สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญตาราง

สารบัญรูป

สารบัญรูป (ต่อ)

บทที่ 1
บทนำ

1.1 ความเป็นมาของโครงงาน

เนื่องจากปัจจุบันอุณหภูมิตามสถานที่ ณ จุดต่างๆ มีการเปลี่ยนแปลงทำให้เกิดการเปลี่ยนแปลงของค่าอุณหภูมิ อยู่ตลอดเวลาการควบคุมจึงเป็นเรื่องที่สำคัญ แต่สิ่งที่สำคัญที่สุดคือเรื่องของการทราบค่าการเปลี่ยนแปลงอุณหภูมิ ดังนั้นการเฝ้าระวังจึงเป็นทางออกอีกทางหนึ่ง ฉะนั้นตัววัดอุณหภูมิ จะต้องมีการรายงานผลที่เที่ยงตรง และสม่ำเสมอ เพื่อที่จะทำให้เราทราบค่าการเปลี่ยนแปลงของอุณหภูมิและเมื่อเกิดปัญหาก็จะแก้ปัญหาได้ทันการ

โดยปกติจะใช้คนในการตรวจเช็คอุณหภูมิ ซึ่งจะทำให้เสียเวลาในการตรวจ ณ จุดต่างๆ คณะผู้จัดทำจึงมีแรงจูงใจที่จะเสนอ เครื่องวัดอุณหภูมิแบบไร้สายที่สามารถวัดจุดต่างๆ ได้หลายจุดและระยะไม่เกิน 50 เมตร แทนที่ระบบสายที่ไม่สามารถเคลื่อนย้ายจุดที่ไม่ต้องการวัด และยุ่งยากในการติดตั้งรวมไปถึงการลงทุนที่สุงกว่า ในการส่งข้อมูลระบบไร้สายจึงเหมาะสมที่สุด โดยข้อมูลค่า อุณหภูมิที่ได้จะถูกนำมา มอดูเลต และส่งออกผ่านตัวกลาง คลื่นวิทยุในย่านความถี่ UHF ไปยังเครื่อง รับเพื่อทำการถอดรหัสข้อมูลที่ส่งมาและแสดงผลบนคอมพิวเตอร์ ด้วยโปรแกรมภาษา Visual Basic เนื่องจากจะแสดงค่าได้ต่อเนื่อง และสามรถ ตั้งค่าเวลาในการแสดงค่าได้ เพื่อที่จะนำไปวัดค่าอุณหภูมิในห้องServer

1.2 วัตถุประสงค์ของโครงงาน

1.2.1 เพื่อศึกษาการทำงานของตัวตรวจวัดค่าอุณหภูมิ

1.2.2 เพื่อศึกษาการเขียนโปรแกรม Microcontroller

1.2.2 เพื่อศึกษาการรับ-ส่งข้อมูลของ Module ไร้สาย

1.2.3 เพื่อศึกษาโปรแกรม Visual Basic

1.3 ขอบเขตของโครงงาน

1.3.1 ส่งค่าผลการตรวจวัดอุณหภูมิ มากกว่า 1 จุด ระยะไม่เกิน 50 เมตร

1.3.2 ใช้ Module ในการรับส่งข้อมูลไร้สายในคลื่นวิทยุ

1.3.3 นำค่าอุณหภูมิที่ได้มาแสดงโดยใช้โปรแกรม Visual Basic

1.3.4 สามารถตั้งค่าเวลาในการแสดงผลและเก็บบันทึกค่าลงใน Harddisk

1.4 ขั้นตอนและวิธีการดำเนินงาน

1.4.1 วางแผนงานและกำหนดขอบเขตของโครงงาน

1.4.2 ศึกษาทฤษฎีการทำงานของการรับส่งข้อมูลไร้สาย

1.4.3 ออกแบบและสร้างวงจรตรวจวัดอุณหภูมิ ส่งค่าผ่าน Module ไร้สาย

1.4.4 แสดงผลการทดลอง ด้วย Visual Basic และบันทึกผล

1.4.5 สรุปผลของโครงงานและข้อเสนอแนะ

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1.5.1 นำเครื่องตรวจวัดอุณหภูมิ ที่ได้แทนระบบสายและใช้แทนคนในการตรวจวัด

1.5.2 นำ Microcontroller มาใช้ในการควบคุมการส่งค่าของ ตัววัดอุณหภูมิ

1.5.3 มีความเที่ยงและแม่นยำในการแสดงค่าอุณหภูมิ

1.5.4.สามารถแสดงผลการตรวจและเก็บข้อมูลได้ถี่ขึ้น

1.5.5.สามารถย้ายจุดในการตรวจวัดได้

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

ในบทนี้จะกล่าวถึง การนำเอาทฤษฎีที่เกี่ยวข้องมาใช้ในการสร้างเครื่องวัดอุณหภูมิ ไร้สายเนื่องจากในการสร้างจะต้องมีองค์ประกอบหลาย ๆ ด้านด้วยกัน ดังนั้นจำเป็นจะต้องมีการอ้างอิงถึงทฤษฎีที่เกี่ยวข้อง เพื่อบอกถึงเหตุผลที่นำมาใช้ในการออกแบบ และสร้าง โดยเครื่องวัดอุณหภูมิไร้สายมีทฤษฎีที่เกี่ยวข้อง ดังนี้

2.1 ตัวตรวจวัดอุณหภูมิ

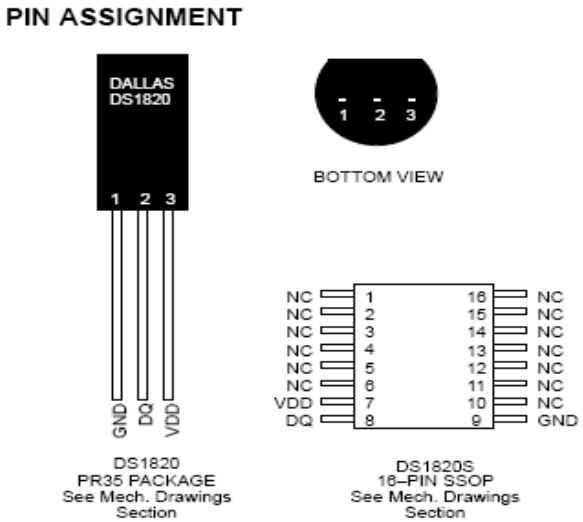
2.2 การเชื่อมต่ออุปกรณ์แบบ 1–Wire Bus

2.3 การใช้โปรแกรมควบคุมไมโครคอนโทรลเลอร์

2.4 โมดูลเครื่องรับ-ส่งไร้สาย

2.5 การพัฒนาโปรแกรม Visual Basic

2.1 ตัวตรวจวัดอุณหภูมิ DS1820



รูปที่ 2.1 ตัวตรวจวัดอุณหภูมิ DS1820

2.1.1 หลักการเบื้องต้นของไอซี DS1820

ไอซี DS1820 เป็นไอซีที่มีระบบการสื่อสารข้อมูลอนุกรมแบบหนึ่งสายซึ่งถือได้ว่าเป็นระบบที่มีความชาญฉลาด และใช้จำนวนสายสัญญาณเพียง 1 เส้นเท่านั้น โดยไม่ต้องมีสายสัญญานาฬิกามาควบคุมจังหวะการถ่ายทอดข้อมูลเหมือนกับระบบสื่อสารข้อมูลอนุกรมในแบบอื่น สายข้อมูลจะทำหน้าที่เสมือนเป็นสายสัญญาณนาฬิกาในตัว ส่วนค่าของข้อมูลจะพิจารณาจากลักษณะของรูปสัญญาณที่ปรากฏบนสายสัญญาณในแต่ละช่องของเวลาซึ่งเรียกว่า ไทม์สล็อต (Time Slot) โดยคาบเวลาต่ำสุดและสูงสุดของสถานะต่าง ๆ ในการสื่อสารข้อมูลในแต่ละไทม์สล็อตมีการกำหนดขอบเขตไว้อย่างชัดเจนการถ่ายทอดข้อมูลจะเกิดขึ้นในแต่ละไทม์สล็อตนั้น รูปแบบการถ่ายทอด ข้อมูลจะเป็นแบบอะซิงโครนัสในระดับบิต ไม่มีการกำหนดความยาวของข้อมูลเป็นระดับไบต์ระบบสื่อสารแบบนี้เหมาะที่จะใช้ในการสื่อสารข้อมูลระหว่างไอซีแผงวงจรเดียวกัน

2.1.2 การอินเตอร์เฟสผ่านสายเส้นเดียว

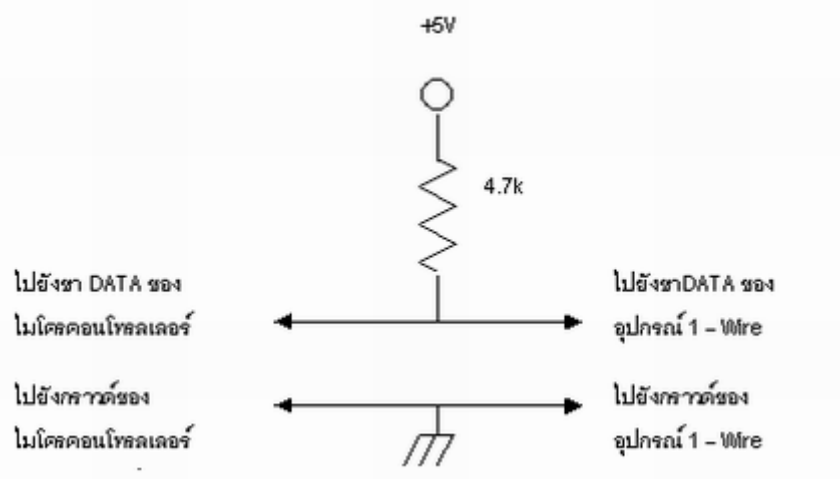
การเชื่อมต่อหรือการอินเตอร์เฟส (Interface) ระหว่างไมโครคอนโทรลเลอร์กับอุปกรณ์ภายนอก โดยใช้จำนวนสายสัญญาณให้น้อยที่สุดได้มีการพัฒนามาอย่างต่อเนื่องจากหลายบริษัทผู้ผลิต เช่น การเชื่อมต่ออุปกรณ์ต่อพ่วงแบบอนุกรม (Serial Peripheral Interface, SPI) ในไมโครคอนโทรลเลอร์ 68HC1 ของโมโตโรลา การเชื่อมต่อแบบ SPI นี้ช่วยให้ ไมโครคอนโทรลเลอร์แลกเปลี่ยนข้อมูลกับอุปกรณ์ต่อพ่วงได้ด้วยความเร็วถึง 1 ล้านบิตต่อวินาที โดยใช้สายรับส่งสัญญาณเพียง 3 หรือ 4 เส้น รวมกับสายกราวด์อีกหนึ่งเส้น

2.1.3 คุณสมบัติของ DS1820

- DS1820 สามารถ Interface โดยใช้สายสัญญาณเพียงเส้นเดียว
- DS1820 เพียงตัวเดียว สามารถวัดอุณหภูมิได้โดยไม่ต้องต่ออุปกรณ์ร่วม
- DS1820 มีย่านวัดอยู่ที่ +125 ถึง -55 C
- DS1820 มีความละเอียดในการวัดได้ 0.5 C

2.2 การเชื่อมต่ออุปกรณ์แบบ 1–Wire Bus

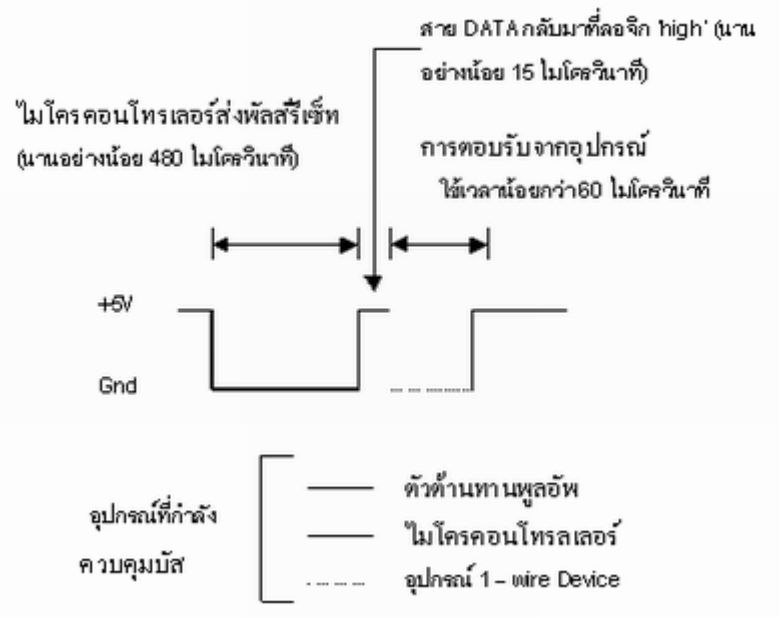
อุปกรณ์ที่สนับสนุนระบบบัสเพียงเส้นเดียวจะมีสายสัญญาณเพียง 2 เส้นเท่านั้นคือสายกราวด์และสายสัญญาณ ซึ่งเรียกอีกอย่างว่า สาย DATA สายนี้จะจัดการเกี่ยวกับทั้งสัญญาณข้อมูลและสัญญาณนาฬิกาที่ใช้ในการแลกเปลี่ยนข้อมูล สาย DATA นี้จะเป็นชนิด Open Drain ดังนั้นในการออกแบบวงจร จะต้องออกแบบให้มีตัวต้านทานมาพูลอัพสาย DATA นี้ด้วยให้ดูรูปแผนผังแสดงการต่อระบบบัสของ 1–Wire Bus ดังรูปที่ 2.2



รูปที่2.2 แผนผังของระบบบัสแบบ 1-Wire Bus

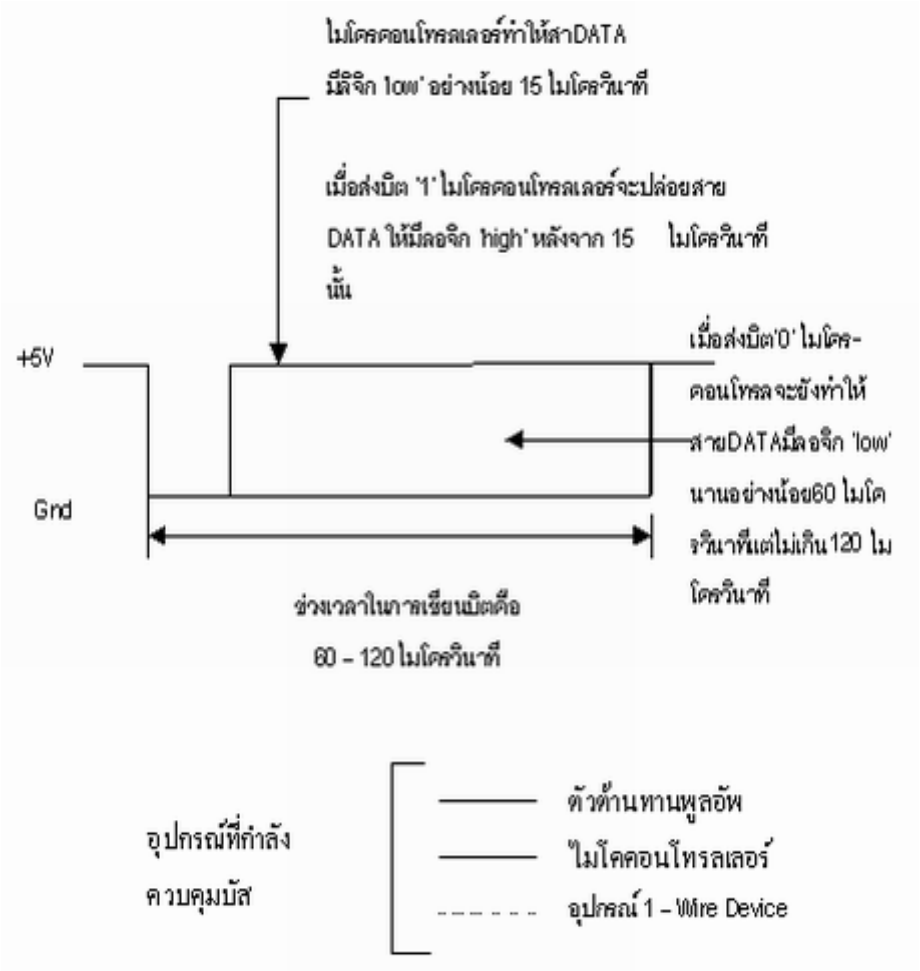
จากการแลกเปลี่ยนข้อมูลระหว่างไมโครคอนโทรลเลอร์กับอุปกรณ์ที่ใช้ 1–Wire Bus นี้ไม่ได้ง่าย ๆ เหมือนกับการส่งข้อมูลผ่านทางบัสแบบ SPI เพราะในระบบ 1–Wire Bus นั้นสาย DATA จะต้องจัดการเกี่ยวกับจังหวะเวลา (Timing) ระดับสัญญาณ (Level) และทิศทาง (Direction)ของข้อมูลทั้งหมด ทำให้การเขียนซอฟท์แวร์ของไมโครคอนโทรลเลอร์ที่ติดต่อกับอุปกรณ์พวกนี้ต้องมีความซับซ้อนมากขึ้น ในสภาวะพัก (Quiescent State) อุปกรณ์ที่ใช้บัสแบบ 1–Wire Bus จะทำให้สาย DATA อยู่ในสภาวะลอย (float) ทำให้ขานี้มีแรงดันเท่ากับแรงดันพูลอัพ (Vcc) ซึ่งปกติก็คือ 5 โวลต์นั้นเอง ส่วนไมโครคอนโทรลเลอร์ก็จะปล่อยให้ขาเอาต์พุตที่ใช้ติดต่อกับขา DATA นี้อยู่ในสภาวะ ความต้านทานสูง (High-Impedance State) เช่นกัน เมื่ออุปกรณ์ทั้งสองชนิดนี้ปล่อยให้ขา DATA อยู่ในสภาวะลอยแล้วนี้ความต้านทานพูลอัพก็จะช่วยรักษาระดับแรงดันไฟเลี้ยงที่จ่ายให้กับอุปกรณ์ที่ใช้บัสแบบ 1–Wire Bus นี้ได้อย่างสม่ำเสมอเพราะอุปกรณ์ที่ใช้บัสแบบ 1–Wire Bus นี้จะใช้พลังงานในการทำงานน้อยมาก ในการแลกเปลี่ยนข้อมูลกันไมโครคอนโทรลเลอร์และอุปกรณ์ที่ใช้บัสแบบ 1–Wire Bus จะต้องดำเนินการอย่างระมัดระวังตามลำดับขั้นตอนในการทำให้สาย DATA มีลอจิกเป็น ’low’ ปล่อยให้สาย DATA กลับมามีลอจิกเป็น ‘high’ และตรวจจับการตอบรับกับอุปกรณ์อีกด้านหนึ่ง ช่วงจังหวะเวลาที่ใช้ในกระบวนการนี้จะถูกกำหนดโดยข้อกำหนดเฉพาะของระบบ 1–Wire Bus นี้ซึ่งต้องใช้ไมโครคอนโทรลเลอร์ที่มีการตอบสนองได้อย่างรวดเร็วด้วย ดังนั้นเราจะต้องตรวจสอบความสามารถของระบบให้ดีเสียก่อนที่จะใช้ระบบบัสแบบ 1–Wire Bus นี้ จากสภาวะพักข้างต้นการแลกเปลี่ยนข้อมูลจะเริ่มขึ้นด้วยการที่ไมโครคอนโทรลเลอร์กระทำกระบวนการรีเซ็ต (Reset Sequence) ซึ่งทำได้โดยการทำให้สาย

