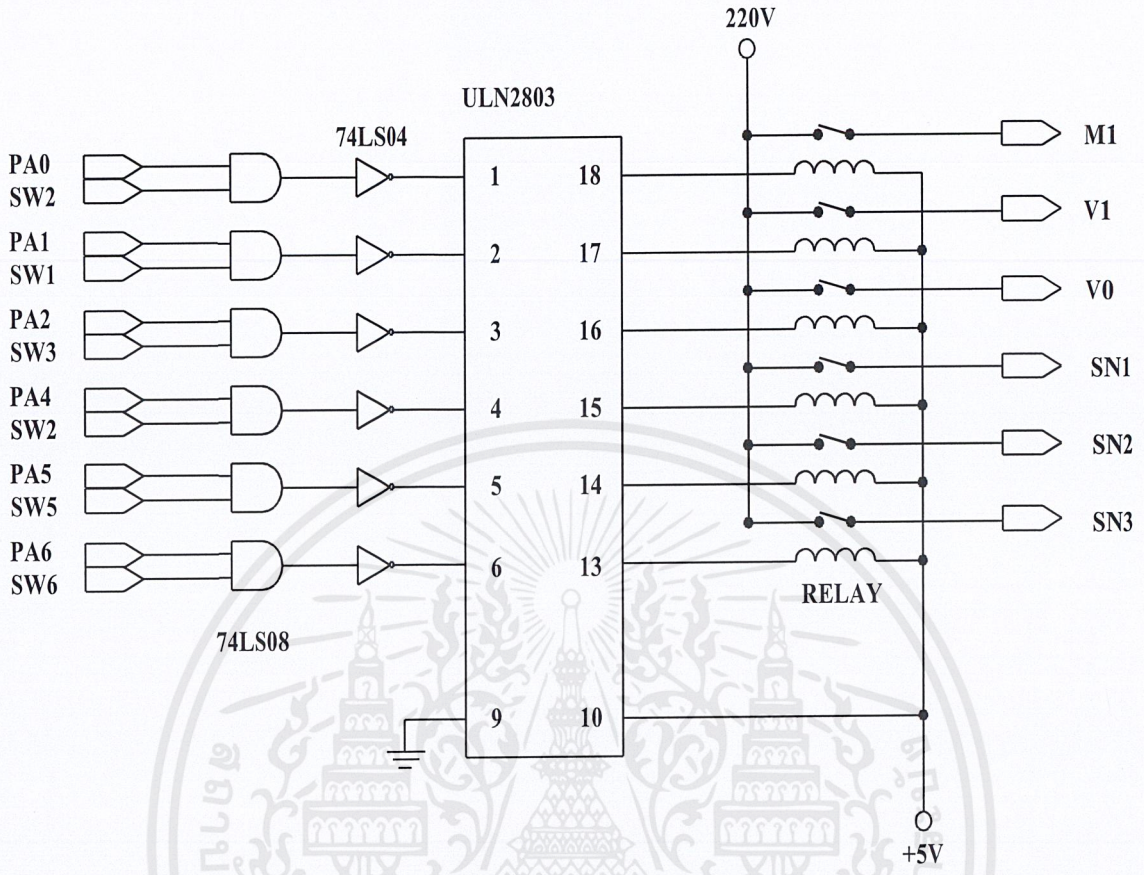


รูปที่ ก.3 ชุดควบคุมขับสัญญาณโซลีนอยด์วาล์ว

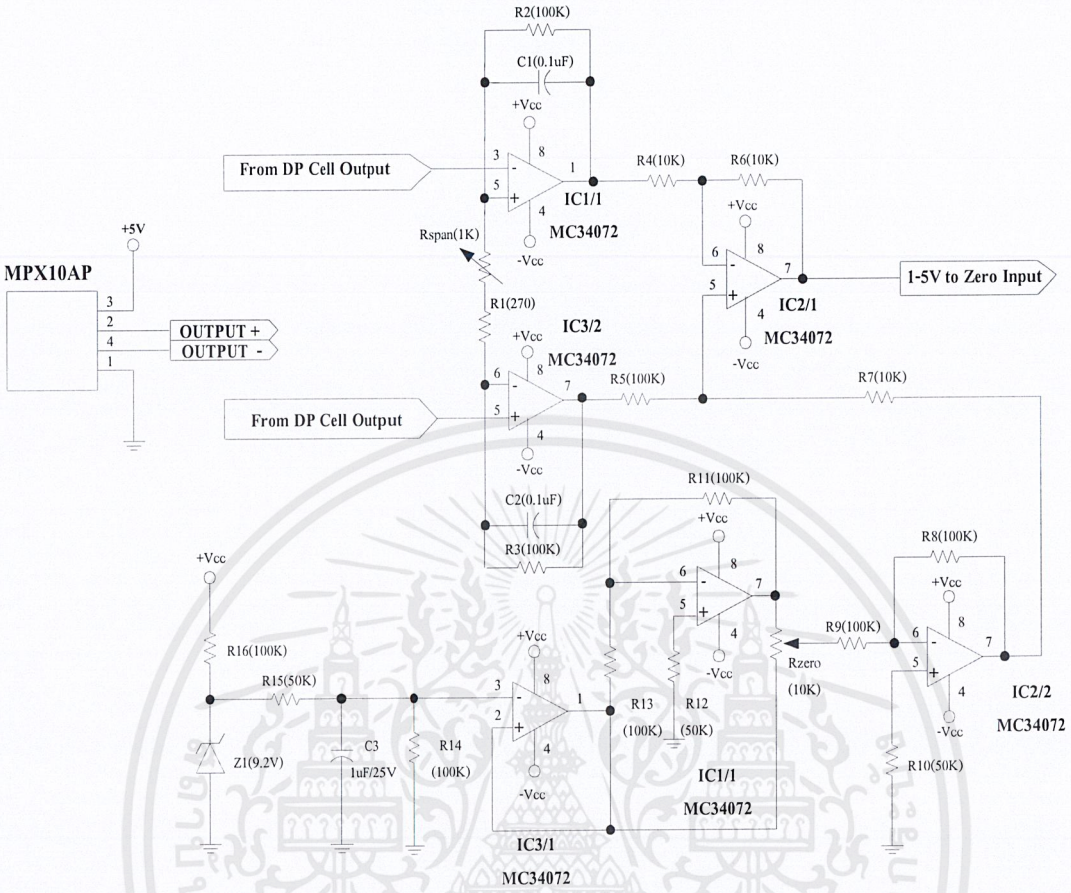


รูปที่ ก.4 ชุดประยุกต์ใช้งานพร้อมการควบคุมทางคอมพิวเตอร์

ภาคผนวก ข
วงจรต่างๆที่ใช้ในโครงงาน



รูปที่ ข.1 วงจรขับสัญญาณโซลีนอยด์วาล์ว



รูปที่ ข.2 วงจร Instrument Amplifier

ภาคผนวก ค

โปรแกรมการทำงาน

1. การประกาศคลาสและตัวแปรที่ใช้ในโปรแกรมชุดควบคุมระดับน้ำ

```
// ProcessCTRL.h : main header file for the PROCESSCTRL application
//
#if!defined(AFX_PROCESSCTRL_H__655F8963_CC8A_11D4_82FF_FF079E21A832_
_INCLUDED_)
#define
AFX_PROCESSCTRL_H__655F8963_CC8A_11D4_82FF_FF079E21A832__INCLUDED_
#if _MSC_VER > 1000
#pragma one
#endif // _MSC_VER > 1000
#ifndef __AFXWIN_H__
#error include 'stdafx.h' before including this file for PCH
#endif
#include "resource.h"          // main symbols

/////////////////////////////////////////////////////////////////////////

// CProcessCTRLApp:
// See ProcessCTRL.cpp for the implementation of this class //
class CProcessCTRLApp : public CWinApp
{
public:
      CProcessCTRLApp();
// Overrides
// ClassWizard generated virtual function overrides
      //{{AFX_VIRTUAL(CProcessCTRLApp)
      public:
      virtual BOOL InitInstance();
      //}}AFX_VIRTUAL
// Implementation
      //{{AFX_MSG(CProcessCTRLApp)
      //}}AFX_MSG
      DECLARE_MESSAGE_MAP()
};
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations
immediately before the previous line.
#endif //
!defined(AFX_PROCESSCTRL_H__655F8963_CC8A_11D4_82FF_FF079E21A832__IN
CLUDED_)

/////////////////////////////////////////////////////////////////////////
```

