ชุดทดลองระบบควบคุมกระบวนการ

PROCESS CONTROL SYSTEM

นายรวี ไทยประธาน
นายรังสฤษฏ์ แสงบัวเผื่อน
นายสิรณัฐ โกมลมิศร์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมแมคคาทรอนิกส์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2553

PROCESS CONTROL SYSTEM

Mr. Rawee Thaiprathan

Mr. Rungsarit Saengbuaphuen

Mr. Siranat Komolmisr

THIS THESIS IS SUMMITED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE DEGREE OF

BECHELOR OF ENGINEERING IN MECHATRONICS ENGINEERING

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

ACADEMIC YEAR 2010

ปริญญานิพนธ์ปีการศึกษา 2553

สาขาวิชาวิศวกรรมการวัดและควบคุม คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ชุดทดลองระบบควบคุมกระบวนการ
PROCESS CONTROL SYSTEM

ผู้จัดทำ

นายรวี	ไทยประธาน	50011271
นายรังสฤษฎ์	แสงบัวเผื่อน	50011278
นายสิรณัฐ	โกมลมิศร์	50011680

.....อาจารย์ที่ปรึกษา
(รองศาสตราจารย์ ดร.วรพงศ์ ตั้งศรีรัตน์)

ชุดทดลองระบบควบคุมกระบวนการ

โดย

นายรวี	ไทยประธาน	50011271
นายรังสฤษฏ์	แสงบัวเผื่อน	50011278
นายสิรณัฐ	โกมลมิศร์	50011680

อาจารย์ที่ปรึกษา

รองศาสตราจารย์ ดร. วรพงศ์ ตั้งศรีรัตน์

ปีการศึกษา 2553

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้ เป็นการนำเสนอเกี่ยวกับระบบควบคุมกระบวนการโดยใช้ไมโครคอนโทลเลอร์ PIC16F877 เป็นหน่วยประมวลผลกลางและควบคุมแบบกระจายส่วน ซึ่งการทดลองครั้งนี้ได้สร้างชุดทดลองระบบควบคุมกระบวนการ โดยประกอบด้วย ระบบควบคุมแบบป้อนกลับเป็นหลัก ซึ่งอาศัยการทำงานโดยใช้การวัดค่า อุณหภูมิ ระดับน้ำ การสั่งการควบคุมวาล์วอัตโนมัติ โดยนำค่าต่างๆที่เกิดการไปเปลี่ยนในกระบวนการไปใช้ในการควบคุมอุปกรณ์อื่นๆ ชุดทดลองควบคุมกระบวนการติดต่อกับคอมพิวเตอร์โดยใช้มาตรฐาน RS-232 ในการสื่อสาร สั่งการควบคุมอุปกรณ์ทั้งหมดและแสดงผลบนหน้าจอคอมพิวเตอร์ พร้อมทั้งบันทึกค่าที่ได้และอัพโหลดค่าที่ได้ไปแสดงผลยังระบบอินเตอร์เน็ต

PROCESS CONTROL SYSTEM

By

Mr. Rawee Thaiprathan 50011271

Mr. Rungsarit Saengbuaphuen 50011278

Mr. Siranat Komolmisr 50011680

Advisor

Assoc. Prof. Dr. Worapong Tangsrirat

Academic Year 2010

ABSTRACT

This project presents the process control system using microcontroller PIC16F877 as a central processing unit and distributed control system. This system build up based on the feedback control system, in order to measure and control process parameters such as temperature, water level and automatic controlled valve. The measured valves are sent to the microcontroller in order to control the instrument in process and displayed on a personal computer. The RS-232 communication standard is use to communicate between microcontroller and the computer. At the same time the process parameters and status can be uploaded and displayed on the internet

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลงด้วยดี เนื่องด้วยได้รับการสนับสนุนอย่างดีเยี่ยม โดยคณะผู้จัดทำขอกราบขอบคุณบิดามารดาและบุพการีทุกท่านที่ได้อุปการะอย่างดีเยี่ยม ไม่ว่าจะเป็นผู้ให้ชีวิต ผู้ฟูมฟักเลี้ยงดู อบรมสั่งสอน ตลอดจนส่งเสียด้านการศึกษา และขอบคุณสมาชิกในครอบครัวที่เป็นกำลังใจให้เสมอมา

ขอกราบขอบพระคุณอาจารย์วรพงศ์ ตั้งศรีรัตน์ อาจารย์ที่ปรึกษาโครงงานเป็นอย่างสูงที่สละเวลาอันมีค่าเพื่อมาให้คำปรึกษาและแนะนำแนวทางในการดำเนินโครงงานจนทำให้โครงงานประสบความสำเร็จเป็นอย่างดี และขอกราบขอบพระคุณอาจารย์ทุกท่านที่ประสิทประสาทวิชาและให้คำแนะนำกับคณะผู้จัดทำ ทำให้มีความรู้เพื่อมาใช้ในการดำเนินการ โครงงานนี้ได้อย่างสำเร็จลุล่วงได้ด้วยดี

ขอขอบคุณสาขาวิชาระบบควบคุมและแมคคาทรอนิกส์ที่เอื้อเฟื้อเครื่องมือและอุปกรณ์อย่างครบครันรวมถึงเอื้อเฟื้อสถานที่ในการทำโครงงานชิ้นนี้ได้อย่างสะดวกสบาย

ขอขอบคุณเพื่อนๆทุกคนที่คอยให้กำลังใจ และให้คำแนะนำ ตลอดจนสร้างเสียงหัวเราะที่ทำให้คณะผู้จัดทำผ่อนคลายจากความตึงเครียด ตลอดช่วงระยะเวลาการทำโครงงาน

ผู้จัดทำ

นายรวี ไทยประธาน

นายรังสฤษฎ์ แสงบัวเผื่อน

นายสิรณัฐ โกมลมิศร์

สารบัญ

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

สารบัญรูป

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญตาราง

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปริญญานิพนธ์

เนื่องจากในปัจจุบันโลกของเรานั้นพัฒนาเข้าสู่ยุคของอุตสาหกรรมอย่างถึงขั้นเกือบเต็มรูปแบบ การทำงานต่างๆ มีการใช้แรงงานมนุษย์น้อยลง โดยมนุษย์ส่วนใหญ่จะหันไปเป็นผู้คิดค้นหรือผู้สั่งการมากกว่าที่จะเป็นผู้ถูกใช้แรงงาน แต่ในส่วนที่เป็นการทำงานก็ได้มีการนำเครื่องจักรกลต่างๆ ที่มนุษย์คิดค้นเข้ามาทำหน้าที่แทนโดยที่สามารถทำงานได้ละเอียดกว่ามนุษย์ และไม่ต้องการเวลาพักผ่อน การทำงานของเครื่องจักรนั้นจะถูกควบคุมโดยโปรแกรมคอมพิวเตอร์ ซึ่งมนุษย์เป็นผู้สั่งงาน และส่วนใหญ่การทำงานของเครื่องจักรกลเหล่านี้จะเป็นการทำงานร่วมกันกับจักรกลตัวอื่นๆ โดยทำการประสานงานกันโดยอาศัยระบบควบคุมแบบต่างๆ และหนึ่งในจำนวนนั้นก็คือระบบควบคุมแบบป้อนกลับ

ในการเรียนการสอนปัจจุบันนักศึกษาส่วนใหญ่ได้เรียนรู้จากตำราต่างๆ ในด้านทฤษฎี ซึ่งเป็นผลว่า เมื่อนักศึกษาสำเร็จการศึกษาแล้ว ได้ไปทดลองทำงานในสถานที่จริง ก็จะเกิดความสับสน หรือเกิดความไม่แน่ใจ เนื่องจากที่ได้ศึกษามานั้น เป็นเพียงตัวอักษรจากตำราเรียน หรือแค่เพียงทฤษฎีเพียงอย่างเดียวเท่านั้น

ชุดทดลองการวัดและการควบคุมกระบวนการนี้ ได้จัดทำขึ้นมาเพื่อตอบสนองหรือช่วยลดปัญหาทางด้านนี้ โดยเป็นการให้นักศึกษาได้ทดลอง ได้เรียนรู้เกี่ยวกับระบบควบคุมแบบป้อนกลับว่าเป็นอย่างไร ได้วิเคราะห์ถึงความเกี่ยวเนื่องกันและกันของอุปกรณ์ต่างๆ ภายในระบบ โดยจะสามารถบันทึกผลการทดลอง เก็บค่า และตลอดจนนำไปวิเคราะห์สิ่งต่างๆ ได้

1.2 วัตถุประสงค์ในการทำปริญญานิพนธ์

1. เพื่อศึกษาการใช้งานและหลักการทำงานของเซนเซอร์ตรวจวัดอุณหภูมิ วัดระดับน้ำ และวัดระยะขจัดเชิงมุม
2. เพื่อศึกษาการรับส่งข้อมูลกับอุปกรณ์ภายนอกของคอมพิวเตอร์โดยผ่านพอร์ตอนุกรม
3. เพื่อศึกษาหลักการและแนวคิดในการเขียนโปรแกรมเพื่อใช้ในสร้างตัวคอนโทรลเลอร์
4. เพื่อศึกษาหลักการและแนวคิดในการเขียนโปรแกรมเพื่อใช้ในการแสดงผลและควบคุมกระบวนการ
5. เพื่อศึกษาเกี่ยวกับระบบควบคุมแบบป้อนกลับ

1.3 ขั้นตอนการศึกษาและจัดทำโครงงาน

1. ทำการศึกษาเกี่ยวกับการวางท่อและออกแบบทางเดินของน้ำในท่อ
2. ศึกษาและทดลองการใช้เครื่องมือวัดในกระบวนการได้
3. ศึกษาการใช้งานไมโครคอนโทรลเลอร์ PIC16F877 และเขียนโปรแกรมเพื่อควบคุมกระบวนการ
4. ทำการศึกษาและการใช้งานโปรแกรม Visual basic เพื่อสร้างแอพพลิเคชั่นในการรับค่าจากอุปกรณ์ต่างๆในระบบ ปรับแต่ง ตั้งค่า คำนวณและแสดงผลบนหน้าจอคอมพิวเตอร์
5. ทดลองการควบคุมและบันทึกผล
6. สรุปผลการทดลองและเปรียบเทียบข้อมูลทางทฤษฎีที่ได้ศึกษามา
7. ทำรายงานและเตรียมส่งโครงงาน

1.4 รายละเอียดของปริญญานิพนธ์

เนื้อหาที่จะกล่าวในปริญญานิพนธ์ฉบับนี้ประกอบด้วย

บทที่ 1 บทนำ กล่าวนำถึงวัตถุประสงค์ ขั้นตอนการศึกษา และการจัดทำโครงงาน พร้อมทั้งรายละเอียดของปริญญานิพนธ์ของแต่ละบท

บทที่ 2 ทฤษฎีและความรู้ที่เกี่ยวข้อง กล่าวถึงอุปกรณ์ต่างๆที่ใช้ในกระบวนการควบคุมระดับน้ำ ทฤษฎีที่เกี่ยวข้องกับการควบคุม, วงจรต่างๆที่ใช้ในโครงงาน, การเขียนโปรแกรมวิชวลเบสิก, และไมโครคอนโทรลเลอร์ รวมถึงการติดต่อข้อมูลระหว่างกระบวนการกับคอมพิวเตอร์ และนำเอาความรู้ไปประยุกต์ใช้ในการจัดทำโครงงาน

บทที่ 3 หลักการออกแบบ นำเสนอการประกอบโครงสร้างของระบบ รวมถึงแนวคิดในการออกแบบระบบควบคุม

บทที่ 4 การทดลอง เป็นส่วนการทดสอบองค์ประกอบต่างๆ ในระบบ ตลอดจนการทดลองการใช้งานระบบควบคุมกระบวนการ

บทที่ 5 บทวิจารณ์และสรุป จะสรุปผลการดำเนินงาน ปัญหาที่เกิดขึ้น และแนวทางการปรับปรุงพัฒนาโครงงานนี้ต่อไป

บทที่ 2
ทฤษฎีและหลักการ

2.1 ระบบควบคุม

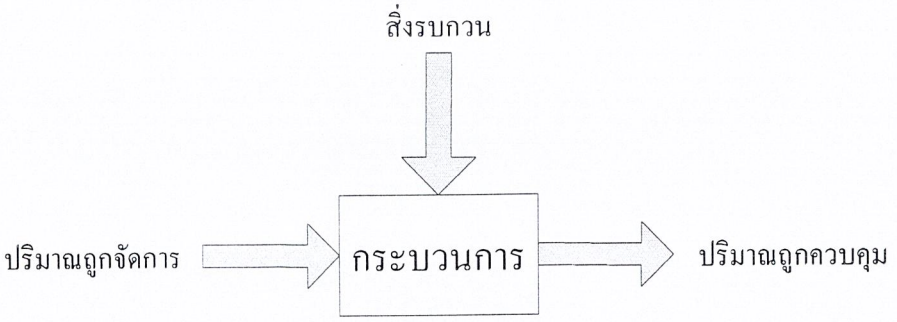
2.1.1 ประวัติการควบคุม

การควบคุมแบบป้อนกลับมีใช้ครั้งแรกเมื่อเจมส์วัตต์ได้ใช้ตัวบังคับแบบลูกหมุน (flyball governor) กับเครื่องจักรไอน้ำประมาณปี 1775 การใช้งานทฤษฎีที่เกี่ยวกับการควบคุมป้อนกลับส่วนมากต้องใช้ตัวบังคับ (Governors) รวมถึงการใช้งานอุตสาหกรรมด้วย การควบคุมอัตโนมัติเริ่มใช้กว้างขึ้นหลังจากทศวรรษ 1920 ทฤษฎีเกี่ยวกับการควบคุมอัตโนมัติถูกตีพิมพ์ครั้งแรกเมื่อ 1932 การเติบโตของการใช้งานในอุตสาหกรรมได้เข้าสู่ภาวะคงตัวและแข็งแกร่ง

เทคโนโลยีใหม่ๆ มากมายถูกนำมาใช้ควบคุมฮาร์ดแวร์ของกระบวนการ ขณะที่เทคนิคการวางระบบอัตโนมัติที่ใช้ในอุตสาหกรรมได้มีการพัฒนาและเจริญเต็มที่ในช่วงระยะเวลา 70 ปี ตัวอย่างการใช้งานที่สำคัญคือ การใช้คอมพิวเตอร์ดิจิตัล และไมโครโพรเซสเซอร์ในการควบคุมกระบวนการตั้งแต่ทศวรรษ 1960 ผลลัพธ์คือ การวางระบบอัตโนมัติในกระบวนการได้รับความสำคัญและแรงเสริมจากเทคโนโลยีต่างๆ ปัจจุบันอุตสาหกรรมส่วนมากได้จัดสรรเงินทุนมากกว่าร้อยละ 10 สำหรับเครื่องมือวัดและควบคุม และตัวเลขร้อยละนี้ ได้เพิ่มเป็นสองเท่าในช่วง 30 ปีที่ผ่านมา และไม่มีแนวโน้มว่าจะลดลงแต่อย่างใด

2.1.2 ตัวแปรที่เกี่ยวข้องกับการควบคุม

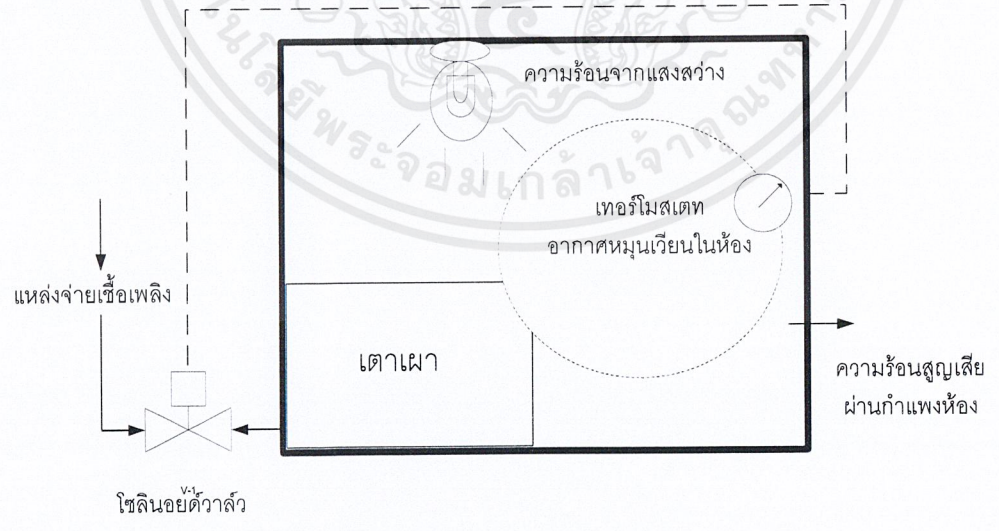
เพื่อการเข้าใจการควบคุมกระบวนการอัตโนมัติ จำเป็นต้องทำความเข้าใจกับตัวแปรสำคัญที่เกี่ยวข้องกับการควบคุมกระบวนการสามตัวแปรคือ ตัวแปรถูกควบคุม (Controlled Variable) หรือปริมาณที่ถูกควบคุม ตัวแปรจัดการ (Manipulated Variable) หรือปริมาณที่ถูกจัดการ ตัวแปรของการรบกวน ตัวแปรควบคุมจะเป็นเงื่อนไขที่ต้องการควบคุมหรือต้องการรักษาให้อยู่ในระดับที่ต้องการ ซึ่งอาจจะเป็นอัตราการไหล ระดับ ความดัน อุณหภูมิ การจัดอัตราส่วนหรืออะไรที่เป็นตัวแปรกระบวนการ โดยมีการตั้งเป้าหมายการควบคุมตัวแปรให้เป็นไปตามค่าที่ต้องการเรียกว่าค่าปรับตั้ง (Set Point) หรือค่าอินพุตอ้างอิง



รูปที่ 2.1 ตัวแปรที่เกี่ยวข้องกับการควบคุม

ในแต่ละตัวแปรถูกควบคุมจะมีตัวแปรจัดการรวมอยู่ด้วย ในกระบวนการควบคุมการไหล อัตราการไหลจะถูกจัดการผ่านตัววาล์วควบคุม เมื่อมีการรบกวนเข้ามาในกระบวนการจะทำให้ตัวแปรถูกควบคุมเบี่ยงเบนไปจากค่าเป้าหมายที่ต้องการ ระบบควบคุมอัตโนมัติต้องทำการปรับตัวแปรจัดการ เพื่อรักษาตัวแปรถูกควบคุมให้อยู่ที่ค่าปรับตั้งได้โดยไม่ถูกกระทบจากการรบกวน ในกรณีที่ต้องการเปลี่ยนเป้าหมายการควบคุมก็ต้องมีการปรับเลื่อนค่าตัวแปรจัดการใหม่ด้วย

รูปที่ 2.2 แสดงระบบการให้ความร้อนภายในบ้าน ตัวแปรถูกควบคุมคือ อุณหภูมิของห้องที่ต้องการให้ความสบายและสามารถวัดค่าได้สะดวก การรบกวนที่ทำให้อุณหภูมิของห้องเปลี่ยนแปลงคือ อุณหภูมิจากภายนอกห้อง จำนวนผู้คนที่อยู่ในห้อง เครื่องใช้ไฟฟ้าที่กระจายความร้อนภายในห้อง ระบบควบคุมอัตโนมัติถูกออกแบบให้จัดการกับการไหลของเชื้อเพลิงที่จ่ายให้เตาผิง เพื่อรักษาอุณหภูมิห้องให้คงที่ตามต้องการไม่ให้แปรเปลี่ยนไปตามการรบกวน



รูปที่ 2.2 ระบบให้ความร้อนภายในบ้าน

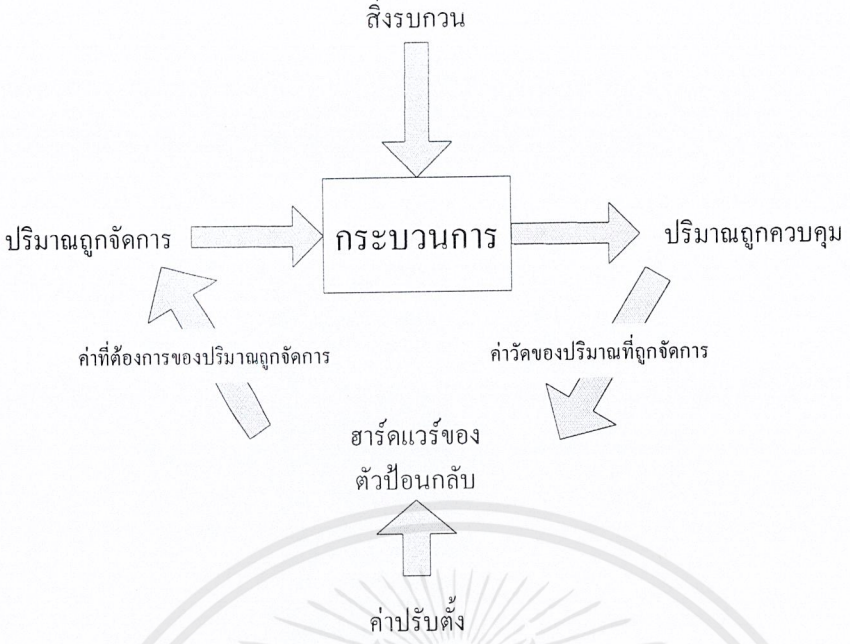
2.1.3 การควบคุมด้วยมือ

การควบคุมด้วยมือ (Typical Manual Control) เป็นกระบวนการที่มีตัวแปรถูกควบคุมตัวแปรเดียวอยู่ที่ทางไหลออกจากกระบวนการ มีตัวชี้บอก (Indicator) ที่แสดงค่าตัวแปรให้ผู้ควบคุมเครื่อง (Operator) อ่านค่าได้ตลอดเวลา ผู้ควบคุมเครื่องจะคาดการณ์ล่วงหน้าจากค่าที่อ่านได้นำไปปรับอัตราการไหลเข้ากระบวนการ เพื่อให้ได้ค่าตัวแปรถูกควบคุมเป็นไปตามเป้าหมายที่อยู่ในใจตลอดเวลา และผู้ควบคุมเครื่องเป็นผู้ตัดสินใจทุกอย่างเกี่ยวกับการควบคุม จะเห็นว่ามีปัญหาหลายอย่างในการควบคุมด้วยมือ หากผู้ควบคุมเครื่องมีความชำนาญในการคาดการณ์ล่วงหน้าไม่เพียงพอ ความเมื่อยล้าจากการเฝ้าดูอาการเป็นเวลานานๆ ซึ่งล้วนจะทำให้การควบคุมผิดพลาดไม่เป็นไปตามเป้าหมายการควบคุม

รูปที่ 2.3 การควบคุมด้วยมือ

2.1.4 การควบคุมป้อนกลับ

การควบคุมกระบวนการอัตโนมัติแบบง่ายๆ จะใช้วิธีการป้อนกลับซึ่งเป็นวิธีพื้นฐานที่มีใช้กันกว้างขวาง มีการติดตั้งตัวรับรู้ (Sensor) หรืออุปกรณ์วัด เพื่อวัดค่าจริงของตัวแปรถูกควบคุมแล้วส่งถ่ายไปยังฮาร์ดแวร์ที่ควบคุมป้อนกลับ โดยทำการเปรียบเทียบระหว่างค่าปรับตั้งหรือค่าที่ต้องของตัวแปรถูกควบคุมกับค่าที่วัดได้จริงจะได้ค่าผลต่างหรือค่าผิดพลาด (Errors) ที่ใช้ในการคำนวณค่าตัวแปรจัดการ เพื่อส่งไปปรับอุปกรณ์ที่จัดการกับอินพุตของกระบวนการ (ซึ่งมักจะเป็นวาล์วควบคุม) โดยอัตโนมัติ



รูปที่ 2.4 พื้นฐานการควบคุมป้อนกลับ

ข้อดีของการควบคุมป้อนกลับ คือ ผู้ออกแบบไม่จำเป็นต้องรู้ว่าจะมีสัญญาณรบกวนอะไรที่จะมากระทบกระบวนการ และไม่จำเป็นต้องรู้ค่าความสัมพันธ์ระหว่างสัญญาณรบกวนเหล่านั้นหรือผลกระทบท้ายสุดของสัญญาณรบกวนที่มีต่อตัวแปรถูกควบคุม

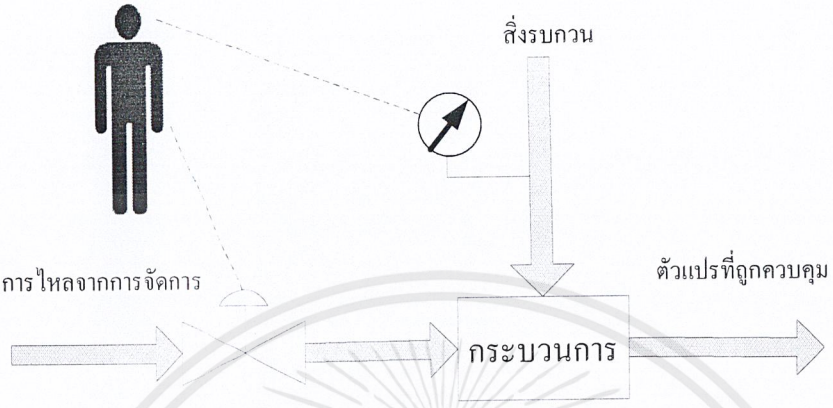
สิ่งที่ผู้ออกแบบต้องให้ความสำคัญคือ ฮาร์ดแวร์ที่ใช้ในวง และฮาร์ดแวร์แต่ละส่วนจะทำงานเข้ากันได้หรือไม่ โดยที่ยุทธวิธีการควบคุมแบบป้อนกลับยังเหมือนเดิมเสมอ การควบคุมป้อนกลับดังกล่าว เป็นเทคนิคการควบคุมกระบวนการอัตโนมัติ ซึ่งเป็นพื้นฐานที่ใช้กันกว้างขวางในอุตสาหกรรม

2.1.5 การควบคุมป้อนไปหน้าด้วยมือ

การควบคุมป้อนไปหน้ามีพื้นฐานแตกต่างจากการควบคุมป้อนกลับ ขณะที่มีสัญญาณรบกวนเข้าไปในกระบวนการผู้ควบคุมเครื่องจะเห็นการรบกวนตามธรรมชาตินี้จากการชี้บอก และทำการปรับตัวแปรจัดการ ตามสัญญาณรบกวนที่เข้ามาเพื่อป้องกันการเปลี่ยนแปลงใดๆ ที่จะเกิดท้ายที่สุด หรือป้องกันการเปลี่ยนแปลงในตัวแปรถูกควบคุมที่เกิดจากสัญญาณรบกวน การควบคุมป้อนกลับทำงานเพื่อกำจัดค่าผิดพลาด แต่การควบคุมป้อนไปหน้าให้ผลในการป้องกันค่าผิดพลาดจากที่เกิดขึ้นในตอนแรก ประโยชน์ของการควบคุมป้อนไปหน้าจึงเห็นได้ชัด

การควบคุมป้อนไปหน้าเพิ่มขีดความต้องการของผู้ปฏิบัติเป็นอย่างมาก ผู้ปฏิบัติต้องรู้ล่วงหน้าว่ามีสัญญาณรบกวนอะไรจะเข้ามาในกระบวนการ และต้องมีการตระเตรียมการวัดสัญญาณรบกวนเหล่านี้ อีกประการหนึ่งผู้ควบคุมในห้องควบคุม ต้องรู้แน่นอนว่าจะปรับตัวแปร

จัดการเมื่อไร และอย่างไร เพื่อชดเชยผลกระทบของสัญญาณรบกวนได้ถูกต้อง หากผู้ปฏิบัติมีความสามารถเฉพาะเหล่านี้ และหากพวกเขาพร้อมที่จะลงมือปฏิบัติ สัญญาณถูกควบคุมจะไม่เปลี่ยนไปจากค่าที่ต้องการหรือค่าปรับตั้ง



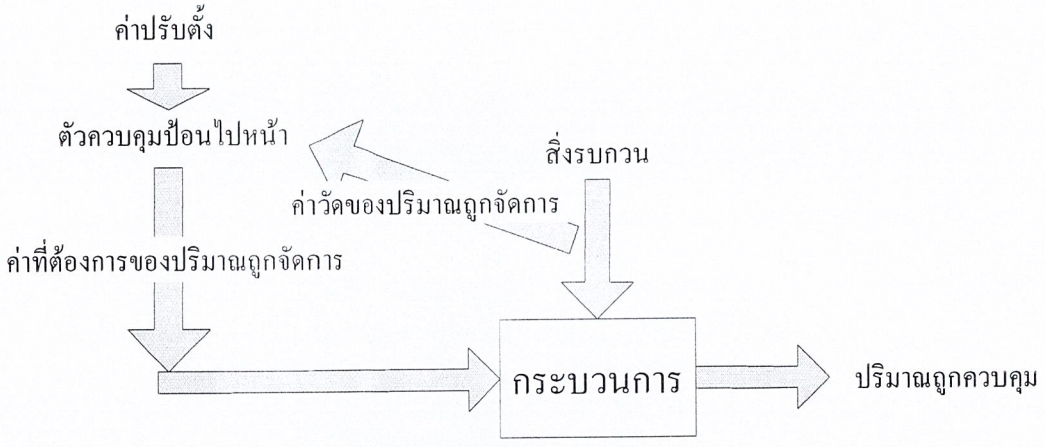
รูปที่ 2.5 การควบคุมป้อนไปหน้าด้วยมือ

ถ้าหากผู้ควบคุมเครื่องกระทำการผิดพลาดบางอย่างหรือไม่สามารถคาดคะเนสัญญาณรบกวนทั้งหมดที่อาจมีผลกระทบต่อกระบวนการ ตัวแปรถูกควบคุมจะเบี่ยงเบนจากค่าที่ต้องการได้ และการควบคุมป้อนไปหน้าอย่างเดียว จะมีค่าผิดพลาดที่ไม่สามารถแก้ไขได้เกิดขึ้น

2.1.6 การควบคุมป้อนไปหน้าอัตโนมัติ

รูปที่ 2.6 แสดงแนวคิดทั่วไปของการควบคุมป้อนไปหน้าอัตโนมัติ มีสัญญาณรบกวนป้อนเข้ากระบวนการ และมีตัวรับรู้คอยวัดสัญญาณรบกวนเหล่านี้ ตัวควบคุมป้อนไปหน้า (Feedforward Controller) จะคำนวณค่าของตัวแปรจัดการที่ต้องการตามสัญญาณรบกวนที่วัดได้หรือรับรู้ได้ ค่าปรับตั้งที่แสดงค่าตัวแปรถูกควบคุมที่ต้องการได้ถูกป้อนให้กับตัวควบคุมป้อนไปหน้า

จะเห็นได้ชัดว่าตัวควบคุมป้อนไปหน้าต้องทำการคำนวณสลับซับซ้อนมาก การคำนวณนี้ต้องสะท้อนและเข้าใจผลกระทบของสัญญาณรบกวนที่มีต่อตัวแปรถูกควบคุมได้อย่างถูกต้อง ด้วยความเข้าใจดังกล่าว ตัวควบคุมป้อนไปหน้าสามารถคำนวณผลลัพธ์ของขนาดตัวแปรจัดการที่ใช้ชดเชยสัญญาณรบกวนได้แน่นอน การคำนวณนี้สามารถอธิบายความเข้าใจเฉพาะของผลกระทบของตัวแปรจัดการที่มีผลต่อตัวแปรถูกควบคุมได้ ถ้าหากความสัมพันธ์ทางคณิตศาสตร์ทั้งหมดได้มาพร้อม ตัวควบคุมก็สามารถคำนวณการเปลี่ยนแปลงตัวแปรจัดการที่จำเป็นต่อการชดเชยการเปลี่ยนแปลงสัญญาณรบกวนได้อย่างอัตโนมัติ

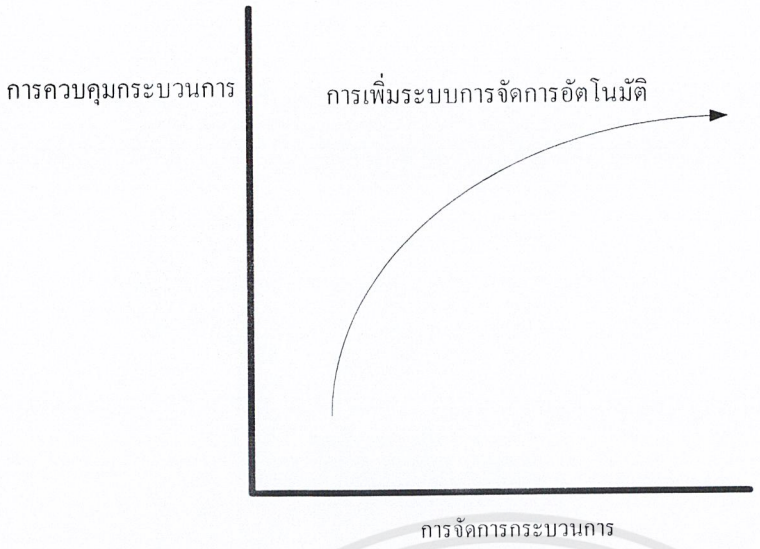


รูปที่ 2.6 พื้นฐานการควบคุมป้อนไปหน้า

การเพิ่มความเข้าใจทางทฤษฎีเป็นสิ่งที่ต้องการ การควบคุมป้อนไปหน้าเป็นแนวคิดที่ใช้ประโยชน์ได้มาก มีการเพิ่มหลักวิชาการและความต้องการทางวิศวกรรมของทั้งผู้ออกแบบและผู้ปฏิบัติ ผลลัพธ์คือ การควบคุมป้อนไปหน้าจึงมีไว้สำหรับใช้กับวงควบคุมที่สำคัญในกระบวนการบางวงเท่านั้น ขณะที่การใช้งานมีไม่มาก แต่ก็มีความสำคัญมาก

2.1.7 การควบคุมกระบวนการและการจัดการกระบวนการ

การวางระบบกระบวนการอัตโนมัติ เพื่อได้มาซึ่งผลประโยชน์สูงสุดของกระบวนการ ในหัวข้อก่อนของเรื่องนี้ได้สมมติว่ารู้ค่าปริมาณที่ต้องการควบคุม หากเมื่อใดรู้ค่าที่ต้องการนี้ ต้องใช้เทคนิคการวางระบบอัตโนมัติ เพื่อบรรลุหรือรักษาค่าที่ต้องการเหล่านี้ หรือค่าปรับตั้ง อย่างไรก็ตาม มีคำถามที่สำคัญบางคำถามเกี่ยวกับผลประโยชน์ของกระบวนการ ซึ่งต้องตอบว่าค่าที่ต้องการเป็นเท่าไร อันนี้เป็นพื้นฐานของงานการจัดการ และมักปล่อยให้ผู้ควบคุมเครื่องเป็นผู้พิจารณา แต่ในไม่กี่ปีที่ผ่านมามีความก้าวหน้าที่สำคัญทางการวางระบบกระบวนการอัตโนมัติ ทำให้งานการจัดการต่างๆ ได้กลายเป็นอัตโนมัติภายในตัวของมัน และความสามารถในการบรรลุถึงผลลัพธ์ทางหลักวิชาการและไขปัญหาทางฮาร์ดแวร์สำหรับคำถามการจัดการที่สำคัญของภาพการควบคุม



รูปที่ 2.7 การควบคุมและการจัดการกระบวนการ

ในกระบวนการพิเศษ ขณะที่ระดับการวางระบบอัตโนมัติได้เพิ่มขึ้น ขั้นตอนเริ่มแรกส่วนใหญ่ยังใช้การควบคุมกระบวนการแบบดั้งเดิม (conventional control) เช่น การควบคุมป้อนกลับ อย่างไรก็ตาม ขณะที่ระดับการวางระบบอัตโนมัติเพิ่มขึ้น การวางระบบอัตโนมัติส่วนมาก จะเกี่ยวข้องกับการจัดการกระบวนการซึ่งได้แสดงในรูปที่ 2.7

การรวมกันของปรากฏการทั้งสอง (การควบคุมกระบวนการและการจัดการกระบวนการ) ต้องถูกสะท้อนเข้าไปในภาพรวมของความเข้าใจ และเล็งเห็นถึงความสำคัญของการวางระบบกระบวนการอัตโนมัติ

2.2 ไมโครคอนโทรลเลอร์ PIC16F877

รูปที่ 2.8 ไมโครคอนโทรลเลอร์ PIC16F877

คุณสมบัติที่น่าสนใจ

- มีหน่วยความจำโปรแกรมแบบแฟลช เขียนโปรแกรมใหม่ได้นับแสนครั้ง

- มีหน่วยความจำข้อมูลอีอีพรอมที่บันทึกข้อมูลใหม่ได้นับล้านครั้ง คือเป็นการแยกหน่วยความจำโปรแกรมและหน่วยความจำข้อมูลออกจากกัน และมีบัสสำหรับติดต่อแยกกันด้วย โดยมีโครงสร้างดังนี้ โครงสร้างหลักคือ ซีพียู, หน่วยความจำโปรแกรม, หน่วยความจำข้อมูล, ส่วนติดต่ออินพุต - เอาต์พุต, วงจรสัญญาณนาฬิกา, วงจรรีเซตหลัก, วงจรไฟเลี้ยง, และส่วนจัดการอินเตอร์รัปต์ และยังมีโมดูลพิเศษต่าง ๆ เพื่อเข้ามา เช่น

- วงจรบราวเอาต์รีเซต (brown-out reset)

- ส่วนแก้ไขข้อมูลในวงจร หรือดีบักเกอร์ (In-circuit debugger)

- วงจรโปรแกรมแรงดันค่า (low-voltage programming)

- ไทเมอร์ 3 ตัว

- วงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอล 10 บิต

- โมดูลเชื่อมต่ออุปกรณ์อนุกรม (SPI : Serial Peripheral Interfacing)

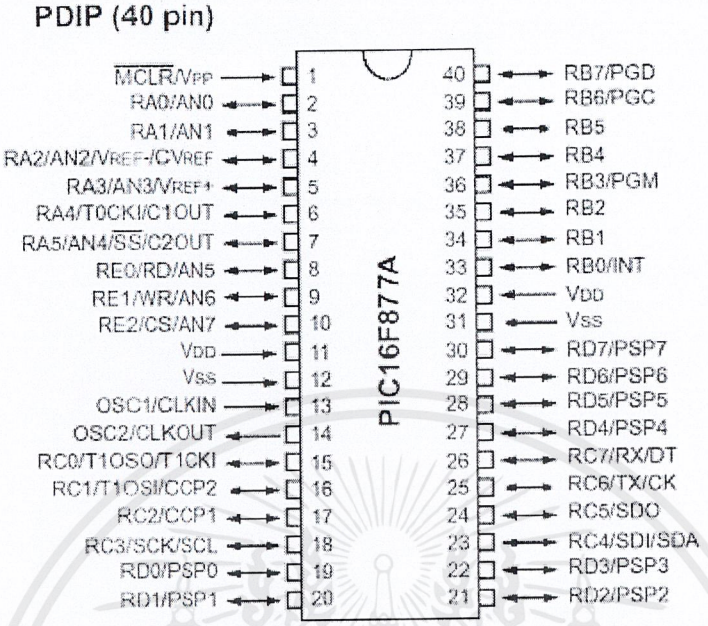
- โมดูลเชื่อมต่ออุปกรณ์ระบบบัส I2C

- โมดูลสื่อสารอนุกรม (USART : Universal Synchronous Asynchronous Receiver Transmitter)

- โมดูลเปรียบเทียบสัญญาณ - ตรวจจับสัญญาณ – วงจรมอดูเลชั่นทางความกว้างของพัลส์ (CCP : Compare Capture Pulse-width modulation)

- หากเป็นเบอร์ PIC16F877 จะมีวงจรเปรียบเทียบแรงดันอะนาลอก และโมดูลสร้างแรงดันอ้างอิงเพิ่มเข้ามาด้วย

คุณสมบัติพื้นฐาน ของ PIC16F877



รูปที่ 2.9 แสดงขาต่างๆ ของไมโครคอนโทรลเลอร์ PIC16F877

- มีคำสั่ง 35 คำสั่ง
- ทำตามคำสั่งด้วยสัญญาณหนึ่งลูก (ยกเว้น คำสั่งกระโดด)
- ทำงานด้วยความถี่สัญญาณนาฬิกาตั้งแต่ ไฟตรง จนถึง 20 Mhz
- หน่วยความจำโปรแกรม 8k (14-Bit Words)
- หน่วยความจำข้อมูลแรมหรือรีจิสเตอร์ 368 ไบต์
- หน่วยความจำข้อมูลอีอีพรอม 256 ไบต์
- ตอบสนองสัญญาณอินเตอร์รัปต์ 15 แหล่ง
- วงจรเพาเวอร์ออนรีเซต (POR)
- สแต็ก 8 ระดับ
- เพาเวอร์อัปไทเมอร์ (PWRT)
- ออสซิลเลเตอร์สตาร์ตอัปไทเมอร์ (OST)
- วงจรวอตช์ด็อกไทเมอร์ (WDT) วงจรออสซิลเลเตอร์ในตัว
- เลือกระดับป้องการข้อมูลตัวโปรแกรม
- โหมดประหยัดพลังงาน
- สามารถโปรแกรมที่แรงดัน +5V

- แก้ไข้ตัวโปรแกรมในหน่วยความจำผ่านพอร์ตเพียง 2 ขา ด้วยกระบวนการ ICD : In-circuit Debugger

- ซีพียูสามารถอ่านและเขียนหน่วยความจำโปรแกรมได้

- กระแสซิงก์และซอร์ต 25mA

- แรงดัน +2 ถึง +5V

- ใช้พลังงานน้อยกว่า 2mA ที่แรงดัน +5V และความถี่ 4 Mhz 20uA ที่แรงดัน +3V และความถี่ 32khz น้อยกว่า 1uA ในโหมดประหยัดพลังงาน หรือสแตนด์บาย

คุณสมบัติพิเศษ

- มีไทเมอร์ทั้งหมด 3 ตัว คือ
- ไทเมอร์ 0 ขนาด 8 บิต ปรีสเกลเลอร์ 8 บิต ในตัว
- ไทเมอร์ 1 ขนาด 16 บิต ปรีสเกลเลอร์
- ไทเมอร์ 2 ขนาด 8 บิต ปรีสเกลเลอร์ โพสต์สเกลเลอร์ และรีจิสเตอร์คาบเวลา 8 บิต ในตัว
- โมดูล CCP 2 ชุด
- ตรวจจับสัญญาณ (Capture) ขนาด 16 บิต ความละเอียด 12.5 ns
- เปรียบเทียบสัญญาณ (Compare) ขนาด 16 บิต ความละเอียด 200 ns
- วงจร PWM ความละเอียด 10 บิต
- วงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอล 10 บิต 8 ช่อง
- วงจรเชื่อมต่ออุปกรณ์อนุกรมทั้ง SPI และ I2C
- วงจรสื่อสารข้อมูลอนุกรม (USART) พร้อมการตรวจจับแอดเดรส 9 บิต
- วงจรตรวจจับระดับแรงดันไฟเลี้ยง (Brown-out detection) เพื่อรีเซตซีพียู หรือ (BOR : Brown-out reset)

2.3 การส่งข้อมูลผ่านพอร์ตอนุกรม

การเคลื่อนย้ายข้อมูลจากคอมพิวเตอร์ไปยังอุปกรณ์ต่อพ่วงภายนอกหรือคอมพิวเตอร์ด้วยกัน มีด้วยกัน 2 รูปแบบคือ รับส่งข้อมูลแบบขนานและรับส่งข้อมูลแบบอนุกรม

การรับส่งข้อมูลแบบขนานเป็นการรับหรือส่งข้อมูลคราวละ 4 หรือ 8 บิตในเวลาเดียวกัน ทำให้การรับและส่งข้อมูลมีความเร็วสูง ทว่าจำนวนของสายที่ใช้ในการถ่ายทอดข้อมูลต้องมีมากเท่ากับจำนวนบิตของข้อมูลที่ทำการถ่ายทอดด้วย นอกจากนั้นยังมีสายที่ใช้สำหรับควบคุมและตรวจสอบการรับส่งข้อมูลด้วย ซึ่งอาจต้องใช้สายมากเป็น 2 เท่าของจำนวนบิตข้อมูลก็ได้ ส่งผลให้

ราคาของสายที่ใช้ในการเชื่อมต่อแบบขนานมักจะมีราคาแพง อีกข้อจำกัดหนึ่งของการถ่ายทอดข้อมูลแบบขนานคือ ระยะทางในการถ่ายทอดข้อมูล โดยปกติจะอยู่ที่ประมาณ 10-15 ฟุต

ในขณะที่การรับส่งข้อมูลแบบอนุกรมจะเป็นการรับส่งข้อมูลครั้งละ 1 บิต โดยมีรูปแบบการรับส่งที่เป็นมาตรฐาน ต้องมีการตรวจสอบความพร้อมในการรับและส่งข้อมูลของตัวส่งและตัวรับ การรับส่งข้อมูลแบบอนุกรมมีข้อดีในเรื่องของจำนวนสายสัญญาณที่น้อยมากและไม่แปรผันตามจำนวนบิตของข้อมูล ระยะทางในการรับส่งข้อมูลสูงกว่าแบบขนานมาก โดยปกติถ้าเป็นพอร์ตอนุกรม อาร์เอส-232ซี (RS-232C) จะสามารถต่อสายได้ยาว 50 ฟุตโดยประมาณ

2.3.1 การสื่อสารแบบอนุกรม

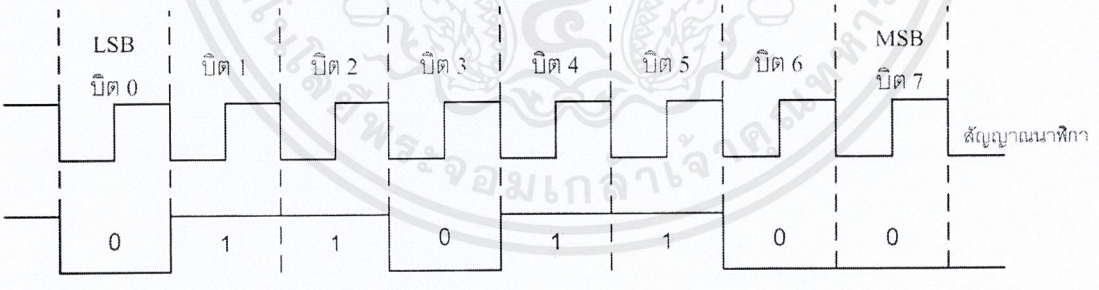
การสื่อสารแบบอนุกรมแบ่งได้เป็น 2 แบบคือ

1.การสื่อสารอนุกรมแบบซิงโครนัส

2.การสื่อสารอนุกรมแบบอะซิงโครนัส

2.3.1.1 การสื่อสารข้อมูลแบบซิงโครนัส

การสื่อสารแบบซิงโครนัสจะมีสัญญาณนาฬิการ่วมอยู่กับการรับและส่งสัญญาณด้วย ตัวอย่างการส่งข้อมูลแบบซิงโครนัสก็คือคีย์บอร์ดของคอมพิวเตอร์ ซึ่งสายเส้นหนึ่งจะเป็นสายของสัญญาณนาฬิกา ส่วนสายอีกเส้นจะเป็นสายของข้อมูล ดังนั้นการติดต่อกันแบบซิงโครนัสนี้จะต้องใช้สายในการเชื่อมต่ออย่างน้อยที่สุด 3 เส้นคือ สัญญาณนาฬิกา, ข้อมูลและกราวด์ รูปที่ 2.10 แสดงให้เห็นถึงไดอะแกรมเวลาของการสื่อสารข้อมูลแบบซิงโครนัส



รูปที่ 2.10 การสื่อสารแบบซิงโครนัส

2.3.1.2 การสื่อสารข้อมูลแบบอะซิงโครนัส

การสื่อสารข้อมูลแบบอะซิงโครนัสคือการรับและส่งข้อมูลโดยไม่จำเป็นต้องมีสัญญาณนาฬิการ่วมด้วย แต่จะใช้การกำหนดค่าอัตราเร็วในการรับและส่งข้อมูลให้มีค่าเท่ากัน ซึ่งเรียกอัตราเร็วนี้ว่า อัตราบอด หรือบอดเรต (baud rate) มีหน่วยเป็น บิตต่อวินาที (bit per second : bps)

บทที่ 2
ทฤษฎีและหลักการ

2.1 ระบบควบคุม

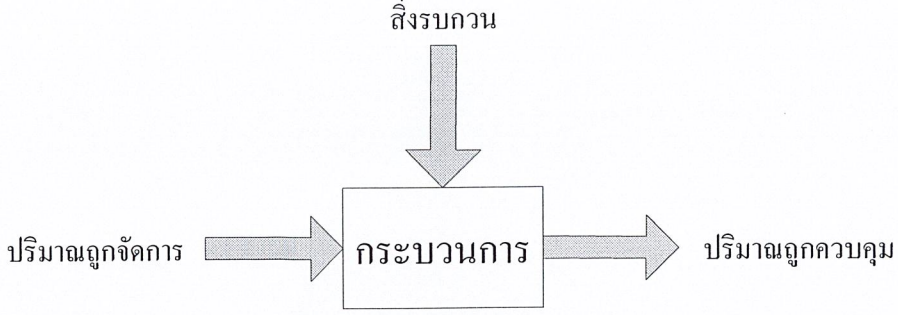
2.1.1 ประวัติการควบคุม

การควบคุมแบบป้อนกลับมีใช้ครั้งแรกเมื่อเจมส์วัตต์ได้ใช้ตัวบังคับแบบลูกหมุน (flyball governor) กับเครื่องจักรไอน้ำประมาณปี 1775 การใช้งานทฤษฎีที่เกี่ยวกับการควบคุมป้อนกลับส่วนมากต้องใช้ตัวบังคับ (Governors) รวมถึงการใช้งานอุตสาหกรรมด้วย การควบคุมอัตโนมัติเริ่มใช้กว้างขึ้นหลังจากทศวรรษ 1920 ทฤษฎีเกี่ยวกับการควบคุมอัตโนมัติถูกตีพิมพ์ครั้งแรกเมื่อ 1932 การเติบโตของการใช้งานในอุตสาหกรรมได้เข้าสู่ภาวะคงตัวและแข็งแกร่ง

เทคโนโลยีใหม่ๆ มากมายถูกนำมาใช้ควบคุมฮาร์ดแวร์ของกระบวนการ ขณะที่เทคนิคการวางระบบอัตโนมัติที่ใช้ในอุตสาหกรรมได้มีการพัฒนาและเจริญเต็มที่ในช่วงระยะเวลา 70 ปี ตัวอย่างการใช้งานที่สำคัญคือ การใช้คอมพิวเตอร์ดิจิตัล และไมโครโพรเซสเซอร์ในการควบคุมกระบวนการตั้งแต่ทศวรรษ 1960 ผลลัพธ์คือ การวางระบบอัตโนมัติในกระบวนการได้รับความสำคัญและแรงเสริมจากเทคโนโลยีต่างๆ ปัจจุบันอุตสาหกรรมส่วนมากได้จัดสรรเงินทุนมากกว่าร้อยละ 10 สำหรับเครื่องมือวัดและควบคุม และตัวเลขร้อยละนี้ ได้เพิ่มเป็นสองเท่าในช่วง 30 ปีที่ผ่านมา และไม่มีแนวโน้มว่าจะลดลงแต่อย่างใด

