ระบบควบคุมอุณหภูมิแบบหลายจุด

MULTIPOINT TEMPERATURE CONTROLLER

โดย

นาย เฉลิมพงศ์ ทองคำ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

ภาควิชาวิศวกรรมสารสนเทศ

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2550

MULTIPOINT TEMPERATURE CONTROLLER

BY

Mr.CHALERMPONG THONGKAM

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENT FOR THE DEGREE OF

BACHELOR IN DEPARTMENT OF INFORMATION ENGINEERING

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2007

หัวข้อปริญญานิพนธ์ ระบบควบคุมอุณหภูมิแบบหลายจุด

ชื่อนักศึกษา นายเฉลิมพงศ์ ทองคำ รหัสประจำตัว 46015663

อาจารย์ที่ปรึกษา ผศ. ดลชัย สุขเจริญผล

ระดับการศึกษา ปริญญาตรี วิศวกรรมศาสตร์บัณฑิต

สาขาวิศวกรรมสารสนเทศ

ภาควิชา วิศวกรรมสารสนเทศ

ปีการศึกษา 2550

ปริญญานิพนธ์ฉบับนี้ได้รับความเห็นชอบจากอาจารย์ที่ปรึกษาเป็นที่เรียบร้อยแล้ว

.....

ผศ. ดลชัย สุขเจริญผล

อาจารย์ผู้ควบคุม

หัวข้อปริญญานิพนธ์ ระบบควบคุมอุณหภูมิแบบหลายจุด

ชื่อนักศึกษา นายเฉลิมพงศ์ ทองคำ รหัสประจำตัว 46015663

อาจารย์ที่ปรึกษา ผศ. ดลชัย สุขเจริญผล

ระดับการศึกษา ปริญญาตรี วิศวกรรมศาสตร์บัณฑิต

สาขาวิศวกรรมสารสนเทศ

ภาควิชา วิศวกรรมสารสนเทศ

ปีการศึกษา 2550

บทคัดย่อ

ปริญญานิพนธ์นี้กล่าวถึงการออกแบบ และการสร้างระบบควบคุมอุณหภูมิแบบหลายจุด ให้มีความถูกต้องและแม่นยำเพิ่มมากยิ่งขึ้น โดยการนำระบบควบคุมแบบ PID แบบปิด มาใช้ร่วมกับ ไมโครคอนโทรลเลอร์ F^2MC-8L เพื่อเพิ่มประสิทธิภาพของระบบ เช่น ความแน่นอน ความเสถียรภาพ และ ความไวของระบบ ในรูปแบบของระบบจะมีเซนเซอร์อุณหภูมิกระจายอยู่รอบๆ พื้นที่ของอุปกรณ์อิเล็กทรอนิกส์ โดยที่ไมโครคอนโทรลเลอร์จะนำค่าของอุณหภูมิที่วัดได้แต่ละจุดมาคำนวณหาค่าเฉลี่ย เพื่อนำมาใช้ในการควบคุม ฮีทเตอร์ ระบบทำความเย็นต่างๆ

Thesis Title	MULTIPOINT TEMPERATURE CONTROLLER
Student	Mr. Chalermpong Thongkam ID. 46015663
Advisor	Asst. Prof. Dolchai Sookcharoenphol
Graduate Level	Bachelor Degree of Information Engineering
Department	Information Engineering
Academic Year	2007

ABSTRACT

This project is design and implemcnt of Multipoint Temperature Controller for increase an accuracy of the system. A PID closc loop control is selected to dcvelopment an microcontroller F^2MC-8L for increasc an efficiency of thc system. Such is stability. In application, multi-point sensors is distribution around the heat arcas of the electronic equipment microcontroller is calculate average temperatures. Avcrage temperature is calculated by a microcontroller and use the value to control a cooling system.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงด้วยดีเป็นผลเนื่องมาจากการได้รับความเมตตากรุณาจาก ผศ. ดลชัย สุขเจริญผล ผู้ซึ่งช่วยเป็นอาจารย์ที่ปรึกษาและช่วยในการให้ความรู้ให้คำแนะนำรวมทั้งยังช่วยตรวจทานแก้ไขในการทำปริญญานิพนธ์ฉบับนี้ด้วยดีเสมอมาผู้จัดทำขอกราบขอบพระคุณอาจารย์เป็นอย่างสูงด้วยความจริงใจ และขอขอบคุณพี่ๆ เพื่อนๆ ในภาควิชาวิศวกรรมสารสนเทศทุกคนที่คอยให้คำดิชม รวมทั้งภาควิชาวิศวกรรมสารสนเทศที่คอยเอื้อเฟื้อเครื่องมืออุปกรณ์ต่างในการทำปริญญานิพนธ์ฉบับนี้

ท้ายที่สุดนี้คณะผู้จัดทำขอกราบขอบพระคุณ บิดา มารดา ผู้ให้การสนับสนุนทางการศึกษาและเป็นกำลังใจมาโดยตลอด

นาย เฉลิมพงศ์ ทองคำ

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญรูป

สารบัญรูป (ต่อ)

ระบบควบคุมอุณหภูมิแบบหลายจุด

MULTIPOINT TEMPERATURE CONTROLLER

โดย

นาย เฉลิมพงศ์ ทองคำ

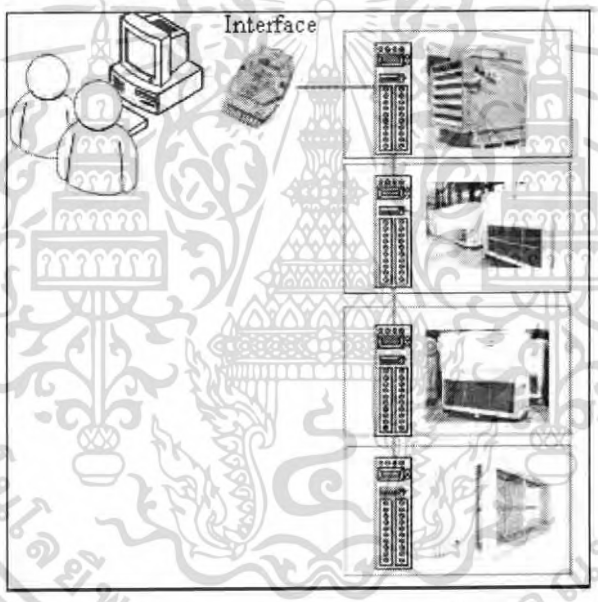
สารบัญตาราง

ตารางที่ **หน้า**

บทที่ 1
บทนำ

1.1 แนวคิดและที่มาของปัญหา

เนื่องด้วยระบบที่ต้องการควบคุมอุณหภูมิในปัจจุบันที่ใช้กันอยู่เช่น การปรับอากาศในอาคาร การอบผลผลิตทางการเกษตร การควบคุมอุณหภูมิในอุตสาหกรรมต่างๆ ที่ต้องการการควบคุมอุณหภูมิในบริเวณกว้าง และต้องใช้อุปกรณ์วัดและจ่ายอุณหภูมิในหลายจุด ยังไม่มีการควบคุมการจ่ายอุณหภูมิที่ดีพอทำให้เกิดผลเสียเช่น อุณหภูมิในห้องที่ต้องการควบคุมอุณหภูมิเย็นหรือร้อนเกินไปในบางช่วง หรือสูญเสียพลังงานในการจ่ายอุณหภูมิมากเกินไปโดยไม่จำเป็น



รูปที่1.1 ลักษณะการนำไปใช้งาน

1.2 จุดประสงค์

1.2.1 เพื่อพัฒนาระบบการควบคุมอุณหภูมิ

1.2.2 เพื่อประหยัดพลังงานที่ต้องใช้ในการควบคุมอุณหภูมิ

1.2.3 ลดความยุ่งยากในการควบคุมอุณหภูมิที่ใช้ระบบ Manual

1.3 ขอบเขตของโครงงาน

1. 3.1 ระบบจะทำการวัดอุณหภูมิในจุดต่างๆเพื่อนำมาประมวลผลในการควบคุมตัวจ่ายอุณหภูมิตามจุดต่างๆ

1.3.2 ใช้ PID-control system ในการควบคุม

1.3.3 ใช้ ไมโครคอนโทรลเลอร์ที่มีประสิทธิสูงในการทำงานเพื่อลดข้อผิดพลาดของไมโครคอนโทรลเลอร์ในการ ทำงานในพื้นที่ ที่มีอุณหภูมิสูง

1.3.4 สามารถปรับแต่งและแสดงผลการทำงานผ่านคอมพิวเตอร์ได้(Virtual instruments)

1.4 ผลที่คาดว่าจะได้รับ

1.4.1 ระบบที่พัฒนาขึ้น จะสามารถนำไปใช้ในอุตสาหกรรม และในอาคารที่ต้องการๆ ควบคุมอุณหภูมิแบบอัตโนมัติได้

1.4.2 สามารถประหยัดพลังงานที่ต้องใช้ในการควบคุมอุณหภูมิได้

1.5 วิธีการดำเนินงาน

1.5.1 โครงงานนี้เริ่มต้นด้วยการศึกษาทฤษฎี ต่างๆเกี่ยวกับงาน ซึ่งมีเรื่องหลักคือ *PID* และ Modbus Protocol ที่มีรายละเอียดในบทที่ 2 เรื่องการออกแบบงานต่างๆ ในบทที่ 3

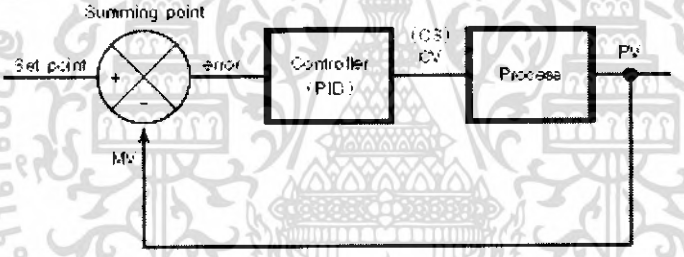
บทที่ 2

ทฤษฎีที่เกี่ยวข้อง

2.1 PID Control

2.1.1 การควบคุมแบบ PID

จากการที่ระบบควบคุม จะต้องแม่นยำแน่นอนและต้องประหยัดจึงทำให้มี การคิดระบบควบคุมต่างๆ เพื่อใช้ในงานควบคุมให้ได้ผลตามที่ต้องการ การควบคุมแบบPID เป็นการควบคุมแบบหนึ่งที่ที่จะต้องตอบสนองต่อความต้องการ และระบบการควบคุมดังกล่าว ใน Loop ของการควบคุมจะประกอบด้วยส่วนต่างๆดังรูป



รูปที่ 2.1 แสดง Loop การควบคุมที่สมบูรณ์แบบ

คือจุดรวม (Summing point) วงจรควบคุม(Controller) และตัวดำเนินการ (Process) การทำงานของ Loop การควบคุมจะเริ่มจากการป้อนค่าเป้าหมาย (Set point) ที่ต้องการเข้ามายัง Summing point ทางด้านบวก ส่วนทางด้านลบของ Summing point จะรับค่าจากเอาต์พุต Process หรือค่า MV สัญญาณทั้ง 2 ที่เข้ามายัง Summing point จะถูกหักล้างออกมาเป็นค่า Errorทางด้านเอาต์พุตเพื่อส่งต่อให้กับอินพุตของ Controller ให้ทำการควบคุมค่าแรงดันที่ออกทางเอาท์พุตให้กับอินพุตของ Process ส่วนทางด้านเอาท์พุตของ Process ก็จะเปลี่ยนแปลงค่าไปเรื่อยๆเพื่อนำไปหักล้างกับค่า Set point

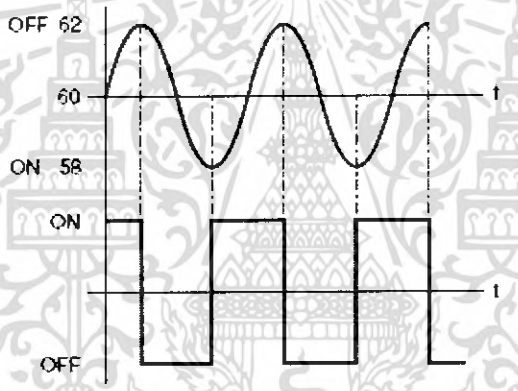
การทำงานจะวนลูปแบบนี้ไปเรื่อยๆจนกระทั่งค่า Set point และค่า MV เท่ากันเมื่อนำมาหักล้างกัน Error ก็จะเหลือศูนย์การทำงานของลูปควบคุมก็จะหยุดทันที นั่นหมายความว่า Process ทำงานได้

ตรงตามค่า Set point ที่ต้องการพอดี หลังจากนี้ถ้ามีการเปลี่ยนแปลงค่า Set point เกิดขึ้นการทำงานของลูปควบคุมก็จะเข้าหาจุด Set point ไปเรื่อยๆ

2.1.2 ระบบการควบคุมแบบต่างๆ

โดยทั่วไปในระบบควบคุมการทำงานของ Controller จากในรูปที่ 1 ซึ่งเป็นส่วนสำคัญที่ต้องนำมาใช้ควบคุมการทำงานทั้งหมดของระบบ โดยได้แบ่งชนิดของระบบ ออกเป็น 4 ชนิด ได้แก่

- ON OFF control
- Proportional control (P-Control)
- Intergral หรือ Reset control (I-Control)
- Derivative control (D-Control)



รูปที่ 2.2 การทำงานของ ON-OFF Control

2.1.3 การควบคุม ON-OFF Control

เป็นลักษณะการควบคุมแบบวิธีการปิด-เปิดธรรมดาเหมือนการปิดเปิดสวิตช์ไฟฟ้าตัวอย่างเช่น เทอร์โมสตัสในเตารีดหรือการปิด-เปิด วาล์ว จากรูปที่ 2 จะสังเกตเห็นว่าการทำงาน ON-OFF Control นั้นเอาท์พุตจะไม่คงที่จะมีการเปลี่ยนแปลงตลอดทำให้ไม่สามารถจะใช้ในการควบคุมบางอย่างได้ ปกติการควบคุมแบบ ON-OFF จะมี Dead band ซึ่งหมายความว่าเอาท์พุตของ Controller ที่ออกมาจะมีการแกว่งอยู่ในย่านที่ Controller ยอมได้ เช่น จากรูปจะเห็นว่าอุณหภูมิจะเปลี่ยนแปลงอยู่ระหว่าง 58 องศาเซลเซียส ถึง $62C°$ ถ้ากำหนด Set point ที่ค่า $60°C$ การทำงานของ ON-OFF Control จะมี Dead band เท่ากับ 62-58 = $4°C$ หรือจะพูดได้ว่า Controller เกิดการแกว่งระหว่าง $62C°$ ถึง $58°C$

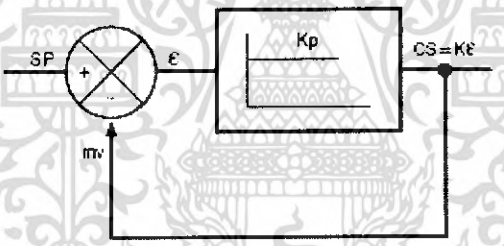
ซึ่งการแกว่งนี้ จะขึ้นกับค่า Dead time ของ Process เอง สรุปว่าการควบคุมแบบ ON-Off จะใช้ได้ดีกับ Process ที่มี Dead time สั้นแต่มีค่าคงตัวของเวลายาว

2.1.4 การควบคุมแบบ Proportional (P)

การควบคุมแบบ P เป็นการควบคุมโดยการกำหนดค่า Gain ของ Controller หรือเรียกได้ว่าเป็นการกำหนดอัตราส่วนของเอ้าท์พุตต่ออินพุต หรืออาจจะกำหนดออกมาอยู่ในรูปของ Proportional Gain หรือ Proportional Band ก็ได้ โดยมีสมการหาค่า PB ดังนี้

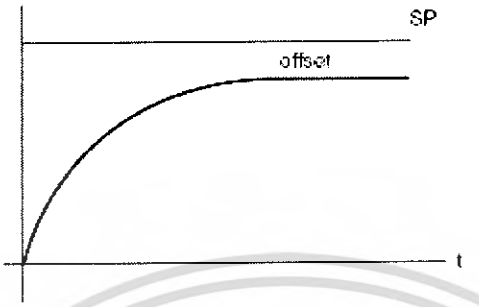
$$\text{Proportional Band} = \frac{1 \times 100}{\text{Proportional Gain}}$$

$$\text{หรือ } PB = \frac{1 \times 100}{K_P}$$



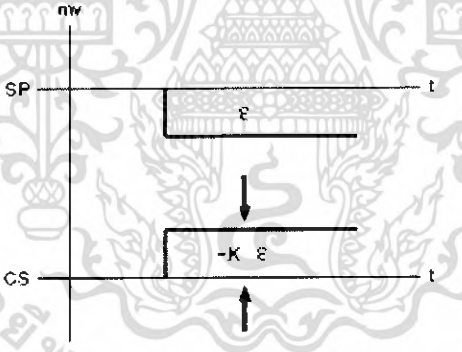
รูปที่ 2.3 แสดงบล็อกไดอะแกรมของการควบคุมแบบ P

จากรูปที่2.3 เป็นการควบคุมแบบ P ซึ่งจะใช้ Gain เป็นตัวควบคุมระบบเพียงอย่างเดียว ถ้าค่า Gain มากการเปลี่ยนแปลงจะเร็ว ซึ่งอาจทำให้เกิดการการแกว่งของ Process ได้ ถ้าค่า Gain น้อยกว่าการเปลี่ยนแปลงของค่าอาจทำให้ Process ขาดการควบคุม (เกิดOffset) และถ้าปรับให้ Gain มีค่า = α หรือ PB = 0 การควบคุมแบบ P จะกลายเป็นการควบคุมแบบ ON-OFF ทันที ในกรณีที่มีค่าGain (K) มีค่าน้อยเกินไปอาจเกิดการที่ค่าที่วัดได้ไม่เท่ากับค่า SP หรือ Controller หา Set point ไม่พบเราเรียกว่าเกิดการ Offset ซึ่งสามารถแก้ไขได้โดยการให้ค่า Bias ที่เหมาะสม



รูปที่2.4 กราฟแสดงการเกิดค่า Offset ในระบบการควบคุมแบบ P

ส่วนในรูปที่ 2.4 เป็นกราฟแสดงการเกิด Offset จะเห็นว่าค่าเอาต์พุตไม่สามารถหาค่า Set point (SP) ได้ การเปลี่ยนแปลงของการควบคุมแบบ P จะมีผลตอบสนองออกมาเป็นกราฟดังรูปที่ 2. 5

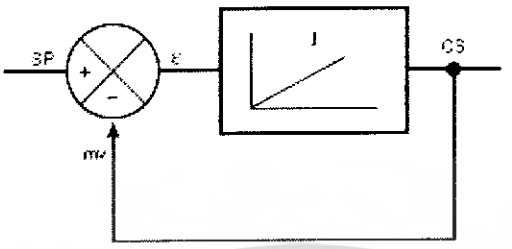


รูปที่ 2.5 กราฟแสดงผลตอบสนองของการควบคุมแบบ P

จะเห็นว่าลักษณะของกราฟจะเป็นเส้นตรงทั้งหมด ค่า SP และค่า CS จะมีลักษณะรูปกราฟเหมือนกันแต่จะแตกต่างกันที่เฟส เนื่องจากการควบคุมแบบ P จะไม่มีค่าเวลาเกี่ยวข้อง เมื่ออินพุตเปลี่ยนเอาต์พุตก็เปลี่ยนตามทันที

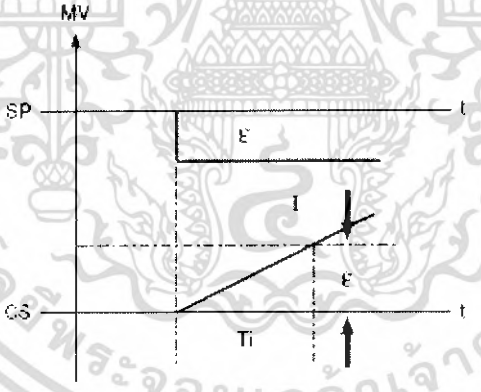
2.1.5 การควบคุมแบบ Integral (I)