DATA มีลอจิก ‘low’ เป็นเวลาอย่างน้อย 480 ไมโครวินาที แล้วจึงปล่อยให้กลับมาอยู่ในสภาวะ ‘high’ อีกครั้งหนึ่ง ดังรูปที่ 2.3



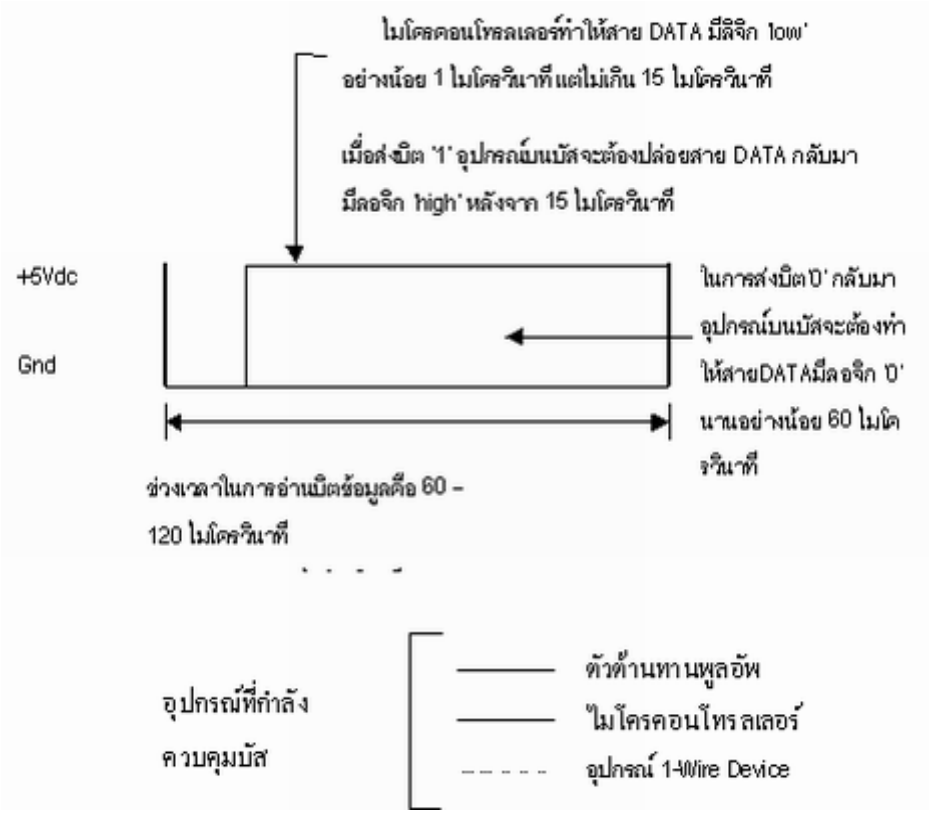
รูปที่ 2.3 จังหวะในการทำกระบวนการตรวจสอบว่ามีอุปกรณ์อยู่บนบัส

เมื่ออุปกรณ์ที่ใช้บัสแบบ 1–Wire Bus นี้ตรวจพบสภาวะ RESET นี้ มันก็จะตอบสนองด้วยการส่งพัลส์กลับไปเพื่อบอกให้ไมโครคอนโทรลเลอร์รู้ว่าบนสายบัสนี้มีอุปกรณ์แบบ 1–Wire Device กำลังทำงานอยู่ โดย อุปกรณ์แบบ 1–Wire Device จะปล่อยให้สาย DATA อยู่ในลอจิก ‘high’ อย่างน้อย 15 ไมโครวินาที แต่ไม่เกิน 60 ไมโครวินาที จากนั้นมันก็จะให้สาย DATA ลงมามีลอจิกเป็น ‘low’ ในช่วงเวลาประมาณ 60-240 ไมโครวินาที แล้วจึงปล่อยให้กลับไปมีลอจิก ‘high’เช่นเดิม ช่วงเวลานี้มีชื่อเรียกกันหลายชื่อเช่น ช่วงเวลาเริ่มติดต่อ (Initialization) หลังจากอุปกรณ์ 1–Wire Device ปล่อยให้สาย DATA กลับมาอยู่ในลอจิก ‘high’ แล้ว ไมโครคอรโทรลเลอร์จะต้องปล่อยให้สาย DATA อยู่ในลอจิกนี้นานอย่างน้อย 240 ไมโครวินาที ต่อไปจากนั้นไมโครคอนโทรลเลอร์ก็จะส่งคำสั่งเริ่มต้น (Initial Command) ขนาด 1 ไบต์ไปยังอุปกรณ์ 1–Wire Bus ซึ่งคำสั่งนี้อาจจะเป็นคำสั่งอะไรก็ได้ ไมโครคอนโทรลเลอร์จะส่งบิตของ คำสั่งนั้นออกไป โดยการเปลี่ยนสถานะของสาย DATA กลับไปกลับมาโดยตอนแรกจะให้เป็น ลอจิก ‘low’ แล้วจึงกลับมาเป็น ‘high’ ตามช่วงจังหวะเวลาที่เหมาะสม ช่วงเวลานานที่ ไมโครคอนโทรลเลอร์ทำให้สาย DATA มีลอจิก ‘low’ จะเป็นตัวแยกแยะว่าบิตไหนที่มีลอจิกเป็น ‘1’ บิตไหนมีลอจิกเป็น ‘0’ ให้ดูแผนภูมิเวลาในการเขียนบิต ‘1’ หรือ ‘0’ ดังรูปที่ 2.4



รูปที่ 2.4 จังหวะไมโครคอนโทรลเลอร์ใช้เขียนข้อมูลไปยังอุปกรณ์บนบัส

ช่วงเวลาระหว่าง 15-120 ไมโครวินาที หลังไมโครคอนโทรลเลอร์ทำให้สาย DATA มีลอจิก ‘low’ ในตอนนี้จะเรียกว่าช่วงการอ่านสถานะบิตข้อมูล (Sampling Window)



รูปที่ 2.5 จังหวะไมโครคอนโทรลเลอร์อ่านบิตข้อมูล จากอุปกรณ์บนบัส

ในการอ่านบิตข้อมูลจากบัสแบบ 1-Wire Bus ตอนแรกไมโครคอนโทรลเลอร์จะต้องทำให้สาย DATA มีลอจิก ‘low’ เป็นเวลานานไม่เกิน 15 ไมโครวินาทีแล้วจึงปล่อยให้สาย DATA กลับมามีลอจิก ‘high’ เช่นเดิมจากนั้นอุปกรณ์ 1–Wire Device ก็จะเข้าควบคุมสาย DATA แทนโดยจะส่งบิต ‘0’ โดยการทำให้สาย DATA มีลอจิกเป็น ‘low’ และส่งบิต ‘1’ โดยการปล่อยให้สาย DATA กลับมามีลอจิก ‘high’ ตามเดิม เมื่อส่งข้อมูลออกไปยังไมโครคอนโทรลเลอร์เรียบร้อยแล้ว จะมีการพักอยู่ชั่วขณะหนึ่ง จากนั้นอุปกรณ์ 1–Wire Device จะปล่อยการควบคุมจากสาย DATA ให้เป็นอิสระ และรอรับ คำสั่งการอ่านของข้อมูลครั้งต่อไป ถ้าอุปกรณ์ 1–Wire Device ส่งบิต ‘0’ ออกไปไมโครคอนโทรลเลอร์ก็จะสามารถตรวจสอบจุดสิ้นสุดของลอจิก ‘0’ นี้ง่าย เพราะสาย DATA จะกลับมาอยู่ที่ลอจิก ‘high’ ตามเดิม แต่ถ้าตรวจสอบจุดสิ้นสุดของการส่งบิต ‘1’ ของอุปกรณ์ 1–Wire Device จะต้องใช้เทคนิคมากกว่านี้ เพราะสาย DATA จะอยู่ที่ลอจิก ‘high’ อยู่แล้วนี่ก็เป็นเหตุผลว่าทำไมจังหวะเวลาในการอ่านเขียนข้อมูลจึงเป็นเรื่องที่สำคัญมาก เมื่อจะอ่านบิต ‘1’ จากอุปกรณ์ 1–Wire Device ไมโครคอนโทรลเลอร์จะต้องทำตามช่วงเวลาที่แสดงในแผนภูมิเวลาอย่างเคร่งครัด และต้องไม่ทำการอ่านลอจิกของบิตถัดมา จนกว่าจะผ่านเวลาไปแล้วอย่างน้อย 60 ไมโครวินาที

2.3 ไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 ซึ่งมีหน่วยความจำภายในเป็นแบบแฟลช(Flash Memory) ของ Atmel Corporation มีเบอร์ขึ้นต้นด้วย AT89 เหตุผลที่ใช้ไมโครคอนโทรลเลอร์แบบนี้ในการเรียนรู้เพื่อใช้งานไมโครคอนโทรลเลอร์ตระกูล MCS-51 มีด้วยกันหลายประการดังนี้

1.หน่วยความจำโปรแกรมภายในเป็นแบบแฟลชสามารถลบ และเขียนใหม่ได้เป็นพันครั้งจึงสามารถใช้งานในรูปแบบของไมโครคอนโทรลเลอร์ชิปเดี่ยวโดยไม่ต้องใช้หน่วยความจำภายนอกส่งผลให้ใช้งานพอร์ตอินพุตเอาต์พุตของไมโครคอนโทรลเลอร์ได้อย่างเต็มประสิทธิภาพ

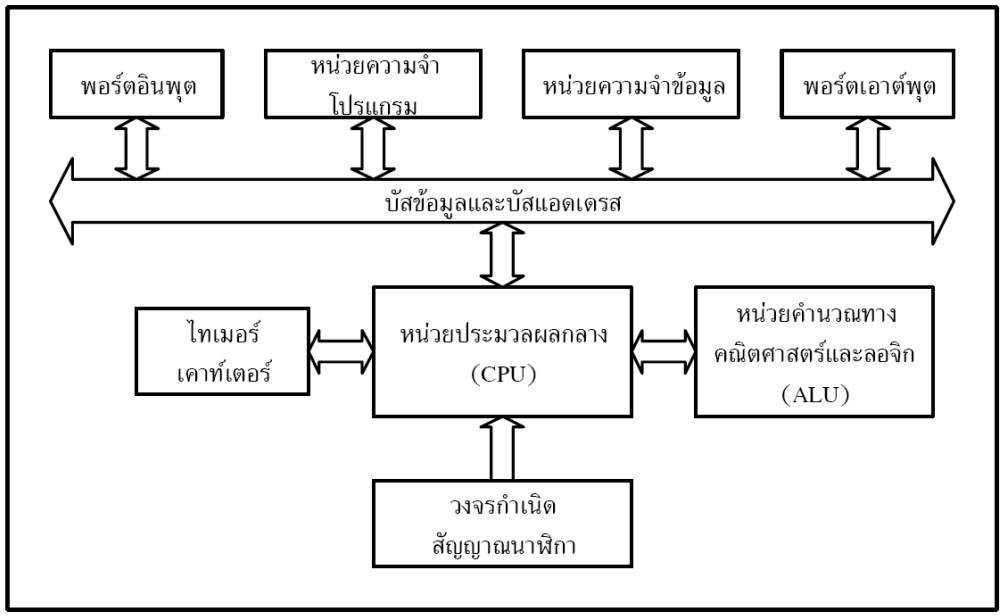
2. ต้นทุนและเวลาในการพัฒนาระบบไมโครคอนโทรลเลอร์ลดลงเนื่องจากไม่ต้องใช้เครื่องมือพัฒนาจำพวกอีมูเลเตอร์ และเครื่องโปรแกรมอีพรอม

3. บริษัทผู้ผลิตได้ทำการผลิตไมโครคอนโทรลเลอร์ตระกูลนี้ออกมาหลายเบอร์ และมีความสามารถแตกต่างกันไป ทำให้มีทางเลือกในการใช้งานสูง

4. ด้วยการใช้หน่วยความจำภายในตัวไมโครคอนโทรลเลอร์ทำให้สามารถป้องกันการลอกข้อมูลของหน่วยความจำโปรแกรมได้เป็นอย่างดี

5. ในบางเบอร์ของไมโครคอนโทรลเลอร์ที่ผลิตโดย Atmel สามารถทำการโปรแกรมข้อมูลในหน่วยความจำโปรแกรมได้โดยที่ไม่ต้องถอดตัวไมโครคอนโทรลเลอร์ออกมาทำการโปรแกรมใหม่หรือเรียกว่า การโปรแกรมในวงจร หรือ ในระบบ (In-system programming) โดยใช้ลักษณะการติดต่อแบบ SPI (Serial Peripheral Interface) ทำให้การพัฒนาหรือการซ่อมบำรุงตลอดจนการปรับปรุงหรือ อัปเกรดข้อมูลในหน่วยความจำโปรแกรมทำได้อย่างสะดวก ภายใต้งบประมาณที่ไม่สูงมากนัก

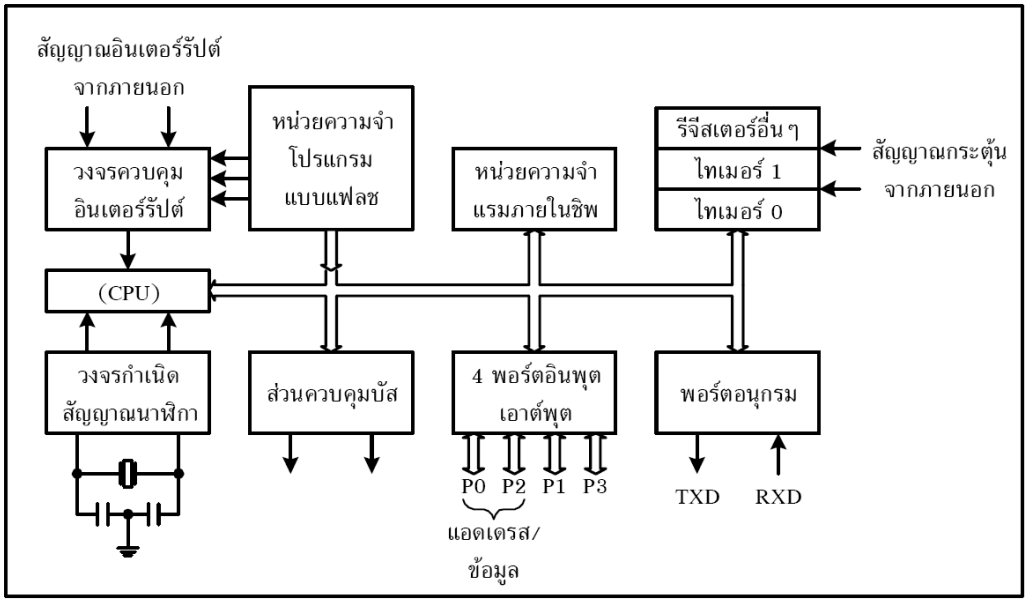
6. ชุดคำสั่งและสถาปัตยกรรมพื้นฐานกับไมโครคอนโทรลเลอร์ MCS-51 ของผู้ผลิตไม่ว่าจะเป็นอินเทล, ซีเมนส์หรือ ดัลลัส



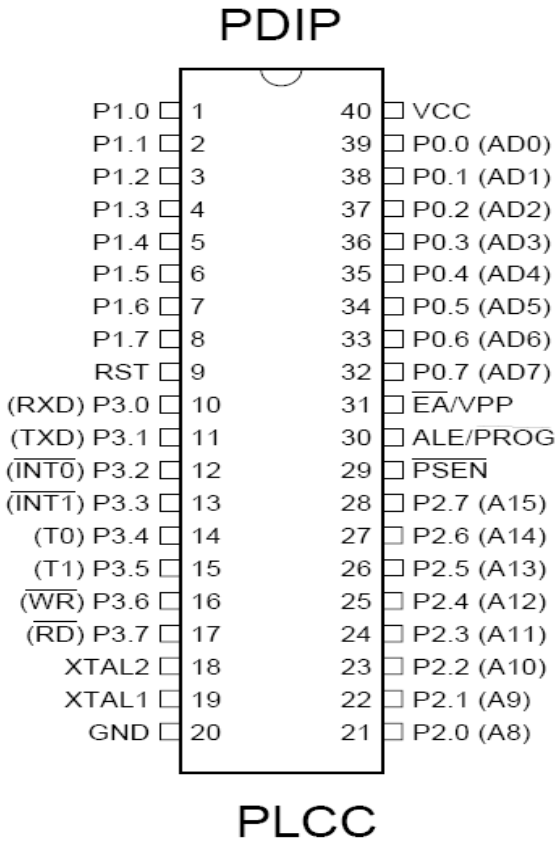
รูปที่ 2.6 โครงสร้างพื้นฐานของไมโครคอนโทรลเลอร์

2.3.1 คุณสมบัติทางเทคนิคของไมโครคอนโทรลเลอร์ อนุกรม AT89xx

1. เป็นไมโครคอนโทรลเลอร์ที่ใช้ซีพียูขนาด 8 บิต
2. ภายในมีหน่วยความจำโปรแกรมเป็นแบบแฟลชสามารถลบ และเขียนใหม่ได้
3. หน่วยความจำพื้นฐานเป็นหน่วยความจำแบบแรม ในบางเบอร์มีหน่วยความจำแบบอีพรอมเพิ่มเติม
4. ขาพอร์ตเป็นแบบสองทิศทาง สามารถใช้งานเป็นได้ทั้งอินพุตและเอาต์พุต
5. มีวงจรสื่อสารอนุกรมแบบฟลูดูเพล็กซ์
6. ไทเมอร์/เคาน์เตอร์ขนาด 16 บิตอย่างน้อย 2 ตัว
7. สามารถรองรับแหล่งกำเนิดอินเตอร์รัปต์ได้ 6 ประเภท
8. สามารถขยายหน่วยความจำภายนอกเพิ่มเติมได้สูงสุด 64 กิโลไบต์
9. มีวงจรกำเนิดสัญญาณนาฬิกาอยู่ภายในชิป



รูปที่ 2.7 โครงสร้างพื้นฐานไมโครคอนโทรลเลอร์ แบบแฟลชอนุกรมAT89Cxx



รูปที่ 2.8 แสดงการจัดขาของไมโครคอนโทรลเลอร์ MCS-51 (AT89C51)

2.3.2 หน้าที่การใช้งานของขาไมโครคอนโทรลเลอร์อนุกรมAT89C5x

- ขา +V_{CC} (ขา 40) ใช้สำหรับต่อไฟเลี้ยง

- ขา GND (ขา 20) สำหรับต่อกราวด์ของระบบ

- ขา พอร์ต 0 (ขา 32 ถึง 39) จะมี 8 ขา คือ P0.0-P0.7 โดยสามารถกำหนดให้เป็นได้ทั้งอินพุตและเอาต์พุต โดยหากต้องการให้ขา พอร์ต 0 ขาใดเป็นอินพุตก็สามารถทำได้โดยการเขียนข้อมูล ‘1’ ไปยังขาที่ต้องการติดต่อด้วยแต่ถ้าต้องการเป็นเอาต์พุตก็สามารถทำได้โดยการเขียนข้อมูลส่งไปยังขาที่ต้องการติดต่อ และยังใช้ในการติดต่อกับขาแอดเดรสไบต์ต่ำของหน่วยความจำภายนอก (A0-A7) และขาข้อมูล (D0-D7) โดยใช้กระบวนการมัลติเพล็กซ์เข้าช่วยในการทำงาน