2.1.2 ตัวแปรที่เกี่ยวข้องกับการควบคุม

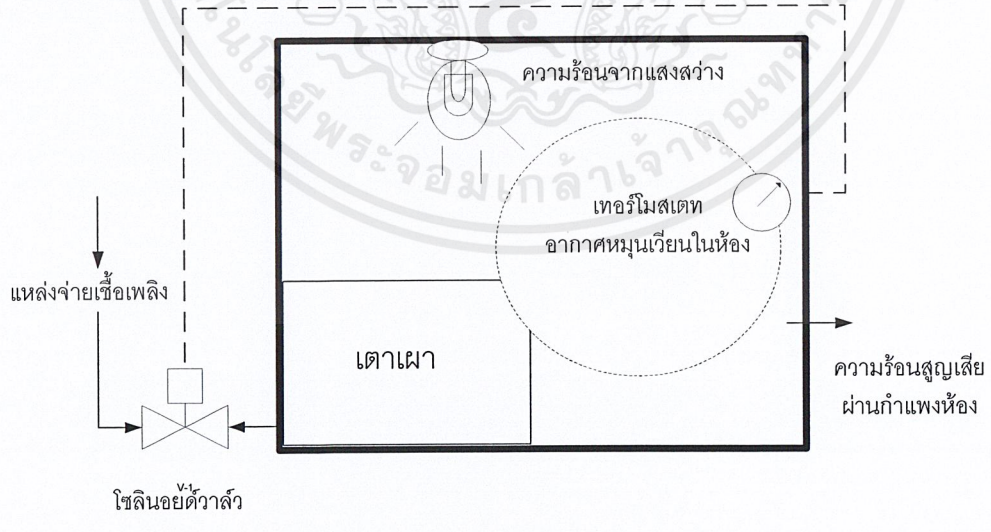
เพื่อการเข้าใจการควบคุมกระบวนการอัตโนมัติ จำเป็นต้องทำความเข้าใจกับตัวแปรสำคัญที่เกี่ยวข้องกับการควบคุมกระบวนการสามตัวแปรคือ ตัวแปรถูกควบคุม (Controlled Variable) หรือปริมาณที่ถูกควบคุม ตัวแปรจัดการ (Manipulated Variable) หรือปริมาณที่ถูกจัดการ ตัวแปรของการรบกวน ตัวแปรควบคุมจะเป็นเงื่อนไขที่ต้องการควบคุมหรือต้องการรักษาให้อยู่ในระดับที่ต้องการ ซึ่งอาจจะเป็นอัตราการไหล ระดับ ความดัน อุณหภูมิ การจัดอัตราส่วนหรืออะไรที่เป็นตัวแปรกระบวนการ โดยมีการตั้งเป้าหมายการควบคุมตัวแปรให้เป็นไปตามค่าที่ต้องการเรียกว่าค่าปรับตั้ง (Set Point) หรือค่าอินพุตอ้างอิง



รูปที่ 2.1 ตัวแปรที่เกี่ยวข้องกับการควบคุม

ในแต่ละตัวแปรถูกควบคุมจะมีตัวแปรจัดการรวมอยู่ด้วย ในกระบวนการควบคุมการไหล อัตราการไหลจะถูกจัดการผ่านตัววาล์วควบคุม เมื่อมีการรบกวนเข้ามาในกระบวนการจะทำให้ตัวแปรถูกควบคุมเบี่ยงเบนไปจากค่าเป้าหมายที่ต้องการ ระบบควบคุมอัตโนมัติต้องทำการปรับตัวแปรจัดการ เพื่อรักษาตัวแปรถูกควบคุมให้อยู่ที่ค่าปรับตั้งได้โดยไม่ถูกกระทบจากการรบกวน ในกรณีที่ต้องการเปลี่ยนเป้าหมายการควบคุมก็ต้องมีการปรับเลื่อนค่าตัวแปรจัดการใหม่ด้วย

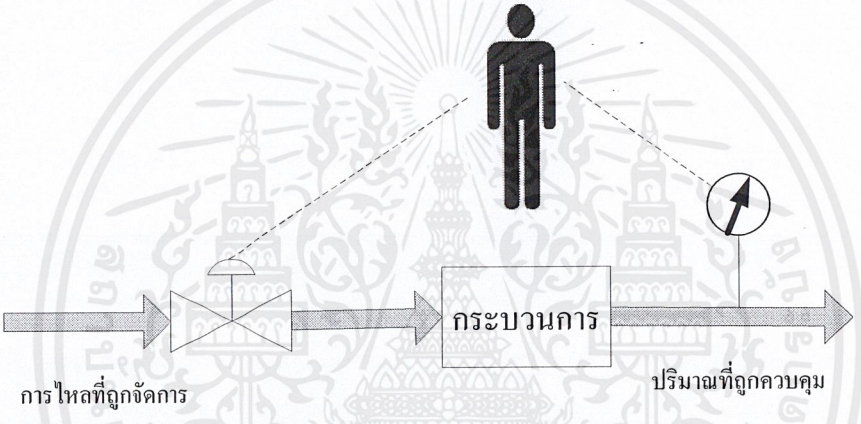
รูปที่ 2.2 แสดงระบบการให้ความร้อนภายในบ้าน ตัวแปรถูกควบคุมคือ อุณหภูมิของห้องที่ต้องการให้ความสบายและสามารถวัดค่าได้สะดวก การรบกวนที่ทำให้อุณหภูมิของห้องเปลี่ยนแปลงคือ อุณหภูมิจากภายนอกห้อง จำนวนผู้คนที่อยู่ในห้อง เครื่องใช้ไฟฟ้าที่กระจายความร้อนภายในห้อง ระบบควบคุมอัตโนมัติถูกออกแบบให้จัดการกับการไหลของเชื้อเพลิงที่จ่ายให้เตาผิง เพื่อรักษาอุณหภูมิห้องให้คงที่ตามต้องการไม่ให้แปรเปลี่ยนไปตามการรบกวน



รูปที่ 2.2 ระบบให้ความร้อนภายในบ้าน

2.1.3 การควบคุมด้วยมือ

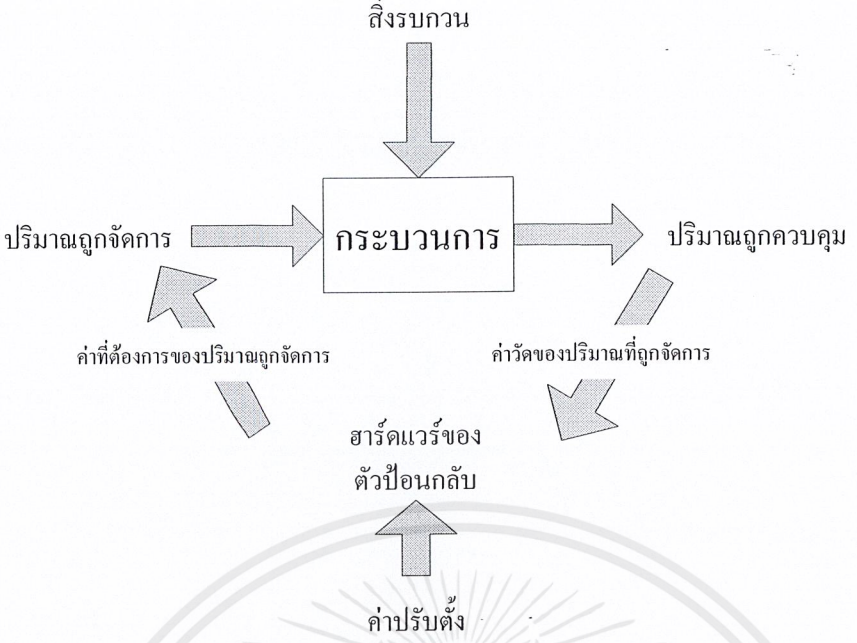
การควบคุมด้วยมือ (Typical Manual Control) เป็นกระบวนการที่มีตัวแปรถูกควบคุมตัวแปรเดียวอยู่ที่ทางไหลออกจากกระบวนการ มีตัวชี้บอก (Indicator) ที่แสดงค่าตัวแปรให้ผู้ควบคุมเครื่อง (Operator) อ่านค่าได้ตลอดเวลา ผู้ควบคุมเครื่องจะคาดการณ์ล่วงหน้าจากค่าที่อ่านได้นำไปปรับอัตราการไหลเข้ากระบวนการ เพื่อให้ได้ค่าตัวแปรถูกควบคุมเป็นไปตามเป้าหมายที่อยู่ในใจตลอดเวลา และผู้ควบคุมเครื่องเป็นผู้ตัดสินใจทุกอย่างเกี่ยวกับการควบคุม จะเห็นว่ามีปัญหาหลายอย่างในการควบคุมด้วยมือ หากผู้ควบคุมเครื่องมีความชำนาญในการคาดการณ์ล่วงหน้าไม่เพียงพอ ความเมื่อยล้าจากการเฝ้าดูอาการเป็นเวลานานๆ ซึ่งล้วนจะทำให้การควบคุมผิดพลาดไม่เป็นไปตามเป้าหมายการควบคุม



รูปที่ 2.3 การควบคุมด้วยมือ

2.1.4 การควบคุมป้อนกลับ

การควบคุมกระบวนการอัตโนมัติแบบง่ายๆ จะใช้วิธีการป้อนกลับซึ่งเป็นวิธีพื้นฐานที่มีใช้กันกว้างขวาง มีการติดตั้งตัวรับรู้ (Sensor) หรืออุปกรณ์วัด เพื่อวัดค่าจริงของตัวแปรถูกควบคุมแล้วส่งถ่ายไปยังฮาร์ดแวร์ที่ควบคุมป้อนกลับ โดยทำการเปรียบเทียบระหว่างค่าปรับตั้งหรือค่าที่ต้องของตัวแปรถูกควบคุมกับค่าที่วัดได้จริงจะได้ค่าผลต่างหรือค่าผิดพลาด (Errors) ที่ใช้ในการคำนวณค่าตัวแปรจัดการ เพื่อส่งไปปรับอุปกรณ์ที่จัดการกับอินพุตของกระบวนการ (ซึ่งมักจะเป็นวาล์วควบคุม) โดยอัตโนมัติ



รูปที่ 2.4 พื้นฐานการควบคุมป้อนกลับ

ข้อดีของการควบคุมป้อนกลับ คือ ผู้ออกแบบไม่จำเป็นต้องรู้ว่าจะมีสัญญาณรบกวนอะไรที่จะมากระทบกระบวนการ และไม่จำเป็นต้องรู้ค่าความสัมพันธ์ระหว่างสัญญาณรบกวนเหล่านั้นหรือผลกระทบท้ายสุดของสัญญาณรบกวนที่มีต่อตัวแปรถูกควบคุม

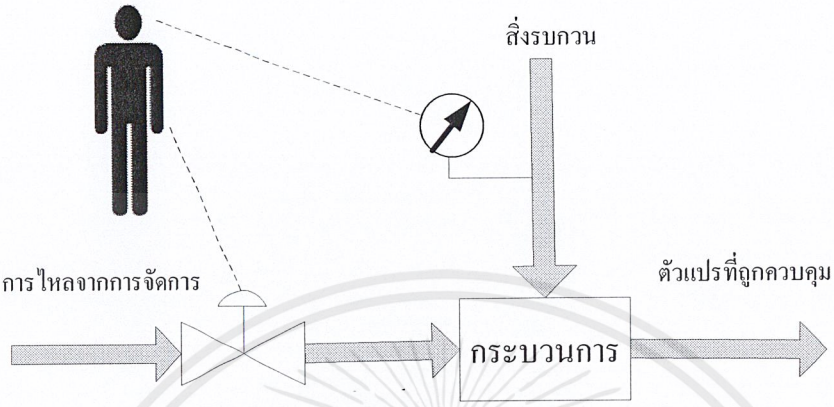
สิ่งที่ผู้ออกแบบต้องให้ความสำคัญคือ ฮาร์ดแวร์ที่ใช้ในวง และฮาร์ดแวร์แต่ละส่วนจะทำงานเข้ากันได้หรือไม่ โดยที่ยุทธวิธีการควบคุมแบบป้อนกลับยังเหมือนเดิมเสมอ การควบคุมป้อนกลับดังกล่าว เป็นเทคนิคการควบคุมกระบวนการอัตโนมัติ ซึ่งเป็นพื้นฐานที่ใช้กันกว้างขวางในอุตสาหกรรม

2.1.5 การควบคุมป้อนไปหน้าด้วยมือ

การควบคุมป้อนไปหน้ามีพื้นฐานแตกต่างจากการควบคุมป้อนกลับ ขณะที่มีสัญญาณรบกวนเข้าไปในกระบวนการผู้ควบคุมเครื่องจะเห็นการรบกวนตามธรรมชาตินี้จากการชี้บอก และทำการปรับตัวแปรจัดการ ตามสัญญาณรบกวนที่เข้ามาเพื่อป้องกันการเปลี่ยนแปลงใดๆ ที่จะเกิดท้ายที่สุด หรือป้องกันการเปลี่ยนแปลงในตัวแปรถูกควบคุมที่เกิดจากสัญญาณรบกวน การควบคุมป้อนกลับทำงานเพื่อกำจัดค่าผิดพลาด แต่การควบคุมป้อนไปหน้าให้ผลในการป้องกันค่าผิดพลาดจากที่เกิดขึ้นในตอนแรก ประโยชน์ของการควบคุมป้อนไปหน้าจึงเห็นได้ชัด

การควบคุมป้อนไปหน้าเพิ่มขีดความต้องการของผู้ปฏิบัติเป็นอย่างมาก ผู้ปฏิบัติต้องรู้ล่วงหน้าว่ามีสัญญาณรบกวนอะไรจะเข้ามาในกระบวนการ และต้องมีการตระเตรียมการวัดสัญญาณรบกวนเหล่านี้ อีกประการหนึ่งผู้ควบคุมในห้องควบคุม ต้องรู้แน่นอนว่าจะปรับตัวแปร

จัดการเมื่อไร และอย่างไร เพื่อชดเชยผลกระทบของสัญญาณรบกวนได้ถูกต้อง หากผู้ปฏิบัติมีความสามารถเฉพาะเหล่านี้ และหากพวกเขาพร้อมที่จะลงมือปฏิบัติ สัญญาณถูกควบคุมจะไม่เปลี่ยนไปจากค่าที่ต้องการหรือค่าปรับตั้ง



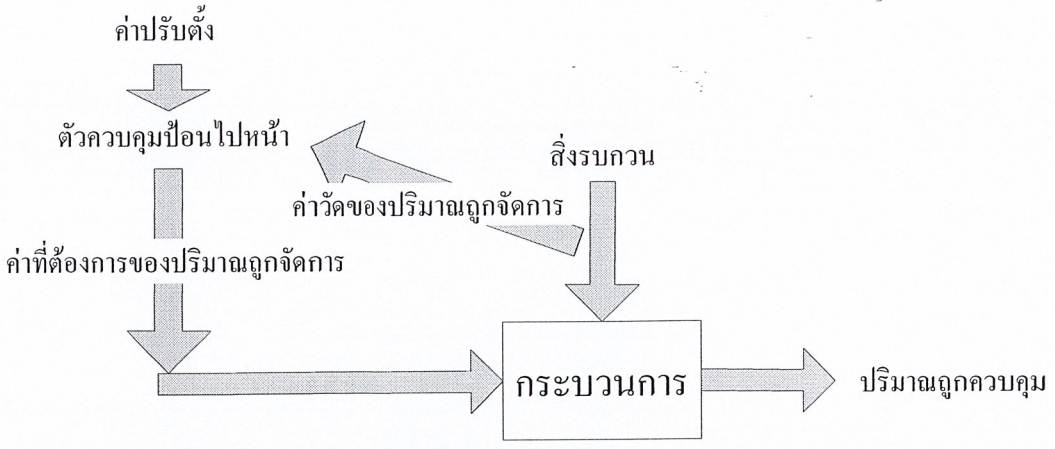
รูปที่ 2.5 การควบคุมป้อนไปหน้าด้วยมือ

ถ้าหากผู้ควบคุมเครื่องกระทำการผิดพลาดบางอย่างหรือไม่สามารถคาดคะเนสัญญาณรบกวนทั้งหมดที่อาจมีผลกระทบต่อกระบวนการ ตัวแปรถูกควบคุมจะเบี่ยงเบนจากค่าที่ต้องการได้ และการควบคุมป้อนไปหน้าอย่างเดียว จะมีค่าผิดพลาดที่ไม่สามารถแก้ไขได้เกิดขึ้น

2.1.6 การควบคุมป้อนไปหน้าอัตโนมัติ

รูปที่ 2.6 แสดงแนวคิดทั่วไปของการควบคุมป้อนไปหน้าอัตโนมัติ มีสัญญาณรบกวนป้อนเข้ากระบวนการ และมีตัวรับรู้คอยวัดสัญญาณรบกวนเหล่านี้ ตัวควบคุมป้อนไปหน้า (Feedforward Controller) จะคำนวณค่าของตัวแปรจัดการที่ต้องการตามสัญญาณรบกวนที่วัดได้หรือรับรู้ได้ ค่าปรับตั้งที่แสดงค่าตัวแปรถูกควบคุมที่ต้องการได้ถูกป้อนให้กับตัวควบคุมป้อนไปหน้า

จะเห็นได้ชัดว่าตัวควบคุมป้อนไปหน้าต้องทำการคำนวณสลับซับซ้อนมาก การคำนวณนี้ต้องสะท้อนและเข้าใจผลกระทบของสัญญาณรบกวนที่มีต่อตัวแปรถูกควบคุมได้อย่างถูกต้อง ด้วยความเข้าใจดังกล่าว ตัวควบคุมป้อนไปหน้าสามารถคำนวณผลลัพธ์ของขนาดตัวแปรจัดการที่ใช้ชดเชยสัญญาณรบกวนได้แน่นอน การคำนวณนี้สามารถอธิบายความเข้าใจเฉพาะของผลกระทบของตัวแปรจัดการที่มีผลต่อตัวแปรถูกควบคุมได้ ถ้าหากความสัมพันธ์ทางคณิตศาสตร์ทั้งหมดได้มาพร้อม ตัวควบคุมก็สามารถคำนวณการเปลี่ยนแปลงตัวแปรจัดการที่จำเป็นต่อการชดเชยการเปลี่ยนแปลงสัญญาณรบกวนได้อย่างอัตโนมัติ

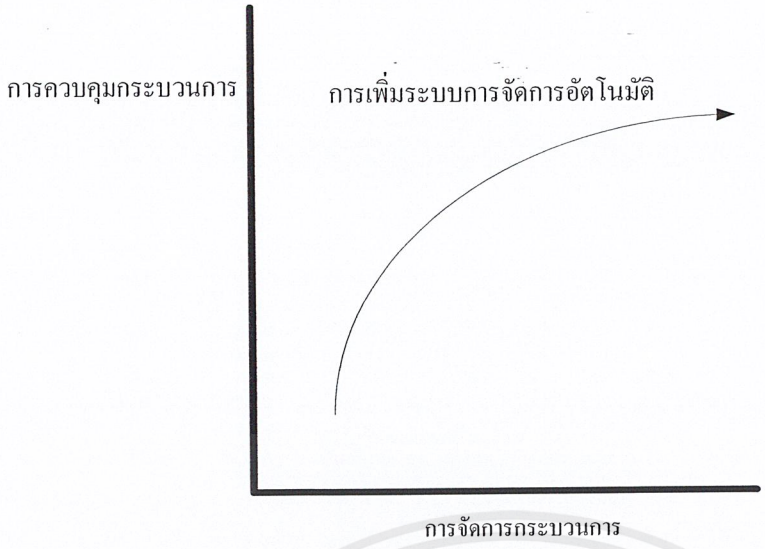


รูปที่ 2.6 พื้นฐานการควบคุมป้อนไปหน้า

การเพิ่มความเข้าใจทางทฤษฎีเป็นสิ่งที่ต้องการ การควบคุมป้อนไปหน้าเป็นแนวคิดที่ใช้ประโยชน์ได้มาก มีการเพิ่มหลักวิชาการและความต้องการทางวิศวกรรมของทั้งผู้ออกแบบและผู้ปฏิบัติ ผลลัพธ์คือ การควบคุมป้อนไปหน้าจึงมีไว้สำหรับใช้กับวงควบคุมที่สำคัญในกระบวนการบางวงเท่านั้น ขณะที่การใช้งานมีไม่มาก แต่ก็มีความสำคัญมาก

2.1.7 การควบคุมกระบวนการและการจัดการกระบวนการ

การวางระบบกระบวนการอัตโนมัติ เพื่อได้มาซึ่งผลประโยชน์สูงสุดของกระบวนการ ในหัวข้อก่อนของเรื่องนี้ได้สมมติว่ารู้ค่าปริมาณที่ต้องการควบคุม หากเมื่อใดรู้ค่าที่ต้องการนี้ ต้องใช้เทคนิคการวางระบบอัตโนมัติ เพื่อบรรลุหรือรักษาค่าที่ต้องการเหล่านี้ หรือค่าปรับตั้ง อย่างไรก็ตาม มีคำถามที่สำคัญบางคำถามเกี่ยวกับผลประโยชน์ของกระบวนการ ซึ่งต้องตอบว่าค่าที่ต้องการเป็นเท่าไร อันนี้เป็นพื้นฐานของงานการจัดการ และมักปล่อยให้ผู้ควบคุมเครื่องเป็นผู้พิจารณา แต่ในไม่กี่ปีที่ผ่านมามีความก้าวหน้าที่สำคัญทางการวางระบบกระบวนการอัตโนมัติ ทำให้งานการจัดการต่างๆ ได้กลายเป็นอัตโนมัติภายในตัวของมัน และความสามารถในการบรรลุถึงผลลัพธ์ทางหลักวิชาการและไขปัญหาทางฮาร์ดแวร์สำหรับคำถามการจัดการที่สำคัญของภาพการควบคุม



รูปที่ 2.7 การควบคุมและการจัดการกระบวนการ

ในกระบวนการพิเศษ ขณะที่ระดับการวางระบบอัตโนมัติได้เพิ่มขึ้น ขั้นตอนเริ่มแรกส่วนใหญ่ยังใช้การควบคุมกระบวนการแบบดั้งเดิม (conventional control) เช่น การควบคุมป้อนกลับ อย่างไรก็ตาม ขณะที่ระดับการวางระบบอัตโนมัติเพิ่มขึ้น การวางระบบอัตโนมัติส่วนมาก จะเกี่ยวข้องกับการจัดการกระบวนการซึ่งได้แสดงในรูปที่ 2.7

การรวมกันของปรากฏการทั้งสอง (การควบคุมกระบวนการและการจัดการกระบวนการ) ต้องถูกสะท้อนเข้าไปในภาพรวมของความเข้าใจ และเล็งเห็นถึงความสำคัญของการวางระบบกระบวนการอัตโนมัติ

2.2 ไมโครคอนโทรลเลอร์ PIC16F877

รูปที่ 2.8 ไมโครคอนโทรลเลอร์ PIC16F877

คุณสมบัติที่น่าสนใจ

- มีหน่วยความจำโปรแกรมแบบแฟลช เขียนโปรแกรมใหม่ได้นับแสนครั้ง

- มีหน่วยความจำข้อมูลอีอีพรอมที่บันทึกข้อมูลใหม่ได้นับล้านครั้ง คือเป็นการแยกหน่วยความจำโปรแกรมและหน่วยความจำข้อมูลออกจากกัน และมีบัสสำหรับติดต่อแยกกันด้วย โดยมีโครงสร้างดังนี้ โครงสร้างหลักคือ ซีพียู, หน่วยความจำโปรแกรม, หน่วยความจำข้อมูล, ส่วนติดต่ออินพุต - เอาต์พุต, วงจรสัญญาณนาฬิกา, วงจรรีเซตหลัก, วงจรไฟเลี้ยง, และส่วนจัดการอินเตอร์รัปต์ และยังมีโมดูลพิเศษต่าง ๆ เพื่อเข้ามา เช่น

- วงจรบราวเอาต์รีเซต (brown-out reset)

- ส่วนแก้ไขข้อมูลในวงจร หรือดีบักเกอร์ (In-circuit debugger)

- วงจรโปรแกรมแรงดันค่า (low-voltage programming)

- ไทเมอร์ 3 ตัว

- วงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอล 10 บิต

- โมดูลเชื่อมต่ออุปกรณ์อนุกรม (SPI : Serial Peripheral Interfacing)

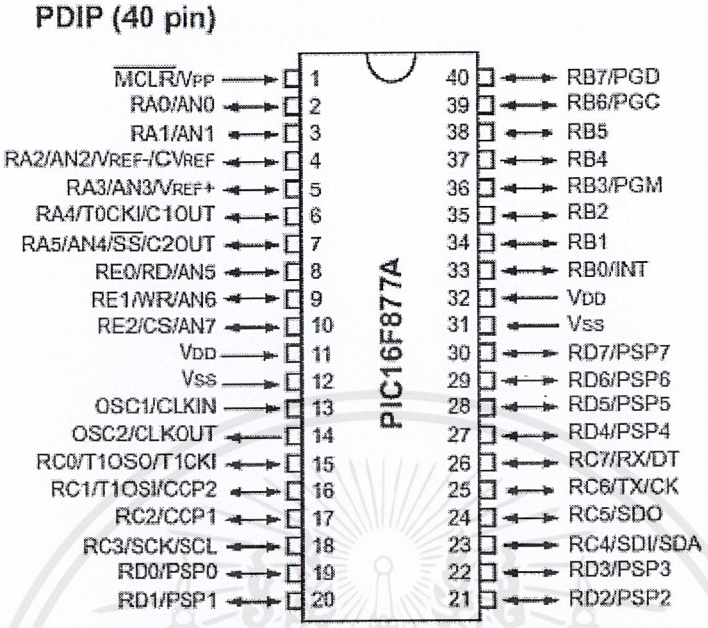
- โมดูลเชื่อมต่ออุปกรณ์ระบบบัส I2C

- โมดูลสื่อสารอนุกรม (USART : Universal Synchronous Asynchronous Receiver Transmitter)

- โมดูลเปรียบเทียบสัญญาณ - ตรวจจับสัญญาณ – วงจรมอดูเลชั่นทางความกว้างของพัลส์ (CCP : Compare Capture Pulse-width modulation)

- หากเป็นเบอร์ PIC16F877 จะมีวงจรเปรียบเทียบแรงดันอะนาลอก และโมดูลสร้างแรงดันอ้างอิงเพิ่มเข้ามาด้วย

คุณสมบัติพื้นฐาน ของ PIC16F877



รูปที่ 2.9 แสดงขาต่างๆ ของไมโครคอนโทรลเลอร์ PIC16F877

- มีคำสั่ง 35 คำสั่ง
- ทำตามคำสั่งด้วยสัญญาณหนึ่งลูก (ยกเว้น คำสั่งกระโดด)
- ทำงานด้วยความถี่สัญญาณนาฬิกาตั้งแต่ ไฟตรง จนถึง 20 Mhz
- หน่วยความจำโปรแกรม 8k (14-Bit Words)
- หน่วยความจำข้อมูลแรมหรือรีจิสเตอร์ 368 ไบต์
- หน่วยความจำข้อมูลอีอีพรอม 256 ไบต์
- ตอบสนองสัญญาณอินเตอร์รัปต์ 15 แหล่ง
- วงจรเพาเวอร์ออนรีเซต (POR)
- สแต็ก 8 ระดับ
- เพาเวอร์อัปไทเมอร์ (PWRT)
- ออสซิลเลเตอร์สตาร์ตอัปไทเมอร์ (OST)
- วงจรวอตช์ด็อกไทเมอร์ (WDT) วงจรออสซิลเลเตอร์ในตัว
- เลือกระดับป้องการข้อมูลตัวโปรแกรม
- โหมดประหยัดพลังงาน
- สามารถโปรแกรมที่แรงดัน +5V

- แก้ไข้ตัวโปรแกรมในหน่วยความจำผ่านพอร์ตเพียง 2 ขา ด้วยกระบวนการ ICD : In-circuit Debugger

- ซีพียูสามารถอ่านและเขียนหน่วยความจำโปรแกรมได้

- กระแสซิงก์และซอร์ต 25mA

- แรงดัน +2 ถึง +5V

- ใช้พลังงานน้อยกว่า 2mA ที่แรงดัน +5V และความถี่ 4 Mhz 20uA ที่แรงดัน +3V และความถี่ 32khz น้อยกว่า 1uA ในโหมประหยัดพลังงาน หรือสแตนด์บาย

คุณสมบัติพิเศษ

- มีไทเมอร์ทั้งหมด 3 ตัว คือ

- ไทเมอร์ 0 ขนาด 8 บิต ปรีสเกลเลอร์ 8 บิต ในตัว

- ไทเมอร์ 1 ขนาด 16 บิต ปรีสเกลเลอร์

- ไทเมอร์ 2 ขนาด 8 บิต ปรีสเกลเลอร์ โพสต์สเกลเลอร์ และรีจิสเตอร์คาบเวลา 8 บิต ในตัว

- โมดูล CCP 2 ชุด

- ตรวจจับสัญญาณ (Capture) ขนาด 16 บิต ความละเอียด 12.5 ns

- เปรียบเทียบสัญญาณ (Compare) ขนาด 16 บิต ความละเอียด 200 ns

- วงจร PWM ความละเอียด 10 บิต

- วงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอล 10 บิต 8 ช่อง

- วงจรเชื่อมต่ออุปกรณ์อนุกรมทั้ง SPI และ I2C

- วงจรสื่อสารข้อมูลอนุกรม (USART) พร้อมการตรวจจับแอดเดรส 9 บิต

- วงจรตรวจจับระดับแรงดันไฟเลี้ยง (Brown-out detection) เพื่อรีเซตซีพียู หรือ (BOR : Brown-out reset)

2.3 การส่งข้อมูลผ่านพอร์ตอนุกรม

การเคลื่อนย้ายข้อมูลจากคอมพิวเตอร์ไปยังอุปกรณ์ต่อพ่วงภายนอกหรือคอมพิวเตอร์ด้วยกัน มีด้วยกัน 2 รูปแบบคือ รับส่งข้อมูลแบบขนานและรับส่งข้อมูลแบบอนุกรม

การรับส่งข้อมูลแบบขนานเป็นการรับหรือส่งข้อมูลคราวละ 4 หรือ 8 บิตในเวลาเดียวกัน ทำให้การรับและส่งข้อมูลมีความเร็วสูง ทว่าจำนวนของสายที่ใช้ในการถ่ายทอดข้อมูลต้องมีมากเท่ากับจำนวนบิตของข้อมูลที่ทำการถ่ายทอดด้วย นอกจากนั้นยังมีสายที่ใช้สำหรับควบคุมและตรวจสอบการรับส่งข้อมูลด้วย ซึ่งอาจต้องใช้สายมากเป็น 2 เท่าของจำนวนบิตข้อมูลก็ได้ ส่งผลให้

ราคาของสายที่ใช้ในการเชื่อมต่อแบบขนานมักจะมีราคาแพง อีกข้อจำกัดหนึ่งของการถ่ายทอดข้อมูลแบบขนานคือ ระยะทางในการถ่ายทอดข้อมูล โดยปกติจะอยู่ที่ประมาณ 10-15 ฟุต

ในขณะที่การรับส่งข้อมูลแบบอนุกรมจะเป็นการรับส่งข้อมูลครั้งละ 1 บิต โดยมีรูปแบบการรับส่งที่เป็นมาตรฐาน ต้องมีการตรวจสอบความพร้อมในการรับและส่งข้อมูลของตัวส่งและตัวรับ การรับส่งข้อมูลแบบอนุกรมมีข้อดีในเรื่องของจำนวนสายสัญญาณที่น้อยมากและไม่แปรผันตามจำนวนบิตของข้อมูล ระยะทางในการรับส่งข้อมูลสูงกว่าแบบขนานมาก โดยปกติถ้าเป็นพอร์ตอนุกรม อาร์เอส-232ซี (RS-232C) จะสามารถต่อสายได้ยาว 50 ฟุตโดยประมาณ

2.3.1 การสื่อสารแบบอนุกรม

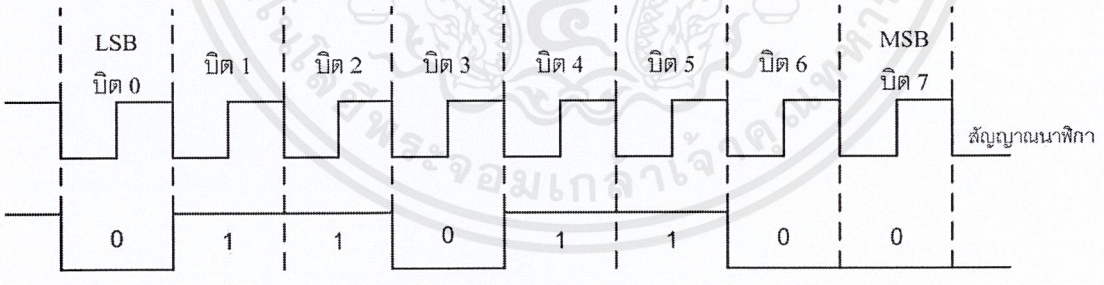
การสื่อสารแบบอนุกรมแบ่งได้เป็น 2 แบบคือ

1.การสื่อสารอนุกรมแบบซิงโครนัส

2.การสื่อสารอนุกรมแบบอะซิงโครนัส

2.3.1.1 การสื่อสารข้อมูลแบบซิงโครนัส

การสื่อสารแบบซิงโครนัสจะมีสัญญาณนาฬิการ่วมอยู่กับการรับและส่งสัญญาณด้วย ตัวอย่างการส่งข้อมูลแบบซิงโครนัสก็คือคีย์บอร์ดของคอมพิวเตอร์ ซึ่งสายเส้นหนึ่งจะเป็นสายของสัญญาณนาฬิกา ส่วนสายอีกเส้นจะเป็นสายของข้อมูล ดังนั้นการติดต่อกันแบบซิงโครนัสนี้จะต้องใช้สายในการเชื่อมต่ออย่างน้อยที่สุด 3 เส้นคือ สัญญาณนาฬิกา, ข้อมูลและกราวด์ รูปที่ 2.10 แสดงให้เห็นถึงไดอะแกรมเวลาของการสื่อสารข้อมูลแบบซิงโครนัส



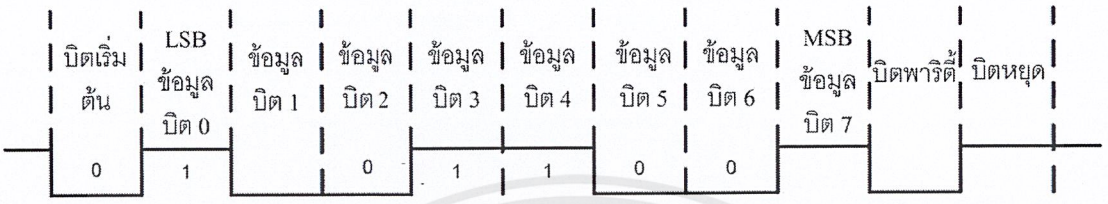
รูปที่ 2.10 การสื่อสารแบบซิงโครนัส

2.3.1.2 การสื่อสารข้อมูลแบบอะซิงโครนัส

การสื่อสารข้อมูลแบบอะซิงโครนัสคือการรับและส่งข้อมูลโดยไม่จำเป็นต้องมีสัญญาณนาฬิการ่วมด้วย แต่จะใช้การกำหนดค่าอัตราเร็วในการรับและส่งข้อมูลให้มีค่าเท่ากัน ซึ่งเรียกอัตราเร็วนี้ว่า อัตราบอด หรือบอดเรต (baud rate) มีหน่วยเป็น บิตต่อวินาที (bit per second : bps)

รูปแบบของข้อมูลที่ใช้ในการรับส่งแบบอะซิงโครนัสประกอบด้วย 4 ส่วนด้วยกันคือ

1. บิตเริ่มต้น (start bit) มีขนาด 1 บิต
2. บิตข้อมูลแบบอนุกรม มีขนาด 5,6,7 หรือ 8 บิต
3. บิตตรวจสอบพาริตี้ (parity bit) มีขนาด 1 บิตหรือไม่มี
4. บิตปิดท้ายหรือบิตหยุด (stop bit) มีขนาด 1, 1.5 หรือ 2 บิต



รูปที่ 2.11 การสื่อสารข้อมูลแบบอะซิงโครนัส

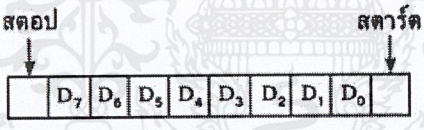
แสดงรูปแบบของข้อมูลอนุกรมแบบอะซิงโครนัส เมื่อไม่มีการส่งข้อมูล ขา DATA จะมีสถานะลอจิก "1" เรียกสถานะนี้ว่า สถานะหยุดรอ (waiting stage) การเริ่มต้นส่งข้อมูลจะเริ่มจากการให้ขา DATA มีลอจิก "0" ด้วยช่วงระยะเวลา 1 บิต เรียกบิตนี้ว่า บิตเริ่มต้นจากนั้นบิตข้อมูลจะถูกส่งออกไป โดยเริ่มจากบิตที่มีนัยสำคัญต่ำสุดหรือบิต LSB ก่อน ซึ่งข้อมูลที่ต้องการส่งอาจมีจำนวน 5,6,7 หรือ 8 บิตก็ได้ จากนั้นตามด้วยบิตพาริตี้ซึ่งใช้ในการตรวจสอบความผิดพลาดที่เกิดขึ้นจากการส่งข้อมูล บิตสุดท้ายที่จะส่งคือ บิตปิดท้ายหรือบิตหยุดโดยจะเป็นการทำให้ขา DATA มีสถานะลอจิก "1" อีกครั้งด้วยระยะเวลาอย่างน้อย 1 บิต,1.5 บิต หรือ 2 บิต เพื่อเป็นการแสดงว่าสิ้นสุดข้อมูลแล้ว

อัตราความเร็วในการรับและส่งข้อมูลของการรับส่งข้อมูลแบบอะซิงโครนัสหรืออัตราบอดที่ใช้สำหรับพอร์ตอนุกรม อาร์เอส-232 มีด้วยกันหลายค่า ได้แก่ 110, 150, 300, 600, 1,200, 2,400, 4,800, 9,600 และ 19,200 บิตต่อวินาที โดยมีค่าเพิ่มมากขึ้นตามเทคโนโลยีของคอมพิวเตอร์ เนื่องจากอัตราบอดคือค่าของจำนวนบิตที่สามารถส่งได้ใน 1 วินาที สมมติว่า ข้อมูลอนุกรมมีขนาด 8 บิต ไม่มีการตรวจสอบพาริตี้ มีบิตเริ่มต้น 1 บิต และบิตปิดท้าย 1 บิต ความยาวของข้อมูล 1 ไบต์จะมีความยาวเท่ากับ 10 บิต ถ้าใช้อัตราบอดในการส่งข้อมูลเท่ากับ 9,600 บิตต่อวินาที ก็จะสามารถรับส่งข้อมูลได้ด้วยความเร็ว 960 ไบต์ต่อวินาที

ตารางที่ 2.1 ค่าอัตราบอดและอัตราการเร็วในการส่งข้อมูล

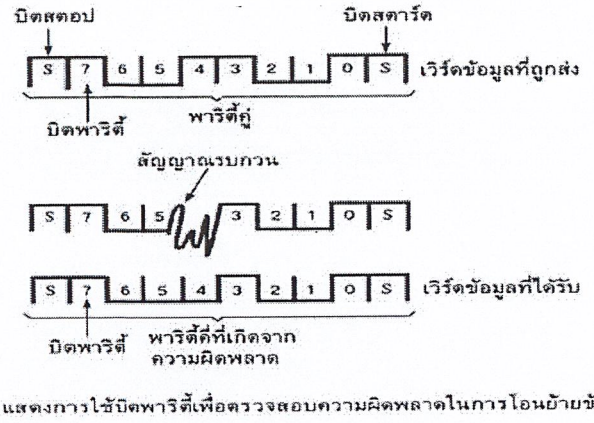
Baud Rate	Bytes/Second
110	10
150	15
300	30
600	60
1200	120
2400	240
4800	480
9600	960
19200	1920
38400	3840

จากตารางพบว่าในการส่งข้อมูลด้วยอัตราบอด 110 จะมีรูปแบบต่างจากอัตราอื่น ๆ คือ จะใช้บิตหยุด 2 บิต ดังนั้นจึงต้องส่งข้อมูลที่มีขนาด 11 บิต



รูปที่ 2.12 การส่งข้อมูลขนาด 8 บิตที่ใช้ในการโอนย้ายข้อมูลแบบอนุกรม

การตรวจสอบพาริตี้สามารถกำหนดให้เป็นแบบคี่ (odd), แบบคู่ (even) หรือไม่มีการตรวจสอบพาริตี้ก็ได้ พาริตี้คี่หรือพาริตี้คู่แสดงถึงจำนวนลอจิก "1" ทั้งหมดภายในข้อมูลที่ส่งไป 1 ไบต์รวมบิตพาริตี้ว่ามีจำนวนเป็นเลขคู่หรือเลขคี่ ยกตัวอย่าง ข้อมูลที่จะทำการส่งมีขนาด 8 บิต มีค่าเท่ากับ 99H หรือ 10011001B จะเห็นว่าข้อมูลในไบต์นี้มีจำนวนลอจิก "1" จำนวน 4 ตัวซึ่งเป็นเลขคู่ ดังนั้นถ้ากำหนดค่าพาริตี้เป็นคู่ ค่าของบิตพาริตี้จะต้องมีลอจิกเป็น "0" แต่ถ้ากำหนดพาริตี้เป็นคี่ ค่าของบิตพาริตี้จะต้องเป็น "1" เพื่อให้ข้อมูล 1 ไบต์รวมทั้งบิตพาริตี้เป็นคี่ ถ้าข้อมูลมีค่าพาริตี้ ไม่ตรงตามที่ กำหนดบิตพาริตี้ในรีจิเตอร์สถานะของ UART (Universal Asynchronous Receiver Transmitter) ก็จะถูกเซตเพื่อแสดงว่าข้อมูลที่รับมาผิดพลาดและโปรแกรมที่ทำการรับข้อมูลนั้นก็จะขอให้มีการส่งข้อมูลมาใหม่



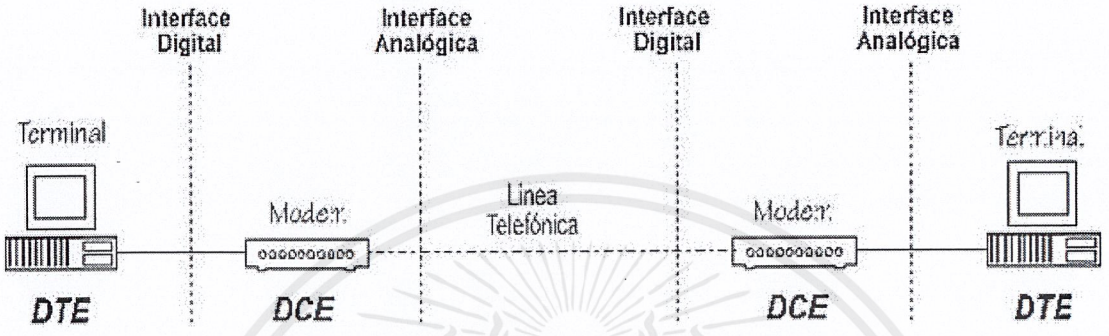
รูปที่ 2.13 การใช้บิตพาริตี้เพื่อตรวจสอบความผิดพลาดในการส่งข้อมูลแบบอนุกรม

การใช้พาริตี้นี้จะสามารถตรวจสอบความผิดพลาด ได้เพียง 50 เปอร์เซ็นต์ของความผิดพลาดทั้งหมดทั้งนี้เนื่องจากมันจะจับความผิดพลาดได้เฉพาะกรณีที่ข้อมูลนั้นเกิดข้อผิดพลาดเป็นจำนวนคี่บิต ถ้าข้อมูลนั้นเกิดความผิดพลาดเป็นจำนวนคู่บิตก็จะไม่ทำให้ค่าพาริตี้เปลี่ยนแปลงความผิดพลาดก็จะไม่ถูกตรวจพบ UART ยังสามารถตรวจสอบ framing error ได้ framing error จะเกิดขึ้นเมื่อ UART ได้รับข้อมูลที่มีบิตเริ่มต้นและบิตหยุดในตำแหน่งที่ไม่ถูกต้องซึ่ง บิตพาริตี้ถูกสร้างขึ้นจากภาคส่งข้อมูลของ UART เป็นอุปกรณ์ที่ใช้ในการรับและส่งข้อมูลอนุกรม (ซึ่งจะกล่าวถึงในรายละเอียดภายหลัง) ซึ่งทางภาครับจะต้องกำหนดคุณสมบัติการตรวจสอบพาริตี้ที่ตรงกันเอาไว้ว่าจะตรวจสอบพาริตี้คี่หรือพาริตี้คู่ จากนั้นภาครับของ UART จะทำการตรวจสอบค่าพาริตี้ที่เกิดขึ้นว่าเป็นคู่หรือเป็นคี่ โดยการนับจำนวนลอจิก 1 ทั้งหมดรวมทั้งบิตพาริตี้ด้วย ถ้ากำหนดพาริตี้ไว้เป็นคู่แต่อ่านค่าตัวเลขในการนับออกมาได้ตัวเลขเป็นคี่ ทางภาครับจะแสดงข้อผิดพลาดออกมาให้ผู้ใช้ทราบ กระบวนการดังกล่าวเป็นวิธีการตรวจสอบความผิดพลาดที่เกิดขึ้นในการรับส่งข้อมูลที่ง่ายที่สุด แต่มันสามารถตรวจสอบได้เมื่อมีบิตข้อมูลที่ทำการรับส่งผิดพลาดเพียงบิตเดียวเท่านั้น

2.3.2 มาตรฐานพอร์ตอนุกรมแบบอาร์เอส-232 (RS-232)

มาตรฐานการเชื่อมต่อแบบอนุกรมอาร์เอส-232 เป็นมาตรฐานอุตสาหกรรมที่ออกแบบมาเพื่อใช้ในการส่งข้อมูลอนุกรมแบบอะซิงโครนัส 2 ทิศทาง โดยมาตรฐาน อาร์เอส-232 ในอดีตนั้นถูกออกแบบมาเพื่อการส่งผ่านข้อมูลจากคอมพิวเตอร์ไปยังโมเด็มเพียงอย่างเดียว เพื่อที่จะนำข้อมูลจากโมเด็มนี้ส่งผ่านสายโทรศัพท์ไปยังคอมพิวเตอร์อีกชุดซึ่งอยู่ห่างไกลกัน โดยสมาคมอุตสาหกรรมอิเล็กทรอนิกส์ (Electronic Industries Association : EIA) ได้วางมาตรฐานที่มีชื่อเรียกกันว่า อีไอเอ อาร์เอส-232 (EIA RS-232) มาตรฐานนี้ในช่วงแรกจะใช้คอนเน็กเตอร์เป็นแบบ DB-25 โดยกำหนดความยาวสูงสุดของสายสัญญาณไว้ที่ 50 ฟุต มีระดับสัญญาณตั้งแต่ -3 โวลต์

จนถึง -12 โวลต์ แสดงว่า มีข้อมูล (mark) และ +3 โวลต์ ถึง +12 โวลต์ แสดงว่าเป็น ช่องว่าง (space) มาตรฐาน อาร์เอส-232 ถูกใช้ในการกำหนดรูปแบบการสื่อสารข้อมูลกันระหว่างอุปกรณ์เชื่อมต่อข้อมูล (Data Terminal Equipment: DTE) ซึ่งเป็นอุปกรณ์สำหรับส่งข้อมูล (เอาต์พุต) กับวงจรข้อมูลปลายทาง (Data Circuit Terminating: DCE) ซึ่งเป็นอุปกรณ์สำหรับรับข้อมูล (อินพุต)



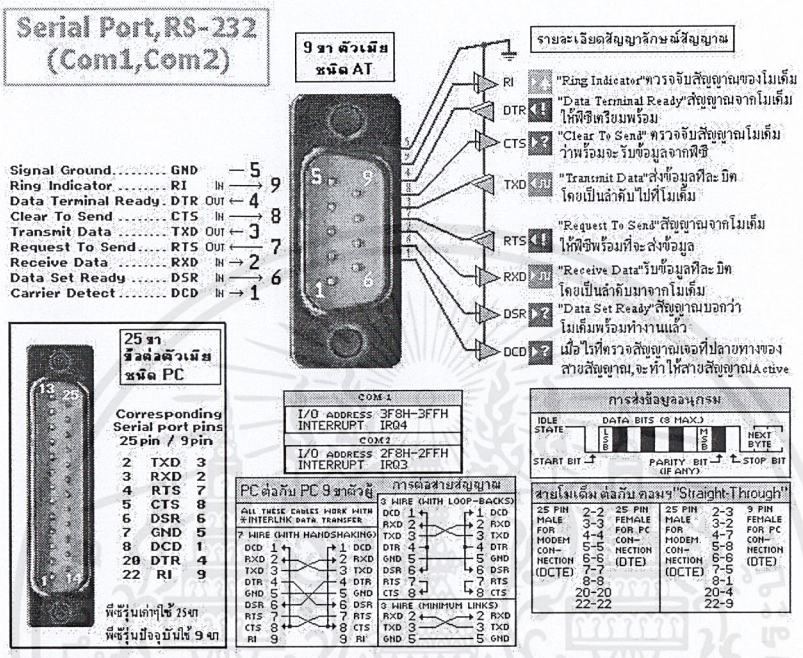
รูปที่ 2.14 DTE และ DCE

อุปกรณ์ DTE จะต้องเป็นอุปกรณ์ที่มีการประมวลผลในตัวเช่น ไมโครคอนโทรลเลอร์หรือไมโครคอมพิวเตอร์ ซึ่งมีความสามารถในการสร้างบิตข้อมูลแบบอนุกรมได้ ส่วนอุปกรณ์ DCE ทำหน้าที่เป็นเพียงตัวรับข้อมูลที่ส่งมาจาก DTE เท่านั้น

ข้อแตกต่างของอุปกรณ์ DTE และอุปกรณ์ DCE อย่างหนึ่งที่เห็นได้ชัดคือ คอนเน็กเตอร์ของ DTE จะเป็นตัวผู้ ส่วนคอนเน็กเตอร์ของ DCE จะเป็นตัวเมีย ซึ่งพอร์ตอนุกรมของคอมพิวเตอร์ที่ใช้กันอยู่ทั่วไปจะเป็นแบบ DTE ส่วนคอนเน็กเตอร์ที่อยู่ที่โมเด็มจะเป็นแบบ DCE สำหรับการใช้งานในคอมพิวเตอร์ พอร์ตอนุกรม อาร์เอส-232 ถูกใช้เพื่อเชื่อมต่อกับโมเด็ม, เมาส์ และเครื่องพิมพ์ที่มีสามารถติดต่อทางพอร์ตอนุกรมได้