การควบคุมแบบ I เป็นการควบคุมแบบอินทีเกรดสัญญาณ ดังในรูป



รูปที่2.6 แสดงบล็อกไดอะแกรมของการควบคุมแบบ I

ลักษณะของสัญญาณที่ผ่านการควบคุมแบบ I จะลาดชัน คือจะค่อยๆไต่สูงขึ้นจนถึงจุด Set Point การควบคุมแบบ I จะให้ผลตอบสนองต่อการควบคุมได้ดีกว่าการควบคุมแบบ P เนื่องจาการควบคุมแบบ I เมื่ออินพุตเปลี่ยนเอาท์พุตจะไม่เปลี่ยน โดยทันที แต่จะค่อยเปลี่ยนไปดังรูปที่ 7

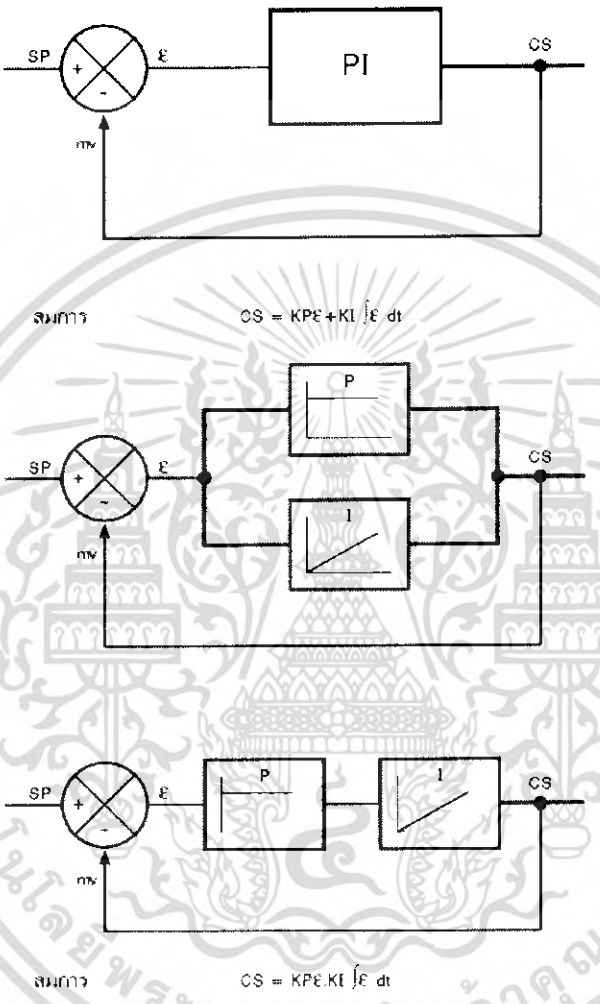


รูปที่ 2.7 แสดงการเปลี่ยนแปลงของการควบคุมแบบ I

จะเห็นว่าค่าเอาต์พุต หรือ CS จะค่อยๆไต่สูงขึ้นจนถึงค่าเป้าหมาย (SP) ในช่วงระยะเวลา Ti ซึ่งเป็นตัวกำหนดค่าเวลาของการควบคุมแบบ I ซึ่งสามารถเปลี่ยนแปลงได้ การควบคุมแบบ I มีลักษณะของสมการคำนวณดังนี้

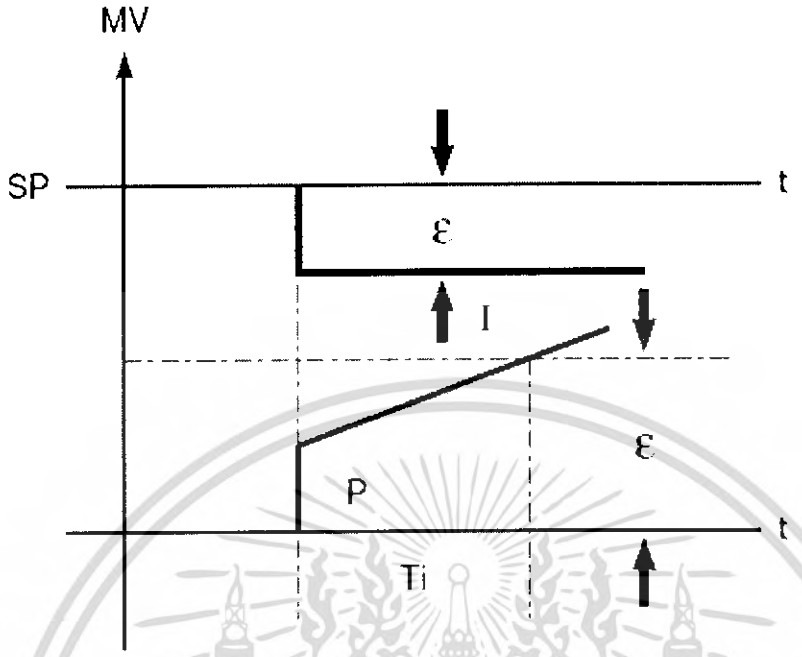
$$CS = \frac{1}{Ti} \int \sum dt$$

โดยปกติการควบคุมแบบ I จะไม่ใช้ตัวเดียวแต่จะใช้ควบคู่กับแบบ P โดยเรียกว่าเป็น PI Control ดังรูปที่2. 8 เป็นการควบคุมแบบ PI



รูปที่ 2.8 แสดงบล็อกไดอะแกรมของการควบคุมแบบ PI ในแต่ละแบบ

ซึ่งมีวิธีใช้งานอยู่ 2 แบบ แบบแรกเป็นการต่อแบบขนานโดยนำค่า P กับ I มาบวกกัน ส่วนแบบที่ 2 เป็นการต่อแบบอนุกรมโดยนำค่า P กับ I มาคูณกัน การต่อแบบทั้ง 2 แบบนี้จะช่วยให้การควบคุมมีประสิทธิภาพเพิ่มขึ้นกว่าการควบคุมแบบใดแบบหนึ่งเพียงอย่างเดียว

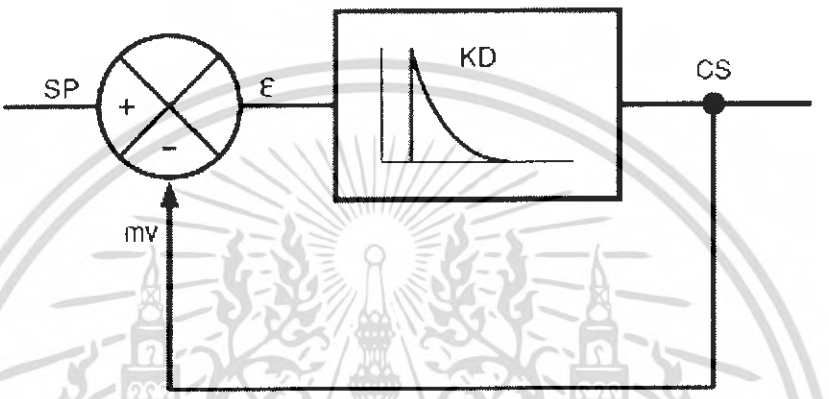


รูปที่2. 9 แสดงกราฟการเปลี่ยนแปลงของการควบคุมแบบ PI

ในรูปที่ 2.9 แสดงการเปลี่ยนแปลงของการควบคุมแบบ PI จะเห็นว่าค่าสัญญาณเอาต์พุตเริ่มต้นจะขึ้นเป็นเส้นตรงซึ่งเป็นผลของการควบคุมแบบ P จากนั้นเอาต์พุตก็จะเป็นเส้นลาดชัน ซึ่งเป็นผลของการควบคุมแบบ I นั่นเอง จุดเด่นของการควบคุมแบบ PI คือจะช่วยให้การควบคุมเอาต์พุตเข้าหาเป้าหมายได้เร็ว สรุปแล้วการเพิ่มการควบคุมแบบ I เข้าไปในการควบคุมแบบ P นั้น (PI Control) ก็เพื่อแก้ค่า Offset อันเกิดจาก P Control แต่ถ้ากำหนดค่า Ti น้อยเกินไปอาจทำให้ระบบเกิดการขาดเสถียรภาพ หรือเกิดออสซิลเลต (Oscilate)

2.1.6 การควบคุมแบบ **Derivative (D)**

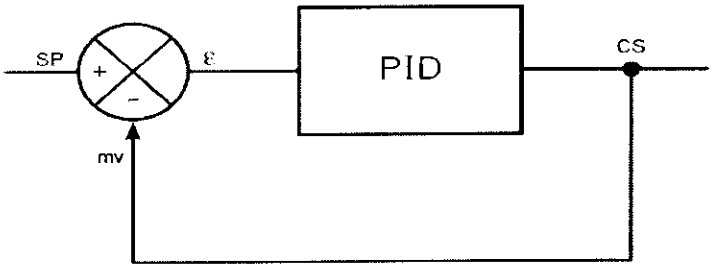
สมการ $CS = KD\, d\varepsilon / dt$

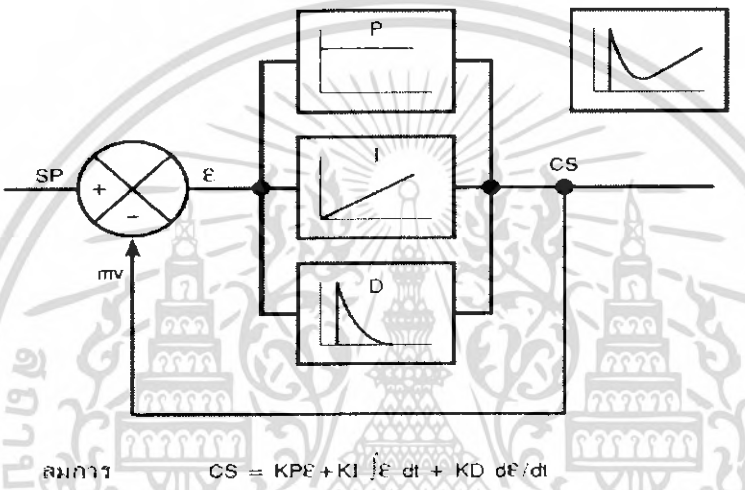


รูปที่2.10 แสดงบล็อกไดอะแกรมของการควบคุมแบบ D

การควบคุมแบบ (D) เป็นการควบคุมแบบดิฟสัญญาณ ลักษณะของสัญญาณเอาต์พุตเมื่อผ่านการควบคุมแบบ D แล้วจะเป็นลักษณะของการดิฟสัญญาณ คือช่วงเริ่มต้นสัญญาณจะขึ้นไปที่ค่าสูงสุดก่อนแล้วจะค่อยๆลดลงเป็นทางลาดจนถึงค่าเป้าหมายที่ตั้งไว้

การควบคุมแบบ D นี้จะไม่ใช้ตัวเดียวแต่ จะใช้ร่วมกับแบบ P เป็น PD Control หรือใช้รวมกันเป็น PID Control ดังรูปที่ 2.11

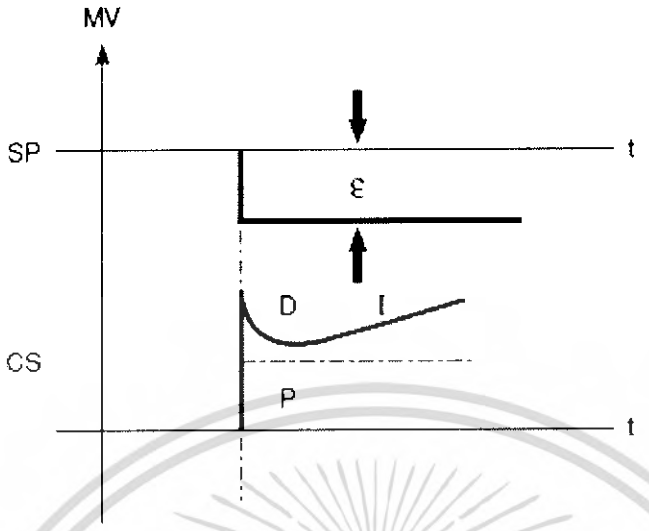




รูปที่ 2.11 แสดงบล็อกไดอะแกรมของการควบคุมแบบ PID

ซึ่งใช้การควบคุมแบบ P, แบบ I, และแบบ D มาต่อขนานกันทุกตัว ซึ่งจะช่วยใช้ระบบการควบคุมเข้าหาเป้าหมายได้เร็วและดีกว่าการควบคุมแบบอื่นๆ เช่นเมื่อมีการเปลี่ยนแปลงค่าเป้าหมาย (SP) หรือ Process เปลี่ยนไป การควบคุมแบบ D จะทำหน้าที่ปรับค่าเพื่อให้เข้าสู่เป้าหมายได้อย่างรวดเร็ว ส่วนแบบ PI จะทำให้การ เข้าสู่เป้าหมายไม่เกิดการแกว่ง

การควบคุมแบบ PID จะใช้กับค่าเป้าหมาย (SP) ที่เปลี่ยนแปลงอยู่ตลอดเวลา หรือจะใช้กับการควบคุมตัว Process ที่มีความเร็วต่อการตอบสนองช้าแต่มีค่า Dead time มากๆเช่น Temperature หรือตัว Process ที่มีการเปลี่ยนแปลงตลอดเวลา หรือการเกิดการรบกวนจากภายนอกของ Process ได้ง่าย ซึ่งลักษณะของสัญญาณเอาต์พุต (CS) จะมีการเปลี่ยนแปลงไปตามรูปที่ 2.12



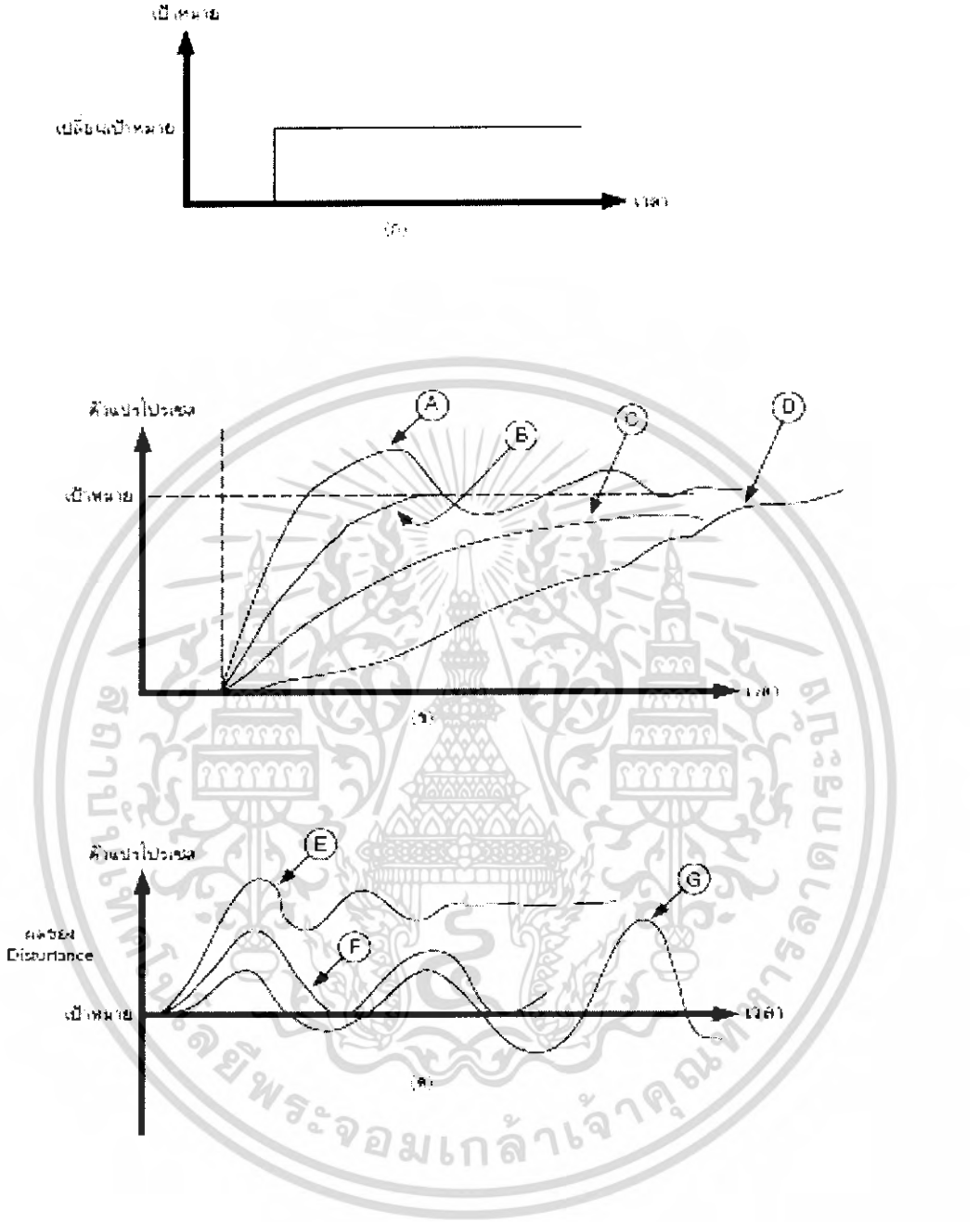
รูปที่2.12 แสดงกราฟการเปลี่ยนแปลงของการควบคุมแบบ PID

จะเห็นว่าสัญลักษณ์ของสัญญาณเอาต์พุต (CS) จะมีการผสมกันของแบบ P, แบบ I, และแบบ D รวมกันอยู่อย่างเห็นได้ชัด

2.2 การปรับค่าในระบบควบคุมแบบ PID

2.2.1 ผลตอบของการควบคุมที่ดี

ในระบบควบคุมป้อนกลับตัวควบคุมจะพยายามรักษาให้ค่าตัวแปรโปรเซสมีค่าเทากับค่าเป้าหมายอยู่เสมอ ในกรณีที่เกิด Disturbance ในระบบหรือมีการเปลี่ยนค่าเป้าหมายจะทำให้ตัวแปรโปรเซสมีค่าต่างจากค่าเป้าหมายไปขณะหนึ่ง จากนั้นตัวควบคุมจะพยายามควบคุมให้ตัวแปรโปรเซสมีค่าเท่ากับเป้าหมายนี้ในที่สุด ลักษณะการนำค่าตัวแปรโปรเซสให้เข้าใกล้ค่าเป้าหมายนี้ จะแตกต่างกันตามคุณสมบัติของระบบควบคุม บางระบบควบคุมอาจควบคุมให้ตัวแปรโปรเซสส่งเข้าหาเป้าหมายได้เร็ว แต่บางระบบอาจทำได้ดีกว่า เราสามารถทดสอบความสามารถของระบบควบคุมนี้โดยดูที่ผลการตอบของการควบคุมตามในรูปที่ 2.13



รูปที่ 2.13 แสดงกราฟผลตอบของระบบควบคุม

จากระบบควบคุมแบบ PID ถ้าลองเปลี่ยนค่าเป้าหมายตามรูปที่ 2.13 (ก) ตัวควบคุมจะพยายามควบคุมให้ตัวแปรโปรเซสวิ่งเข้าหาเป้าหมายนั้น ผลตอบของตัวแปรโปรเซสจะมีหลายแบบดังแสดงในรูปที่ 2.13 (ข) ซึ่งสามารถอธิบายแต่ละจุดได้ดังนี้

- จุด A มี Overshoot และการแกว่ง เข้าหาเป้าหมายได้ไม่ดี

- จุด B มีการตอนรับเร็ว โปรเซสเข้าหาเป้าหมายได้เร็ว
- จุด C ตัวแปรโปรเซสมีค่าไม่เท่ากับค่าเป้าหมาย แม้เวลาจะผ่านพ้นไปนานเรียกว่าเกิด offset
- จุด D มีการตอบรับช้ามากโปรเซสเข้าหาเป้าหมายได้ช้า

เมื่อตัวแปรโปรเซสหยุดนิ่ง ที่ค่าเป้าหมาย และในขณะนั้นเกิดมี Disturbance เข้ามารบกวนในโปรเซส (Disturbance ได้แก่ การเปลี่ยนแปลงของโหลดหรือการเปลี่ยนแปลงของสภาพแวดล้อมเป็นต้น) ผลของโปรเซสมีหลายแบบดังแสดงในรูปที่ 2.13(ค)

- จุด E เกิด offset ได้ค่าผิดจากเป้าหมายเดิม
- จุด F เกิดการแกว่งเล็กน้อยก่อนกลับสู่ค่าเป้าหมายเดิม
- จุด G เกิดการแกว่งและขาดเสถียรภาพ