- ขา พอร์ต 1 (ขา 1-8) จะมี 8 ขา คือ P1.0-P1.7 โดยสามารถกำหนดให้เป็นได้ทั้งอินพุต และเอาต์พุต โดยหากต้องการให้ขา พอร์ต 1 ขาใดเป็นอินพุตก็สามารถทำได้โดยการเขียนข้อมูล ‘1’ ไปยังขาที่ต้องการติดต่อด้วยและนอกจากนี้ในกลุ่มของ AT89Sxx จะใช้ขาP1.0 เป็นขาอินพุต สำหรับค่าของไทเมอร์ 1 และ P1.1 เป็นขาอินพุต สำหรับค่าของไทเมอร์2 ในขณะที่ P1.4-P1.7 ใช้เป็นขาเชื่อมต่อแบบ SPI เพื่อทำการโปรแกรมข้อมูลในระบบ

- ขา พอร์ต 2 (ขา 21-28) จะมี 8 ขา คือ P2.0-P2.7 ใช้เป็นอินพุตและเอาท์พุตพอร์ตและใช้งานในการติดต่อกับขาแอดเดรสไบต์สูงของหน่วยความจำภายนอก A8-A15- ขา พอร์ต 3 (ขา 10-17) จะมี 8 ขา คือ P3.0-P3.7 ใช้เป็นอินพุตและเอาท์พุตพอร์ตและถูกใช้งานในหน้าที่พิเศษอื่นๆอีกหลายอย่างดังนี้

P3.0 ใช้เป็นขาอินพุตสำหรับรับข้อมูลจากการสื่อสารแบบอนุกรมหรือขา RXD

P3.1 ใช้เป็นขาอินพุตสำหรับส่งข้อมูลจากการสื่อสารแบบอนุกรมหรือขา TXD

P3.2 ใช้เป็นขาอินพุตรับสัญญาณอินเตอร์รัปต์จากภายนอกช่อง 0 หรือขา INT0

P3.3 ใช้เป็นขาอินพุตรับสัญญาณอินเตอร์รัปต์จากภายนอกช่อง 1 หรือขา INT1

P3.4 ใช้เป็นขาอินพุตสำหรับรับสัญญาณไทเมอร์จากช่อง 0 หรือขา T0

P3.5 ใช้เป็นขาอินพุตสำหรับรับสัญญาณไทเมอร์จากช่อง 1 หรือขา T1

P3.6 ใช้เป็นขาอินพุตหรือเป็นขา WR ในกรณีที่ใช้เชื่อมต่อกับหน่วยความจำภายนอก

P3.7 ใช้เป็นขาอินพุตหรือเป็นขา RD ในกรณีที่ใช้เชื่อมต่อกับหน่วยความจำภายนอก

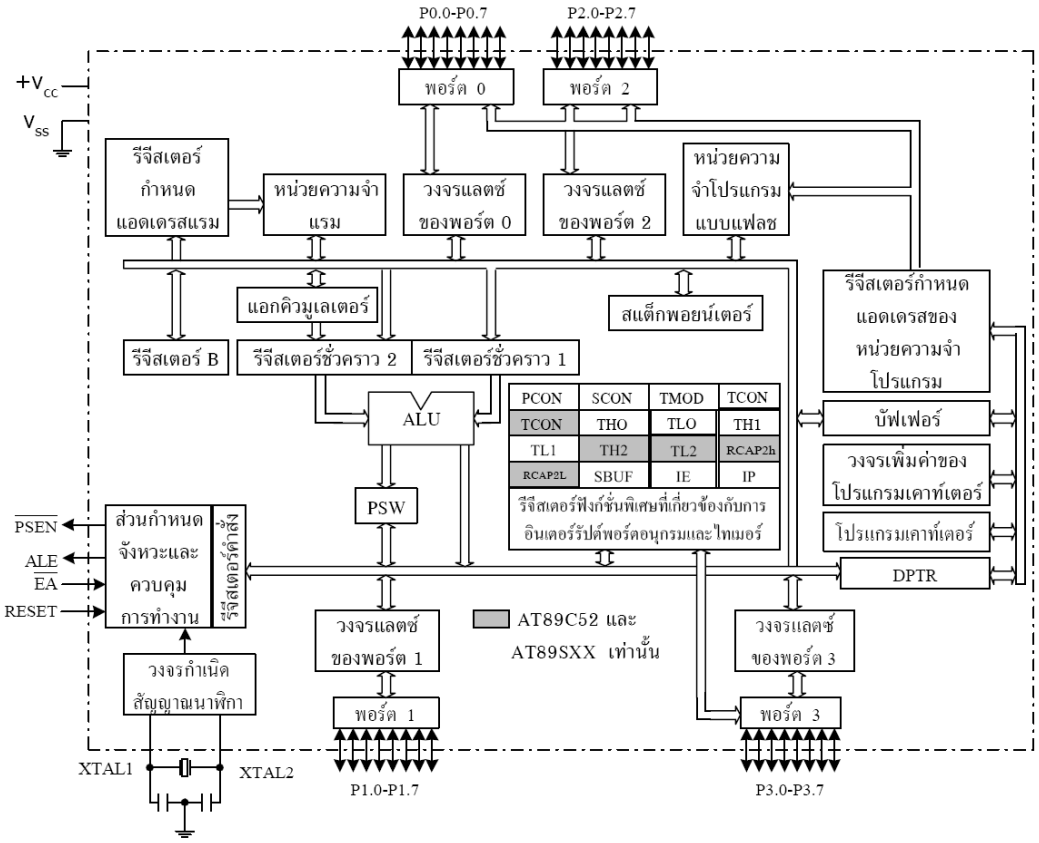
- ขา RESET (ขา9) ใช้ในการรีเซ็ตการทำงานของไมโครคอนโทรลเลอร์เพื่อเริ่มต้นการทำงานใหม่

- ขา ALE/PROG (ขา30) เป็นขาควบคุมการแลตซ์ของขาพอร์ต 0 เมื่อมีการใช้งานหน่วยความจำภายนอกหากขานี้มีสถานะเป็นลอจิก ‘0’ และใช้เป็นขาสำหรับใช้รับพัลส์ของการโปรแกรมข้อมูลลงใน ไมโครคอนโทรลเลอร์ MCS-51 ถ้ามีสถานะเป็นลอจิก ‘1’ ในรุ่นที่มีหน่วยความจำเป็นแบบ อีพรอม

- ขา PSEN (ขา9) ใช้ส่งสัญญาณร้องขอติดต่อกับหน่วยความจำโปรแกรมภายนอกในช่วงของการอ่านเขียนข้อมูลกับหน่วยความจำภายนอก เมื่อใช้โปรแกรมจากหน่วยความจำโปรแกรมภายในชิพ จะไม่ส่งสัญญาณออกมาที่ขานี้

- ขา EA/VPP (ขา31) ใช้สำหรับเลือกให้ MCS-51 ติดต่อกับหน่วยความจำโปรแกรมภายนอกหรือจากหน่วยความจำภายใน MCS-51 เองโดยหากมีสถานะเป็น ‘0’ จะเลือกใช้โปรแกรมภายนอก หากมีสถานะเป็น ‘1’ จะเลือกใช้หน่วยความจำภายใน MCS-51 และใช้เป็นขาอินพุตสำหรับรับแรงดันไฟสูง สำหรับการโปรแกรมหน่วยความจำภายใน สำหรับMCS-51 แบบแฟลชต้องการแรงดันในการโปรแกรม +12VDC

- ขา XTAL1 และ XTAL2 (ขา 19 และ ขา 18) เป็นขาต่อคริสตอล เพื่อสร้างสัญญาณนาฬิกาในการกำหนดจังหวะในการทำงานของ MCS-51



รูปที่ 2.9 โครงสร้างภายในของ MCS-51 แบบแฟลชของ Atmel

2.4 โมดูลเครื่องรับ-ส่งไร้สาย

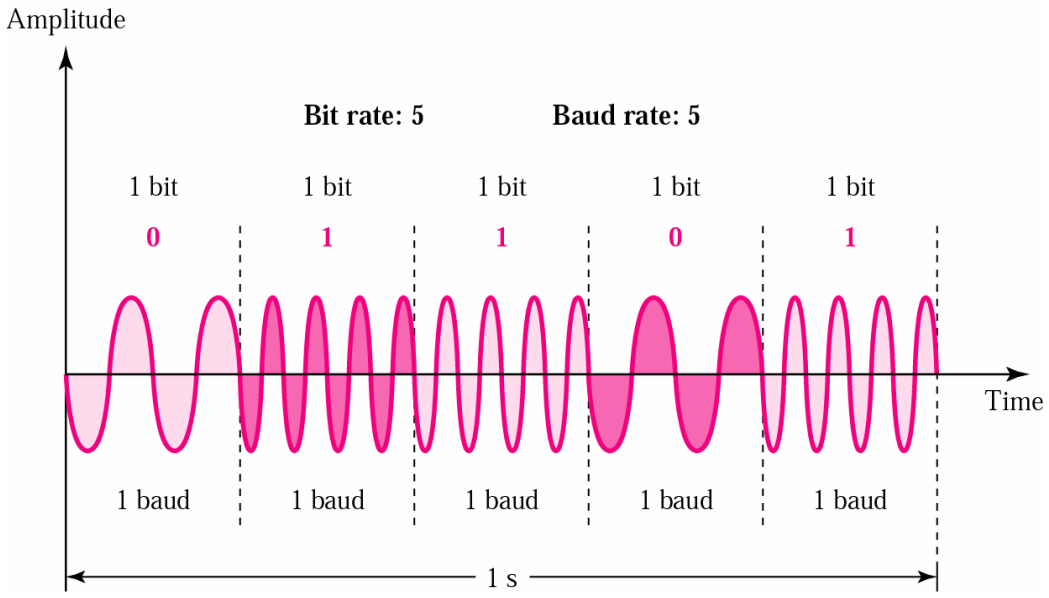
โมดูลสื่อสารผ่านคลื่นวิทยุย่าน UHF แบบฮาร์ฟดูเพล็กซ์ ซึ่งเป็นวิธีการสื่อสารแบบสองทิศทางสลับกันรับ-ส่งข้อมูลผ่านช่องสัญญาณเดียวกัน ส่งเป็นแพ็กเก็ตจึงง่ายต่อการเชื่อมต่อเป็นระบบ เครื่อข่ายไร้สาย ระหว่าง จุดต่อจุด หรือ จุดต่อหลายจุด โมดูลสามารถตั้งค่าชื่อผู้รับและชื่อผู้ส่งของชุดข้อมูลในการสื่อสาร ซึ่งส่งเป็นแพ็กเก็ต มีการรอตอบกลับ ACK (Acknowledge) เมื่อโมดูลปลายทางได้รับข้อมูลถูกต้อง พร้อมการจัดสรรการเข้าใช้ช่องสัญญาณ ด้วยเทคนิค CSMA/CA เพื่อหลีกเลี่ยงไม่ให้เกิดการชนกันของช่องสัญญาณ

โมดูลเครื่องส่ง-เครื่องรับวิทยุ ความถี่ 433MHz ใช้เทคนิคการมอดูเลต และดีมอดูเลตแบบ FSK ใช้วิธีการรับส่งข้อมูลแบบทิศทางเดียว

2.4.1 Frequency Shift Keying (FSK)

FSK ในการมอดูเลตแบบFSK ขนาดของคลื่นพาห์จะไม่เปลี่ยนแปลงที่เปลี่ยนแปลงคือความถี่ของคลื่นพาห์นั่นคือ เมื่อบิตมีค่าเป็น1 ความถี่ของคลื่นพาห์จะสูงกว่าปกติและเมื่อบิตมีค่าเป็น **0** ความถี่ของคลื่นพาห์ก็จะต่ำกว่าปกติ การเปลี่ยน ความถี่ ของสัญญาณพาหะ ตามบิตข้อมูล เช่น ความถี่ f_{c0} เมื่อข้อมูลเป็น 0 ความถี่ f_{c1} เมื่อข้อมูลเป็น 1 ตามสมการ

$$FSK(t)=\begin{cases} A\sin(2\pi f_{c0}t+\phi) & ,data(t)=0 \\ A\sin(2\pi f_{c1}t+\phi) & ,data(t)=1 \end{cases} \qquad (1)$$



รูปที่ 2.10 การผสมสัญญาณของข้อมูลด้วยวิธีการ FSK

จุดเด่นของ FSK

คือ มีภูมิต้านทานต่อสัญญาณรบกวนมากกว่าวิธี Modulation แบบ ASK เนื่องจากอุปกรณ์ด้านรับ มองหา ความถี่เฉพาะ ที่อยู่ในช่วงเวลาหนึ่งๆ โดยไม่สนใจ Noise กระชากระยะสั้น (Transient Noise)

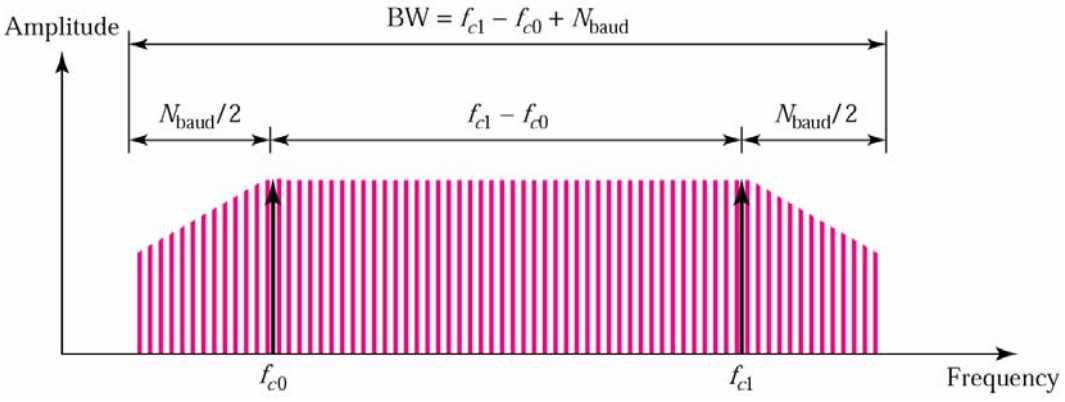
จุดด้อยของ FSK

คือ ต้องการ Bandwidth กว้างกว่าวิธี Modulation แบบ ASK เมื่อส่งข้อมูลที่มี Baud Rate เท่ากัน

แถบความที่ของ FSK สามารถหาได้จาก ผลรวม ระหว่าง ความแตกต่างของความถี่พาหะ และ อัตราเร็วของข้อมูล ดังสมการ

$$BW_{FSK} = (f_{c1-} f_{c0}) + N_{baud} \qquad (2)$$

เมื่อ N_{baud} คืออัตราเร็วของข้อมูล และ f_{c1} และ f_{c0} คือความถี่สูงสุด และต่ำสุดตามลำดับ ดังรูปที่ 2.17



รูปที่ 2.11 Spectrum ของสัญญาณ FSK ที่มีความถี่ของสัญญาณพาหะ

2.4.2 หลักกการทำงานของโมดูลเครื่องรับ-ส่งไร้สาย

การควบคุมการไหลของข้อมูล (Flow control)เนื่องจากโมดูลมีหน่วยความจำบัฟเฟอร์รองรับรับข้อมูลจำกัด ดังนั้นจึงต้องมีกลไกควบคุมการไหลของข้อมูลให้สัมพันธ์กัน เพื่อป้องกันไม่ให้รับข้อมูลผิดพลาด โดยใช้ ขา RTS Flow control (Hardware)RTS : Request to Send หากขา RTS มีลอจิกเป็น “1” ให้อุปกรณ์ โฮสต์ (Host) หยุดการส่งข้อมูล ชั่วคราวการจัดสรรการใช้ช่องสัญญาณ (Carrier Sense Multiple Access/Collision Avoidance)เป็นการจัดสรรการเข้าใช้ช่องสัญญาณของแต่ล่ะช่องสัญญาณ ซึ่งการทำงานของกลไกนี้เรียกย่อๆว่า CSMA/CA คือเมื่อ

โมดูลหนึ่งต้องการเข้าใช้ช่องสัญญาณ จะต้องตรวจสอบช่องสัญญาณก่อนว่ามีการใช้ช่องสัญญาณอยู่หรือไม่ และรอจนกว่าช่องสัญญาณว่าง เมื่อช่องสัญญาณว่างจะต้องรอต่อไปอีกระยะหนึ่ง (Random Back-off) ด้วยการสุ่มค่าเวลา สุ่มได้ระยะเวลาน้อยกว่า ก็มีสิทธิในการใช้ช่องสัญญาณก่อน

2.4.2 รูปแบบการจัดการสื่อสาร

โมดูลตัวนี้รองรับการทำงาน 4 โหมด คือ

1 การสื่อสารระหว่างจุดต่อจุด (Point – To – Point) ส่งข้อมูลเป็นแพ็กเกต รอการตอบกลับ (Acknowledge) และส่งซ้ำหากไม่ได้รับการตอบกลับ

2 การสื่อสารระหว่างจุดต่อหลายจุด (Point-To-Multipoint) พร้อมรอการตอบกลับควบคุมด้วยชุดคำสั่งเสษโดยตรง

3 การสื่อสารแบบแพร่กระจาย (Broadcast Multi-drop) ไม่มีการรอตอบกลับ (Unacknowledged)

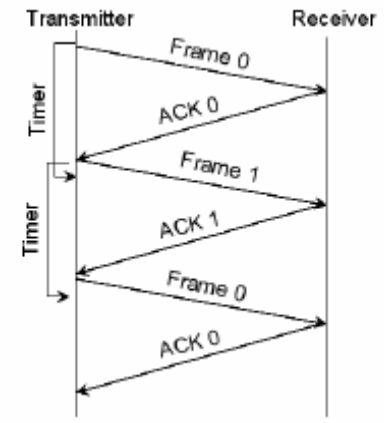
4 สื่อสารด้วยคำสั่งระยะไกล (Remote configured)

ชนิดข้อมูล	ลำดับข้อมูล	ชื่อผู้รับ(DST)	ชื่อผู้ส่ง(SRC)	ข้อมูล(DATA)
1 Byte	1 Byte	1 Byte	1 Byte	60 Byte

2.4.2.1 การสื่อสารระหว่างจุดต่อจุดรอการตอบ

การสื่อสารระหว่างจุดต่อจุดรอการตอบกลับ (Acknowledged Point-to-Point) ในโหมดนี้ตั้งค่าให้โมดูลตัวใดตัวหนึ่งหรือทั้งสองเป็นตัวแม่ (Master) พร้อมกำหนดอัตตราความเร็วสื่อสารช่องสัญญาณที่ใช้งาน และชื่อปลายทางผู้รับ และ ชื่อต้นทางผู้ส่ง การส่งข้อมูลจะส่งเป็นชุดข้อมูลแพ็กเก็ต หรือเฟรมระหว่าง โมดูลทั้งสองตัว การสื่อสาร ระหว่างจุดต่อจุด นั้นมีการรอการตอบกลับ (Acknowledge) และส่งซ้ำ 3 ครั้งหากไม่ได้รับการตอบกลับเมื่อการรับส่งถูกต้อง โมดูลจะส่งข้อมูลที่ได้รับออกทางพอร์ตอนุกรม (RS232 TTL)