2. ส่วนเนื้อหาของฟังก์ชั่นในการเลือกใช้โปรแกรม

```
// ProcessCTRL.cpp : Defines the class behaviors for the
application.//
#include "stdafx.h"
#include "ProcessCTRL.h"
#include "ProcessCTRLDlg.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
```

```
#endif

/////////////////////////////////////////////////////////////////////////

// CProcessCTRLApp
BEGIN_MESSAGE_MAP(CProcessCTRLApp, CWinApp)
	//{{AFX_MSG_MAP(CProcessCTRLApp)
	//}}AFX_MSG
	ON_COMMAND(ID_HELP, CWinApp::OnHelp)
	END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////

// CProcessCTRLApp construction
CProcessCTRLApp::CProcessCTRLApp()
	{
	}
// The one and only CProcessCTRLApp object
CProcessCTRLApp theApp;
// CProcessCTRLApp initialization
BOOL CProcessCTRLApp::InitInstance()
{
	AfxEnableControlContainer();		// Standard initialization
#ifdef _AFXDLL
	Enable3dControls();			// Call this when using MFC
in a shared DLL
#else
	Enable3dControlsStatic();		// Call this when linking
to MFC statically
#endif
	CProcessCTRLDlg dlg;
	m_pMainWnd = &dlg;
	int nResponse = dlg.DoModal();
	if (nResponse == IDOK)
	{
	}
	else if (nResponse == IDCANCEL)
	{
	}
// Since the dialog has been closed, return FALSE so that we exit
the
//  application, rather than start the application's message pump.
	return FALSE;
	}
```

3. การประกาศไฟล์ที่ใช้ไดอะล็อกในโปรแกรม

```
// ProcessCTRLDlg.h : header file//
#if
!defined(AFX_PROCESSCTRLDLG_H__655F8965_CC8A_11D4_82FF_FF079E21A832_
_INCLUDED_)
#define
AFX_PROCESSCTRLDLG_H__655F8965_CC8A_11D4_82FF_FF079E21A832__INCLUDED
_
```

```
#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

/////////////////////////////////////////////////////////////////////////

// CProcessCTRLDlg dialog
class CProcessCTRLDlg : public CDialog
{

// Construction
public:
      CProcessCTRLDlg(CWnd* pParent = NULL);// standard constructor

//My function
      void initmanual();
      void initauto();
      void pid_valve();

      int old_error;
      unsigned short portA;
      unsigned short TankB;
      bool isTank1;
      bool isStart;
      bool isAuto;
      bool justEnd;


//end My function

// Dialog Data
      //{{AFX_DATA(CProcessCTRLDlg)
      enum { IDD = IDD_PROCESSCTRL_DIALOG };
      CEdit m_status2;
      CEdit m_status1;
      CSpinButtonCtrl   m_tank2_spin;
      CEdit m_tank2_edit;
      CSpinButtonCtrl   m_tank1_spin;
      CEdit m_tank1_edit;
      CSpinButtonCtrl   m_p_spin;
      CEdit m_p_edit;
      CSpinButtonCtrl   m_i_spin;
      CEdit m_i_edit;
      CSpinButtonCtrl   m_d_spin;
      CEdit m_d_edit;
      CEdit m_pid_edit;
      CSpinButtonCtrl   m_pid_spin;
      CProgressCtrl     m_tank3;
      CProgressCtrl     m_tank2;
      CProgressCtrl     m_tank1;
      int         m_d_edit_value;
      int         m_i_edit_value;
      int         m_p_edit_value;
      int         m_pid_edit_value;
```

```
	int		m_tank1_edit_value;
	int		m_tank2_edit_value;
	CString		m_str_status1;
	CString		m_str_status2;
//}}AFX_DATA

// ClassWizard generated virtual function overrides
//{{AFX_VIRTUAL(CProcessCTRLDlg)
	protected:
	virtual void DoDataExchange(CDataExchange* pDX);   // DDX/DDV
support

//}}AFX_VIRTUAL

// Implementation
protected:
	HICON m_hIcon;

	// Generated message map functions
	//{{AFX_MSG(CProcessCTRLDlg)
	virtual BOOL OnInitDialog();
	afx_msg void OnSysCommand(UINT nID, LPARAM lParam);
	afx_msg void OnPaint();
	afx_msg HCURSOR OnQueryDragIcon();
	afx_msg void OnChangemode();
	afx_msg void OnTimer(UINT nIDEvent);
	afx_msg void OnM1on();
	afx_msg void OnM1off();
	afx_msg void OnV0on();
	afx_msg void OnV0off();
	afx_msg void OnV1on();
	afx_msg void OnV1off();
	afx_msg void OnChangePidEdit();
	afx_msg void OnStart();
	afx_msg void OnStop();
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations
immediately before the previous line.

#endif //
!defined(AFX_PROCESSCTRLDLG_H__655F8965_CC8A_11D4_82FF_FF079E21A832_
_INCLUDED_)
```