เมื่อพิจารณาผลตอบของการควบคุมชนิดต่างๆ เหล่านี้จึงจะพอสรุปได้ว่าระบบการควบคุมที่ดีจะต้องมีคุณสมบัติดังนี้ คือ มีเสถียรภาพไม่เกิดการแกว่ง (Oscillation) เมื่อถูกการกระตุ้นและการตอบรับการเปลี่ยนค่าเป้าหมาย Disturbance ได้รวดเร็วไม่เกิด offset

2.2.2 ความยากง่ายในการควบคุมของโปรเซส

ค่า Dead Time เป็นศัตรูตัวร้ายของตัวควบคุม ลักษณะสมบัติของโปรเซสทั่วๆ ไป มักจะมีค่า Dead Time ปรากฏให้เห็นจากรูปคลื่นของผลโปรเซสต่อ Step input เราสามารถหาค่า dead time (LE) และค่าคงตัวเวลา (TE) โดยประมาณได้ ซึ่งอัตราส่วน LE/TE นี้ จะเป็นค่าที่ใช้ประเมินความยากง่ายในการควบคุม และใช้เลือกแบบการควบคุมที่เหมาะสมกับโปรเซสตามในตารางที่ 2.2.2

LE/TE	แบบการควบคุมที่เหมาะสม
$LE/TE < 0.2$	ON-OFF,P,PI
$0.2 < LE/TE < 1.0$	PI,PID
$1.0 < LE/TE$	Feed Forward, Computer Control

ตารางที่ 2.2.2 แสดงการควบคุมที่เหมาะสมกับโปรเซส

2.2.3 ผลของ PID ต่อเสถียรภาพของระบบ

ในระบบควบคุมแบบป้อนกลับ ซึ่งใช้ตัวควบคุมแบบ PID นั้น ถ้าทดลองเปลี่ยนค่า PB, T_I และ T_D จะมีผลต่อผลตอบของระบบการควบคุมดังนี้

ผลของการควบคุมแบบ P เมื่อลดค่า PB ลง ทำให้อัตราขยายสูงขึ้นจะมีผลทำให้ค่า offset ลดลง รูปคลื่นช่วงเวลาของการแกว่งเล็กลง และอัตราส่วนของช่วงกว้างการแกว่งเพิ่มขึ้น ระบบขาดเสถียรภาพมากขึ้น

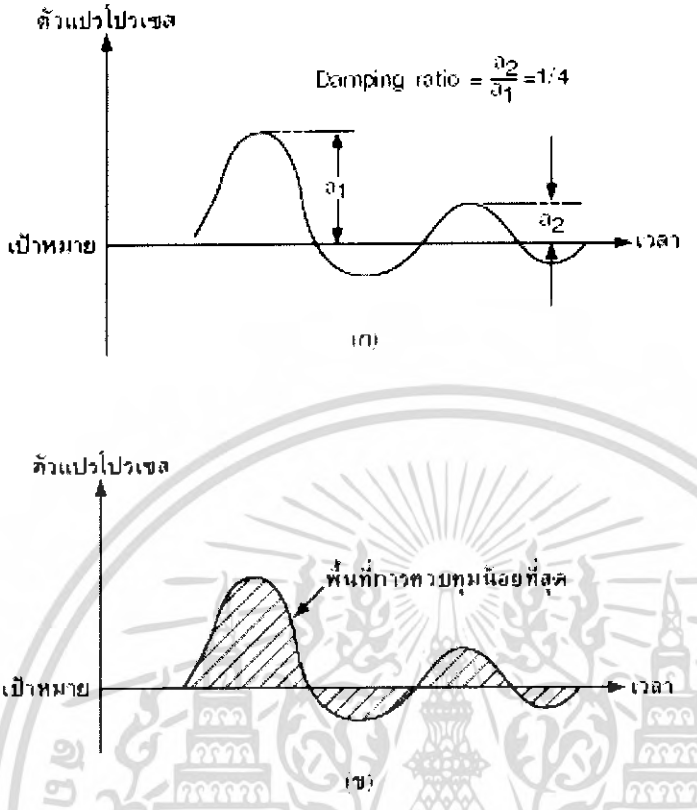
ผลการควบคุมแบบ I เมื่อให้การควบคุมแบบ P และ D คงที่แล้วทดลองลดค่า T_I จะมีผลทำให้ค่า offset หายไป ผลตอบจะเร็วขึ้น และอัตราส่วนของช่วงกว้างการแกว่งเพิ่มขึ้นระบบขาดเสถียรภาพมากขึ้น

ผลการควบคุมแบบ D เมื่อให้การควบคุมแบบ P และ I คงที่แล้ว ทดลองเพิ่มค่า T_D ให้ยาวขึ้น จะมีผลทำให้อัตราส่วนของช่วงกว้างการแกว่งลดลงระบบมีเสถียรภาพมากขึ้น และรูปคลื่นช่วงเวลาของการแกว่งสั้นลง

ตามปกติการใช้ค่าเวลา T_D ให้ยาวมีแนวโน้มที่จะให้ระบบมีเสถียรภาพก็จริง แต่ก็มีจุดอ่อนตรงตอบรับต่อนอยส์ (Noise) ได้ง่าย ทำให้ผลตอบของระบบไวเกินไป

2.2.3 คุณภาพของการควบคุม

ดังที่ได้กล่าวมาแล้วว่า ระบบควบคุมที่ดีจะต้องมีคุณสมบัติ คือมีเสถียรภาพผลตอบเร็ว และไม่เกิด Offset แต่จะใช้อะไรในการตัดสินว่าการควบคุมไหนจะดีที่สุดนั้น จะต้องมีเกณฑ์ตัดสินคุณภาพการควบคุม ซึ่งจะเห็นว่าในการควบคุมโปรเซส อัตราส่วนช่วงกว้างการแกว่ง (Damping ratio criterion) เป็นการนิยมใช้มากที่สุด เพราะสามารถวัดผลได้ง่าย การควบคุมที่ดีจะให้อัตราส่วนช่วงกว้างการแกว่างประมาณ 25% หรือ 1 ต่อ 4 ตามรูปที่ 2.14 (ก)



รูปที่2.14 แสดงรูปกราฟที่มีพื้นที่การควบคุมน้อยที่สุด

จากรูปที่ 2.14 (ข) เมื่อโปรเซสแกว่งเข้าหาเป้าหมาย จะเห็นว่าพื้นที่การควบคุมที่ดีจะต้องแกว่งเข้าหาเป้าหมายประมาณ 4 ลูกคลื่น เพื่อให้ระบบควบคุมสามารถเข้าหาเป้าหมายได้เร็วและไม่เกิด offset ถ้าหากพื้นที่การควบคุมแกว่งมากกว่านี้จะทำให้ระบบขาดเสถียรภาพใช้งานได้ไม่ดีเท่าที่ควร

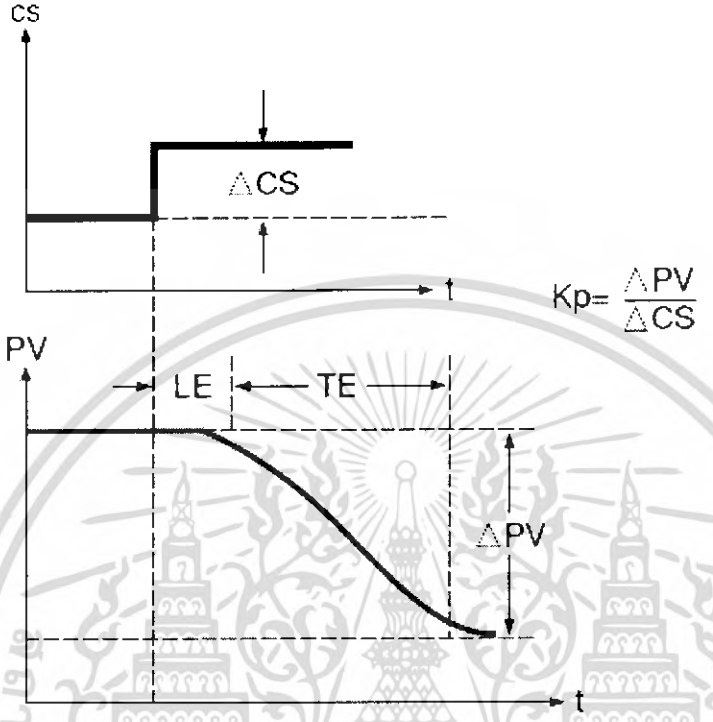
2.2.4 วิธีการปรับค่า PID

ได้มีนักคณิตศาสตร์คิดวิธีที่จะหาทางตั้งค่า PID เมื่อให้ได้การควบคุมที่มีคุณภาพที่ดีที่สุดหลายวิธีด้วยกัน แต่ในที่นี้ขอยกตัวอย่างเพียงแค่ 2 วิธีเท่านั้น ซึ่งในทางปฏิบัติเป็นวิธีที่นิยมใช้กันมาก การปรับค่า PID เพื่อให้ได้ค่า Damping ratio เท่ากับ 25% สามารถทำได้ 2 วิธี ดังนี้

วิธีที่ 1

1. ให้เปลี่ยนระบบควบคุมเป็นวงรอบเปิด (Open Loop)
2. หา Process Characteristic โดยเปลี่ยนค่าสัญญาณควบคุมไป ΔCS
3. แล้วบันทึกรูปคลื่นของตัวแปรโปรเซส

4. หา Process gain (Kp), dead time (LE) และค่าคงตัวเวลา (TE) จาก Process Characteristic ในรูปที่ 2.15 และคำนวณหาค่า KP ตามสูตร ดังนี้



รูปที่ 2.15 แสดงลักษณะของรูปคลื่นโปรเซสในการปรับค่า PID ในวิธีที่ 1

5. นำค่า Kp , LE , TE ที่หาได้ไปคำนวณหา PB , T_I และ T_D จากตารางที่ 2

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

ตารางที่ 2.2.3 แสดงการคำนวณหาค่า PB, TI และ TD ในการปรับค่า PID

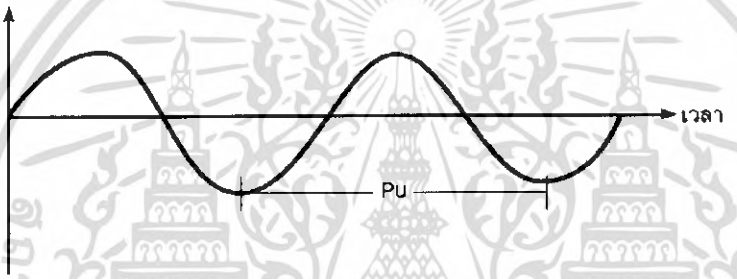
วิธีที่ 2

1. ให้ระบบควบคุมเป็นแบบวงรอบปิด (Closed loop)

2. ตั้ง T_I สูงสุด (max) และ T_D ต่ำสุด (O หรือ min) ใช้การควบคุมแบบ P อย่างเดียว

3. ครั้งแรกตั้งค่า PB ไว้ที่ค่าสูงสุด แล้วลองลดค่า PB ลงมา ลองเปลี่ยนค่าเป้าหมายเพื่อดูผลตอบแทนและลดค่า PB ให้ต่ำลงเรื่อยๆ จนถึงค่าที่เมื่อเปลี่ยนค่าเป้าหมายไปเล็กน้อย จะทำให้โปรเซสเกิดการแกว่งต่อเนื่องไปตลอด ค่า PB ในขณะนั้นเรียกว่า PB_U (Ultimate proportional band)

4. หาคาบเวลาในการแกว่ง ให้เท่ากับ P_U (ตามในรูปที่ 2.16)

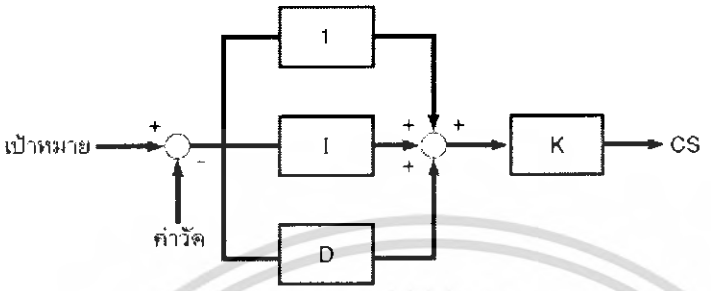


รูปที่ 2.16 แสดงการหาคาบเวลาในการแกว่งของการปรับค่า PID ในวิธีที่ 2

5. นำค่า PBU และ PU ที่หาได้ไปใช้คำนวณหาค่า PB, T_I และ T_D จากตารางที่ 3

ชนิดการควบคุม	PB	T_I	T_D
P	2 PB_U	∞	0
PI	2.2 PB_U	0.83 Pu	0
PID	1.7 PB_U	0.5 Pu	0.125 Pu

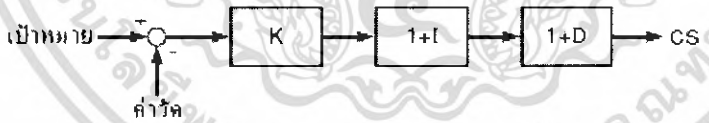
ตารางที่ 2.2.4 แสดงการคำนวณหาค่า PB, TI และ TD ในการปรับค่า PID ในวิธีที่ 2



รูปที่ 2.17 แสดงบล็อกไดอแกรมของตัวควบคุมแบบดีที่สุด

2.2.5 การรบกวนซึ่งกันและกันของค่า PID

จากรูปที่2.17 แสดงบล็อกไดอะแกรมของตัวควบคุมแบบอุดมคติ ซึ่งค่า PID จะสามารถตั้งได้อย่างอิสระไม่มีการรบกวนซึ่งกันและกัน ตัวควบคุมแบบนี้มักจะมีราคาแพง โดยทั่วๆ ไปตัวควบคุมที่มีขายในท้องตลาดมักจะดัดแปลงวงจรให้ง่ายลง โดยมีบล็อกไดอะแกรมดังรูปที่ 2.18



รูปที่ 2.18 แสดงบล็อกไดอะแกรมของตัวควบคุมที่มีขายทั่วๆไป

จะเห็นว่า I และ D คอนโทรล ไม่เป็นอย่างอุดมคติ และจะมีการรบกวนซึ่งกันและกัน (Mutual interfere) ซึ่งได้แก่การเปลี่ยนค่า T_I จะมีผลให้ PB และ T_D เปลี่ยนแปลงไป หรือถ้าเปลี่ยนค่า T_D ก็จะมีผลทำให้ PB และ T_Iเปลี่ยนไปเช่นกัน

ดังนั้นเมื่อทำการปรับค่า PID ที่เหมาะสมได้แล้ว ให้ทำการคำนวณหาค่า PB , T_I และ T_D จากวิธีการปรับค่า PID ทั้ง 2 วิธีข้างต้น ซึ่งจริงๆ แล้วค่าเหล่านี้ยังนำมาตั้งที่ตัวควบคุมไม่ได้ ก่อนอื่นต้อง

กำหนดให้ PB , T_I , T_D เป็นค่าตั้งที่ตัวควบคุม (Dial Setting) และให้ PB' , T'I , T'D เป็นค่าที่คำนวณได้ (Effective value) ทั้งสองค่าจะมีความสัมพันธ์กันตามสูตรดังนี้

$$PB = \mu PB'$$

$$T_I = \frac{T_I'}{\mu}$$

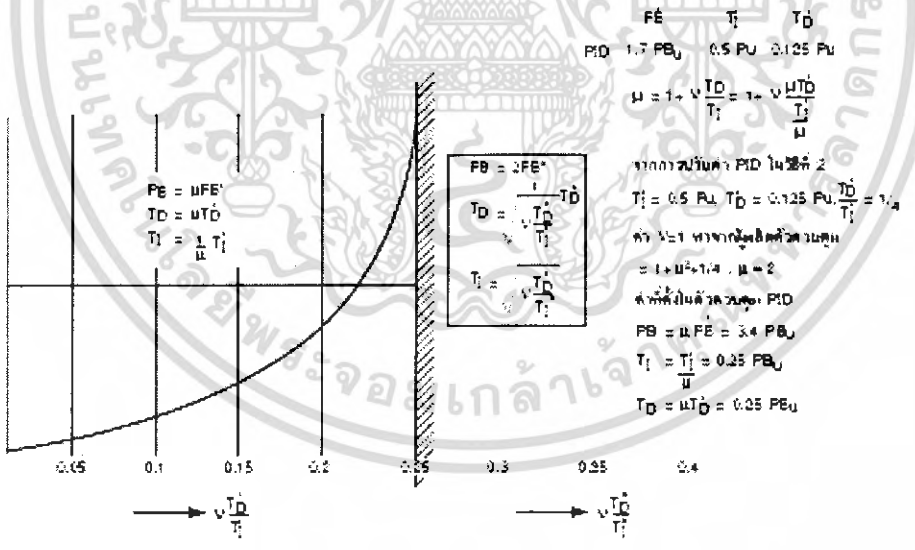
$$T_D = \mu T_D'$$

โดยที่ μ คือสัมประสิทธิ์การรบกวนซึ่งกันและกัน (Mutual Interference Coefficient)

ค่าμ นี้จะมีค่าไม่คงที่แน่นอน จะขึ้นกับอัตราส่วนของ TI กับ TD และขึ้นกับชนิดของตัวควบคุมด้วย μ นี้สัมพันธ์กับค่า

$$V\frac{T_D'}{T_I'}$$

(V คือค่าคงตัวค่าหนึ่งซึ่งบริษัทผู้ผลิตตัวควบคุมจะต้องกำหนดให้) ซึ่งแสดงดามกราฟ



ใน

รูปที่ 2.19 กราฟแสดงการหาค่า μ

$$V\frac{T_D'}{T_I'}$$

ต้องมีค่าน้อยกว่า 0.25 ถ้ามีค่ามากกว่าจะหาความสัมพันธ์ไม่ได้

2.2.6 วิธีการตั้งค่า PID (Dial Setting) ในตัวควบคุมที่มีการรบกวนกัน

1. คำนวณหาค่า PB', T_I', T_D' ที่ได้จาก Optimum tuning method
2. คำนวณหาค่า $V\frac{T_D'}{T_I'}$ โดยที่ V หาจากผู้ผลิตตัวควบคุม
3. หา μ จากกราฟในรูปที่ 7
4. คำนวณหาค่า PB , T_I , T_D จากสูตรข้างบน
5. นำค่า PB ,T_I , T_D ที่คำนวณได้ไปตั้งที่ตัวควบคุม

สรุปแล้วรายละเอียดที่ต้องใช้เกี่ยวกับการปรับค่า PID จะประกอบด้วย 3 อย่าง คือ ค่า PB , T_I , T_D ซึ่งจะเป็นค่าที่ได้จากการคำนวณหรือเป็นค่าที่ได้จากการทดลองสุ่มตัวอย่างก็ได้ เพื่อที่จะนำค่าเหล่านี้ไปใช้ปรับแต่งในระบบควบคุมแบบ PID ให้ได้ผลตอบของการควบคุมที่มีประสิทธิภาพมากที่สุดปกติการปรับค่า PID ให้ได้ผลดีนั้นจะต้องทำการทดลองปรับหลายๆ ครั้งจนกว่าจะได้ค่า PB , T_I , T_D ที่เหมาะสมพร้อมที่จะนำไปใช้งานจริงได้