รูปที่ 2.12 การสื่อสารระหว่างจุดต่อจุดรอการตอบ



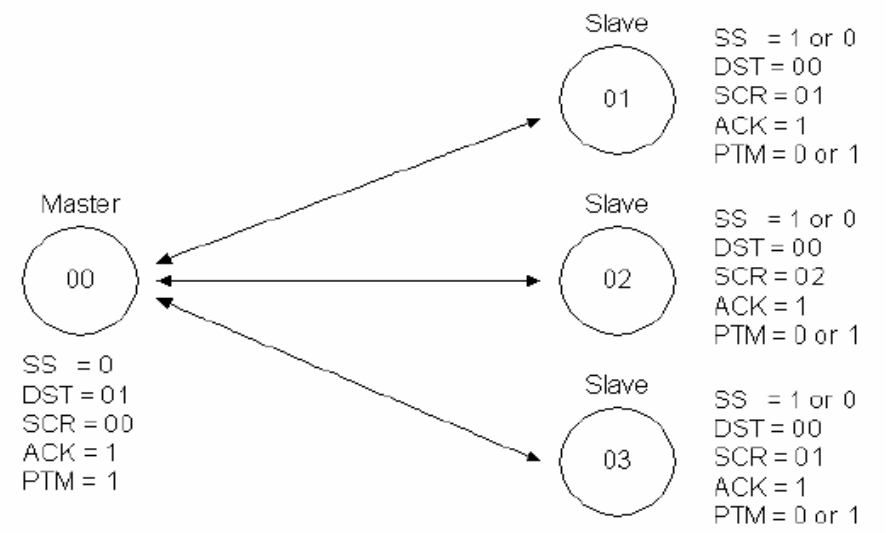
รูปที่ 2.13.การทำงานระหว่างจุดต่อจุดรอการตอบ

2.4.2.2 การสื่อสารระหว่างจุดต่อหลายจุดรอการตอบกลับ (Ack Point-to- Multipoint)

รูปแบบการเชื่อมต่อแบบจุดต่อหลายจุดนั้น โมดูลต้นทางหรือตัวแม่ที่ต้องการส่งข้อมูลและเชื่อมต่อกับอุปกรณ์โฮลต์ (Host) สามารถตั้งค่าเพื่อเปลี่ยนชื่อผู้รับของชุดข้อมูล หรือ ช่องสัญญาณสื่อสารได้ การตั้งค่าระหว่างการใช้งานโมดูลนั้นมี 2 วิธี วิธีแรกตั้งค่าผ่านโปรแกรม HyperTerminal ด้วยชุดคำสั่ง AT- command เซ็ตขา CONF เป็นโลจิก ‘0’เพื่อเข้าโหมดการตั้งค่า ข้อมูลที่ตั้งค่าไว้จะถูกเก็บไว้ในหน่วยความจำถาวร แบบ EEPROM เขียนทับได้ประมาณ 100,000 ครั้ง ส่วนวิธีที่สองได้เพิ่มชุดคำสั่งพิเศษเข้ามา เพื่อตั้งค่าโมดูลได้ชั่วคราวเท่านั้น จะเก็บข้อมูลไว้ในหน่วยความจำแบบ RAM โดยการส่งชุดคำสั่งพิเศษไปยังโมดูล ต้องตั้งค่า PTM เป็น‘1’

หากต้องการตั้งค่าชื่อผู้รับของชุดข้อมูลเป็นชื่อ 01 ให้ส่งชุดคำสั่งพิเศษ “+++DST01” หรือต้องการตั้งค่าช่องสัญญาณสื่อสารของโมดูลเป็นช่อง 01 ให้ส่งชุดคำสั่งพิเศษ “+++CH01” แล้วรอ

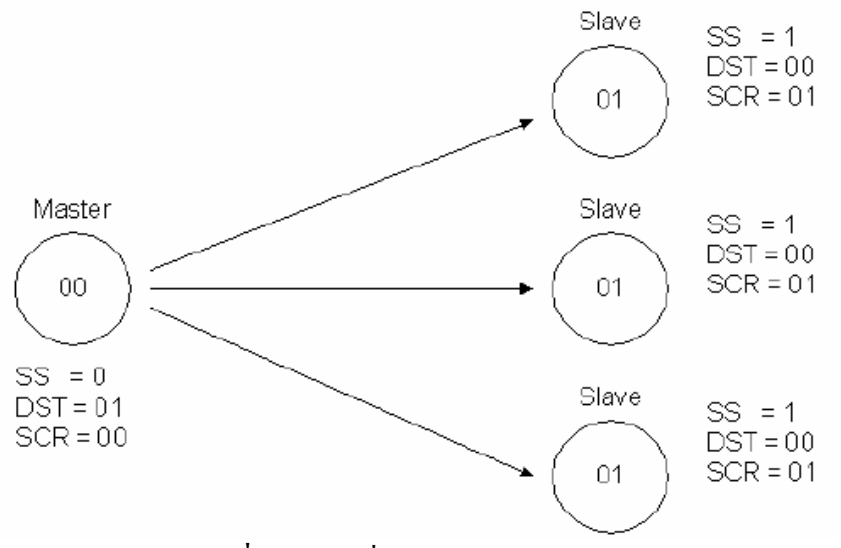
การยืนยันการตั้งค่าจากโมดูลด้วยข้อความ “OK” เมื่อได้รับการยืนยัน สามารถส่งชุดข้อมูลที่ต้องการออกทาง RF ได้



รูปที่ 2.14 การสื่อสารระหว่างจุดต่อหลายจุดรอการตอบกลับ

2.4.2.3. การสื่อสารแบบแพร่กระจาย (Broadcast Multi-drop)

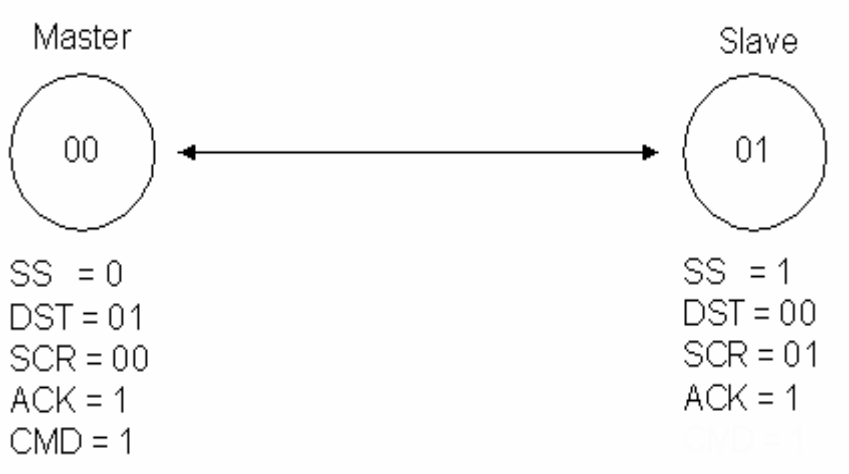
การทำงานในโหมดนี้เป็การสื่อสารแบบส่งข้อมูลทิศทางเดียวระหว่างโมดูลตัวแม่ (Master) กับโมดูลตัวลูก (Slave) ไม่มีการรอตอบกลับ โดยการตั้งค่า SCR ของโมดูลปลายทางทุกตัวมีชื่อเหมือนกัน ทำให้โมดูลปลายทางทุกตัวจะได้รับข้อมูล และส่งออกทางพอร์ตอนุกรม (RS232 TTL) พร้อมกัน



รูปที่ 2.15 การสื่อสารแบบแพร่กระจาย

2.4.2.4. การสื่อสารด้วยคำสั่งระยะไกล (Remote configured)

ในโหมดนี้สามารถส่งชุดคำสั่งไปตั้งค่าโมดูลปลายทาง ที่อยู่ระยะไกลที่อยู่ในรัศมี การทำงาน ต้องตั้งค่า CMD เป็น ‘1’ สำหรับส่งชุดคำสั่ง กำหนดโมดูลต้นทางเป็นตัวแม่ (Master) และโมดูลปลายทางเป็นตัวลูก(Slave) รูปแบบคำสั่งนั้น พิมพ์ เครื่องหมาย ‘>’ ตามด้วยหมายเลขประจำตัวโมดูล 4 หลัก (Serial No) แล้วคั้นด้วย เครื่องหมาย **‘,’** และตามด้วยชุดคำสั่ง AT- command รองรับคำสั่ง ATCH,ATPA,ATDA,ATSA, ATABRและ ATACK



รูปที่ 2.16 การสื่อสารด้วยคำสั่งระยะไกล

2.5 โปรแกรม Visual Basic

2.5.1 รู้จักกับ Visual Basic

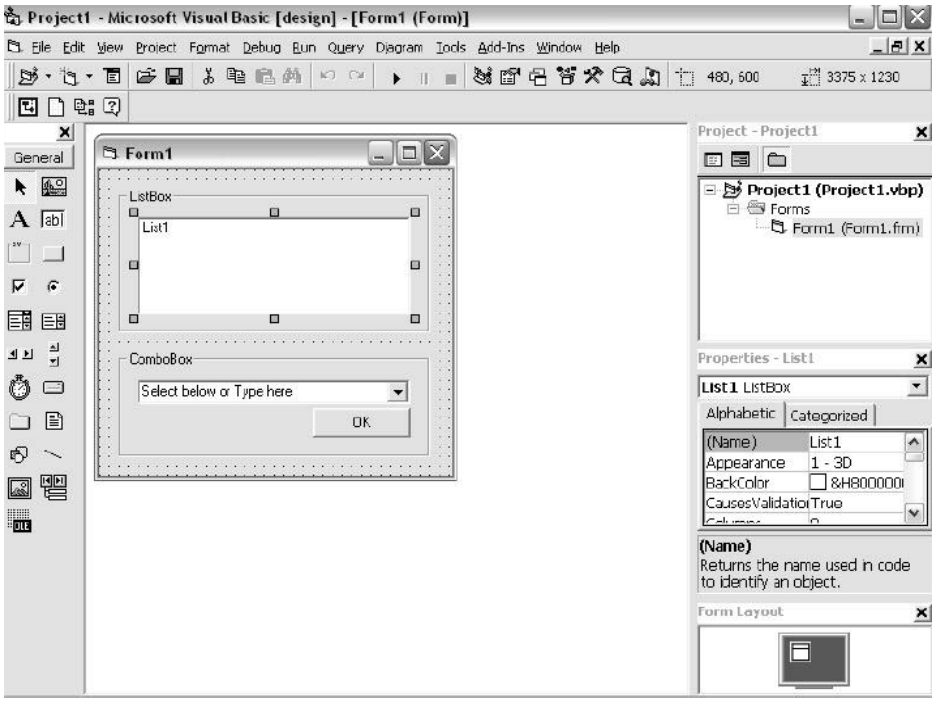
โปรแกรม Visual Basic (VB) เป็นโปรแกรมสำหรับพัฒนาโปรแกรมประยุกต์ที่กำลังเป็นที่นิยมใช้อยู่ในปัจจุบัน โปรแกรม Visual Basic เป็นโปรแกรมที่ได้เปลี่ยนรูปแบบการเขียนโปรแกรมใหม่ โดยมีชุดคำสั่งมาสนับสนุนการทำงาน มีเครื่องมือต่าง ๆ ที่เรียกกันว่า คอนโทรล (Controls) ไว้สำหรับช่วยในการออกแบบโปรแกรม โดยเน้นการออกแบบหน้าจอแบบกราฟฟิก หรือที่เรียกว่า Graphic User Interface (GUI) ทำให้การจัดรูปแบบหน้าจอเป็นไปได้ง่าย และในการเขียนโปรแกรมนั้นจะเขียนแบบ Event-Driven Programming คือ โปรแกรมจะทำงานก็ต่อเมื่อเหตุการณ์ (Event) เกิดขึ้น ตัวอย่างของเหตุการณ์ได้แก่ ผู้ใช้เลื่อนเมาส์ ผู้ใช้กดปุ่มบนคีย์บอร์ด ผู้ใช้กดปุ่มเมาส์ เป็นต้น

เครื่องมือ หรือ คอนโทรล ต่าง ๆ ที่ Visual Basic ได้เตรียมไว้ให้ ไม่ว่าจะเป็น Form TextBox Label ฯลฯ ถือว่าเป็นวัตถุ (Object ในที่นี้ขอใช้คำว่า ออบเจ็กต์) นั่นหมายความว่า ไม่ว่าจะเป็นเครื่องมือใด ๆ ใน Visual Basic จะเป็นออบเจ็กต์ทั้งสิ้น สามารถที่จะควบคุมการทำงาน แก้ไข

คุณสมบัติของออบเจ็กต์นั้นได้โดยตรง ในทุกๆ ออบเจ็กต์จะมีคุณสมบัติ (properties) และเมธอด (Methods) ประจำตัว ซึ่งในแต่ละออบเจ็กต์ อาจจะมีคุณสมบัติและเมธอดที่เหมือน หรือต่างกันก็ได้ ขึ้นอยู่กับชนิดของออบเจ็กต์

ในการพัฒนาโปรแกรมประยุกต์ด้วย Visual Basic การเขียนโค้ดจะถูกแบ่งออกเป็นส่วนๆ เรียกว่า โพรซีเดอร์ (procedure) แต่ละโพรซีเดอร์จะประกอบไปด้วย ชุดคำสั่งที่พิมพ์เข้าไปแล้ว ทำให้คอนโทรลหรือออบเจ็กต์นั้น ๆ ตอบสนองการกระทำของผู้ใช้ ซึ่งเรียกว่าการเขียนโปรแกรมเชิงวัตถุ (Object Oriented Programming-OOP) แต่ตัวภาษา Visual Basic ยังไม่ถือว่าเป็นการเขียนโปรแกรมแบบ OOP อย่างแท้จริง เนื่องจากข้อจำกัดหลายๆ อย่างที่ Visual Basic ไม่สามารถทำได้

การเขียนโปรแกรม คือการสั่งให้คอมพิวเตอร์ทำงานตามที่เราต้องการ สำหรับโปรแกรม Visual Basic 6.0 เป็นเครื่องมือในการสร้างโปรแกรมบนระบบปฏิบัติการ Windows ที่ใช้งานง่าย โดยในโปรแกรม Visual Basic จะเป็นการเลือกเครื่องมือต่าง ๆ มาออกแบบหน้าจอของโปรแกรมที่จะสร้าง ซึ่งเราเรียกการเขียนโปรแกรมลักษณะนี้ว่า Visual Programming การเขียนโปรแกรมแบบนี้เราไม่จำเป็นต้องใช้คำสั่งต่าง ๆ มากก็สามารถสร้างโปรแกรมได้อย่างรวดเร็ว สำหรับการเขียนโปรแกรมแบบ Visual Programming มีลักษณะดังรูปที่ 2.17



รูปที่ 2.17 แสดงการเขียนโปรแกรมแบบ Visual Programming

จากการเขียนโปรแกรมแบบ Visual Programming ข้างต้นเมื่อเราสั่งรันโปรแกรมจะได้ดังรูปที่ 2.18

รูปที่ 2.18 การรันโปรแกรมจะมีลักษณะเหมือนกับหน้าตาที่ได้ออกแบบไว้

หลังจากที่ได้ออกแบบหน้าจอโปรแกรมแล้ว จะต้องเขียนโปรแกรมควบคุมการทำงานด้วย โดยใช้ภาษา Basic (ย่อมาจาก Beginners All-Purpose Symbolic Instruction Code) ซึ่งเป็นภาษาที่ใช้งานง่าย เหมาะสำหรับผู้เริ่มต้นศึกษาการเขียนโปรแกรมบนวินโดวส์

2.5.2 การนำ Visual Basic ไปใช้งาน

สำหรับ VB6 นั้นเป็นเครื่องมือที่สร้างโปรแกรมต่าง ๆ ได้ดังนี้

2.5.2.1 โปรแกรมทั่วไปที่รันบนระบบปฏิบัติการ Windows

เราสามารถสร้างโปรแกรมทางด้านกราฟิก โปรแกรมจัดการไฟล์ โปรแกรมคำนวณเลขพื้นฐานเป็นต้น

2.5.2.2 โปรแกรมฐานข้อมูล

VB6 ช่วยสร้างโปรแกรมฐานข้อมูลเป็นเรื่องง่าย เนื่องจากมีเครื่องมือเกี่ยวกับฐานข้อมูลอย่างครบถ้วน เช่น เครื่องมือในการติดต่อฐานข้อมูล ทั้ง Microsoft Access หรือฐานข้อมูล ระบบ Client Server เช่น Microsoft SQL Server โดยการติดต่อฐานข้อมูลนั้น เราเพียงแต่กำหนดค่าตำแหน่งของฐานข้อมูลพร้อมกับข้อมูลที่จำเป็นในการติดต่อกับฐานข้อมูลเท่านั้น ก็จะสามารถติดต่อกับฐานข้อมูลได้ทันที

2.5.2.3 คอมโพเนนต์ทางด้าน ActiveX

ซึ่งได้แก่ ActiveX Component , ActiveX Control และ ActiveX Document ซึ่งเป็นเครื่องมือที่ช่วยให้สามารถนำส่วนของโปรแกรมที่เราได้สร้างแล้วไปใช้ในโปรแกรมอื่น ๆ เช่น Micro Excel เป็นต้น

2.5.2.4 โปรแกรมที่รันบนอินเตอร์เน็ตหรืออินทราเน็ตผ่านทาง Web Browser

สามารถสร้างโปรแกรมที่รันบนอินเตอร์เน็ตได้โดยง่ายดาย โดยไม่ต้องเรียนรู้การเขียนคำสั่งด้วยภาษา HTML (Hypertext Markup Language) หรือภาษาสคริปต์ที่ใช้งานบนอินเตอร์เน็ต

2.5.3 หลักการในการเขียนโปรแกรมด้วย VB6

สำหรับขั้นตอนในการสร้างโปรแกรม VB 6 สามารถแบ่งออกได้เป็น 2 ขั้นตอนหลัก คือ

2.5.3.1 การออกแบบหน้าจอของโปรแกรม

ซึ่งส่วนนี้ทำหน้าที่ติดต่อกับผู้ใช้งาน เรียกว่า “ยูสเซอร์อินเตอร์เฟส: User Interface” คอนโทรล (Controls) ซึ่งเป็นเครื่องมือในการออกแบบหน้าจอของโปรแกรม เครื่องมือเหล่านี้จะอยู่ในทูลบอกซ์ โดยให้เราเลือกคอนโทรลที่ตรงกับจุดประสงค์ในการใช้งาน และนำมาวางบนฟอร์มว่างที่ปรากฏอยู่

สำหรับคอนโทรลที่เป็นพื้นฐานใน VB6 ได้แก่

เท็กบอกซ์ (Text box) ผู้ใช้สามารถเติมหรือแก้ไขข้อความโดยผ่านทางคีย์บอร์ดเข้าสู่โปรแกรม สามารถแสดงข้อมูลต่าง ๆ ของโปรแกรมได้ และใช้ในการสร้างไดอะลอกบอกซ์ให้ใส่รหัสผ่านได้

ปุ่มคำสั่ง (Command button) เป็นคอนโทรลที่จะทำงานตามที่เรากำหนดเมื่อมีการกดปุ่มคำสั่ง โดยสร้างสิ่งที่จะตอบสนองในอีเวนต์ Click

ออบชั่นบัตตอน (Option button) เป็นตัวเลือกให้แก่ผู้ใช้ โดยผู้ใช้สามารถเลือกออบชั่นบัตตอนที่อยู่บนฟอร์มหรือในเฟรมได้ครั้งละ 1 ตัวเท่านั้น

ลิสต์บอกซ์ (List box) เป็นคอนโทรลที่แสดงรายการให้ผู้ใช้เลือก โดยที่เลือกได้ครั้งละรายการหรือมากกว่าก็ได้

ไทเมอร์ (Timer) ใช้ในการสั่งให้มีการทำงานทุก ๆ ช่วงเวลากำหนด

เลเบล (Label) ใช้ในการแสดงข้อความบนฟอร์ม ซึ่งใช้อธิบายข้อมูลบางอย่างแก่ผู้ใช้ โดยที่ผู้ใช้ไม่สามารถแก้ไขข้อความในเลเบลได้

เช็กบอกซ์ (Check box) ใช้สำหรับเป็นตัวเลือกที่ผู้ใช้สามารถเลือกได้ 2 สถานะ คือ เช็ก (มีเครื่องหมายถูก) หรือไม่เช็ก (ไม่มีเครื่องหมายถูก) โดยที่สามารถเช็กได้หลายเช็กบอกซ์พร้อมกัน