4.ส่วนของเนื้อหาการกำหนดฟังกชั่นไดอะล๊อกในการใช้งานของโปรแกรม

```
// ProcessCTRLDlg.cpp : implementation file//

#include "stdafx.h"
#include "ProcessCTRL.h"
#include "ProcessCTRLDlg.h"
```

```
#include <conio.h>

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////

// CAboutDlg dialog used for App About

class CAboutDlg : public CDialog
{
public:
	CAboutDlg();

// Dialog Data
	//{{AFX_DATA(CAboutDlg)
	enum { IDD = IDD_ABOUTBOX };
	//}}AFX_DATA

	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CAboutDlg)
	protected:
	virtual void DoDataExchange(CDataExchange* pDX);  // DDX/DDV
support
	//}}AFX_VIRTUAL

// Implementation
protected:

	//{{AFX_MSG(CAboutDlg)
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
	//{{AFX_DATA_INIT(CAboutDlg)
	//}}AFX_DATA_INIT
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CAboutDlg)
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
	//{{AFX_MSG_MAP(CAboutDlg)
		// No message handlers
	//}}AFX_MSG_MAP
```

```
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////

// CProcessCTRLDlg dialog

CProcessCTRLDlg::CProcessCTRLDlg(CWnd* pParent /*=NULL*/)
	: CDialog(CProcessCTRLDlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(CProcessCTRLDlg)
	m_d_edit_value = 0;
	m_i_edit_value = 0;
	m_p_edit_value = 0;
	m_pid_edit_value = 0;
	m_tank1_edit_value = 0;
	m_tank2_edit_value = 0;
	m_str_status1 = _T("");
	m_str_status2 = _T("");
	//}}AFX_DATA_INIT
	m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CProcessCTRLDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CProcessCTRLDlg)
	DDX_Control(pDX, IDC_STATUS2, m_status2);
	DDX_Control(pDX, IDC_STATUS1, m_status1);
	DDX_Control(pDX, IDC_TANK2_SPIN, m_tank2_spin);
	DDX_Control(pDX, IDC_TANK2_EDIT, m_tank2_edit);
	DDX_Control(pDX, IDC_TANK1_SPIN, m_tank1_spin);
	DDX_Control(pDX, IDC_TANK1_EDIT, m_tank1_edit);
	DDX_Control(pDX, IDC_P_SPIN, m_p_spin);
	DDX_Control(pDX, IDC_P_EDIT, m_p_edit);
	DDX_Control(pDX, IDC_I_SPIN, m_i_spin);
	DDX_Control(pDX, IDC_I_EDIT, m_i_edit);
	DDX_Control(pDX, IDC_D_SPIN, m_d_spin);
	DDX_Control(pDX, IDC_D_EDIT, m_d_edit);
	DDX_Control(pDX, IDC_PID_EDIT, m_pid_edit);
	DDX_Control(pDX, IDC_PID_SPIN, m_pid_spin);
	DDX_Control(pDX, IDC_TANK3, m_tank3);
	DDX_Control(pDX, IDC_TANK2, m_tank2);
	DDX_Control(pDX, IDC_TANK1, m_tank1);
	DDX_Text(pDX, IDC_D_EDIT, m_d_edit_value);
	DDX_Text(pDX, IDC_I_EDIT, m_i_edit_value);
	DDX_Text(pDX, IDC_P_EDIT, m_p_edit_value);
	DDX_Text(pDX, IDC_PID_EDIT, m_pid_edit_value);
	DDX_Text(pDX, IDC_TANK1_EDIT, m_tank1_edit_value);
	DDX_Text(pDX, IDC_TANK2_EDIT, m_tank2_edit_value);
	DDX_Text(pDX, IDC_STATUS1, m_str_status1);
	DDX_Text(pDX, IDC_STATUS2, m_str_status2);
	//}}AFX_DATA_MAP
}
```

```
BEGIN_MESSAGE_MAP(CProcessCTRLDlg, CDialog)
	//{{AFX_MSG_MAP(CProcessCTRLDlg)
	ON_WM_SYSCOMMAND()
	ON_WM_PAINT()
	ON_WM_QUERYDRAGICON()
	ON_BN_CLICKED(IDC_CHANGEMODE, OnChangemode)
	ON_WM_TIMER()
	ON_BN_CLICKED(IDC_M10N, OnM1on)
	ON_BN_CLICKED(IDC_M10FF, OnM1off)
	ON_BN_CLICKED(IDC_V00N, OnV0on)
	ON_BN_CLICKED(IDC_V00FF, OnV0off)
	ON_BN_CLICKED(IDC_V10N, OnV1on)
	ON_BN_CLICKED(IDC_V10FF, OnV1off)
	ON_EN_CHANGE(IDC_PID_EDIT, OnChangePidEdit)
	ON_BN_CLICKED(IDC_START, OnStart)
	ON_BN_CLICKED(IDC_STOP, OnStop)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////

// CProcessCTRLDlg message handlers

BOOL CProcessCTRLDlg::OnInitDialog()
{
	CDialog::OnInitDialog();

	// Add "About..." menu item to system menu.

	// IDM_ABOUTBOX must be in the system command range.
	ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
	ASSERT(IDM_ABOUTBOX < 0xF000);

	CMenu* pSysMenu = GetSystemMenu(FALSE);
	if (pSysMenu != NULL)
	{
		CString strAboutMenu;
		strAboutMenu.LoadString(IDS_ABOUTBOX);
		if (!strAboutMenu.IsEmpty())
		{
			pSysMenu->AppendMenu(MF_SEPARATOR);
			pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX,
strAboutMenu);
		}
	}

	SetIcon(m_hIcon, TRUE);			// Set big icon
	SetIcon(m_hIcon, FALSE);		// Set small icon

	// TODO: Add extra initialization here

	SetTimer(ID_CLOCK_TIMER,50,NULL);

	_outp(0x303,0x88);
```

```
	portA = 0xFF;
	_outp(0x300,portA);

	old_error=0;

	isTank1=true;
	_outp(0x301,0x02);//0000 0010

	isStart=false;

	m_tank1.SetRange(0,100); m_tank1.SetStep(1); m_tank1.SetPos
(0);
	m_tank2.SetRange(0,100); m_tank2.SetStep(1); m_tank2.SetPos
(0);
	m_tank3.SetRange(0,100); m_tank3.SetStep(1); m_tank3.SetPos
(75);

	initmanual();



	return TRUE;  // return TRUE  unless you set the focus to a
control
}

void CProcessCTRLDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
	if ((nID & 0xFFF0) == IDM_ABOUTBOX)
	{
		CAboutDlg dlgAbout;
		dlgAbout.DoModal();
	}
	else
	{
		CDialog::OnSysCommand(nID, lParam);
	}
}

// If you add a minimize button to your dialog, you will need the
code below
//  to draw the icon.  For MFC applications using the document/view
model,
//  this is automatically done for you by the framework.

void CProcessCTRLDlg::OnPaint()
{
	if (IsIconic())
	{
		CPaintDC dc(this); // device context for painting

		SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(),
0);
```

```
            // Center icon in client rectangle
            int cxIcon = GetSystemMetrics(SM_CXICON);
            int cyIcon = GetSystemMetrics(SM_CYICON);
            CRect rect;
            GetClientRect(&rect);
            int x = (rect.Width() - cxIcon + 1) / 2;
            int y = (rect.Height() - cyIcon + 1) / 2;

            // Draw the icon
            dc.DrawIcon(x, y, m_hIcon);
      }
      else
      {
            CDialog::OnPaint();
      }
}

HCURSOR CProcessCTRLDlg::OnQueryDragIcon()
{
      return (HCURSOR) m_hIcon;
}

void CProcessCTRLDlg::initmanual()
{
      isAuto=false;
      isStart=false;
      justEnd=false;

      portA = 0xFF;
      _outp(0x300,portA);

      this->SetWindowText("ProcessCTRL : manual mode");


      GetDlgItem(IDC_TANK1_EDIT)->EnableWindow(0); GetDlgItem
(IDC_TANK1_SPIN)->EnableWindow(0);
      GetDlgItem(IDC_TANK2_EDIT)->EnableWindow(0); GetDlgItem
(IDC_TANK2_SPIN)->EnableWindow(0);
      GetDlgItem(IDC_START)->EnableWindow(0); GetDlgItem(IDC_STOP)->
EnableWindow(0);

      GetDlgItem(IDC_PID_SPIN)->EnableWindow(1);

      m_pid_spin.SetRange(0,100); m_pid_spin.SetBuddy(&m_pid_edit);
m_pid_spin.SetPos(0);

      GetDlgItem(IDC_P_EDIT)->EnableWindow(0); GetDlgItem
(IDC_P_SPIN)->EnableWindow(0);
      GetDlgItem(IDC_I_EDIT)->EnableWindow(0); GetDlgItem
(IDC_I_SPIN)->EnableWindow(0);
      GetDlgItem(IDC_D_EDIT)->EnableWindow(0); GetDlgItem
(IDC_D_SPIN)->EnableWindow(0);
```

```
	CButton *m1on = (CButton*)GetDlgItem(IDC_M1ON);m1on->SetCheck
(0);
	CButton *v0on = (CButton*)GetDlgItem(IDC_V0ON);v0on->SetCheck
(0);
	CButton *v1on = (CButton*)GetDlgItem(IDC_V1ON);v1on->SetCheck
(0);

	CButton *m1off = (CButton*)GetDlgItem(IDC_M1OFF);m1off->
SetCheck(1);
	CButton *v0off = (CButton*)GetDlgItem(IDC_V0OFF);v0off->
SetCheck(1);
	CButton *v1off = (CButton*)GetDlgItem(IDC_V1OFF);v1off->
SetCheck(1);

	m_pid_edit.SetReadOnly(false);

}

void CProcessCTRLDlg::initauto(){

	isAuto=true;
	isStart=false;
	justEnd=false;

	portA = 0xFF;
	_outp(0x300,portA);

	this->SetWindowText("ProcessCTRL : automatic mode");

	GetDlgItem(IDC_TANK1_EDIT)->EnableWindow(1); GetDlgItem
(IDC_TANK1_SPIN)->EnableWindow(1);
	GetDlgItem(IDC_TANK2_EDIT)->EnableWindow(1); GetDlgItem
(IDC_TANK2_SPIN)->EnableWindow(1);
	GetDlgItem(IDC_START)->EnableWindow(1); GetDlgItem(IDC_STOP)->
EnableWindow(1);

	GetDlgItem(IDC_PID_SPIN)->EnableWindow(0);

	GetDlgItem(IDC_P_EDIT)->EnableWindow(1); GetDlgItem
(IDC_P_SPIN)->EnableWindow(1);
	GetDlgItem(IDC_I_EDIT)->EnableWindow(1); GetDlgItem
(IDC_I_SPIN)->EnableWindow(1);
	GetDlgItem(IDC_D_EDIT)->EnableWindow(1); GetDlgItem
(IDC_D_SPIN)->EnableWindow(1);
	m_tank1_spin.SetRange(0,100); m_tank1_spin.SetBuddy
(&m_tank1_edit); m_tank1_spin.SetPos(0);
	m_tank2_spin.SetRange(0,100); m_tank2_spin.SetBuddy
(&m_tank2_edit); m_tank2_spin.SetPos(0);

	m_p_spin.SetRange(0,100); m_p_spin.SetBuddy(&m_p_edit);
m_p_spin.SetPos(0);
	m_i_spin.SetRange(0,100); m_i_spin.SetBuddy(&m_i_edit);
m_i_spin.SetPos(0);
```

```
      m_d_spin.SetRange(0,100); m_d_spin.SetBuddy(&m_d_edit);
m_d_spin.SetPos(0);

CButton *m1on = (CButton*)GetDlgItem(IDC_M1ON);m1on->SetCheck(0);
CButton *v0on = (CButton*)GetDlgItem(IDC_V0ON);v0on->SetCheck(0);
CButton *v1on = (CButton*)GetDlgItem(IDC_V1ON);v1on->SetCheck(0);
CButton *m1off =(CButton*)GetDlgItem(IDC_M1OFF);m1off->SetCheck(1);
CButton *v0off =(CButton*)GetDlgItem(IDC_V0OFF);v0off->SetCheck(1);
CButton *v1off = (CButton*)GetDlgItem(IDC_V1OFF);v1off->SetCheck(1);

      m_pid_edit.SetReadOnly(true);
}
void CProcessCTRLDlg::OnChangemode()
{

      int result_mode;

      if(isAuto){
result_mode = MessageBox("Are you sure to switch to manual mode ?",

                  "switch mode",MB_YESNO);
      }else{
result_mode = MessageBox("Are you sure to switch to automatic mode
?",

                  "switch mode",MB_YESNO);
      }

      switch(result_mode){
      case IDYES:
            if(isAuto){
            this->SetWindowText("ProcessCTRL : manual mode");
      initmanual();
            }else{
      this->SetWindowText("ProcessCTRL : automatic mode");
      initauto();
            }
            break;
      case IDNO:;
      }
}

void CProcessCTRLDlg::OnTimer(UINT nIDEvent)
{
// TODO: Add your message handler code here and/or call default
      UpdateData(true);

      CButton *m1on, *m1off, *v1on, *v1off, *v0on, *v0off;

      m1on = (CButton*)GetDlgItem(IDC_M1ON);
      m1off = (CButton*)GetDlgItem(IDC_M1OFF);

      v1on = (CButton*)GetDlgItem(IDC_V1ON);
      v1off = (CButton*)GetDlgItem(IDC_V1OFF);
```

```
      v0on = (CButton*)GetDlgItem(IDC_V0ON);
      v0off = (CButton*)GetDlgItem(IDC_V0OFF);


      unsigned short portB;
      if(isTank1){
            portB = _inp(0x30A);
            m_tank1.SetPos(portB*100/255);
            m_str_status1.Format("%d",portB*100/255);
            m_status1.SetWindowText(m_str_status1);

            //auto mode//

            if(isAuto){

            if( (m_tank1_edit_value) < (portB*100/255) ){
//m1 off, v1 on
            m1on->SetCheck(0); m1off->SetCheck(1); this->OnM1off();
            v1on->SetCheck(1); v1off->SetCheck(0); this->OnV1on();
            }else if( (m_tank1_edit_value) > (portB*100/255) ){
//m1 on, v1 off
            m1on->SetCheck(1); m1off->SetCheck(0); this->OnM1on();
            v1on->SetCheck(0); v1off->SetCheck(1); this->OnV1off();
                  }
            }

//end//
            _outp(0x301,0x04);//0000 0100
            isTank1 = false;
      }else{
            portB = _inp(0x30A);
            TankB = portB;
            m_tank2.SetPos(portB*100/255);
            m_str_status2.Format("%d",portB*100/255);
            m_status2.SetWindowText(m_str_status2);

      //auto mode//

            if(isAuto){

            if(isStart){

            if( (m_tank2_edit_value) < (portB*100/255) ){
// v0 on
      v0on->SetCheck(1); v0off->SetCheck(0); this->OnV0on();
      }else if( (m_tank2_edit_value) > (portB*100/255) ){
// v0 off
      v0on->SetCheck(0); v0off->SetCheck(1); this->OnV0off();
      }else{
                  isStart=false;
                  justEnd=true;
                  }
            }
```

```
                    if(justEnd){
// v0 off
      v0on->SetCheck(0); v0off->SetCheck(1); this->OnV0off();
      m_p_spin.SetPos(0); m_i_spin.SetPos(0); m_d_spin.SetPos(0);
                    pid_valve();
                    justEnd=false;
                    }

              }

//end//
              _outp(0x301,0x02);//0000 00010
              isTank1 = true;
      }


      CDialog::OnTimer(nIDEvent);
}

void CProcessCTRLDlg::OnM1on()
{

// TODO: Add your control notification handler code here

      portA = portA & 0xFE;//1111 1110
      _outp(0x300,portA);
}

void CProcessCTRLDlg::OnM1off()
{

// TODO: Add your control notification handler code here

      portA = portA | 0x01;//0000 0001
      _outp(0x300,portA);

}

void CProcessCTRLDlg::OnV0on()
{

// TODO: Add your control notification handler code here

      portA = portA & 0xFB;//1111 1011
      _outp(0x300,portA);

}

void CProcessCTRLDlg::OnV0off()
{

// TODO: Add your control notification handler code here
```

```
	portA = portA | 0x04;//0000 0100
	_outp(0x300,portA);

}

void CProcessCTRLDlg::OnV1on()
{

// TODO: Add your control notification handler code here

	portA = portA & 0xFD;//1111 1101
	_outp(0x300,portA);

}

void CProcessCTRLDlg::OnV1off()
{
	// TODO: Add your control notification handler code here

	portA = portA | 0x02;//0000 0010
	_outp(0x300,portA);

}

void CProcessCTRLDlg::OnChangePidEdit()
{
	// TODO: If this is a RICHEDIT control, the control will not
	// send this notification unless you override the
CDialog::OnInitDialog()
	// function and call CRichEditCtrl().SetEventMask()
	// with the ENM_CHANGE flag ORed into the mask.

	// TODO: Add your control notification handler code here
	UpdateData(true);

	if(m_pid_edit_value==0) portA = portA | 0x38;//0011 1000
	else if(m_pid_edit_value<=33){
		portA = portA & 0xF7;//1111 0111
		portA = portA | 0x30;//0011 0000
	}else if(m_pid_edit_value<=66){
		portA = portA & 0xE7;//1110 0111
		portA = portA | 0x20;//0010 0000
	}else portA = portA & 0xC7;//1100 0111

	_outp(0x300,portA);
}

void CProcessCTRLDlg::pid_valve(){

	int A, B, C, P, I, D, m;
	float M;

	A = m_tank2_edit_value;
	B = (int)TankB*100/255;
```