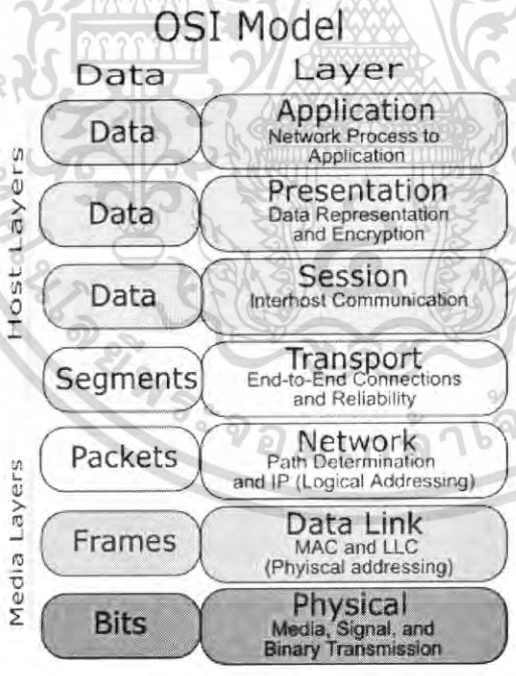
2.3 MODBUS PROTOCOL

MODBUS เป็นโปรโตคอลที่นิยมใช้กันมากในภาคอุตสาหกรรม โดยเฉพาะในระบบควบคุมอัตโนมัติ ด้วยโครงสร้างที่ได้รับการออกแบบให้ใช้งานง่าย สามารถนำไปใช้กับระบบรับส่งข้อมูลได้หลายประเภท เช่น ระบบรับส่งข้อมูลแบบอนุกรม หรือเครือข่าย LAN ทำให้ได้รับความนิยมใช้งานอย่างกว้างขวาง โดยจะสังเกตเห็นได้ว่า อุปกรณ์ที่สามารถติดต่อสื่อสารได้ในภาคอุตสาหกรรม ไม่ว่าจะเป็นอุปกรณ์ควบคุมอัตโนมัติ เครื่องมือวัดคุมชนิดต่างๆ ส่วนใหญ่จะสามารถสื่อสารด้วยโปรโตคอล MODBUS ได้ทั้งนั้น หรือแม้กระทั่งในงานออกแบบระบบไมโครคอนโทรลเลอร์ ที่มีพอร์ตสื่อสารข้อมูลอนุกรมให้อยู่แล้ว ก็สามารถเขียนโปรแกรมควบคุมพอร์ตให้สื่อสารข้อมูลด้วยโปรโตคอล MODBUS ได้ ซึ่งนั่นหมายความว่า เราสามารถออกแบบระบบไมโครคอนโทรลเลอร์ของเราเอง ให้สามารถติดต่อพูดคุยกับอุปกรณ์มาตรฐานที่มีขายอยู่ในท้องตลาดได้ ทำให้เป็นการเพิ่มระดับขีด

ความสามารถในงานออกแบบระบบไมโครคอนโทรลเลอร์ให้สูงขึ้นไปอีก และหากได้ทำความเข้าใจกับโปรโตคอลตัวนี้แล้ว จะพบว่าไม่ยากเลยที่จะทดลองพัฒนาขึ้นมาใช้งานเอง

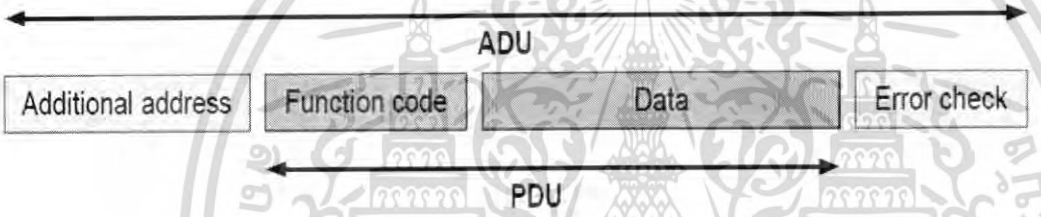
2.3.1 ลำดับชั้นของ MODBUS ในแบบจำลอง OSI

MODBUS เป็นโปรโตคอลรับส่งข้อมูลในลำดับชั้นแอพพลิเคชั่น หรือ application layer เป็นลำดับชั้นที่ 7 ซึ่งจัดเป็นลำดับชั้นสูงสุดในแบบจำลอง OSI การที่ MODBUS ถูกออกแบบให้อยู่ในลำดับชั้นสูงสุดนั้น มีข้อดีโดยทำให้สามารถนำไปใช้กับระบบบัสข้อมูล หรือเครือข่ายแบบต่างๆได้หลากหลายชนิด เนื่องจากความแตกต่างเหล่านี้เป็นเรื่องของลำดับชั้นที่อยู่ถัดลงมา โดยเฉพาะในลำดับชั้นกายภาพ หรือ Physical Layer ซึ่งเป็นตัวกำหนดลักษณะรูปแบบทางกายภาพของตัวกลางนำสัญญาณข้อมูล ซึ่งไม่มีผลต่อลำดับชั้นที่อยู่สูงกว่า



รูปที่2.3.1 MODBUS กับแบบจำลอง OSI

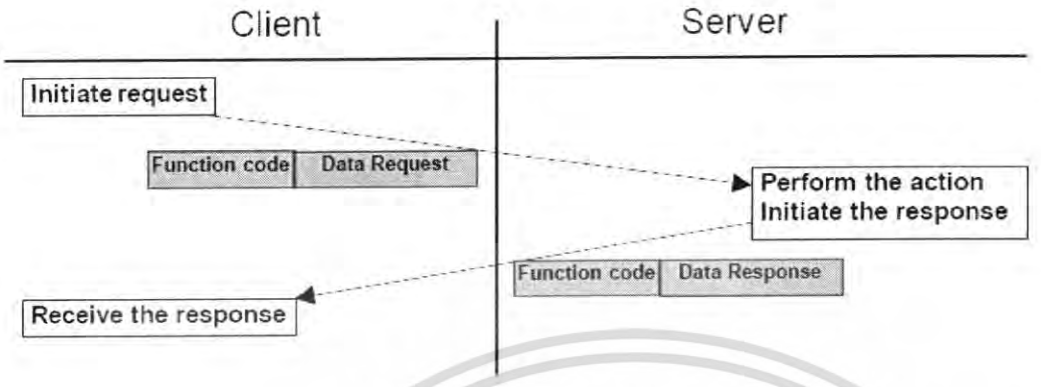
MODBUS มีรูปแบบการสื่อสารข้อมูลแบบ ไคลเอนท์/เซิร์ฟเวอร์ หรือ มาสเตอร์/สเลฟ สามารถเชื่อมต่อกับระบบบัสข้อมูลหรือเครือข่ายชนิดต่างๆ ได้หลายชนิด เช่น โปรโตคอล TCP/IP บนเครือข่าย Ethernet , บัสรับส่งข้อมูลอนุกรมแบบต่างๆ เช่น EIA/TIA-232-E , EIA-422 , EIA/TIA-485-A ทั้งแบบที่ใช้ตัวกลางนำสัญญาณข้อมูลแบบเคเบิลตัวนำไฟฟ้า , เคเบิลใยแก้วนำแสง , คลื่นสัญญาณวิทยุ เป็นต้น ส่วนจะเชื่อมต่อกันอย่างไรนั้น ขึ้นอยู่กับชนิดลักษณะของลำดับชั้นกายภาพ เช่น หากเลือกแบบ EIA/TIA-232 หรือ EIA/IA-485 ก็จะใช้ตัวเชื่อมต่อที่ถูกออกแบบให้อยู่ในลำดับชั้นเชื่อมประสานข้อมูล หรือ data-link layer เฟรมข้อมูลที่ใช้ในโปรโตคอล MODBUS เรียกว่า Protocol Data Unit หรือ PDU ประกอบด้วยฟิลด์ข้อมูล 2 ส่วน ส่วนแรกคือ หมายเลขคำสั่งหรือ Function code ที่ต้องการให้



รูปที่ 2.3.2 เฟรมข้อมูล ADU และ PDU

ดำเนินการกับข้อมูลที่อยู่ในส่วนที่สอง PDU มีความเป็นอิสระ ไม่ขึ้นอยู่กับลักษณะทางกายภาพของบัสหรือเครือข่าย ซึ่งเมื่อนำไปใช้จริงบนบัสหรือเครือข่ายชนิดใด ก็จะเพิ่มส่วนประกอบสำคัญอีก 2 ส่วนลงไป ได้แก่ ข้อมูลแอดเดรส (additional address) และอีกส่วนหนึ่งคือข้อมูลที่ได้จากกระบวนการเข้ารหัสเพื่อใช้ตรวจสอบความถูกต้องของข้อมูลที่ได้รับ (error check) เรียก PDU ที่เพิ่มส่วนประกอบ 2 ส่วนนี้ว่า Application Data Unit (ADU)

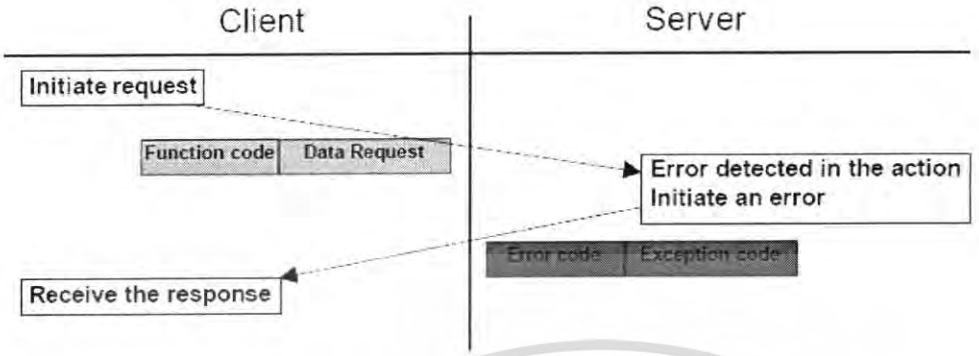
ในระบบโปรโตคอล MODBUS ไคลเอนท์จะเป็นตัวเริ่มต้นการสื่อสาร โดยสร้างเฟรมข้อมูล ADU ส่งไปยังเซิร์ฟเวอร์ เพื่อแจ้งให้ทราบว่าต้องการทำอะไร ตามหมายเลขฟังก์ชั่นที่ระบุใน ADU ส่วนข้อมูลที่ระบุถัดจากหมายเลขฟังก์ชั่นนั้น ใช้ประกอบการทำงานตามฟังก์ชั่น เช่น เป็นพารามิเตอร์บอกตำแหน่งแอดเดรสของข้อมูลที่จะทำการอ่านด้วยฟังก์ชั่นอ่านข้อมูล ข้อมูลมีทั้งแบบลอจิก (0) หรือ (1) และแบบรีจิสเตอร์หรือแอนาลอก (เช่น ค่า 134.25) รวมทั้งข้อมูลรหัสตัวอักษร ซึ่งก็จัดเป็นข้อมูลแบบรีจิสเตอร์ประเภทหนึ่ง



รูปที่ 2.3.3 การรับส่งข้อมูลระหว่างไคลเอนท์กับเซิร์ฟเวอร์

สำหรับในบางฟังก์ชั่น อาจไม่ปรากฏฟิลด์ข้อมูล หรืออีกนัยหนึ่งก็คือ เฟรมข้อมูลสำหรับฟังก์ชั่นนั้นๆ ไม่ต้องอาศัยข้อมูลประกอบเพิ่มเติมแต่อย่างใด

รูปที่ 2.3.3 แสดงขั้นตอนการรับส่งเฟรมข้อมูลระหว่างไคลเอนท์กับเซิร์ฟเวอร์ เริ่มต้นโดย ไคลเอนท์ส่งเฟรมข้อมูลระบุหมายเลขฟังก์ชั่น ประเภท ตำแหน่ง และจำนวนข้อมูลที่ต้องการ (data request) ไปที่เซิร์ฟเวอร์ เมื่อเซิร์ฟเวอร์ได้รับเฟรมข้อมูลถูกต้องครบถ้วน และสามารถประมวลผลฟังก์ชั่นตามที่ระบุได้เสร็จเรียบร้อย ก็จะส่งเฟรมข้อมูลที่บรรจุข้อมูลที่ต้องการ (data response) กลับมาให้ไคลเอนท์ตัวนั้น โดยที่ส่วนแรกของเฟรมข้อมูลจะระบุหมายเลขฟังก์ชั่นเดิมที่ได้รับจากไคลเอนท์ แต่หากเกิดความผิดพลาดขึ้นมา ซึ่งอาจจะเป็นการที่เฟรมข้อมูลที่เซิร์ฟเวอร์ได้รับมีความผิดพลาด หรือเฟรมข้อมูลถูกต้อง แต่เซิร์ฟเวอร์ไม่สามารถทำงานตามฟังก์ชั่นที่ระบุได้ เซิร์ฟเวอร์ก็จะส่งเฟรมข้อมูลที่บรรจุข้อมูลหมายเลขความผิดพลาดหรือ Exception code ไป



รูปที่ 2.3.4 เกิดความผิดพลาดในการรับส่งข้อมูล

ให้กับไคลเอนท์แทน เพื่อแจ้งให้ทราบว่าเกิดปัญหาอะไร ในกรณีนี้ ส่วนแรกของเฟรมข้อมูลที่เซิร์ฟเวอร์ส่งกลับมานั้น เรียกว่า Exception function code ที่เป็นหมายเลขฟังก์ชั่นเดิม แต่ปรับค่าบิตบนสุด (MSB) ให้เป็น 1 เนื่องจากในอดีต การใช้งานโปรโตคอล MODBUS เริ่มต้นในระบบสื่อสารข้อมูลแบบอนุกรม ที่มีการกำหนดให้ขนาดของเฟรมข้อมูลในการสื่อสารซึ่งก็คือ ADU มีความยาวไม่เกิน 256 ไบต์ และเมื่อหักเอาส่วนที่เป็น address ขนาด 1 ไบต์ และรหัสตรวจสอบความผิดพลาด หรือ error check ขนาด 2 ไบต์ออก ก็จะเหลือเป็นเฟรมข้อมูล PDU ขนาด 253 ไบต์ และยึดค่านี้เป็นมาตรฐาน จากนั้น เมื่อมีการนำเอาโปรโตคอล MODBUS ไปใช้กับการสื่อสารในระบบอื่นๆ เช่น TCP/IP ค่าความยาวของ PDU ค่านี้ ก็ถูกนำไปใช้อ้างอิงด้วยเช่นกัน ระบบการจัดแบ่งกลุ่มข้อมูลที่ใช้ร่วมกับการสื่อสารด้วยโปรโตคอล

Primary tables	Object type	Type of access	Comments
Discretes Input	Single bit	Read-Only	This type of data can be provided by an I/O system.
Coils	Single bit	Read-Write	This type of data can be alterable by an application program.
Input Registers	16-bit word	Read-Only	This type of data can be provided by an I/O system.
Holding Registers	16-bit word	Read-Write	This type of data can be alterable by an application program.

ตารางที่2.3.1 แสดงกลุ่มข้อมูลชนิดต่างๆ

MODBUS มีลักษณะดังในรูปที่ 5 แบ่งออกเป็น 4 กลุ่ม ดังนี้

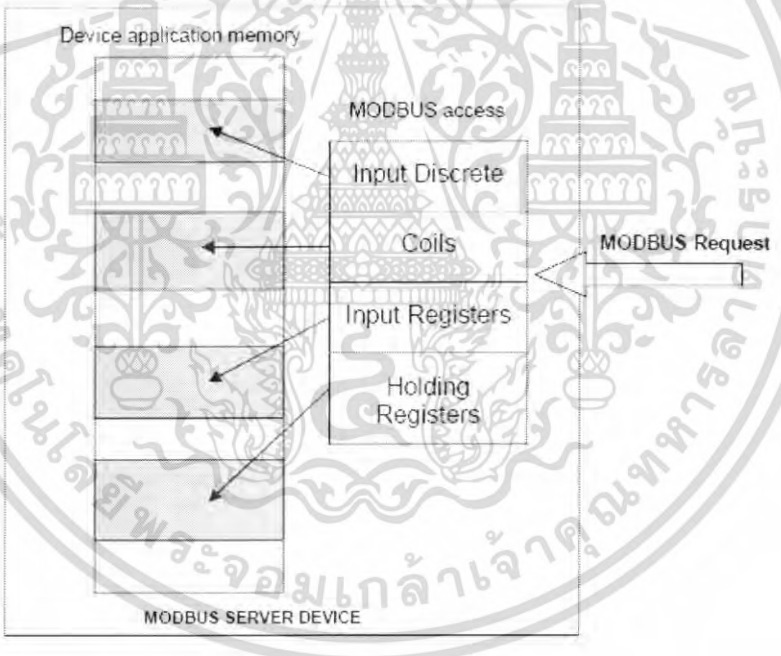
1. Discrete Input เป็นกลุ่มข้อมูลแบบบิต ใช้อ่านอย่างเดียว แก้ไขไม่ได้ เปรียบเสมือนกับหน่วยอินพุตที่เชื่อมต่อรับค่าสถานะจากวงจรอินพุตที่เป็นฮาร์ดแวร์ภายนอก รวมทั้งอาจใช้เป็นพารามิเตอร์แสดงสถานะต่างๆ ของระบบ ดังเช่น ภายในตัว PLC จะมีหน่วยความจำพิเศษแบบบิตที่ใช้บอกค่าสถานะต่างๆ เช่น พัลส์สัญญาณนาฬิกา, พัลส์ที่มีค่าเป็น 1 เฉพาะในลูปการทำงานแรก, บิตแสดงโหมดการทำงานของ PLC เป็นต้น ซึ่งบิตพิเศษเหล่านี้ ผู้ใช้งานไม่สามารถเปลี่ยนแปลงแก้ไขได้ ทำได้เพียงอ่านค่าเท่านั้น

2. Coils เป็นกลุ่มข้อมูลแบบบิตเหมือนกัน แต่สามารถอ่านและเขียนได้ ใช้เป็นประโยชน์สำหรับการประมวลผลฟังก์ชั่นต่างๆ เช่น ใช้เก็บค่าตัวแปรแบบบิตชั่วคราวในขณะประมวลผล

3. Input Registers เป็นกลุ่มข้อมูลแบบรีจิสเตอร์ที่อ่านได้อย่างเดียว เปรียบเสมือนกับหน่วยความจำที่เชื่อมต่อรับค่าแอนาลอกจากอุปกรณ์ฮาร์ดแวร์ภายนอก เช่น วงจรแอนาลอกอินพุต

4. Holding Registers เป็นกลุ่มข้อมูลแบบรีจิสเตอร์ที่สามารถอ่านและเขียนได้ เช่นเดียวกับ Coils คือ ใช้ประโยชน์ในการประมวลผลฟังก์ชั่นต่างๆ

ภายในเฟรมข้อมูล MODBUS PDU ข้อมูลถูกอ้างอิงตำแหน่งด้วยหมายเลขตั้งแต่ 0 ถึง 65535 โดยข้อมูลภายในกลุ่มข้อมูลแต่ละกลุ่มจะถูกกำหนดหมายเลขอ้างอิงเริ่มต้นที่เลข 1 เหมือนกันทุกกลุ่ม ค่าข้อมูลในตำแหน่งที่ถูกระบุถึง จะถูกอ่านและส่งต่อไปยัง Device application ซึ่งเป็นแอพพลิเคชั่น หรือการประมวลผลที่เกิดขึ้นจากฟังก์ชั่นที่ถูกระบุหมายเลขในเฟรมข้อมูลนั้นๆ ซึ่งแอพพลิเคชั่นจะอ่านเขียนข้อมูลกับกลุ่มข้อมูลอย่างไรนั้น ขึ้นอยู่กับโครงสร้างการทำงานของฮาร์ดแวร์แต่ละประเภท



รูปที่ 2.3.5 การเข้าถึงและอ่านค่าข้อมูลชนิดต่างๆ

2.3.2 หมายเลขคำสั่งหรือ Function Code

การประมวลผล หรือทำงานในฟังก์ชั่นต่างๆเกิดขึ้นได้โดยที่ระบบ จะไปทำอ่านคำสั่งตามตำแหน่งที่ระบุในหมายเลขฟังก์ชั่น หลักการก็คือ ดูว่าฟังก์ชั่นที่ต้องการให้ประมวลผลมีหมายเลขใด

จากนั้นก็จะไปค้นดูในตารางอ้างอิงว่าฟังก์ชั่นหมายเลขที่ระบุนี้ จะสั่งให้ระบบทำงานอะไรเช่น ฟังก์ชั่นหมายเลข 02 สั่งให้ระบบอ่านข้อมูลจากหน่วยความจำข้อมูลอินพุตแบบบิต