เฟรม (Frame) ใช้ในการจัดกลุ่มคอนโทรลที่ต้องการใช้ร่วมกันเป็นกลุ่ม เช่น ใช้ในการแบ่งกลุ่มออบชั่นบัตตอนออกเป็นกลุ่ม ๆ หรือใช้เพื่อความเรียบร้อยและความสวยงามของหน้าจอของโปรแกรมมากขึ้น

คอมโบบอกซ์ (Combo box) เป็นคอนโทรลที่รวมเอาเท็กบอกซ์กับลิสต์บอกซ์ไว้ด้วยกัน คือสามารถเลือกรายการได้เหมือนลิสต์บอกซ์และสามารถแก้ไขรายการได้เหมือนเท็กบอกซ์

รูปที่ 2.19 แสดงคอนโทรลต่าง ๆ ในทูลบอกซ์ของ VB6

สำหรับคอนโทรลตัวอื่น ๆ และรายละเอียดการทำงานและการนำไปใช้งานของคอนโทรลต่าง ๆ ให้ผู้ที่สนใจไปศึกษาเพิ่มเติมจากคู่มือการใช้งาน VB6

2.5.3.2 การเขียนโปรแกรม

ซึ่งใน VB6 เป็นการกำหนดคุณสมบัติของคอนโทรลบนฟอร์มให้เหมาะสม และการเขียนคำสั่งตอบสนองต่ออีเวนต์

คุณสมบัติ (Properties) คือ ลักษณะต่าง ๆ ของคอนโทรลที่ถูกนำมาวางบนฟอร์ม

สำหรับการเขียนโปรแกรมแบบ Event-Driven คือการใช้คำสั่งกำหนดให้คอนโทรลตอบสนองต่อเหตุการณ์บางอย่างที่เกิดขึ้น (เรียกว่า อีเวนต์: Event) เมื่อโปรแกรมที่สร้างถูกนำมาใช้งาน เช่น เพื่อต้องการให้โปรแกรมเกิดการตอบสนองเมื่อปุ่มคำสั่งถูก Click

2.5.4 พื้นฐานในการเขียนโปรแกรมด้วย VB6

2.6.4.1 ตัวแปรและการประกาศตัวแปร (Variables and Variable Declaration)

ตัวแปรมีหน้าที่เก็บข้อมูลในการทำงานของโปรแกรมไว้เป็นการชั่วคราว ตัวที่เรากำหนดขึ้นมาจะต้องประกอบด้วยชื่อและชนิดข้อมูล

สำหรับการประกาศตัวแปรด้วยคำสั่ง Dim จะตามด้วยชื่อตัวแปร คำว่า As และตามด้วยชื่อชนิดข้อมูล

การประกาศตัวแปร จะเป็นการบอกว่าโปรแกรมมีตัวแปรนี้อยู่เพื่อที่จะนำไปใช้งานในโปรแกรมได้ การประกาศตัวแปรเราจะใช้คำสั่งดังรูปแบบต่อไปนี้

Dim <ชื่อตัวแปร> [As Type]

เราจะต้องกำหนดชื่อของตัวแปร เพื่อที่จะนำไปใช้อ้างอิงตลอดทั้งโปรแกรม สำหรับคำว่า As จะเป็นการบอกว่าเป็นข้อมูลชนิดใด เช่น

Dim TotalPrice As Integer

สำหรับการตั้งชื่อตัวแปรมีกฎดังต่อไปนี้

- ชื่อต้องไม่ซ้ำกันในขอบเขตเดียวกัน
- ชื่อต้องไม่ซ้ำกับคีเวิร์ด (คำสงวน) ของ VB6 เช่น Dim, Integer
- ความยาวไม่เกิน 255 ตัวอักขระ
- จะต้องเริ่มต้นด้วย A-Z หรือ a-z

2.5.4.2 ขอบเขตการประกาศตัวแปร

ขอบเขตการประกาศตัวแปร คือ การกำหนดให้ตัวแปรนั้นสามารถมองเห็นได้ในส่วนใดบ้าง เช่นต้องการให้เข้าถึงตัวแปรได้เฉพาะในโปรแกรมย่อยเท่านั้น หรือให้ทุกโปรแกรมย่อยในโมดูล (เฉพาะฟอร์มเดียว) หรือให้เห็นทั้งโปรเจ็กต์

คำสั่งที่ใช้ในการประกาศตัวแปรจะมีรูปแบบดังนี้

[Private|Public] <ชื่อตัวแปร> [As Type]

สำหรับการประกาศตัวแปรแบบ Private โปรแกรมจะมองเห็นตัวแปรเฉพาะโมดูลเท่านั้น ส่วน Public โปรแกรมจะมองเห็นตัวแปรทั้งโปรเจ็กต์ตัวอย่างเช่น

Private Name1 As String

Public Name2 As String

2.5.4.3 ตัวแปรแบบสเตติก (Static Variables)

หลังจากที่โปรแกรมย่อยทำงานจบแล้วตัวแปรแบบ Dim จะถูกทำลายเมื่อมีการเรียกใช้งานในโปรแกรมย่อยถัดไป ดังนั้นเราสามารถเก็บรักษาค่าตัวแปรนั้นโดยการประกาศตัวแปรแบบสเตติกโดยใช้คำว่า Static แทน Dim ตัวอย่างเช่น

Static Count As Integer

2.5.4.4 ค่าคงที่ (Constant)

ค่าคงที่มีไว้เก็บค่าที่ไม่มีการเปลี่ยนแปลงตลอดทั้งโปรแกรม การประกาศค่าคงที่ทำได้โดยใช้คำสั่งที่มีรูปแบบดังต่อไปนี้

[Private|Public] Const <ชื่อตัวแปร> [As Type]

2.5.4.5 ชนิดของข้อมูล

ชนิดของข้อมูลที่กำหนดให้ตัวแปรเพื่อทำให้การจัดเก็บข้อมูลมีประสิทธิภาพมากขึ้นเพราะใช้พื้นที่ขนาดเหมาะสมในการเก็บข้อมูล โดย VB6 จะมีชนิดข้อมูลดังต่อไปนี้

ตารางที่ 2.1 แสดงชนิดของข้อมูลชนิดต่าง ๆ ใน VB6

ชนิดของข้อมูล	**ขนาด (ไบต์)**	**คำอธิบาย**
Byte	1	มีค่าตั้งแต่ 0-225
Integer	2	ข้อมูลตัวเลขมีค่าตั้งแต่ -32,767 ถึง 32,767
Long	4	ค่าตั้งแต่ -2,147,483,648 ถึง 2,147,483,648
Single	4	เก็บข้อมูลตัวเลขทศนิยมที่มีความละเอียดต่ำ
Double	8	เก็บข้อมูลตัวเลขทศนิยมที่มีความละเอียดสูง
Currency	8	เก็บข้อมูลที่เป็นเลขทศนิยม 4 ตำแหน่งเสมอ
Decimal		เก็บข้อมูลที่เก็บตัวเลขได้ใหญ่มาก
Boolean	1	มีค่า True (-1) กับ False (0)
String	?	ใช้เก็บข้อมูลข้อความที่เป็นชุดอักขระ
Date		ข้อมูลชนิดวันที่ และเวลา
Variant		เป็นชนิดข้อมูลที่ใช้แทนข้อมูลชนิดอื่นได้
Object		เป็นข้อมูลชนิดคอนโทรล

2.5.5 การใช้โอเปอเรเตอร์

2.5.5.1 โอเปอเรเตอร์ในการคำนวณทางคณิตศาสตร์

ตารางที่ 2.2 แสดงโอเปอเรเตอร์ในการคำนวณทางคณิตศาสตร์ของ VB6

การกระทำ	สัญลักษณ์	ตัวอย่าง	ผลลัพธ์
การบวก	+	2+4	6
การลบ	-	4-2	2
การคูณ	*	4*2	8
การหาร	/	5/2	2.5
การหารแบบจำนวนเต็ม	\	5\3	1
การหารเศษ	Mod	10 Mod 3	10 Mod 3
การยกกำลัง	^	2^4	16

2.5.5.2 โอเปอเรเตอร์ในทางตรรกะ

โอเปอเรเตอร์ในทางตรรกะจะให้ผลลัพธ์เป็นค่า True และ False ซึ่งมีรูปแบบดังต่อไปนี้ โอเปอเรเตอร์ And – ถ้าทุกนิพจน์ที่มา And กันมีค่าเป็น True ผลลัพธ์ที่ได้จะมีค่าเป็น True

ตารางที่ 2.3 แสดงผลลัพธ์ที่ได้จากโอเปอเรเตอร์ And

A	B	A And B
True	True	True
True	False	False
False	True	False
False	False	False

โอเปอเรเตอร์ Or - ถ้าทุกนิพจน์ที่มา Or กันมีค่าเป็น True เพียงนิพจน์เดียว ผลลัพธ์ที่ได้จะมีค่าเป็น True

ตารางที่ 2.4 แสดงผลลัพธ์ที่ได้จากโอเปอเรเตอร์ Or

A	B	A Or B
True	True	True
True	False	True
False	True	True
False	False	False

โอเปอเรเตอร์ Xor – โอเปอเรเตอร์นี้จะตรวจสอบว่ามีนิพจน์ที่มีค่าตรงกันหรือไม่ ถ้าไม่ตรงกันผลลัพธ์ที่จะมีค่าเป็น True

ตารางที่ 2.5 แสดงผลลัพธ์ที่ได้จากโอเปอเรเตอร์ Xor

A	B	A Xor B
True	True	False
True	False	True
False	True	True
False	False	False

โอเปอเรเตอร์ Eqv - โอเปอเรเตอร์นี้จะตรวจสอบว่ามีนิพจน์ที่มีค่าตรงกันหรือไม่ ถ้าตรงกันผลลัพธ์ที่จะมีค่าเป็น True

ตารางที่ 2.6 แสดงผลลัพธ์ที่ได้จากโอเปอเรเตอร์ Eqv

A	B	A Eqv B
True	True	True
True	False	False
False	True	False
False	False	True

โอเปอเรเตอร์ Imp – โอเปอเรเตอร์นี้จะมีผลลัพธ์เหมือนกับคำว่า ถ้า.แล้ว. ในเรื่องตรรกะจะมีผลลัพธ์ดังตาราง

ตารางที่ 2.7 แสดงผลลัพธ์ที่ได้จากโอเปอเรเตอร์ Imp

A	B	A Imp B
True	True	True
True	False	False
False	True	True
False	False	True

โอเปอเรเตอร์ Not – จะทำการเปลี่ยนค่านิพจน์เป็นค่าตรงข้าม

ตารางที่ 2.8 แสดงผลลัพธ์ที่ได้จากโอเปอเรเตอร์ Not

A	**Not A**
True	False
False	True

2.5.5.3 โอเปอเรเตอร์ในการทำงานเกี่ยวกับข้อมูลชนิดสตริง

โอเปอเรเตอร์กลุ่มนี้จะใช้เชื่อม String กับ String เข้าด้วยกัน หรือ String กับข้อมูลตัวเลข+ ใช้เชื่อม String กับ String

& ใช้เชื่อม String กับ Numeric หรือ String กับ String ก็ได้

1. โอเปอเรเตอร์ในการเปรียบเทียบ

โอเปอเรเตอร์ประเภทนี้จะใช้สำหรับเปรียบเทียบระหว่างค่า 2 ค่า โดยมีผลลัพธ์เป็น True หรือ False อย่างใดอย่างหนึ่งเท่านั้น

ตารางที่ 2.9 แสดงโอเปอเรเตอร์ที่ใช้ในการเปรียบเทียบ

สัญลักษณ์	ความหมาย	รูปแบบการใช้
=	เท่ากับ	A = B
<>	ไม่เท่ากับ	A <> B
<	น้อยกว่า	A < B
>	มากกว่า	A > B
<=	น้อยกว่าหรือเท่ากับ	A <= B
>=	มากกว่าหรือเท่ากับ	A >= B

2.ฟังก์ชันในการเปลี่ยนชนิดข้อมูล

ในโปรแกรม VB6 ได้เตรียมฟังก์ชันในการเปลี่ยนแปลงข้อมูลจากข้อมูลชนิดหนึ่งไปเป็นข้อมูลอีกชนิดหนึ่ง เพื่ออำนวยความสะดวกในการทำงานกับข้อมูลต่าง ๆ ดังนี้

ตารางที่ 2.10 แสดงฟังก์ชันการเปลี่ยนชนิดข้อมูล

ชื่อฟังก์ชั่น	ไปเป็นข้อมูลชนิด	ตัวอย่างการใช้งาน	ผลลัพธ์
CBool	Boolean	CBool(“True”), CBool(1=0)	True, False
CByte	Byte	CByte(“234.123”)	234
CInt	Integer	CInt(“3000.457”)	3000
CLng	Long	CLng(“10000000”)	10000000
CCur	Currency	CCur(“4598.8976”)	4598.8976
CSng	Single	CSng(“10000.7117”)	10000.71
CDbl	Double	CDbl(“10000.7117”)	10000.7117
CDate	Date	CDate(“7 Jan 98”)	1/7/98
CStr	String	CStr(975.7773)	“975.7773”
CVar	Variant		
CVErr	Error		

2.5.6 คำสั่งที่ใช้ในการควบคุมการทำงาน

ในการทำงานของโปรแกรมจะทำงานจากบนลงล่าง และจากซ้ายไปขวา แต่ในบางครั้งเราสามารถเลือกให้โปรแกรมทำงานตามเงื่อนไข หรือในบางอย่างต้องการการทำงานซ้ำกัน หลาย ๆ ครั้ง ซึ่งคำสั่งจะถูกทำซ้ำจนครบตามจำนวนครั้งที่กำหนด หรือทำจนกว่าเงื่อนไข ที่ระบุไว้เป็นจริง

2.5.6.1 คำสั่งในการเลือกเส้นทางการทำงาน

คำสั่งที่ใช้เลือกเส้นทางการทำงานของโปรแกรม จะทำให้สามารถเลือกเส้นทางการทำงานตามเงื่อนไขที่อยู่ในโปรแกรมได้ คำสั่งประเภทนี้มี 2 แบบ คือ

- คำสั่ง If-Then-Else

มีรูปแบบคำสั่งดังนี้

```
If <นิพจน์ที่เป็น Boolean> Then
        ' คำสั่งสำหรับนิพจน์ที่มีค่าเป็น True
Else
        ' คำสั่งสำหรับนิพจน์ที่มีค่าเป็น False
End If
```

คำสั่งข้างต้น จะเป็นคำสั่งที่ตรวจสอบเงื่อนไขของคำสั่งว่า If ถ้าเงื่อนไขเป็น True จะทำตามคำสั่งที่อยู่หลังคำว่า Then แต่ถ้าเงื่อนไขเป็น False จะทำตามคำสั่งที่อยู่หลังคำว่า Else จนถึงคำว่า End If แต่ถ้าไม่ต้องการให้มีการทำงานในส่วนของ Else เราสามารถย่อคำสั่ง ได้ดังนี้

```
If <นิพจน์ที่เป็น Boolean> Then
        ' คำสั่งสำหรับนิพจน์ที่มีค่าเป็น True
End If
```

นอกจากนี้ยังสามารถซ้อนคำสั่ง If ได้ด้วยเพื่อการตรวจสอบเงื่อนไขที่ซับซ้อนขึ้นดังตัวอย่าง

```
If <นิพจน์ที่เป็น Boolean 1> Then
        ' คำสั่งสำหรับนิพจน์ 1 ที่มีค่าเป็น True
ElseIf <นิพจน์ที่เป็น Boolean 2> Then
        ' คำสั่งสำหรับนิพจน์ 2 ที่มีค่าเป็น True
ElseIf <นิพจน์ที่เป็น Boolean 3> Then
        ' คำสั่งสำหรับนิพจน์ 3 ที่มีค่าเป็น True
Else
```

’ คำสั่งสำหรับทั้ง 3 นิพจน์ที่มีค่าเป็น False

End If

- คำสั่ง Select-Case

เป็นคำสั่งที่ใช้ในการเลือกเส้นทางการทำงานของโปรแกรมจากค่าของนิพจน์ที่กำหนด เป็นคำสั่งที่มีรูปแบบดังนี้

Select Case <นิพจน์>

Case <ค่าของนิพจน์กลุ่มแรก>

’คำสั่งทำงานเมื่อค่านิพจน์ตรงกับกลุ่มแรก

Case <ค่าของนิพจน์กลุ่มสอง>

’คำสั่งทำงานเมื่อค่านิพจน์ตรงกับกลุ่มที่สอง

Case Else

’คำสั่งเมื่อไม่ตรงกับกรณีทั้งสอง

End Select

คำสั่ง Select-Case นั้นจะทำงานตามค่าที่อยู่หลังคำว่า Select Case ถ้าตรงกับค่าที่อยู่หลังคำว่า Case ใดก็จะทำงานตามคำสั่งที่อยู่หลัง Case นั้น แต่ถ้าไม่ตรงกับค่าใดเลยจะทำงานตามคำสั่งที่อยู่หลังคำว่า Case Else จนถึงคำว่า End Select

2.5.6.2 คำสั่งที่ใช้ในการทำซ้ำ

คำสั่งที่ใช้ในการทำงานซ้ำเป็นจำนวนครั้งที่แน่นอน หรือจนกว่าเงื่อนไขเป็นไปดังที่เราต้องการได้ มีดังนี้

- For-Next

คำสั่งนี้จะในการทำงานซ้ำเป็นจำนวนครั้งที่แน่นอน

For <ชื่อตัวแปร> = <ค่าเริ่มต้น> To <ค่าสุดท้าย> [Step <ค่าที่เพิ่มขึ้นของตัวแปร>]

’ชุดคำสั่ง

[Exit For]

Next <ชื่อตัวแปร>

การทำงานของคำสั่งนี้ จะทำงานโดยมีตัวแปรตัวหนึ่งนับจำนวนครั้งการทำซ้ำโดยการกำหนดค่าเริ่มต้น ค่าสุดท้าย และค่าที่เพิ่มขึ้นในแต่ละครั้ง เมื่อทำงานจบชุดคำสั่งมาถึง Next <ชื่อตัวแปร> ก็จะทำการเพิ่มค่าตัวแปรขึ้นไปเท่ากับค่าที่หลังคำว่า Step และจะทำซ้ำจนกว่าค่าของตัวแปรจะมีค่ามากที่สุดที่มีการกำหนดไว้ ถ้ามีคำสั่ง Exit For จะทำให้การทำงานหลุดออกจากลูปทันที

- Do-Loop

เป็นคำสั่งที่ใช้ในการทำซ้ำ โดยการทำซ้ำขึ้นอยู่กับเงื่อนไขหลังคำว่า While หรือUntil รูปแบบคำสั่งมีดังนี้

Do While <Condition>

'ชุดคำสั่ง

[Exit Do]

Loop

คำสั่งนี้จะทำซ้ำขณะที่เงื่อนไขยังคงเป็นจริงอยู่ จนกว่าจะเป็นเท็จจริงออกจากลูปถ้ามีคำสั่ง Exit Do จะออกจากการทำซ้ำทันที

นอกจากนี้ยังมีคำสั่งอีกรูปคือ

Do Until <Condition>

'ชุดคำสั่ง

[Exit Do]

Loop

คำสั่งนี้จะเหมือนกับ Do While.Loop แต่ต่างกันตรงที่จะทำซ้ำในขณะที่ Condition ยังคงเป็นเท็จอยู่ ทำจนกว่าเงื่อนไขเป็นจริงจึงจะออกจากลูป

2.5.7 การทำงานกับออบเจ็กต์

เนื่องจากทุก ๆ ส่วนประกอบใน VB6 นั้นเป็นออบเจ็กต์ทั้งหมด ไม่ว่าจะเป็นฟอร์ม คอนโทรลหรือแม้กระทั่งการติดต่อกับฐานข้อมูล ดังนั้นถ้าเราต้องการใช้ VB6 ให้เต็มประสิทธิภาพ จำเป็นต้องเข้าใจการใช้งานออบเจ็กต์