1. การประกาศคลาสและตัวแปรที่ใช้ในโปรแกรมชุดควบคุมระดับน้ำ

```
// ProcessCTRL.h : main header file for the PROCESSCTRL application
//
#if!defined(AFX_PROCESSCTRL_H__655F8963_CC8A_11D4_82FF_FF079E21A832_
_INCLUDED_)
#define
AFX_PROCESSCTRL_H__655F8963_CC8A_11D4_82FF_FF079E21A832__INCLUDED_
#if _MSC_VER > 1000
#pragma one
#endif // _MSC_VER > 1000
#ifndef __AFXWIN_H__
#error include 'stdafx.h' before including this file for PCH
#endif
#include "resource.h"          // main symbols

/////////////////////////////////////////////////////////////////////////

// CProcessCTRLApp:
// See ProcessCTRL.cpp for the implementation of this class //
class CProcessCTRLApp : public CWinApp
{
public:
      CProcessCTRLApp();
// Overrides
// ClassWizard generated virtual function overrides
      //{{AFX_VIRTUAL(CProcessCTRLApp)
      public:
      virtual BOOL InitInstance();
      //}}AFX_VIRTUAL
// Implementation
      //{{AFX_MSG(CProcessCTRLApp)
      //}}AFX_MSG
      DECLARE_MESSAGE_MAP()
};
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations
immediately before the previous line.
#endif //
!defined(AFX_PROCESSCTRL_H__655F8963_CC8A_11D4_82FF_FF079E21A832__IN
CLUDED_)

/////////////////////////////////////////////////////////////////////////
```