รูปที่2.3.1 แสดงการจัดแบ่งช่วงค่าหมายเลขที่ใช้อ้างอิงฟังก์ชั่น โดยแบ่งออกเป็น 2 ประเภท ได้แก่ ฟังก์ชั่นแบบ Public function code ซึ่งเป็นฟังก์ชั่นมาตรฐานของโปรโตคอล MODBUS ที่อุปกรณ์ทุกตัวที่สื่อสารด้วย MODBUS ทุกตัวจะต้องรู้จักและเรียกใช้งานได้ ส่วนฟังก์ชั่นที่ผู้ผลิตอุปกรณ์สามารถออกแบบเองได้ เรียกว่า user-defined function code หมายเลขเหล่านี้ ทั้ง Public และ user-defined จะต้องไม่ซ้ำกันเลย และมีค่าอยู่ในช่วงค่าที่กำหนดให้เท่านั้น เช่น ผู้ผลิตจะต้องเลือกค่าหมายเลขภายในช่วง 65 ถึง 71 และ 100 ถึง 109 สำหรับใช้ระบุตำแหน่งอ้างอิงของฟังก์ชั่นแบบ user-defined ที่พัฒนาขึ้นมาใช้งานกับผลิตภัณฑ์ของตน ส่วนฟังก์ชั่นแบบ Public นั้น ได้มีการระบุค่าตำแหน่งเอาไว้ตายตัวแน่นอนแล้ว ผู้ผลิตควรยึดตามค่าที่กำหนดไว้เหมือนๆ กัน ไม่ควรเปลี่ยนแปลง เพราะอาจเกิดปัญหาเมื่อต่อใช้งานร่วมกับผลิตภัณฑ์ของผู้ผลิตรายอื่น ทำให้ไม่สามารถทำงานร่วมกันได้

รูปที่ 2.3.6 ช่วงค่าแอดเดรสของฟังก์ชั่นทั้งแบบ Public และ User Defined

ฟังก์ชั่นแบบ Public สำหรับโปรโตคอล MODBUS แบ่งออกเป็นกลุ่มประเภทต่างๆ ดังแสดงในรูปที่ 8 สังเกตเห็นว่า มี 2 กลุ่มใหญ่ คือ กลุ่มฟังก์ชั่นการเข้าถึงข้อมูล (Data Access) และกลุ่มฟังก์ชั่นตรวจสอบระบบ (Diagnostics) และฟังก์ชั่นอีก1ฟังก์ชั่น ที่ใช้ในการอินเตอร์เฟสระบบ กลุ่มฟังก์ชั่นการเข้าถึงข้อมูลประกอบด้วยฟังก์ชั่นอ่านเขียนข้อมูลบนกลุ่มข้อมูลทั้ง 4 ชนิด รวมทั้ง

ฟังก์ชั่นอ่านเขียนไฟล์ข้อมูลด้วย ส่วนกลุ่มฟังก์ชั่นตรวจสอบระบบใช้สำหรับตรวจสอบการทำงานของระบบ เช่น ฟังก์ชั่นหมายเลข 07 คือฟังก์ชั่น แปลความหมายรหัสความผิดพลาดจากในการรับส่งข้อมูลที่ทำหน้าที่ตรวจดูว่าความผิดพลาดที่เกิดขึ้นนั้นคือความผิดพลาดอะไร หรือฟังก์ชั่นตรวจสอบบันทึกเหตุการณ์ต่างๆ ที่เกิดขึ้นในกระบวนการสื่อสาร ภายในช่วงระยะเวลาหนึ่ง ผลิตภัณฑ์ทุกตัวที่ถูกออกแบบให้สามารถสื่อสารด้วยโปรโตคอล MODBUS จะต้องสามารถประมวลผลฟังก์ชั่นเหล่านี้ได้ทุกฟังก์ชั่น ส่วนที่จะมีฟังก์ชั่นหรือความสามารถพิเศษอื่นๆ เพิ่มเติมอย่างไรนั้น ขึ้นอยู่กับผู้ผลิตแต่ละรายที่ต้องการพัฒนาให้ผลิตภัณฑ์ของตนมีจุดเด่นจุดขายเหนือกว่าคู่แข่งอย่างไร

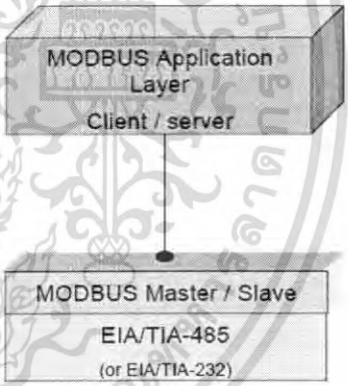
				Function Codes		
				code	Sub Code	(HEX)
Data Access	Bit access	Physical Discrete Inputs	Read Discrete Inputs	02		02
		Internal Bits Or Physical coils	Read Coils	01		01
			Write Single Coil	05		05
			Write Multiple Coils	15		0F
	16 bits access	Physical Input Registers	Read Input Register	04		04
		Internal Registers Or Physical Output Registers	Read Holding Registers	03		03
			Write Single Register	06		06
			Write Multiple Registers	16		10
			Read/Write Multiple Registers	23		17
			Mask Write Register	22		16
			Read FIFO queue	24		18
	File record access		Read File record	20	6	14
			Write File record	21	6	15
	Diagnostics		Read Exception status	07		07
			Diagnostic	08	00-18	
			Get Com event counter	11		0B
			Get Com Event Log	12		0C
			Report Slave ID	17		11
			Read device Identification	43	14	2B
Other			Encapsulated Interface Transport	43		2B

รูปที่2.3.7 ฟังก์ชั่น Public ชนิดต่างๆ

2.3.3 โปรโตคอล MODBUS Serial Line

MODBUS ในช่วงแรก ถูกพัฒนาขึ้นมาใช้งานกับระบบสื่อสารขอมูลแบบอนุกรม ได้แก่ EIA/TIA-232 และ EIA/TIA-485 หรือที่เรียกกันว่า RS-232 และ RS-485 ตามลำดับ สาเหตุเป็นเพราะว่า ในสมัยก่อน ระบบสื่อสารข้อมูลแบบอนุกรมเป็นที่นิยมใช้ในงานอุตสาหกรรม แม้กระทั่งในปัจจุบัน RS-485 ก็ยังเป็นที่นิยมใช้กันอยู่ เพราะต้นทุนต่ำ ใช้สายสัญญาณเพียงแค่ 2 เส้น อีกทั้งยังสามารถรับส่งข้อมูลๆได้ระยะทางไกลถึง 1.2 กิโลเมตร ดังนั้น โครงสร้างของการสื่อสารด้วยโปรโตคอล MODBUS จึงต้องมีตัวกลางที่ทำหน้าที่เชื่อมประสานระหว่างลำดับชั้นบนคือแอพพลิเคชั่นโปรโตคอล กับลำดับชั้นล่าง คือฮาร์ดแวร์ที่ทำหน้าที่ดำเนินการทางกายภาพในการรับส่งข้อมูล ก็คือวงจรสื่อสารข้อมูลแบบอนุกรม ตัวกลางที่ว่านี้ เรียกว่า โปรโตคอล MODBUS Serial

Layer	ISO/OSI Model	
7	Application	MODBUS Application Protocol
6	Presentation	Empty
5	Session	Empty
4	Transport	Empty
3	Network	Empty
2	Data Link	MODBUS Serial Line Protocol
1	Physical	EIA/TIA-485 (or EIA/TIA-232)



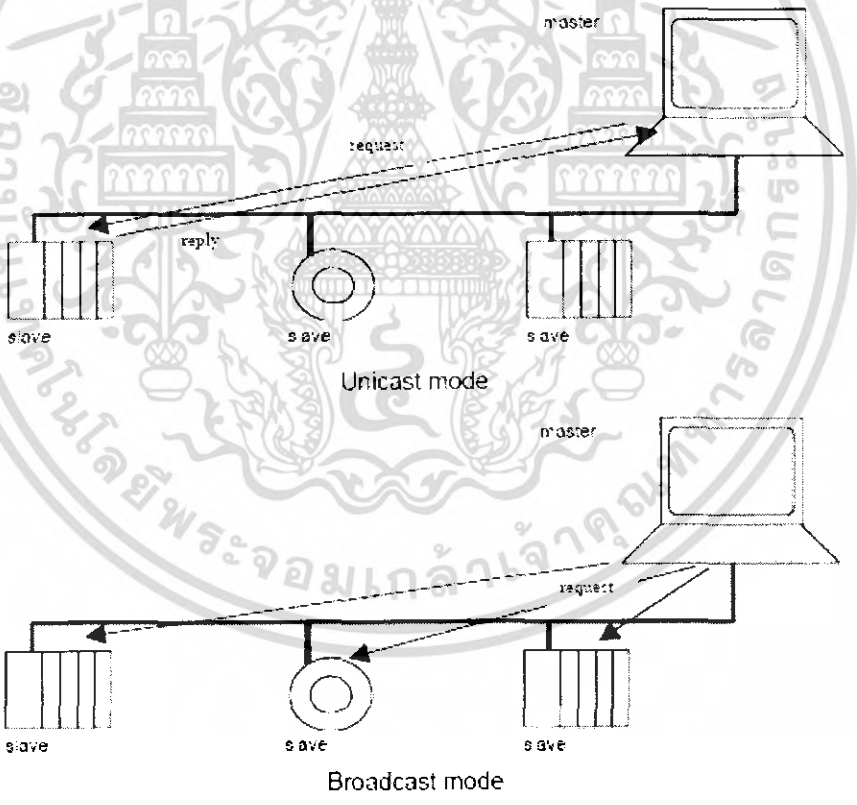
รูปที่ 2.3.8 ลำดับชั้นของ MODBUS Serial Line

Line ใช้เชื่อมประสานกับระบบสื่อสารข้อมูลแบบอนุกรม จัดอยู่ในลำดับชั้นที่ 2 ของแบบจำลอง OSI คือ Data Link Layer ทำหน้าที่แปลงข้อมูลจากให้อยู่ในสภาพพร้อมส่ง และแปลงกลับข้อมูลที่รับเข้ามาเพื่อส่งไปให้ลำดับชั้นบน ซึ่งก็คือการแปลงเฟรมข้อมูลระหว่าง PDU กับ ADU นั่นเอง โปรโตคอล MODBUS Serial Line มีรูปแบบการทำงานเป็นแบบ มาสเตอร์-สเลฟ โดยที่ระบบบัสข้อมูลหนึ่งๆ จะมี Master เพียง 1 ตัว ทำหน้าที่ควบคุมจัดการการสื่อสารของสมาชิกทุกตัวภายในระบบบัสข้อมูล ที่เรียกว่าสเลฟ สามารถมีจำนวนสเลฟได้มากสุดไม่เกิน 247 ตัว ระบบสื่อสารแบบนี้

จะเริ่มต้นด้วยมาสเตอร์เสมอ สเลฟจะไม่สามารถเริ่มต้นการสื่อสารได้ และจะส่งข้อมูลออกมาที่บัสข้อมูลก็ต่อเมื่อได้รับการติดต่อร้องขอจากมาสเตอร์เท่านั้น และสเลฟแต่ละตัวก็ไม่สามารถสื่อสารกันเองได้ ในขณะเวลาหนึ่ง มาสเตอร์จะสามารถควบคุมการสื่อสารได้เพียง 1 กระบวนการเท่านั้น ไม่สามารถควบคุมการสื่อสารพร้อมๆ กันหลายกระบวนการได้ ดังนั้น การสื่อสารภายในบัสข้อมูลจึงเป็นแบบครั้งต่อครั้ง

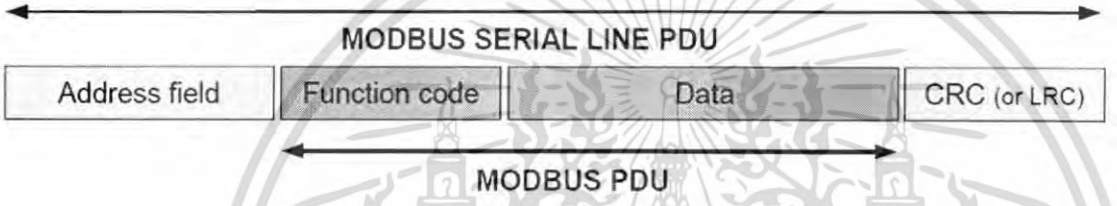
มาสเตอร์สามารถติดต่อกับสเลฟได้ด้วยโหมดการติดต่อ 2 โหมด คือ

1. โหมด unicast มาสเตอร์เริ่มต้นด้วยการประกาศหมายเลขของสเลฟตัวที่ต้องการติดต่อด้วยลงไปในบัส จากนั้น สเลฟตัวที่ถูกระบุถึง จะส่งเฟรมข้อมูล reply เพื่อตอบกลับไปยังมาสเตอร์ ดังนั้นในโหมดนี้ การสื่อสาร 1 ครั้ง จึงประกอบด้วยเฟรมข้อมูล 2 เฟรม คือเฟรมแรกที่มาสเตอร์ส่งออกมาและเฟรมข้อมูลที่สเลฟตอบกลับไปยังมาสเตอร์



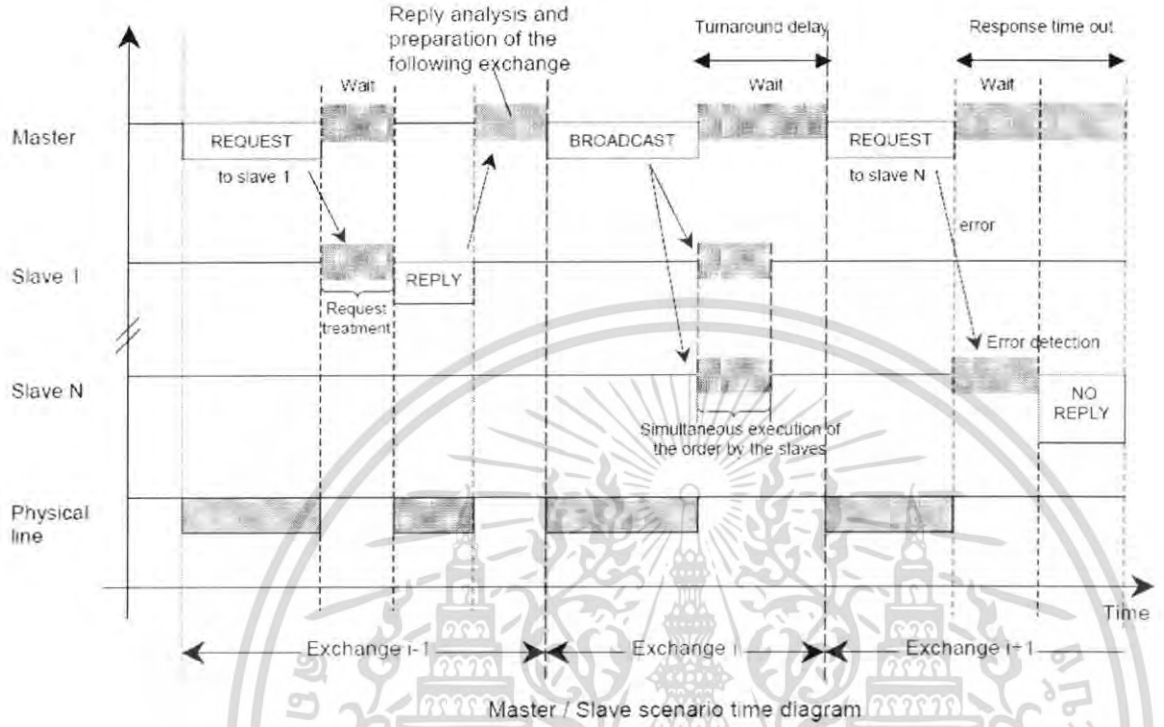
รูปที่ 2.3.9 การติดต่อสเลฟแบบ unicast และ broadcast

2. โหมด broadcast เป็นการติดต่อจากมาสเตอร์ไปที่สเลฟทุกตัวในบัส ก็คือการติดต่อแบบกระจายสัญญาณนั่นเอง สเลฟทุกตัวจะรับข้อมูลจากมาสเตอร์ แต่ไม่มีการตอบกลับ ดังนั้น โหมดนี้จึงถูกใช้ในการเขียนข้อมูลบางประเภทลงไปในสเลฟทุกตัว เช่น ค่าพารามิเตอร์เริ่มต้นในการทำงานของสเลฟ หรือ คำสั่งกำหนดค่าเริ่มต้นเมื่อระบบเริ่มทำงาน (Initialization Command) หมายเลขแอดเดรสที่ใช้ในเฟรมข้อมูลในโหมด broadcast คือ หมายเลข 0 ส่วนหมายเลขแอดเดรสตั้งแต่ 248 จนถึง 255 ถูกสงวนเอาไว้ไม่ให้ใช้งาน คาดว่าคงเผื่อเอาไว้สำหรับการปรับปรุงพัฒนาโปรโตคอลในอนาคต



รูปที่ 2.3.10 เฟรมข้อมูลของ MODBUS Serial Line

เฟรมข้อมูลของโปรโตคอล MODBUS Serial Line มีองค์ประกอบดังแสดงในรูปที่ 2.3.10 จะสังเกตเห็นได้ว่า MODBUS PDU ก็คือ PDU ที่กล่าวถึงในช่วงแรก ไม่ว่าจะนำไปใช้กับระบบสื่อสารแบบอนุกรม แบบ TCP/IP หรือแบบอื่นๆ ก็ไม่แตกต่างกัน ส่วนที่แตกต่างก็คือ ADU ซึ่งในที่นี้ สำหรับ MODBUS Serial Line เรียกว่า MODBUS Serial Line PDU เพิ่มเติมส่วนการระบุแอดเดรสของสเลฟและรหัสตรวจสอบความถูกต้องของข้อมูลแบบ CRC หรือ LRC



รูปที่ 2.3.11 แผนผังเวลาแสดงตัวอย่างการรับส่งเฟรมข้อมูลระหว่างมาสเตอร์กับสเลฟ

รูปที่ 2.3.11 แสดงแผนผังเวลาในการเปลี่ยนแปลงระดับสัญญาณบนบัสข้อมูล เริ่มต้นด้วยมาสเตอร์ส่งเฟรมข้อมูลออกมาที่บัสข้อมูลเพื่อติดต่อกับสเลฟหมายเลข 1 ด้วยโหมด unicast จากนั้นมาสเตอร์จะรอให้สเลฟแต่ละตัวทำการเปรียบเทียบแอดเดรสของตัวเองว่าตรงกับแอดเดรสที่ระบุในเฟรมข้อมูล คือ 1 หรือไม่ ซึ่งในที่สุด สเลฟหมายเลข 1 จะรับรู้ว่า มาสเตอร์ส่งเฟรมข้อมูลมาให้ตัวเอง และรับเอาข้อมูลภายในเฟรมนั้นเข้ามาประมวลผลตามหมายเลขฟังก์ชั่นและข้อมูลประกอบภายในเฟรมข้อมูล เมื่อเสร็จเรียบร้อย จึงส่งเฟรมข้อมูลตอบกลับลงมาในบัสข้อมูลเพื่อให้มาสเตอร์อ่านไปอีกที จากนั้น มาสเตอร์ส่งเฟรมข้อมูลออกมาให้กับสเลฟทุกตัว ด้วยโหมด broadcast ทำให้สเลฟทุกตัวรับเฟรมข้อมูลนั้นเอาไว้ โดยไม่มีการตอบกลับแต่อย่างใด ปิดท้ายด้วยการส่งเฟรมข้อมูลจากมาสเตอร์มาที่สเลฟหมายเลข N แต่คราวนี้ เกิดปัญหาในขณะที่สเลฟหมายเลข N กำลังส่งเฟรมตอบกลับมายังมาสเตอร์ ทำให้มาสเตอร์รอรับข้อมูลตอบกลับจนหมดเวลา ก็จะรู้ว่าเกิดความผิดพลาดในการสื่อสาร

2.3.4 โหมดการรับส่งข้อมูลอนุกรม

การรับส่งข้อมูลอนุกรมด้วยโปรโตคอล MODBUS สามารถเลือกได้ 2 โหมด คือ โหมด RTU และโหมด ASCII ซึ่งทั้ง 2 โหมดนี้ มีความแตกต่างกันที่การกำหนดรูปแบบของชุดข้อมูลภายในเฟรม จะเลือกโหมดใดก็ได้ แต่มีเงื่อนไขว่า อุปกรณ์ทุกตัวที่ต่อร่วมอยู่ในบัสหรือเครือข่ายเดียวกัน จะต้องถูกปรับตั้งให้เลือกใช้โหมดเดียวกันทั้งหมด

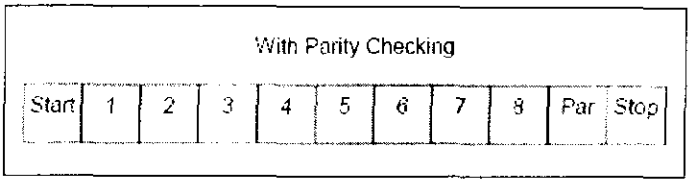
1. โหมด RTU

Slave Address	Function Code	Data	CRC
1 byte	1 byte	0 up to 252 byte(s)	2 bytes CRC Low, CRC Hi