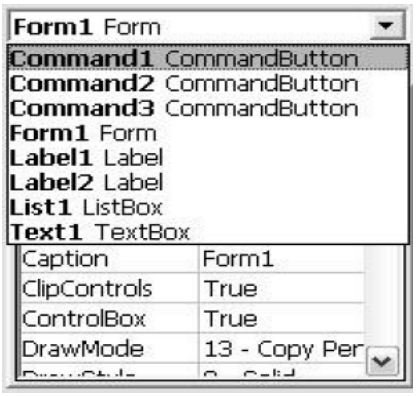
2.5.7.1 ออบเจ็กต์ใน VB6

ตัวอย่างออบเจ็กต์ใน VB6 คือ คอนโทรลต่าง ๆ ซึ่งออบเจ็กต์ประกอบด้วยสองส่วน คือ

- Data จะเก็บข้อมูลต่าง ๆ ของออบเจ็กต์ไว้ ซึ่งก็คือ Properties นั่นเอง
- Code จะเก็บคำสั่งต่าง ๆ ที่จะให้ออบเจ็กต์นั้นทำงาน

สรุปก็คือ ออบเจ็กต์เป็นการรวมคำสั่งและข้อมูลไว้ด้วยกัน

ออบเจ็กต์นั้นสร้างมาจากคลาส ซึ่งคลาสก็คือ การสร้างรายละเอียดต่าง ๆ ของออบเจ็กต์ที่จะสร้าง เช่น กำหนดว่าจะมีคุณสมบัติอะไรบ้าง, มีเมดถอดที่ทำงานอย่างไร และมีอีเวนต์ว่าจะเกิดขึ้นในขณะใด สรุปก็คือเป็นการกำหนดพฤติกรรมของออบเจ็กต์ที่จะสร้างขึ้นมานั่นเอง ตัวอย่างความสัมพันธ์ระหว่างออบเจ็กต์และคลาส คือ คอนโทรลที่อยู่ในทูลบอกซ์นั้นจะแทนคลาส เมื่อสร้างคอนโทรลนั้นบนฟอร์มก็จะเป็นการสร้างอินสแตนซ์ (Instance) ของคลาส ซึ่งก็คือออบเจ็กต์ ดังในรูป Properties Window ดังรูปที่ 2.20



รูปที่ 2.20 แสดง Properties Window

ออบเจ็กต์ต่าง ๆ ที่สร้างขึ้นจากคลาสเดียวกันจะเป็นออบเจ็กต์ที่ไม่เกี่ยวกัน แต่จะมีคุณสมบัติ เมดถอด และอีเวนต์เหมือนกันเพราะมาจากคาสเดียวกัน เช่น Command1 และ Command 2 สร้างขึ้นจากคลาสเดียวกัน มีคุณสมบัติ เมดถอด และอีเวนต์เหมือนกัน

2.5.7.2 การทำงานกับคุณสมบัติของออบเจ็กต์

สำหรับคุณสมบัติของออบเจ็กต์นั้น จะเป็นการกำหนดลักษณะของออบเจ็กต์ให้ตรงกับสิ่งที่จะใช้ เช่น คุณสมบัติ Text ของเท็กบอกซ์ เป็นต้น

- การกำหนดค่าให้คุณสมบัติ

จะใช้รูปแบบดังต่อไปนี้ในการกำหนดค่าคุณสมบัติของออบเจ็กต์

<ชื่อออบเจ็กต์>.<ชื่อคุณสมบัติ> = ค่าที่กำหนดให้

ตัวอย่างเช่น

Command1.Caption = “Click”

Command1.Enabled = True

Command1.Visible = True

- การอ่านค่าคุณสมบัติของออบเจ็กต์

จะใช้รูปแบบดังต่อไปนี้ในการอ่านค่าคุณสมบัติของออบเจ็กต์

ตัวแปรที่จะรับค่า = <ชื่อออบเจ็กต์>.<ชื่อคุณสมบัติ>

เราอาจจะใช้คุณสมบัติมาเป็นส่วนประกอบในนิพจน์ในการคำนวณ ตัวอย่างเช่น

Str = Command1.Caption

Bol =Command1.Enabled And False

2.5.7.3 การใช้งานเมดถอดของออบเจ็กต์

เมดถอดของคอนโทรลบางเมดถอดสามารถแก้ไขค่าคุณสมบัติของคอนโทรล และบางเมดถอดสามารถส่งค่ากลับมาเหมือนกับฟังก์ชัน โดยสามารถเรียกใช้เมดถอดได้ในลักษณะเดียวกันกับการเรียกใช้โปรแกรมย่อย หรือฟังก์ชัน ซึ่งรวมทั้งการส่งพารามิเตอร์ให้กับเมดถอดด้วยสำหรับการทำงานกับเมดถอดจะใช้รูปแบบดังต่อไปนี้

<ชื่อออบเจ็กต์>.<ชื่อเมดถอด> 'สำหรับเรียกใช้เมดถอดที่ไม่ส่งค่ากลับมา

ตัวแปรที่รับค่า=<ชื่อออบเจ็กต์>.<ชื่อเมดถอด> 'สำหรับเมดถอดที่ส่งค่ากลับมา

2.5.7.4 การประกาศออบเจ็กต์

การประกาศตัวแปรออบเจ็กต์สามารถทำได้เช่นเดียวกับตัวแปรทั่วไป ดังรูปแบบต่อไปนี้

[Dim|ReDim|Static|Private|Public] <ชื่อตัวแปร> As [New] <ชื่อคลาส>

เราสามารถประกาศตัวแปรออบเจ็กต์ที่อ้างถึงฟอร์มใด ๆ ในโปรแกรมได้ด้วยคำสั่งดังนี้

```
Dim ObjForm As Form
```

เราสามารถประกาศตัวแปรออบเจ็กต์ที่อ้างถึงคอนโทรลใด ๆ ในโปรแกรมได้ด้วยคำสั่งดังนี้

```
Dim ObjCtrl As Control
```

หรือจะประกาศเจาะจงลงไปถึงคอนโทรลชนิดใด เช่น

```
Dim ObjTxt As TextBox
Dim ObjLstBox As ListBox
```

เนื่องจากเราไม่สามารถประกาศตัวแปรแล้วอ้างอิงถึงคอนโทรลนั้นแบบเจาะจงได้ แต่จะใช้การอ้างถึงคอนโทรลนั้นภายหลัง ดังตัวอย่าง

```
Dim ObjTxt As TextBox
Set ObjTxt = Text1
```

หรือ

```
Dim ObjCtrl As Control
Set ObjCtrl = Form1.Text1
```

2.5.7.5 การสร้างออบเจ็กต์ใหม่

ออบเจ็กต์เป็นอินสแตนซ์ของคลาส ดังนั้นถ้าต้องการสร้างออบเจ็กต์ใหม่จากคลาส ให้ใช้คำว่า New ในการประกาศตัวแปร ตัวอย่างเช่น

```
Dim ObjFrm As New Form1
ObjFrm.Show
```

จากตัวอย่างข้างต้นเป็นการประกาศตัวแปรออบเจ็กต์ที่สร้างสำเนาของฟอร์มชื่อ Form1ใหม่ ซึ่งมีทุกอย่างเหมือนกับ Form1 ทั้งหมด นอกจากการสร้างออบเจ็กต์ใหม่ในการประกาศตัวแปรแล้ว สามารถใช้คำสั่ง Set แล้วตามด้วยคีเวิร์ด New แทนได้ด้วย ดังตัวอย่าง

```
Dim ObjFrm As Form1
Set ObjFrm = New Form1
ObjFrm.Show
```

2.5.7.6 การยกเลิกการอ้างถึงออบเจ็กต์

เมื่อเราใช้ออบเจ็กต์ต่าง ๆ ทำงานจนเสร็จแล้ว จะต้องยกเลิกการอ้างถึงออบเจ็กต์ต่าง ๆ เหล่านี้ เพื่อให้ทรัพยากรต่าง ๆ ที่ใช้สำหรับออบเจ็กต์เหล่านั้นถูกคืนกลับสู่ระบบเพื่อนำไปใช้อย่างอื่นต่อได้ จะมีรูปแบบคำสั่งดังนี้

```
Set ObjCtrl = Nothing
```

2.5.7.7 คำสั่งควบคุมการทำงานเกี่ยวกับออบเจ็กต์

- คำสั่ง For Each… Next

เป็นคำสั่งที่คล้ายกับ For… Next แต่คำสั่งนี้จะวนตามจำนวนสมาชิกที่อยู่ในอาร์เรย์ หรือคอนเลคชั่น

```
For Each <Element> In <Group>
    'ชุดคำสั่ง
Next <Element>
```

ตัวอย่างเช่น

```
Sub Form_Load()
    Dim Ctrl As Control
    For Each Ctrl In Form1.Controls
        Debug.Print Ctrl.Name
    Next Ctrl
End Sub
```

- คำสั่ง With…End With

คำสั่งนี้จะช่วยให้เราทำงานกับออบเจ็กต์ได้สะดวกขึ้น โดยที่ไม่ต้องอ้างอิงถึงชื่อออบเจ็กต์นั้นบ่อย ๆ

```
With <ObjectName>
    'ชุดคำสั่ง
```

```
End With
```

ตัวอย่างเช่น

```
With Command1
    .Caption = “Click Me”
    .Left = 400
    .Enabled = True
End With
```

2.5.8 การเขียนโปรแกรมเพื่อใช้งานพอร์ตอนุกรมด้วย Visual Basic

2.5.8.1 คอนโทรล MSComm

MSComm จัดเตรียมทางเลือกไว้สองทางเพื่อความสะดวกในการสื่อสารข้อมูล ทางแรกคือการสื่อสารข้อมูลที่กระตุ้นด้วยเหตุการณ์ (Event-driven communications) เป็นรูปแบบการใช้งานที่มีประสิทธิภาพมากสำหรับการตอบสนองแบบทันทีทันใด ส่วนทางเลือกที่สองเป็นการคอยตรวจสอบค่าเหตุการณ์และความผิดพลาดที่เกิดขึ้นด้วยการดูค่าที่เปลี่ยนแปลงภายในคุณสมบัติ CommEvent หลังจากที่โปรแกรมทำงานในฟังก์ชันต่าง ๆ เรียบร้อยแล้ว ซึ่งวิธีนี้ใช้งานได้ดีในกรณีที่โปรแกรมมีขนาดเล็ก

คอนโทรล MSComm 1 ตัวสามารถควบคุมการทำงานของพอร์ตอนุกรมได้ 1 พอร์ต ถ้าในโปรแกรมที่ใช้งานต้องการติดต่อกับพอร์ตอนุกรมมากกว่า 1 พอร์ต จะต้องใช้คอนโทรล MSComm มากกว่า 1 ตัวเพื่อควบคุมการทำงานในแต่ละพอร์ต แอดเดรสของอนุกรมและแอดเดรสของการเกิดอินเตอร์รัพต์สามารถเปลี่ยนแปลงได้จากการแก้ไขค่าที่ Control Panel

ถึงแม้ว่าคอนโทรล MSComm จะมีคุณสมบัติ (Properties) มากมาย แต่สามารถทำความเข้าใจไม่ยากดังนี้

2.5.8.2 CommPort

ใช้ในการกำหนดและอ่านค่าพอร์ตอนุกรมที่ติดต่ออยู่ (COM1, COM2, COM3, COM4) รูปแบบการใช้งาน

```
Object.CommPort [= value]
```

โดย value เป็นค่าของพอร์ตอนุกรม ชนิดของข้อมูลเป็น Integer ค่า value สามารถกำหนดได้ในช่วง 1-16 (ค่าเริ่มต้นที่กำหนดไว้ที่ 1) เมื่อมีการกำหนดค่าแล้วทำการเปิดพอร์ตโดยใช้คุณสมบัติ PortOpen แต่ว่าพอร์ตนั้นไม่มีอยู่ในระบบ MSComm จะสร้างสัญญาณแสดงข้อผิดพลาด error 68 ขึ้นมา ซึ่งหมายถึง อุปกรณ์ตัวนี้ไม่มีอยู่ในระบบ ดังนั้นการเขียนโปรแกรมจึงต้องกำหนดตำแหน่งของพอร์ตอนุกรมก่อนที่ใช้คำสั่ง OpenPort

2.5.8.3 Setting

ในการกำหนดและอ่านค่าอัตราบอร์ด, พาริตี้, จำนวนของบิดข้อมูล, จำนวนบิตปิดท้าย รูปแบบการใช้งาน

Object.Setting [= value]

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

ค่าบอร์ดเรตมาตรฐานที่ใช้กับ MSComm มีดังนี้

110 บิตต่อวินาที

300 บิตต่อวินาที

600 บิตต่อวินาที

1200 บิตต่อวินาที

2400 บิตต่อวินาที

9600 บิตต่อวินาที (ค่าปกติ)

14400 บิตต่อวินาที

19200 บิตต่อวินาที

28800 บิตต่อวินาที

38400 บิตต่อวินาที (สงวน)

56000 บิตต่อวินาที (สงวน)

128000 บิตต่อวินาที (สงวน)

256000 บิตต่อวินาที (สงวน)

สำหรับค่ามาตรฐานในการกำหนดค่าพาริตี้มีดังนี้

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

ค่าที่ใช้ในการกำหนดจำนวนบิตมี 5 ค่า คือ 4, 5, 6, 7 และ 8 (ค่าปกติ)

ค่าที่ระบุจำนวนบิตปิดท้ายมีค่า คือ 1 (ค่าปกติ),1.5 และ 2

ตัวอย่างการใช้งานคำสั่ง Setting โดยจะเป็นการกำหนดค่าบอร์ดเรตเท่ากับ 9600 ไม่มีพาริตี้ จำนวนบิตข้อมูล 8 บิต และบิตปิดท้าย 1 บิต สามารถโปรแกรมได้ดังนี้

MSComm1.Setting = “9600,N,8,1”

สาเหตุที่ค่ากำหนดจะต้องอยู่ในเครื่องหมายคำพูด เนื่องจากค่าที่กำหนดนี้อยู่ในรูปตัวแปรString

2.5.8.4 PortOpen

ในการกำหนดและอ่านค่าสถานะของพอร์ตอนุกรม เพื่อเปิดและปิดพอร์ตอนุกรมรูปแบบการใช้งาน

Object.PortOpen [= value]

ค่า value มีชนิดข้อมูลเป็นแบบบูลีน คือ True กับ False โดย True หมายถึงการเปิดพอร์ตอนุกรม และ False หมายถึงการปิดพอร์ตอนุกรม สำหรับการปิดพอร์ตอนุกรมนั้นจะมีการเคลียร์บัฟเฟอร์รับข้อมูลและบัฟเฟอร์ส่งข้อมูลด้วย คอนโทรล MSComm จะปิดพอร์ตอนุกรมโดยอัตโนมัติเมื่อออกจากโปรแกรม ก่อนที่จะใช้คุณสมบัติ PortOpen ต้องตรวจสอบให้แน่ใจว่าคุณสมบัติ CommPort นั้นได้ทำการกำหนดตำแหน่งของพอร์ตอนุกรมไว้ถูกต้องแล้ว มิเช่นนั้น MScomm จะแสดงข้อผิดพลาด Error 68 แจ้งแก่ผู้ใช้งาน หรือถ้าพอร์ตอนุกรมนั้นถูกเปิดไว้แล้วโปรแกรมก็จะแสดงข้อผิดพลาดเช่นเดียวกัน

ถ้าคุณสมบัติ DTREnabled หรือ RTSEnabled ถูกกำหนดไว้เป็น True ก่อนที่จะทำการเปิดพอร์ต ค่าคุณสมบัติของ DTREnabled หรือ RTSEnabled จะถูกเซตเป็น False หลังจากปิดพอร์ต แต่ถ้าเซตเป็น False หลังจากการปิดโปรแกรมแล้ว ค่าที่กำหนดu3652 ไว้จะเป็นค่าเดิมดังตัวอย่างการใช้คำสั่งเปิดพอร์ต เพื่อติดต่อสื่อสารกับพอร์ตอนุกรม COM1 และมีบอร์ดเรต 9600 บิตต่อวินาที ไม่มีพาริตี้ จำนวนข้อมูล 8 บิต และบิตปิดท้าย 1 บิต ดังนี้

MSComm1.Setting = “9600,N,8,1”

MSComm1.Commport = 1

MSComm1.PortOpen = True

2.5.8.5 Input

อ่านค่าและลบค่าของข้อมูลจากบัฟเฟอร์ภาครับ

รูปแบบการใช้งาน

Object.Input

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

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

ตัวอย่างโปรแกรมแสดงให้เห็นวิธีในการรับข้อมูลจากบัฟเฟอร์โดยรับข้อมูลทั้งหมด

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

2.5.8.6 InBufferCount

ส่งค่าจำนวนของตัวอักษรที่อยู่ในบัฟเฟอร์ภาครับ

รูปแบบการใช้งาน

```
Object.InBufferCount [= value]
```

คำสั่งนี้จะแสดงค่าจำนวนของอักษร ซึ่งรับมาจากภายนอกและยังเก็บอยู่ในบัฟเฟอร์ภาครับ เพื่อให้ผู้ใช้งานอ่านค่าออกไป สำหรับการเคลียร์ค่าบัฟเฟอร์ภาครับทำได้โดยกำหนดให้ InBufferCount มีค่าเป็น 0

2.5.8.7 InBufferSize

กำหนดและคืนค่าขนาดของบัฟเฟอร์ภาครับในหน่วยเป็นไบต์

รูปแบบการใช้งาน

```
Object.InBufferSize [= value]
```

คำสั่งนี้ใช้เพื่อกำหนดขนาดของบัฟเฟอร์ภาครับ โดยค่าเริ่มต้นถูกกำหนดไว้ที่ 1024 ไบต์ การกำหนดค่าบัฟเฟอร์ภาครับขนาดใหญ่จะทำให้หน่วยความจำที่เหลือสำหรับการใช้งานส่วนอื่นๆ จะเหลือน้อย อย่างไรก็ตามการกำหนดค่าบัฟเฟอร์ภาครับน้อยเกินไปจะทำให้เกิดโอเวอร์โฟลวหรือข้อมูลล้นบัฟเฟอร์ เว้นแต่จะมีการใช้แฮนด์เช็ก ดังนั้นค่าปานกลางที่เหมาะสมก็คือค่า 1024 ไบต์ ซึ่งเป็นค่าเริ่มต้นนั่นเอง ถ้าโปรแกรมมีการเกิดโอเวอร์โฟลวแล้วจึงค่อยปรับค่าขนาดของบัฟเฟอร์ให้มีค่ามากขึ้น

2.5.8.8 InputLen

กำหนดค่าและคืนค่าจำนวนของตัวอักษรที่อ่านจากบัฟเฟอร์ภาครับ

รูปแบบการใช้งาน

Object.InputLen [= value]

ค่าเริ่มต้นของคุณสมบัติ InputLen มีค่าเท่ากับ “0” การกำหนดค่าเท่ากับ “0” จะทำให้คำสั่ง Input ของ MSComm อ่านค่าข้อมูลภายในบัฟเฟอร์ภาครับทั้งหมด

ถ้าไม่มีข้อมูลอยู่ในบัฟเฟอร์ภาครับมากเท่ากับ InputLen คำสั่ง Input จะส่งค่าว่าง (“”) กลับออกมา ผู้ใช้งานสามารถตรวจสอบข้อมูลในบัฟเฟอร์ภาครับก่อนแล้วจึงค่อยอ่านค่าข้อมูลจากบัฟเฟอร์ภาครับได้โดยใช้คุณสมบัติ InBufferCount โดยกำหนดให้มีข้อมูลในบัฟเฟอร์ภาครับก่อนแล้วจึงค่อยอ่านข้อมูลจากบัฟเฟอร์ภาครับ