2. ส่วนเนื้อหาของฟังก์ชั่นในการเลือกใช้โปรแกรม

```
// ProcessCTRL.cpp : Defines the class behaviors for the
application.//
#include "stdafx.h"
#include "ProcessCTRL.h"
#include "ProcessCTRLDlg.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
```

```
#endif

/////////////////////////////////////////////////////////////////////////

// CProcessCTRLApp
BEGIN_MESSAGE_MAP(CProcessCTRLApp, CWinApp)
	//{{AFX_MSG_MAP(CProcessCTRLApp)
	//}}AFX_MSG
	ON_COMMAND(ID_HELP, CWinApp::OnHelp)
	END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////

// CProcessCTRLApp construction
CProcessCTRLApp::CProcessCTRLApp()
	{
	}
// The one and only CProcessCTRLApp object
CProcessCTRLApp theApp;
// CProcessCTRLApp initialization
BOOL CProcessCTRLApp::InitInstance()
{
	AfxEnableControlContainer();		// Standard initialization
#ifdef _AFXDLL
	Enable3dControls();			// Call this when using MFC
in a shared DLL
#else
	Enable3dControlsStatic();		// Call this when linking
to MFC statically
#endif
	CProcessCTRLDlg dlg;
	m_pMainWnd = &dlg;
	int nResponse = dlg.DoModal();
	if (nResponse == IDOK)
	{
	}
	else if (nResponse == IDCANCEL)
	{
	}
// Since the dialog has been closed, return FALSE so that we exit
the
//  application, rather than start the application's message pump.
	return FALSE;
	}
```

3. การประกาศไฟล์ที่ใช้ไดอะล็อกในโปรแกรม

```
// ProcessCTRLDlg.h : header file//
#if
!defined(AFX_PROCESSCTRLDLG_H__655F8965_CC8A_11D4_82FF_FF079E21A832_
_INCLUDED_)
#define
AFX_PROCESSCTRLDLG_H__655F8965_CC8A_11D4_82FF_FF079E21A832__INCLUDED
_
```

```
#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

/////////////////////////////////////////////////////////////////////////

// CProcessCTRLDlg dialog
class CProcessCTRLDlg : public CDialog
{

// Construction
public:
	CProcessCTRLDlg(CWnd* pParent = NULL);// standard constructor

//My function
	void initmanual();
	void initauto();
	void pid_valve();

	int old_error;
	unsigned short portA;
	unsigned short TankB;
	bool isTank1;
	bool isStart;
	bool isAuto;
	bool justEnd;


//end My function

// Dialog Data
	//{{AFX_DATA(CProcessCTRLDlg)
	enum { IDD = IDD_PROCESSCTRL_DIALOG };
	CEdit m_status2;
	CEdit m_status1;
	CSpinButtonCtrl	m_tank2_spin;
	CEdit m_tank2_edit;
	CSpinButtonCtrl	m_tank1_spin;
	CEdit m_tank1_edit;
	CSpinButtonCtrl	m_p_spin;
	CEdit m_p_edit;
	CSpinButtonCtrl	m_i_spin;
	CEdit m_i_edit;
	CSpinButtonCtrl	m_d_spin;
	CEdit m_d_edit;
	CEdit m_pid_edit;
	CSpinButtonCtrl	m_pid_spin;
	CProgressCtrl	m_tank3;
	CProgressCtrl	m_tank2;
	CProgressCtrl	m_tank1;
	int		m_d_edit_value;
	int		m_i_edit_value;
	int		m_p_edit_value;
	int		m_pid_edit_value;
```

```
	int		m_tank1_edit_value;
	int		m_tank2_edit_value;
	CString		m_str_status1;
	CString		m_str_status2;
//}}AFX_DATA

// ClassWizard generated virtual function overrides
//{{AFX_VIRTUAL(CProcessCTRLDlg)
	protected:
	virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV
support

//}}AFX_VIRTUAL

// Implementation
protected:
	HICON m_hIcon;

	// Generated message map functions
	//{{AFX_MSG(CProcessCTRLDlg)
	virtual BOOL OnInitDialog();
	afx_msg void OnSysCommand(UINT nID, LPARAM lParam);
	afx_msg void OnPaint();
	afx_msg HCURSOR OnQueryDragIcon();
	afx_msg void OnChangemode();
	afx_msg void OnTimer(UINT nIDEvent);
	afx_msg void OnM1on();
	afx_msg void OnM1off();
	afx_msg void OnV0on();
	afx_msg void OnV0off();
	afx_msg void OnV1on();
	afx_msg void OnV1off();
	afx_msg void OnChangePidEdit();
	afx_msg void OnStart();
	afx_msg void OnStop();
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations
immediately before the previous line.

#endif //
!defined(AFX_PROCESSCTRLDLG_H__655F8965_CC8A_11D4_82FF_FF079E21A832_
_INCLUDED_)
```