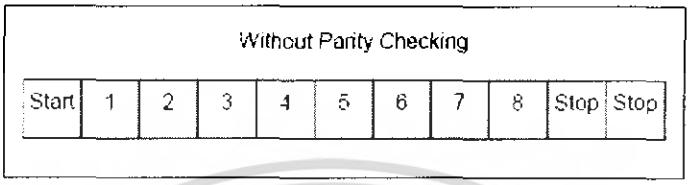
RTU Message Frame

รูปที่ 2.3.12 เฟรมข้อมูลในโหมด RTU

เฟรมข้อมูลในโหมด RTU ประกอบด้วยข้อมูลแสดงตำแหน่งแอดเดรสของสเลฟ 1 ไบต์ , หมายเลขฟังก์ชั่น 1 ไบต์ , ข้อมูลที่ทำการรับส่งจำนวนมากสุดไม่เกิน 252 ไบต์ , และรหัสตรวจสอบความถูกต้องของข้อมูลแบบ CRC (Cyclical Redundancy Checking) ขนาด 2 ไบต์ ค่าCRC นี้ เป็นค่าที่คำนวณมาจากข้อมูลทุกไบต์ ไม่รวมบิต start , stop , และ parity check สำหรับสูตรที่ใช้ในการคำนวณนั้น ผู้เขียนจะขอไม่กล่าวถึง เพียงแต่ให้ทราบว่ามีประโยชน์อย่างไร โดยที่สเลฟตัวที่ส่งข้อมูลออกมาจะสร้างรหัส CRC แล้วส่งตามท้ายไบต์ข้อมูลออกมา หลังจากนั้น เมื่อมาสเตอร์ได้รับเฟรมข้อมูล และถอดข้อมูลออกจากเฟรมแล้ว จะทำการคำนวณค่า CRC ตามสูตรเดียวกันกับสเลฟ เพื่อทำการเปรียบเทียบค่า CRC ทั้ง 2 ค่า ว่าตรงกันหรือไม่ หากไม่ตรงกัน แสดงว่า เกิดความผิดพลาดในการรับส่งข้อมูล



Bit Sequence in RTU mode

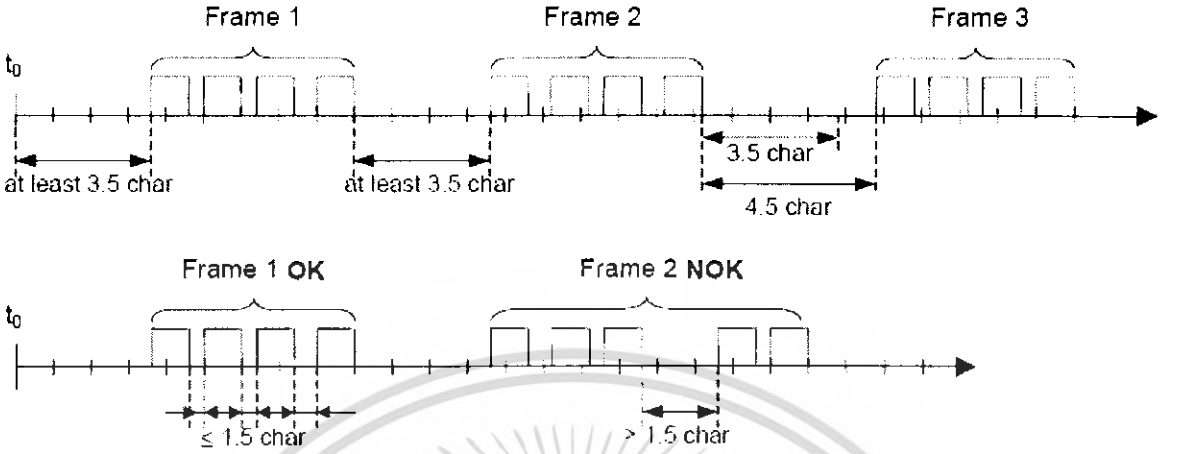


Bit Sequence in RTU mode (specific case of No Parity)

รูปที่ 2.3.13 การส่งข้อมูล 1 ไบต์ในโหมด RTU

ในโหมด RTU การส่งข้อมูล 1 ไบต์ ไม่ว่าจะเป็นข้อมูลส่วนใดภายในเฟรม จะต้องทำการส่งบิตข้อมูลรวม 11 บิต คือ บิตเริ่มต้น (Start) 1 บิต , บิตข้อมูล 8 บิตซึ่งก็คือ 1 ไบต์ , บิตตรวจสอบ parity ของข้อมูล 1 บิต , และบิตหยุด (Stop) 1 บิต หรือหากเลือกแบบไม่มีบิต parity ก็จะเป็นบิต stop แทน รวม 2 บิต สำหรับการกำหนดให้มีบิต parity นั้น สามารถเลือกเป็นแบบคู่ (*even parity*) หรือคี่ (*odd parity*) ก็ได้ และหากต้องการออกแบบให้สอดคล้องกับอุปกรณ์ที่มีใช้กันทั่วไปมากที่สุด ควรเลือกแบบคู่ โดยที่สามารถปรับเปลี่ยนเป็นแบบคี่ หรือไม่มีการตรวจสอบ parity (no parity) ได้ด้วย

รูปที่ 2.3.13 แสดงช่วงเวลาที่เหมาะสมในกระบวนการส่งเฟรมข้อมูลออกมาในบัสข้อมูล เมื่อส่งเฟรมข้อมูลออกไป 1 เฟรมแล้ว จะต้องรอเวลาอย่างน้อยเท่ากับเวลาที่ใช้ส่งข้อมูลจำนวน 3.5 ตัวอักษร จึงจะสามารถส่งเฟรมข้อมูลต่อไปได้ และภายในเฟรมแต่ละเฟรม ซึ่งประกอบด้วยชุดบิตข้อมูลจำนวนหลายชุด ก็จะอยู่ห่างกันไม่เกิน 1.5 ตัวอักษร วัตถุประสงค์ในการกำหนดช่วงเวลาระหว่างเฟรมข้อมูล และชุดบิตข้อมูลภายในเฟรม ก็เพื่อให้อุปกรณ์ต่างๆ ไม่ว่าจะเป็นมาสเตอร์หรือสเลฟ สามารถรับรู้ถึงจุดเริ่มต้นและจุดสิ้นสุดของเฟรมข้อมูลแต่ละเฟรมได้ และสามารถตรวจสอบได้ว่า การรับส่งข้อมูลในขณะนั้น เกิดความผิดพลาดขึ้นมาหรือไม่ โดยตรวจสอบกับช่วงระยะห่างของเวลาที่ควรจะเป็น กับค่าที่วัดได้จริง



รูปที่ 2.3.14 การกำหนดช่วงระยะห่างทางเวลาระหว่างเฟรมข้อมูลแต่ละเฟรม และระหว่างชุดข้อมูลแต่ละชุดภายในเฟรมเดียวกัน

2. โหมด ASCII

การรับส่งข้อมูลในโหมด ASCII มีความแตกต่างจากโหมด RTU ตรงที่ ในโหมด RTU ข้อมูลที่จะส่งขนาด 1 ไบต์ นำมารวมกับบิตประกอบต่างๆ ก็สามารถส่งออกไปได้เลย แต่สำหรับโหมด ASCII จะมองข้อมูล 1 ไบต์นั้นออกเป็นตัวอักษร 2 ตัว เช่น ค่า 0x5B ซึ่งเป็นเลขฐานสิบหก ก็จะถูกมองเป็นตัวอักษร '5' และตัวอักษร 'B' จากนั้น ก็จะทำการค้นหาค่ารหัส ASCII ของตัวอักษรทั้ง 2 ตัวนั้น ซึ่งได้แก่ 0x35 สำหรับ '5' และ 0x42 สำหรับ 'B' แล้วทำการส่งรหัส ASCII ทั้ง 2 ค่านี้ออกไป ซึ่งจะได้ผลเท่ากับการส่งค่า 0x5B ซึ่งเป็นข้อมูลขนาด 1 ไบต์ ในโหมด RTU

จะเห็นได้ว่าการส่งข้อมูลในโหมด ASCII จะต้องทำงานมากกว่าการส่งข้อมูลในโหมด RTU ซึ่งทำให้อัตราเร็วในการสื่อสารมีค่าต่ำกว่า โหมด ASCII สาเหตุที่เป็นแบบนี้ก็เพราะว่า โหมด ASCII ได้ถูกออกแบบมาสำหรับอุปกรณ์ที่ไม่มีความสามารถในการกำหนดช่วงระยะห่างทางเวลาในการส่งเฟรมข้อมูล อย่างเช่นในโหมด RTU ที่อุปกรณ์

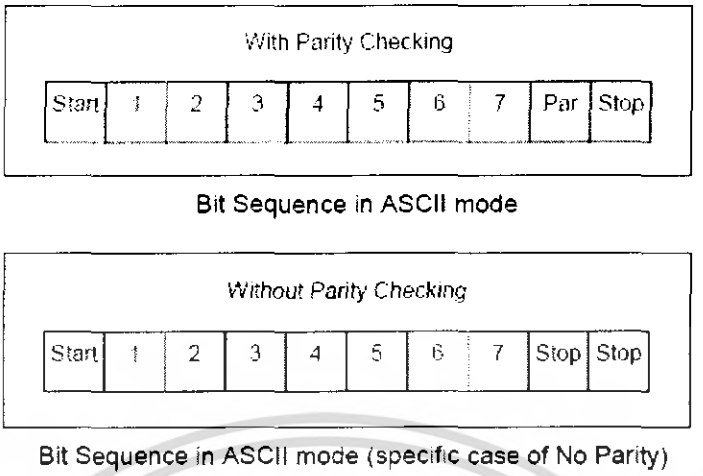
Start	Address	Function	Data	LRC	End
1 char	2 chars	2 chars	0 up to 2x252 char(s)	2 chars	2 chars CR LF

ASCII Message Frame

รูปที่ 2.3.15 เฟรมข้อมูลในโหมด ASCII

สามารถกำหนดได้ว่าจะส่งเฟรมข้อมูลแต่ละเฟรมออกมาด้วยเวลาห่างกันเท่าใด และอุปกรณ์ที่รอรับข้อมูลก็ต้องสามารถตรวจจับแยกแยะได้ด้วยว่าเฟรมข้อมูลแต่ละเฟรมที่รับเข้ามานั้น มีระยะเวลาห่างกันภายในช่วงเวลาที่กำหนดหรือไม่ เพื่อทำให้สามารถตรวจสอบหาจุดเริ่มต้นและจุดสิ้นสุดของเฟรมข้อมูลแต่ละเฟรมได้ แต่ในความเป็นจริง ยังมีอุปกรณ์อีกหลายชนิด ที่ไม่มีความสามารถพิเศษแบบนี้ จึงต้องใช้วิธีอื่นที่จะช่วยให้สามารถรับรู้จุดเริ่มต้นและจุดสิ้นสุดของเฟรมข้อมูลได้ ได้แก่โหมด ASCII ซึ่งในโหมดนี้ จะเริ่มต้นเฟรมข้อมูลด้วยการส่งรหัส ASCII ที่กำหนดให้หมายถึงจุดเริ่มต้น คือ 0x3A ซึ่งตรงกับตัวอักษร ':' ตามด้วยแอดเดรสของสเลฟ, หมายเลขฟังก์ชั่น, ข้อมูล, รหัสตรวจสอบ LRC และรหัส ASCII 2 ตัว ที่กำหนดให้หมายถึงจุดสิ้นสุด คือ รหัส 0x0D และ 0x0A คือรหัส CR (Carriage Return) และ LF (Line Feed) ตามลำดับ (ดูรูปที่ 2.3.5) โดยขณะที่บัสข้อมูลว่างจากการรับส่งข้อมูล อุปกรณ์ทุกตัวจะคอยตรวจจับข้อมูลในบัสว่ามีการส่งรหัส ASCIIของ ':' ออกมาหรือไม่ หรือมี ก็จะรับรู้ว่าขณะนี้ ได้มีการเริ่มต้นส่งเฟรมข้อมูลออกมาแล้ว ก็จะเข้าสู่กระบวนการรับข้อมูลต่อไป

รูปที่ 2.3.16 แสดงชุดของบิตข้อมูลที่ส่งทั้งหมดในการส่งข้อมูลแต่ละตัวอักษร จะเห็นได้ว่าหน่วยของชุดข้อมูลในโหมด ASCII คือตัวอักษร ไม่เหมือนในโหมด RTU ที่มีหน่วยเป็นไบต์ เพราะโหมด ASCII เป็นการส่งข้อมูลในรูปแบบของรหัส ASCII ของตัวอักษร ซึ่งสามารถกำหนดได้ด้วยบิตข้อมูลจำนวน 7 บิต ไม่ต้องใช้ถึง 8 บิต ดังนั้น บิตที่ต้องส่งต่อการส่งรหัส ASCII 1 ตัวได้แก่ บิต Start 1 บิต, บิตข้อมูลรหัส ASCII 7 บิต, บิตตรวจสอบ parity 1 บิต, และบิต Stop 1 บิต รวมทั้งหมดเท่ากับ 10 บิต และเช่นเดียวกับโหมด RTU คือ สามารถเลือกประเภทของบิตตรวจสอบ parity ได้ ว่าเป็นแบบคู่, คี่, หรือไม่มีบิตตรวจสอบ ซึ่งจะเปลี่ยนเป็นบิต Stop แทน



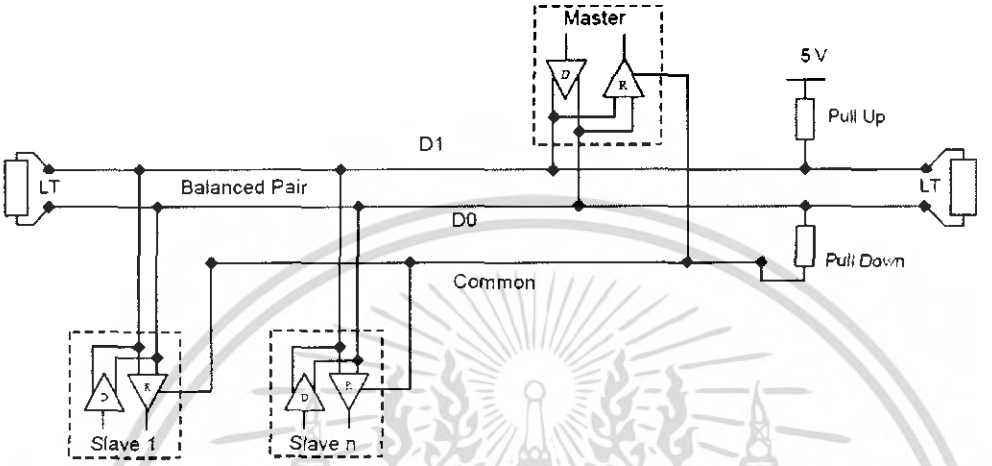
รูปที่ 2.3.16 การส่งข้อมูล 1 ตัวอักษรในโหมด ASCII

เนื่องจากการส่งข้อมูลในโหมด ASCII เป็นการแปลงข้อมูลจากเลขฐานสิบหก เป็นรหัส ASCII ของตัวอักษรที่แสดงค่าเลขฐานสิบหก ดังนั้น รหัส ASCII ที่ปรากฏในบัสข้อมูล นอกเหนือไปจากรหัสเริ่มต้นและรหัสสิ้นสุดแล้ว จะเป็นรหัส ASCII ของ ตัวอักษรตั้งแต่ '0' ถึง '9' และ 'A' ถึง 'F' เท่านั้น ถึงแม้โหมด ASCII จะไม่ต้องกำหนดช่วงระยะห่างทางเวลาของเฟรมข้อมูลแต่ละเฟรม แต่อุปกรณ์ยังต้องสามารถตรวจจับช่วงระยะห่างทางเวลาระหว่างการส่งข้อมูลรหัส ASCII แต่ละตัวได้ ซึ่งหากเว้นช่วงห่างกันนานเกินไป แสดงว่าเกิดความผิดพลาดในการสื่อสาร โดยปกติ จะกำหนดค่าเวลานี้ไว้ที่ 1 วินาที เรียกค่าเวลานี้ว่า time-out period หากเปรียบเทียบระหว่างเฟรมข้อมูลในโหมด ASCII กับโหมด RTU จะพบว่า การส่งข้อมูลในโหมด ASCII นั้น หากต้องการส่งไบต์ข้อมูลให้ได้เท่ากับโหมด RTU จะต้องส่งข้อมูลรหัส ASCI ออกไปเป็นจำนวน 2 เท่าของจำนวนไบต์ข้อมูล เช่นในโหมด RTU เฟรมข้อมูล 1 เฟรม สามารถส่งข้อมูลได้มากสุด 252 ไบต์ ซึ่งหากเป็นโหมด ASCII จะต้องส่งข้อมูลตัวอักษรออกไปทั้งหมด 2x252 เท่ากับ 504 ตัวอักษร และเพื่อให้มาตรฐานขนาดของเฟรมข้อมูลของทั้ง 2 โหมดมีขนาดเท่ากัน จึงกำหนดให้ค่า 504 เป็นค่าจำนวนตัวอักษรมากสุดในการส่งเฟรมข้อมูลด้วยโหมด ASCII

2.3.5 รูปแบบการเชื่อมต่อบัสข้อมูล

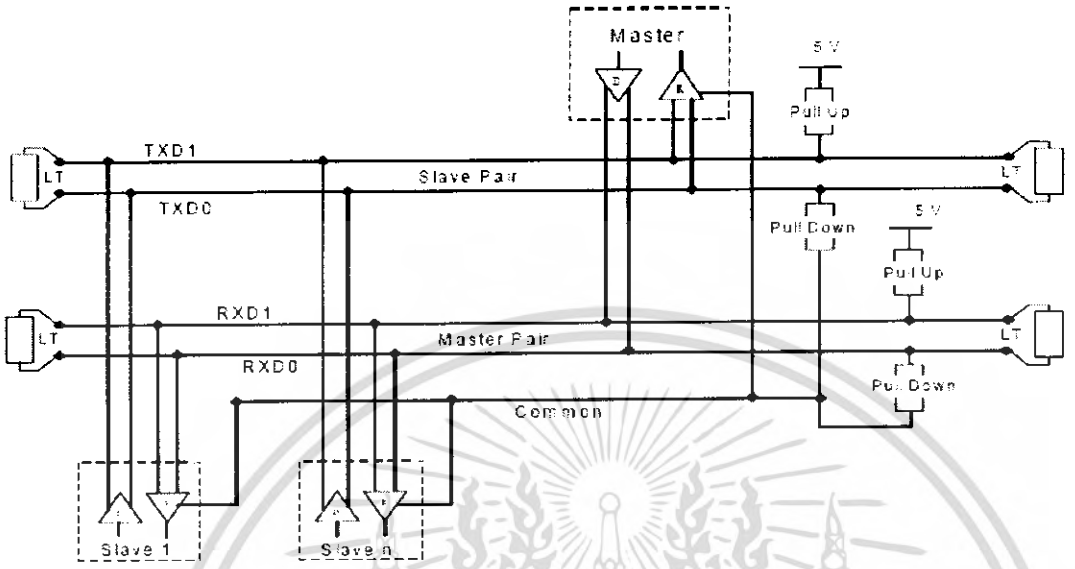
การเชื่อมต่อหรือเดินสายไฟฟ้าสำหรับใช้ทำหน้าที่เป็นบัสรับส่งสัญญาณข้อมูลภายในเครือข่าย สามารถกระทำได้ 2 วิธี คือ การเดินสายสัญญาณแบบ 2 เส้น และแบบ 4 เส้น โดยใช้มาตรฐาน EIA/TIA-485 เป็นตัวรับส่งข้อมูลในลำดับชั้นกายภาพ (Physical Layer) หรือระดับฮาร์ดแวร์ ซึ่งมีข้อดี

คือ สามารถใช้รับส่งข้อมูลได้ระยะทางไกลถึง 1.2 กิโลเมตร โดยอาศัยหลักการขับกระแสไฟฟ้าระหว่างคู่สายสัญญาณ ที่มีแรงดันแตกต่างกันตามค่าสถานะของข้อมูล ที่ต้องการส่งการเดินสายสัญญาณแบบ 2 เส้น ได้รับความนิยมมากกว่า เนื่องจาก ใช้สายสัญญาณที่มีจำนวนเส้น