2.3.3 คอนเน็กเตอร์สำหรับพอร์ตอาร์เอส-232 และการเชื่อมต่อ

มาตรฐานการเชื่อมต่อแบบอาร์เอส-232 จะใช้คอนเน็กเตอร์แบบ DB-25 ตัวผู้ หรือ DB-9 ตัวผู้ ซึ่งคอนเน็กเตอร์แบบ DB-25 จะมีขาต่อใช้งาน เพียง 9 เส้นเช่นเดียวกับคอนเน็กเตอร์แบบ DB-9 เนื่องจากขาอื่นๆ ที่เคยมีการใช้งานมาในอดีตไม่ค่อยมีสำคัญมากนักจึงถูกยกเลิกไป



รูปที่ 2.15 แสดงการจัดขาคอนเน็กเตอร์พอร์ตอนุกรมและลักษณะการต่อกับอุปกรณ์ภายนอก

1.ขา Data Carrier Detect : DCD หรืออาจเรียกว่า Carrier Detect : CD ขานี้จะแอกตีฟเมื่อมีการส่งสัญญาณพาหะจากอุปกรณ์สื่อสารข้อมูล เช่น โมเด็ม สำหรับการใช้งานปกติ ขานี้จะไม่ได้ถูกใช้งานมากนัก

2.ขา Receive Data : RD หรือ RxD ขานี้ใช้เพื่อรับสัญญาณอนุกรมเข้ามายังคอมพิวเตอร์ โดยจะนำข้อมูลที่อ่านได้ไปเก็บไว้ในรีจิสเตอร์บัฟเฟอร์

3.ขา Transmitted Data : TD หรือ TxD ขานี้ใช้เพื่อส่งข้อมูลอนุกรมออกจากคอมพิวเตอร์ โดยการนำข้อมูลที่เก็บอยู่ในบัฟเฟอร์สำหรับส่งข้อมูลส่งออกไป

4.ขา Data Terminal Ready : DTR เป็นขาเอาต์พุตที่ใช้สำหรับส่งสัญญาณออกจากคอมพิวเตอร์เพื่อให้อุปกรณ์ปลายทางรับรู้ว่าต้องการติดต่อกับอุปกรณ์ปลายทาง โดยขา DTR นี้จะต้องเชื่อมต่อกับขา DSR ของอุปกรณ์ปลายทาง และขา DTR ของอุปกรณ์ปลายทางจะต้องเชื่อมต่อกับขา DSR ของคอมพิวเตอร์ และถ้าใช้การเชื่อมต่อแบบใช้สายในการเชื่อมต่อเพียง 3 เส้น จะต้องเชื่อมต่อขา DTR และ DSR ของพอร์ตอนุกรมเข้าด้วยกัน และจะต้องต่อเชื่อมเข้ากับขา DCD ด้วยในกรณีที่โปรแกรมสื่อสารที่ใช้มีการตรวจจับสัญญาณพาหะ

5.ขา Signal Ground : GND เป็นขากราวด์ของสัญญาณ

6.ขา Data Set Ready : DSR ขานี้จะใช้ควบคู่กับขา DTR เพื่อตรวจสอบการเชื่อมต่อกันระหว่างคอมพิวเตอร์กับอุปกรณ์ปลายทาง ซึ่งขา DSR นี้จะเป็นขาสำหรับรับข้อมูลจากภายนอก

7.ขา Request To Send : RTS เป็นขาเอาต์พุตสำหรับส่งสัญญาณร้องขอให้อุปกรณ์ปลายทางส่งข้อมูลมาให้คอมพิวเตอร์ โดยขาที่รับสัญญาณ RTS ก็คือขา CTS ซึ่งในกรณีที่มีการเชื่อมต่อแบบ 3 สาย จะต้องเชื่อมต่อขา RTS และ CTS เข้าด้วยกัน เพื่อให้การรับและส่งข้อมูลสามารถเกิดขึ้นได้ตลอดเวลา

8.ขา Clear To Send : CTS เป็นขาอินพุตทำหน้าที่รอรับสัญญาณที่ส่งเข้ามา เมื่อมีการส่งสัญญาณเข้ามาที่ขานี้ ข้อมูลที่ขา TxD จะถูกส่งออกไป ขานี้จะใช้เพื่อตรวจสอบอุปกรณ์ต่อพ่วงว่าพร้อมที่จะรับข้อมูลแล้วหรือยัง

9.ขา Ring Indicator : RI ใช้แสดงสถานะสัญญาณเรียกจากสายโทรศัพท์ ปกติในการสื่อสารโดยทั่วไปสายนี้จะไม่ถูกใช้งาน จะใช้งานก็ต่อเมื่อมีการเชื่อมต่อกับโมเด็มแล้วยังมีความต้องการตรวจสอบสัญญาณเรียกจากสายโทรศัพท์

ตารางที่ 2.2 หน้าที่ของขาต่างๆในคอนเน็กเตอร์แบบ DB-25 ตัวผู้ และ DB-9 ตัวผู้

D-Type 25 Pin	D-Type 9 Pin	สัญลักษณ์	ชื่อสัญญาณ
Pin 2	Pin 3	TD	Transmit Data
Pin 3	Pin 2	RD	Receive Data
Pin 4	Pin 7	RTS	Request To Send
Pin 5	Pin 8	CTS	Clear To Send
Pin 6	Pin 6	DSR	Data Set Ready
Pin 7	Pin 5	SG	Signal Ground
Pin 8	Pin 1	CD	Carrier Detect
Pin 20	Pin 4	DTR	Data Terminal Ready
Pin 22	Pin 9	RI	Ring Indicator

2.3.4 UART

UART มาจากคำว่า Universal Asynchronous Receiver Transmitter ซึ่งหมายถึงอุปกรณ์ที่ทำหน้าที่รับและส่งข้อมูลแบบอะซิงโครนัสนั่นเอง สำหรับการสื่อสารอนุกรมบนคอมพิวเตอร์แล้ว UART ถือว่าเป็นหัวใจสำคัญของการสื่อสารอนุกรม

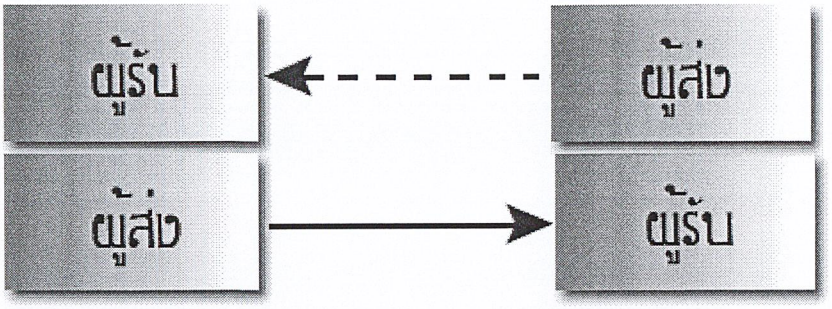
หน้าที่หลักของ UART คือแปลงข้อมูลที่อยู่ในรูปแบบขนานจากซีพียูให้อยู่ในรูปแบบอนุกรมแบบอะซิงโครนัส แล้วส่งทำการส่งออกไป และแปลงสัญญาณอนุกรมแบบอะซิงโครนัสที่ป้อนเข้ามายัง UART ให้เป็นแบบขนานก่อนที่จะส่งเข้าสู่ซีพียู ซึ่งนอกจาก UART จะส่งข้อมูลไปยังซีพียูแล้ว ยังแจ้งรายละเอียดอื่นๆ ของข้อมูล ให้คอมพิวเตอร์รับทราบด้วย อาทิ อัตราเร็วในการรับส่งข้อมูลหรืออัตราบอด, รูปแบบการส่งข้อมูล, ความผิดพลาดที่เกิดขึ้นระหว่างการส่งข้อมูล เช่น ผิดพลาดจากพาริตี้, เฟรมข้อมูล, โอเวอร์รัน เป็นต้น

ภายใน UART จะมีวงจรสร้างอัตราบอดโปรแกรมได้ (programmable baudrate generator) โดยการกำหนดค่าตัวหารให้กับสัญญาณนาฬิกาของ UART โดยตัวหารนี้จะมีขนาด 16 บิต ดังนั้นจะสามารถกำหนดตัวหารอยู่ในช่วง 1 - 65,535 ซึ่งมี รูปแบบในการส่งสัญญาณสำหรับการสื่อสารข้อมูลระหว่างอุปกรณ์กับเครื่อง คอมพิวเตอร์ในเครือข่ายมีลักษณะของรูปแบบการส่งสัญญาณเป็น 3 รูปแบบดังนี้คือ

1. ซิมเพล็กซ์ (Simplex) การส่งข้อมูลแบบซิมเพล็กซ์มีรูปแบบคือสัญญาณข้อมูลถูกส่งในทิศทางเดียว โดยฝ่ายส่งทำหน้าที่ส่งและฝ่ายรับทำหน้าที่รับ ซึ่งทั้งสองฝ่ายจะทำหน้าที่เพียงส่ง และรับเพียงอย่างเดียวเท่านั้น ตัวอย่างของการส่งข้อมูลนี้คือการดูโทรทัศน์

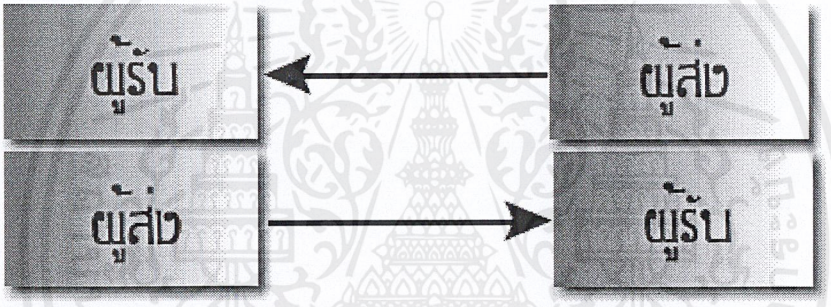
รูปที่ 2.16 แสดงการส่งข้อมูลแบบซิมเพล็กซ์

2. ฮาล์ฟดูเพล็กซ์ (Half Duplex) การส่งข้อมูลแบบฮาล์ฟดูเพล็กซ์ ซึ่งมีรูปแบบการส่งสัญญาณข้อมูลที่สวนทางกันได้โดยการสลับกัน จะส่งหรือรับสัญญาณในเวลาเดียวกันไม่ได้ เช่น การสื่อสารของระบบวิทยุมือถือ



รูปที่ 2.17 แสดงการส่งข้อมูลแบบฮาล์ฟดูเพล็กซ์

3. ฟูลดูเพล็กซ์ (Full Duplex) การส่งข้อมูลแบบฟูลดูเพล็กซ์ ซึ่งมีรูปแบบการส่งข้อมูลสองทิศทางได้ในเวลาเดียวกันหรือส่งข้อมูลได้พร้อมกันสองทาง เช่น การสื่อสารโดยใช้โทรศัพท์



รูปที่ 2.18 แสดงการส่งข้อมูลแบบฟูลดูเพล็กซ์

2.4 Visual Basic

วิชวลเบสิก ถือได้ว่าเป็นเครื่องมือที่ใช้เขียนโปรแกรมบน วินโดวส์ (Windows) ที่ได้รับความนิยมสูงที่สุด ทั้งนี้เพราะผ่านการพัฒนาอย่างต่อเนื่องจากเวอร์ชันแรกที่ทำงานบนดอส (DOS) แล้วมาได้รับความนิยมในเวอร์ชัน 3.0 ที่ทำงานบน วินโดวส์ 3.1 จนก้าวมาถึงเวอร์ชันล่าสุดคือเวอร์ชัน 6.0 ที่ได้รับความนิยมทั้งเมืองไทยและทั่วโลก

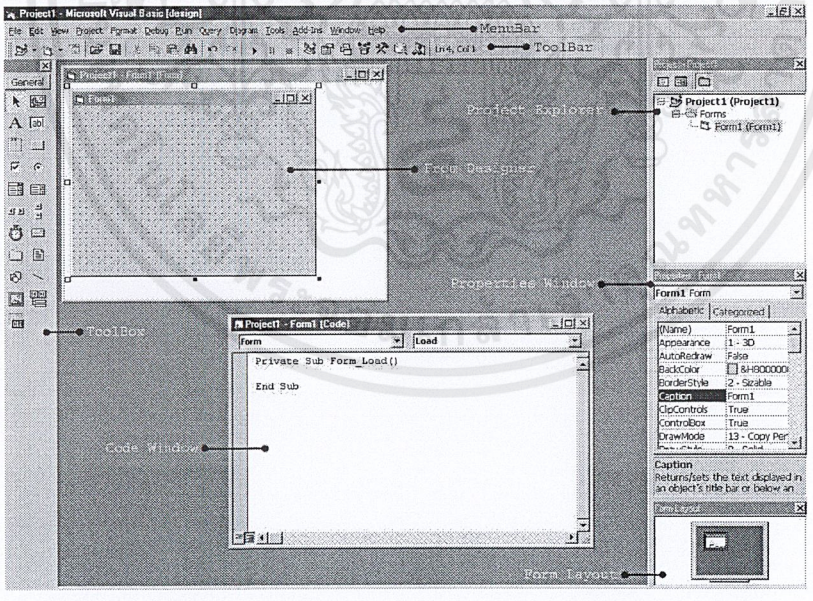
2.4.1 จุดเด่นของวิชวลเบสิก

สร้างแอพพลิเคชั่นได้ง่ายและรวดเร็ววิชวล เบสิก ได้รับการวางตัวเป็นเครื่องมือที่ช่วยให้สร้างแอพพลิเคชั่นได้อย่างรวดเร็วและง่าย เพื่อลดเวลาการสร้างแอพพลิเคชั่นให้สั้นลง ทั้งนี้เพราะมีการขจัดงานที่ต้องทำซ้ำๆ ออกไป และขจัดสิ่งไม่จำเป็นต้องรู้เกี่ยวกับการควบคุมฮาร์ดแวร์ การจัดการภายในของวินโดว์ออกไปเหลือเฉพาะการให้ความสำคัญกับปัญหาของงานจริงๆเท่านั้น ภาษาเขียนโปรแกรมที่ง่ายต่อการเรียนรู้วิชวล เบสิก นั้นจะใช้ภาษา เบสิก (Basic) ในการเขียน

โปรแกรมซึ่งทำให้อ่านโค้ดแล้วมีความใกล้เคียงกับภาษาที่เราใช้งานปกติอีกทั้งยังสื่อความหมายเข้าใจได้ง่ายกว่าภาษาโปรแกรมอื่นๆ รวมเครื่องมืออำนวยความสะดวกในการเขียนโปรแกรม นอกเหนือจากจะง่ายต่อการเรียนรู้แล้ววิชวล เบสิกยังมีเครื่องมือที่ช่วยให้การเขียนโปรแกรมเป็นเรื่องที่ไม่ยุ่งยากเพราะมีเครื่องมือช่วยให้ไม่ต้องจำไวยากรณ์ภาษาที่ยุ่งยาก ตรวจสอบอัตโนมัติว่าโปรแกรมที่เขียนนั้นถูกต้องตามหลักภาษาหรือไม่ มีการแยกแยะส่วนของโปรแกรมอย่างเป็นระเบียบ ทำให้งานของโปรแกรมเมอร์ลดลงได้มากนอกจากจะมีเครื่องมือช่วยในการเขียนโปรแกรมแล้วยังมีเครื่องมือที่ใช้ทดสอบแก้ไขโปรแกรม (Debugger) ที่เขียนขึ้นมาว่าทำงานได้ถูกต้องหรือไม่ มีระบบขอความช่วยเหลือ (Online Help) ไว้อ้างอิง และขอความช่วยเหลือในจุดที่เราสงสัยข้องใจเครื่องมือทั้งหมดที่กล่าวมาถูกจัดรวมไว้ในสภาพแวดล้อมการทำงานเดียวกัน (เรียกย่อๆ ว่า ไอดีอี, IDE ซึ่งมาจาก Integrate Development Environment) ทำให้เรียกใช้งานได้สะดวกตั้งแต่เขียนโปรแกรม ทดสอบ แก้ไข สร้างชุดติดตั้ง รวมทั้งระบบขอความช่วยเหลือ ซึ่งเราสามารถเพิ่มเติมเครื่องมือชนิดใหม่ๆ เข้าไปได้เรื่อยๆ

2.4.2 องค์ประกอบต่างๆ ของวิชวล เบสิก 6.0

เมื่อเราเปิดใช้งานวิชวล เบสิก 6.0 ก็จะพบกับลักษณะการทำงานที่เรียกว่า ไอดีอี คือรวบรวมเครื่องไม้เครื่องมือ ข้อมูลที่ใช้งานต่างๆ ไว้ในหน้าจอเดียว ทำให้เรียกใช้งานได้ง่าย



รูปที่ 2.19 แสดงองค์ประกอบต่างๆ ของวิชวล เบสิก

MenuBar เมนูบาร์ เป็นส่วนที่รับคำสั่งในแบบเมนู เมื่อผู้ใช้ทำการสร้างแอพพลิเคชั่นด้วยวิชวลเบสิก เป็นเหมือนศูนย์กลางที่ควบคุมการสร้างแอพพลิเคชั่น

ToolBar ในการใช้งานเมนูบาร์สั่งงานอาจจะมีขั้นตอนที่ยุ่งยาก เพื่อลดขั้นตอนลง ผู้ใช้สามารถ คลิกที่ทูลบาร์เพียงครั้งเดียว ก็สามารถสั่งงานที่เราต้องการได้ (เป็นเหมือนคีย์ลัดในการทำงาน)

ToolBox ทูลบ็อกซ์ เป็นกล่องเก็บ ActiveX Control ซึ่งนำมาประกอบเป็นส่วนต่างๆ ของแอพพลิเคชั่น

Project Explorer เป็นครื่องมือที่ใช้ควบคุมการทำงานของโปรเจ็กต์ (ซึ่งจะกล่าวถึงความหมายของโปรเจ็กต์ในหัวข้อต่อไป)

Properties Window เป็นส่วนกำหนดพร็อพเพอร์ตี้ให้กับออบเจ็กต์ต่างๆ

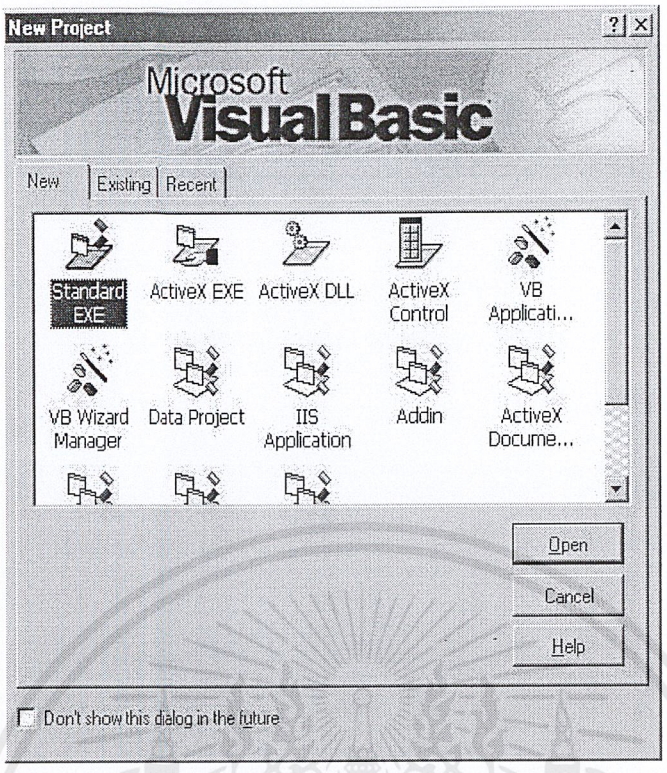
Form Layout ฟอร์มเลย์เอาต์ เป็นหน้าตาคร่าวๆ ของฟอร์มที่ได้จากการรันแอพพลิเคชั่นทำให้ผู้ใช้ทราบตำแหน่งของส่วนต่างๆที่จะปรากฏบนจอภาพเมื่อแอพพลิเคชันทำงาน

Form Designe ฟอร์มดีไซเนอร์ เป็นส่วนที่มองเห็นในขณะออกแบบแอพพลิเคชั่นของวิชวลเบสิก ซึ่งผู้ใช้จะสามารถออกแบบหน้าตาของแอพพลิเคชั่นผ่านฟอร์มดีไซเนอร์

Code Window โค้ดวินโดว์ เป็นส่วนที่เราเขียนโปรแกรม (เรียกสั้นๆ ว่าเขียนโค้ด) เพื่อควบคุมการทำงานของแอพพลิเคชั่น

2.4.3 รูปแบบการสร้างแอพพลิเคชั่นด้วยวิชวล เบสิก

เมื่อเปิดวิชวล เบสิก 6.0 ขึ้นมา จะพบกับไดอะล็อกบ็อกซ์ New Project ซึ่งแสดงประเภทของแอพพลิเคชั่นที่สร้างได้ สำหรับแอพพลิเคชั่นแต่ละประเภทที่สร้างได้นั้นมีความหมายดังตารางที่ 2.5



รูปที่ 2.20 หน้าต่าง New Project

ตารางที่ 2.3 ตารางแสดงประเภทของแอพพลิเคชั่น

ชนิดของแอพพลิเคชั่น	คำอธิบาย
Standard EXE	เป็นแอพพลิเคชั่นทั่วไปที่มีการใช้ใน วินโดว์ เมื่อสร้างแล้วจะได้ไฟล์ที่มีนามสกุลเป็น .EXE
ActiveX EXE	เป็นการสร้างแอพพลิเคชั่นชนิดที่เรียกว่า out – of – procees OLE
ActiveX DLL	เป็นการสร้างแอพพลิเคชั่นชนิดที่เรียกว่า in – procees OLE server
ActiveX Control	เป็นการสร้าง ActiveX Control เมื่อสร้างแล้วจะได้ไฟล์นามสกุลเป็น .OCX
ชนิดของแอพพลิเคชั่น	คำอธิบาย
VB Application Wizard	เป็นการสร้างวิซาร์ดเพื่อใช้งานร่วมกับแอพพลิเคชั่น ซึ่งมักจะเป็นแอพพลิเคชั่นที่มีความซับซ้อน จึงต้องมีวิซาร์ดช่วยลดความยุ่งยาก
VB Wizard Manager	เป็นเครื่องมือที่ใช้สร้างวิซาร์ด

ตารางที่ 2.3 (ต่อ) ตารางแสดงประเภทของแอพพลิเคชั่น

Data Project	เป็นการสร้างแอพพลิเคชั่นเพื่อให้ทำงานร่วมกับ Data Object
DHTML Application	เป็นการสร้างแอพพลิเคชั่นอินเตอร์เน็ตในฝั่งไคลเอนท์
IIS Application	เป็นการสร้างแอพพลิเคชั่นอินเตอร์เน็ตในฝั่งเซิร์ฟเวอร์
Addin	เป็นเครื่องมือที่ช่วยในการสร้าง Add-in ซึ่งวิชวล เบสิก ให้สร้างเครื่องมือใช้เฉพาะกับงานของเราได้
ActiveX Document DLL	เป็นการสร้างแอพพลิเคชั่นชนิดที่เรียกว่า in- procees ActiveX Document
ActiveX Document EXE	เป็นการสร้างแอพพลิเคชั่นชนิดที่เรียกว่า out-of- procees ActiveX Document

ในไดอะล็อกบ๊อกซ์ New Project ยังมีอีก 2 แท็บซึ่งมีความหมายดังนี้

แท็บ Existing แสดงโปรเจ็กต์ที่เคยมีการสร้างมาก่อน ซึ่งช่วยให้เราไม่ต้องเสียเวลาค้นหาโฟลเดอร์ที่เก็บโปรเจ็กต์เดิมๆ

แท็บ Recent แสดงโปรเจ็กต์ที่เคยมีการสร้างมาก่อน และถูกเรียกมาแก้ไขล่าสุด

ออบเจ็กต์ (Object) และ เมธอด (Method)

1. ออบเจ็กต์ คือการนำอุปกรณ์ต่างๆ ที่มีอยู่ในโปรแกรมมาประกอบกันเป็นแอพพลิเคชั่น ในส่วนที่พบเห็นกันมากเป็นลักษณะ ของปุ่มกด ฟอร์ม หรือปุ่มตัวเลือกต่างๆ โดยที่เรายังสามารถกำหนดพร็อพเพอร์ตี้ต่างๆ ให้กับออบเจ็กต์ได้ที่ Properties Window

2. เมธอด คือ ความสามารถของออบเจกต์ ซึ่งจะถูกเรียกผ่านทางการเขียนโปรแกรม การใช้เมธอดด้านกราฟฟิก

2.1 การลบกราฟฟิกด้วยเมธอด Cls

เมทถอด Cls ใช้ในการลบกราฟฟิกที่วาด ใช้ได้กับ Form คอนโทรล Image หรือ คอนโทรล PictureBox โดยจะมีรูปแบบคำสั่งดังนี้

<ชื่อคอนโทรล>.Cls

เมธอดนี้มีประโยชน์ เมื่อต้องการลบกราฟฟิกที่วาดด้วยเมธอดอื่น เพื่อวาดกราฟฟิกใหม่

2.2 การวาดจุดกราฟฟิกด้วยเมธอด Pset

เมธอดPset ใช้ในการวาดจุดภาพ (เรียกว่า Pixel) ในฟอร์ม คอนโทรล Image หรือ คอนโทรล PictureBox โดยจะมีรูปแบบคำสั่งดังนี้

<ชื่อคอนโทรล>.Pset (x,y),[Color]

โดยที่ x,y คือ ตำแหน่งที่จะวาดจุดภายในของคอนตัวโทรลนั้น

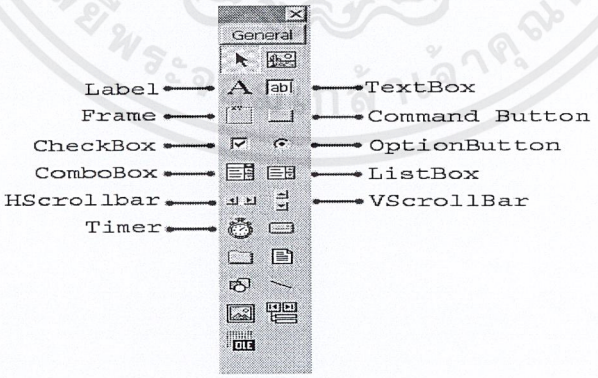
Color คือ เป็นสีของจุดที่วาด ถ้าไม่มีการเซตค่าจะใช้สีตาม คุณสมบัติ ForeColor ของ คอนโทรลนั้นแทน

2.4.4 ขั้นตอนการสร้างแอพพลิเคชั่นด้วยวิชวลเบสิก

1. ออกแบบแอพพลิเคชั่น
2. ตกแต่งหน้าตาแอพพลิเคชั่น
3. เขียนโค้ดกำกับการทำงานของแอพพลิเคชั่น
4. ทดสอบการทำงานของแอพพลิเคชั่น
5. บันทึกเก็บไว้ในคอมพิวเตอร์
6. การสร้างไฟล์ .EXE (Make)

2.4.5 ActiveX Control เบื้องต้น

ActiveX Control ถูกสร้างขึ้นมาเพื่อที่ใช้ในสร้างแอพพลิเคชั่นให้ง่ายขึ้น โดยมีส่วนประกอบพื้นฐานที่ควรรู้จักมีดังนี้

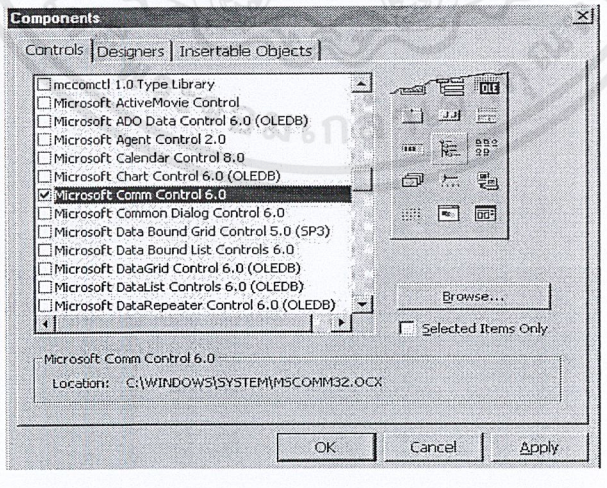


รูปที่ 2.21 แสดง AciveX Control พื้นฐานที่ปรากฏใน Toolbox

ตารางที่ 2.4 แสดงหน้าที่ของ ActiveX Control แต่ละแบบ

ActiveX Control	คำอธิบาย
Label	เป็นแถบข้อความ มักใช้เขียนข้อความให้อ่านอย่างเดียว
TextBox	เป็นช่องให้ผู้ใช้งานกรอกข้อความ
Frame	เป็นกรอบที่จัดกลุ่ม คอนโทรล ต่างๆ ออกเป็นพวกๆ
Command Button	เป็นปุ่มกดให้ผู้ใช้งานกด <Enter> หรือคลิกที่ปุ่มนี้
ChectBox	เป็นปุ่มให้ผู้ใช้งานคลิกเลือก ซึ่งจะเลือกได้กี่ตัวก็ได้
OptionButton	เป็นปุ่มให้ผู้ใช้งานคลิกเลือก ซึ่งเลือกได้เพียง 1 ตัวเท่านั้น
ComboBox	เป็นรายการข้อมูลให้ผู้ใช้งานเลือก
ListBox	เป็นรายการข้อมูลให้ผู้ใช้งานเลือก
HScrollBar	เป็นแถบเลื่อนตามแนวนอน
VScrollBar	เป็นแถบเลื่อนตามแนวตั้ง
Timer	เป็นตัวจับเวลา

นอกเหนือจากที่กล่าวมาแล้วยังมี ActiveX Control ต่างๆ ที่ถูกซ่อน ซึ่งหนึ่งในนั้นที่มีความสำคัญมากที่สุดที่ใช้ในโครงงานนี้ก็คือ Microsoft Comm Control 6.0 ซึ่งเป็น ActiveX Control ที่ใช้เกี่ยวกับการติดต่อกับพอร์ตอนุกรม โดยสามารถเพิ่มเติมเข้ามาไว้ใน Toolbox โดยเลือกเมนู Project > Components... หรือคลิกขวาที่ Toolbox แล้วเลือก Component...จะปรากฏหน้าจอดังรูปที่ 2.22



รูปที่ 2.22 แสดงการเพิ่ม Microsoft Comm Control 6.0 ใช้งานกับพอร์ตอนุกรม

2.4.6 การเชื่อมต่อแบบอนุกรม

2.4.6.1 คอลโทรลที่ใช้ติดต่อ (คอนโทรล MSComm)

สำหรับการใช้งานวิชวล เบสิก ตั้งแต่เวอร์ชัน 2 เป็นต้นมา ในวิชวล เบสิก จะมีคัสตอมคอนโทรลสำหรับการสื่อสารอนุกรมผ่านทางพอร์ตอนุกรมของคอมพิวเตอร์มาให้ โดยในวิชวล เบสิก เวอร์ชัน 2 และเวอร์ชัน 3 จะใช้ชื่อว่า MSCOMM.VBX ส่วนเวอร์ชัน 4 ใช้ชื่อว่า MSCOMM16.OCX สำหรับการทำงานกับระบบปฏิบัติการ 16 บิต และ MSCOMM32.OCX สำหรับการทำงานกับระบบปฏิบัติการ 32 บิต สำหรับในวิชวล เบสิก เวอร์ชัน 5 จะมีเพียง MSCOMM32.OCX เท่านั้นเพราะถูกออกแบบมาให้ใช้งานกับระบบปฏิบัติการ 32 บิต

MSComm จัดเตรียมทางเลือกเอาไว้ 2 ทางเพื่อความสะดวกในการสื่อสารข้อมูล ทางแรกคือ การสื่อสารข้อมูลที่กระตุ้นด้วยเหตุการณ์ (event-driven communicatios) เป็นรูปแบบการใช้งานที่มีประสิทธิภาพมากสำหรับการตอบสนองแบบทันทีทันใด เช่น เมื่อตัวอักษรถูกส่งมาที่พอร์ตอนุกรมหรือเกิดการเปลี่ยนแปลงที่ Data Carrier Detect (DCD) หรือขา Request To Send (RTS) เหตุการณ์ Oncomm ของ MSComm จะสามารถตรวจจับสัญญาณนั้นได้ทันที ซึ่งจะกล่าวถึงรายละเอียดในหัวข้อคุณสมบัติ CommEvent ต่อไป ส่วนทางเลือกที่สองเป็นการคอยตรวจสอบค่าเหตุการณ์และความผิดพลาดที่เกิดขึ้นด้วยการดูค่าที่เปลี่ยนแปลงภายในคุณสมบัติ CommEvent หลังจากให้โปรแกรมทำงานในฟังก์ชั่นต่างๆ ไปเรียบร้อยแล้ว ซึ่งวิธีนี้ใช้งานได้ดีในกรณีที่โปรแกรมมีขนาดเล็ก

คอนโทรล MSCommm 1 ตัวสามารถควบคุมการทำงานของพอร์ตอนุกรมได้ 1 พอร์ต ถ้าในโปรแกรมที่ใช้งานต้องการติดต่อกับพอร์ตอนุกรมมากกว่า 1 พอร์ตจะต้องใช้คอนโทรล MSComm มากกว่า 1 ตัวเพื่อควบคุมพอร์ตอนุกรมในแต่ละพอร์ต แอดเดรสของพอร์ตอนุกรมและแอดเดรสของการเกิดอินเตอร์รัปต์สามารถเปลี่ยนแปลงได้จากการแก้ไขค่าที่ Control Panel

2.4.6.2 คุณสมบัติ (property) ของ MSComm

1. CommPort ใช้ในการกำหนดและอ่านค่าพอร์ตอนุกรมที่ติดต่ออยู่ (COM1, COM2, COM3, COM4) รูปแบบการใช้งาน

object.CommPort[= value]

โดย Value เป็นค่าของพอร์ตอนุกรม ชนิดของข้อมูลเป็น Integer ค่า Value สามารถกำหนดได้ในช่วง 1-16 (ค่าเริ่มต้นกำหนดไว้ที่ 1) เมื่อมีการกำหนดค่าแล้วทำการเปิดพอร์ตโดยใช้คุณสมบัติ PortOpen แต่ว่าพอร์ตนั้นไม่มีอยู่ในระบบ MSComm จะสร้างสัญญาณแสดงข้อผิดพลาด error 68

ขึ้นมา ซึ่งหมายถึง อุปกรณ์ตัวนี้ไม่มีอยู่ในระบบ ดังนั้นการเขียนโปรแกรมจึงจำเป็นต้องกำหนดตำแหน่งของพอร์ตอนุกรมก่อนที่ใช้คำสั่ง OpenPort

2. Setting ใช้ในการกำหนดและอ่านค่าอัตราบอด, พาริตี้, จำนวนของบิตข้อมูล, จำนวนของบิตปิดท้ายรูปแบบการใช้งาน

object.Settings [= value]

ค่า Value มีชนิดข้อมูลเป็นแบบ String มีรูปแบบเป็น “BBBB,P,D,S” โดย BBBB เป็นค่าอัตราบอด P เป็นค่าพาริตี้ D เป็นจำนวนของบิตข้อมูล และ S เป็นจำนวนของบิตปิดท้าย ปกติแล้วค่านี้ถูกกำหนดไว้เป็น “9600, N, 8, 1”

ค่าอัตราบอดมาตรฐานที่ใช้กับ MSComm มีดังนี้

110 บิตต่อวินาที
300 บิตต่อวินาที
600 บิตต่อวินาที
1,200 บิตต่อวินาที
2,400 บิตต่อวินาที
9,600 บิตต่อวินาที (ค่าปกติ)
14,400 บิตต่อวินาที
19,200 บิตต่อวินาที
28,800 บิตต่อวินาที
38,400 บิตต่อวินาที (สงวน)
56,000 บิตต่อวินาที (สงวน)
128,000 บิตต่อวินาที (สงวน)
256,000 บิตต่อวินาที (สงวน)

สำหรับค่ามาตรฐานในการกำหนดค่าพาริตี้มีดังนี้

สัญลักษณ์	รายละเอียด
E	พาริตี้คู่ (Even)
M	ลอจิก “1” (Mark)
N	ไม่ใช้ (ค่าปกติ)
O	พาริตี้คี่ (Odd)
S	ลอจิก “0” (Space)

ค่าที่ใช้ในการกำหนดจำนวนบิตมี 5 ค่าคือ 4, 5, 6, 7 และ 8 (เป็นค่าปกติ)
ค่าที่ระบุจำนวนบิตปิดท้ายมี 3 ค่าคือ 1 (เป็นค่าปกติ), 1.5 และ 2
ตัวอย่างการใช้งานคำสั่ง Settings โดยจะเป็นการกำหนดค่าอัตราบอดเท่ากับ 9600 ไม่มีพาริตี้ จำนวนบิตข้อมูล 8 บิต และ บิตปิดท้าย 1 บิต สามารถเขียนโปรแกรมได้ดังนี้

```
MSComm1.Settings = “9600, N, 8, 1”
```

หมายเหตุ สาเหตุที่ค่าที่กำหนดจะต้องอยู่ภายในเครื่องหมายอัญประกาศ “” เนื่องจาก ค่าที่กำหนดนี้อยู่ในรูปตัวแปร String

3. PortOpenใช้ในการกำหนดและอ่านค่าสถานะของพอร์ตอนุกรม เพื่อเปิดและปิดพอร์ตอนุกรมรูปแบบการใช้งาน

```
object.PortOpen [          = value  ]
```

ค่า Value มีชนิดข้อมูลเป็นแบบบูลีนคือ True กับ False โดย True หมายถึงการเปิดพอร์ตอนุกรมและ False หมายถึงการปิดพอร์ตอนุกรม สำหรับการปิดพอร์ตนั้นจะมีการเคลียร์บัฟเฟอร์รับข้อมูลและบัฟเฟอร์ส่งข้อมูลด้วย คอนโทรล MSComm จะปิดพอร์ตอนุกรมโดยอัตโนมัติเมื่อออกจากโปรแกรม ก่อนที่จะใช้คุณสมบัติ PortOpen ต้องตรวจสอบให้แน่ใจก่อนว่าคุณสมบัติ CommPort นั้นได้ทำการกำหนดตำแหน่งของพอร์ตอนุกรมไว้ถูกต้องหรือไม่ มิเช่นนั้น MSComm จะแสดงข้อผิดพลาด Error 68 แจ้งแก่ผู้ใช้งาน หรือถ้าพอร์ตอนุกรมนั้นถูกเปิดเอาไว้แล้ว โปรแกรมก็จะแจ้งข้อผิดพลาดออกมาเช่นเดียวกัน

ถ้าคุณสมบัติ DTREnable หรือ RTSEnable ถูกกำหนดให้เป็น True ก่อนที่จะทำการเปิดพอร์ต ค่าคุณสมบัติของ DTREnable หรือ RTSEnable จะถูกเซตเป็น False หลังจากปิดพอร์ต แต่ถ้าเซตเป็น False หลังจากปิดโปรแกรมแล้ว ค่าที่กำหนดไว้จะเป็นค่าเดิม

ตัวอย่างการใช้คำสั่งเปิดพอร์ต เพื่อติดต่อสื่อสารกับพอร์ตอนุกรม COM1 และมีอัตราบอด 9,600 บิตต่อวินาที ไม่มีพาริตี้ จำนวนบิตข้อมูล 8 บิต และบิตปิดท้าย 1 บิต มีดังนี้

```
MSComm1.Settings      = "9600, n, 8, 1"
MSComm1.CommPort  = 1
MSComm1.PortOpen    = True
```

4. Input อ่านค่าและลบค่าขบวนข้อมูลจากบัฟเฟอร์ภาครับ รูปแบบการใช้งาน

```
object.Input
```

คุณสมบัติ InputLen เป็นตัวกำหนดจำนวนของตัวอักษรที่จะอ่าน โดยคุณสมบัติ Input การกำหนดค่าให้ InputLen เท่ากับ 0 เป็นการกำหนดให้คุณสมบัติ Input ทำการอ่านค่าข้อมูลในบัฟเฟอร์รับข้อมูลทั้งหมด

คุณสมบัติ InputMode เป็นตัวกำหนดชนิดของข้อมูลที่คุณสมบัติ Input รับเข้ามา ถ้า InputMode ถูกกำหนดเป็น comInputModeText คุณสมบัติ Input จะส่งค่าข้อมูลกลับมาในรูปแบบของข้อความชนิดข้อมูลเป็นแบบ Variant ถ้า InputMode กำหนดเป็น comInputModeBinary คุณสมบัติ Input จะส่งข้อมูลกลับมาในรูปของไบนารี่และชนิดข้อมูลเป็นแบบ Variant

ตัวอย่างโปรแกรมแสดงให้เห็นถึงวิธีในการรับข้อมูลจากบัฟเฟอร์รับข้อมูลทั้งหมด

```
Private Sub Command1_Click()
Dim InString as String
    MSComm1.InputLen = 0                    ' Retrieve all available data.
    If MSComm1.InBufferCount Then           ' Check for data.
    InString = MSComm1.Input                ' Read data
    End If
End Sub
```

5. InBufferCount ส่งค่าจำนวนของตัวอักษรที่อยู่ในบัฟเฟอร์ภาครับรูปแบบการใช้งานคำสั่ง

object.InBufferCount [= value]

คำสั่ง InBufferCount จะแสดงค่าจำนวนของตัวอักษร ซึ่งรับมาจากภายนอกและยังเก็บอยู่ในบัฟเฟอร์ภาครับ เพื่อให้ผู้ใช้งานอ่านค่าออกไป สำหรับการเคลียร์ค่าบัฟเฟอร์ภาครับทำได้โดยกำหนดให้ InBufferCount มีค่าเป็น 0

หมายเหตุ อย่าสับสนระหว่างคำสั่ง InBufferSize และ InBufferCount คำสั่ง InBufferSize นั้นใช้เพื่อกำหนดขนาดของบัฟเฟอร์ภาครับ

6. InBufferSize กำหนดและคืนค่าขนาดของบัฟเฟอร์ภาครับในหน่วยเป็นไบต์ รูปแบบการใช้งานคำสั่ง

object.InBufferSize [= value]

คำสั่ง InBufferSize ใช้เพื่อกำหนดขนาดของบัฟเฟอร์ภาครับ ค่าเริ่มต้นกำหนดไว้ที่ 1,024 ไบต์

หมายเหตุ การกำหนดค่าบัฟเฟอร์ภาครับขนาดใหญ่จะทำให้ หน่วยความจำที่เหลือสำหรับการใช้งานส่วนอื่นๆ จะเหลือน้อย อย่างไรก็ตามการกำหนดค่า บัฟเฟอร์ภาครับที่น้อยเกินไปจะทำให้เกิดการ โอเวอร์โฟลวหรือข้อมูลล้นบัฟเฟอร์ เว้นแต่จะมีการใช้แฮนด์เช็ก ดังนั้นค่าปานกลางที่เหมาะสมก็คือค่า 1,024 ซึ่งเป็นค่าเริ่มต้นนั่นเอง แต่ถ้าโปรแกรมมีการเกิดโอเวอร์โฟลวแล้วจึงค่อยปรับเพิ่มค่าขนาดของบัฟเฟอร์ให้มีค่ามากขึ้น

7. InputLen กำหนดค่าและคืนค่าจำนวนของตัวอักษรที่อ่านจากบัฟเฟอร์ภาครับ รูปแบบการใช้งานคำสั่ง

object.InputLen [= value]

ค่าเริ่มต้นของคุณสมบัติ InputLen มีค่าเท่ากับ “0” การกำหนดค่าเท่ากับ “0” จะทำให้ คำสั่ง Input ของ MSComm อ่านค่าข้อมูลที่อยู่ภายในบัฟเฟอร์ภาครับทั้งหมด

ถ้าไม่มีข้อมูลอยู่ในบัฟเฟอร์ภาครับมากเท่ากับจำนวน InputLen คำสั่ง Input จะส่งค่าว่าง (“”) กลับออกมา ผู้ใช้งานสามารถตรวจสอบข้อมูลในข้อมูลในบัฟเฟอร์ภาครับได้โดยใช้คุณสมบัติ InBufferCount โดยกำหนดให้มีข้อมูลอยู่ในบัฟเฟอร์ภาครับก่อนแล้วจึงค่อยอ่านข้อมูลจากบัฟเฟอร์ภาครับ

คุณสมบัตินี้มักใช้กับการอ่านค่าข้อมูลจากเครื่องมือหรือเครื่องจักรที่มีการกำหนดค่าขนาดความยาวของข้อมูลเอาไว้แล้ว

ตัวอย่างโปรแกรมการอ่านค่าตัวอักษรออกมา 10 ตัวอักษร

```
Private Command1_Click()
        Dim CommData as String
        MSComm1.InputLen  = 10                    'Specify a 10 character block of data.
        CommData = MSComm1.Input                  'Read data.
End Sub
```

8. InputMode กำหนดค่าและคืนค่าชนิดของข้อมูลที่รับโดยคำสั่ง Input รูปแบบการใช้งานคำสั่ง

```
object.InputMode [        = value ]
```

คุณสมบัติ InputMode ใช้กำหนดว่าข้อมูลชนิดไหนที่รับเข้ามาผ่านคำสั่ง Input โดยข้อมูลจะเลือกได้ 2 ประเภทคือ

comInputModeText สำหรับข้อมูลที่อยู่ในรูปข้อความตัวอักษรตามมาตรฐาน ANSI โดยจะต้องกำหนดค่าเป็น “0” และค่าเริ่มต้นของการรับค่าข้อมูลก็จะเป็นค่านี้

comInputModeBinary สำหรับข้อมูลอื่นๆ ซึ่งจะเก็บในรูปไบนารี่รวมกันอยู่เป็นไบต์ข้อมูล

ตัวอย่างการใช้งาน InputMode ต่อไปนี้จะทำการอ่านค่าข้อมูล 10 ไบต์จากพอร์ตอนุกรมและเก็บข้อมูลไว้ในตัวแปรแบบอาเรย์ ชนิดข้อมูลเป็นแบบไบต์

```
Private Sub Command1_Click()
        Dim Buffer as Variant
        Dim Arr() as Byte
        MSComm1.CommProt = 1
        MSComm1.PortOpen = True
```

```
MSComm1.InputMode = comInputModeBinary
Binary data
Do Until MSComm1.InBufferCount < 10
Input buffer
DoEvents
Loop
Buffer = MSComm1.Input
Arr =Buffer
End Sub
```

9. Output ใช้ในการส่งขบวนของข้อมูลไปยังบัฟเฟอร์ส่งข้อมูล รูปแบบการใช้งาน

```
object.Output [          = value ]
```

ค่า value เป็นค่าของตัวอักษรที่เขียนไปยังบัฟเฟอร์ส่งข้อมูล คุณสมบัติ Output สามารถใช้ในการส่งข้อมูลตัวอักษรหรือข้อมูลไบนารี่ก็ได้ โดยการส่งข้อมูลเป็นรูปแบบตัวอักษรจะต้องกำหนดข้อมูลเป็นแบบ Variant และมีข้อมูลภายในเป็นแบบ String สำหรับการส่งข้อมูลไบนารี่จะต้องกำหนดชนิดของข้อมูลเป็นแบบ Variant และมีข้อมูลภายในเป็นแบบ Byte

ตัวอย่างโปรแกรมการส่งค่าที่ป้อนจากคีย์บอร์ดไปยังพอร์ตอนุกรม โดยใช้คุณสมบัติ Output

```
Pritvate Sub Form_KeyPress (KeyAscii As Interger)
Dim Buffer as Variant
MSComm1.CommPort = 1
MSComm1.PortOpen = True
Buffer = Chr$(KeyAscii)
MSComm1.Output =Buffer
End Sub
```

10. OutBufferCount คืนค่าจำนวนของข้อมูลตัวอักษรที่เก็บอยู่ในบัฟเฟอร์ภาคส่ง และสามารถใช้คำสั่งนี้เพื่อเคลียร์บัฟเฟอร์ภาคส่งได้ด้วย รูปแบบการใช้งานคำสั่ง

object.OutBufferCount [= value]

ผู้ใช้งานสามารถเคลียร์บัฟเฟอร์ภาคส่งได้โดยการกำหนดค่า OutBufferCount เท่ากับ “0” หมายเหตุ ระวังการสับสนระหว่างคุณสมบัติ OutBufferCount กับ OutBufferSize ซึ่ง OutBufferSize ใช้เพื่อกำหนดขนาดของบัฟเฟอร์ภาคส่ง

11. OutBufferSize กำหนดค่าและคืนค่าขนาดของบัฟเฟอร์ภาคส่ง ชนิดตัวแปรเป็นแบบไบต์ รูปแบบการใช้งานคำสั่ง

object.OutBufferSize [= object]

คุณสมบัติ OutBufferSize ใช้สำหรับกำหนดขนาดของบัฟเฟอร์ภาคส่ง โดยค่าปกติที่ใช้งานจะมีค่าเท่ากับ 512 ไบต์

หมายเหตุ การกำหนดค่าบัฟเฟอร์ภาคส่งที่มากเกินไปจะทำให้ มีหน่วยความจำเหลือให้ใช้งานน้อย แต่อย่างไรก็ตามถ้ากำหนดค่าน้อยเกิน จะทำให้เกิดข้อมูลล้นบัฟเฟอร์ขึ้นได้ ยกเว้นจะมีการใช้ แฮนด์เช็ค วิธีการที่ถูกต้องในการกำหนดค่าคือ ทดลองใช้ค่าเริ่มต้นคือค่า 512 ไบต์ดูก่อนถ้าโปรแกรมทำงานแล้วเกิดการล้นของข้อมูลค่อยเพิ่มค่าของ OutBufferSize ให้มากขึ้น

12. ParityReplace กำหนดและคืนค่าตัวอักษรที่ไปวางแทนในตำแหน่งที่เกิดข้อผิดพลาดจากพาริตี้ รูปแบบการใช้งานคำสั่ง

object.ParityReplace [= value]

บิตพาริตี้ เป็นบิตที่ทางภาคส่งข้อมูลทำการส่งมาพร้อมกับข้อมูล เพื่อตรวจสอบข้อผิดพลาดของข้อมูล โดยเมื่อมีการใช้บิตพาริตี้ คอนโทรล MSCOM จะทำการบอกบิตทุกบิตที่มีค่าลอจิก “1” ในแต่ละไบต์ และทำการตรวจสอบผลลัพธ์ว่าบิตที่อ่านได้นั้นมีจำนวนลอจิก “1” เป็นเลขคู่หรือคี่ และตรงกับค่าที่กำหนดไว้แต่ต้นหรือไม่ ถ้าค่าที่นำมาบวกแล้วมีพาริตี้ไม่ตรงแสดงว่าการรับส่งข้อมูลผิดพลาด

การกำหนดค่า เริ่มต้นให้กับ ParityReplace นั้นกำหนดให้ใช้เครื่องหมาย (?) ไปวางไว้ที่ตำแหน่งที่เกิดพาริตี้ผิดพลาด ถ้ากำหนดค่า ParityReplace ให้เป็นค่าว่าง (“”) จะเป็นการยกเลิกการใช้งาน ParityReplace และไม่มีป้อนข้อมูลแทนเมื่อตรวจพบข้อผิดพลาด

ParityReplace ใช้ชนิดข้อมูลเป็นแบบสตริง แต่จะกำหนดได้เพียงไบต์เดียวเท่านั้น ซึ่งจะสามารถใช้ค่าใดๆ ก็ได้ที่เป็นโค้ด ANSI มีค่าอยู่ระหว่าง 0-255