คุณสมบัตินี้มักใช้กับอ่านค่าข้อมูลจาก เครื่องมือ หรือ เครื่องจักร ที่มีการกำหนดค่า ขนาดความยาวของข้อมูลเอาไว้แล้ว

ตัวอย่างโปรแกรมการอ่านค่าตัวอักษรออกมา 10 ตัวอักษร

```
Private Sub Command1_Chick ()
Dim CommData As String
    MSComm1.InputLen = 10 ’Specify a 10 character block of data
    CommData = MSComm1.Input ’Read data
End Sub
```

2.5.8.9 InputMode

กำหนดค่าและคืนค่าชนิดของข้อมูลที่รับโดยคำสั่ง Input

รูปแบบการใช้งาน

Object.InputMode [= value]

คุณสมบัติ InputMode ใช้กำหนดว่าข้อมูลชนิดไหนที่รับเข้ามาผ่านคำสั่ง Input โดยข้อมูลจะเลือกได้ 2 ประเภท คือ

comInputModeText สำหรับข้อมูลที่อยู่ในรูปข้อความตัวอักษรตามมาตรฐาน ANSI โดยจะต้องกำหนดค่าเป็น “0” และค่าเริ่มต้นของการรับค่าของข้อมูลก็จะเป็นค่านี้

comInputModeBinary สำหรับข้อมูลอื่น ๆ ซึ่งจะเก็บในรูปไบนารี่รวมกันอยู่เป็นไบต์ข้อมูลและเก็บข้อมูลไว้ในตัวแปรแบบอาร์เรย์ ชนิดของข้อมูลเป็นแบบไบต์

```
Private Sub Command1_Click ()
Dim Buffer As Variant
Dim Arr () As Byte
```

```
MSComm1.CommPort = 1 'Set and open port
MSComm1.PortOpen = True
MSComm1.InputMode = comInputModeBinary 'Set to read binary data
Do Until MSComm1.InBufferCount<10 'Wait until 10 byte
    DoEvents
Loop
Buffer = MSComm1.Input 'Store binary data in buffer
Arr = Buffer 'Assign to byte array for processing
End Sub
```

2.5.8.10 Output

ใช้ในการส่งขบวนของข้อมูลไปยังบัฟเฟอร์ส่งข้อมูล

รูปแบบการใช้งาน

```
Object.Output [= value]
```

ค่า value เป็นค่าของตัวอักษรที่เขียนไปยังบัฟเฟอร์ส่งข้อมูล คุณสมบัติ Output สามารถใช้ในการส่งข้อมูลตัวอักษรหรือข้อมูลไบนารี่ก็ได้ โดยในการส่งข้อมูลเป็นรูปแบบตัวอักษรจะต้องกำหนดข้อมูลเป็นแบบ Variant และมีข้อมูลภายในเป็นแบบ String สำหรับการส่งข้อมูลไบนารี่จะต้องกำหนดชนิดข้อมูลเป็นแบบ Variant และมีข้อมูลเป็นแบบ Byte

ตัวอย่างโปรแกรมการส่งค่าที่ป้อนจากคีย์บอร์ดไปยังพอร์ตอนุกรม

```
Private Sub Command_Click ()
Dim Buffer As Variant
    MSComm1.CommPort = 1
    MSComm1.PortOpen = True
    Buffer = Chr$(KeyAscii)
    MSComm1.Output = Buffer 'Send data
End Sub
```

2.5.8.11 OutputBufferCount

คืนค่าจำนวนของข้อมูลตัวอักษร ที่เก็บอยู่ในบัฟเฟอร์ ภาคส่ง และสามารถใช้คำสั่งนี้ เพื่อเคลียร์บัฟเฟอร์ภาคส่งได้ด้วย

รูปแบบการใช้งาน

```
Object.OutBufferCount [= value]
```

ผู้ใช้งานสามารถเคลียร์บัฟเฟอร์ภาคส่งได้โดยการกำหนดค่า OutBufferCount เท่ากับ “0”

2.5.8.12 OutputBufferSize

กำหนดค่าและคืนค่าขนาดของบัฟเฟอร์ภาคส่ง ชนิดตัu3623 วแปรเป็นแบบไบต์

รูปแบบการใช้งาน

Object. OutputBufferSize [= value]

คุณสมบัติ OutputBufferSize ใช้สำหรับกำหนดขนาดของบัฟเฟอร์ภาคส่ง โดยค่าปกติที่ใช้งานจะมีค่าเท่ากับ 512 ไบต์

การกำหนดค่าบัฟเฟอร์ภาคส่งที่มากเกินไปจะทำให้มีหน่วยความจำเหลือให้ใช้งานน้อย แต่ถ้าน้อยเกินไปจะทำให้เกิดข้อมูลล้นบัฟเฟอร์ ยกเว้นจะมีการใช้ แฮนด์เช็ก วิธีการที่ถูกต้องในการกำหนดค่าคือ ทดลองเริ่มใช้ค่าเริ่มต้นคือ 512 ไบต์ดูก่อน ถ้าข้อมูลล้นบัฟเฟอร์ก็ค่อยเพิ่มค่า OutputBufferSize ให้มากขึ้น

2.5.8.13 ParityReplace

กำหนดและคืนค่าตัวอักษรที่ไปแทนในตำแหน่งที่เกิดข้อผิดพลาดจากพาริตี้

รูปแบบการใช้งาน

Object.ParityReplace [= value]

บิตพาริตี้เป็นบิตที่ทางภาคส่งข้อมูลทำการส่งมาพร้อมกับข้อมูล เพื่อตรวจสอบข้อผิดพลาดของข้อมูล โดยเมื่อมีการใช้บิตพาริตี้ คอนโทรล MScomm จะทำการบวกบิตทุกบิตที่มีค่าลอจิก “1” ในแต่ละไบต์และทำการตรวจสอบผลลัพธ์ว่าบิตที่อ่านได้นั้นมีจำนวนลอจิก “1” เป็นเลขคู่หรือคี่และตรงกับค่าที่กำหนดไว้แต่ต้นหรือไม่ ถ้าไม่ตรงแสดงว่าการรับส่งข้อมูลผิดพลาด

การกำหนดค่าเริ่มต้นให้กับ ParityReplace นั้นกำหนดให้ใช้เครื่องหมาย (?) ไปวางไว้ที่ตำแหน่งที่เกิดพาริตี้ผิดพลาด ถ้ากำหนดค่า ParityReplace ให้เป็นค่าว่าง (“”) จะเป็นการยกเลิกการใช้งาน ParityReplace และไม่มีการป้อนข้อมูลแทนเมื่อตรวจพบข้อผิดพลาด

ParityReplace ใช้ชนิดข้อมูลเป็นแบบสตริง แต่การกำหนดจะกำหนดได้เพียงไบต์เดียวเท่านั้น ซึ่งสามารถใช้ค่าต่าง ๆ ก็ได้ ที่เป็นโค้ด ANSI มีค่าอยู่ระหว่าง 0-255

2.5.8.14 DTREnable

ใช้ในการกำหนดสถานะลอจิกของขา Data Terminal Ready (DTR) โดยสัญญาณของขา DTR จะส่งจากคอมพิวเตอร์ไปยังโมเด็มเพื่อแสดงว่าคอมพิวเตอร์พร้อมที่จะรับข้อมูลแล้ว ชนิดของข้อมูลเป็นแบบบูลีน

รูปแบบการใช้งาน

Object.DTREnable [= value]

เมื่อขา DTR ถูกกำหนดสถานะให้เป็น True ที่ขา DTR จะมีสถานะลอจิก “1” เมื่อทำการเปิดพอร์ต และจะมีสถานะลอจิก “0” เมื่อมีการปิดพอร์ต เมื่อขา DTR ถูกกำหนดสถานะเป็น Falseที่ขา DTR จะมีสถานะลอจิก “0” ตลอดเวลา

2.5.8.15 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”

เมื่อขา RTS ถูกกำหนดให้เป็น True ขา RTS จะมีสถานะลอจิก “1” เมื่อเปิดพอร์ตและมีสถานะลอจิก “0” เมื่อปิดพอร์ต

2.5.8.16 EOFEnable

เป็นการกำหนดให้ MSComm รอสัญลักษณ์แสดงส่วนท้ายสุดของไฟล์ระหว่างการรับอินพุตเข้ามา ถ้าพบสัญลักษณ์ EOF ภาคอินพุตจะหยุดรับข้อมูล และเหตุการณ์ OnComm จะถูกกระตุ้นให้ทำงาน คุณสมบัติ CommEvent จะมีค่าเท่ากับ 7 หรือ CommEvEOF

รูปแบบการใช้งาน

Object.EOFEnable [= value]

โดย value เป็นค่าสถานะ True หรือ False เพื่ออีนาเบิลหรือดิสเอเบิลการทำงานของเหตุการณ์ OnComm เมื่อตรวจพบสัญลักษณ์ EOF โดย

True หมายถึง เหตุการณ์ OnComm จะถูกกระตุ้นให้ทำงานด้วย EOF

False หมายถึง เหตุการณ์ OnComm จะไม่ถูกกระตุ้นให้ทำงานด้วย EOF (ค่าปกติ)

เมื่อ EOFEnable กำหนดให้เป็น False ส่วนควบคุมจะไม่มีการตรวจสอบสัญลักษณ์ EOF

2.5.8.17 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) และเกิดไทม์เอาท์ คอนโทรล MSComm จะกำหนดให้คุณสมบัติ CommEvent มีค่าเป็น commEventCTSTO (Clear To Send Timeout) และกระตุ้นให้เกิดเหตุการณ์ OnComm

2.5.8.18 CDHolding

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

รูปแบบการใช้งาน

Object.CDHolding

เมื่อขา DCD เป็นลอจิก “1” (CDHolding = True) และเกิดไทม์เอาท์ คอนโทรล MSComm จะกำหนดให้คุณสมบัติ CommEvent มีค่าเป็น commEventCDTO (Carrier Detect Timeout Error) และกระตุ้นให้เกิดเหตุการณ์ OnComm

2.5.8.19 DSRHolding

ผู้ใช้งานสามารถตรวจสอบการทำงานของขา SDR ได้ว่ามีสถานะลอจิกเป็น “1” หรือ “0”โดยค่าที่อ่านได้จะเป็นบูลีน True และ False ถ้าค่า DSRHolding เป็น True ขา DSR จะมีสถานะลอจิกเป็น “1” ถ้าค่า DSRHolding เป็น False ขา DSR จะมีสถานะลอจิกเป็น “0”

รูปแบบการใช้งาน

Object.DSRHolding

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

2.5.8.20 Handshaking

กำหนดคุณสมบัติและคืนค่ารูปแบบแฮนด์เช็กทางฮาร์ดแวร์

รูปแบบการใช้งาน

Object.Handshaking [= value]

ค่าตัวแปร value ที่ใช้กำหนดค่ากำหนดได้ 4 รูปแบบดังนี้

1. comNone ค่าที่กำหนดคือ 0 เป็นการกำหนดให้ไม่มีการu3649 แฮนด์เช็ก (เป็นค่าเริ่มต้น)

2. comXOnXOff ค่าที่กำหนดคือ 1 เป็นการกำหนดให้แฮนด์เช็กแบบ XOn/XOff

3. comRTS ค่าที่กำหนดคือ 2 เป็นการกำหนดให้ใช้ขา RTS/CTS (Request ToSend/Clear To Send)

4. comRTSXOnXOff ค่าที่กำหนดคือ 3 เป็นการกำหนดให้ใช้แบบ Request To Sendและ XOn/XOff

คุณสมบัติ Handshaking ใช้เพื่อกำหนดรูปแบบการสื่อสารภายในระหว่างที่ข้อมูลถูกส่งไปยังบัฟเฟอร์ภาครับ เมื่อข้อมูลตัวอักษรถูกส่งมาถึงพอร์ตอนุกรม อุปกรณ์สื่อสารข้อมูลจะทำการย้ายข้อมูลไปยังบัฟเฟอร์ภาครับ เพื่อที่จะให้โปรแกรมสามารถอ่านค่าไปใช้งานได้ ถ้าไม่มีบัฟเฟอร์ภาครับโปรแกรมที่ใช้งานจะต้องทำการอ่านค่าข้อมูลสูญหายได้เนื่องจากว่าการเปลี่ยนแปลงของข้อมูลที่ส่งเข้ามามีการเปลี่ยนแปลงอย่างรวดเร็ว

คุณสมบัติ Handshaking ช่วยให้ผู้ใช้งานแน่ใจได้ว่าข้อมูลที่ได้รับมานั้นไม่มีการสูญหาย เมื่อบัฟเฟอร์ภาครับที่รับข้อมูลนั้นเกิดข้อมูลล้นหรือโอเวอร์โฟลว (Overflow) โดยใช้วิธีการตรวจสอบความพร้อมของบัฟเฟอร์ว่าพร้อมรับข้อมูลหรือไม่ก่อนที่จะส่งข้อมูลมาให้

2.5.8.21 Break

ใช้ในการเซตและเคลียร์ค่าสัญญาณ Break ชนิดของข้อมูลเป็นแบบ Boolean

รูปแบบการใช้งาน

Object.Break [= value]

โดย value เป็นค่าบูลีน ถ้า value = True หมายถึงการส่งสัญญาณ Break ออกไป ถ้า value = False หมายถึงการเคลียร์สัญญาณ Break

เมื่อกำหนดให้สัญญาณ Break เป็น True จะเป็นการหยุดการส่งข้อมูลชั่วคราวจนกว่าจะมีการสั่งให้สัญญาณ Break เป็น False

ตัวอย่างการส่งสัญญาณ Break ออกไปเป็นช่วงเวลา 1/10 วินาที

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

2.5.8.22เหตุการณ์ OnComm

เหตุการณ์ OnComm จะถูกสร้างขึ้นเมื่อค่าของคุณสมบัติ CommEvent มีการเปลี่ยนแปลงเพื่อแสดงผลการเปลี่ยนแปลงเหล่านั้นแบบทันทีทันใดหรือแสดงข้อผิดพลาดที่เกิดขึ้น

ตัวอย่างโปรแกรมย่อย OnComm เพื่อนำเหตุการณ์ CommEvent มาแสดง

```
Private Sub MSComm_OnComm
```

```
Select Case MSComm1.CommEvent
        'Handle each event or error by placing
        'Code below each case statement
        'Error
        Case comEventBreak 'A Break was received
        Case comEventCDTO 'CD (RLSD) timeout
        Case comEventCTSTO 'CTS timout
        Case comEventDSRTO 'DSR timeout
        Case comEventFrame 'Framing error
        Case comEventOverrun 'Data lost
        Case comEventRxOver 'Receive buffer overflow
        Case comEventRxParity 'Parity Error
        Case comEventTxFull 'Transmit buffer full
        'Events
        Case comEvCD 'Change the in CD line
        Case comEvCTS 'Change the in CTS line
        Case comEvDSR 'Change the in DSR line
        Case comEventRing 'Change the in Ring Indicator
        Case comEventReceive 'Received Rtheshold # of chars
        Case comEventSend ' Rtheshold number in the transmit buffer
        Case comEvEOF 'An EOF character was found in the input stream
    End Select
End Sub
```

2.5.9 ค่าคงที่คุณสมบัติของคอนโทรล MSComm

2.5.9.1 ค่าคงที่สำหรับคุณสมบัติ Handshaking

ค่าคงที่	ค่า	รายละเอียด
comNone	0	ไม่ใช้การตรวจสอบแฮนด์เช็ก
comXOnXOff	1	ใช้การตรวจสอบแฮน3609นด์เช็กแบบ XOn/XOff
comRTS	2	ใช้การตรวจสอบผ่านทางขา RTS และ CTS
comRTSXOnXOff		3 กำหนดการตรวจสอบแฮนด์เช็กทั้งแบบ RTS,

CTS และ XOn/XOff

2.5.9.2 ค่าคงที่สำหรับคุณสมบัติ OnComm

ค่าคงที่ ค่า รายละเอียด

comEvSend 1 ส่งค่าเหตุการณ์

comEvReceive 2 รับค่าเหตุการณ์

comEvCTS 3 มีการเปลี่ยนแปลงที่ขา CTS

comEvDSR 4 มีการเปลี่ยนแปลงที่ขา DSR

comEvCD 5 มีการเปลี่ยนแปลงที่ขา CD

comEvRing 6 ตรวจจับสัญญาณกระดิ่งของโทรศัพท์

comEvEOF 7 ตรวจพบตำแหน่งท้ายสุดของไฟล์

2.5.9.3 ค่าคงที่สำหรับคุณสมบัติ Error

ค่าคงที่ ค่า รายละเอียด

comEventBreak 1001 ได้รับสัญญาณ Break

comEventCTSTO 1002 ขา CTS เกิดไทม์เอาท์

comEventDSRTO 1003 ขา DSR เกิดไทม์เอาท์

comEventFrame 1004 เกิดข้อผิดพลาดที่เฟรมข้อมูล (Framing error)

comEventOverrun 1006 พอร์ตอนุกรมเกิดโอเวอร์รัน (Fort overrun)

comEventCDTO 1007 ขา DCD เกิดไทม์เอาท์

comEventRxOver 1008 บัฟเฟอร์รับข้อมูลเกิดโอเวอร์โฟลว

comEventRxParity 1009 เกิดข้อผิดพลาดขึ้นที่พาริตี้ (Parity error)

comEventTxFull 1010 บัฟเฟอร์ส่งข้อมูลเต็ม

2.5.9.4 ค่าคงที่สำหรับคุณสมบัติ InputMode

ค่าคงที่ ค่า รายละเอียด

comInputModeText 0 ข้อมูลที่รับมีคุณสมบัติเป็นข้อความ (ค่าปกติ)

ComInputModeBoolean 1 ข้อมูลที่รับเข้ามาเป็นข้อมูลไบนารี่

2.5.9.4 การใช้ MSComm เพื่อการติดต่อฮาร์ดแวร์

จากที่ได้กล่าวมาตอนต้น จะเห็นได้ว่าวิธีการอ่านค่าหรือเขียนค่าไปยังสถานะและขาควบคุมของพอร์ตอนุกรมสารมารถทำได้ง่าย โดยใช้คำสั่งเหล่านี้

DTREnable

RTSEnable

CTSHolding

CDHolding

DSRHolding

Break

บทที่ 3
การออกแบบ

ในบทนี้จะกล่าวถึง การสร้างเครื่องวัดอุณหภูมิไร้สายโดยใช้หลักการทำงานของการ รับ-ส่งสัญญาณด้วยคลื่นวิทยุย่านความถี่สูงยิ่ง UHF มาประยุกต์ใช้ในโครงงาน โดยแยกการออกแบบและการสร้างออกเป็นส่วนหลักๆได้ดังนี้