รูปที่ 2.3.17 IEA/TIA-485

ตัวนำไฟฟ้าภายในน้อยกว่า ทำให้ประหยัดปริมาตรภายในรางเดินสายไฟ และการเชื่อมต่อสายสัญญาณเข้ากับอุปกรณ์ก็ทำได้ง่ายกว่า เพราะมีสายสัญญาณที่ต้องเชื่อมต่อเพียงแค่ 2 เส้นเท่านั้น ทำหน้าที่ทั้งส่งและรับข้อมูล รูปที่ 18 แสดงการเดินสายสัญญาณระหว่างมาสเตอร์และสเลฟ โดยมีตัวความต้านทานปิดหัวท้ายของสายสัญญาณ เรียกว่า LT (Line Termination) ทำหน้าที่ป้องกันผลกระทบการคลื่นสัญญาณสะท้อนภายในคู่สายสัญญาณ ส่วนตัวความต้านทานอีก 2 ตัว คือ Pull Up และ Pull Down ทำหน้าที่ช่วยดึงระดับของสถานะของสายสัญญาณให้มีค่าที่ควรจะเป็น ในขณะที่บัสว่าง เพราะมิฉะนั้นแล้ว สายสัญญาณอาจมีสถานะเป็น High Impedance ซึ่งจะแปลความหมายในการสื่อสารไม่ได้ และจะทำให้ระบบไม่ทำงาน หรือทำงานผิดพลาด



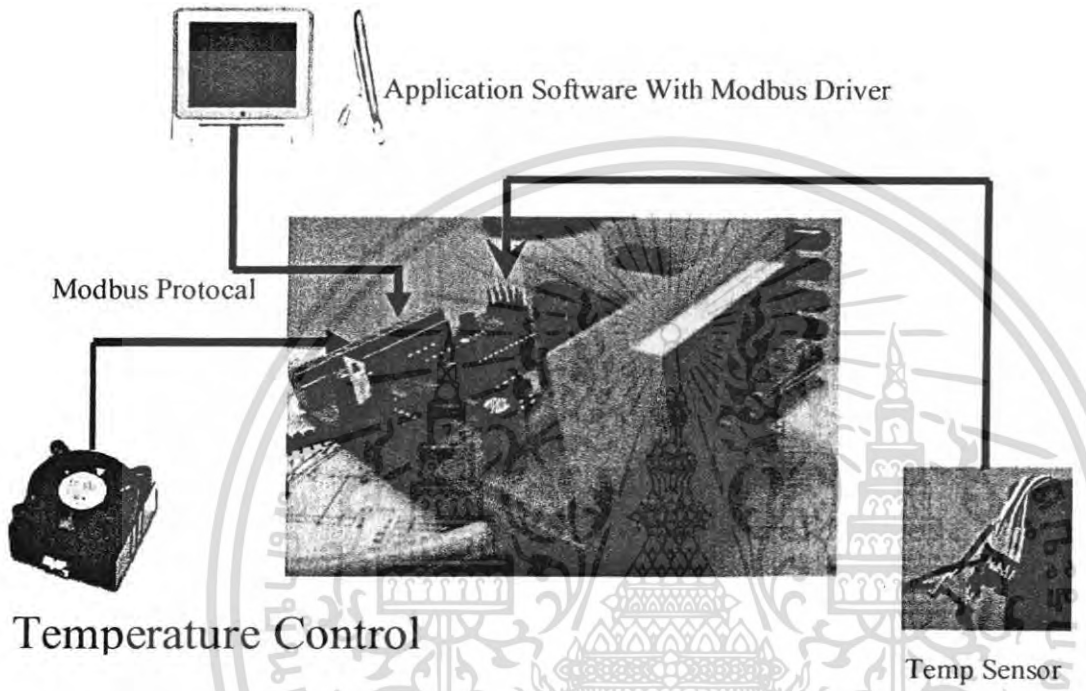
รูปที่ 2.3.18 IEA/TIA-485

จะเห็นได้ว่า นอกจากสายสัญญาณ D0 และ D1 แล้ว การเดินสายสัญญาณ ยังต้องมีการเดินสาย common อีก 1 เส้น เพื่อใช้เป็นจุดอ้างอิงระดับสัญญาณของอุปกรณ์ทุกตัวภายในระบบบัสข้อมูลเดียวกัน ซึ่งการเดินสายสัญญาณแบบ 4 สาย ก็จำเป็นต้องใช้สาย common ด้วยเช่นกัน ความแตกต่างก็คือ การเดินสายสัญญาณแบบ 4 เส้น ใช้สายสัญญาณรับส่งข้อมูล 2 คู่แยกกัน คู่หนึ่งสำหรับส่งข้อมูล อีกคู่สำหรับรับข้อมูล

บทสรุป MODBUS PROTOCOL

จากที่ได้กล่าวมาทั้งหมด คงสามารถมองเห็นภาพการทำงานของกระบวนการรับส่งข้อมูลด้วยโปรโตคอล MODBUS ได้ดีพอสมควร ซึ่งจะเห็นได้ว่า MODBUS เป็นโปรโคคอลที่ไม่ซับซ้อน มีความยืดหยุ่นสูง โดยสามารถนำไปใช้กับระบบสื่อสารข้อมูลได้หลายชนิด ซึ่งก็คือคุณสมบัติของความเป็นระบบเปิด และเปิดโอกาสให้ผู้ผลิตผลิตภัณฑ์ที่สามารถสื่อสารด้วย MODBUS สามารถพัฒนาฟังก์ชั่นพิเศษต่างๆ ขึ้นมา เพื่อเป็นจุดขายที่แตกต่างไปจากผลิตภัณฑ์ของคู่แข่งได้อีกด้วย และด้วยจุดเด่นต่างเหล่านี้ ทำให้ MODBUS ยังคงเป็นโปรโตคอลยอดนิยมที่ยังไม่ล้าสมัย ดังเช่นโปรโตคอลในภาคอุตสาหกรรมบางตัว

บทที่ 3
การออกแบบโครงงาน



รูปที่ 3.1 แสดงถึงโครงสร้างของระบบควบคุมอุณหภูมิแบบหลายจุด

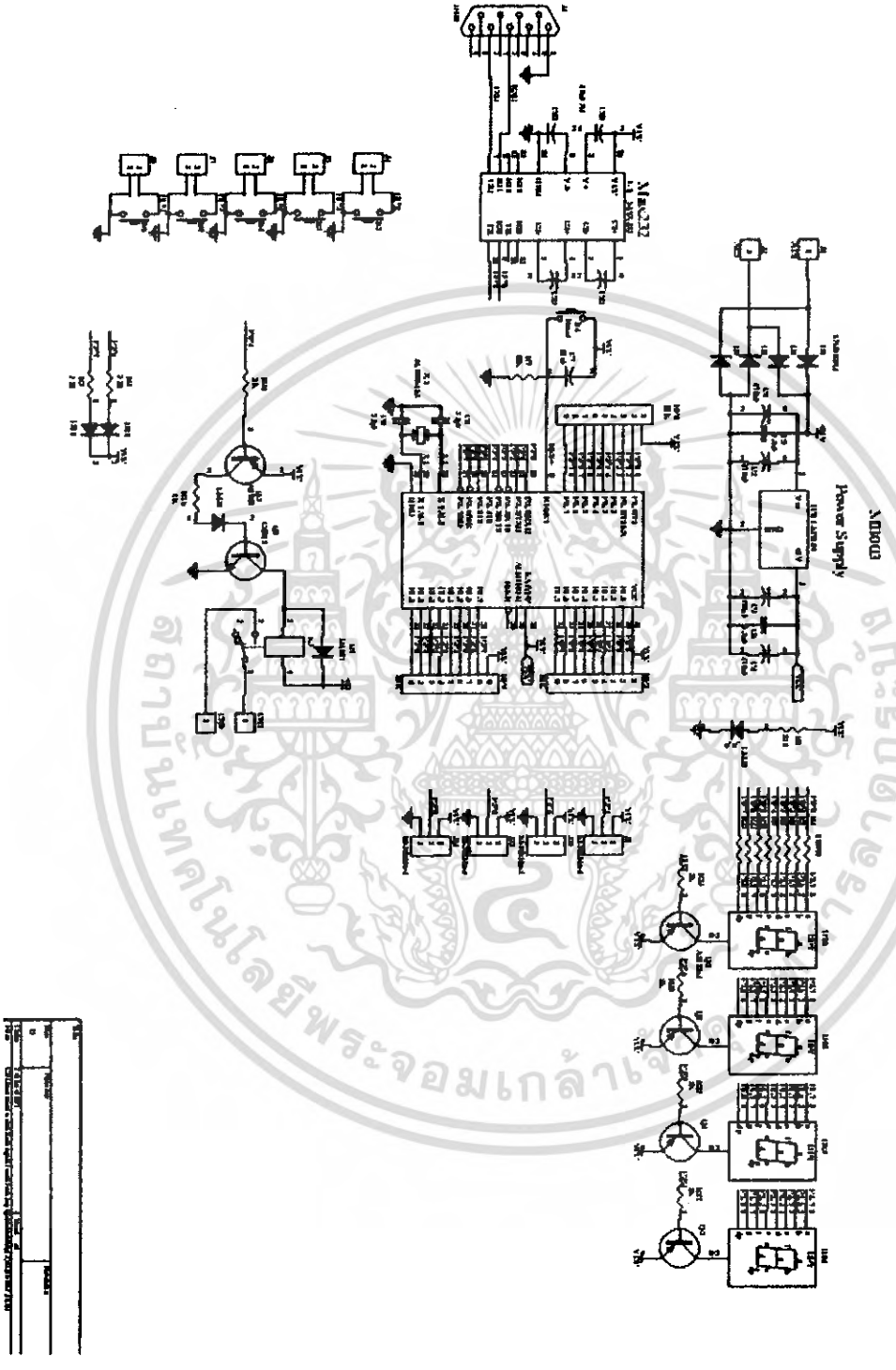
จุดประสงค์หลักของการควบคุม คือ ต้องการให้เอาต์พุตของระบบลู่เข้าสู่ค่าอ้างอิง (setpoint) และไม่ส่งผลต่อค่าพุ่งเกิน (overshoot) และค่าผิดพลาดที่สภาวะคงตัวสูงเกินไป โดยในปริญญานิพนธ์นี้ ต้องการควบคุมอุปกรณ์ตัวที่จ่ายอุณหภูมิ เพื่อให้ได้อุณหภูมิบริเวณที่ต้องการควบคุมอุณหภูมิคงที่ตามค่าอ้างอิง การออกแบบระบบควบคุมอุณหภูมิแบบหลายจุดมีดังนี้

การออกแบบฮาร์ดแวร์ และ ซอฟต์แวร์

3.1 ส่วนควบคุม

ใช้ไมโครคอนโทรลเลอร์ตระกูล F^2MC-8L เบอร์ MB89N202 ของบริษัท Fujitsu เนื่องจากเป็นไมโครคอนโทรลเลอร์ที่มีประสิทธิภาพสูงและใช้งานแพร่หลายให้วงการอุตสาหกรรม โดยวงจรของระบบควบคุมอุณหภูมิแบบหลายจุดแสดงในรูปที่ 3.2 ลายวงจรแสดงในรูปที่ 3.3 ภาพของบอร์ดจริงแสดงในรูป 3.4

โดยระบบควบคุมอุณหภูมิจะใช้ระบบวงปิด (Closed loop control) แบบ PID ซึ่งสามารถกำหนดค่าผ่านระบบคอมพิวเตอร์ โดยมีความละเอียดในการวัด ±0.5 องศาเซลเซียส แสดงผลด้วยเซเว่นเซ็กเมนต์พร้อมหลอดไปแสดงผลการทำงาน สามารถทำงานแบบตัวเดี่ยว (stand alone) และสามารถเชื่อมต่อแบบระบบเน็ตเวิร์ค โดยสามารถควบคุมและแสดงผลได้ผ่านคอมพิวเตอร์



รูปที่3.2 แสดงวงจรของระบบควบคุมอุณหภูมิแบบหลายจุด

3.2 ส่วนติดต่อพอร์ตอนุกรม

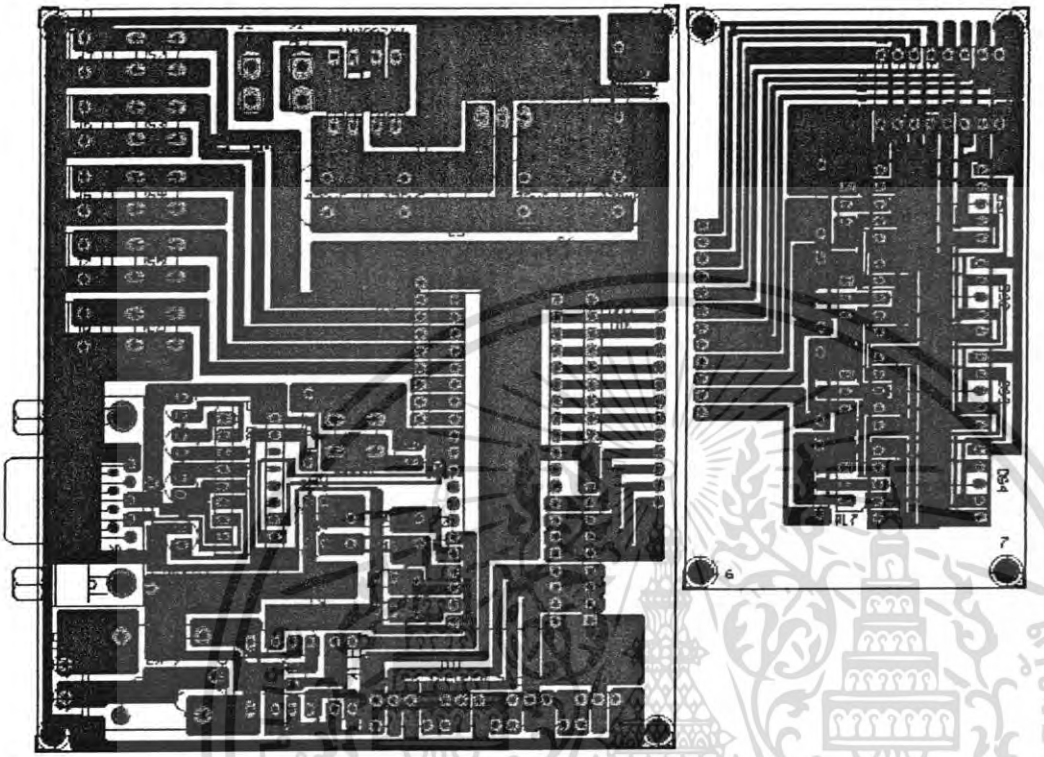
พอร์ตสื่อสารข้อมูลแบบอนุกรม เป็นพอร์ตภายในของไมโครคอนโทรลเลอร์ ซึ่งติดต่อได้ด้วยซอฟต์แวร์โดยใช้ไอซี max232 เป็นการสื่อสารผ่านพอร์ต RS232 โดยใช้ MODBUS PROTOCOL โดยในปริญญานิพนธ์นี้จะใช้ MODBUS RTU ในการติดต่อสื่อสารระหว่างบอร์ดควบคุมอุณหภูมิกับเครื่องคอมพิวเตอร์ การทำงานของพอร์ตอนุกรมจะใช้การทำงานโดยการอินเทอรัปท์ผ่านพอร์ตอนุกรม โดยปกติจะบอร์ดคอนโทรลเลอร์จะทำการอ่านค่าอุณหภูมิแสดงผลและควบคุมการจ่ายอุณหภูมิ เมื่อมีการอินเทอรัปท์ผ่านพอร์ตอนุกรม ก็จะกระโดดไปทำงานในส่วนอินเทอรัปท์นี้ และหยุดการทำงานเมื่อจบกระบวนการทำงาน ไปทำงานในโปรแกรมหลักต่อไป

3.3 เซ็นเซอร์

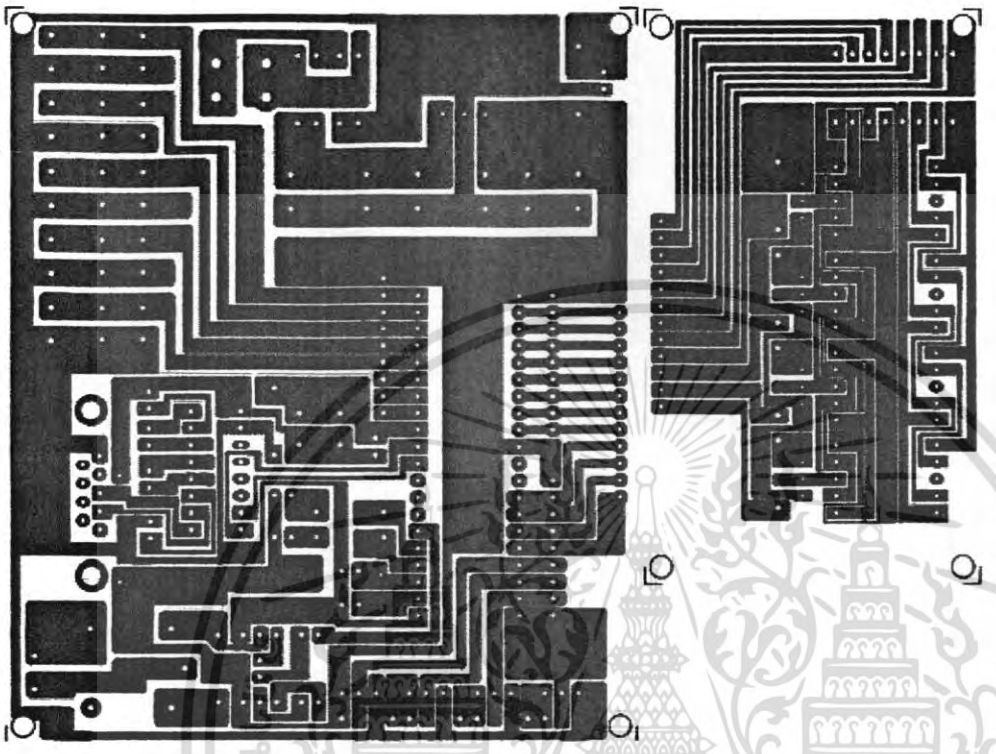
ใช้การต่อแบบ 1 wire bus (DS1820) โดยต่อสายสัญญาณทั้งหมดเข้าที่ขาเพียงขาเดียว จะต้องมีการพูลอัพสายสัญญาณด้วยตัวต้านทานประมาณ 10 กิโลโอห์ม การจ่ายไฟ 5 โวลต์ จะเป็นการจ่ายมาจากจุดเดียวกันหรือคนล่ะจุดก็ได้

ในการติดตั้งเซนเซอร์ จะต้องมีการติดตั้งให้เหมาะสมกับสภาพแวดล้อมนั้นๆ เพื่อให้อุณหภูมิมีค่าใกล้เคียงกันในแต่ล่ะจุดและทั่วถึง