13. DTREnable ใช้ในการกำหนดสถานะลอจิกของขา Data Terminal Ready (DTR) โดยสัญญาณของขา DTR จะส่งจากคอมพิวเตอร์ไปยังโมเด็มเพื่อแสดงว่าคอมพิวเตอร์พร้อมที่จะรับข้อมูลแล้ว ชนิดของข้อมูลเป็นแบบบูลีน รูปแบบการใช้งาน

object.DTREnable [= value]

ค่า Value เป็นค่าสถานะ True หรือ False เพื่อกำหนดลอจิกของขา DTR ให้เป็น “0” หรือ “1” โดย

True หมายถึง ให้ขา DTR มีลอจิก “1”

False หมายถึง ให้ขา DTR มีลิอจิก “0” (ค่าปกติ)

หมายเหตุ เมื่อขา DTR ถูกกำหนดสถานะให้เป็น True ที่ขา DTR จะมีสถานะลอจิก “1” เมื่อทำการเปิดพอร์ตและจะมีสถานะเป็น”0” เมื่อมีการปิดพอร์ต เมื่อขา DTR ถูกกำหนดสถานะเป็น False ที่ขา DTR จะมีสถานะลอจิก “0” ตลอดเวลาไม่ว่าจะใช้คำสั่งเปิดพอร์ตหรือปิดพอร์ต

สำหรับการใช้งานโมเด็ม การทำให้ขา DTR เป็นลอจิก “0” จะเป็นการวางหูโทรศัพท์หรือยกเลิกการติดต่อ

14. RTSEnable ใช้เพื่อกำหนดสถานะลอจิกให้ขา Request To Send (RTS) โดยขา RTS จะเป็นสัญญาณที่ส่งจากคอมพิวเตอร์ไปยังโมเด็มเพื่อร้องขอส่งข้อมูล ชนิดของข้อมูลเป็นแบบ Boolean รูปแบบการใช้งาน

object.RTSEnable [= value]

ค่า Value เป็นค่าสถานะ True หรือ False เพื่อกำหนดลอจิก “0” หรือ “1” ให้ขา RTS โดย

True หมายถึง ให้ขา RTS มีลอจิก “1”

False หมายถึง ให้ขา RTS มีลอจิก “0” (เป็นค่าปกติ)

หมายเหตุ เมื่อขา RTSEnable ถูกกำหนดให้เป็น True ขา RTS จะมีสถานะลอจิก “1” เมื่อเปิดพอร์ตและมีสถานะลอจิก “0” เมื่อปิดพอร์ต และเมื่อมีการกลับสถานะของขา RTS ขา TxD จะมีสถานะลอจิกเป็น “1” แต่ที่คุณสมบัติ Break ยังคงเป็นค่าเดิม

15. EOFEnable เป็นการกำหนดให้ MSComm รอสัญลักษณ์แสดงส่วนท้ายสุดของไฟล์ (End of file : EOF) ระหว่างการรับอินพุทเข้ามา ถ้าพบสัญลักษณ์ EOF ภาคอินพุทจะหยุดรับข้อมูล และเหตุการณ์ OnComm จะถูกกระตุ้นให้ทำงาน คุณสมบัติ CommEvent จะมีค่าเท่ากับ 7 หรือ ComEvEOF รูปแบบการใช้งาน

object.EOFEnable [= value]

โดย value เป็นค่าสถานะ True หรือ False เพื่อเอ็นเอเบิลหรือดิสเอเบิลการทำงานของเหตุการณ์ OnComm เมื่อตรวจพบสัญลักษณ์ EOF โดย

True หมายถึง เหตุการณ์ OnComm จะถูกกระตุ้นให้ทำงาน EOF

False หมายถึง เหตุการณ์ OnComm จะไม่ถูกกระตุ้นให้ทำงานด้วย EOF (เป็นค่าปกติ)

เมื่อ EOFEnable กำหนดให้เป็น False ส่วนควบคุมจะไม่มีการตรวจสอบสัญลักษณ์ EOF

16. CTSHolding ผู้ใช้งานสามารถตรวจสอบการทำงานของขา Clear To Send (CTS) ได้ว่ามีสถานะลอจิก “0” หรือ “1” โดยค่าที่อ่านได้จะเป็นบูลีน True และ False ถ้าค่า CTSHolding เป็น True ขา CTS จะมีสถานะลอจิกเป็น “1” ถ้าค่า CTSHolding เป็น False ขา CTS จะมีสถานะลอจิกเป็น “0” รูปแบบการใช้งาน

object. CTSHolding

เมื่อขา CTS เป็นลอจิก “0” (CTSHolding = False) และเกิดไทม์เอาต์ คอนโทรล MSComm จะกำหนดให้คุณสมบัติ CommEvent มีค่าเป็น comEventCTSTO (Clear To Send Timeout) และกระตุ้นให้เกิดเหตุการณ์ OnComm

17. CDHolding ผู้ใช้งานสามารถตรวจสอบการทำงานของขา Data Carrier Detect (DCD) ได้ว่ามีสถานะลอจิกเป็น “1” หรือ “0” โดยค่าที่อ่านได้จะเป็นบูลีน True และ False ถ้าค่า CDHolding เป็น True ขา DCD จะมีสถานะลอจิก “1” ถ้าค่า CDHolding เป็น False ขา DCD จะมีสถานะลอจิก “0” รูปแบบการใช้งาน

object.CDHolding

เมื่อขา DCD มีลอจิก “1” (CDHolding = True) และเกิดไทม์เอาต์ คอนโทรล MSComm จะกำหนดให้คุณสมบัติ CommEvent มีค่าเป็น comEventCDTO (Carrier Detect Timeout Error) และกระตุ้นให้เกิดเหตุการณ์ OnComm

19. DSRHolding ผู้ใช้งานสามารถตรวจสอบการทำงานของขา DSR ได้ว่ามีสถานะลอจิก “1” หรือ “0” โดยค่าที่อ่านได้จะเป็นบูลีน True และ False ถ้าค่า DSRHolding เป็น True ขา DSR จะมีสถานะลอจิก “1” ถ้าค่า DSRHolding เป็น False ขา DSR จะมีสถานะลอจิก “0” รูปแบบการใช้งาน

object.DSRHolding

เมื่อขา DSR เป็นลอจิก “1” (DSRHolding = True) และเกิดไทม์เอาต์ คอนโทรล MSComm จะกำหนดให้คุณสมบัติ CommEvent มีค่าเป็น comEventDSRTO (Data Set Ready Timeout) และกระตุ้นให้เกิดเหตุการณ์ OnComm

20. Handshaking กำหนดคุณสมบัติและคืนค่ารูปแบบแฮนด์เช็กทางฮาร์ดแวร์ รูปแบบการใช้งานคำสั่ง

object.Handshaking [= value]

ค่าตัวแปร Value ที่ใช้กำหนดค่ากำหนดได้ 4 รูปแบบด้วยกันคือ

1. comNone ค่าที่กำหนดคือ 0 เป็นการกำหนดให้ไม่มีการแฮนด์เช็ก (เป็นค่าเริ่มต้น)
2. comXOnOff ค่าที่กำหนดคือ 1 เป็นการกำหนดให้ใช้แฮนด์เช็กแบบ XON/XOFF
3. comRTS ค่าที่กำหนดคือ 2 เป็นการกำหนดให้ใช้ขา RTS/CTS (Request To Send/Clear To Send)
4. comRTSXOnOff ค่าที่กำหนดคือ 3 เป็นการกำหนดให้ใช้ทั้งแบบ Request To Send และ XON/XOFF

คุณสมบัติ Handshaking ใช้เพื่อกำหนดรูปแบบการสื่อสารภายใน ระหว่างที่ข้อมูลถูกส่งไปยังบัฟเฟอร์ภาครับ เมื่อข้อมูลตัวอักษรถูกส่งมาถึงพอร์ตอนุกรม อุปกรณ์สื่อสารข้อมูลจะทำการย้ายข้อมูลไปยังบัฟเฟอร์ภาครับ เพื่อที่จะให้โปรแกรมสามารถอ่านค่าไปใช้งานได้ ถ้าไม่มีบัฟเฟอร์ภาครับโปรแกรมที่ใช้งานจะต้องทำการอ่านค่าข้อมูลโดยตรงจากฮาร์ดแวร์ของพอร์ตอนุกรม ซึ่ง

ผู้ใช้งานจะเกิดปัญหาข้อมูลสูญหายได้ เนื่องจากว่าการเปลี่ยนแปลงของข้อมูลที่ส่งเข้ามามีการเปลี่ยนแปลงอย่างรวดเร็ว

คุณสมบัติ handshaking ช่วยให้ผู้ใช้งานแน่ใจได้ว่าข้อมูลที่ได้รับมานั้นไม่มีการสูญหายเมื่อบัฟเฟอร์ภาครับที่รับข้อมูลนั้นเกิดข้อมูลล้นหรือโอเวอร์โฟลว (overflow) โดยใช้วิธีการตรวจสอบความพร้อมของบัฟเฟอร์ว่าพร้อมรับข้อมูลหรือไม่ก่อนที่จะส่งข้อมูลมาให้

21. Break ใช้ในการเซตและเคลียร์ค่าสัญญาณ Break ชนิดของข้อมูลเป็นแบบ Boolean รูปแบบการใช้งาน

object.Break [= value]

โดย Value เป็นค่าบูลีน ถ้า Value = True หมายถึง การส่งสัญญาณ Break ออกไป (ขา TxD เป็นลอจิก “1”) ถ้า Value = False หมายถึงการเคลียร์สัญญาณ Break (ขา TxD เป็นลอจิก “0”)

เมื่อกำหนดให้สัญญาณ Break เป็น True จะเป็นการหยุดการส่งข้อมูลชั่วคราวจนกว่าจะมีการสั่งให้สัญญาณ Break เป็น False

ตัวอย่าง เป็นวิธีการส่งสัญญาณ Break ออกไปเป็นช่วงเวลาสั้นๆ ที่ 1/10 ของวินาที

```
MSComm1.Break = True
Duration! = Timer + .1
Do Until Timer > Duration!
Dummy =DoEvents()
Loop
MSComm1.Break = False
```

2.4.6.3 ค่าคงที่คุณสมบัติของคอนโทรล MSComm

ตารางที่ 2.5 แสดงค่าคงที่สำหรับคุณสมบัติ Handshake

ค่าคงที่	ค่า	รายละเอียด
comNone	0	ไม่ใช้การตรวจสอบแฮนด์เชก
comXonXoff	1	ไม่ใช้การตรวจสอบแฮนด์เชกแบบ Xon/Xof
comRTS	2	ใช้ในการตรวจสอบแฮนด์เชกผ่านทางขา RTS และ CTS
comRTSXOnXoff	3	การตรวจสอบแฮนด์เชก แบบ RTS, CTS และXon/Xoff

ตารางที่ 2.6 แสดงค่าคงที่สำหรับคุณสมบัติ OnComm

ค่าคงที่	ค่า	รายละเอียด
comEvSend	1	ส่งค่าเหตุการณ์ (send event)
comEvReceive	2	รับค่าเหตุการณ์ (receive event)
comEvCTS	3	มีการเปลี่ยนแปลงที่ขา CTS
comEvDSR	4	มีการเปลี่ยนแปลงที่ขา DSR
comEvCD	5	มีการเปลี่ยนแปลงที่ขา DCD
comEvRing	6	ตรวจจับสัญญาณกระดิ่งของโทรศัพท์
comEvEOF	7	ตรวจพบตำแหน่งท้ายสุดของไฟล์ (End of file)

ตารางที่ 2.7 แสดงค่าคงที่สำหรับคุณสมบัติ Error

ค่าคงที่	ค่า	รายละเอียด
comEventBreak	1001	ได้รับสัญญาณ Break
comEventCTSTO	1002	ขา CTS เกิดไทม์เอาต์
comEventDSRTO	1003	ขา DSR เกิดไทม์เอาต์
comEventFrame	1004	เกิดข้อผิดพลาดที่เฟรมข้อมูล (Framing error)
comEventOverrun	1006	พอร์ตอนุกรมเกิดโอเวอร์รัน (Port overrun)
comEventCDTO	1007	ขา DCD เกิดไทม์เอาต์
comEventRxOver	1008	บัฟเฟอร์รับข้อมูลเกิดโอเวอร์โฟลว์
comEventRxParit	1009	เกิดข้อผิดพลาดที่พาริตี้ (Parity error)
comEventTxFull	1010	บัฟเฟอร์ส่งข้อมูลเต็ม

ตารางที่ 2.8 แสดงค่าคงที่สำหรับคุณสมบัติ InputMode

ค่าคงที่	ค่า	รายละเอียด
comInputModeText	0	ข้อมูลที่รับมีคุณสมบัติเป็นข้อความ (ค่าปกติ)
comInputModeBinary	1	ข้อมูลที่รับเข้ามาเป็นข้อมูลไบนารี่

2.4.7 ฟังก์ชั่น ค่าคงที่ การส่งค่าพารามิเตอร์ และตัวอย่างโปรแกรมย่อยที่ใช้ในโปรแกรมวิชวล เบสิก

2.4.7.1 ฟังก์ชั่น (Function)

1. ฟังก์ชั่นเกี่ยวกับข้อความ (String)

Instr ฟังก์ชั่นหาตำแหน่งของข้อความ รูปแบบ

Instr Z (ตำแหน่งเริ่มต้นในการนับ,ข้อความ,ข้อความที่ต้องการหาตำแหน่ง)

ตัวอย่าง Instr (“ABCDEFC”,”C”) เป็นการค้นหาตัวอักษร “C” ซึ่งผลลัพธ์คือตำแน่งที่ 3 หรือถ้าต้องการกำหนดตำแหน่งเริ่มต้นในการนับได้โดยเริ่มนับจากตัวที่ 4 ฟังก์ชั่นก็สามารถเขียนได้เป็น

Instr (4,“ABCDEFC”,”C”)

Mid

ฟังก์ชั่นข้อความโดยกำหนดค่าภายในฟังก์ชั่น

รูปแบบ Mid(ข้อความ,ตำแหน่งเริ่มต้น,จำนวนตัวนับจากตำแหน่งเริ่มต้น)

ตัวอย่าง Mid(“I love SoftPress”,3,5) เป็นการตัดข้อความโดยเริ่มต้นจากตัวที่ 3 และนับถัดไปอีก 5 ผลที่ได้คือ love

Space

ฟังก์ชั่นเพิ่มช่องว่างตามจำนวนทที่กำหนดไว้ในฟังก์ชั่น

รูปแบบ Space(จำนวนช่องว่าง)

ตัวอย่าง Space(100) ให้แสดงช่องว่าง 100 ช่อง

2. ฟังก์ชั่นเกี่ยวกับตัวเลข

Val

แปลงค่าตัวเลข

รูปแบบ Val(ตัวเลขที่เป็นข้อความ)

ฟังก์ชั่นนี้มักใช้กับการแปลงค่าจากการรับข้อมูลผ่านคอนโทรลเท็กซ์บอกซ์ ถึงแม้จะพิมพ์ข้อมูลเป็นตัวเลขแต่ข้อมูลดังกล่าวจะกลายเป็นข้อมูลประเภทข้อความโดยอัตโนมัติ ซึ่งถ้าจะนำข้อมูลดังกล่าวมาคำนวณจะต้องใช้ฟังก์ชั่น Val

Round

ฟังก์ชันตัวเลข

รูปแบบ Round (ตัวเลข,จำนวนหลักทศนิยม)

ตัวอย่าง Round(1234.1234,3) ผลที่ได้คือ 1234.123

2.4.7.2 ค่าคงที่ (Constant)

ชนิดของข้อมูล (Data Type)

ชนิดของข้อมูลที่เรากำหนดให้กับตัวแปรนั้นมีจุดประสงค์เพื่อให้การเก็บข้อมูลของเรามีประสิทธิภาพ เพราะใช้พื้นที่ขนาดเหมาะสมในการเก็บข้อมูลโดยในวิชวล เบสิก 6 จะมีชนิดข้อมูลต่างๆดังตารางต่อไปนี้

ตารางที่ 2.9 ชนิดข้อมูลต่างๆที่ใช้ในโปรแกรมวิชวล เบสิก

ชนิดของข้อมูล	ขนาดไบต์	คำอธิบาย
Byte	1	มีค่าตั้งแต่ 0-255
Integer	2	มีค่าตั้งแต่ -32,767 ถึง 32,768
Long	4	มีค่าตั้งแต่ -2,147,483,648 ถึง 2,147,483,647
Single	4	จะแสดงรายละเอียดต่อไป
Double	8	จะแสดงรายละเอียดต่อไป
Currency	8	มีค่าตั้งแต่ -922,337,203,685,477.5808 ถึง -922,337,203,685,477.5807
Dec		จะแสดงรายละเอียดต่อไป
Boolean	1	มีค่า True (แทนค่าด้วยค่า 1) กับ False (แทนค่าด้วยค่า 0)

ตารางที่ 2.9 (ต่อ) ชนิดข้อมูลต่างๆที่ใช้ในโปรแกรมวิชวล เบสิก

String		ใช้ในการเก็บข้อความที่เป็นชุดของตัวอักขระ
Date		เป็นข้อมูลชนิดวันที่และเวลา
Variant		เป็นข้อมูลที่สามารถใช้แทนชนิดของข้อมูลอื่นได้
Object		เป็นข้อมูลชนิดคอนโทรล

หมายเหตุ

Single

เป็นชนิดข้อมูลประเภททศนิยมยกกำลังที่มีความละเอียดแบบ double precision (ความละเอียดสูง) จะใช้เนื้อที่ 8 ไบต์ในการเก็บข้อมูลชนิดนี้ มีค่าดังต่อไปนี้

4.94065645841247E-324 ถึง 1.79769313486232E308 สำหรับเลขบวก

-1.79769313486232E308 ถึง -4.94065645841247E-324

0

Double

เป็นชนิดข้อมูลประเภททศนิยมยกกำลังที่มีความละเอียดแบบ single precision (ความละเอียดต่ำ) จะใช้เนื้อที่ 4 ไบต์ในการเก็บข้อมูลชนิดนี้ มีค่าดังต่อไปนี้

1.40129E-45 ถึง 3.402823E38 สำหรับเลขบวก

-3.402823E38 ถึง -1.40129E-45 สำหรับเลขลบ

0

Dec

เป็นชนิดข้อมูลประเภทตัวเลขที่สามารถเก็บค่าตัวเลขได้ใหญ่มาก ซึ่งจะเก็บตัวเลขจำนวนเต็มที่มีค่าอยู่ระหว่าง -79,228,162,514,264,337,593,543,950,335
ถึง +79,228,162,514,264,337,593,543,950,335 แต่ถ้าเราจะเก็บข้อมูลเป็นเลขทศนิยมจะมีค่าอยู่ระหว่าง -7.9228162514264337593543950335 ถึง +7.9228162514264337593543950335 ใช้เนื้อที่หน่วยความจำ 12 ไบต์ในการเก็บข้อมูลชนิดนี้

2.4.7.3 โปรแกรมย่อย ฟังก์ชั่น และการส่งพารามิเตอร์

วิธีการเขียนโปรแกรมที่ดีคือการแยกโปรแกรมออกเป็นส่วนย่อยๆที่ทำงานอย่างหนึ่งจนเสร็จ ที่เราเรียกว่าโปรแกรมย่อย (Procedure) ข้อดีของการใช้โปรแกรมย่อยคือ ช่วยให้โปรแกรมของเราทำความเข้าใจได้ง่ายขึ้นเพราะมีการแบ่งส่วนย่อยๆ ช่วยให้เรานำโปรแกรมย่อยไปใช้กับโปรแกรมอื่นได้ ถ้าโปรแกรมนั้นต้องการฟังก์ชั่นที่เหมือนหรือใกล้เคียงกัน ลดความซ้ำซ้อนในการเขียนโปรแกรมโดยนำส่วนที่ต้องใช้ซ้ำๆมาเขียนเป็นโปรแกรมย่อย

ในวิชวลเบสิก 6.0 จะมีโปรแกรมย่อยที่ใช้งานทั่วไปอยู่ 2 ประเภทคือ ในโปรแกรมย่อย Sub เป็นโปรแกรมย่อยที่จะทำงานอย่างหนึ่ง โปรแกรมย่อย Function เป็นโปรแกรมย่อยที่จะคือผลลัพธ์ที่ได้จากการทำงานกลับมาด้วย

1. โปรแกรมย่อย Sub เป็นโปรแกรมที่ไม่มีการส่งค่ากลับมา ที่มีรูปแบบดังนี้

```
[Private| Public] [Static] Sub <ชื่อของโปรแกรมย่อย> (พารามิเตอร์ที่ส่งมา)
    ' ชุดคำสั่ง
    [Exit Sub]
End Sub
```

คำว่า Private หรือ Static จะเป็นคำประกาศขอบเขตว่าต้องการให้โปรแกรมย่อยนี้มีขอบเขตอยู่ในเฉพาะในโมดูลนี้ หรือทุกโมดูลในโปรเจคที่สร้างขึ้น

คำว่า Static เป็นการบอกว่า ให้ตัวแปรที่อยู่ในโปรแกรมย่อยนี้ เป็นแบบ Static ทั้งหมด

คำสั่ง Exit Sub จะทำให้ออกจากโปรแกรมย่อยทันที เมื่อพบกับคำสั่งนี้

คำว่า End Sub เป็นคำสั่งที่บอกว่าเป็นการจบการทำงานของโปรแกรมย่อยนี้

2. โปรแกรมย่อย Function เป็นโปรแกรมย่อยที่มีการส่งค่ากลับมาด้วย มีรูปแบบดังนี้

```
[Private| Public] [Static] Function <ชื่อของโปรแกรมย่อย> (พารามิเตอร์ที่ส่งมา) As Type
    ' ชุดคำสั่ง
    [Exit Function]
End Function
```

คำว่า Private หรือ Static จะเป็นคำประกาศขอบเขตว่าต้องการให้โปรแกรมย่อยนี้มีขอบเขตอยู่ในเฉพาะในโมดูลนี้ หรือทุกโมดูลในโปรเจคของเรา

คำว่า Static เป็นการบอกว่า ให้ตัวแปรที่อยู่ในโปรแกรมย่อยนี้ เป็นแบบ Static ทั้งหมด

คำสั่ง Exit Function จะทำให้ออกจากโปรแกรมย่อยทันที เมื่อพบกับคำสั่งนี้

คำว่า End Function เป็นคำสั่งที่บอกว่าเป็นการจบการทำงานของโปรแกรมย่อยนี้

คำว่า As Type หลัง Function ใช้กำหนดชนิดข้อมูลที่ฟังก์ชั่นนี้ส่งกลับมาให้โปรแกรมหลัก

3. การส่งค่าพารามิเตอร์ให้กับโปรแกรมย่อยในการเรียกใช้โปรแกรมย่อยบางครั้ง เราต้องส่งข้อมูลที่โปรแกรมย่อยต้องใช้ในการทำงานจากโปรแกรมหลัก ซึ่งเราเรียกว่าการส่งพารามิเตอร์ การส่งพารามิเตอร์แบ่งเป็น 2 ประเภทคือ

การส่งค่าแบบส่งค่าไปทางเดียว (Pass By Value)

เป็นการส่งค่าของตัวแปรไปอย่างเดียว เมื่อมีการแก้ไขค่านั้นๆ ในโปรแกรมย่อยที่รับค่าเข้ามา จะไม่กระทบต่อค่าของตัวแปรเดียวกันในโปรแกรมหลัก เราจะใช้ว่าคำว่า ByVal อยู่หน้าพารามิเตอร์ที่ส่งพื่อบอกว่าเป็นพารามิเตอร์แบบส่งค่าไปอย่างเดียวเช่น

```
Sub MySub (ByVal Y As Integer)
        Y=Y*3
End Sub
```

ถ้ามีการเรียกโปรแกรมย่อย MySub ดังนี้

```
Dim A As Integer
A = 3
MySub A
```

จากคำสั่งข้างต้น หลังจากการเรียกใช้โปรแกรมย่อย MySub แล้วค่าของ A ในโปรแกรมหลักจะไม่มีการเปลี่ยนแปลงไป คือ จะมีค่า 3 เหมือนเดิม ไม่ใช่ 9

การส่งค่าแบบส่งค่าแบบอ้างอิง (Pass By Reference)

การส่งค่าพารามิเตอร์แบบนี้ จะทำให้โปรแกรมย่อยที่ถูกเรียกใช้สามารถแก้ไขค่าของตัวแปรที่ส่งเป็นพารามิเตอร์ให้แก่โปรแกรมย่อยนั้นได้ ซึ่งการส่งพารามิเตอร์แบบนี้จะเป็นการส่งพารามิเตอร์ที่ วิชวล เบสิก 6.0 ใช้อยู่โดยทั่วไปเช่น

```
Sub MySub (ByRef Y As Integer)
        Y=Y*3
End Sub
```

ถ้ามีการเรียกโปรแกรมย่อย MySub ดังนี้

```
Dim A As Integer
A = 3
MySub A
```

จากคำสั่งข้างต้น หลังจากการเรียกใช้โปรแกรมย่อย MySub แล้วค่าของ A ในโปรแกรมหลักจะมีการเปลี่ยนแปลงเป็น 9 ไม่ใช่ 3 สำหรับคีย์เวิร์ด ByRef นั้นไม่จำเป็นต้อง ใส่ก็ได้ เพราะเนื่องจากเป็นค่าเบื้องต้นอยู่แล้ว

4. DoEvents

ในระบบปฏิบัติการวินโดวส์ เราสามารถทำงานได้หลายๆอย่างพร้อมๆกัน ด้วยคุณลักษณะเฉพาะของวินโดวส์ที่เราเรียกว่า Preemptive Multitasking ด้วยคุณสมบัตินี้ทำให้โปรแกรมสามารถทำงานเบื้องหน้า (ติดต่อกับผู้ใช้ในขณะนั้น) และทำงานเบื้องหลัง (ไม่ได้ติดต่อกับผู้ใช้ แต่จะทำงานตอนที่โปรแกรมเบื้องหน้าหยุดพัก หรือรอการตัดสินใจของผู้ใช้อยู่)

สำหรับการทำงานแบบเบื้องหลังนั้น (Background Processing) โปรแกรมจะไม่ตอบสนองต่ออีเว็นต์ได้เพราะระบบวินโดวส์จะส่งอีเว็นต์จากผู้ใช้ไปให้โปรแกรมเบื้องหน้าเท่านั้นซึ่งปัญหานี้เราสามารถแก้ได้

ในวิชวลเบสิก 6.0 มีคำสั่ง DoEvents ที่ช่วยให้โปรแกรมที่ทำงานเบื้องหลังสามารถตอบสนองกับอีเว็นต์อื่นได้ในระหว่างการทำงานอันยาวนาน เช่นให้กดปุ่ม Cancel ได้ยกเลิก การทำงานได้ในระหว่างการทำสำเนาหลายๆไฟล์ ที่ต้องใช้เวลาในการทำงานซึ่งคำสั่งนี้มีรูปแบบต่อไปนี้

ตัวอย่างการใช้งานคำสั่ง DoEvents

ในโปรแกรมการนับเลข 0 - 10000

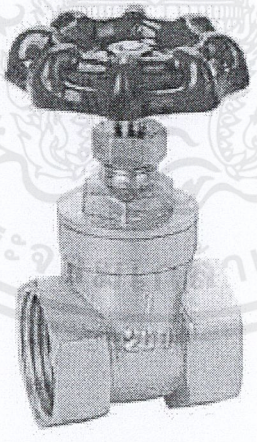
No DoEvents คือ จะไม่มีการแสดงตัวเลขในเท็กซ์บ็อกซ์จนกว่าการนับเลขจะเสร็จสิ้น

DoEvents คือ เมื่อมีการนับเลขเพิ่มขึ้นหนึ่งค่าจะแสดงค่าที่นับได้ในเท็กซ์บ็อกซ์โดยมีการเปลี่ยนแปลงตลอดการนับ

2.5 วาล์วแบบประตู (Gate valve)

Gate valve เป็นวาล์วชนิดหนึ่งที่ใช้กันแพร่หลายมากที่สุดตัวหนึ่ง โครงสร้างของวาล์วนั้นจะมีส่วนที่เป็นแผ่นจาน (disk หรือ gate) ที่มีขนาดใหญ่กว่าขนาดเส้นผ่านศูนย์กลางของท่อเล็กน้อย เลื่อนขึ้น-ลงในทิศทางที่ตั้งฉากกับทิศทางการไหล เมื่อวาล์วอยู่ในตำแหน่งปิด แรงดันของของไหลทางด้าน upstream จะดันตัว disk ให้ไปยันกับตัว body ของวาล์วที่อยู่ทางด้าน downstream เป็นการปิดผนึกไม่ให้ของไหลไหลผ่านไปได้

ข้อดีของ gate valve คือมีความกว้าง (วัดในทิศทางการไหล) ไม่มาก ใช้พื้นที่ในการติดตั้งน้อย ค่าความดันลด (pressure drop) คร่อมวาล์วต่ำมากเมื่อวาล์วเปิดเต็มที่ เหมาะสำหรับงานประเภทปิด-เปิด วาล์วชนิดนี้ไม่เหมาะสำหรับใช้ในการควบคุมการไหลเพราะความสัมพันธ์ระหว่างระยะที่วาล์วเปิดกับอัตราการไหลนั้นไม่ดี (กล่าวคือบางช่วงวาล์วขยับเพียงเล็กน้อยจะมีอัตราการไหลเปลี่ยนแปลงมาก แต่บางช่วงวาล์วขยับไปเยอะแต่อัตราการไหลเปลี่ยนเพียงเล็กน้อย) และไม่เหมาะกับการเปิดหรี่หรือเปิดเพียงเล็กน้อย (crack opening) เช่นหมุน hand wheel เพียงแค่ไม่ถึง 1 รอบ เพียงแค่รู้สึกว่ามีของไหลเริ่มไหลผ่านก็หยุดหมุน เพราะในขณะที่วาล์วเปิดเพียงเล็กน้อยนั้น ของไหลจะไหลผ่านด้วยความเร็วที่สูงมาก และมีความดันที่ต่ำ (pressure head เปลี่ยนไปเป็น velocity head) จะทำให้ตัวแผ่นจานเกิดการสั่นอย่างรุนแรงจนสามารถทำให้ตัวแผ่นจานหรือ seat ของตัว body เองเกิดการสึกหรอได้ ซึ่งจะทำให้ไม่สามารถปิดวาล์วได้สนิทอีกต่อไป



รูปที่ 2.23 โครงสร้าง Gate valve

2.6 เฟืองและชุดเฟืองทดแบบเฟืองขบเฟือง

2.6.1 ส่วนประกอบต่างๆของเฟือง

วงกลมพิทช์ (Pitch circle) – ขนาดของวงกลมที่ใช้ในการคำนวณ วงกลมพิทช์ของเฟืองขับและตามจะมีการสัมผัสกันตลอดเวลา

เฟืองขับ (Pinion) – เฟืองตัวที่เล็กที่สุดของชุดเฟืองคู่

Circular pitch, p_c – ระยะห่างที่วัดได้บนวงกลมพิทช์จากจุดหนึ่งบนฟันหนึ่ง ไปยังจุดในตำแหน่งเดียวกันของฟันถัดไป

Diamentral pitch, P_d – จำนวนฟันของเฟืองต่อหนึ่งหน่วยความยาวของเส้นผ่านศูนย์กลางของวงกลมพิทช์ หน่วยของมันจะมีค่าเป็น ส่วนกลับของนิ้ว (1/นิ้ว)

แอดเดนดัม (Addendum), a – ระยะในแนวรัศมีที่วัดจาก วงกลมพิทช์ไปยังระยะสูงสุดของฟัน

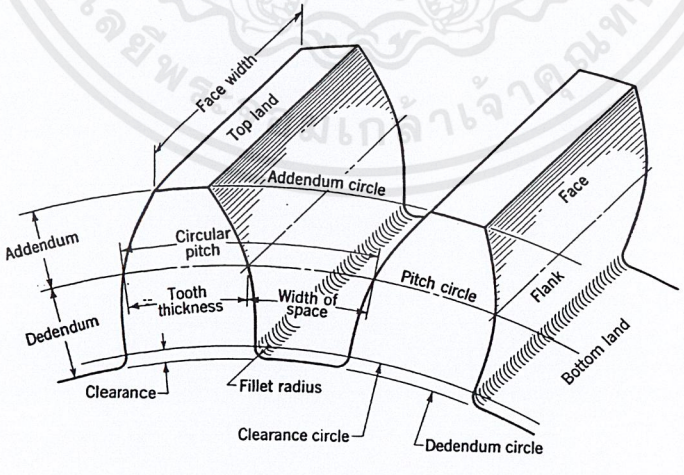
ดีเดนดัม (Dedendum), d – ระยะในแนวรัศมีที่วัดจากระยะต่ำสุดของฟันไปยังวงกลมพิทช์

Whole depth, h_t – ผลบวกของแอดเดนดัมและดีเดนดัม (ความสูงของฟัน)

Clearance circle – วงกลมที่สัมผัสกับวงกลมแอดเดนดัมของคู่เฟืองขับ

Clearance, c – ระยะความแตกต่างของดีเดนดัมของเฟืองตัวหนึ่งกับระยะแอดเดนดัมของเฟืองที่มาขบด้วย

Backlash – ขนาดของช่องว่างระหว่างฟันลบด้วยความหนาของฟันที่มาขบ โดยค่า นี้จะวัดที่วงกลมพิทช์



รูปที่ 2.24 ส่วนประกอบต่างๆของเฟือง

2.6.2 สมการที่สำคัญในการคำนวน

$$P_d = \frac{N}{D} \tag{2.1}$$

เมื่อ P_d คือ Diametral pitch, teeth per inch

N คือ จำนวนฟัน

D คือ ขนาดของวงกลมพิทช์ (นิ้ว)

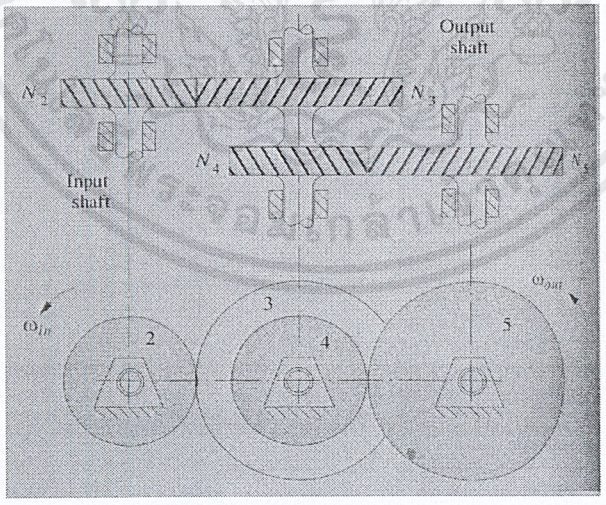
$$p_c = \frac{\pi D}{N} \tag{2.2}$$

เมื่อ P_c คือ circular pitch, in.

P_d คือ π

2.6.3 ฟันเฟืองแบบขบวนเฟือง

ถ้าต้องการอัตราทดมากกว่า 10:1 เราต้องฟันเฟืองแบบขบวนเฟืองในขบวนเฟืองแบบนี้จะมีเฟืองสองตัวอยู่บนเพลาอันเดียวกัน ดังนั้นมันจะมีอัตราเร็วเชิงมุมเท่ากัน



รูปที่ 2.25 ขบวนเฟืองที่เฟือง 3 และ 4 อยู่บนเพลาเดียวกัน

อัตราทดของขบวนเฟืองชุดนี้

$$m_v = \left(-\frac{N_2}{N_3}\right)\left(-\frac{N_4}{N_5}\right) = \frac{N_2 N_4}{N_3 N_5} \qquad (2.3)$$

2.7 มอเตอร์กระแสตรง

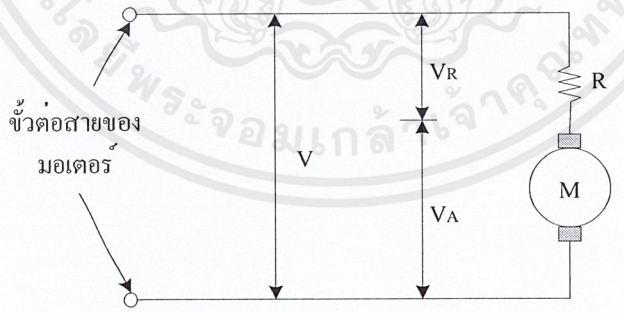
2.7.1 หลักการทำงานของมอเตอร์กระแสตรง

เมื่อมีการผ่านกระแสไฟฟ้าเข้าไปยังขดลวดในสนามแม่เหล็กจะทำให้เกิดแรงแม่เหล็กซึ่งมีสัดส่วนของแรงขึ้นกับกระแสแรงของสนามแม่เหล็ก โดยแรงจะเกิดขึ้นเป็นมุมฉากกับกระแสและสนามแม่เหล็ก ขณะที่ทิศทางของแรงกลับตรงกันข้ามกัน ถ้าหากกระแสของสนามแม่เหล็กไหลย้อนกลับจะทำให้เกิดการเปลี่ยนแปลงของกระแส และ สนามแม่เหล็กเป็นผลทำให้ทิศทางของแรงเปลี่ยนไป ด้วยคุณสมบัตินี้ทำให้มอเตอร์กระแสตรงกลับทิศทางการหมุนได้

สนามแม่เหล็กของมอเตอร์ส่วนหนึ่งเกิดขึ้นจากแม่เหล็กถาวรซึ่งจะถูกยึดติดกับแผ่นเหล็กหรือ เหล็กกล้า โดยปกติส่วนนี้จะเป็นส่วนที่ยึดอยู่กับที่ และ ขดลวดเหนี่ยวนำจะพันอยู่กับส่วนที่เป็นแกนหมุนของมอเตอร์

2.7.2 คุณสมบัติของมอเตอร์กระแสตรง

ในการอธิบายคุณสมบัติของมอเตอร์กระแสตรงให้ละเอียดนั้นต้องพิจารณาแรงดันที่ป้อนและความต้านทานของโรเตอร์ด้วย วงจรภายในของมอเตอร์เขียนได้ดังรูปที่ 1



รูปที่ 2.26 วงจรภายในของมอเตอร์กระแสตรง

โดยสมมติให้ทุ่นโรเตอร์ไม่มีความต้านทานอยู่เลย อนุกรมกับความต้านทานซึ่งในที่นี้ก็คือความต้านทานของขดลวดนั่นเอง แรงดันที่ขั้วต่อสายของมอเตอร์ก็คือผลบวกระหว่างแรงดันที่ทุ่นโรเตอร์ (V_A) และ แรงดันตกคร่อมความต้านทานขดลวด (V_R)

แรงดัน V_A ถูกเรียกว่า แรงเคลื่อนเหนี่ยวนำป้อนกลับ (BACK EMF) ซึ่งเกิดขึ้นในโรเตอร์ ขณะที่หมุนแรงดันที่เกิดขึ้นนี้เป็นไปตากกฎของการเหนี่ยวนำแม่เหล็กไฟฟ้าจากการเคลื่อนที่ของตัวนำในสนามแม่เหล็ก สัมพันธ์กับแรงเคลื่อนเหนี่ยวนำแม่เหล็ก และ ความเร็วในการเคลื่อนที่ของตัวนำ แรงดันที่เกิดขึ้นจะมีขั้วตรงกันข้ามกับแรงดันที่ป้อนให้กับมอเตอร์ และ แปรผันตรงกับความเร็วในการหมุน ผลบวกของแรงดันที่ทุ่นโรเตอร์ (V_A) และแรงดันตกคร่อมขดลวด (V_R) ต้องเท่ากับแรงดันที่ป้อนให้กับมอเตอร์ (V)

$$V = V_A + V_R \qquad (V) \tag{2.4}$$

เมื่อพิจารณาตั้งแต่มอเตอร์หยุดนิ่ง ความเร็วมีค่าเป็นศูนย์ ดังนั้น $V_A = 0$, $V_R = V$ กระแสที่ไหลในมอเตอร์หาได้จาก

$$I = V_R / R \qquad (A) \tag{2.5}$$

เมื่อมอเตอร์เริ่มหมุนจะมีความเร็ว และ V_A เพิ่มขึ้นเป็นเส้นตรงตามความเร็ว V_R ซึ่งมีค่าเท่ากับความแตกต่างระหว่าง V_A และ V จะเริ่มลดลงกระแส I ก็จะเริ่มลดลงเช่นกันขณะที่มอเตอร์ยังมีความเร่งอยู่ ความเร็วจะเพิ่มขึ้น แรงบิดจะลดลงจนกว่าจะถึงจุดซึ่งแรงบิดของมอเตอร์รับภาระโหลดได้สมดุลพอดี ขณะที่มอเตอร์ไม่มีโหลด และ หมุนอย่างอิสระจะมีเพียงค่าความฝืดของแบริ่ง และ แรงต้านอากาศทำให้ V_A เกือบเท่ากับค่า V

2.8 เซนเซอร์

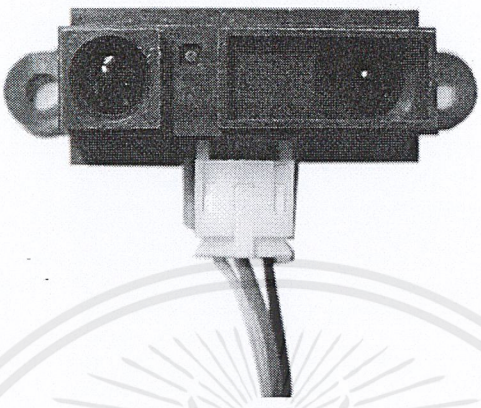
2.8.1 อุปกรณ์ตรวจวัดระยะทาง (Distance Measuring Sensor)

อุปกรณ์ตรวจวัดระยะทาง (GP2Y0A21YK) เป็นอุปกรณ์ที่ใช้ตรวจวัดระดับน้ำในถัง ซึ่งสามารถตรวจวัดระยะทางได้ 10 ถึง 80 เซนติเมตร และให้ค่าเอาต์พุตเป็นแรงดันไฟฟ้า

คุณสมบัติ

1. สีผิวของวัตถุต่าง จะมีผลกระทบน้อยมากต่อการสะท้อนสัญญาณของตัวเซนเซอร์ จะมีผลบ้างในวัตถุที่มีสีดำ ซึ่งเป็นสีที่ทำให้การสะท้อนของสัญญาณอินฟราเรดทำได้ไม่ดีนัก
2. ระยะทางในการตรวจจับวัตถุ 10 ถึง 80 เซนติเมตร
3. ให้ค่าเอาต์พุตเป็นแรงดันไฟฟ้า (Analog Voltage)

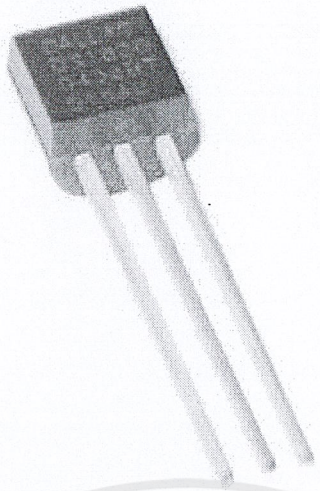
4. ไม่จำเป็นต้องมีวงจรควบคุมภายนอกเพียงแค่จ่ายไฟเลี้ยง (Vcc, GND) ก็สามารถนำเอาต์พุตของตัวเซนเซอร์ไปใช้งานได้เลย ซึ่งแรงดัน Vcc ที่ใช้จะอยู่ในช่วง 4.5V ถึง 5.5V โดยปกติจะใช้ที่ 5V



รูปที่ 2.27 อุปกรณ์ตรวจวัดระยะทาง

2.8.2 ตัววัดอุณหภูมิ (Temperature Sensor)

ไอซี DS1820 เป็นไอซีที่มีระบบการสื่อสารข้อมูลอนุกรมแบบ 1 สาย ซึ่งถือได้ว่าเป็นระบบที่มีความชาญฉลาดและใช้จำนวนสายสัญญาณเพียง 1 เส้นเท่านั้น โดยไม่ต้องมีสายสัญญาณ Clock มาควบคุมการถ่ายทอดข้อมูล เหมือนกับระบบสื่อสารข้อมูลอนุกรมในแบบอื่นๆ สายข้อมูลจะทำหน้าที่เสมือนเป็นสัญญาณ Clock ในตัว ส่วนค่าของข้อมูลจะพิจารณาจากลักษณะของรูปสัญลักษณ์ ที่ปรากฏบนสายสัญญาณ ในแต่ละช่องของเวลา ซึ่งเรียกว่า “Time-Slot” โดยคาบเวลาต่ำสุดและสูงสุดของสถานะต่างๆ ในการสื่อสารข้อมูลในแต่ละ Time-Slot มีการกำหนดขอบเขตไว้อย่างชัดเจน การถ่ายทอดข้อมูลจะเกิดขึ้นในแต่ละ Time-Slot นั้น รูปแบบการถ่ายทอดข้อมูลจะเป็นแบบ Asynchronous ในระดับ bit ไม่มีการกำหนดความยาวของข้อมูลเป็นระดับ byte ระบบสื่อสารแบบนี้เหมาะที่จะใช้ในการสื่อสารข้อมูลระหว่างไอซีบนแผงวงจรเดียวกัน



รูปที่ 2.28 ไอซี DS 1820

2.8.2.1 คุณสมบัติของไอซี DS1820

1. อินเตอร์เฟสสัญญาณผ่านขาเอาท์พุตเพียงพอร์ตเดียวแบบ 1 สายข้อมูล (1 - WireTM)

2. ขยายจุดตรวจจับอุณหภูมิได้หลายๆ จุดบนสายข้อมูลเพียง 1 สายข้อมูล

3. ไม่ต้องใช้อุปกรณ์ภายนอกมาต่อร่วม

4. สามารถควบคุมการทำงานเพาเวอร์ออนได้ผ่านทางสายข้อมูล

5. เพาเวอร์ขณะสแตนด์บายเป็นศูนย์

6. ย่านการวัดอุณหภูมิตั้งแต่ -55 องศาเซลเซียสถึง +125 องศาเซลเซียสที่ 0.5 องศาเซลเซียสต่อสเต็ป หรือตั้งแต่ย่าน -67 องศาฟาเรนไฮ ถึง +257 องศาฟาเรนไฮที่ 0.9 องศาฟาเรนไฮต่อสเต็ป

7. อุณหภูมิจะถูกอ่านออกมาเป็นค่าทางดิจิตอล 9 บิต

8. อัตราความเร็วในการแปลงจากอุณหภูมิมาเป็นค่าตัวเลขทางดิจิตอลเท่ากับ 200 มิลลิวินาที

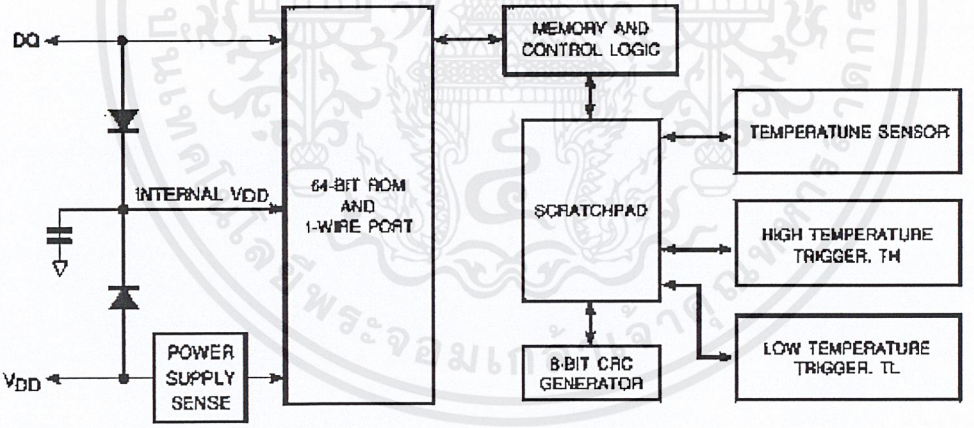
9. ผู้ใช้งานสามารถกำหนดการเซตค่าเตือนย่านอุณหภูมิได้ในแบบ non – volatile

10. การเตือนย่านอุณหภูมินั้นสามารถกำหนดรหัสผ่านการสั่งการและแอดแดรสของอุปกรณ์ได้จากภายนอกพื้นที่ตรวจวัดอุณหภูมิผ่านทางโปรแกรมภายนอก

11. เหมาะกับการประยุกต์ใช้งานตรวจวัดอุณหภูมิและติดตั้งไว้ในอุปกรณ์ควบคุมเทอร์โมสแตติก, ระบบโรงงานอุตสาหกรรม, ผลิตภัณฑ์, เทอร์โมมิเตอร์ หรือระบบอื่นๆ ที่มีส่วนตรวจจับอุณหภูมิทำงานร่วมอยู่

2.8.2.2 บล็อกไดอะแกรมภายในของ DS1820

บล็อกไดอะแกรมส่วนประกอบของการทำงานต่างๆ ภายในตัว DS1820 มีส่วนประกอบหลักๆ 3 ส่วนด้วยกันคือ หน่วยความจำเลเซอร์รอมขนาด 64 บิต, ส่วนเซ็นเซอร์อุณหภูมิและส่วนกระตุ้นเตือนอุณหภูมิแบบ non – volatile (TH และ TL) โดยอุปกรณ์ตรวจวัดอุณหภูมินี้จะถูกควบคุมสภาวะการเพาเวอร์ออนและเพาเวอร์ออฟจากไลน์ข้อมูลเพียง 1 สายข้อมูลจากการเก็บรักษากำลังงานสำรองไว้ในตัวเก็บประจุภายใน ในช่วงระหว่างคาบเวลาเมื่อสัญญาณภายในไลน์มีสถานะเป็น high และจะทำงานต่อเนื่องไปเรื่อยๆ และการหยุดการทำงานก็จะเกิดขึ้นจากการหยุดจ่ายแหล่งจ่ายในช่วงระหว่างค่าเวลานั้นเป็น low ของไลน์ข้อมูลและจะหยุดอยู่เช่นนั้นจนกว่าขาไลน์ข้อมูลจะกลับมาเป็น high อีกครั้ง และแหล่งจ่ายไฟหลักนี้ก็จะได้จากแหล่งจ่ายไฟ +5 โวลต์ภายนอก การติดต่อข้อมูลกับ DS 1820 จะติดต่อผ่านพอร์ตเพียงพอร์ตเดียวคือ 1 – Wire port ภายในพอร์ต 1 – Wire นี้ในส่วนของหน่วยความจำและควบคุมฟังก์ชันโปรโตคอลของรอมจะถูกทำการเซตค่าเสียก่อน ในส่วนสำคัญของการทำงานฟังก์ชันอันดับแรกซึ่งเป็นหนึ่งในห้าลำดับของการสั่งการฟังก์ชันในรอมก็คือการอ่านหน่วยความจำรอมทำการแมตช์รอมค้นหารอมกระโดดข้ามรอมเตือนการค้นหา