4.ส่วนของเนื้อหาการกำหนดฟังกชั่นไดอะล๊อกในการใช้งานของโปรแกรม

```
// ProcessCTRLDlg.cpp : implementation file//

#include "stdafx.h"
#include "ProcessCTRL.h"
#include "ProcessCTRLDlg.h"
```

```
#include <conio.h>

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////

// CAboutDlg dialog used for App About

class CAboutDlg : public CDialog
{
public:
	CAboutDlg();

// Dialog Data
	//{{AFX_DATA(CAboutDlg)
	enum { IDD = IDD_ABOUTBOX };
	//}}AFX_DATA

	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CAboutDlg)
	protected:
	virtual void DoDataExchange(CDataExchange* pDX);  // DDX/DDV
support
	//}}AFX_VIRTUAL

// Implementation
protected:

	//{{AFX_MSG(CAboutDlg)
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
	//{{AFX_DATA_INIT(CAboutDlg)
	//}}AFX_DATA_INIT
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CAboutDlg)
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
	//{{AFX_MSG_MAP(CAboutDlg)
		// No message handlers
	//}}AFX_MSG_MAP
```

```
END_MESSAGE_MAP()

*************************************************************************

// CProcessCTRLDlg dialog

CProcessCTRLDlg::CProcessCTRLDlg(CWnd* pParent /*=NULL*/)
	: CDialog(CProcessCTRLDlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(CProcessCTRLDlg)
	m_d_edit_value = 0;
	m_i_edit_value = 0;
	m_p_edit_value = 0;
	m_pid_edit_value = 0;
	m_tank1_edit_value = 0;
	m_tank2_edit_value = 0;
	m_str_status1 = _T("");
	m_str_status2 = _T("");
	//}}AFX_DATA_INIT
	m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CProcessCTRLDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CProcessCTRLDlg)
	DDX_Control(pDX, IDC_STATUS2, m_status2);
	DDX_Control(pDX, IDC_STATUS1, m_status1);
	DDX_Control(pDX, IDC_TANK2_SPIN, m_tank2_spin);
	DDX_Control(pDX, IDC_TANK2_EDIT, m_tank2_edit);
	DDX_Control(pDX, IDC_TANK1_SPIN, m_tank1_spin);
	DDX_Control(pDX, IDC_TANK1_EDIT, m_tank1_edit);
	DDX_Control(pDX, IDC_P_SPIN, m_p_spin);
	DDX_Control(pDX, IDC_P_EDIT, m_p_edit);
	DDX_Control(pDX, IDC_I_SPIN, m_i_spin);
	DDX_Control(pDX, IDC_I_EDIT, m_i_edit);
	DDX_Control(pDX, IDC_D_SPIN, m_d_spin);
	DDX_Control(pDX, IDC_D_EDIT, m_d_edit);
	DDX_Control(pDX, IDC_PID_EDIT, m_pid_edit);
	DDX_Control(pDX, IDC_PID_SPIN, m_pid_spin);
	DDX_Control(pDX, IDC_TANK3, m_tank3);
	DDX_Control(pDX, IDC_TANK2, m_tank2);
	DDX_Control(pDX, IDC_TANK1, m_tank1);
	DDX_Text(pDX, IDC_D_EDIT, m_d_edit_value);
	DDX_Text(pDX, IDC_I_EDIT, m_i_edit_value);
	DDX_Text(pDX, IDC_P_EDIT, m_p_edit_value);
	DDX_Text(pDX, IDC_PID_EDIT, m_pid_edit_value);
	DDX_Text(pDX, IDC_TANK1_EDIT, m_tank1_edit_value);
	DDX_Text(pDX, IDC_TANK2_EDIT, m_tank2_edit_value);
	DDX_Text(pDX, IDC_STATUS1, m_str_status1);
	DDX_Text(pDX, IDC_STATUS2, m_str_status2);
	//}}AFX_DATA_MAP
}
```

```
BEGIN_MESSAGE_MAP(CProcessCTRLDlg, CDialog)
	//{{AFX_MSG_MAP(CProcessCTRLDlg)
	ON_WM_SYSCOMMAND()
	ON_WM_PAINT()
	ON_WM_QUERYDRAGICON()
	ON_BN_CLICKED(IDC_CHANGEMODE, OnChangemode)
	ON_WM_TIMER()
	ON_BN_CLICKED(IDC_M1ON, OnM1on)
	ON_BN_CLICKED(IDC_M1OFF, OnM1off)
	ON_BN_CLICKED(IDC_V0ON, OnV0on)
	ON_BN_CLICKED(IDC_V0OFF, OnV0off)
	ON_BN_CLICKED(IDC_V1ON, OnV1on)
	ON_BN_CLICKED(IDC_V1OFF, OnV1off)
	ON_EN_CHANGE(IDC_PID_EDIT, OnChangePidEdit)
	ON_BN_CLICKED(IDC_START, OnStart)
	ON_BN_CLICKED(IDC_STOP, OnStop)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////

// CProcessCTRLDlg message handlers

BOOL CProcessCTRLDlg::OnInitDialog()
{
	CDialog::OnInitDialog();

	// Add "About..." menu item to system menu.

	// IDM_ABOUTBOX must be in the system command range.
	ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
	ASSERT(IDM_ABOUTBOX < 0xF000);

	CMenu* pSysMenu = GetSystemMenu(FALSE);
	if (pSysMenu != NULL)
	{
		CString strAboutMenu;
		strAboutMenu.LoadString(IDS_ABOUTBOX);
		if (!strAboutMenu.IsEmpty())
		{
			pSysMenu->AppendMenu(MF_SEPARATOR);
			pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX,
strAboutMenu);
		}
	}

	SetIcon(m_hIcon, TRUE);			// Set big icon
	SetIcon(m_hIcon, FALSE);		// Set small icon

	// TODO: Add extra initialization here

	SetTimer(ID_CLOCK_TIMER,50,NULL);

	_outp(0x303,0x88);
```

```
	portA = 0xFF;
	_outp(0x300,portA);

	old_error=0;

	isTank1=true;
	_outp(0x301,0x02);//0000 0010

	isStart=false;

	m_tank1.SetRange(0,100); m_tank1.SetStep(1); m_tank1.SetPos
(0);
	m_tank2.SetRange(0,100); m_tank2.SetStep(1); m_tank2.SetPos
(0);
	m_tank3.SetRange(0,100); m_tank3.SetStep(1); m_tank3.SetPos
(75);

	initmanual();



	return TRUE;  // return TRUE  unless you set the focus to a
control
}

void CProcessCTRLDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
	if ((nID & 0xFFF0) == IDM_ABOUTBOX)
	{
		CAboutDlg dlgAbout;
		dlgAbout.DoModal();
	}
	else
	{
		CDialog::OnSysCommand(nID, lParam);
	}
}

// If you add a minimize button to your dialog, you will need the
code below
//  to draw the icon.  For MFC applications using the document/view
model,
//  this is automatically done for you by the framework.

void CProcessCTRLDlg::OnPaint()
{
	if (IsIconic())
	{
		CPaintDC dc(this); // device context for painting

		SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(),
0);
```

```
            // Center icon in client rectangle
            int cxIcon = GetSystemMetrics(SM_CXICON);
            int cyIcon = GetSystemMetrics(SM_CYICON);
            CRect rect;
            GetClientRect(&rect);
            int x = (rect.Width() - cxIcon + 1) / 2;
            int y = (rect.Height() - cyIcon + 1) / 2;

            // Draw the icon
            dc.DrawIcon(x, y, m_hIcon);
      }
      else
      {
            CDialog::OnPaint();
      }
}

HCURSOR CProcessCTRLDlg::OnQueryDragIcon()
{
      return (HCURSOR) m_hIcon;
}

void CProcessCTRLDlg::initmanual()
{
      isAuto=false;
      isStart=false;
      justEnd=false;

      portA = 0xFF;
      _outp(0x300,portA);

      this->SetWindowText("ProcessCTRL : manual mode");


      GetDlgItem(IDC_TANK1_EDIT)->EnableWindow(0); GetDlgItem
(IDC_TANK1_SPIN)->EnableWindow(0);
      GetDlgItem(IDC_TANK2_EDIT)->EnableWindow(0); GetDlgItem
(IDC_TANK2_SPIN)->EnableWindow(0);
      GetDlgItem(IDC_START)->EnableWindow(0); GetDlgItem(IDC_STOP)->
EnableWindow(0);

      GetDlgItem(IDC_PID_SPIN)->EnableWindow(1);

      m_pid_spin.SetRange(0,100); m_pid_spin.SetBuddy(&m_pid_edit);
m_pid_spin.SetPos(0);

      GetDlgItem(IDC_P_EDIT)->EnableWindow(0); GetDlgItem
(IDC_P_SPIN)->EnableWindow(0);
      GetDlgItem(IDC_I_EDIT)->EnableWindow(0); GetDlgItem
(IDC_I_SPIN)->EnableWindow(0);
      GetDlgItem(IDC_D_EDIT)->EnableWindow(0); GetDlgItem
(IDC_D_SPIN)->EnableWindow(0);
```

```
	CButton *m1on = (CButton*)GetDlgItem(IDC_M1ON);m1on->SetCheck
(0);
	CButton *v0on = (CButton*)GetDlgItem(IDC_V0ON);v0on->SetCheck
(0);
	CButton *v1on = (CButton*)GetDlgItem(IDC_V1ON);v1on->SetCheck
(0);

	CButton *m1off = (CButton*)GetDlgItem(IDC_M1OFF);m1off->
SetCheck(1);
	CButton *v0off = (CButton*)GetDlgItem(IDC_V0OFF);v0off->
SetCheck(1);
	CButton *v1off = (CButton*)GetDlgItem(IDC_V1OFF);v1off->
SetCheck(1);

	m_pid_edit.SetReadOnly(false);

}

void CProcessCTRLDlg::initauto(){

	isAuto=true;
	isStart=false;
	justEnd=false;

	portA = 0xFF;
	_outp(0x300,portA);

	this->SetWindowText("ProcessCTRL : automatic mode");

	GetDlgItem(IDC_TANK1_EDIT)->EnableWindow(1); GetDlgItem
(IDC_TANK1_SPIN)->EnableWindow(1);
	GetDlgItem(IDC_TANK2_EDIT)->EnableWindow(1); GetDlgItem
(IDC_TANK2_SPIN)->EnableWindow(1);
	GetDlgItem(IDC_START)->EnableWindow(1); GetDlgItem(IDC_STOP)->
EnableWindow(1);

	GetDlgItem(IDC_PID_SPIN)->EnableWindow(0);

	GetDlgItem(IDC_P_EDIT)->EnableWindow(1); GetDlgItem
(IDC_P_SPIN)->EnableWindow(1);
	GetDlgItem(IDC_I_EDIT)->EnableWindow(1); GetDlgItem
(IDC_I_SPIN)->EnableWindow(1);
	GetDlgItem(IDC_D_EDIT)->EnableWindow(1); GetDlgItem
(IDC_D_SPIN)->EnableWindow(1);
	m_tank1_spin.SetRange(0,100); m_tank1_spin.SetBuddy
(&m_tank1_edit); m_tank1_spin.SetPos(0);
	m_tank2_spin.SetRange(0,100); m_tank2_spin.SetBuddy
(&m_tank2_edit); m_tank2_spin.SetPos(0);

	m_p_spin.SetRange(0,100); m_p_spin.SetBuddy(&m_p_edit);
m_p_spin.SetPos(0);
	m_i_spin.SetRange(0,100); m_i_spin.SetBuddy(&m_i_edit);
m_i_spin.SetPos(0);
```

```
      m_d_spin.SetRange(0,100); m_d_spin.SetBuddy(&m_d_edit);
m_d_spin.SetPos(0);

CButton *m1on = (CButton*)GetDlgItem(IDC_M1ON);m1on->SetCheck(0);
CButton *v0on = (CButton*)GetDlgItem(IDC_V0ON);v0on->SetCheck(0);
CButton *v1on = (CButton*)GetDlgItem(IDC_V1ON);v1on->SetCheck(0);
CButton *m1off =(CButton*)GetDlgItem(IDC_M1OFF);m1off->SetCheck(1);
CButton *v0off =(CButton*)GetDlgItem(IDC_V0OFF);v0off->SetCheck(1);
CButton *v1off = (CButton*)GetDlgItem(IDC_V1OFF);v1off->SetCheck(1);

      m_pid_edit.SetReadOnly(true);
}
void CProcessCTRLDlg::OnChangemode()
{

      int result_mode;

      if(isAuto){
result_mode = MessageBox("Are you sure to switch to manual mode ?",

                  "switch mode",MB_YESNO);
      }else{
result_mode = MessageBox("Are you sure to switch to automatic mode
?",

                  "switch mode",MB_YESNO);
      }

      switch(result_mode){
      case IDYES:
            if(isAuto){
            this->SetWindowText("ProcessCTRL : manual mode");
      initmanual();
            }else{
      this->SetWindowText("ProcessCTRL : automatic mode");
      initauto();
            }
            break;
      case IDNO:;
      }
}

void CProcessCTRLDlg::OnTimer(UINT nIDEvent)
{
// TODO: Add your message handler code here and/or call default
      UpdateData(true);

      CButton *m1on, *m1off, *v1on, *v1off, *v0on, *v0off;

      m1on = (CButton*)GetDlgItem(IDC_M1ON);
      m1off = (CButton*)GetDlgItem(IDC_M1OFF);

      v1on = (CButton*)GetDlgItem(IDC_V1ON);
      v1off = (CButton*)GetDlgItem(IDC_V1OFF);
```

```
	v0on = (CButton*)GetDlgItem(IDC_V0ON);
	v0off = (CButton*)GetDlgItem(IDC_V0OFF);


	unsigned short portB;
	if(isTank1){
		portB = _inp(0x30A);
		m_tank1.SetPos(portB*100/255);
		m_str_status1.Format("%d",portB*100/255);
		m_status1.SetWindowText(m_str_status1);

		//auto mode//

		if(isAuto){

		if( (m_tank1_edit_value) < (portB*100/255) ){
//m1 off, v1 on
		m1on->SetCheck(0); m1off->SetCheck(1); this->OnM1off();
		v1on->SetCheck(1); v1off->SetCheck(0); this->OnV1on();
		}else if( (m_tank1_edit_value) > (portB*100/255) ){
//m1 on, v1 off
		m1on->SetCheck(1); m1off->SetCheck(0); this->OnM1on();
		v1on->SetCheck(0); v1off->SetCheck(1); this->OnV1off();
			}
		}

//end//
		_outp(0x301,0x04);//0000 0100
		isTank1 = false;
	}else{
		portB = _inp(0x30A);
		TankB = portB;
		m_tank2.SetPos(portB*100/255);
		m_str_status2.Format("%d",portB*100/255);
		m_status2.SetWindowText(m_str_status2);

	//auto mode//

		if(isAuto){

		if(isStart){

		if( (m_tank2_edit_value) < (portB*100/255) ){
// v0 on
	v0on->SetCheck(1); v0off->SetCheck(0); this->OnV0on();
	}else if( (m_tank2_edit_value) > (portB*100/255) ){
// v0 off
	v0on->SetCheck(0); v0off->SetCheck(1); this->OnV0off();
	}else{
			isStart=false;
			justEnd=true;
			}
		}
```

```
                    if(justEnd){
// v0 off
      v0on->SetCheck(0); v0off->SetCheck(1); this->OnV0off();
      m_p_spin.SetPos(0); m_i_spin.SetPos(0); m_d_spin.SetPos(0);
                    pid_valve();
                    justEnd=false;
                    }

              }

//end//
              _outp(0x301,0x02);//0000 00010
              isTank1 = true;
      }


      CDialog::OnTimer(nIDEvent);
}

void CProcessCTRLDlg::OnM1on()
{

// TODO: Add your control notification handler code here

      portA = portA & 0xFE;//1111 1110
      _outp(0x300,portA);
}

void CProcessCTRLDlg::OnM1off()
{

// TODO: Add your control notification handler code here

      portA = portA | 0x01;//0000 0001
      _outp(0x300,portA);

}

void CProcessCTRLDlg::OnV0on()
{

// TODO: Add your control notification handler code here

      portA = portA & 0xFB;//1111 1011
      _outp(0x300,portA);

}

void CProcessCTRLDlg::OnV0off()
{

// TODO: Add your control notification handler code here
```