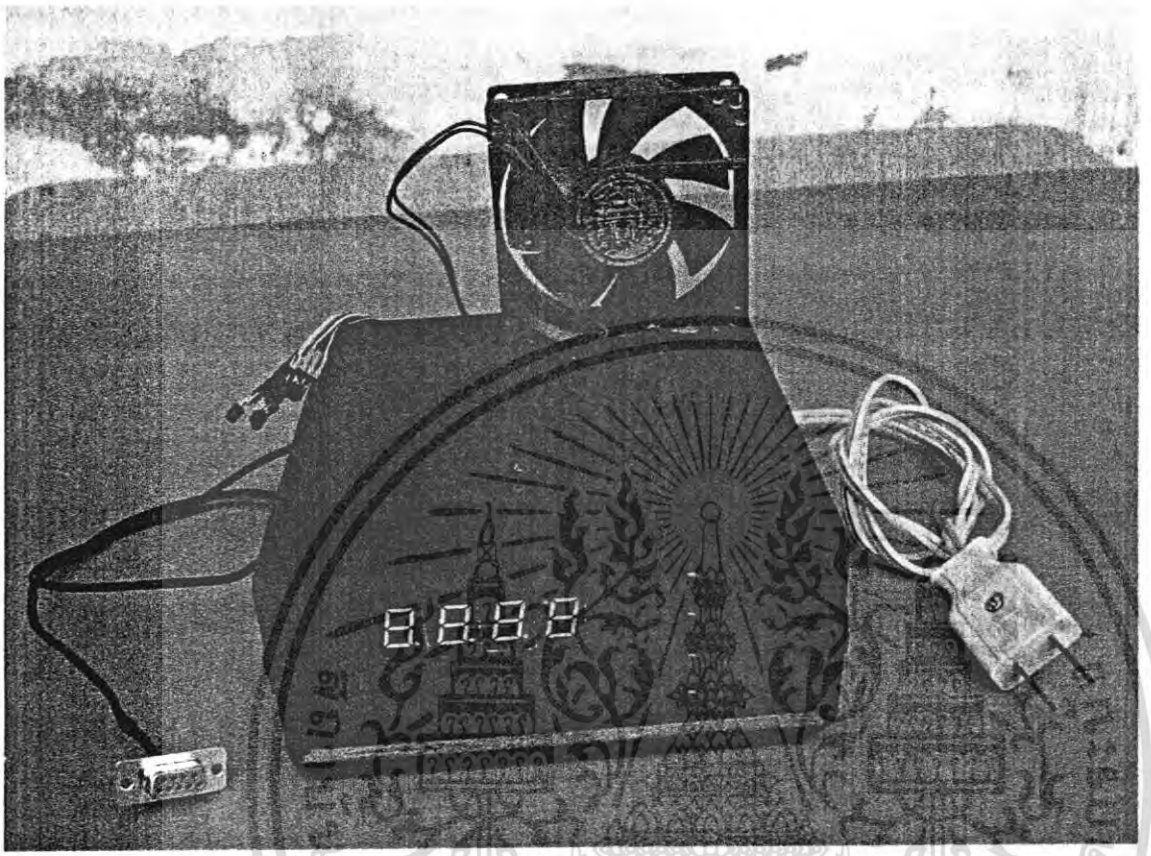
ส่วนของซอฟต์แวร์ การทำงานในส่วนที่เป็นการอ่านหมายเลขประจำตัว (ROM CODE) ของตัวเซ็นเซอร์โดยมีคำสั่งใช้งานอยู่ 5 รูปแบบ โดยอยู่ในรูปของเลขฐาน 16 โดยมีคำสั่งใช้งานอยู่ 2 แบบคือ 33h (Read Rom) และ 55h (Match Rom)



รูปที่3.3 แสดงแผนผังการวางอุปกรณ์ของบอร์ดระบบควบคุมอุณหภูมิแบบหลายจุด



รูปที่3.4 แสดงลายปริ้นขนาดจริงของบอร์ดควบคุมอุณหภูมิ



รูปที่3.5 แสดงภาพของบอร์ดระบบควบคุมอุณหภูมิแบบหลายจุด

3.4 การออกแบบการทำงานในส่วนของซอฟต์แวร์

การทำงานของโปรแกรม จะเป็นการควบคุมอุณหภูมิให้ได้เท่ากับค่าที่ต้องการ โดยมีรูปแบบการทำงานคือ

1. ต้องมีการกำหนดค่าที่ต้องการลงไปเป็นค่าอุณหภูมิ 2 หลักทางเครื่องคอมพิวเตอร์เพื่อเป็นค่าอ้างอิง
2. ส่งข้อมูลที่กำหนดไว้ผ่านโปรแกรม Modbus Commander ซึ่งจะต้องกำหนดค่า Address ของบอร์ดควบคุมอุณหภูมิแต่ละตัวด้วยว่าจะส่งไปให้ตัวใด
3. รับข้อมูลผ่านพอร์ตอนุกรมมาที่ตัวบอร์ด และโปรแกรมอินเทอรัปท์จะงานตามคำสั่งที่ส่งมาทาง Mosbus ว่าจะให้รับค่าอ้างอิงหรือจะให้ส่งค่าออกไปแสดงผลที่คอมพิวเตอร์

4. โปรแกรมที่ดัวบอร์ดจะทำงานตามปกติคือการอ่านค่าอุณหภูมิและแสดงผลไปที่เซเว่นเซ็กเมนต์และควบคุมอุณหภูมิจนกว่าจะมีการอินเทอรัปท์เข้ามาถึงจะกระโดดเข้าไปทำงานตามส่วนอินเทอรัปท์
5. ให้มีการอ่านค่าอุณหภูมิ 1 รอบก่อน เพื่อดูว่าอุณหภูมิตอนนี้เท่าไรนำค่าอุณหภูมิที่อ่านมาได้ลบกับค่าอ้างอิง จะได้ค่าความผิดพลาดออกมา
6. นำค่าที่วัดได้และค่าอ้างอิงมาเปรียบเทียบกันว่าควรจะเพิ่มหรือลดอุณหภูมิและทำการเพิ่มหรือลดอุณหภูมิตามที่เปรียบเทียบได้
7. ระบบจะทำงานวนอยู่อย่างนี้ตลอดเพื่อทำการปรับค่าอุณหภูมิจนกว่าจนกว่าค่าอุณหภูมิจะเข้าอุณหภูมิจะเข้าใกล้ค่าอ้างอิง

ในส่วนการทำงานของซอฟต์แวร์เครื่องคอมพิวเตอร์จะทำงานผ่านโปรแกรม Modbus Commander เพื่อติดต่อกับดัวบอร์ดซึ่งเป็น Freeware ที่สามารถใช้งานได้ เพราะ Mosbus Protocol เป็นโปรโตคอลที่นิยมใช้แพร่หลายในวงการอุตสาหกรรม โดยโปรแกรม Modbus Commander จะทำการเขียนหรืออ่านค่าลิงค์กับโปรแกรม Microsoft Excel ได้ทำให้สามารถง่ายต่อการพัฒนากับโปรแกรม Visual Basic

3.5 การควบหุมอุณหภูมิหรือตัวจ่ายอุณหภูมิ

เนื่องจากระบบที่ออกแบบจะสามารถใช้ได้ทั้งการควบคุมทั้งความเย็นและความร้อนตัวจ่ายอุณหภูมิจึงมีทั้งความเย็นเช่นควบคุมการปิดเปิดช่องลมในระบบปรับอากาศ ความคุมการหล่อเย็นของเครื่องจักร หรือ ควบคุมความร้อนในตู้อบหรือฮีทเตอร์ จึงสามารถแบ่งการคุมคุมอุณหภูมิได้ 2 แบบคือ

1.การควบคุมความเย็น

การควบคุมความเย็นดังนั้นการจ่ายอุณหภูมิจะเป็นการจ่ายความเย็นหรือการควบคุมพัดลมซึ่งจะทำให้ค่าอุณหภูมิที่วัดได้ลดลง การเปรียบเทียบเมื่ออุณหภูมิสูงกว่าค่าอ้างอิงถึงจะมีการจ่ายความเย็นออกไป และหยุดจ่ายความเย็นเมื่ออุณหภูมิต่ำกว่าค่าอ้างอิงเล็กน้อยถึงจะหยุดการจ่ายอุณหภูมิให้เป็นไปตามค่าผิดพลาดที่ยอมรับได้ถ้าการจ่ายอุณหภูมิเป็นแบบปิดเปิด หรือค่อยๆลดค่าการจ่ายอุณหภูมิลงจนกว่าจะถึงค่าอ้างอิงถ้าการจ่ายอุณหภูมิเป็นแบบเปอร์เซ็นต์

2.การควบคุมความร้อน

การควบคุมความร้อนดังนั้นการจ่ายอุณหภูมิจะเป็นการจ่ายความร้อนหรือการควบคุมฮีทเตอร์ซึ่งจะทำให้ค่าอุณหภูมิที่วัดได้เพิ่มขึ้น การเปรียบเทียบเมื่ออุณหภูมิต่ำกว่าค่าอ้างอิงถึงจะมีการจ่ายความร้อนออกไป และหยุดจ่ายความร้อนเมื่ออุณหภูมิสูงกว่าค่าอ้างอิงเล็กน้อย ถึงจะหยุดการจ่ายอุณหภูมิให้เป็นไปตามค่าผิดพลาดที่ยอมรับได้ถ้าการจ่ายอุณหภูมิเป็นแบบปิดเปิด หรือค่อยๆลดค่าการจ่ายอุณหภูมิลงจนกว่าจะถึงค่าอ้างอิงถ้าการจ่ายอุณหภูมิเป็นแบบเปอร์เซ็นต์

บทที่ 4

การทดลองและทดสอบระบบ

4.1 การทดลองการทำงานของเซ็นเซอร์

การทดสอบการอ่าน Rom Code ของเซนเซอร์ การทดสอบทำโดยให้อ่านเลขประจำตัวออกมาเรื่อยๆดังรูปที่ 4.1

```
$ff16 ff00 ff08 ff00 ff92 ff6d ffa4 ff10 @
$ff16 ff00 ff08 ff00 ff92 ff6d ffa4 ff10 @
```

รูปที่4.1 Rom Code ที่สามารถอ่านได้จากอุปกรณ์

การทดสอบการอ่านค่าอุณหภูมิของตัวเซนเซอร์ การทดสอบทำโดยให้อ่านอุณหภูมิออกมาเรื่อยๆ และนำสิ่งที่มีอุณหภูมิสูงกว่ามาแตะตัวเซ็นเซอร์จะเห็นการทำงานว่าอุณหภูมิที่อ่านได้จะค่อยๆเพิ่มขึ้น ดังรูป 4.2

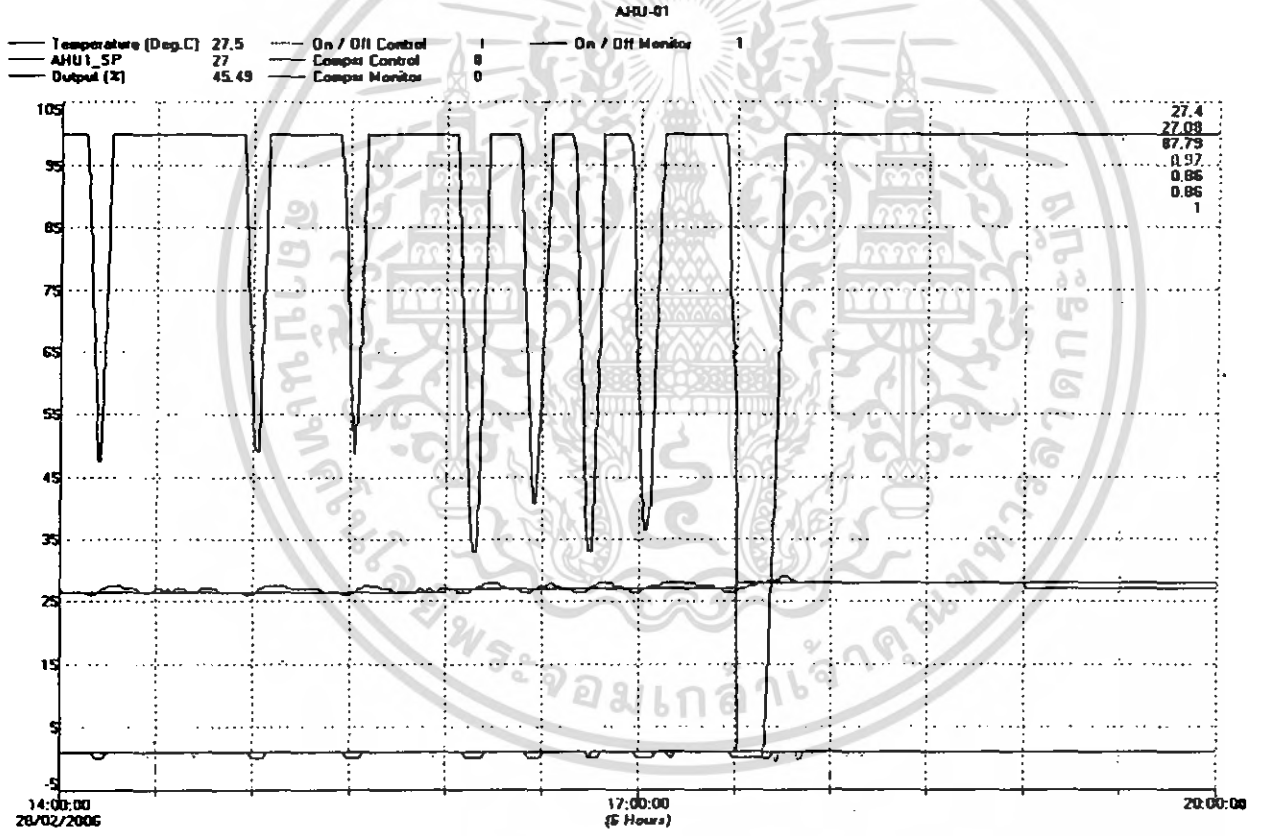
```
$ff16 ff00 ff08 ff00 ff92 ff6d ffa4 ff10 @
Temp1 22 degrees C
$ff16 ff00 ff08 ff00 ff92 ff6d ffa4 ff10 @
Temp1 23 degrees C
$ff16 ff00 ff08 ff00 ff92 ff6d ffa4 ff10 @
Temp1 24 degrees C
$ff16 ff00 ff08 ff00 ff92 ff6d ffa4 ff10 @
Temp1 25 degrees C
```

รูปที่4.2 การทดสอบการอ่านค่าของเซ็นเซอร์

4.2 การทดลองการทำงานของระบบควบคุมอุณหภูมิแบบหลายจุด

โดยที่โปรแกรมของบอร์ดระบบควบคุมอุณหภูมิแบบหลายจุดจะทำการอ่านค่าอุณหภูมิจากเซ็นเซอร์แต่ละตัว และนำค่านั้นมาเฉลี่ยกันแล้วนำมาเปรียบเทียบกับค่า Set point แล้วนำมาเป็น Output ในการควบคุม โดยในการทดลองจะทดลองกับห้องที่ใช้เครื่องปรับอากาศเครื่องปรับอากาศโดยตั้งค่า Set point ไว้ที่ 27 องศาเซลเซียสและใช้การควบคุมแบบ On/Off แล้วนำค่าต่างๆที่ได้มาพล็อตเป็นกราฟจะได้กราฟดังรูปที่ 4.3

ผลการทดลองการทำงานของระบบควบคุมอุณหภูมิแบบหลายจุด



รูปที่ 4.3 รูปกราฟแสดงผลการทำงานของระบบควบคุมอุณหภูมิแบบหลายจุด

จากรูปที่ 4.3 เส้นตรงที่ 27 องศาเซลเซียสจะคือ Set point และอุณหภูมิจริงจะขึ้นลงอยู่ใกล้เคียงกับค่า Set point ส่วนเส้นกราฟด้านล่างจะเป็นการควบคุมแบบ On/Off ที่ใช้การอยู่ และเส้นกราฟด้านบนจะเป็นการควบคุมแบบเปอร์เซ็นต์ เมื่ออุณหภูมิสูงขึ้นเครื่องปรับอากาศจะทำงาน และจะหยุดทำงานเมื่ออุณหภูมิต่ำกว่า Set point เล็กน้อย และจะทำงานวนแบบนี้ต่อไป

รูปที่ 4.4 แสดงค่า Set Point ของระบบควบคุมอุณหภูมิแบบหลายจุด

จากรูปที่ 4.4 จะเป็นการเซ็ทค่าอุณหภูมิ เพื่อที่จะใช้ในการอ้างอิงในการควบคุมอุณหภูมิแบบหลายจุด ซึ่งเราสามารถเซ็ทให้ระบบทำงานที่อุณหภูมิไหนก็ได้ ดังรูปซึ่งเซ็ทอ้างอิงให้ระบบทำงานไว้ที่ 27 องศาเซลเซียส คือ เมื่ออุณหภูมิเฉลี่ยทั้ง 4 จุด ของเซนเซอร์ (DS1820) มากกว่า 27 องศาเซลเซียส ระบบควบคุมก็จะทำงานทันที แล้วจะหยุดทำงานอีกที เมื่ออุณหภูมิต่ำกว่า 27 องศาเซลเซียส

บทที่ 5
สรุปการพัฒนาโครงงาน

5.1 สรุปการพัฒนาโครงงาน

ระบบที่นำมาใช้มีการออกแบบอย่างมีแบบแผนเพื่อให้เหมาะสมกับวัตถุประสงค์ของการนำไปใช้งาน และมีความยืดหยุ่นพอที่จะปรับเปลี่ยนให้มีการใช้งานได้หลายรูปแบบ และได้ออกแบบให้ใช้งานร่วมกับระบบอื่นๆ ได้ หรือสามารถนำมาต่อแบบเนตเวิร์กเพื่อการควบคุมที่ใหญ่ขึ้นได้ และสามารถสรุปได้ดังนี้

- สามารถควบคุมอุณหภูมิได้ละเอียดและประหยัดพลังงานที่ใช้
- มีเซ็นเซอร์หลายจุดสามารถดูได้ว่าจุดไหนอุณหภูมิมีเท่าไดเพื่อสามารถปรับอุณหภูมิจุดนั้นได้อย่างถูกต้อง
- สามารถปรับเปลี่ยนอุปกรณ์แต่ละตัวได้โดยง่าย เมื่อต้องการเปลี่ยนสภาพแวดล้อมในการควบคุม
- สามารถเพื่ออุปกรณ์เพื่อขยายระบบได้โดยง่าย

การเลือกใช้ระบบ Modbus ก็เพื่อให้ลดการใช้สายสัญญาณและจำนวนพอร์ทและใช้งานกับอุปกรณ์อื่นได้ถ้ามีการใช้งานในโรงงาน แต่อาจจะมีปัญหาถ้าจำเป็นต้องต่อสายในระยะทางไกลจำเป็นจะต้องใช้สายที่มีคุณภาพดีมาก

5.2 ปัญหาที่เกิดขึ้น

- ปัญหาในการคิดอัลกอริทึมสำหรับใช้งานเนื่องจากมีความหลากหลายในการใช้งานอัลกอริทึมที่ใช้และไม่ตอบสนองความต้องการได้ จึงต้องทำการศึกษาใหม่
- การใช้ UART INTERRUPT เป็นปัญหาพอสมควรในการออกแบบซอฟต์แวร์ เนื่องจากการอินเตอร์รัปผ่านพอร์ตอนุกรมในรูปแบบของ Modbus Protocol ซึ่งในช่วงแรกค่าที่อ่านได้จาก Modbus อ่านมาผิดพลาด ซึ่งจำเป็นต้องการศึกษา code คำสั่งและ Frame ข้อมูลของ Modbus อย่างละเอียด

- การใช้สายสัญญาณไม่ดีพอจึงไม่สามารถต่อสายสัญญานออกไปได้ไกล ถ้าใช้สายสัญญาณทั่วไป

อาจจะสามารถต่อออกไปได้ไกลเพียง 15 เมตรเท่านั้นซึ่งก็เพียงพอในการใช้งานกับระบบที่ไม่ใหญ่มาก

- การใช้เครื่องปรับอากาศขนาดเล็กเกินไปอาจจะทำให้ถึงค่า Set point ได้ช้าลง

5.3 แนวทางในการพัฒนาต่อ

ระบบควบคุมอุณหภูมิแบบหลายจุดนี้ถูกออกแบบและได้ทดลอง ในสถานการณ์ที่จำลองขึ้นมา ซึ่งสามารถพัฒนานำไปใช้ได้กับ การปรับอากาศในอาคาร การอบผลผลิตทางการเกษตร การควบคุมอุณหภูมิในอุตสาหกรรมต่างๆ ที่ต้องการการควบคุมอุณหภูมิในบริเวณกว้าง และต้องใช้อุปกรณ์วัดและจ่ายอุณหภูมิในหลายจุด ตัวอย่างเช่นเช่นการอบผลผลิตทางการเกษตร อาจนำ Output ที่ได้ไปควบคุมการจ่ายก๊าซ LPG เพื่อจ่ายเข้าไปในตู้อบ และวัดค่าของอุณหภูมิในแต่ละจุดออกมาเพื่อควบคุมการจ่าย ก๊าซ LPG เพื่อให้อุณหภูมิคงที่และได้ผลผลิตทางการเกษตรที่ผ่านการอบที่มีคุณภาพที่ควบคุมได้และใกล้เคียงกันออกมาก ทำให้มีผลผลิตที่มีคุณภาพมากขึ้นและสามารถประหยัดการใช้พลังงานได้