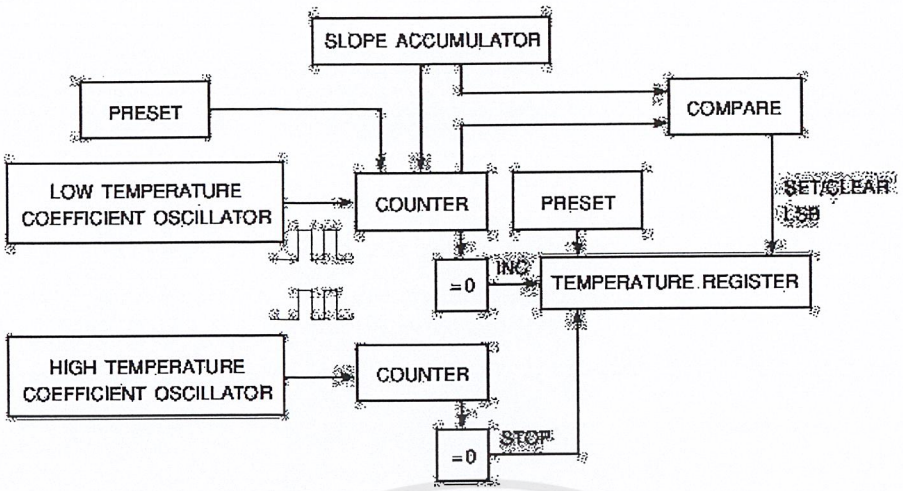
รูปที่ 2.29 บล็อกไดอะแกรมภายในของ DS1820

2.8.2.3 การทำงานของ DS 1820

ซึ่งการทำงานของระบบการสั่งการนี้จะทำงานบนพื้นที่หน่วยความจำเลเซอร์รอมขนาด 64 บิต ผ่านพอร์ตไอซีแต่ละตัวและสามารถให้เอาต์พุตเดี่ยวเพื่อการกำหนดคุณสมบัติของอุปกรณ์ตรวจจับอุณหภูมินี้หลายๆ ตัวก็ทำได้โดยสั่งการผ่านไลน์ข้อมูล 1- wire นี้ หลังจากฟังก์ชันในรอมถูกลำดับการทำงานแล้ว ก็พร้อมที่จะถูกใช้งาน และสามารถที่จะเข้าถึงการทำงานภายในตัวไอซีได้

ทั้งหมด หน่วยความจำและส่วนควบคุมฟังก์ชั่นก็จะถูกเข้าถึงการทำงานได้ และส่วนจัดเก็บค่าที่เซต ไว้สามารถหรืออาจจะถูกเก็บไว้ในพื้นที่ 1 ส่วนจากทั้งหมด 6 ส่วนของหน่วยความจำและส่วน ควบคุมฟังก์ชั่นการสั่งการส่วนควบคุมฟังก์ชั่นการสั่งการหนึ่งส่วนจะถูกกำหนดคุณสมบัติของ DS1820 ให้อยู่ในรูปแบบของการวัดค่าของอุณหภูมิซึ่งผลของการวัดนี้จะถูกบันทึกไว้ใน DS1820 ในส่วนของหน่วยความจำส่วนหนึ่ง (scratchpad) และบางครั้งก็จะอ่านออกมาได้จากตารางสารบัญ ของหน่วยความจำฟังก์ชันการสั่งการซึ่งเป็นการอ่านออกมาเฉพาะหัวข้อที่ถูกบันทึกไว้ใน หน่วยความจำ scratchpad สัญญาณกระตุ้นเตือนค่าอุณหภูมิสูงเกินและต่ำเกิน (TH และ TL) จะ ประกอบด้วย 1 ไบต์ EEPROM ถ้าสัญญาณการเตือนการค้นหาไม่ถูกจ่ายเข้าไปยัง DS1820 รีจิสเตอร์เหล่านี้บางครั้งจะถูกใช้ได้อย่างทั่วๆ ไปจากหน่วยความจำที่ผู้อ่านกำหนดได้ และการ เขียนเข้าไปในส่วนของการเตือน TH และ TLจะไม่ใช้หน่วยความจำฟังก์ชันการสั่งการและการเข้า ไปถึงรีจิสเตอร์นี้จะอ่านผ่านหน่วยความจำ scratchpad และข้อมูลอื่นๆ ที่ต้องการอ่านและเขียนจะ กระทำได้ในบิตแรกของ LSBการวัดอุณหภูมิ

DS1820 จะทำการวัดค่าวัดอุณหภูมิโดยอาศัยเทคนิคการวัดแบบออนบอร์ดพิเศษซึ่งเป็น เทคนิคการวัดโดยเฉพาะของอุปกรณ์ชนิดนี้ ในบล็อกไดอะแกรมของวัดค่าอุณหภูมิของ DS1820 ซึ่งจะอาศัยการวัดอุณหภูมิโดยการนับจำนวนวงรอบของสัญญาณนาฬิกาที่ออสซิลเลเตอร์ผลิต ขึ้นมา ช่วงเวลาเกตของสัญญาณนาฬิกาที่ออสซิลเลเตอร์ผลิตขึ้นมาจะเป็นการกำหนดได้จากช่วง คาบเวลาที่ค่าสัมประสิทธิ์อุณหภูมิต่ำสุด ไปจนถึงค่าสัมประสิทธิ์อุณหภูมิสูง ซึ่งจะมีค่าความถี่ สัญญาณนาฬิกาที่ไม่เท่ากัน โดยที่ค่าการนับตัวเลขจะเริ่มนับที่ค่าอุณหภูมิต่ำสุดพื้นฐาน คือ –55 องศาเซลเซียส ถ้าการนับสัญญาณนาฬิกามาถึงค่าศูนย์ก่อนที่เวลาเกตจะเกินมา รีจิสเตอร์อุณหภูมิก็ จะแสดงผลที่ค่า –55 องศาเซลเซียส ถ้าหากค่าอุณหภูมิเพิ่มขึ้น การแสดงผลของอุณหภูมิขณะนั้นก็ จะสูงกว่า –55 องศาเซลเซียส ในทำนองเดียวกันนี้ การตั้งค่าของการนับจะกำหนดได้จากการเพิ่ม ความลาดลงของวงจรนับ ซึ่งวงจรนี้ต้องการการชดเชยสำหรับการแสดงคุณสมบัติของส่วนโค้งของ ออสซิลเลเตอร์ที่อุณหภูมิมีค่าเกินมา วงจรนับก็จะนับสัญญาณนาฬิกาอีกครั้งจนกว่ามันจะได้ค่าเป็น ศูนย์ ถ้าคาบเวลาเกตอยู่ในสภาวะสงบนิ่งไม่มีการปรับแต่งก็จะเกิดการประมวลผลใหม่อีกครั้งหนึ่ง การคำนวณค่าภายใน DS1820 จะให้ความละเอียด 0.5 องศาเซลเซียสต่อสเต็ปของการเปลี่ยนแปลง อุณหภูมิการอ่านค่าอุณหภูมิจะถูกกำหนดไว้ภายใน 16 บิตโดยมีนัยสำคัญ 2ส่วนประกอบ ความสัมพันธ์ของข้อมูลทางเอาต์พุทกับการจัดอุณหภูมิ ข้อมูลจะถูกส่งออกมาเป็นอนุกรมบนการ อินเตอร์เฟสกับสายข้อมูล 1- wire ซึ่ง DS1820 สามารถทำการวัดค่าอุณหภูมิได้เกินย่านตั้งแต่ย่าน การวัดอุณหภูมิตั้งแต่ -55 องศาเซลเซียสถึง +125 องศาเซลเซียส



รูปที่ 2.30 บล็อกไดอะแกรมการวัดค่าอุณหภูมิ

ต่อสเต็ป ค่าอุณหภูมิที่ถูกทำการปรับตั้งไว้ใน DS1820 ในเทอมของ 1/ 2 องศาเซลเซียส LSB ซึ่งจะเป็นไปตามรูปแบบของข้อมูล 9 บิต

รูปที่ 2.31 แสดง-25 องศาเซลเซียส

ที่ MSB บิตเป็นคู่เปรียบเทียบกับทุกบิตใน MSB สูงสุดของรีจิสเตอร์อุณหภูมิขนาด 2 ไบต์ ในหน่วยความจำซึ่งการอ่านค่าอุณหภูมิแบบ 16 บิต ในลักษณะสำคัญต่างๆ

ตารางที่ 2.10 การแบ่งส่วนในหน่วยความจำรอมขนาด 64 บิต

ค่าอุณหภูมิ	ดิจิตอลเอาต์พุต (Binary)	ดิจิตอลเอาต์พุต (Hex)
+125°C	00000000 11111010	00FA
+25°C	00000000 00110010	0032h
+1/2°C	00000000 00000001	0001h
+0°C	00000000 00000000	0000h
-1/2°C	11111111 11111111	FFFFh
-25°C	11111111 11001110	FFCEh
-55°C	11111111 10010010	FF92h

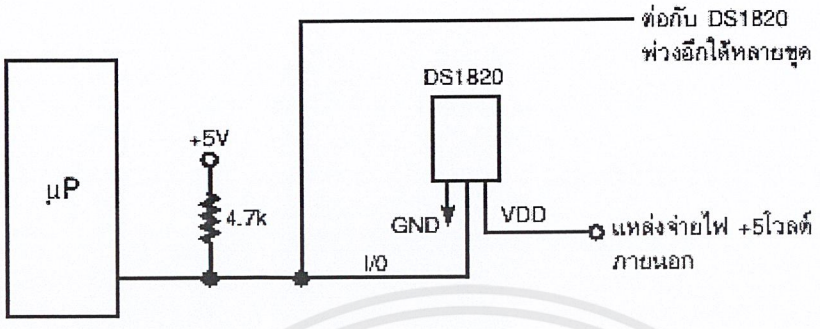
2.8.2.4 การทำงานของสัญญาณเตือน

หลังจากที่ DS1820 มีการตรวจวัดอุณหภูมิเกิดขึ้นแล้วค่าของอุณหภูมิก็จะทำการเปรียบเทียบเพื่อทำเป็นสัญญาณกระตุ้น การการเปรียบเทียบค่าของอุณหภูมิจะเปรียบเทียบกับค่าที่ถูกบันทึกหรือกำหนดไว้ของค่าอุณหภูมิสูงสุด (TH) และค่าอุณหภูมิต่ำสุด (TL) ตลอดย่านอุณหภูมิที่วัดได้ โดยจะใช้พื้นที่รีจิสเตอร์ 8 บิต สำหรับการทำงานนี้ใน MSB ของ TH หรือ TL ที่ตรงกันก็จะถูกส่งไปยัง SB ของรีจิสเตอร์อุณหภูมิขนาด 16 บิต ถ้าผลของการวัดอุณหภูมิมีค่าสูงเกินกว่า TH หรือต่ำกว่า TL ลำดับสัญญาณเตือนภายในอุปกรณ์ก็จะถูกเซต ซึ่งลำดับของสัญญาณเตือนนี้จะถูกอัพเดตทุกครั้งที่มีการวัดค่าอุณหภูมิ เมื่อลำดับสัญญาณเตือนถูกเซต DS1820 จะมีการตอบสนองนำไปสู่การค้นหาสัญญาณเตือนการสั่งการและจะยอมให้ทำการต่อ DS1820 ในลักษณะขนานกันหลายตัวได้ เพื่อทำการจำลองการวัดค่าอุณหภูมิแล้วนำมาเฉลี่ยค่าของการวัดในครั้งนั้นอีกขั้นหนึ่ง

2.8.2.5 การต่อแหล่งจ่ายไฟ

การต่อ DS1820 ร่วมกับไมโครโปรเซสเซอร์ เพื่อการควบคุมจากระยะไกล จะสังเกตเห็นว่าที่ขารับไฟเลี้ยง VDD ของ DS1820 นั้นจะต่อกับกราวด์ แต่จะได้รับไฟเลี้ยงมาทางขา I / O ข้อมูลในแบบ 1 – Wire จากระบบควบคุมหลักไมโครคอนโทรลเลอร์แทน ซึ่งวิธีนี้จะใช้สำหรับการควบคุมจากระยะไกลและไม่ต้องจัดหาแหล่งจ่ายไฟภายนอกให้กับ DS1820 ให้ยุ่งยากเพราะในการอินเตอร์เฟสแบบ 1 – Wire นี้สามารถที่จะทำการกำหนดการทำงาน (power on) ของไอซีให้จ่ายไฟเลี้ยงวงจรภายในตัวไอซีแทน ในรูปที่ 2.14 การจัดแหล่งจ่ายไฟภายนอกให้กับDS1820 ณ จุดที่ติดตั้งใช้งาน และการสั่งการจากไมโครโปรเซสเซอร์ยังทำงานได้เหมือนเดิมและสามารถที่จะต่อ

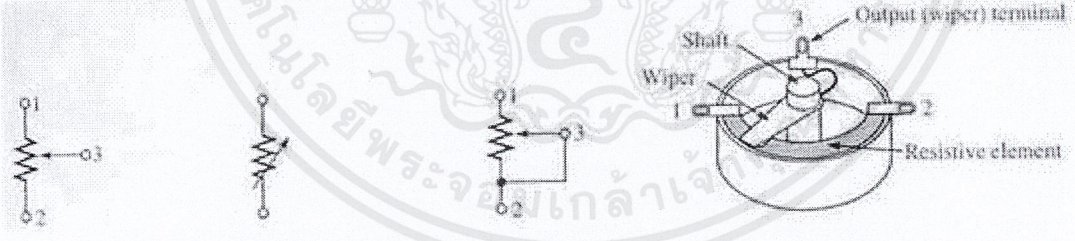
DS1820 ขนานกันโดยใช้การอินเตอร์เฟสผ่านบัส 1 – Wire เดียวกันได้หลายตัว จึงเหมาะกับเป็นการตรวจวัดอุณหภูมิหลายจุดที่ควบคุมได้จากระยะไกลผ่านระบบไมโครคอนโทรลเลอร์



รูปที่ 2.32 การจัดแหล่งจ่ายไฟและการอินเตอร์เฟสร่วมของ DS1820 หลายตัวบนบัส 1 – Wire

2.8.3 โพเทนชิโอมิเตอร์

โพเทนชิโอมิเตอร์หรือพอต (Pot) คือตัวต้านทานที่เปลี่ยนค่าได้ในวงจรต่าง ๆ โครงสร้างส่วนใหญ่จะใช้วัสดุประเภทคาร์บอนผสมกับเซรามิคและเรซินวางบนฉนวน ส่วนแกนหมุนขากลางใช้โลหะที่มีการยืดหยุ่นตัวได้ดี โดยทั่วไปจะเรียกว่าโวลุ่มหรือ VR (Variable Resistor) มีหลายแบบที่นิยมใช้ในปัจจุบันคือแบบ A, B และ C

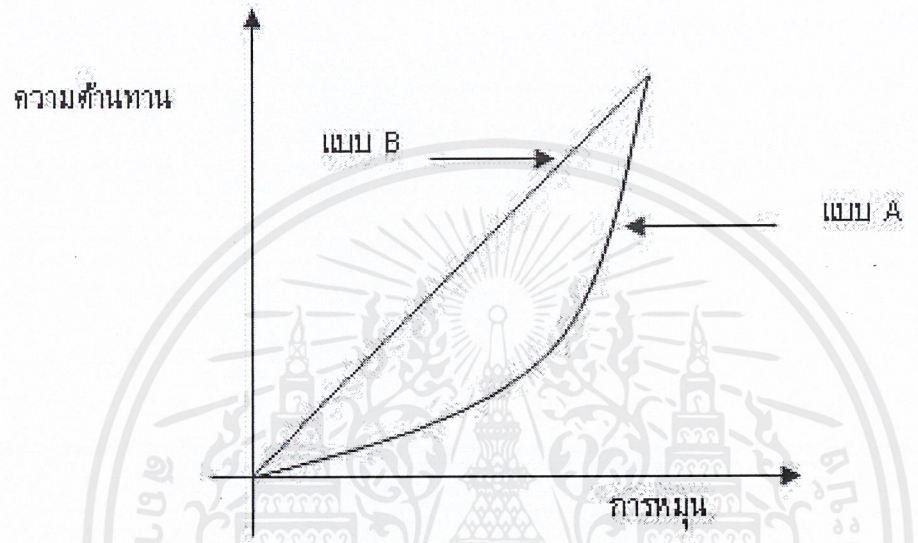


รูปที่ 2.33 แสดงลักษณะรูปร่างและสัญลักษณ์ของโพเทนชิโอมิเตอร์และรีโอสตาท

จะเห็นว่าโพเทนชิโอมิเตอร์มี 3 ขา ซึ่งขาที่ 1 และ 2 จะมีค่าคงที่ ส่วนขาที่ 3 เปลี่ยนแปลงขึ้น-ลงตามที่ต้องการ ส่วนรีโอสตาทนั้นจะมี 2 ขา แต่ในกรณีที่ต้องการต่อโพเทนชิโอมิเตอร์ให้เป็นรี-โอสตาทก็ทำได้โดยการต่อขาที่ 3 เข้ากับขาที่ 2 ก็จะกลายเป็นรีโอสตาท ส่วนรูปสุดท้าย แสดงโครงสร้างทั่ว ๆ ไปของโพเทนชิโอมิเตอร์

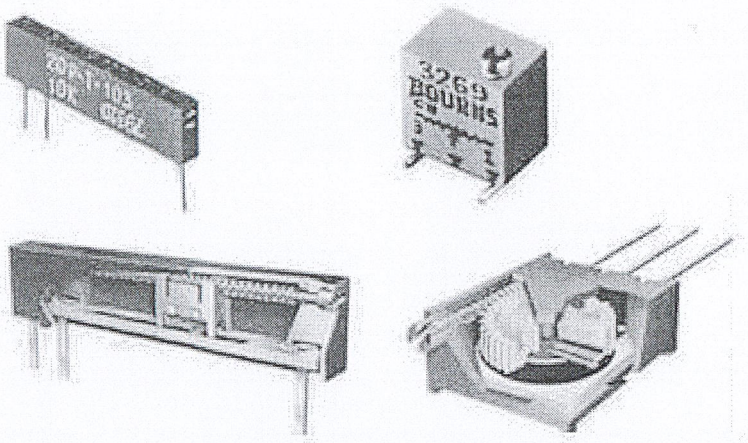
อีกชนิดหนึ่งคือจำพวกฟิล์มคาร์บอนใช้วิธีการฉาบหรือพ่นฟิล์มคาร์บอนลงในสารที่มีโครงสร้างแบบเฟโนลิค (Phenolic) ส่วนแกนหมุนจะใช้โลหะประเภทที่ใช้ทำสปริงเช่นเดียวกัน ตัวอย่างเช่น VR 100 KA หมายความว่า การเปลี่ยนแปลงค่าความต้านทานต่อการหมุนในลักษณะ

ของลอกการิทึม (Logarithmic) หรือแบบล็อกคือหมุนค่าความต้านทานจะค่อย ๆ เปลี่ยนค่า พอถึงระดับกลางค่าความต้านทานจะเปลี่ยนแปลงอย่างรวดเร็วนิยมใช้เป็นโวลลุ่มเร่งความดังของเสียง ส่วนแบบ B นั้นค่าความต้านทานจะเปลี่ยนไปในลักษณะแบบลิเนีย (Linear) หรือเชิงเส้นคือค่าความต้านทานเพิ่มขึ้นตามการหมุนที่เพิ่มขึ้น ส่วนมากนิยมใช้ในวงจรชุดควบคุมความทุ้มแหลมและวงจรแบ่งแรงดัน



รูปที่ 2.34 ความสัมพันธ์ของการเปลี่ยนแปลงค่าความต้านทานแบบ A และแบบ B

ตัวต้านทานแบบโพเทนชิโอมิเตอร์อีกประเภทหนึ่งคือ ตัวต้านทานแบบปรับละเอียด (Trimmer Potentiometers) ตัวต้านทานแบบนี้ส่วนมากมักใช้ประกอบในวงจรประเภทเครื่องมือวัดและทดสอบ เพราะสามารถปรับหมุนเพื่อต้องการเปลี่ยนค่าความต้านทานได้ทีละน้อยและสามารถหมุนได้ 15 รอบหรือมากกว่า ซึ่งเมื่อเทียบกับโพเทนชิโอมิเตอร์แบบที่ใช้ในเครื่องรับวิทยุและเครื่องเสียง ซึ่งจะหมุนได้ไม่ถึง 1 รอบก็จะทำให้ค่าความต้านทานเปลี่ยนแปลงอย่างรวดเร็ว



รูปที่ 2.35 ลักษณะของตัวต้านทานแบบโพเทนชิโอมิเตอร์แบบปรับละเอียด

2.9 เครื่องสูบน้ำ

เครื่องสูบน้ำหรือปั๊มน้ำ เป็นอุปกรณ์สำหรับส่งน้ำหรือถ่ายเทของเหลวจากที่หนึ่งไปยังอีกที่หนึ่ง หรือ หมุนเวียนน้ำหรือของเหลวให้ผสมกันในบริเวณที่จำกัด เช่น Centrifugal pump

ประเภทของเครื่องสูบน้ำแบ่งตามลักษณะการทำงานออกเป็น 2 คือแบบอาศัยแรงกลไกการเหวี่ยงหนีศูนย์ของๆเหลวในการพาของเหลว และแบบอาศัยการแทนที่ของๆเหลวในการพาของเหลวเครื่องสูบน้ำอัตโนมัติ เหมาะสำหรับอาคาร ตึกแถว ทาวน์เฮ้าส์ บ้านเดี่ยวเป็นระบบสวิตซ์เปิด-ปิดอัตโนมัติประหยัดไฟกำลังส่งไปยังจุดต่างๆภายในบ้านได้ดี สามารถต่อกับเครื่องทำน้ำอุ่น เครื่องซักผ้า หรือก๊อกน้ำได้ เหมาะสำหรับอาคารตึกแถว ทาวน์เฮ้าส์ บ้านเดี่ยว เป็นเครื่องสูบอัตโนมัติควบคุมแรงดันคงที่ให้น้ำสม่ำเสมอ เหมาะกับการติดตั้งใช้กับเครื่องทำน้ำอุ่น ไม่เป็นสนิมตลอดอายุการใช้งาน เครื่องสูบน้ำหอยโข่ง เหมาะกับงานเกษตร งานสูบน้ำขึ้นตึกสูง งานสูบจากแท็งค์หรือบ่อ งานหัวจ่ายน้ำ Sprinkle สามารถสูบน้ำได้ในปริมาณที่มากหรือแรงส่งสูงๆ เครื่องสูบน้ำจุ่ม ใช้กับงานสูบน้ำออก เช่น งานน้ำท่วม บ่อน้ำพุ มีกำลังส่งต่ำ แต่สูบน้ำได้ปริมาณมากๆ

เครื่องสูบน้ำเป็นอุปกรณ์สำหรับเพิ่มแรงดันของน้ำ ซึ่งมีทั้งแบบที่ใช้ มอเตอร์(ไฟฟ้า) และแบบที่ใช้ เครื่องยนต์(น้ำมัน) ทำหน้าที่หมุนส่งกำลังให้เครื่องสูบน้ำทำงาน เพื่อเพิ่มแรงดัน และส่งน้ำไปตามท่อ

เครื่องสูบน้ำที่ใช้ในบ้านส่วนใหญ่ จะเป็นแบบไฟฟ้า ซึ่งแบ่งออกเป็น 2 กลุ่มใหญ่ๆ

1. เครื่องสูบน้ำแบบใบพัด ทำงานด้วยการหมุนของใบพัด ทำให้เกิดแรงดันจ่ายไปตามท่อน้ำ ข้อดี คือ ขนาดเล็ก หลักการทำงานง่าย ชิ้นส่วนไม่มาก จ่ายน้ำได้ในปริมาณมาก สร้างแรงดันน้ำได้มากพอควร ถ้าหากต้องการแรงดันสูงสามารถนำเครื่องสูบน้ำมาต่อกันแบบมัลติสเตทได้ ปัจจุบันนิยมใช้กันมาก เครื่องสูบน้ำแบบใบพัด มีชื่อเรียก แตกต่างกันออกไปตามรูปร่างลักษณะของเครื่องสูบน้ำ เช่น เครื่องสูบน้ำอัตโนมัติ เครื่องสูบน้ำหอยโข่ง เครื่องสูบน้ำไดโว่

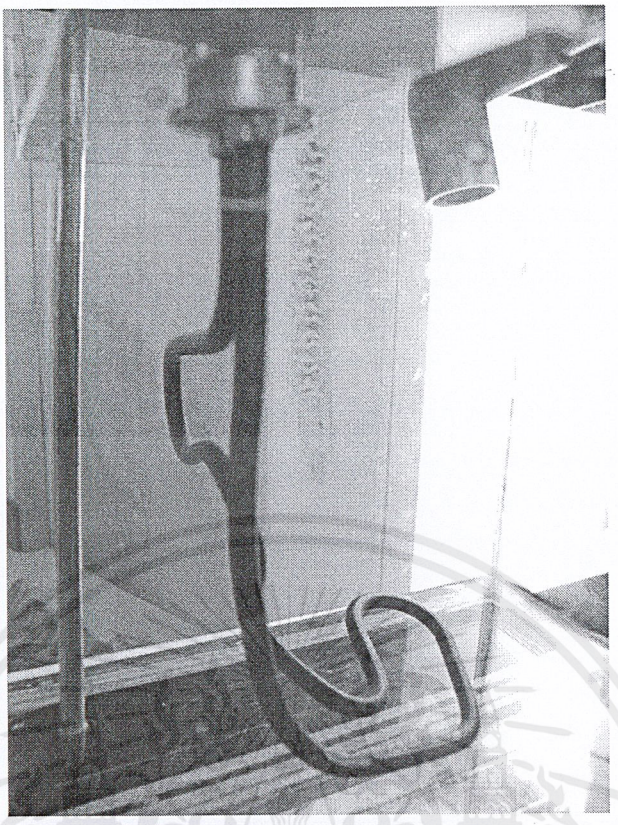
2. เครื่องสูบน้ำแบบลูกสูบ ทำงานด้วย การชักลูกสูบ เลื่อนไป-มา และมีวาล์วเปิด-ปิดน้ำเข้าออก จากลูกสูบ เป็นการเพิ่มแรงดันน้ำโดยตรง สมัยก่อนนิยมใช้กันมาก (โดยเฉพาะในสวน)

ปัจจุบันไม่ค่อยนิยมใช้กันแล้ว ข้อดี คือ สามารถสร้างแรงดันน้ำได้สูง แต่มีข้อเสีย คือ ให้ปริมาณน้ำน้อย และมีการสึกหรอ ของลูกสูบมาก

รูปที่ 2.36 แสดงชุด เครื่องสูบน้ำที่ใช้ในระบบ

2.10 เครื่องกำเนิดคลื่นความร้อนอินฟาเรด

ลักษณะการทำงานของ อุปกรณ์ให้ความร้อนแบบอินฟาเรดเป็นการส่งผ่านความร้อนแบบแผ่รังสี จึงมี ประสิทธิภาพสูง ความสูญเสียต่ำ ประหยัดไฟได้ถึง 30-50 % สามารถให้ความร้อนวัตถุได้ถึงเนื้อใน จึงทำให้ประหยัดเวลาได้ถึง 1-10เท่า มีขนาดเล็กกว่า Heater ทั่วๆไป ทำให้ประหยัดเนื้อที่การติดตั้งและการถอดเปลี่ยนเพื่อซ่อมบำรุงง่าย มีความปลอดภัยสูง เนื่องจากไม่มีเปลวไฟ ตัวเรือนมีความเป็นฉนวนสูง ไฟไม่รั่วให้รังสี 3-10 μM ซึ่งเป็นช่วงนี้วัสดุทุกชนิดสามารถดูดซับรังสีนี้ได้ ซึ่งใช้ง่าย เพียงแค่จ่ายไฟฟ้ากระแสสลับ 220โวลต์โดยตรง

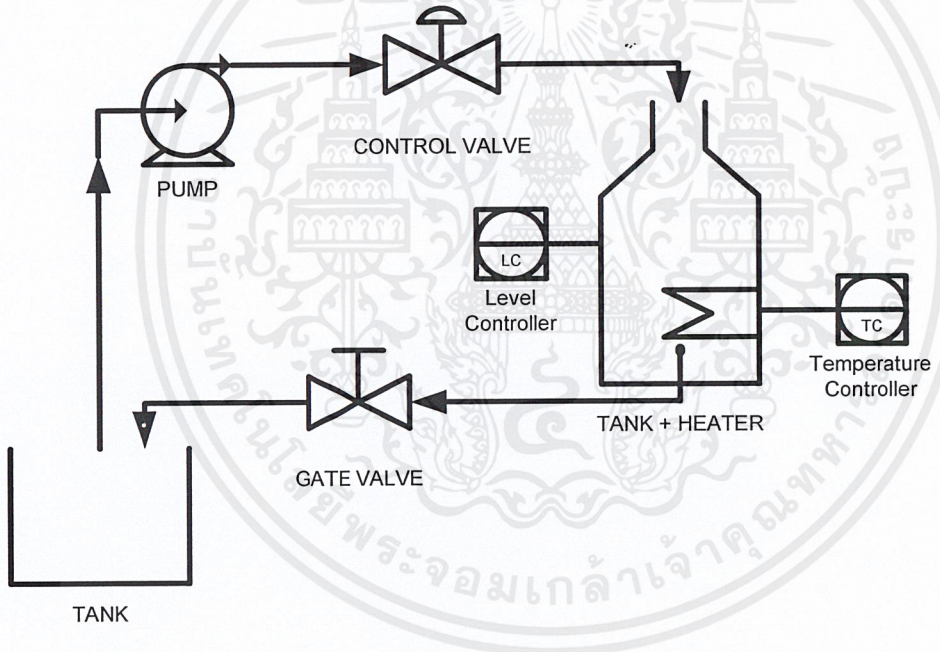


รูปที่ 2.37 เครื่องกำเนิดคลื่นความร้อนอินฟาเรด

บทที่ 3
การออกแบบวงจรและโครงสร้าง

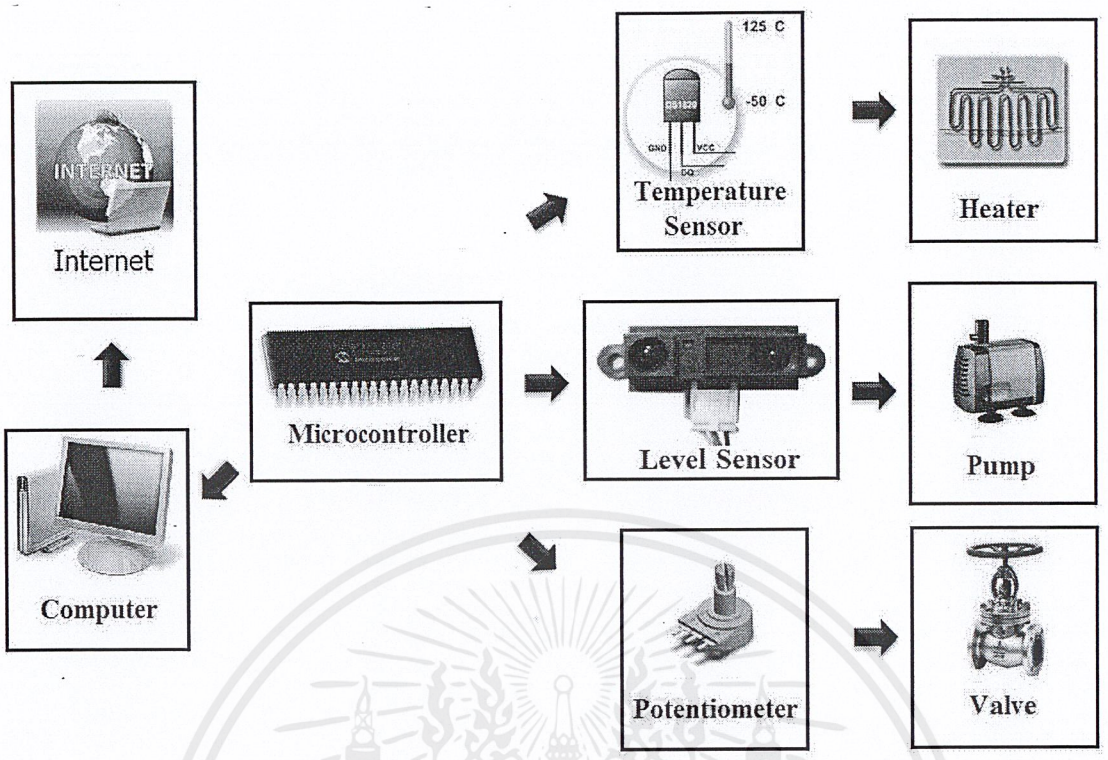
โครงสร้างและส่วนประกอบต่างๆภายในชุดทดลองระบบควบคุมกระบวนการนั้น ประกอบด้วยอุปกรณ์หลากหลายชนิด ซึ่งแบ่งแยกทั้งหมดออกจากกัน โดยแยกกันเป็นส่วนๆ ทำตามหน้าที่ของแต่ละกระบวนการ โดยในแต่ละกระบวนการนั้น การออกแบบจะขึ้นอยู่กับชนิดและขนาดอุปกรณ์และลักษณะการใช้งานของอุปกรณ์ชนิดนั้นๆด้วย

ในช่วงแรกของการออกแบบชุดทดลอง ได้เริ่มจากการสร้างแบบจำลองระบบควบคุมกระบวนการไว้ ดังรูปที่ 3.1



รูปที่ 3.1 แบบจำลองระบบควบคุมกระบวนการ

เมื่อได้ทำการสร้างแบบจำลองแล้ว จึงได้เริ่มออกแบบการทำงานระหว่างเครื่องมือวัดภายในระบบควบคุมกระบวนการกับเครื่องมือควบคุมและแสดงผล ดังรูปที่ 3.2



รูปที่ 3.2 การทำงานภายในระบบควบคุมกระบวนการ

ในชุดทดลองระบบควบคุมกระบวนการนั้น จะประกอบไปด้วยโครงสร้าง อุปกรณ์ต่างๆ วงจรอิเล็กทรอนิกส์ และโปรแกรมควบคุมกระบวนการ ดังนี้

3.1 ตู้บรรจุชุดทดลองระบบควบคุมกระบวนการ

3.2 ระบบทางเดินของน้ำภายในระบบควบคุมกระบวนการ

3.3 ชุดวาล์วควบคุมแบบอัตโนมัติ

3.4 ถังเก็บน้ำและอุปกรณ์ในกระบวนการ

3.5 วงจรไมโครคอนโทรลเลอร์

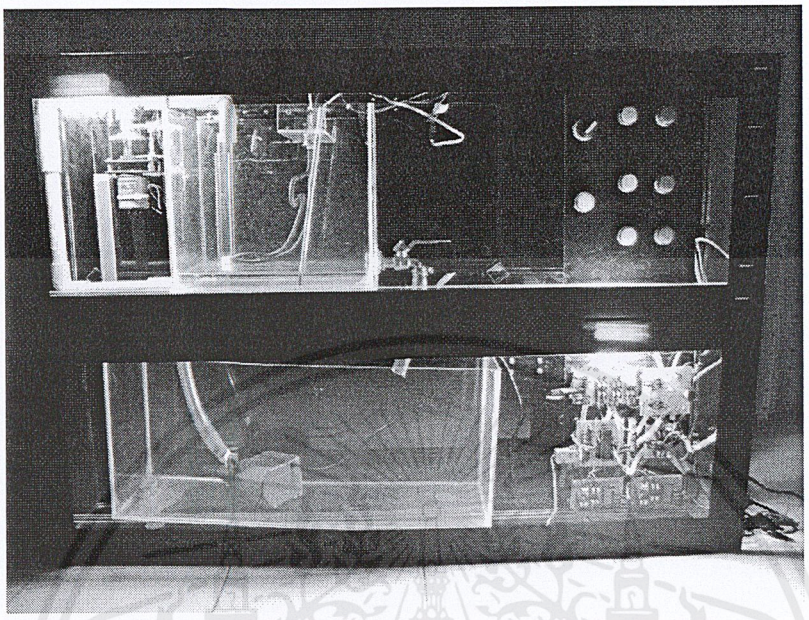
3.6 วงจรควบคุมมอเตอร์ไฟฟ้ากระแสตรง

3.7 วงจรควบคุมการทำงานแบบเปิด –ปิด

3.1 ตู้บรรจุชุดทดลองระบบควบคุมกระบวนการ

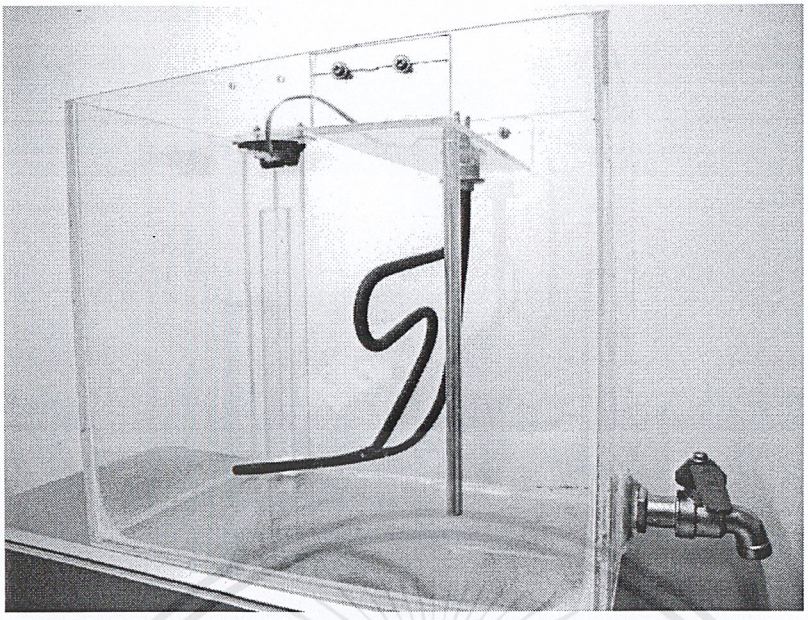
เป็นตู้อะคลีลิกขนาด กว้าง 76 เซนติเมตร ยาว 110 เซนติเมตร และ หนา 20 เซนติเมตร โดยประกอบขึ้นจากแผ่นอะคลีลิกขนาด หนา 5 มิลลิเมตร เคลือบภายนอกด้วยพลาสติกสีดำทึบ ประกอบเป็นตู้ โดยมี 2ชั้น ความสูงของแต่ละชั้นอยู่ที่ชั้นละ 38 เซนติเมตร เจาะรูเชื่อมถึงกัน ระหว่าง 2 ชั้น สามารถเปิดได้ ทางด้านซ้ายของตู้ โดยเลื่อนออก และ ด้านหลังของตู้ทำเป็นประตู

บานพับ สามารถเปิดเพื่อจัดการติดตั้งอุปกรณ์ภายในได้ มีการติดตั้งหลอดไฟเพื่อให้ความสว่างในการมองเห็นที่ชัดเจนยิ่งขึ้นทั้งหมดจำนวน 2 หลอด



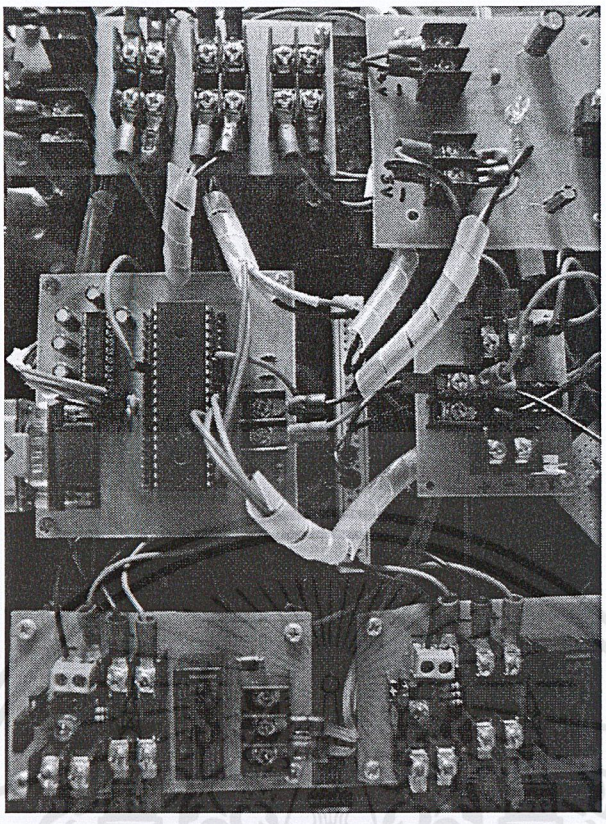
รูปที่ 3.3 กล่องบรรจุชุดชุดทดลองระบบควบคุมกระบวนการ

ในชุดทดลองระบบควบคุมกระบวนการนี้ ยังประกอบด้วยถังเก็บน้ำขนาดใหญ่อีก 2 ถัง ซึ่งต้องถูกบรรจุไว้ภายในตู้อะคลีลิกนี้ โดยการแยกถังเก็บน้ำให้อยู่ในแต่ละชั้น การติดตั้งจะเป็นแบบการวางลงไปโดยมีลูกยางเป็นตัวรองรับน้ำหนักของถังน้ำเอาไว้



รูปที่ 3.4 การติดตั้งถังเก็บน้ำภายในระบบ

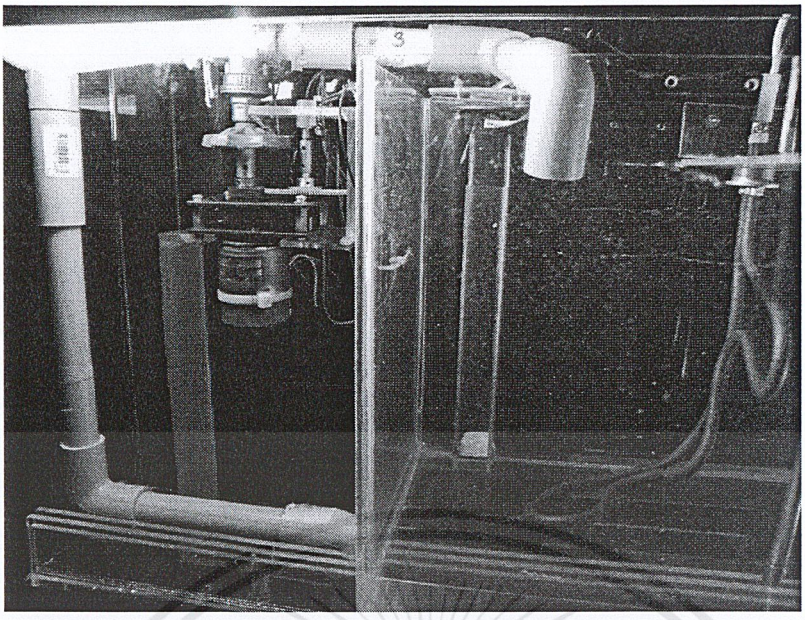
นอกเหนือจากอุปกรณ์ต่างๆที่ถูกติดตั้งในตู้บรรจุชุดทดลองระบบควบคุมกระบวนการนี้แล้ว ภายในยังประกอบด้วยอุปกรณ์อิเล็กทรอนิกส์ต่างๆ ที่หลากหลาย รวมทั้งยังมีการเชื่อมโยงระหว่างอุปกรณ์ต่างๆ กับวงจรอิเล็กทรอนิกส์ด้วยการเดินสายไฟแบบฝังติดกับตัวตู้ สำหรับวงจรอิเล็กทรอนิกส์จะแบ่งออกเป็น 2 ส่วน ด้วยกันคือ ในส่วนของแผงวงจรควบคุม และส่วนของไฟแสดงสถานะการทำงานของอุปกรณ์ โดยจะติดตั้งอยู่บนแผ่นอะคลีลิกแยกออกจากกันเป็นสองส่วน ซึ่งภายในตู้จะมีรางเลื่อนสำหรับใส่แผ่นวงจร ทำให้สะดวกต่อการติดตั้ง



รูปที่ 3.5 การติดตั้งแผงวงจรและการเดินสายไฟ

3.2 ระบบทางเดินของน้ำภายในระบบควบคุมกระบวนการ

ชุดทดลองระบบควบคุมกระบวนการนี้เป็นการทดลองโดยอาศัยการไหลของของไหลเป็นตัวกลางในการทดลอง โดยการเชื่อมโยงกันของระบบน้ำภายในระบบทั้งหมดนั้น อาศัยระบบท่อน้ำและสายยางในการลำเลียงของไหล โดยจะเริ่มจากบริเวณด้านล่างซึ่งจะมีสายยางต่อเข้ากับเครื่องสูบน้ำซึ่งอยู่ภายในถังเก็บน้ำชั้นล่างของตู้ จากนั้นของไหลจะถูกลำเลียงผ่านขึ้นมาเข้าสู่ท่อน้ำ ผ่านเข้าสู่วาล์วน้ำควบคุมแบบอัตโนมัติและจะผ่านเข้าสู่ถังเก็บน้ำซึ่งอยู่ในชั้นบน และเมื่อกระบวนการทดลองภายในถังเก็บน้ำชั้นบนเสร็จสิ้น น้ำก็จะไหลจากถังด้านบนลงไปด้านล่างโดยผ่านวาล์วน้ำแบบมือหมุน ซึ่งต่อกับสายยางลงสู่ด้านล่าง

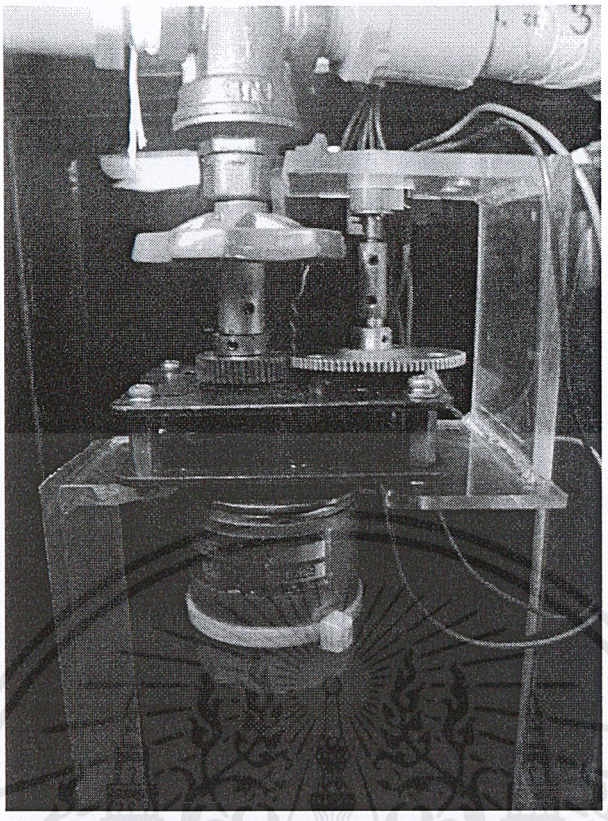


รูปที่ 3.6 ระบบทางเดินของไหลภายในระบบควบคุมกระบวนการ

3.3 ชุดวาล์วควบคุมแบบอัตโนมัติ

ชุดวาล์วควบคุมแบบอัตโนมัติประกอบด้วย มอเตอร์ไฟฟ้ากระแสตรงที่ประกอบเข้ากันกับชุดขบวนเฟืองแบบ Compound gear train โดยประกอบด้วยเฟืองชิ้นย่อยๆทั้งหมดจำนวน 4 ชิ้น เข้าด้วยกัน โดยเฟืองชิ้นที่3 นำไปประกอบเข้ากับวาล์วแบบประตู ซึ่งเฟืองชิ้นที่3นี้จะไปขับกับเฟืองชิ้นที่ 4 ซึ่งประกอบเข้ากันกับโพเทนชิโอมิเตอร์ โดยเมื่อมีการขับเคลื่อนมอเตอร์ไฟฟ้ากระแสตรง ฟันเฟืองจะเกิดการหมุนโดยจะทำให้วาล์วประตูหมุนไปพร้อมๆ กับโพเทนชิโอมิเตอร์

โพเทนชิโอมิเตอร์ ที่ถูกติดตั้งอยู่นั้นจะทำหน้าที่เป็นตัวระบุตำแหน่งของวาล์วประตู ณ ปัจจุบัน โดยใช้หลักการนำค่าความต่างศักย์ปัจจุบันที่โพเทนชิโอมิเตอร์ ไปเปรียบเทียบกับแรงดันและอัตราการเปิด-ปิดของวาล์ว ซึ่งจะทำให้สามารถรู้ได้ว่าวาล์วอยู่ตำแหน่งใดในขณะนั้น



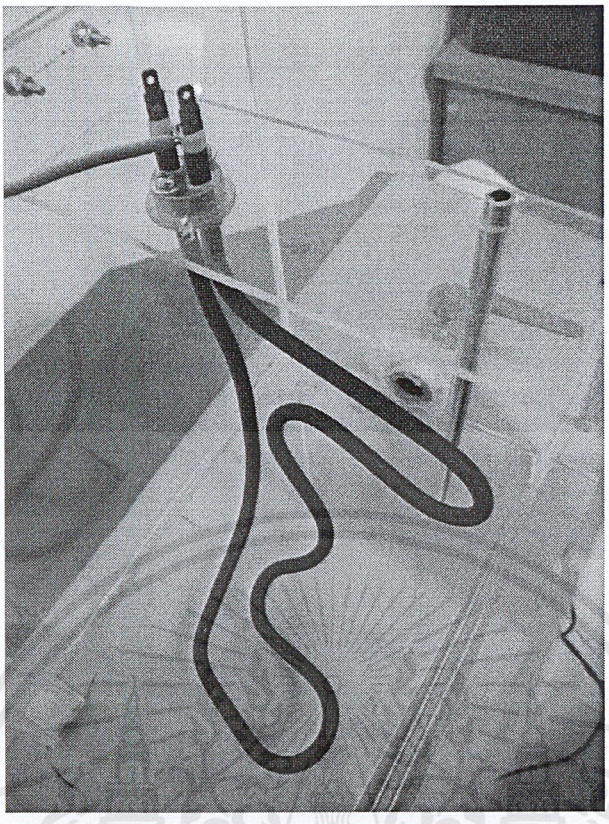
รูปที่ 3.7 ชุดวาล์วควบคุมอัตโนมัติ

3.4 ถังเก็บน้ำและอุปกรณ์ในกระบวนการ

ถังเก็บน้ำจะอยู่บริเวณชั้นบนและชั้นล่างของตู้ ซึ่งถังเก็บน้ำนี้ประกอบด้วยอุปกรณ์ที่ใช้ในกระบวนการต่างๆในระบบ ซึ่งน้ำจะไหลจากถังเก็บน้ำในชั้นล่างเข้าสู่วาล์วควบคุมแบบอัตโนมัติ และเข้ามาสู่ถังเก็บน้ำในชั้นบน ซึ่งจะประกอบด้วยกระบวนการต่างๆดังนี้