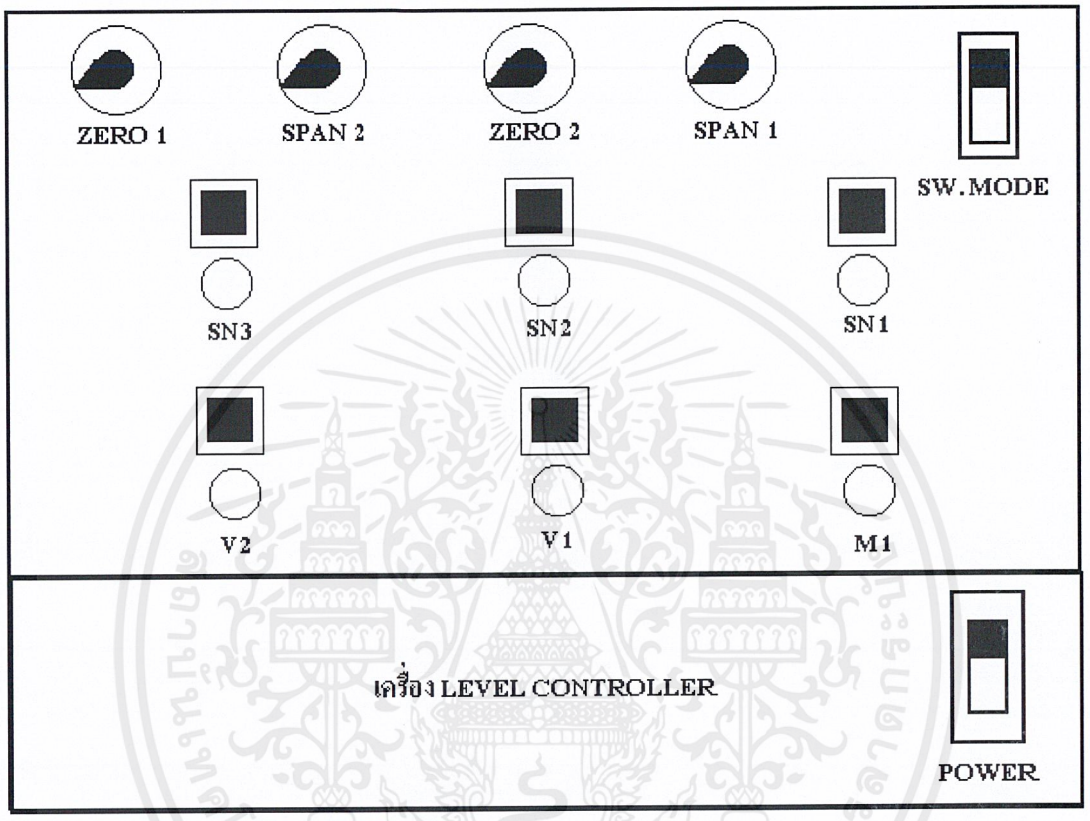
```
	portA = portA | 0x04;//0000 0100
	_outp(0x300,portA);

}

void CProcessCTRLDlg::OnV1on()
{

// TODO: Add your control notification handler code here

	portA = portA & 0xFD;//1111 1101
	_outp(0x300,portA);

}

void CProcessCTRLDlg::OnV1off()
{
	// TODO: Add your control notification handler code here

	portA = portA | 0x02;//0000 0010
	_outp(0x300,portA);

}

void CProcessCTRLDlg::OnChangePidEdit()
{
	// TODO: If this is a RICHEDIT control, the control will not
	// send this notification unless you override the
CDialog::OnInitDialog()
	// function and call CRichEditCtrl().SetEventMask()
	// with the ENM_CHANGE flag ORed into the mask.

	// TODO: Add your control notification handler code here
	UpdateData(true);

	if(m_pid_edit_value==0) portA = portA | 0x38;//0011 1000
	else if(m_pid_edit_value<=33){
		portA = portA & 0xF7;//1111 0111
		portA = portA | 0x30;//0011 0000
	}else if(m_pid_edit_value<=66){
		portA = portA & 0xE7;//1110 0111
		portA = portA | 0x20;//0010 0000
	}else portA = portA & 0xC7;//1100 0111

	_outp(0x300,portA);
}

void CProcessCTRLDlg::pid_valve(){

	int A, B, C, P, I, D, m;
	float M;

	A = m_tank2_edit_value;
	B = (int)TankB*100/255;
```

```
	C = old_error;
	P = m_p_edit_value;
	I = m_i_edit_value;
	D = m_d_edit_value;

	M = float(P*(A-B)+I*((A-B)+C)/2+D*((A-B)-C));
	if(M>100.0) M=100.0;
	m = int(M*255/100);

	_outp(0x308,m);

	m_pid_spin.SetPos(m);

	old_error = A-B;

}


void CProcessCTRLDlg::OnStart()
{
	// TODO: Add your control notification handler code here

	pid_valve();
	isStart=true;
	justEnd=false;


}

void CProcessCTRLDlg::OnStop()
{
	// TODO: Add your control notification handler code here

	isStart = false;
	justEnd = true;

}
```