3.4.1 ชุดการให้ความร้อนและวัดอุณหภูมิ

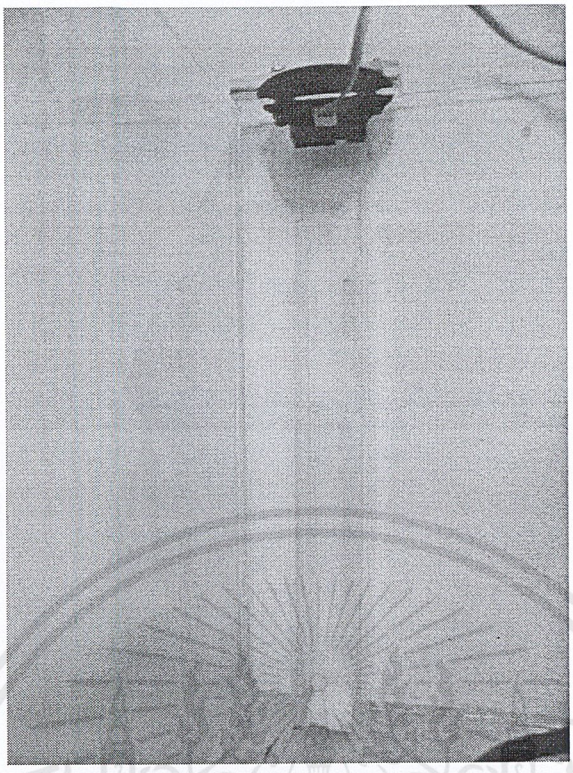
ภายในถังเก็บน้ำประกอบด้วยกระบวนการต่างๆ ซึ่งกระบวนให้ความร้อนและการวัดค่าความร้อนที่เปลี่ยนแปลงไป ก็เป็นส่วนหนึ่งในกระบวนการของถังนี้ อยู่ที่ชั้นบนของตู้ โดยชุดให้ความร้อนจะถูกติดตั้งอยู่ตรงกลางของถังเก็บน้ำ และเซนเซอร์ในการวัดอุณหภูมิก็ติดอยู่บริเวณใกล้เคียง โดยจะเชื่อมต่อกับอุปกรณ์อิเล็กทรอนิกส์โดยการเดินสายไฟเชื่อมต่อด้านบนของตู้



รูปที่ 3.8 ชุดอุปกรณ์ให้ความร้อนและวัดค่าอุณหภูมิ

3.4.2 ชุดอุปกรณ์วัดระดับของเหลว

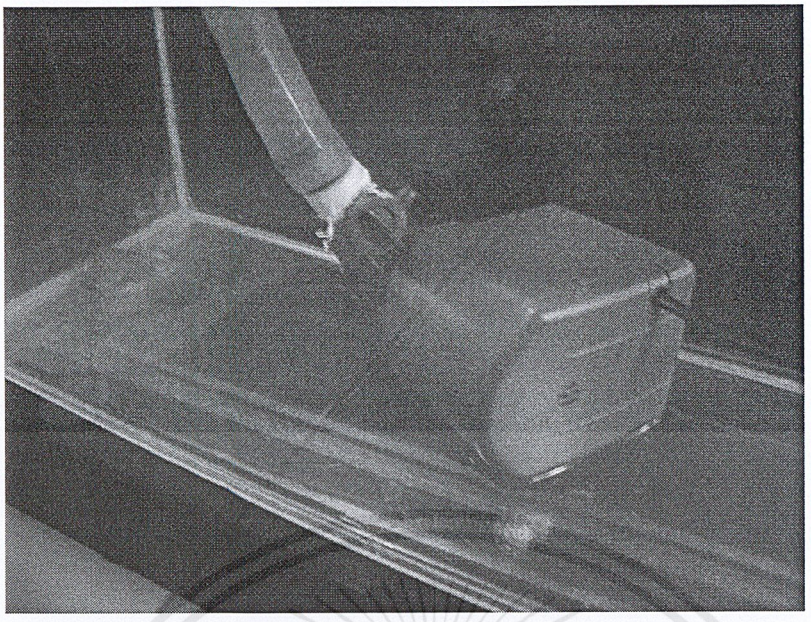
อีกกระบวนการหนึ่งภายในถังเก็บน้ำในด้านบน คือกระบวนการวัดระดับของเหลวโดยจะใช้เซนเซอร์อินฟาเรด ซึ่งเซนเซอร์จะถูกยึดติดกับกระบอกอะคลีลิกซึ่งติดตั้งอยู่บริเวณผนังของถังเก็บน้ำโดยติดให้สูงกว่าพื้นเพียงเล็กน้อย ภายในกระบอกอะคลีลิกบรรจุโฟมขนาดพอดีกับกระบอก ซึ่งเมื่อมีน้ำเข้ามาในถังเก็บน้ำ ความสูงของน้ำในถังจะเท่ากับความสูงของน้ำในกระบอก ทำให้โฟมเคลื่อนที่ขึ้นตามระดับ จากนั้นเซอเซอร์ก็จะอ่านค่าจากระดับความสูงที่เพิ่มขึ้นของโฟม



รูปที่ 3.9 ชุดอุปกรณ์วัดระดับของเหลว

3.4.3 ชุดเครื่องสูบน้ำ

ชุดเครื่องสูบน้ำติดตั้งอยู่ภายในถังเก็บน้ำที่อยู่ภายในชั้นล่างของระบบ โดยจะติดตั้งเครื่องสูบน้ำเพื่อเป็นอุปกรณ์ในการเริ่มต้นกระบวนการทั้งหมด และเมื่อกระบวนการทุกอย่างสิ้นสุด น้ำทั้งหมดก็จะกลับมาสู่ถังเก็บน้ำทางด้านล่าง



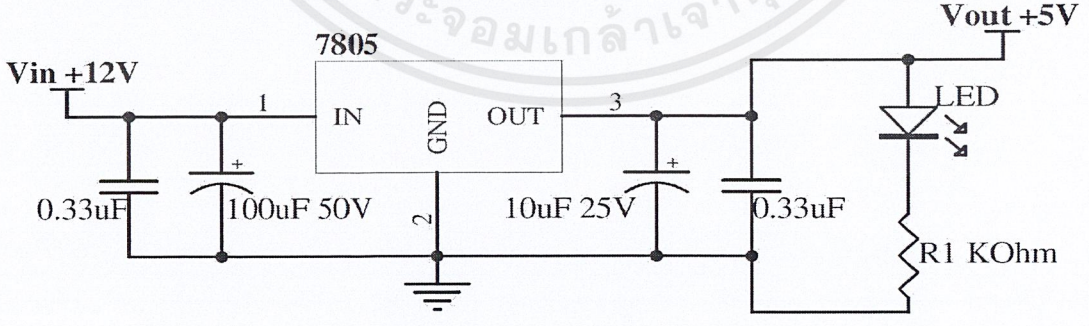
รูปที่ 3.10 การติดตั้งเครื่องสูบน้ำ

3.5 วงจรไมโครคอนโทรลเลอร์

ส่วนของวงจรไมโครคอนโทรลเลอร์ จะประกอบไปด้วย วงจรแปลงแรงดันไฟฟ้ากระแสตรง 5 โวลต์ วงจรหน่วยประมวลผลส่วนกลาง และวงจรเชื่อมต่อผ่านพอร์ตอนุกรม

3.5.1 วงจรแปลงแรงดันไฟฟ้ากระแสตรง 5 โวลต์

เป็นส่วนที่แปลงแรงดันไฟฟ้ากระแสตรงจาก 12 โวลต์ เป็น แรงดันไฟฟ้ากระแสตรง 5 โวลต์ โดยเลือกใช้ ไอซีเร็กกูเลเตอร์สามขา ชนิดจ่ายแรงดันคงที่ เบอร์ 7805



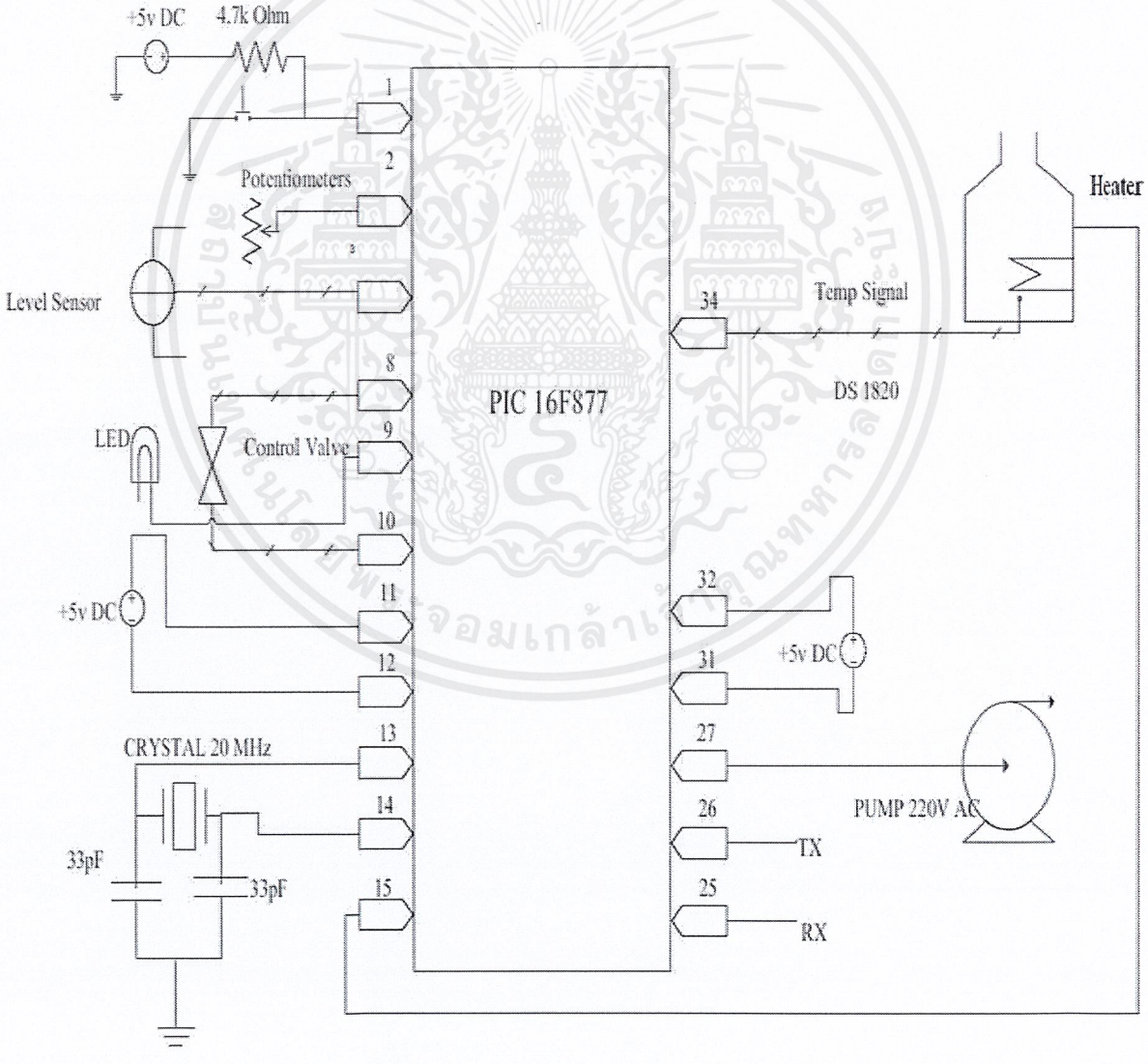
รูปที่ 3.11 วงจรแปลงแรงดันไฟฟ้ากระแสตรง

7805 เป็นไอซีเร็กกูเลเตอร์ ที่แปลงแรงดันไฟฟ้ากระแสตรง อินพุตที่ 10 ถึง 35 โวลต์ และเอาท์พุตที่ 5 โวลต์ มีขา 3 ขา คือ อินพุต กราวน์ และเอาท์พุต ในการต่อวงจร เราจะต่อแหล่งจ่าย

แรงดันไฟฟ้า 12 โวลต์ที่ขาอินพุต และใส่ตัวเก็บประจุที่มีค่าประมาณ 100 ไมโครฟารัดไว้ที่ขาอินพุตด้วย เพื่อป้องกันการเกิดออสซิลเลตที่ความถี่สูง ซึ่งจะทำให้วงจรขาดเสถียรภาพ และที่ขาเอาท์พุตจะใส่ตัวเก็บประจุที่มีค่าประมาณ 10 ไมโครฟารัด เพื่อช่วยปรับปรุงแรงดันเอาท์พุตให้เรียบยิ่งขึ้น เมื่อต่อวงจรดังกล่าว จะได้แรงดันไฟฟ้ากระแสตรง 5 โวลต์ จากนั้นจึงนำแรงดันไฟฟ้ากระแสตรง 5 โวลต์ไปใช้งานในวงจรต่างๆ ต่อไป

3.5.2 วงจรหน่วยประมวลผลส่วนกลาง

เป็นส่วนหลักของวงจรไมโครคอนโทรลเลอร์ ซึ่งเลือกใช้ ไมโครคอนโทรลเลอร์เบอร์ PIC16F877 เป็นหน่วยประมวลผลส่วนกลาง ทำหน้าที่รับค่าจากเครื่องมือวัดในระบบควบคุมและส่งค่านั้นไปยังคอมพิวเตอร์ เพื่อแสดงผลและประมวลผลต่อไป



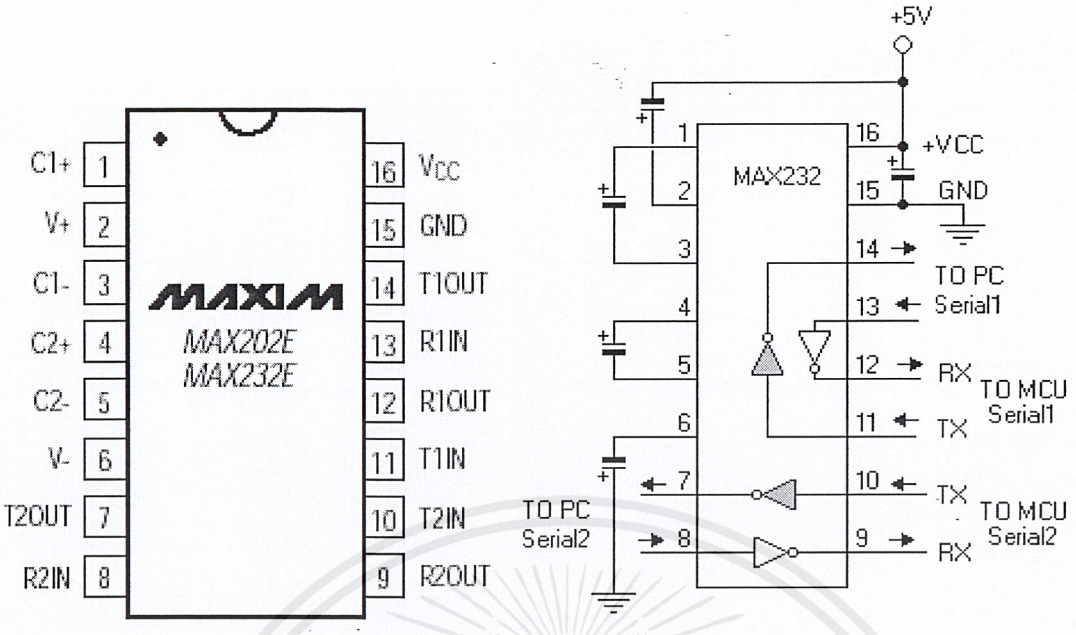
รูปที่ 3.12 วงจรหน่วยประมวลผลส่วนกลาง

ตารางที่ 3.1 แสดงหน้าที่การทำงานของขาไมโครคอนโทรลเลอร์

PIN	หน้าที่การทำงาน
2	รับสัญญาณจาก ตัวต้านทานปรับค่าได้ ซึ่งทำหน้าที่เป็นตัววัดระยะการหมุนของมอเตอร์ไฟฟ้ากระแสตรง
3	รับสัญญาณจาก อุปกรณ์ตรวจวัดระยะทาง เป็นอุปกรณ์ที่ใช้วัดระดับน้ำในถัง
8	ส่งสัญญาณควบคุมการหมุนมอเตอร์ไฟฟ้ากระแสตรง เพื่อเปิด-ปิด วาล์ว
9	ส่งสัญญาณควบคุมการหมุนมอเตอร์ไฟฟ้ากระแสตรง เพื่อเปิด-ปิด วาล์ว
15	ส่งสัญญาณควบคุมการเปิด-ปิด เครื่องทำความร้อน
25	Tx ส่งสัญญาณไปยังพอร์ตอนุกรม
26	Rx รับสัญญาณจากพอร์ตอนุกรม
27	ส่งสัญญาณควบคุมการเปิด-ปิด เครื่องสูบน้ำ
34	รับสัญญาณจาก ตัววัดอุณหภูมิของน้ำในถัง

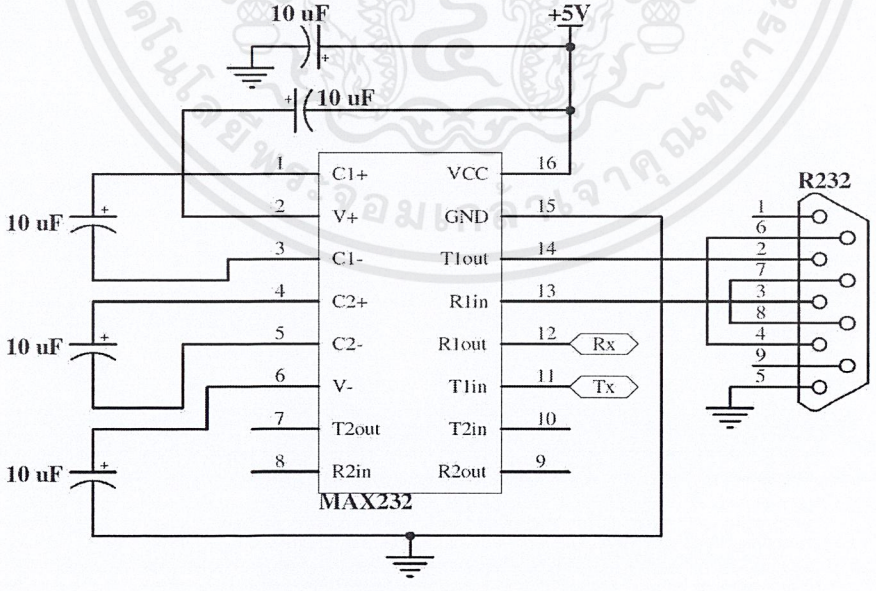
3.5.3 วงจรเชื่อมต่อผ่านพอร์ตอนุกรม

ในการเชื่อมต่อจากไมโครคอนโทรลเลอร์ไปยังคอมพิวเตอร์ เนื่องจากพอร์ตอนุกรมของคอมพิวเตอร์ เป็นมาตรฐาน RS232 แต่ไมโครคอนโทรลเลอร์เป็น TTL (Transistor-Transistor Logic) จึงต้องใช้ไอซี MAX232 ปรับระดับแรงดันให้อยู่ในระดับเดียวกัน



รูปที่ 3.13 แสดงขาและวงจรภายในของ MAX232

ไอซี MAX232 เป็นไอซี 16 ขา ใช้แหล่งจ่ายไฟฟ้ากระแสตรง 5โวลต์ เป็นไฟเลี้ยง ต่อเข้าที่ขา16 ภายในมีวงจรแปลงระดับสัญญาณของ RS232 มาเป็นระดับ TTL 2 ชุด และวงจรแปลงระดับสัญญาณจาก TTL มาเป็นระดับสัญญาณ RS232 2 ชุด



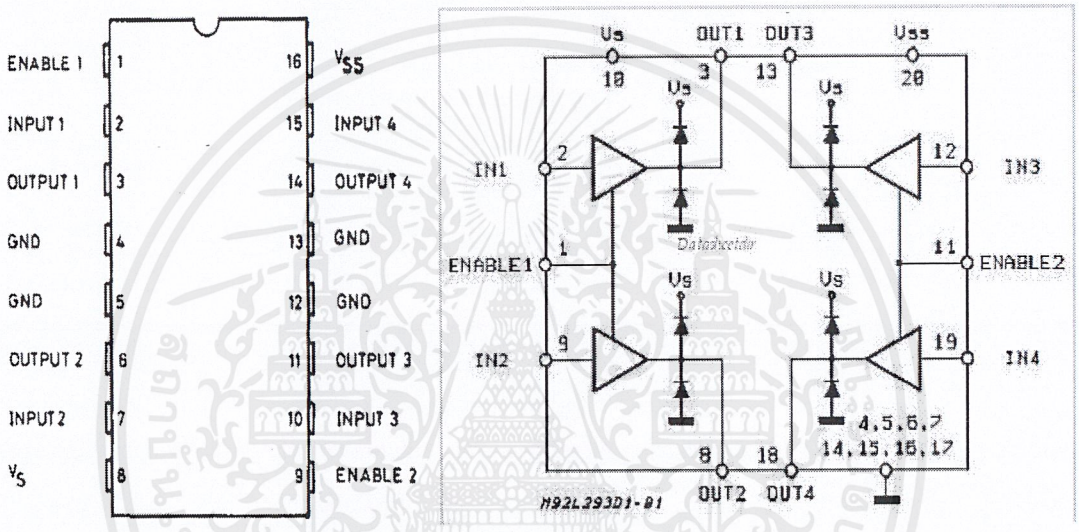
รูปที่ 3.14 วงจรเชื่อมต่อผ่านพอร์ตอนุกรม

จากรูปที่ 3.12 แสดงการต่อวงจรไอซี MAX232 กับพอร์ตอนุกรม RS232 และ Tx (ขา25) Rx (ขา26) จากไมโครคอนโทรลเลอร์ โดยต่อ Tx เข้าที่ T1in (ขา11) และ Rx ที่ R1out (ขา12)

จากนั้นที่ R1in (ขา13) ต่อกับขา 3 ของ RS232 และ T1out (ขา14) ต่อกับขา 2 และขา 5 ของ RS232 ให้ต่อลงกราวน์

3.6 วงจรควบคุมมอเตอร์ไฟฟ้ากระแสตรง

ในการควบคุมมอเตอร์ไฟฟ้ากระแสตรง ต้องอาศัยการต่อวงจรแบบ H-bridge ซึ่งจะสามารถควบคุมทิศทางการหมุนของมอเตอร์ได้ โดยเลือกใช้ไอซี L293D ที่มีวงจรแบบ H-bridge อยู่ภายใน



รูปที่ 3.15 แสดงขาและวงจรภายในของ L293D

ตารางที่ 3.2 แสดงหน้าที่ของขาต่างๆ ของ L239D

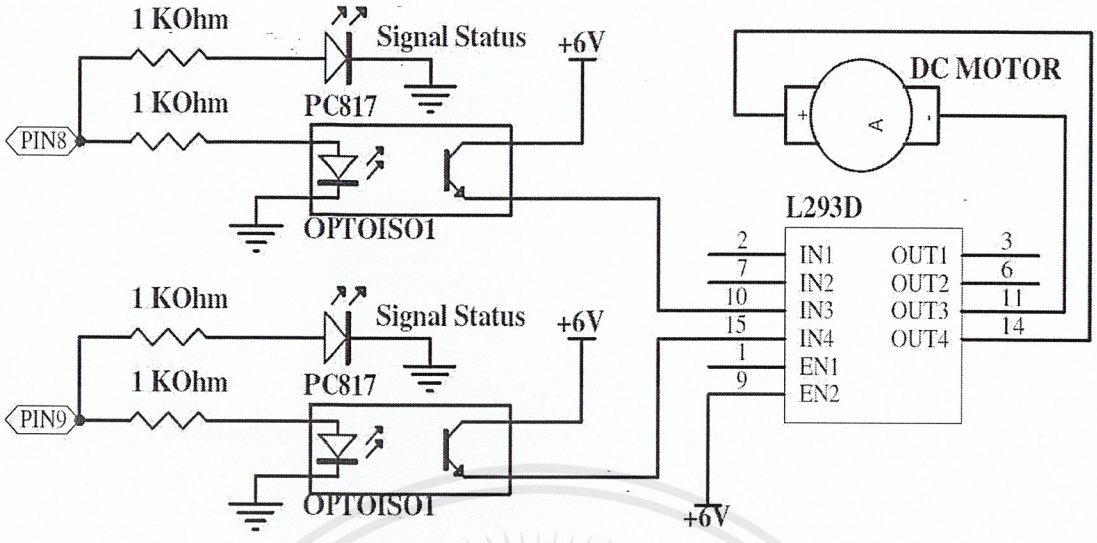
PIN	NAME	FUNCTION
1	ENABLE 1	ขาเปิดฟังก์ชั่นการทำงาน 1
2	INPUT 1	ขารับสัญญาณ 1
3	OUTPUT 1	ขาเอาท์พุตที่ใช้ขับมอเตอร์ 1
4 , 5 , 12 , 13	GND	ขากราวน์
6	OUTPUT 2	ขาเอาท์พุตที่ใช้ขับมอเตอร์ 2
7	INPUT 2	ขารับสัญญาณ 2
8	Vs	แหล่งจ่ายไฟ
9	ENABLE 2	ขาเปิดฟังก์ชั่นการทำงาน 2
10	INPUT 3	ขารับสัญญาณ 3
11	OUTPUT 3	ขาเอาท์พุตที่ใช้ขับมอเตอร์ 3
14	OUTPUT 4	ขาเอาท์พุตที่ใช้ขับมอเตอร์ 4
15	INPUT 4	ขารับสัญญาณ 4
16	Vss	แหล่งจ่ายไฟ

จากรูปที่ 3.13 เป็นการแสดงหน้าที่ของขาต่างๆ ของไอซี L283D โดยโครงงานนี้ใช้มอเตอร์ไฟฟ้ากระแสตรง 1 ตัว จึงเลือกใช้งานเพียงอินพุต 3 และ อินพุต 4 เป็นตัวกำหนดทิศทางหมุนของมอเตอร์ และขา ENABLE 2 เป็นตัวควบคุมการทำงานของมอเตอร์

ตารางที่ 3.3 คำสั่งจากไมโครคอนโทรลเลอร์ และการทำงานของมอเตอร์

INPUT 3	INPUT 4	Description
0	0	Stop
1	0	Forward
0	1	Reverse
1	1	Brake

ตารางที่ 3.3 เป็นการบอกลักษณะการทำงานของมอเตอร์ ที่คำสั่งต่างๆ ซึ่งมาจากไมโครคอนโทรลเลอร์

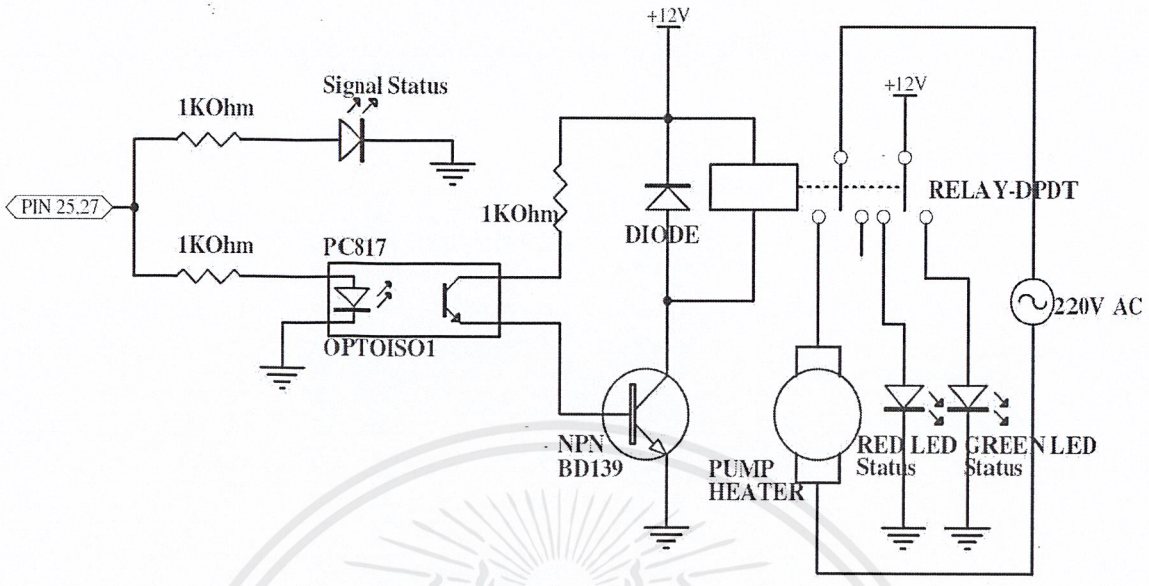


รูปที่ 3.16 วงจรควบคุมมอเตอร์ไฟฟ้ากระแสตรง

วงจรควบคุมมอเตอร์ไฟฟ้ากระแสตรง จะแบ่งเป็นส่วนของการรับสัญญาณจากไมโครคอนโทรลเลอร์ และส่วนของการขับเคลื่อนมอเตอร์ โดยในส่วนของการรับสัญญาณนั้น จะรับสัญญาณมาจาก PIN8 และ PIN9 ของไมโครคอนโทรลเลอร์ มาต่อที่โฟโตทรานซิสเตอร์สองตัว ทำหน้าที่แบ่งแยกสัญญาณระหว่างวงจรไมโครคอนโทรลเลอร์กับมอเตอร์ ซึ่งใช้แหล่งจ่ายไฟฟ้ากระแสตรง 6 โวลต์ และส่วนของการขับมอเตอร์นั้น เมื่อมีการรับสัญญาณจาก PIN8 โฟโตไดโอดภายในออปโตจะทำงาน และโฟโตทรานซิสเตอร์ภายในออปโตจะยอมให้กระแสไฟฟ้าไหลจากขาคอลเล็กเตอร์ไปขาอีมิเตอร์ และไปยังขา IN3 (ขา10) ของ L293D และเนื่องจาก PIN9 ไม่ได้ส่งสัญญาณมา จึงทำให้ไม่มีกระแสไหลเข้าที่ IN4 (ขา15) ของ L293D ทำให้ลอจิกที่ขา IN3 และ IN4 เป็น 1 กับ 0 ตามลำดับ มอเตอร์จึงหมุน และตรงกันข้ามเมื่อ PIN9 ส่งสัญญาณมา และ PIN8 ไม่ได้ส่งสัญญาณ จะทำให้ลอจิกที่ขา IN3 และ IN4 เป็น 0 กับ 1 ตามลำดับ มอเตอร์จึงหมุนกลับทาง

3.7 วงจรควบคุมการทำงานแบบเปิด -ปิด

ในโครงงานนี้ จำเป็นต้องมีการควบคุมการทำงานของเครื่องสูบน้ำ และเครื่องทำความร้อน โดยการควบคุมเป็นแบบเปิด-ปิด จึงเลือกใช้รีเลย์มาเป็นสวิตช์ในการเปิด-ปิด โดยรีเลย์มีขดลวดเหนี่ยวนำที่ 12โวลต์ และมีหน้าสัมผัส 2 ชุด ที่รับกระแสไฟฟ้าสูงสุดได้ 5 แอมแปร์ แรงดันไฟฟ้ากระแสตรงที่ 24 โวลต์ และแรงดันไฟฟ้ากระแสสลับที่ 250 โวลต์ โดยส่วนของวงจรควบคุมการเปิด-ปิด จะรับคำสั่งมาจากไมโครคอนโทรลเลอร์ และในวงจรยังมีโฟโตทรานซิสเตอร์ ที่ทำหน้าที่แบ่งสัญญาณที่มาจากไมโครคอนโทรลเลอร์กับการทำงานในส่วนของรีเลย์ ซึ่งใช้แหล่งจ่ายแรงดันไฟฟ้ากระแสตรง 12 โวลต์ ออกจากกัน

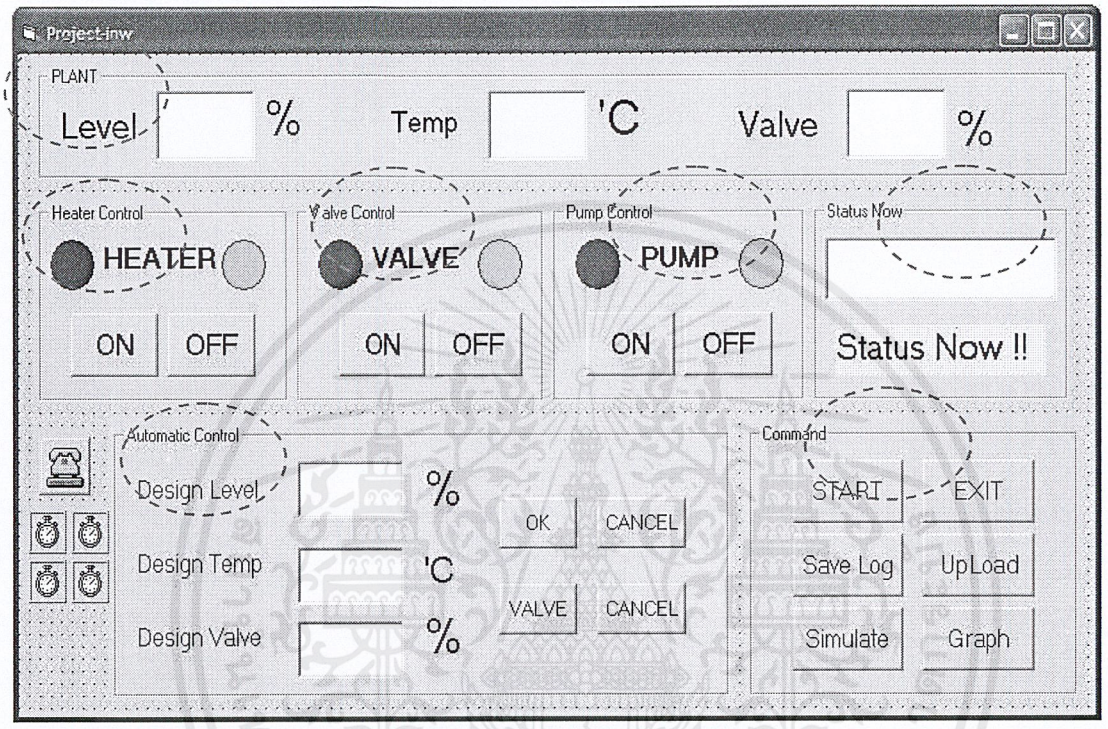


รูปที่ 3.17 วงจรควบคุมการทำงานแบบเปิด -ปิด

จากรูปที่ 3.15 เมื่อไมโครคอนโทรลเลอร์ส่งสัญญาณมาจากขา 15 ไฟแสดงสถานะ (Signal Status) จะทำงาน แจ้งให้ทราบว่ามีสัญญาณส่งมา และโฟโตไดโอดภายในออปโตจะทำงาน ทำให้โฟโตทรานซิสเตอร์ภายในออปโตทำงาน ยอมให้กระแสไหลผ่านจากขาคอลเล็กเตอร์ไปขาอีมิเตอร์ของโฟโตทรานซิสเตอร์ จากนั้นเมื่อกระแสไหลผ่านที่ทรานซิสเตอร์ตัวถัดมา (BD139) ที่ขาเบส ก็จะยอมให้กระแสไหลจากขาคอลเล็กเตอร์มาขาอีมิเตอร์ ส่งผลให้ขดลวดของรีเลย์ เกิดการเหนี่ยวนำหน้าสัมผัสให้ติดกัน และเครื่องสูบน้ำจะทำงาน

3.8 รูปแบบโปรแกรมของคอมพิวเตอร์หน่วยติดต่อและปฏิบัติการของผู้ใช้งาน

ส่วนนี้เป็นลักษณะการออกแบบโปรแกรมติดต่อของผู้ใช้งานกับตัวระบบควบคุมกระบวนการโดยผ่านโปรแกรม Visual Basic เมื่อต้องการควบคุมกระบวนการที่อยู่ในระบบส่วนแรกที่ต้องเจอเป็นรูปแบบการตั้งค่าของการสื่อสารมีลักษณะดังรูปที่ 3.18



รูปที่ 3.18 แสดงโปรแกรมแสดงผลและควบคุมชุดทดลองระบบควบคุมกระบวน

จากรูปที่ 3.18 จะแบ่งส่วนประกอบของโปรแกรมเป็นส่วนๆ คือ

- Plant คือ ส่วนที่จะแสดงผลค่าจากระบบควบคุมกระบวนการ โดยจะรับค่ามาจากไมโครคอนโทรลเลอร์ PIC16F877
- Heater Control คือ ส่วนที่ใช้สำหรับควบคุมการเปิดปิด เครื่องกำเนิดคลื่นความร้อนแบบอินฟราเรด โดยสั่งการจากคอมพิวเตอร์ไปยังไมโครคอนโทรลเลอร์ PIC16F877
- Valve Control คือ ส่วนที่ใช้สำหรับควบคุมการเปิดปิด วาล์วแบบประตู โดยสั่งการจากคอมพิวเตอร์ไปยังไมโครคอนโทรลเลอร์ PIC16F877
- Pump Control คือ ส่วนที่ใช้สำหรับควบคุมการเปิดปิด เครื่องสูบน้ำ โดยสั่งการจากคอมพิวเตอร์ไปยังไมโครคอนโทรลเลอร์ PIC16F877
- Status Now คือ ส่วนที่จะแสดงสถานะต่างๆเมื่อผู้ใช้ทำการสั่งการจากโปรแกรม

- Automatic Control คือ ส่วนที่ใช้สำหรับการควบคุมกระบวนการ ซึ่งจะสามารถกำหนดค่าในกระบวนการที่ต้องการได้ในช่อง Design โดยจะควบคุมสั่งการแบบอัตโนมัติ ในระบบควบคุมวงปิด
- Command คือ ส่วนที่จะรวมฟังก์ชั่นไว้ คือ

  Start เป็นปุ่มกดในการเริ่มต้นการทำงานของโปรแกรม

  Exit เป็นปุ่มกดในการจบการทำงานของโปรแกรม

  Save Log เป็นปุ่มกดในการบันทึกค่าต่างๆในกระบวนการลงใน Text File

  Upload เป็นปุ่มกดในการอัพโหลดค่าต่างๆในกระบวนการ ไปยังระบบอินเตอร์เน็ต

  Simulate เป็นปุ่มกดเพื่อการแสดงการจำลองภาพของระบบ

  Graph เป็นปุ่มกดเพื่อการแสดงกราฟของข้อมูลต่างๆในระบบ

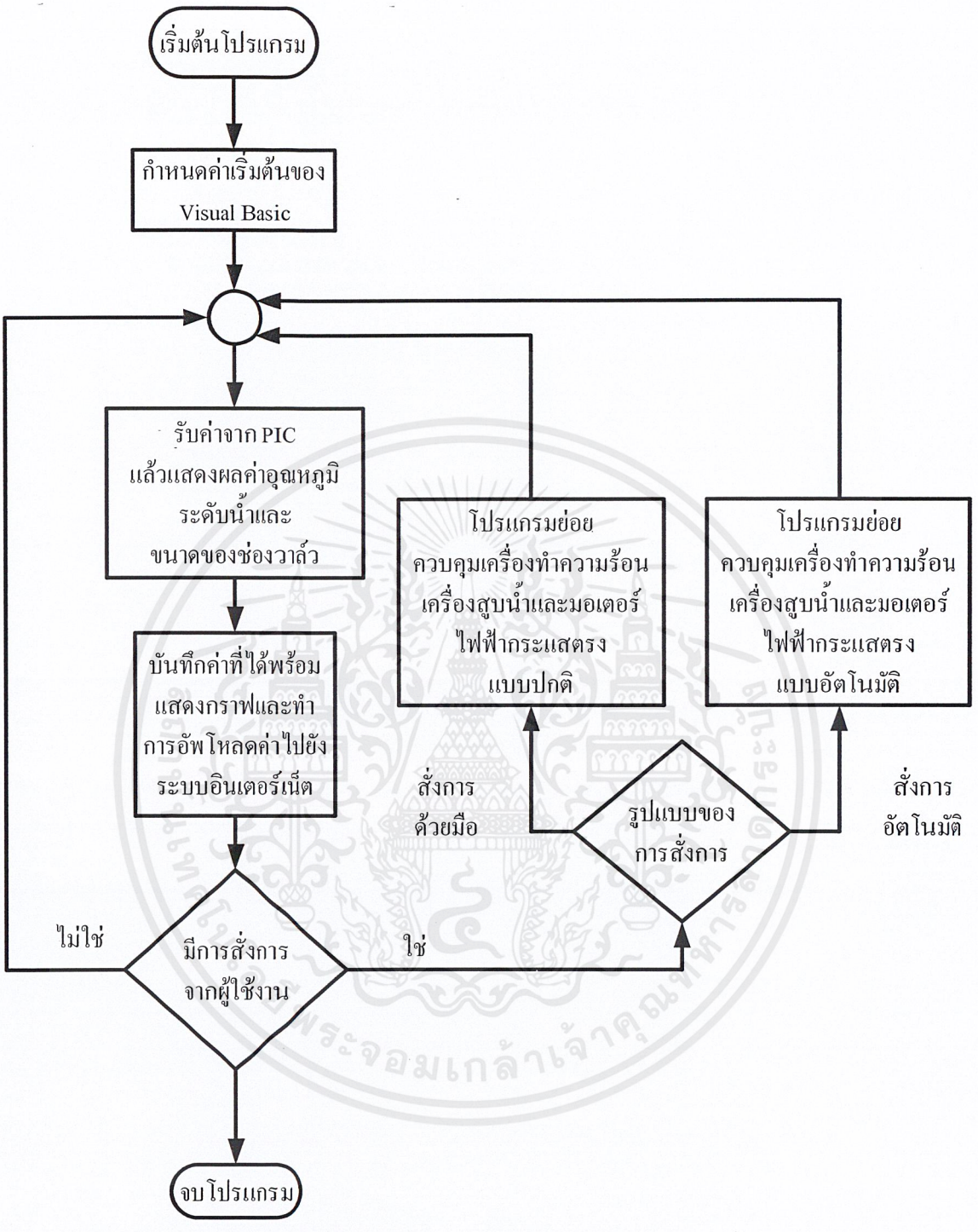
3.9 การทำงานของโปรแกรม Visual Basic

โดยการทำงานของโปรแกรม Visual Basic จะเริ่มจากการกำหนดค่าเริ่มต้นของโปรแกรมแล้วจะทำการรับค่าจากไมโครคอนโทรลเลอร์ ซึ่งค่าที่รับมาจะเป็นค่าจากเครื่องมือวัดในกระบวนการ คือ ค่าอุณหภูมิ ระดับน้ำในถัง และขนาดของช่องวาล์ว มาแสดงผลในโปรแกรมดังกล่าว พร้อมทั้งทำการบันทึกค่าที่รับมาลงในรูปแบบของ Text File และอัพโหลดค่าที่รับมาไปยังระบบ Internet

ถ้ามีการสั่งการจากผู้ใช้งาน โปรแกรมจะเรียกโปรแกรมย่อยเพื่อตอบสนองผู้ใช้ โดยจะมีสองโปรแกรมย่อย คือ โปรแกรมย่อยของการควบคุมแบบปกติ และโปรแกรมย่อยของการควบคุมแบบอัตโนมัติ ซึ่งรายละเอียดของโปรแกรมย่อยดังกล่าว จะมีดังนี้

- โปรแกรมย่อยของการควบคุมแบบปกติ คือ การสั่งการจากผู้ใช้งานโดยการกดปุ่ม ON-OFF ซึ่งจะเป็นการสั่งการแบบปกติ
- โปรแกรมย่อยของการควบคุมแบบอัตโนมัติ คือ การสั่งการจากผู้ใช้งาน โดยผู้ใช้งานจะต้องระบุค่าในกระบวนการที่ต้องการ แล้วโปรแกรมจะทำการควบคุมกระบวนการให้ได้ค่าที่ผู้ใช้งานต้องการได้

การทำงานของโปรแกรม Visual Basic จะแสดงเป็นผังงานได้ ดังรูปที่ 3.19

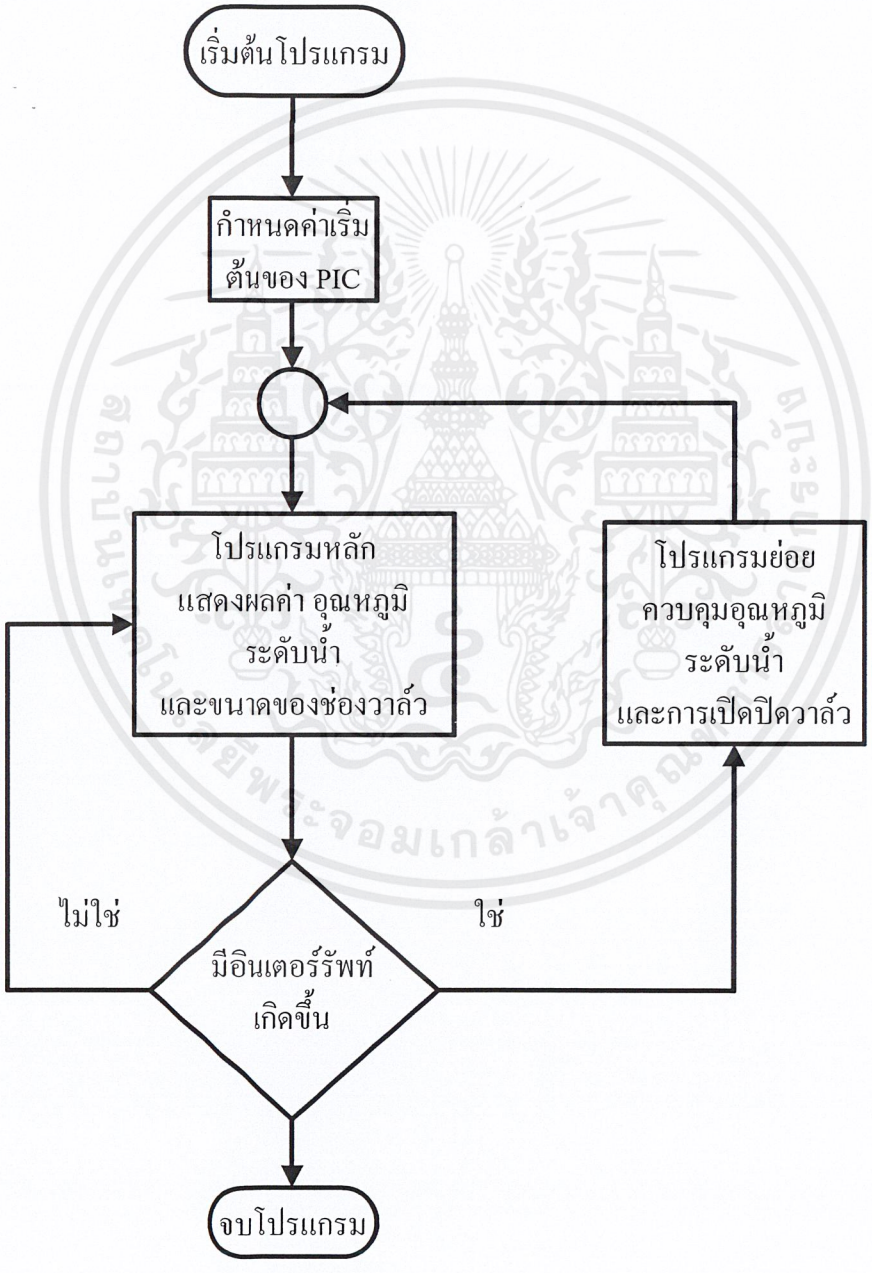


รูปที่ 3.19 ผังงานแสดงการทำงานของโปรแกรม Visual Basic

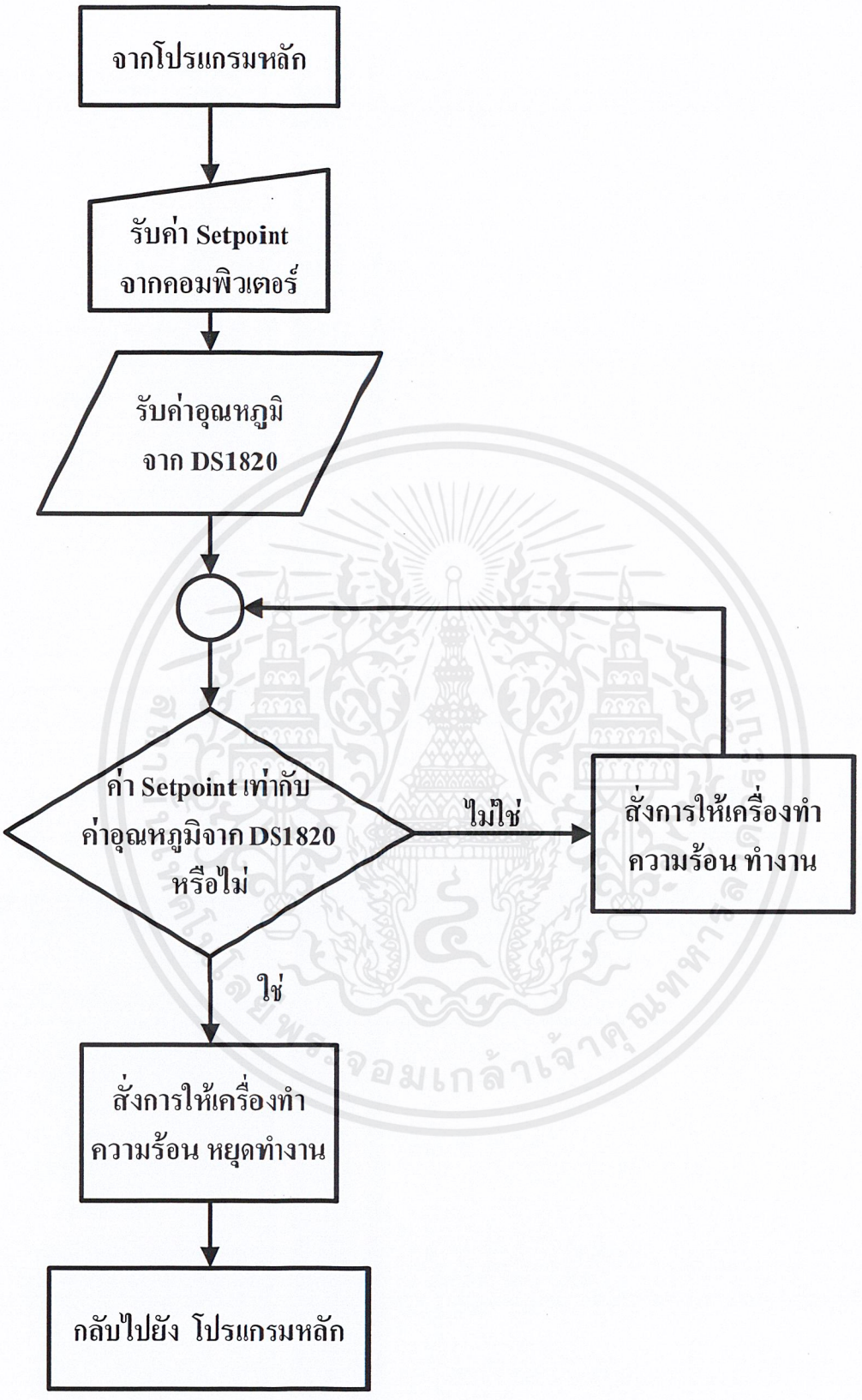
3.10 การทำงานของไมโครคอนโทรลเลอร์ 16F877

ในระบบควบคุมกระบวนการจะใช้ไมโครคอนโทรลเลอร์ 16F877 เป็นตัวประมวลผลกลางในการควบคุมอุปกรณ์และรับ-ส่งค่าจากเครื่องมือวัด ไปยังคอมพิวเตอร์ เพื่อแสดงผลและควบคุมต่อไป

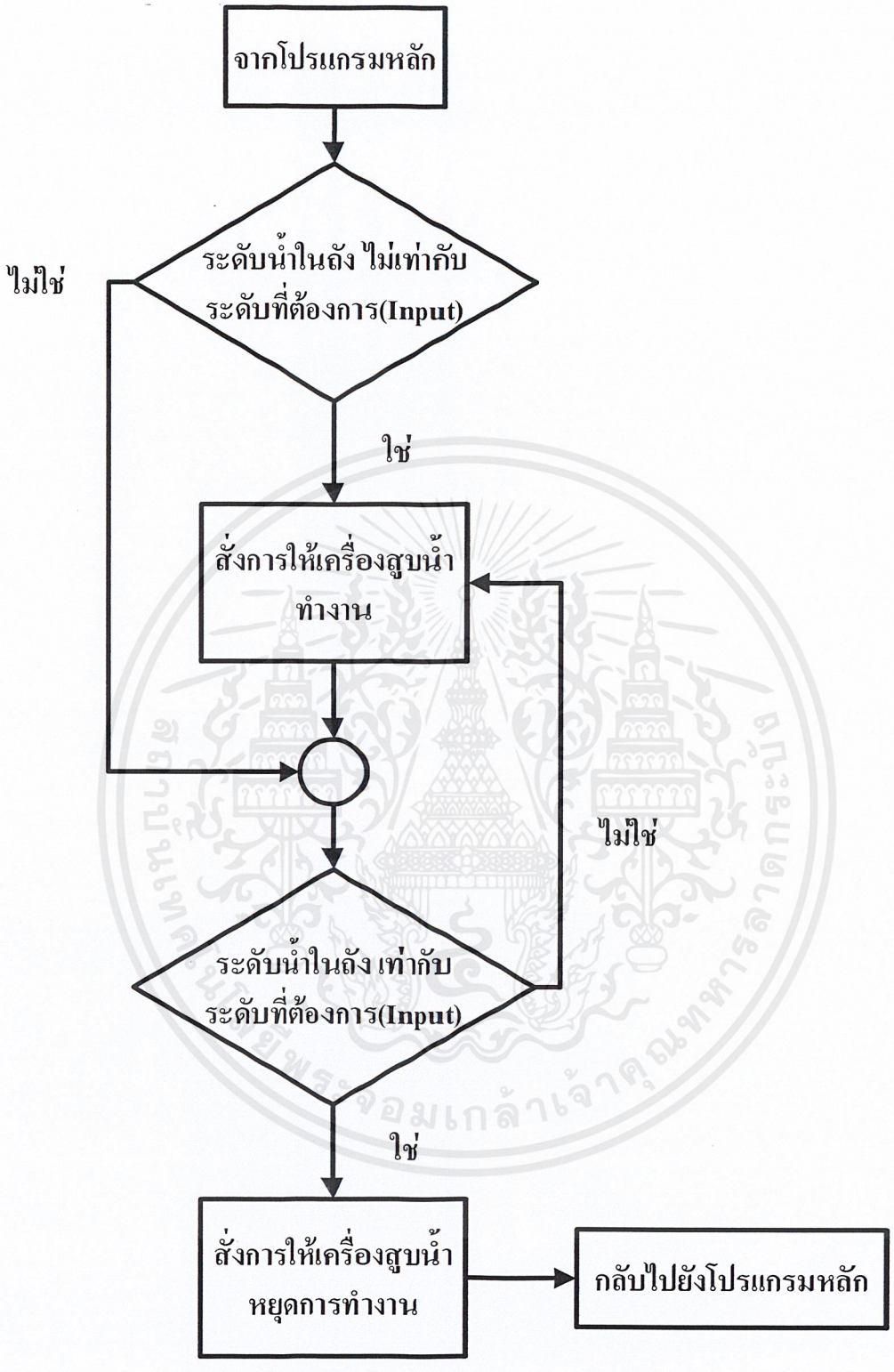
ผังงานแสดงการทำงานของไมโครคอนโทรลเลอร์ จะถูกแสดงไว้ ดังรูปที่ 3.20 ถึง รูปที่ 3.23



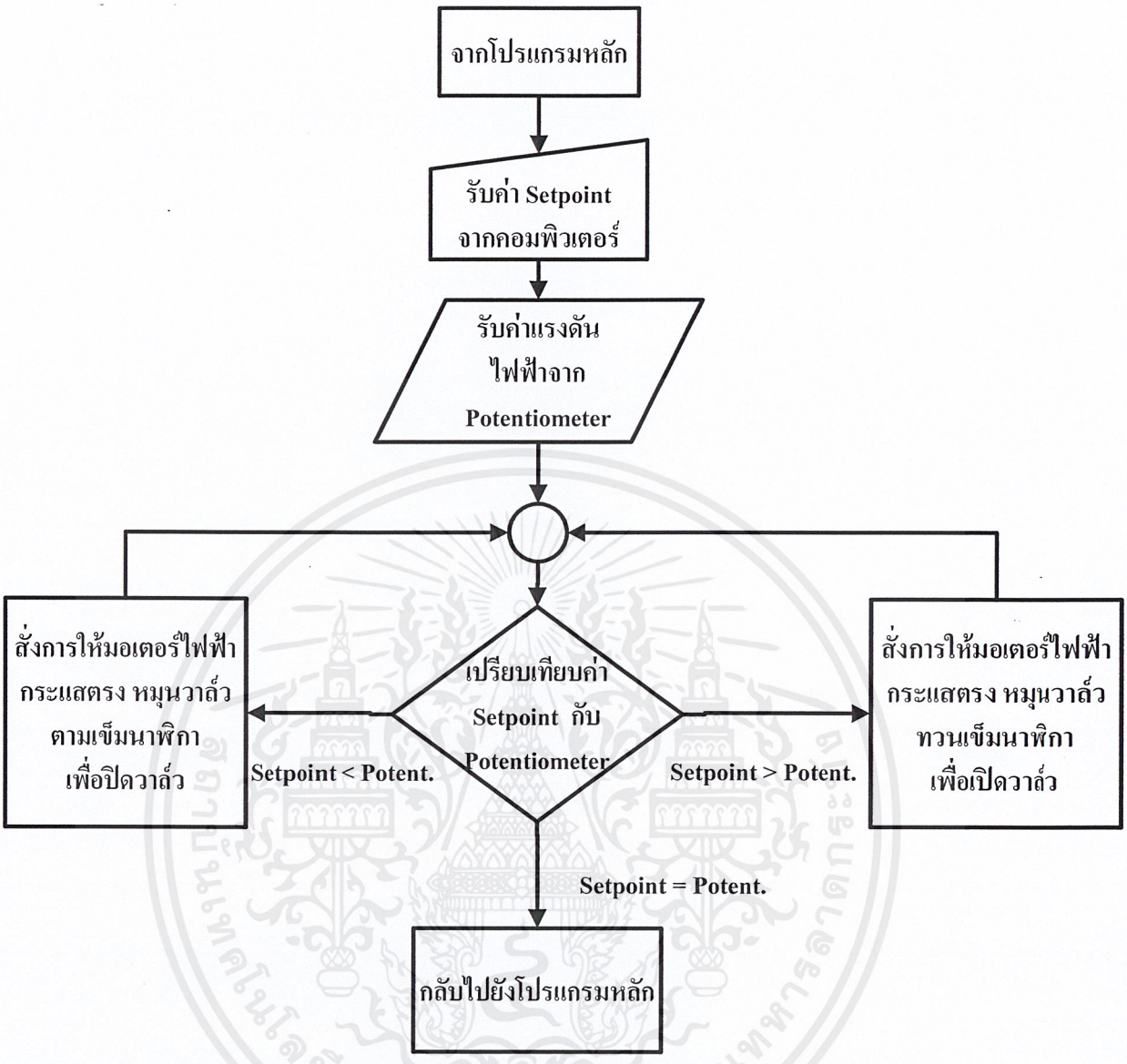
รูปที่ 3.20 ผังงานของโปรแกรมหลักในการควบคุมกระบวนการ



รูปที่ 3.21 ผังงานของโปรแกรมย่อยในการควบคุมอุณหภูมิ



รูปที่ 3.22 ผังงานของโปรแกรมย่อยในการควบคุมระดับน้ำ



รูปที่ 3.23 ผังงานของโปรแกรมย่อยในการควบคุมการเปิดปิดของวาล์ว

บทที่ 4
การทดลอง

4.1 กล่าวนำ

ในบทนี้จะกล่าวถึงการทดลองของกระบวนการต่างๆที่อยู่ภายในชุดทดลองระบบควบคุมกระบวนการ โดยจะแบ่งตามชนิดของอุปกรณ์ในส่วนต่างๆของระบบควบคุมกระบวนการ ซึ่งมีรายละเอียดการทดลองดังนี้

4.2 การทำงานของชุดควบคุมวาล์ว

ในส่วนนี้เป็นการศึกษาเฉพาะการทำงานของภาคการขับเคลื่อนมอเตอร์เพียงเท่านั้น โดยการนำค่าจากตัวต้านทานปรับค่าได้ที่ถูกติดตั้งในชุดมอเตอร์มาตรวจสอบ เพื่อที่จะได้ทราบตำแหน่งของวาล์วในขณะนั้น

4.2.1 การเปรียบเทียบค่าระหว่างผลการทดลองกับค่าทางทฤษฎีและค่าความคลาดเคลื่อน

ในการทดลองส่วนนี้ จะเป็นการวัดแรงดันไฟฟ้าจากตัวต้านทานปรับค่าได้ จากร้อยละการของการเปิด-ปิดของช่องวาล์วที่เปลี่ยนไป โดยจะตรวจสอบค่าของเอาท์พุตที่วัดได้จริงจากการทดลองและเปรียบเทียบค่ากับทางทฤษฎีที่สามารถคำนวณได้ และยังสามารถหาค่าความคลาดเคลื่อนในหน่วยร้อยละ โดยมีการเปรียบเทียบผลการทดลองดังนี้

ตารางที่ 4.1 ตารางเปรียบเทียบผลการทดลองกับผลทางทฤษฎี

การเปิดช่องวาล์ว (%)	ค่าจากทฤษฎี	ค่าจากการทดลอง	ค่าความผิดพลาดที่เกิดขึ้น (%)
	แรงดันไฟฟ้าจากตัวต้านทานปรับค่าได้ (โวลต์)	แรงดันไฟฟ้าจากตัวต้านทานปรับค่าได้ (โวลต์)	
0%	1.7300	1.7300	0.000000
5%	1.8215	1.8300	0.466648
10%	1.9130	1.9300	0.888657
15%	2.0045	2.0200	0.773260
20%	2.0960	2.1100	0.667939
25%	2.1875	2.1900	0.114286

ตารางที่ 4.1 (ต่อ) ตารางเปรียบเทียบผลการทดลองกับผลทางทฤษฎี

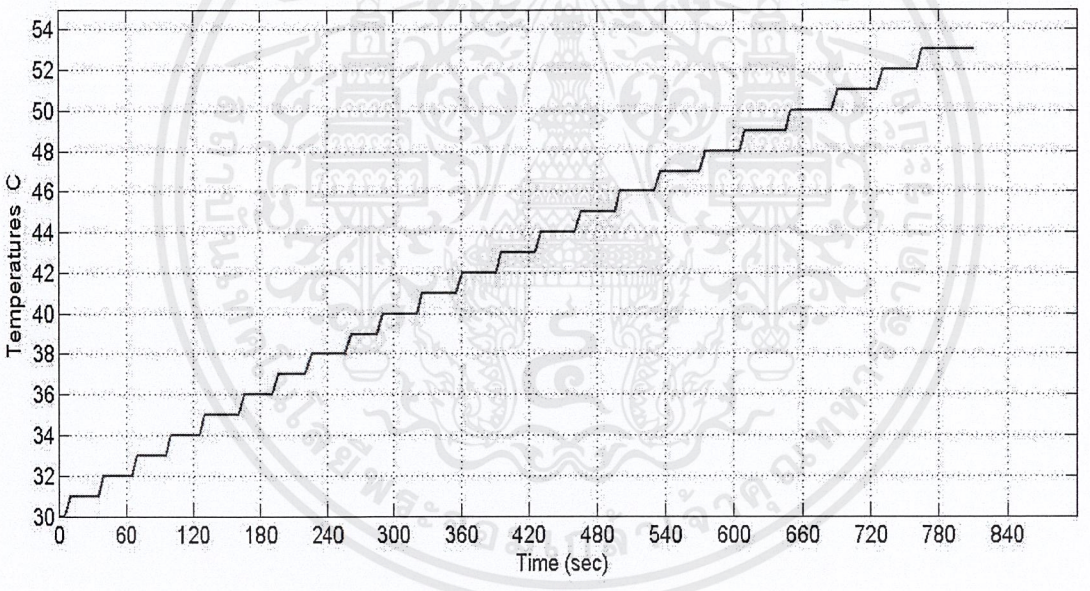
30%	2.2790	2.2900	0.482668
35%	2.3705	2.3800	0.400759
40%	2.4620	2.4700	0.324939
45%	2.5535	2.5700	0.646172
50%	2.6450	2.6500	0.189036
55%	2.7365	2.7400	0.127901
60%	2.8280	2.8400	0.424328
65%	2.9195	2.9300	0.359651
70%	3.0110	3.0200	0.298904
75%	3.1025	3.1100	0.241741
80%	3.1940	3.2100	0.500939
85%	3.2855	3.2900	0.136965
90%	3.3770	3.3900	0.384957
95%	3.4685	3.4800	0.331555
100%	3.56000	3.5600	0.000000
95%	3.4685	3.4700	0.043246
90%	3.3770	3.3800	0.088836
85%	3.2855	3.2900	0.136965
80%	3.1940	3.2100	0.500939
75%	3.1025	3.1000	0.080580
70%	3.0110	3.0100	0.033210
65%	2.9195	2.9200	0.017126
60%	2.8280	2.8400	0.424328
55%	2.7365	2.7500	0.493331
50%	2.6450	2.6500	0.189036
45%	2.5535	2.5600	0.254553
40%	2.4620	2.4700	0.324939
35%	2.3705	2.3700	0.021090
30%	2.2790	2.2900	0.482668
25%	2.1875	2.1900	0.114286

ตารางที่ 4.1 (ต่อ) ตารางเปรียบเทียบผลการทดลองกับผลทางทฤษฎี

20%	2.0960	2.1000	0.190840
15%	2.0045	2.0000	0.224490
10%	1.9130	1.9200	0.365917
5%	1.8215	1.8300	0.466648
0%	1.7300	1.7400	0.578035

4.3 การทำงานของชุดควบคุมอุณหภูมิ

ในส่วนนี้เป็นการทำงานของชุดวัดการเปลี่ยนแปลงของอุณหภูมิของของเหลวมาเทียบกับเวลา (วินาที) โดยการใช้ไอซี DS1820 ประกอบเข้ากับแท่งอลูมิเนียมกลวง ซึ่งปิดปลายข้างหนึ่งไว้แล้วใส่ลงไปในถังเก็บน้ำ โดยจะมีการแสดงผลผ่านทางหน้าจอคอมพิวเตอร์ในรูปแบบของกราฟ



รูปที่ 4.1 กราฟแสดงการวัดอุณหภูมิ ณ เวลาต่างๆ

4.4 การทำงานของชุดควบคุมระดับของเหลว

ในส่วนนี้เป็นการทำงานของชุดวัดระดับของเหลว โดยใช้เซนเซอร์ชนิดอินฟาเรด (GP2Y0A21YK) เป็นอุปกรณ์ที่ใช้ในการรับค่าระยะห่าง จากโฟมที่ลอยขึ้นมาตามระดับของของเหลวที่เพิ่มสูงขึ้น โดยเซนเซอร์จะทำหน้าที่เป็นตัวตรวจวัดระยะห่างระหว่างผิวหน้าของของเหลว (ที่ติดกับโฟม) กับตัวเซนเซอร์ ซึ่งค่าที่ได้จะอยู่ในลักษณะของแรงดันไฟฟ้า จากนั้นจะนำค่าที่ได้ส่งไปยังไมโครคอนโทรลเลอร์ และนำไปแสดงผลในหน้าจอคอมพิวเตอร์ต่อไป

บทที่ 5
บทวิจารณ์และสรุป

5.1 สรุปผลการทดลอง

การทดลองชุดระบบควบคุมกระบวนการสามารถทำงานร่วมกันเป็นระบบได้อย่างดี โดยระบบโดยรวมเป็นไปตามที่ได้ออกไว้อย่างถูกต้องโดยเครื่องสูบน้ำจะทำงานเมื่อมีการสั่งการและจะหยุดทำงานเมื่อระดับน้ำถึงจุดที่ตั้งไว้ เครื่องทำความร้อนจะทำงานโดยการสั่งงานและจะสามารถวัดอุณหภูมิที่เปลี่ยนแปลงไปได้ แต่ยังปรากฏความผิดพลาดให้เห็นอยู่ในส่วนของการหมุนวาล์ว ในบางครั้งการหมุนไปและการหมุนกลับนั้นอาจจะมีความเร็วรอบที่แตกต่างกันเล็กน้อยซึ่งจะส่งผลให้ในช่วงที่วาล์วเปิดสุด หรือปิดสุดนั้นมีปัญหาโดยจะทำให้วาล์วนั้นเปิด-ปิดไม่สมบูรณ์ ส่วนของการวัดระดับของเหลวนั้นค่าที่ได้ในกราฟอาจจะมีความคลาดเคลื่อนเล็กน้อย โดยกราฟของระดับของเหลวในบางช่วงจะไม่คงที่เนื่องจากมีการสั่นสะเทือนของระดับน้ำที่ผิวน้ำทำให้กราฟบางช่วงไม่สม่ำเสมอเท่าที่ควร แต่ปัญหาที่พบทั้งสองอย่างนี้จะไม่ส่งผลกระทบโดยรวมของระบบทั้งหมดแต่จะมีผลเพียงแค่ทำให้ผลการทดลองคลาดเคลื่อนเพียงเล็กน้อย

5.2 ปัญหาที่พบและแนวทางการแก้ไข

ในการจัดทำชุดทดลองระบบควบคุมกระบวนการนี้ ปัญหาที่พบในช่วงแรกคือการออกแบบชุดทดลองและระบบทั้งหมด เนื่องจากระบบทั้งหมดนั้นประกอบด้วยหลายๆกระบวนการจึงทำให้ต้องใช้อุปกรณ์ที่หลากหลายและยังมีพื้นที่ได้การติดตั้งอุปกรณ์ที่จำกัด จึงทำให้ต้องวางแผนและจัดสรรอุปกรณ์ที่เหมาะสมกับความต้องการ นอกจากนี้ยังพบปัญหาในส่วนของอุปกรณ์อิเล็กทรอนิกส์ซึ่งก็คือ ในชุดทดลองระบบควบคุมกระบวนการนี้ มีการทำงานของกระบวนการอยู่หลากหลาย ซึ่งเมื่อนำกระบวนการต่างๆเหล่านี้มาทำงานร่วมกันเป็นระบบ พบว่าอุปกรณ์อิเล็กทรอนิกส์บางตัวมีการขัดข้อง ไม่สามารถทำหน้าที่ในการรับ-ส่ง หรืออ่านค่าต่างๆได้ตามที่ออกแบบไว้ในบางครั้ง

สำหรับปัญหาบางส่วนที่ยังไม่สามารถแก้ไขได้ก็คือ ความเร็วของวาล์วในขณะหมุนไปและขณะหมุนกลับ มีความเร็วที่แตกต่างกันเล็กน้อย โดยเกิดจากการติดตั้งตัวฐานของวาล์วทำให้การเปิด-ปิดวาล์วในตำแหน่งสูงสุดอาจมีค่าความคลาดเคลื่อนเล็กน้อย

5.3 ข้อเสนอแนะและแนวทางค้นคว้าพัฒนา

ในการควบคุมการหมุนเปิด-ปิดของวาล์วนั้น ถ้าต้องการให้มีความแม่นยำและควบคุมได้ดียิ่งขึ้นอาจจะมีการใช้ตัวควบคุมเข้ามาช่วยในระบบ ตัวควบคุมนั้นมีหลากหลายแบบขึ้นอยู่กับว่าต้องการให้ผลลัพท์นั้นเป็นไปในทิศทางใด ซึ่งการเลือกใช้ตัวควบคุมอาจจะเลือกใช้เป็นแบบวงจรอิเล็กทรอนิกส์หรือใช้การเขียนโปรแกรมควบคุมโดยตรงได้เช่นกัน

ภาคผนวก

ภาคผนวก ก

โปรแกรมควบคุมการทำงานของไมโครคอนโทรลเลอร์

โปรแกรมภาษาซี ซึ่งใช้ในการควบคุมการทำงานของไมโครคอนโทรลเลอร์ เพื่อที่จะรับส่งข้อมูลจากเครื่องมือวัด และอุปกรณ์ภายในระบบควบคุมกระบวนการ ไปยังคอมพิวเตอร์เพื่อแสดงผลและควบคุมต่อไป

```
#include <16F877A.h>
#device ADC=10 // setup ADC
#fuses HS,NOWDT,NOPROTECT,NOLVP
#use delay(clock=20000000)
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)

int16 value1,value2; //value1=Potentiometer ,value2=Level
unsigned int temp;
BOOLEAN valve=FALSE;
unsigned char str[4];

#INT_RDA // interrupt for recieve data from computer
void IntRDA_isr(void)
{
  disable_interrupts(INT_RDA);
  gets(str);

  if(str[0]=='o' && str[1]=='p')
    {
      output_high(Pin_D4); // on pump
      str[0]='x';
    }
  else if(str[0]=='c' && str[1]=='p') // off pump
    {
      output_low(Pin_D4);
      str[0]='x';
```

```
  }
else if(str[0]=='o' && str[1]=='h') // on heater
  {
    output_high(Pin_C0);
    str[0]='x';
  }
else if(str[0]=='c' && str[1]=='h') // off heater
  {
    output_low(Pin_C0);
    str[0]='x';
  }
 else if(str[0]=='o' && str[1]=='v') // open valve
  {
      valve=TRUE;
  }
  else if(str[0]=='c' && str[1]=='v') // close valve
  {
      valve=TRUE;
  }
 else if(str[0]=='c' && str[1]=='x') // close valve
  {
      valve=TRUE;
  }
 else if(str[0]=='o' && str[1]=='x') // close valve
  {
      valve=TRUE;
  }
 else if(str[0]=='c' && str[1]=='y') // close valve
  {
      valve=TRUE;
  }
```

```
  else if(str[0]=='o' && str[1]=='y') // close valve
     {
          valve=TRUE;
     }
 enable_interrupts(INT_RDA);
}
//#####################  DS1820 ##########################################//

#ifndef TOUCH_PIN
  #define TOUCH_PIN  PIN_B1
#endif

BYTE touch_read_byte()
{
  BYTE i,data;

  for(i=1;i<=8;++i)
  {
    output_low(TOUCH_PIN);
    delay_us(14);
    output_float(TOUCH_PIN);
    delay_us(5);
    shift_right(&data,1,input(TOUCH_PIN));
    delay_us(100);
  }
  return(data);
}

BYTE touch_write_byte(BYTE data) {
  BYTE i;

  for(i=1;i<=8;++i)
```

```
  {
    output_low(TOUCH_PIN);
    delay_us(10);
    if(shift_right(&data,1,0))
    {
     output_high(TOUCH_PIN);
     delay_us(10);
     if(!input_state(TOUCH_PIN)) return(0);
    }
    else
    {
     output_low(TOUCH_PIN);
     delay_us(10);
     if(input_state(TOUCH_PIN)) return(0);
    }
    delay_us(50);
    output_high(TOUCH_PIN);
    delay_us(50);
  }
  return(TRUE);
}

BYTE touch_present()
{
   BOOLEAN present;

   output_low(TOUCH_PIN);
   delay_us(500);
   output_float(TOUCH_PIN);

   delay_us(5);
```

```
  if(!input(TOUCH_PIN)) return(FALSE);
  delay_us(65);
  present=!input(TOUCH_PIN);
  delay_us(240);
  if(present) return(TRUE);
  else return(FALSE);
}
//#################### END DS1820 ########################################//
void readtemp(void)
{
  BYTE buffer;
  if(touch_present())
  {
    touch_write_byte(0xCC);
    touch_write_byte(0x44);
    delay_ms(100);
    touch_present();
    touch_write_byte(0xCC);
    touch_write_byte(0xBE);
     buffer=touch_read_byte();
  }
 temp=buffer/2;
}
void showtemp(void)
{
    readtemp();
    delay_ms(10);
    printf("t=%d\n",temp); // display temperature value
}
void potentiometer(void)
{
      set_adc_channel(0);
```

```
        delay_us(20);
        value1 = read_adc();
        printf("v=%lu\n",value1); // display voltage value
        delay_us(50);
}
void level(void)
{
        set_adc_channel(1);
        delay_us(20);
        value2 = read_adc();
        printf("l=%lu\n",value2); // display voltage value
        delay_us(50);
}
void CheckValve(void)
{
    if(str[0]=='o' && str[1]=='v')
      {
          output_E(0x05);
          delay_ms(1300);
          output_E(0x00);
          str[0]='x';
      }
    else if(str[0]=='c' && str[1]=='v')
      {
          output_E(0x06);
          delay_ms(1300);
          output_E(0x00);
          str[0]='x';
      }
    else if(str[0]=='o' && str[1]=='x')
      {
          output_E(0x05);
```

```
            delay_ms(300);
            output_E(0x00);
            str[0]='x';
        }
    else if(str[0]=='c' && str[1]=='x')
        {
            output_E(0x06);
            delay_ms(300);
            output_E(0x00);
            str[0]='x';
        }
    else if(str[0]=='o' && str[1]=='y')
        {
            output_E(0x05);
            delay_ms(150);
            output_E(0x00);
            str[0]='x';
        }
    else if(str[0]=='c' && str[1]=='y')
        {
            output_E(0x06);
            delay_ms(150);
            output_E(0x00);
            str[0]='x';
        }
}
void main(void)
{
set_tris_B(0x0F);
set_tris_D(0x00);
set_tris_E(0x00);
delay_us(10);
```

```
output_D(0x00);
output_E(0x00);
delay_us(10);
enable_interrupts(GLOBAL);
enable_interrupts(INT_RDA);
setup_port_a(AN0_AN1_AN3);
setup_adc(ADC_CLOCK_INTERNAL);
delay_us(50);
while(true)
  {
   start: // start program
   while(valve)
    {
      CheckValve();
      valve=FALSE;
      goto start;  // loop to start
    }
     potentiometer();
     delay_ms(500);

     level();
     delay_ms(500);

     showtemp();
     delay_ms(500);
  } // end while(true)
} // end main
```

ภาคผนวก ข

DALLAS SEMICONDUCTOR MAXIM

DS18S20
High-Precision 1-Wire Digital Thermometer

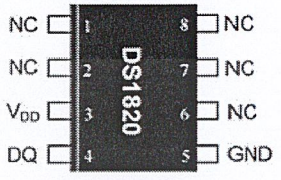
www.maxim-ic.com

FEATURES

- Unique 1-Wire® interface requires only one port pin for communication
- Each device has a unique 64-bit serial code stored in an onboard ROM
- Multidrop capability simplifies distributed temperature sensing applications
- Requires no external components
- Can be powered from data line. Power supply range is 3.0V to 5.5V
- Measures temperatures from –55°C to +125°C (–67°F to +257°F)
- ±0.5°C accuracy from –10°C to +85°C
- 9-bit thermometer resolution
- Converts temperature in 750ms (max.)
- User-definable nonvolatile (NV) alarm settings
- Alarm search command identifies and addresses devices whose temperature is outside of programmed limits (temperature alarm condition)
- Applications include thermostatic controls, industrial systems, consumer products, thermometers, or any thermally sensitive system

PIN ASSIGNMENT

TO-92
(DS18S20)



8-Pin 150mil SO
(DS18S20Z)

PIN DESCRIPTION

GND - Ground
DQ - Data In/Out
V_{DD} - Power Supply Voltage
NC - No Connect

DESCRIPTION

The DS18S20 Digital Thermometer provides 9-bit centigrade temperature measurements and has an alarm function with nonvolatile user-programmable upper and lower trigger points. The DS18S20 communicates over a 1-Wire bus that by definition requires only one data line (and ground) for communication with a central microprocessor. It has an operating temperature range of –55°C to +125°C and is accurate to ±0.5°C over the range of –10°C to +85°C. In addition, the DS18S20 can derive power directly from the data line ("parasite power"), eliminating the need for an external power supply.

Each DS18S20 has a unique 64-bit serial code, which allows multiple DS18S20s to function on the same 1-Wire bus; thus, it is simple to use one microprocessor to control many DS18S20s distributed over a large area. Applications that can benefit from this feature include HVAC environmental controls, temperature monitoring systems inside buildings, equipment or machinery, and process monitoring and control systems.

1-Wire is a registered trademark of Dallas Semiconductor.

DETAILED PIN DESCRIPTIONS Table 1

8-PIN SOIC*	TO-92	SYMBOL	DESCRIPTION
5	1	GND	**Ground.**
4	2	DQ	**Data Input/Output Pin.** Open-drain 1-Wire interface pin. Also provides power to the device when used in parasite power mode (see "Parasite Power" section.)
3	3	V_{DD}	**Optional V_{DD} Pin.** V_{DD} must be grounded for operation in parasite power mode.

*All pins not specified in this table are "No Connect" pins.

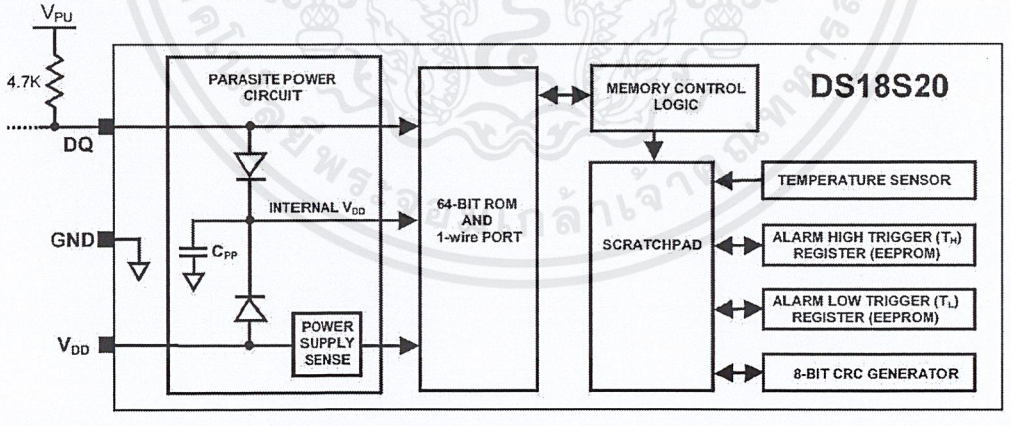
OVERVIEW

Figure 1 shows a block diagram of the DS18S20, and pin descriptions are given in Table 1. The 64-bit ROM stores the device's unique serial code. The scratchpad memory contains the 2-byte temperature register that stores the digital output from the temperature sensor. In addition, the scratchpad provides access to the 1-byte upper and lower alarm trigger registers (T_H and T_L). The T_H and T_L registers are nonvolatile (EEPROM), so they will retain data when the device is powered down.

The DS18S20 uses Dallas' exclusive 1-Wire bus protocol that implements bus communication using one control signal. The control line requires a weak pullup resistor since all devices are linked to the bus via a 3-state or open-drain port (the DQ pin in the case of the DS18S20). In this bus system, the microprocessor (the master device) identifies and addresses devices on the bus using each device's unique 64-bit code. Because each device has a unique code, the number of devices that can be addressed on one bus is virtually unlimited. The 1-Wire bus protocol, including detailed explanations of the commands and "time slots," is covered in the *1-WIRE BUS SYSTEM* section of this datasheet.

Another feature of the DS18S20 is the ability to operate without an external power supply. Power is instead supplied through the 1-Wire pullup resistor via the DQ pin when the bus is high. The high bus signal also charges an internal capacitor (C_{PP}), which then supplies power to the device when the bus is low. This method of deriving power from the 1-Wire bus is referred to as "parasite power." As an alternative, the DS18S20 may also be powered by an external supply on V_{DD}.

DS18S20 BLOCK DIAGRAM Figure 1



OPERATION — MEASURING TEMPERATURE

The core functionality of the DS18S20 is its direct-to-digital temperature sensor. The temperature sensor output has 9-bit resolution, which corresponds to 0.5°C steps. The DS18S20 powers-up in a low-power idle state; to initiate a temperature measurement and A-to-D conversion, the master must issue a Convert T [44h] command. Following the conversion, the resulting thermal data is stored in the 2-byte temperature register in the scratchpad memory and the DS18S20 returns to its idle state. If the DS18S20 is powered by an external supply, the master can issue "read-time slots" (see the *1-WIRE BUS SYSTEM* section) after the Convert T command and the DS18S20 will respond by transmitting 0 while the temperature conversion is in progress and 1 when the conversion is done. If the DS18S20 is powered with parasite power, this notification technique cannot be used since the bus must be pulled high by a strong pullup during the entire temperature conversion. The bus requirements for parasite power are explained in detail in the *POWERING THE DS18S20* section of this datasheet.

The DS18S20 output data is calibrated in degrees centigrade; for Fahrenheit applications, a lookup table or conversion routine must be used. The temperature data is stored as a 16-bit sign-extended two's complement number in the temperature register (see Figure 2). The sign bits (S) indicate if the temperature is positive or negative: for positive numbers S = 0 and for negative numbers S = 1. Table 2 gives examples of digital output data and the corresponding temperature reading.

Resolutions greater than 9 bits can be calculated using the data from the temperature, COUNT REMAIN and COUNT PER °C registers in the scratchpad. Note that the COUNT PER °C register is hard-wired to 16 (10h). After reading the scratchpad, the TEMP_READ value is obtained by truncating the 0.5°C bit (bit 0) from the temperature data (see Figure 2). The extended resolution temperature can then be calculated using the following equation:

$$TEMPERATURE = TEMP\_READ - 0.25 + \frac{COUNT\_PER\_C - COUNT\_REMAIN}{COUNT\_PER\_C}$$

Additional information about high-resolution temperature calculations can be found in *Application Note 105: High Resolution Temperature Measurement with Dallas Direct-to-Digital Temperature Sensors.*

TEMPERATURE REGISTER FORMAT Figure 2

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
LS Byte	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}
	bit 15	**bit 14**	**bit 13**	**bit 12**	**bit 11**	**bit 10**	**bit 9**	**bit 8**
MS Byte	S	S	S	S	S	S	S	S

TEMPERATURE/DATA RELATIONSHIP Table 2

TEMPERATURE	DIGITAL OUTPUT (Binary)	DIGITAL OUTPUT (Hex)
+85.0°C*	0000 0000 1010 1010	00AAh
+25.0°C	0000 0000 0011 0010	0032h
+0.5°C	0000 0000 0000 0001	0001h
0°C	0000 0000 0000 0000	0000h
-0.5°C	1111 1111 1111 1111	FFFFh
-25.0°C	1111 1111 1100 1110	FFCEh
-55.0°C	1111 1111 1001 0010	FF92h

*The power-on reset value of the temperature register is +85°C

OPERATION — ALARM SIGNALING

After the DS18S20 performs a temperature conversion, the temperature value is compared to the user-defined two's complement alarm trigger values stored in the 1-byte T_H and T_L registers (see Figure 3). The sign bit (S) indicates if the value is positive or negative: for positive numbers S = 0 and for negative numbers S = 1. The T_H and T_L registers are nonvolatile (EEPROM) so they will retain data when the device is powered down. T_H and T_L can be accessed through bytes 2 and 3 of the scratchpad as explained in the MEMORY section of this datasheet.

T_H AND T_L REGISTER FORMAT Figure 3

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
S	2^6	2^5	2^5	2^5	2^2	2^1	2^0

Only bits 8 through 1 of the temperature register are used in the T_H and T_L comparison since T_H and T_L are 8-bit registers. If the measured temperature is lower than or equal to T_L or higher than T_H, an alarm condition exists and an alarm flag is set inside the DS18S20. This flag is updated after every temperature measurement; therefore, if the alarm condition goes away, the flag will be turned off after the next temperature conversion.

The master device can check the alarm flag status of all DS18S20s on the bus by issuing an Alarm Search [ECh] command. Any DS18S20s with a set alarm flag will respond to the command, so the master can determine exactly which DS18S20s have experienced an alarm condition. If an alarm condition exists and the T_H or T_L settings have changed, another temperature conversion should be done to validate the alarm condition.

POWERING THE DS18S20

The DS18S20 can be powered by an external supply on the V_{DD} pin, or it can operate in "parasite power" mode, which allows the DS18S20 to function without a local external supply. Parasite power is very useful for applications that require remote temperature sensing or that are very space constrained. Figure 1 shows the DS18S20's parasite-power control circuitry, which "steals" power from the 1-Wire bus via the DQ pin when the bus is high. The stolen charge powers the DS18S20 while the bus is high, and some of the charge is stored on the parasite power capacitor (C_{PP}) to provide power when the bus is low. When the DS18S20 is used in parasite power mode, the V_{DD} pin must be connected to ground.

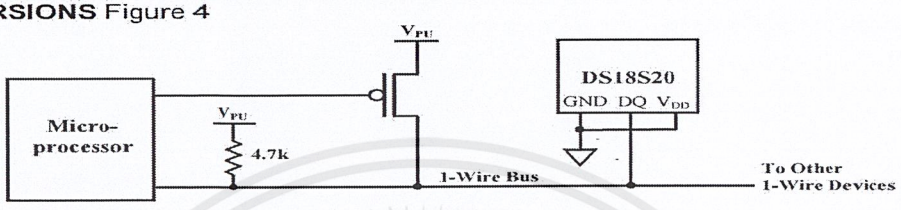
In parasite power mode, the 1-Wire bus and C_{PP} can provide sufficient current to the DS18S20 for most operations as long as the specified timing and voltage requirements are met (refer to the *DC ELECTRICAL CHARACTERISTICS* and the *AC ELECTRICAL CHARACTERISTICS* sections of this data sheet). However, when the DS18S20 is performing temperature conversions or copying data from the scratchpad memory to EEPROM, the operating current can be as high as 1.5mA. This current can cause an unacceptable voltage drop across the weak 1-Wire pullup resistor and is more current than can be supplied by C_{PP}. To assure that the DS18S20 has sufficient supply current, it is necessary to provide a strong pullup on the 1-Wire bus whenever temperature conversions are taking place or data is being copied from the scratchpad to EEPROM. This can be accomplished by using a MOSFET to pull the bus directly to the rail as shown in Figure 4. The 1-Wire bus must be switched to the strong pullup within 10µs (max) after a Convert T [44h] or Copy Scratchpad [48h] command is issued, and the bus must be held high by the pullup for the duration of the conversion (t_{conv}) or data transfer (t_{WR} = 10ms). No other activity can take place on the 1-Wire bus while the pullup is enabled.

The DS18S20 can also be powered by the conventional method of connecting an external power supply to the V_{DD} pin, as shown in Figure 5. The advantage of this method is that the MOSFET pullup is not required, and the 1-Wire bus is free to carry other traffic during the temperature conversion time.

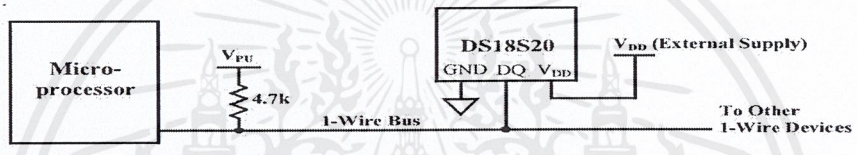
The use of parasite power is not recommended for temperatures above 100°C since the DS18S20 may not be able to sustain communications due to the higher leakage currents that can exist at these temperatures. For applications in which such temperatures are likely, it is strongly recommended that the DS18S20 be powered by an external power supply.

In some situations the bus master may not know whether the DS18S20s on the bus are parasite powered or powered by external supplies. The master needs this information to determine if the strong bus pullup should be used during temperature conversions. To get this information, the master can issue a Skip ROM [CCh] command followed by a Read Power Supply [B4h] command followed by a "read-time slot". During the read-time slot, parasite powered DS18S20s will pull the bus low, and externally powered DS18S20s will let the bus remain high. If the bus is pulled low, the master knows that it must supply the strong pullup on the 1-Wire bus during temperature conversions.

SUPPLYING THE PARASITE-POWERED DS18S20 DURING TEMPERATURE CONVERSIONS Figure 4



POWERING THE DS18S20 WITH AN EXTERNAL SUPPLY Figure 5



64-BIT LASERED ROM CODE

Each DS18S20 contains a unique 64-bit code (see Figure 6) stored in ROM. The least significant 8 bits of the ROM code contain the DS18S20's 1-Wire family code: 10h. The next 48 bits contain a unique serial number. The most significant 8 bits contain a cyclic redundancy check (CRC) byte that is calculated from the first 56 bits of the ROM code. A detailed explanation of the CRC bits is provided in the *CRC GENERATION* section. The 64-bit ROM code and associated ROM function control logic allow the DS18S20 to operate as a 1-Wire device using the protocol detailed in the *1-WIRE BUS SYSTEM* section of this datasheet.

64-BIT LASERED ROM CODE Figure 6

8-BIT CRC		48-BIT SERIAL NUMBER		8-BIT FAMILY CODE (10h)	
MSB	LSB	MSB	LSB	MSB	LSB

MEMORY

The DS18S20's memory is organized as shown in Figure 7. The memory consists of an SRAM scratchpad with nonvolatile EEPROM storage for the high and low alarm trigger registers (T_H and T_L). Note that if the DS18S20 alarm function is not used, the T_H and T_L registers can serve as general-purpose memory. All memory commands are described in detail in the *DS18S20 FUNCTION COMMANDS* section.

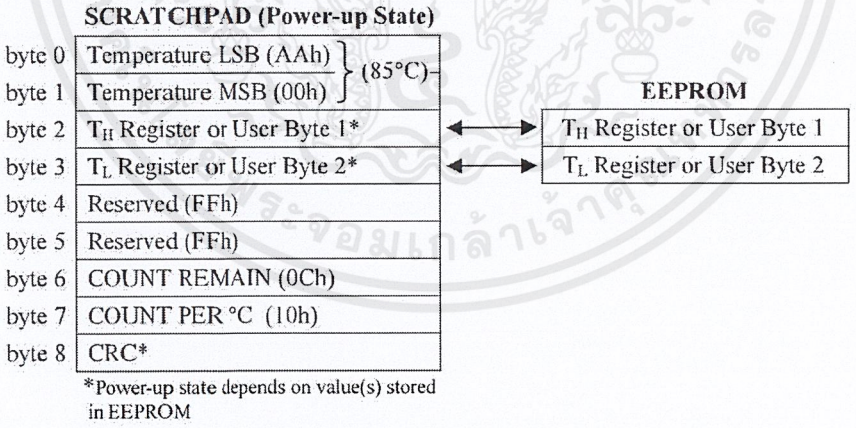
Byte 0 and byte 1 of the scratchpad contain the LSB and the MSB of the temperature register, respectively. These bytes are read-only. Bytes 2 and 3 provide access to T_H and T_L registers. Bytes 4 and 5 are reserved for internal use by the device and cannot be overwritten; these bytes will return all 1s when read. Bytes 6 and 7 contain the COUNT REMAIN and COUNT PER °C registers, which can be used to calculate extended resolution results as explained in the *OPERATION — MEASURING TEMPERATURE* section.

Byte 8 of the scratchpad is read-only and contains the cyclic redundancy check (CRC) code for bytes 0 through 7 of the scratchpad. The DS18S20 generates this CRC using the method described in *the CRC GENERATION* section.

Data is written to bytes 2 and 3 of the scratchpad using the Write Scratchpad [4Eh] command; the data must be transmitted to the DS18S20 starting with the least significant bit of byte 2. To verify data integrity, the scratchpad can be read (using the Read Scratchpad [BEh] command) after the data is written. When reading the scratchpad, data is transferred over the 1-Wire bus starting with the least significant bit of byte 0. To transfer the T_H and T_L data from the scratchpad to EEPROM, the master must issue the Copy Scratchpad [48h] command.

Data in the EEPROM registers is retained when the device is powered down; at power-up the EEPROM data is reloaded into the corresponding scratchpad locations. Data can also be reloaded from EEPROM to the scratchpad at any time using the Recall E^2 [B8h] command. The master can issue "read-time slots" (see the *1-WIRE BUS SYSTEM* section) following the Recall E^2 command and the DS18S20 will indicate the status of the recall by transmitting 0 while the recall is in progress and 1 when the recall is done.

DS18S20 MEMORY MAP Figure 7



*Power-up state depends on value(s) stored in EEPROM

CRC GENERATION

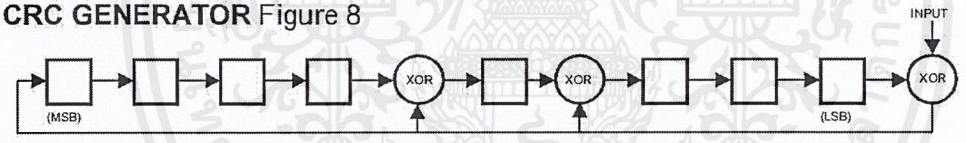
CRC bytes are provided as part of the DS18S20's 64-bit ROM code and in the 9th byte of the scratchpad memory. The ROM code CRC is calculated from the first 56 bits of the ROM code and is contained in the most significant byte of the ROM. The scratchpad CRC is calculated from the data stored in the scratchpad, and therefore it changes when the data in the scratchpad changes. The CRCs provide the bus master with a method of data validation when data is read from the DS18S20. To verify that data has been read correctly, the bus master must re-calculate the CRC from the received data and then compare this value to either the ROM code CRC (for ROM reads) or to the scratchpad CRC (for scratchpad reads). If the calculated CRC matches the read CRC, the data has been received error free. The comparison of CRC values and the decision to continue with an operation are determined entirely by the bus master. There is no circuitry inside the DS18S20 that prevents a command sequence from proceeding if the DS18S20 CRC (ROM or scratchpad) does not match the value generated by the bus master.

The equivalent polynomial function of the CRC (ROM or scratchpad) is:

$$CRC = X^8 + X^5 + X^4 + 1$$

The bus master can re-calculate the CRC and compare it to the CRC values from the DS18S20 using the polynomial generator shown in Figure 8. This circuit consists of a shift register and XOR gates, and the shift register bits are initialized to 0. Starting with the least significant bit of the ROM code or the least significant bit of byte 0 in the scratchpad, one bit at a time should shifted into the shift register. After shifting in the 56th bit from the ROM or the most significant bit of byte 7 from the scratchpad, the polynomial generator will contain the re-calculated CRC. Next, the 8-bit ROM code or scratchpad CRC from the DS18S20 must be shifted into the circuit. At this point, if the re-calculated CRC was correct, the shift register will contain all 0s. Additional information about the Dallas 1-Wire cyclic redundancy check is available in *Application Note 27: Understanding and Using Cyclic Redundancy Checks with Dallas Semiconductor Touch Memory Products.*

CRC GENERATOR Figure 8



1-WIRE BUS SYSTEM

The 1-Wire bus system uses a single bus master to control one or more slave devices. The DS18S20 is always a slave. When there is only one slave on the bus, the system is referred to as a "single-drop" system; the system is "multidrop" if there are multiple slaves on the bus.

All data and commands are transmitted least significant bit first over the 1-Wire bus.

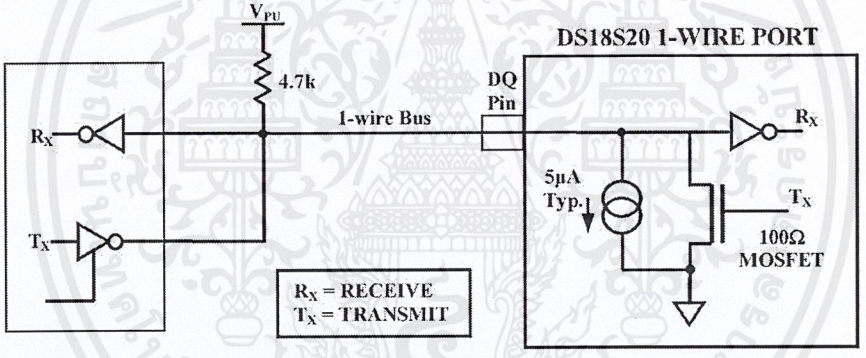
The following discussion of the 1-Wire bus system is broken down into three topics: hardware configuration, transaction sequence, and 1-Wire signaling (signal types and timing).

HARDWARE CONFIGURATION

The 1-Wire bus has by definition only a single data line. Each device (master or slave) interfaces to the data line via an open drain or 3-state port. This allows each device to "release" the data line when the device is not transmitting data so the bus is available for use by another device. The 1-Wire port of the DS18S20 (the DQ pin) is open drain with an internal circuit equivalent to that shown in Figure 9.

The 1-Wire bus requires an external pullup resistor of approximately 5kΩ; thus, the idle state for the 1-Wire bus is high. If for any reason a transaction needs to be suspended, the bus MUST be left in the idle state if the transaction is to resume. Infinite recovery time can occur between bits so long as the 1-Wire bus is in the inactive (high) state during the recovery period. If the bus is held low for more than 480µs, all components on the bus will be reset.

HARDWARE CONFIGURATION Figure 9



TRANSACTION SEQUENCE

The transaction sequence for accessing the DS18S20 is as follows:

Step 1. Initialization

Step 2. ROM Command (followed by any required data exchange)

Step 3. DS18S20 Function Command (followed by any required data exchange)

It is very important to follow this sequence every time the DS18S20 is accessed, as the DS18S20 will not respond if any steps in the sequence are missing or out of order. Exceptions to this rule are the Search ROM [F0h] and Alarm Search [ECh] commands. After issuing either of these ROM commands, the master must return to Step 1 in the sequence.

INITIALIZATION

All transactions on the 1-Wire bus begin with an initialization sequence. The initialization sequence consists of a reset pulse transmitted by the bus master followed by presence pulse(s) transmitted by the slave(s). The presence pulse lets the bus master know that slave devices (such as the DS18S20) are on the bus and are ready to operate. Timing for the reset and presence pulses is detailed in the *1-WIRE SIGNALING* section.

ROM COMMANDS

After the bus master has detected a presence pulse, it can issue a ROM command. These commands operate on the unique 64-bit ROM codes of each slave device and allow the master to single out a specific device if many are present on the 1-Wire bus. These commands also allow the master to determine how many and what types of devices are present on the bus or if any device has experienced an alarm condition. There are five ROM commands, and each command is 8 bits long. The master device must issue an appropriate ROM command before issuing a DS18S20 function command. A flowchart for operation of the ROM commands is shown in Figure 14.

SEARCH ROM [F0h]

When a system is initially powered up, the master must identify the ROM codes of all slave devices on the bus, which allows the master to determine the number of slaves and their device types. The master learns the ROM codes through a process of elimination that requires the master to perform a Search ROM cycle (i.e., Search ROM command followed by data exchange) as many times as necessary to identify all of the slave devices. If there is only one slave on the bus, the simpler Read ROM command (see below) can be used in place of the Search ROM process. For a detailed explanation of the Search ROM procedure, refer to the *iButton® Book of Standards* at www.ibutton.com/ibuttons/standard.pdf. After every Search ROM cycle, the bus master must return to Step 1 (Initialization) in the transaction sequence.