รูปที่ ค.1 โปรแกรมการทำงานชุดควบคุมระดับน้ำ

ภาคผนวก ง
คู่มือการใช้งาน

การใช้งานเครื่อง LEVEL CONTROLLER



รูป ง.1 เครื่อง LEVEL CONTROLLER

หน้าที่ต่างๆของคีย์บอร์ดต่างๆ

คีย์ SN 1	เป็นสวิตซ์ปุ่มกดติดกดดับใช้ควบคุมการทำงานของโซลินอยด์วาล์ว SN 1
คีย์ SN 2	เป็นสวิตซ์ปุ่มกดติดกดดับใช้ควบคุมการทำงานของโซลินอยด์วาล์ว SN 2
คีย์ SN 3	เป็นสวิตซ์ปุ่มกดติดกดดับใช้ควบคุมการทำงานของโซลินอยด์วาล์ว SN 3
คีย์ M 1	เป็นสวิตซ์ปุ่มกดติดกดดับใช้ควบคุมการทำงานของปั้มน้ำ
คีย์ V 1	เป็นสวิตซ์ปุ่มกดติดกดดับใช้ควบคุมการทำงานของ โซลินอยด์วาล์ว V 1
คีย์ V 2	เป็นสวิตซ์ปุ่มกดติดกดดับใช้ควบคุมการทำงานของ โซลินอยด์วาล์ว V 2
คีย์ SW.MODE	เป็นสวิตซ์เลือกใช้งานเครื่อง LEVEL CONROLLER
คีย์ ZERO 1	ทำหน้าที่ปรับค่า ZERO ของ TANK 1
คีย์ SPAN 1	ทำหน้าที่ปรับค่า SPAN ของ TANK 1
คีย์ ZERO 2	ทำหน้าที่ปรับค่า ZERO ของ TANK 2
คีย์ SPAN 2	ทำหน้าที่ปรับค่า SPAN ของ TANK 2
คีย์ POWER	เป็นสวิตซ์เปิด - ปิดเครื่อง LEVEL CONTROLLER
หลอดสัญญาณ SN 1	ทำหน้าที่แสดงการทำงานของ โซลินอยด์วาล์ว SN 1
หลอดสัญญาณ SN2	ทำหน้าที่แสดงการทำงานของ โซลินอยด์วาล์ว SN 2
หลอดสัญญาณ SN3	ทำหน้าที่แสดงการทำงานของ โซลินอยด์วาล์ว SN 3
หลอดสัญญาณ M1	ทำหน้าที่แสดงการทำงานของปั๊มน้ำ
หลอดสัญญาณ V1	ทำหน้าที่แสดงการทำงานของ โซลินอยด์วาล์ว V1
หลอดสัญญาณ V2	ทำหน้าที่แสดงการทำงานของ โซลินอยด์วาล์ว V2

บรรณานุกรม

เกษตร์ ศิริสันฤทธิ์. **หลักการของเครื่องมือวัดทางอุตสาหกรรม.**เล่มที่ 2. พิมพ์ครั้งที่ 2.กรุงเทพฯ: สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง. 2541

บริษัท ศิลาเสิร์ช จำกัด.**Data Book 2**.กรุงเทพฯ

บริษัท อีทีที จำกัด. **ET-PC DIO CARD.** กรุงเทพฯ

พยัพ พลแก้ว. **Differential Pressure Transmitter** . วิทยาลัยเทคนิคปทุมธานี : แผนกช่างเครื่องมือวัดและควบคุมในงานอุตสาหกรรม.

นิรุธ อำนวยศิลป์ . **การเขียนโปรแกรมด้วย Visual C++ .**พิมพ์ครั้งที่ 2 :บริษัท ซักเซสมีเดีย จำกัด. กรุงเทพ . 2521

ประวัติผู้แต่ง

ชื่อผู้ทำปริญญานิพนธ์	นายธงฉาน คำศิริ
วันเดือนปีเกิด	วันที่ 16 ธันวาคม พ.ศ.2521
สถานที่เกิด	โรงพยาบาลจังหวัดระยอง
ภูมิลำเนาเดิม	จังหวัดระยอง
ที่อยู่ปัจจุบัน	018/15 ถนนสนามเป้า ตำบลท่าประดู่ อำเภอเมือง จังหวัดระยอง 2100
โทรศัพท์	-

ประวัติการศึกษา

ประถมศึกษา	โรงเรียนวัดโขดทิมทราราม
มัธยมศึกษาตอนต้น	โรงเรียนระยองวิทยาคม
ประกาศนียบัตรวิชาชีพ (ปวช.)	วิทยาลัยเทคนิคระยอง
ประกาศนียบัตรวิชาชีพชั้นสูง (ปวส.)	วิทยาลัยเทคนิคระยอง
ปริญญาตรี	สาขาเทคโนโลยีการวัดคุมทางอุตสาหกรรม ภาควิชาครุศาสตร์วิศวกรรม คณะครุศาสตร์อุตสาหกรรม สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหาร ลาดกระบัง
คติพจน์	ประสบการณ์สอนให้รู้จักชีวิต

ประวัติผู้แต่ง

ชื่อผู้ทำปริญญานิพนธ์	นายสิทธิชัย บุญอนันต์
วันเดือนปีเกิด	วันที่ 10 มกราคม พ.ศ.2522
สถานที่เกิด	โรงพยาบาลจังหวัดระยอง
ภูมิลำเนาเดิม	จังหวัดระยอง
ที่อยู่ปัจจุบัน	31 ซอยเพลินตา ถนนสุขุมวิท ตำบลเชิงเนิน อำเภอเมือง จังหวัดระยอง 2100
โทรศัพท์	-

ประวัติการศึกษา

ประถมศึกษา	โรงเรียนอนุบาลระยอง
มัธยมศึกษาตอนต้น	โรงเรียนระยองวิทยาลัย
ประกาศนียบัตรวิชาชีพ (ปวช.)	วิทยาลัยเทคนิคระยอง
ประกาศนียบัตรวิชาชีพชั้นสูง (ปวส.)	วิทยาลัยเทคนิคระยอง
ปริญญาตรี	สาขาเทคโนโลยีการวัดคุมทางอุตสาหกรรม ภาควิชาครุศาสตร์วิศวกรรม คณะครุศาสตร์อุตสาหกรรม สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหาร ลาดกระบัง
คติพจน์	เรียนเพื่อรู้ ดูเพื่อเห็น ทำเพื่อเป็น

ประวัติผู้แต่ง

ชื่อผู้ทำปริญญานิพนธ์	นางสาวสุพรรณษา อินทรปัญญา
วันเดือนปีเกิด	วันที่ 5 ธันวาคม พ.ศ.2522
สถานที่เกิด	โรงพยาบาลอำเภองาว
ภูมิลำเนาเดิม	จังหวัดลำปาง
ที่อยู่ปัจจุบัน	100 ม.9 บ้านปันพัฒนา ตำบลปงเตา
	อำเภองาว จังหวัดลำปาง 52110
โทรศัพท์	054 -365540

ประวัติการศึกษา

ประถมศึกษา	โรงเรียนดอนไชยวิทยา
มัธยมศึกษาตอนต้น	โรงเรียนประชารัฐธรรมคุณ
ประกาศนียบัตรวิชาชีพ (ปวช.)	วิทยาลัยเทคนิคลำปาง
ประกาศนียบัตรวิชาชีพชั้นสูง (ปวส.)	วิทยาลัยเทคนิคลำปาง
ปริญญาตรี	สาขาเทคโนโลยีการวัดคุมทางอุตสาหกรรม
	ภาควิชาครุศาสตร์วิศวกรรม
	คณะครุศาสตร์อุตสาหกรรม
	สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหาร
	ลาดกระบัง
คติพจน์	คนดีดีไม่จริง คนเลวเลวไม่หมด