READ ROM [33h]

This command can only be used when there is one slave on the bus. It allows the bus master to read the slave's 64-bit ROM code without using the Search ROM procedure. If this command is used when there is more than one slave present on the bus, a data collision will occur when all the slaves attempt to respond at the same time.

MATCH ROM [55h]

The match ROM command followed by a 64-bit ROM code sequence allows the bus master to address a specific slave device on a multidrop or single-drop bus. Only the slave that exactly matches the 64-bit ROM code sequence will respond to the function command issued by the master; all other slaves on the bus will wait for a reset pulse.

SKIP ROM [CCh]

The master can use this command to address all devices on the bus simultaneously without sending out any ROM code information. For example, the master can make all DS18S20s on the bus perform simultaneous temperature conversions by issuing a Skip ROM command followed by a Convert T [44h] command.

Note that the Read Scratchpad [BEh] command can follow the Skip ROM command only if there is a single slave device on the bus. In this case time is saved by allowing the master to read from the slave without sending the device's 64-bit ROM code. A Skip ROM command followed by a Read Scratchpad command will cause a data collision on the bus if there is more than one slave since multiple devices will attempt to transmit data simultaneously.

ALARM SEARCH [ECh]

The operation of this command is identical to the operation of the Search ROM command except that only slaves with a set alarm flag will respond. This command allows the master device to determine if any DS18S20s experienced an alarm condition during the most recent temperature conversion. After every Alarm Search cycle (i.e., Alarm Search command followed by data exchange), the bus master must return to Step 1 (Initialization) in the transaction sequence. Refer to *the OPERATION — ALARM SIGNALING* section for an explanation of alarm flag operation.

DS18S20 FUNCTION COMMANDS

After the bus master has used a ROM command to address the DS18S20 with which it wishes to communicate, the master can issue one of the DS18S20 function commands. These commands allow the master to write to and read from the DS18S20's scratchpad memory, initiate temperature conversions and determine the power supply mode. The DS18S20 function commands, which are described below, are summarized in Table 4 and illustrated by the flowchart in Figure 15.

CONVERT T [44h]

This command initiates a single temperature conversion. Following the conversion, the resulting thermal data is stored in the 2-byte temperature register in the scratchpad memory and the DS18S20 returns to its low-power idle state. If the device is being used in parasite power mode, within 10μs (max) after this command is issued the master must enable a strong pullup on the 1-Wire bus for the duration of the conversion (t_{conv}) as described in the *POWERING THE DS18S20* section. If the DS18S20 is powered by an external supply, the master can issue read-time slots after the Convert T command and the DS18S20 will respond by transmitting 0 while the temperature conversion is in progress and 1 when the conversion is done. In parasite power mode this notification technique cannot be used since the bus is pulled high by the strong pullup during the conversion.

WRITE SCRATCHPAD [4Eh]

This command allows the master to write 2 bytes of data to the DS18S20's scratchpad. The first byte is written into the T_H register (byte 2 of the scratchpad), and the second byte is written into the T_L register (byte 3 of the scratchpad). Data must be transmitted least significant bit first. Both bytes MUST be written before the master issues a reset, or the data may be corrupted.

READ SCRATCHPAD [BEh]

This command allows the master to read the contents of the scratchpad. The data transfer starts with the least significant bit of byte 0 and continues through the scratchpad until the 9th byte (byte 8 – CRC) is read. The master may issue a reset to terminate reading at any time if only part of the scratchpad data is needed.

COPY SCRATCHPAD [48h]

This command copies the contents of the scratchpad T_H and T_L registers (bytes 2 and 3) to EEPROM. If the device is being used in parasite power mode, within 10μs (max) after this command is issued the master must enable a strong pullup on the 1-Wire bus for at least 10ms as described in the *POWERING THE DS18S20* section.

RECALL E^2 [B8h]

This command recalls the alarm trigger values (T_H and T_L) from EEPROM and places the data in bytes 2 and 3, respectively, in the scratchpad memory. The master device can issue read-time slots following the Recall E^2 command and the DS18S20 will indicate the status of the recall by transmitting 0 while the recall is in progress and 1 when the recall is done. The recall operation happens automatically at power-up, so valid data is available in the scratchpad as soon as power is applied to the device.

READ POWER SUPPLY [B4h]

The master device issues this command followed by a read-time slot to determine if any DS18S20s on the bus are using parasite power. During the read-time slot, parasite powered DS18S20s will pull the bus low, and externally powered DS18S20s will let the bus remain high. Refer to the *POWERING THE DS18S20* section for usage information for this command.

DS18S20 FUNCTION COMMAND SET Table 4

Command	Description	Protocol	1-Wire Bus Activity After Command is Issued	Notes
TEMPERATURE CONVERSION COMMANDS				
Convert T	Initiates temperature conversion.	44h	DS18S20 transmits conversion status to master (not applicable for parasite-powered DS18S20s).	1
MEMORY COMMANDS				
Read Scratchpad	Reads the entire scratchpad including the CRC byte.	BEh	DS18S20 transmits up to 9 data bytes to master.	2
Write Scratchpad	Writes data into scratchpad bytes 2 and 3 (T_H and T_L).	4Eh	Master transmits 2 data bytes to DS18S20.	3
Copy Scratchpad	Copies T_H and T_L data from the scratchpad to EEPROM.	48h	None	1
Recall E^2	Recalls T_H and T_L data from EEPROM to the scratchpad.	B8h	DS18S20 transmits recall status to master.	
Read Power Supply	Signals DS18S20 power supply mode to the master.	B4h	DS18S20 transmits supply status to master.	

NOTES:

1) For parasite-powered DS18S20s, the master must enable a strong pullup on the 1-Wire bus during temperature conversions and copies from the scratchpad to EEPROM. No other bus activity may take place during this time.
2) The master can interrupt the transmission of data at any time by issuing a reset.
3) Both bytes must be written before a reset is issued.

1-WIRE SIGNALING

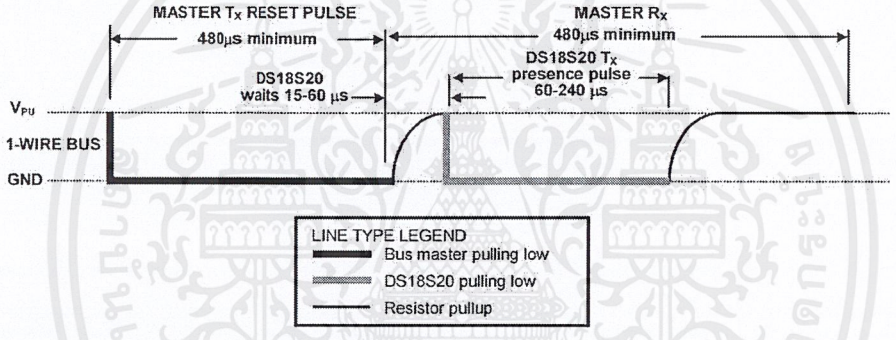
The DS18S20 uses a strict 1-Wire communication protocol to insure data integrity. Several signal types are defined by this protocol: reset pulse, presence pulse, write 0, write 1, read 0, and read 1. All of these signals, with the exception of the presence pulse, are initiated by the bus master.

INITIALIZATION PROCEDURE: RESET AND PRESENCE PULSES

All communication with the DS18S20 begins with an initialization sequence that consists of a reset pulse from the master followed by a presence pulse from the DS18S20. This is illustrated in Figure 10. When the DS18S20 sends the presence pulse in response to the reset, it is indicating to the master that it is on the bus and ready to operate.

During the initialization sequence the bus master transmits (T_X) the reset pulse by pulling the 1-Wire bus low for a minimum of 480µs. The bus master then releases the bus and goes into receive mode (R_X). When the bus is released, the 5k pullup resistor pulls the 1-Wire bus high. When the DS18S20 detects this rising edge, it waits 15µs to 60µs and then transmits a presence pulse by pulling the 1-Wire bus low for 60µs to 240µs.

INITIALIZATION TIMING Figure 10



READ/WRITE TIME SLOTS

The bus master writes data to the DS18S20 during write time slots and reads data from the DS18S20 during read-time slots. One bit of data is transmitted over the 1-Wire bus per time slot.

WRITE TIME SLOTS

There are two types of write time slots: "Write 1" time slots and "Write 0" time slots. The bus master uses a Write 1 time slot to write a logic 1 to the DS18S20 and a Write 0 time slot to write a logic 0 to the DS18S20. All write time slots must be a minimum of 60µs in duration with a minimum of a 1µs recovery time between individual write slots. Both types of write time slots are initiated by the master pulling the 1-Wire bus low (see Figure 11).

To generate a Write 1 time slot, after pulling the 1-Wire bus low, the bus master must release the 1-Wire bus within 15µs. When the bus is released, the 5k pullup resistor will pull the bus high. To generate a Write 0 time slot, after pulling the 1-Wire bus low, the bus master must continue to hold the bus low for the duration of the time slot (at least 60µs). The DS18S20 samples the 1-Wire bus during a window that lasts from 15µs to 60µs after the master initiates the write time slot. If the bus is high during the sampling window, a 1 is written to the DS18S20. If the line is low, a 0 is written to the DS18S20.

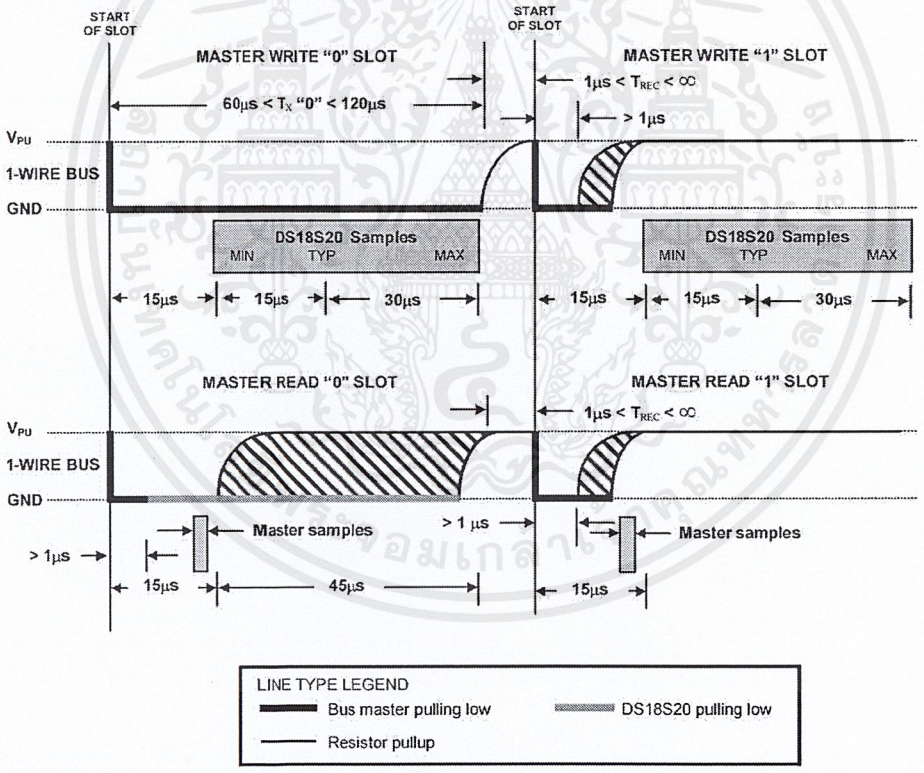
READ-TIME SLOTS

The DS18S20 can only transmit data to the master when the master issues read-time slots. Therefore, the master must generate read-time slots immediately after issuing a Read Scratchpad [BEh] or Read Power Supply [B4h] command, so that the DS18S20 can provide the requested data. In addition, the master can generate read-time slots after issuing Convert T [44h] or Recall E^2 [B8h] commands to find out the status of the operation as explained in the DS18S20 FUNCTION COMMAND section.

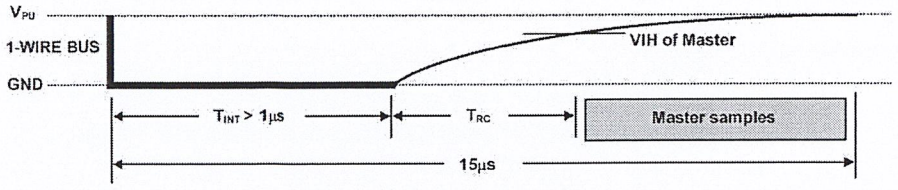
All read-time slots must be a minimum of 60µs in duration with a minimum of a 1µs recovery time between slots. A read-time slot is initiated by the master device pulling the 1-Wire bus low for a minimum of 1µs and then releasing the bus (see Figure 11). After the master initiates the read-time slot, the DS18S20 will begin transmitting a 1 or 0 on bus. The DS18S20 transmits a 1 by leaving the bus high and transmits a 0 by pulling the bus low. When transmitting a 0, the DS18S20 will release the bus by the end of the time slot, and the bus will be pulled back to its high idle state by the pullup resister. Output data from the DS18S20 is valid for 15µs after the falling edge that initiated the read-time slot. Therefore, the master must release the bus and then sample the bus state within 15µs from the start of the slot.

Figure 12 illustrates that the sum of T_{INIT}, T_{RC}, and T_{SAMPLE} must be less than 15µs for a read-time slot. Figure 13 shows that system timing margin is maximized by keeping T_{INIT} and T_{RC} as short as possible and by locating the master sample time during read-time slots towards the end of the 15µs period.

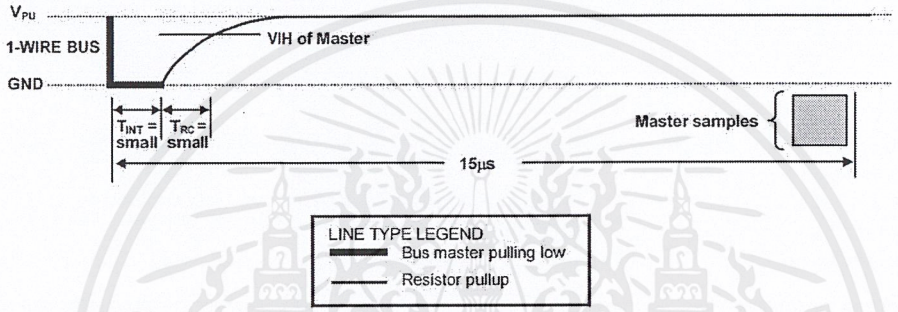
READ/WRITE TIME SLOT TIMING DIAGRAM Figure 11



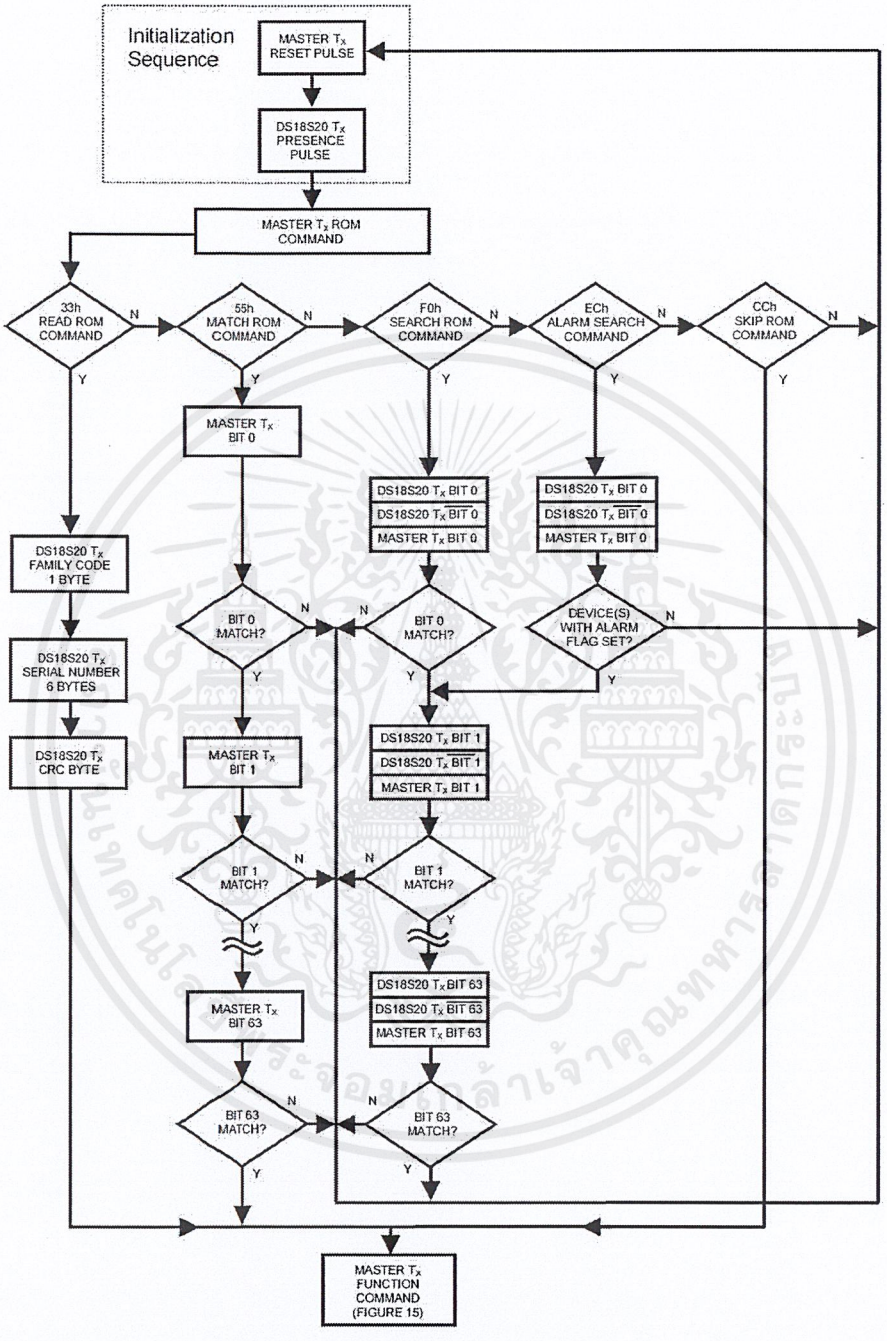
DETAILED MASTER READ 1 TIMING Figure 12



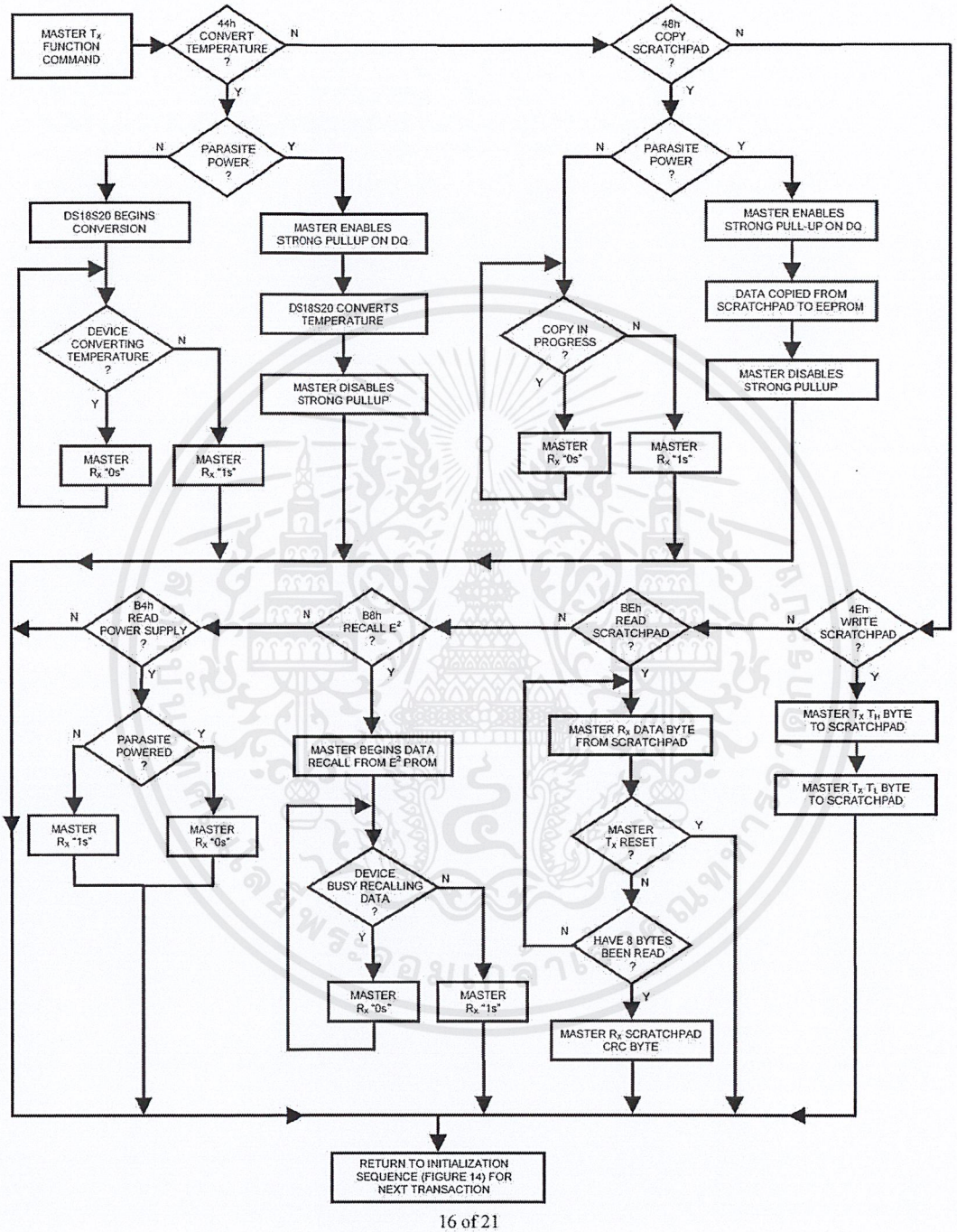
RECOMMENDED MASTER READ 1 TIMING Figure 13



ROM COMMANDS FLOW CHART Figure 14



DS18S20 FUNCTION COMMANDS FLOW CHART Figure 15



DS18S20 OPERATION EXAMPLE 1

In this example there are multiple DS18S20s on the bus and they are using parasite power. The bus master initiates a temperature conversion in a specific DS18S20 and then reads its scratchpad and recalculates the CRC to verify the data.

MASTER MODE	DATA (LSB FIRST)	COMMENTS
TX	Reset	Master issues reset pulse.
RX	Presence	DS18S20s respond with presence pulse.
TX	55h	Master issues Match ROM command.
TX	64-bit ROM code	Master sends DS18S20 ROM code.
TX	44h	Master issues Convert T command.
TX	DQ line held high by strong pullup	Master applies strong pullup to DQ for the duration of the conversion (t_{conv}).
TX	Reset	Master issues reset pulse.
RX	Presence	DS18S20s respond with presence pulse.
TX	55h	Master issues Match ROM command.
TX	64-bit ROM code	Master sends DS18S20 ROM code.
TX	BEh	Master issues Read Scratchpad command.
RX	9 data bytes	Master reads entire scratchpad including CRC. The master then recalculates the CRC of the first eight data bytes from the scratchpad and compares the calculated CRC with the read CRC (byte 9). If they match, the master continues; if not, the read operation is repeated.

DS18S20 OPERATION EXAMPLE 2

In this example there is only one DS18S20 on the bus and it is using parasite power. The master writes to the T_H and T_L registers in the DS18S20 scratchpad and then reads the scratchpad and recalculates the CRC to verify the data. The master then copies the scratchpad contents to EEPROM.

MASTER MODE	DATA (LSB FIRST)	COMMENTS
TX	Reset	Master issues reset pulse.
RX	Presence	DS18S20 responds with presence pulse.
TX	CCh	Master issues Skip ROM command.
TX	4Eh	Master issues Write Scratchpad command.
TX	2 data bytes	Master sends two data bytes to scratchpad (T_H and T_L)
TX	Reset	Master issues reset pulse.
RX	Presence	DS18S20 responds with presence pulse.
TX	CCh	Master issues Skip ROM command.
TX	BEh	Master issues Read Scratchpad command.
RX	9 data bytes	Master reads entire scratchpad including CRC. The master then recalculates the CRC of the first eight data bytes from the scratchpad and compares the calculated CRC with the read CRC (byte 9). If they match, the master continues; if not, the read operation is repeated.
TX	Reset	Master issues reset pulse.
RX	Presence	DS18S20 responds with presence pulse.
TX	CCh	Master issues Skip ROM command.
TX	48h	Master issues Copy Scratchpad command.
TX	DQ line held high by strong pullup	Master applies strong pullup to DQ for at least 10ms while copy operation is in progress.

DS18S20 OPERATION EXAMPLE 3

In this example there is only one DS18S20 on the bus and it is using parasite power. The bus master initiates a temperature conversion then reads the DS18S20 scratchpad and calculates a higher resolution result using the data from the temperature, COUNT REMAIN and COUNT PER °C registers.

MASTER MODE	DATA (LSB FIRST)	COMMENTS
TX	Reset	Master issues reset pulse.
TR	Presence	DS18S20 responds with presence pulse.
TX	CCh	Master issues Skip ROM command.
TX	44h	Master issues Convert T command.
TX	DQ line held high by strong pullup	Master applies strong pullup to DQ for the duration of the conversion (t_{conv}).
TX	Reset	Master issues reset pulse.
RX	Presence	DS18S20 responds with presence pulse.
TX	CCh	Master issues Skip ROM command.
TX	BEh	Master issues Read Scratchpad command.
RX	9 data bytes	Master reads entire scratchpad including CRC. The master then recalculates the CRC of the first eight data bytes from the scratchpad and compares the calculated CRC with the read CRC (byte 9). If they match, the master continues; if not, the read operation is repeated. The master also calculates the TEMP_READ value and stores the contents of the COUNT REMAIN and COUNT PER °C registers.
TX	Reset	Master issues reset pulse.
RX	Presence	DS18S20 responds with presence pulse.
-	-	CPU calculates extended resolution temperature using the equation in the *OPERATION — MEASURING TEMPERATURE* section of this datasheet.

RELATED APPLICATION NOTES

The following Application Notes can be applied to the DS18S20. These notes can be obtained from the Dallas Semiconductor "Application Note Book," via the Dallas website at http://www.dalsemi.com/, or through our faxback service at (214) 450–0441.

Application Note 27: Understanding and Using Cyclic Redundancy Checks with Dallas Semiconductor Touch Memory Product
Application Note 55: Extending the Contact Range of Touch Memories
Application Note 74: Reading and Writing Touch Memories via Serial Interfaces
Application Note 104: Minimalist Temperature Control Demo
Application Note 105: High-Resolution Temperature Measurement with Dallas Direct-to-Digital Temperature Sensors
Application Note 106: Complex MicroLANs
Application Note 108: MicroLAN — In the Long Run
Application Note 162: Interfacing the DS18X20/DS1822 1-Wire Temperature Sensor in a Microcontroller Environment

Sample 1-Wire subroutines that can be used in conjunction with AN74 can be downloaded from the Dallas website or anonymous FTP Site.

ABSOLUTE MAXIMUM RATINGS*

Voltage on Any Pin Relative to Ground	-0.5V to +6.0V
Operating Temperature Range	-55°C to +125°C
Storage Temperature Range	-55°C to +125°C
Solder Temperature	See IPC/JEDEC J-STD-020A
Reflow Oven Temperature	+220°C

**These are stress ratings only and functional operation of the device at these or any other conditions above those indicated in the operation sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods of time may affect reliability.*

DC ELECTRICAL CHARACTERISTICS (-55°C to +125°C; V_{DD} = 3.0V to 5.5V)

PARAMETER	SYMBOL	CONDITION	MIN	TYP	MAX	UNITS	NOTES
Supply Voltage	V_{DD}	Local Power	+3.0		+5.5	V	1
Pullup Supply Voltage	V_{PU}	Parasite Power	+3.0		+5.5	V	1, 2
		Local Power	+3.0		V_{DD}		
Thermometer Error	t_{ERR}	-10°C to +85°C			±0.5	°C	3
		-55°C to +125°C			±2		
Input Logic Low	V_{IL}		-0.3		+0.8	V	1, 4, 5
Input Logic High	V_{IH}	Local Power	+2.2		The lower of 5.5 or V_{DD} + 0.3	V	1, 6
		Parasite Power	+3.0				
Sink Current	I_L	$V_{I/O}$=0.4V	4.0			mA	1
Standby Current	I_{DDS}			750	1000	nA	7, 8
Active Current	I_{DD}	V_{DD}=5V		1	1.5	mA	9
DQ Input Current	I_{DQ}			5		µA	10
Drift				±0.2		°C	11

NOTES:

1) All voltages are referenced to ground.
2) The Pullup Supply Voltage specification assumes that the pullup device is ideal, and therefore the high level of the pullup is equal to V_{PU}. In order to meet the V_{IH} spec of the DS18S20, the actual supply rail for the strong pullup transistor must include margin for the voltage drop across the transistor when it is turned on; thus: $V_{PU\_ACTUAL} = V_{PU\_IDEAL} + V_{TRANSISTOR}$.
3) See typical performance curve in Figure 16
4) Logic low voltages are specified at a sink current of 4mA.
5) To guarantee a presence pulse under low voltage parasite power conditions, V_{ILMAX} may have to be reduced to as low as 0.5V.
6) Logic high voltages are specified at a source current of 1mA.
7) Standby current specified up to 70°C. Standby current typically is 3µA at 125°C.
8) To minimize I_{DDS}, DQ should be within the following ranges: GND ≤ DQ ≤ GND + 0.3V or V_{DD} – 0.3V ≤ DQ ≤ V_{DD}.
9) Active current refers to supply current during active temperature conversions or EEPROM writes.
10) DQ line is high ("hi-Z" state).
11) Drift data is based on a 1000 hour stress test at 125°C with V_{DD} = 5.5V.

AC ELECTRICAL CHARACTERISTICS: NV MEMORY

(-55°C to +100°C; V_{DD} = 3.0V to 5.5V)

PARAMETER	SYMBOL	CONDITION	MIN	TYP	MAX	UNITS
NV Write Cycle Time	t_{WR}			2	10	ms
EEPROM Writes	N_{EEWR}	-55°C to +55°C	50k			writes
EEPROM Data Retention	t_{EEDR}	-55°C to +55°C	10			years

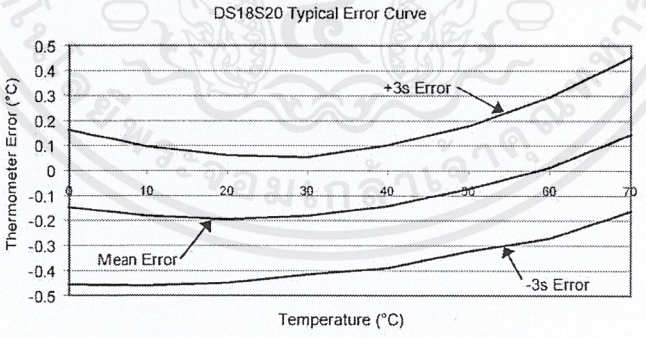
AC ELECTRICAL CHARACTERISTICS (-55°C to +125°C; V_{DD} = 3.0V to 5.5V)

PARAMETER	SYMBOL	CONDITION	MIN	TYP	MAX	UNITS	NOTES
Temperature Conversion Time	t_{CONV}				750	ms	1
Time to Strong Pullup On	t_{SPON}	Start Convert T Command Issued			10	µs	
Time Slot	t_{SLOT}		60		120	µs	1
Recovery Time	t_{REC}		1			µs	1
Write 0 Low Time	t_{LOW0}		60		120	µs	1
Write 1 Low Time	t_{LOW1}		1		15	µs	1
Read Data Valid	t_{RDV}				15	µs	1
Reset Time High	t_{RSTH}		480			µs	1
Reset Time Low	t_{RSTL}		480			µs	1, 2
Presence Detect High	t_{PDHIGH}		15		60	µs	1
Presence Detect Low	t_{PDLOW}		60		240	µs	1
Capacitance	$C_{IN/OUT}$				25	pF	

NOTES:

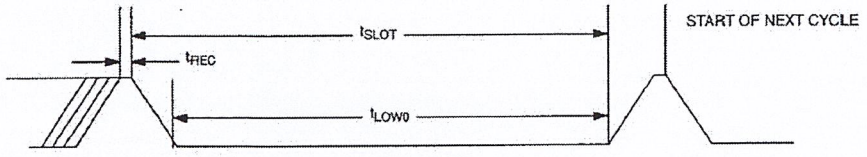
1) Refer to timing diagrams in Figure 17.
2) Under parasite power, if t_{RSTL} > 960µs, a power on reset may occur.

TYPICAL PERFORMANCE CURVE Figure 16

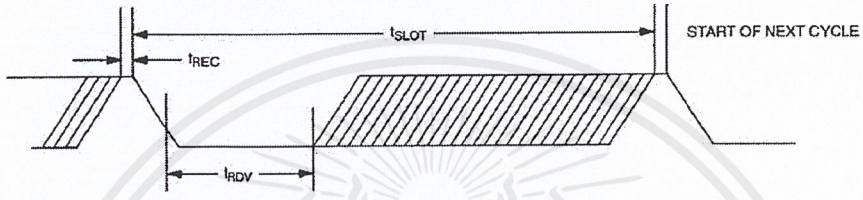


TIMING DIAGRAMS Figure 17

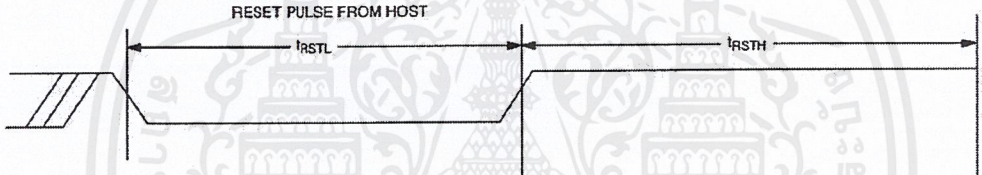
1–WIRE WRITE ZERO TIME SLOT



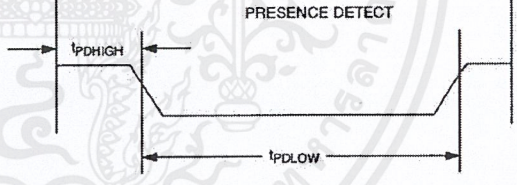
1–WIRE READ ZERO TIME SLOT



1–WIRE RESET PULSE



1–WIRE PRESENCE DETECT

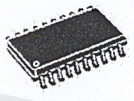


SGS-THOMSON MICROELECTRONICS

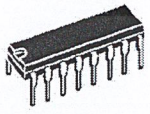
L293D
L293DD

PUSH-PULL FOUR CHANNEL DRIVER WITH DIODES

- 600mA OUTPUT CURRENT CAPABILITY PER CHANNEL
- 1.2A PEAK OUTPUT CURRENT (non repetitive) PER CHANNEL
- ENABLE FACILITY
- OVERTEMPERATURE PROTECTION
- LOGICAL "0" INPUT VOLTAGE UP TO 1.5 V (HIGH NOISE IMMUNITY)
- INTERNAL CLAMP DIODES



SO(12+4+4)



Powerdip (12+2+2)

ORDERING NUMBERS:

L293DD L293D

DESCRIPTION

The Device is a monolithic integrated high voltage, high current four channel driver designed to accept standard DTL or TTL logic levels and drive inductive loads (such as relays solenoides, DC and stepping motors) and switching power transistors.

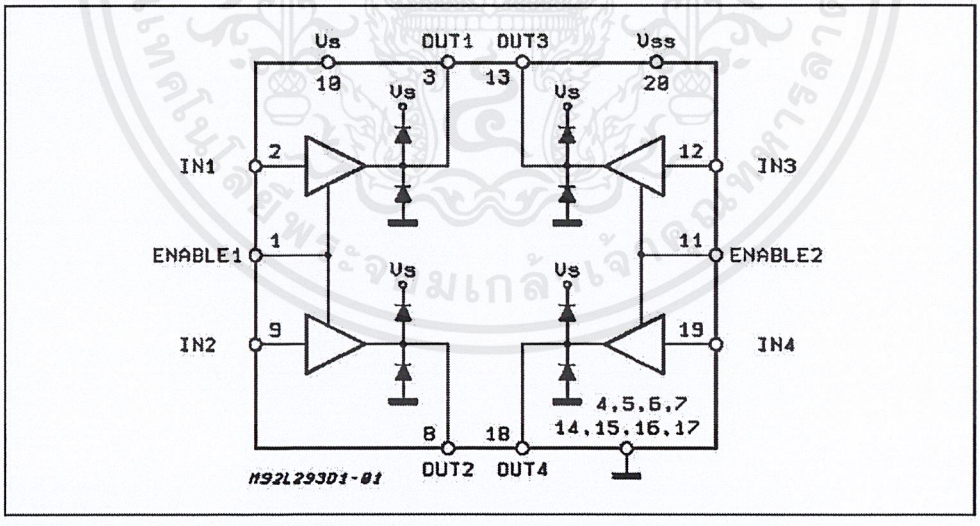
To simplify use as two bridges each pair of channels is equipped with an enable input. A separate supply input is provided for the logic, allowing operation at a lower voltage and internal clamp diodes are included.

This device is suitable for use in switching applications at frequencies up to 5 kHz.

The L293D is assembled in a 16 lead plastic packaage which has 4 center pins connected together and used for heatsinking

The L293DD is assembled in a 20 lead surface mount which has 8 center pins connected together and used for heatsinking.

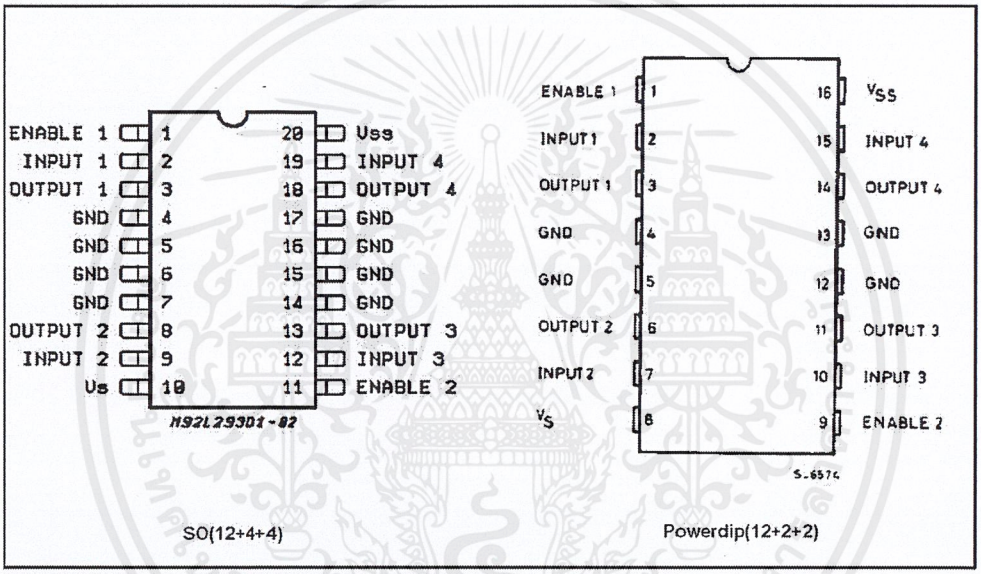
BLOCK DIAGRAM



ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_S	Supply Voltage	36	V
V_{SS}	Logic Supply Voltage	36	V
V_i	Input Voltage	7	V
V_{en}	Enable Voltage	7	V
I_o	Peak Output Current (100 µs non repetitive)	1.2	A
P_{tot}	Total Power Dissipation at T_{pins} = 90 °C	4	W
T_{stg}, T_j	Storage and Junction Temperature	– 40 to 150	°C

PIN CONNECTIONS (Top view)



THERMAL DATA

Symbol	Decription		DIP	SO	Unit
$R_{th\ j\text{-}pins}$	Thermal Resistance Junction-pins	max.	–	14	°C/W
$R_{th\ j\text{-}amb}$	Thermal Resistance junction-ambient	max.	80	50 (*)	°C/W
$R_{th\ j\text{-}case}$	Thermal Resistance Junction-case	max.	14	–	

(*) With 6sq. cm on board heatsink.

ELECTRICAL CHARACTERISTICS (for each channel, V_S = 24 V, V_{SS} = 5 V, T_{amb} = 25 °C, unless otherwise specified)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_S	Supply Voltage (pin 10)		V_{SS}		36	V
V_{SS}	Logic Supply Voltage (pin 20)		4.5		36	V
I_S	Total Quiescent Supply Current (pin 10)	V_i = L ; I_O = 0 ; V_{en} = H		2	6	mA
		V_i = H ; I_O = 0 ; V_{en} = H		16	24	mA
		V_{en} = L			4	mA
I_{SS}	Total Quiescent Logic Supply Current (pin 20)	V_i = L ; I_O = 0 ; V_{en} = H		44	60	mA
		V_i = H ; I_O = 0 ; V_{en} = H		16	22	mA
		V_{en} = L		16	24	mA
V_{IL}	Input Low Voltage (pin 2, 9, 12, 19)		– 0.3		1.5	V
V_{IH}	Input High Voltage (pin 2, 9, 12, 19)	$V_{SS} \leq 7$ V	2.3		V_{SS}	V
		$V_{SS} > 7$ V	2.3		7	V
I_{IL}	Low Voltage Input Current (pin 2, 9, 12, 19)	V_{IL} = 1.5 V			– 10	µA
I_{IH}	High Voltage Input Current (pin 2, 9, 12, 19)	2.3 V ≤ V_{IH} ≤ V_{SS} – 0.6 V		30	100	µA
$V_{en\,L}$	Enable Low Voltage (pin 1, 11)		– 0.3		1.5	V
$V_{en\,H}$	Enable High Voltage (pin 1, 11)	$V_{SS} \leq 7$ V	2.3		V_{SS}	V
		$V_{SS} > 7$ V	2.3		7	V
$I_{en\,L}$	Low Voltage Enable Current (pin 1, 11)	$V_{en\,L}$ = 1.5 V		– 30	– 100	µA
$I_{en\,H}$	High Voltage Enable Current (pin 1, 11)	2.3 V ≤ $V_{en\,H}$ ≤ V_{SS} – 0.6 V			± 10	µA
$V_{CE(sat)H}$	Source Output Saturation Voltage (pins 3, 8, 13, 18)	I_O = – 0.6 A		1.4	1.8	V
$V_{CE(sat)L}$	Sink Output Saturation Voltage (pins 3, 8, 13, 18)	I_O = + 0.6 A		1.2	1.8	V
V_F	Clamp Diode Forward Voltage	I_O = 600nA		1.3		V
t_r	Rise Time (*)	0.1 to 0.9 V_O		250		ns
t_f	Fall Time (*)	0.9 to 0.1 V_O		250		ns
t_{on}	Turn-on Delay (*)	0.5 V_i to 0.5 V_O		750		ns
t_{off}	Turn-off Delay (*)	0.5 V_i to 0.5 V_O		200		ns

(*) See fig. 1.

TRUTH TABLE (one channel)

Input	Enable (*)	Output
H	H	H
L	H	L
H	L	Z
L	L	Z

Z = High output impedance
(*) Relative to the considered channel

Figure 1: Switching Times

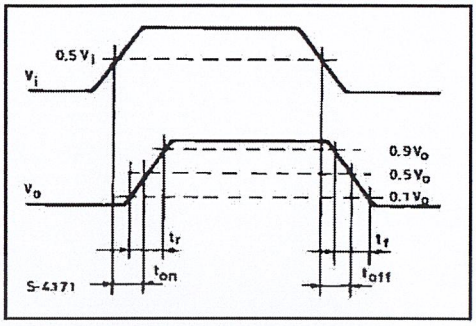
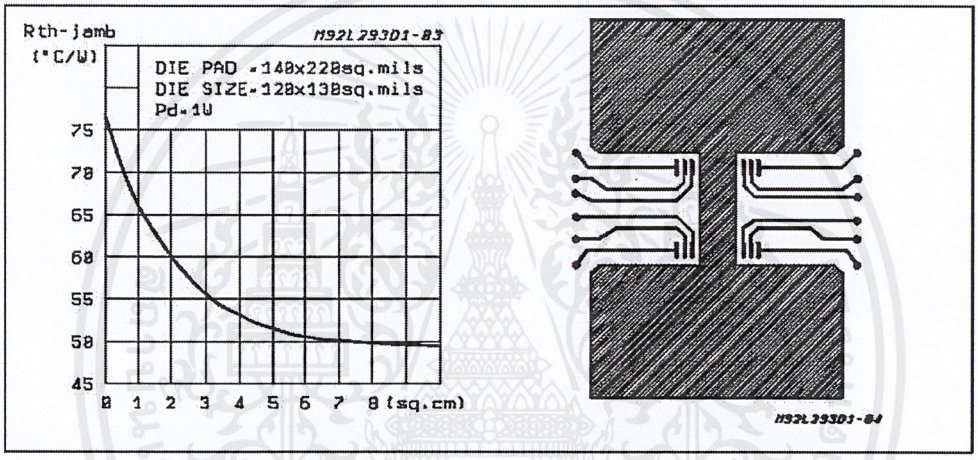
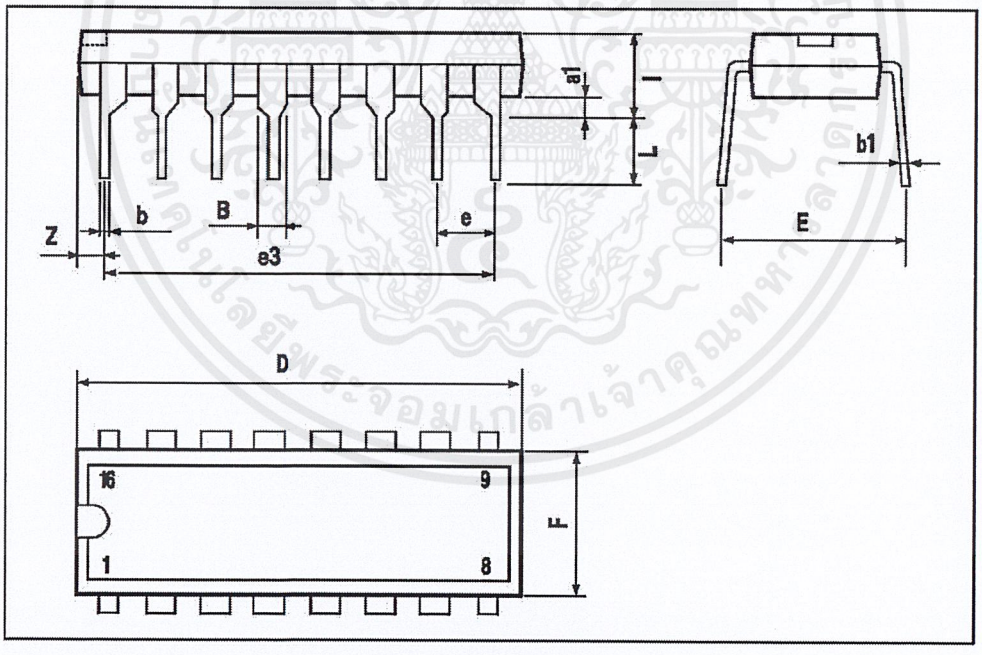


Figure 2: Junction to ambient thermal resistance vs. area on board heatsink (SO12+4+4 package)



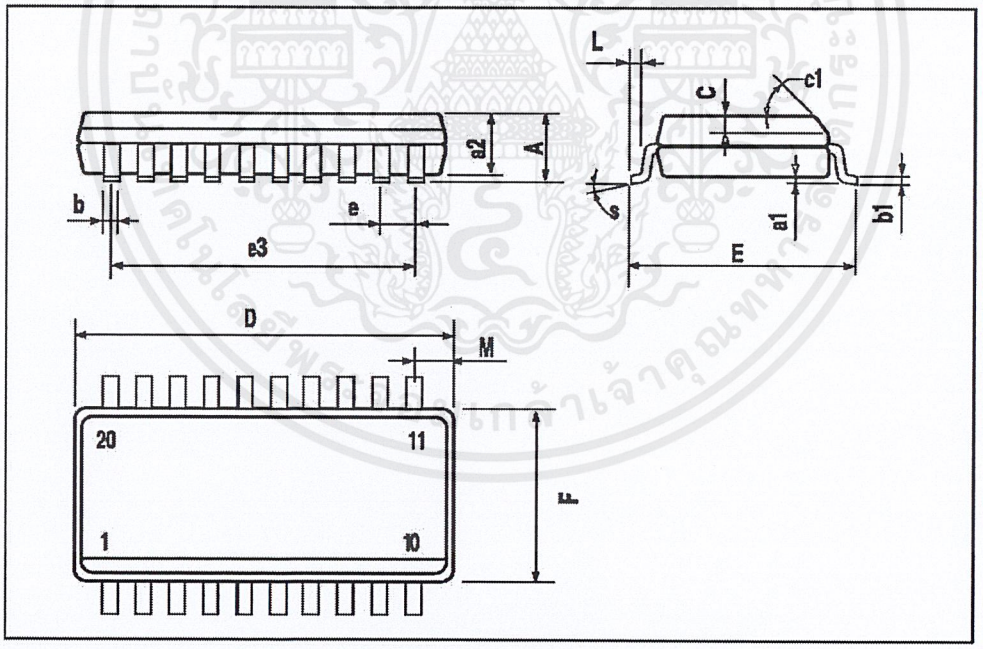
POWERDIP16 PACKAGE MECHANICAL DATA

DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
a1	0.51			0.020		
B	0.85		1.40	0.033		0.055
b		0.50			0.020	
b1	0.38		0.50	0.015		0.020
D			20.0			0.787
E		8.80			0.346	
e		2.54			0.100	
e3		17.78			0.700	
F			7.10			0.280
I			5.10			0.201
L		3.30			0.130	
Z			1.27			0.050



SO20 PACKAGE MECHANICAL DATA

DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
A			2.65			0.104
a1	0.1		0.2	0.004		0.008
a2			2.45			0.096
b	0.35		0.49	0.014		0.019
b1	0.23		0.32	0.009		0.013
C		0.5			0.020	
c1		45			1.772	
D		1	12.6		0.039	0.496
E	10		10.65	0.394		0.419
e		1.27			0.050	
e3		11.43			0.450	
F		1	7.4		0.039	0.291
G	8.8		9.15	0.346		0.360
L	0.5		1.27	0.020		0.050
M			0.75			0.030
S	8° (max.)					



เอกสารอ้างอิง

[1] ผศ.พรสุข รติโรจน์อนันต์. **พื้นฐานการควบคุมกระบวนการ (Fundamentals of Process Control Theory).** กรุงเทพมหานคร : แผนกตำรา คณะวิศวกรรมศาสตร์, สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง. 2549.

[2] ธาริน สิทธิธรรมชารี, ประชา พฤกษ์ประเสริฐ. **Microsoft Visual Basic 6.0.** กรุงเทพมหานคร : บริษัท ซัคเซส มีเดีย จำกัด. 2548.

[3] ประจิน พลังสันติกุล. **เรียนรู้และใช้งาน CCS C คอมไพล์เลอร์ เขียนโปรแกรมภาษา C ควบคุมไมโครคอนโทรลเลอร์ PIC.** พิมพ์ครั้งที่1. กรุงเทพมหานคร : อินโนเวตีฟ เอ็กเพอริเมนต์. 2547.