3.1 บล็อกไดอะแกรมเครื่องวัดอุณหภูมิไร้สาย

3.2 ภาคเครื่องส่ง-เครื่องรับ modual ไร้สาย

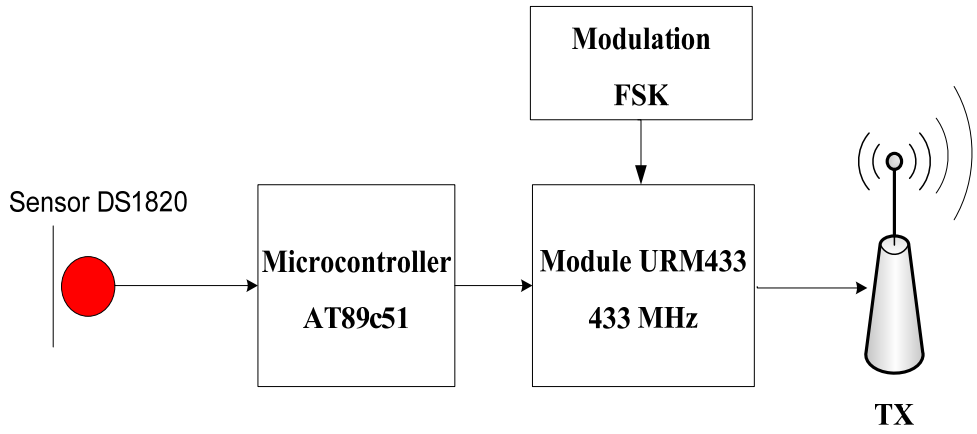
3.3 แผนผังการทำงานของ ไมโครคอลโทรเลอร์

3.4 การทำงานของโปรแกรม Visual Basic

3.1 บล็อกไดอะแกรมเครื่องวัดอุณหภูมิไร้สาย

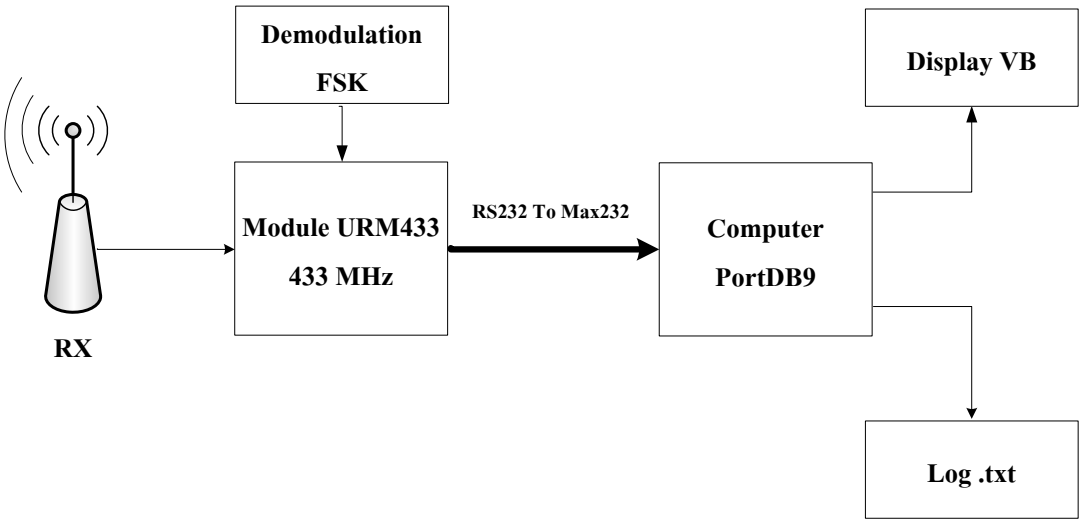
3.1.1 การออกแบบและการสร้างส่วนของฮาร์ดแวร์

จากแนวคิดของการสื่อสารด้วยคลื่นวิทยุ จึงนำเอาหลักการทำงานของการ รับ-ส่งสัญญาณด้วยคลื่นวิทยุมาประยุกต์ใช้ในโครงงานนี้ ทำให้เกิดความสะดวกในการติดตั้งหรือเคลื่อนย้าย โดยใช้คลื่นวิทยุย่านความถี่สูง 433 MHz ในการส่งข้อมูลแบบไร้สายส่วนประกอบภาคเครื่องส่งแสดงดังรู้ที่ 3.1 เริ่มด้วยวงจรตัวตรวจวัด จะทำการวัดอุณหภูมิ และส่งมาให้กับหน่วยประมวลผลเพื่อเปลี่ยนข้อมูลดิจิตอลเป็นข้อมูลอุณหภูมิ โดยกระบวนการทางซอต์ฟแวร์ สัญญาณข้อมูลที่ได้จะส่งป้อนไปยังภาคเครื่องส่งไปผสมกับคลื่นพาหะในวงจรมอดูเลเตอร์ แล้วส่งสัญญาณดังกล่าวไปขยายให้มีกำลังแรงขึ้นในวงจรขยายคลื่นวิทยุจากนั้นจึงส่งคลื่นวิทยุออกทางสายอากาศเพื่อกระจายคลื่นวิทยุออกไป วิธีออกแบบส่วนประกอบของรูปที่ 3.1 และรูปที่3.2 ทำได้ดังนี้



รูปที่ 3.1 บล็อกไดอะแกรมภาคเครื่องส่ง

ส่วนประกอบภาคเครื่องรับแสดงดังรูปที่ 3.2 วงจรเลือกรับคลื่นวิทยุโดยจะรับคลื่นวิทยุที่มีย่านความถี่เดียวกับเครื่องส่งจากนั้นทำการขยายคลื่นวิทยุเพื่อให้สัญญาณแรงขึ้นแล้วส่งเข้าวงจรดีมอดูเลเตอร์ เพื่อแยกเอาสัญญาณข้อมูลแบบดิจิตอล ออกมา ข้อมูลที่ได้จะนำมาแปลงให้เป็นข้อมูลที่ต้องการด้วยเงื่อนไขของโปรแกรม ข้อมูลของอุณหภูมิจะแปลงให้อยู่ในรูปของเลขฐานสิบ พร้อมแสดงผลบนจอมอนิเตอร์

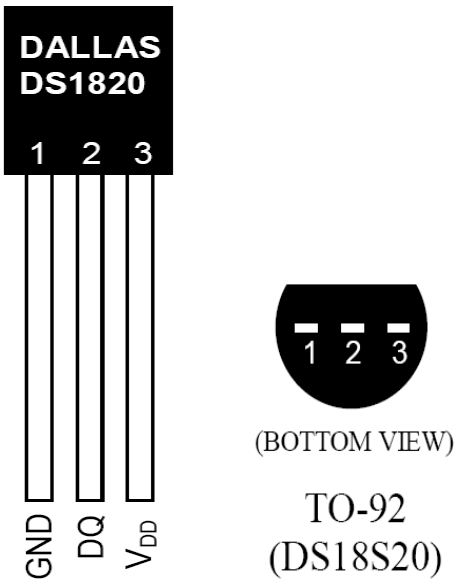


รูปที่ 3.2 บล็อกไดอะแกรมภาคเครื่องรับ

3.1.2 การเลือกใช้ตัวตรวจวัดอุณหภูมิ

เนื่องจากเครื่องวัดอุณหภูมิไร้สายนี้ ต้องการตัวตรวจวัดที่จะนำไปใช้วัดอุณหภูมิสัญญาณเอาต์พุตที่ได้เป็นสัญญาณดิจิตอลเพื่อใช้ติดต่อกับไมโครคอนโทรเลอร์ได้ทันที มีค่าความผิดพลาด

ต่ำ และมีขนาดเล็กเพื่อต้องการจ่ายกระแสต่ำ ซึ่งคุณสมบัติดังกล่าวตรงกับตัวตรวจวัดอุณหภูมิเบอร์ DS1820 ดังรูปที่ 3.3

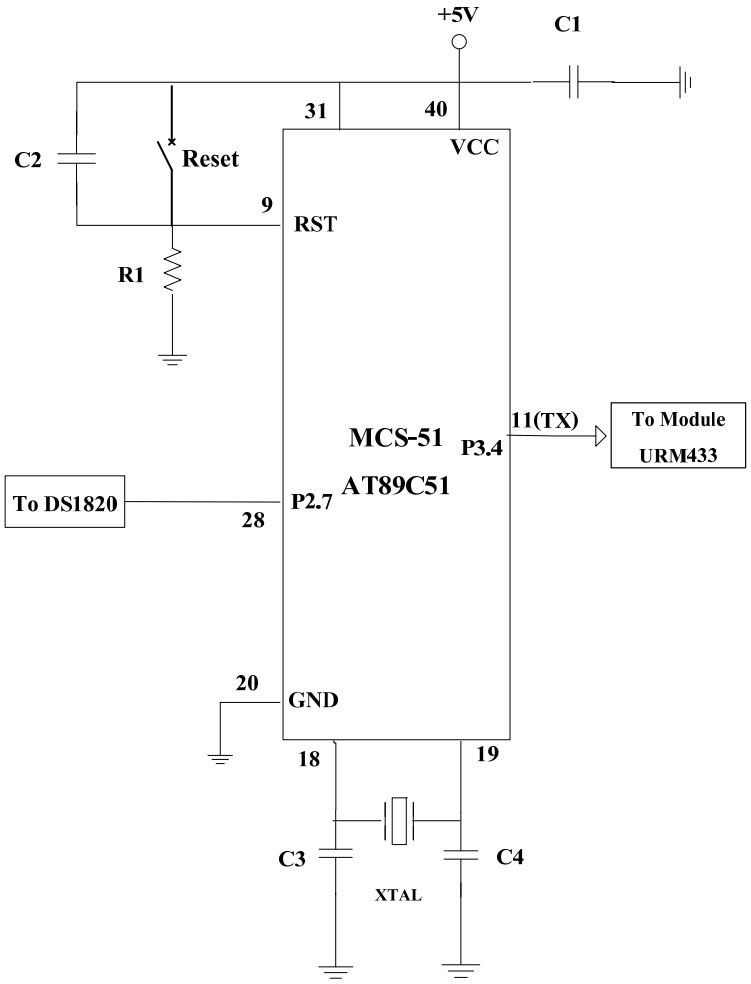


รูปที่ 3.3 ตัวตรวจวัดอุณหภูมิเบอร์ DS1820

3.1.3 การออกแบบหน่วยประมวลผลตัวตรวจวัด

เนื่องจากเอาต์พุตตัวตรวจวัดอุณหภูมิจะต้องต่อใช้งานร่วมกับ ไมโครคอนโทรลเลอร์ ดังนั้นที่จะเลือกใช้ต้องสามารถตรวจสอบจำแนกข้อมูลอนุกรมได้ (รหัสอุณหภูมิ Code) ซึ่งเป็นมาตรฐานสื่อสาร 1–Wire Bus เช่น เบอร์ AT89C51 , AT89C2051 ในที่นี้จึงเลือกไมโครคอนโทรลเลอร์ เบอร์ AT89C51

การต่อใช้งานไมโครคอนโทรลเลอร์ เบอร์ AT89C51 ดังรูปที่ 3.4 ประกอบด้วยวงจรเพาเวอร์ออนรีเซต วงจรกำเนิดความถี่ ตัวต้านทานพูลอัพ (R pull up) และการเชื่อมต่อระหว่างไมโครคอนโทรลเลอร์กับตัวตรวจวัด



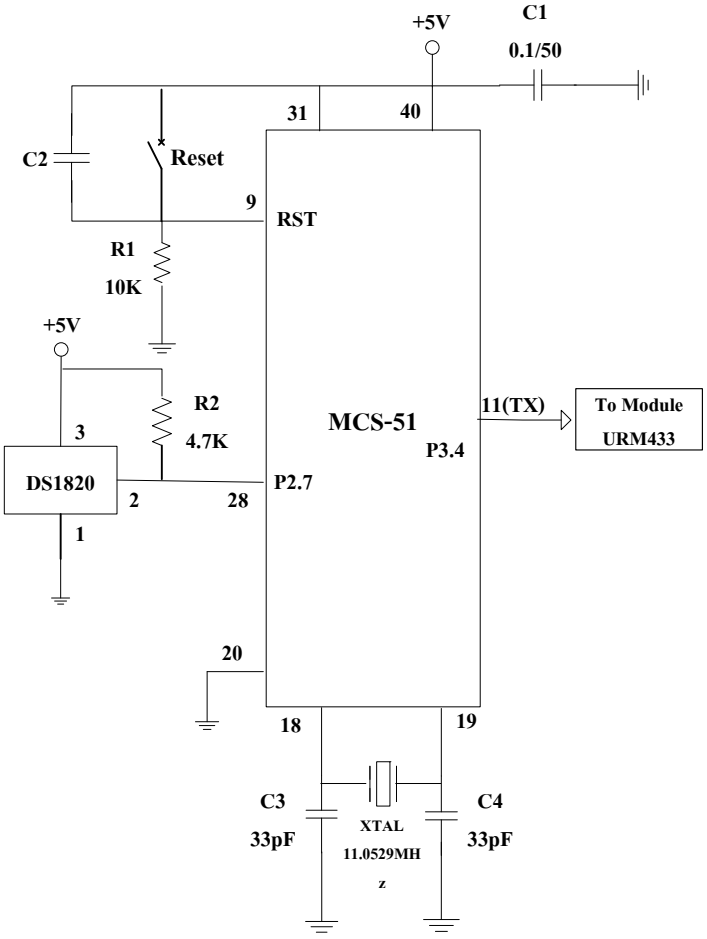
รูปที่ 3.4 การต่อใช้งานไมโครคอนโทรลเลอร์

1) การออกแบบวงจรกำเนิดความถี่ ในคู่มืออุปกรณ์ (Datasheet) ซึ่งเก็บไว้ในภาคผนวกหน้าที่ กำหนดไว้ว่า ถ้าใช้คริสตอล (Xtal) ขนาดความถี่ 11.0529 MHz ควรเลือกใช้ตัวเก็บประจุที่มีค่าอยู่ระหว่าง 30 pF ผิดพลาดได้ 10 pF จึงเลือกใช้ค่า C_3 และ C_4 = 33 pF

2) การออกแบบตัวต้านทานพลูอัพ (Rp) การเชื่อมต่ออุปกรณ์แบบ 1 – Wire Bus จะต้องต่อกับตัวต้านทานแบบ pull up 2-10kΩ ตามหัวข้อที่ 2.5 การเชื่อมต่ออุปกรณ์แบบ I^2C เพื่อความสะดวกในการกำหนดค่าทางการค้า จึงเลือกใช้ R_2 = 4.7 kΩ

3) การออกแบบวงจรเพาเวอร์ออนรีเซตเพื่อทำการกำหนดค่าเริ่มต้นต่างๆให้กับระบบ ใช้ R_1 = 10 k และ C_3 = 0.1uF

4) การออกแบบเชื่อมต่อระหว่างไมโครคอนโทรลเลอร์กับตัวตรวจวัดอุณหภูมิดังรูปที่ 3.5 เป็นการนำเอาวงจรตัวตรวจวัดอุณหภูมิ วงจรกำเนิดความถี่ และตัวต้านทานพลูอัพ ที่ได้ออกแบบในขั้นตอนที่แล้วมาต่อใช้งานร่วมกันกับไมโครคอนโทรลเลอร์เบอร์ AT89C51



รูปที่ 3.5 การเชื่อมต่อระหว่างไมโครคอนโทรลเลอร์กับตัวตรวจวัด

3.2 ภาคเครื่องส่ง-เครื่องรับ modual ไร้สาย

ต้องการส่งข้อมูลแบบไร้สายผ่านตัวกลางคลื่นวิทยุความถี่ 433 MHz และมอดูเลชั่นแบบ FSK (Frequency Shift Keying) ความสามารถในการส่งข้อมูล ระยะ 50 เมตร ในตัวเครื่องวัด ได้สร้างเครื่องวัดอุณหภูมิแบบไร้สายพร้อมแสดงผลและบันทึกข้อมูลบนคอมพิวเตอร์โดยใช้โมดูลเครื่องส่งไร้สาย URM 433 ดังรูปที่ 3.6 ทำให้คณะผู้จัดทำเลือกใช้โมดูลดังกล่าว ซึ่งมีคุณสมบัติทางเทคนิค (Specification) ดังนี้

รูปที่ 3.6 โมดูลเครื่องส่ง-เครื่องรับ

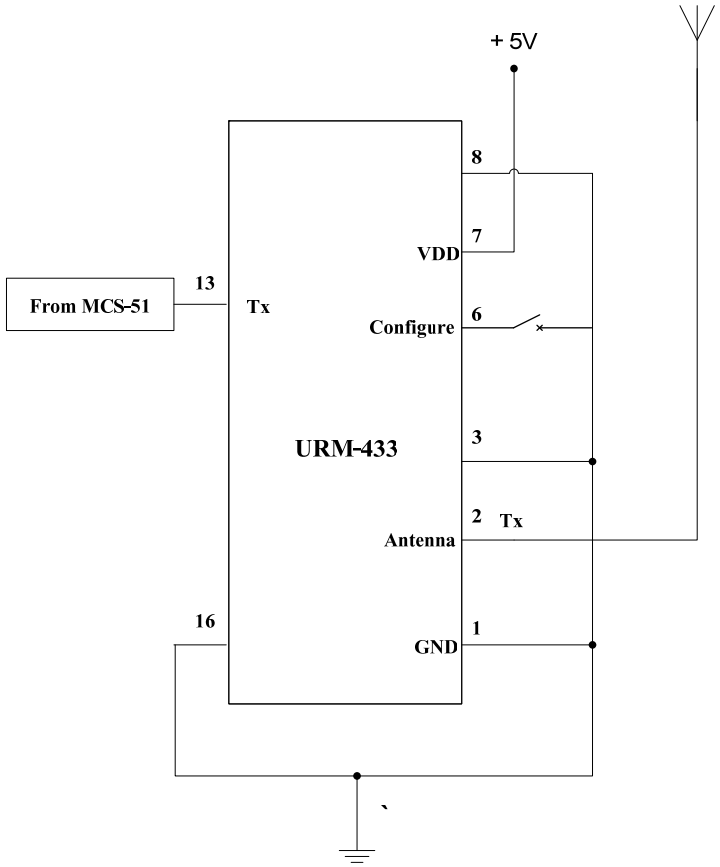
1) ส่งข้อมูลแบบไร้สายผ่านตัวกลางคลื่นวิทยุย่านความถี่ UHF ที่ความถี่ 433 MHz
2) มอดูเลชั่นแบบดิจิตอล (FSK)
3) ความสามารถในการส่งข้อมูลระยะ 50 เมตร ภายในอาคาร

3.2.1การเลือกใช้โมดูลเครื่องรับ-ส่ง

ทำหน้าที่รับสัญญาณที่มาจากโมดูลเครื่องส่งเลือกรับคลื่นวิทยุ ขยายคลื่นวิทยุ แยกคลื่นพาหะจากสัญญาณโมดูลเครื่องรับ ซึ่งทั้ง ภาครับและภาคส่งจะใช้โมดูล ไร้สายรุ่นเดียวกัน (URM-433) โดยตัวรับและตัวส่งจะทำการ configure คนละแบบ คือภาค ส่งจะตั้งค่าเป็น slave ซึ่งจะมีจำนวน 2 จุด คือ sensor 1 และ sensor 2 ทั้ง 2 ตัว จะสลับการส่งคือถ้ามีตัวใดส่งอยู่อีกตัวจะรอจนเสร็จอีกตัวถึงจะส่งค่าอุณหภูมิได้ ส่วนทางภาครับจะมี โมดูล 1 ตัว ซึ่งจะนำค่าที่ได้ส่งไปยังโปรแกรม VB เพื่อแสดงค่าอุณหภูมิ

3.2.2 การออกแบบเชื่อมต่อระหว่างโมดูลภาคเครื่องส่งกับหน่วยประมวลผลตัวตรวจวัด

ซึ่งในหัวข้อที่ 3.1.3 ได้กำหนดขาไมโครคอนโทรลเลอร์ที่ใช้ในการส่งข้อมูลไว้ที่ขา 28 (P2.7) ดังนั้นในการเชื่อมต่อระหว่างโมดูลภาคเครื่องส่งกับหน่วยประมวลผลตัวตรวจวัด โมดูลภาคเครื่องส่งไร้สาย จึงสามารถต่อใช้งานร่วมกันได้เลย โดยขา2 ซึ่งเป็นขารับข้อมูลเข้าของโมดูลภาคเครื่องส่งไร้สาย จะต่อร่วมกับไมโครคอนโทรลเลอร์ที่ ขา 12 ดังรูปที่ 3.7



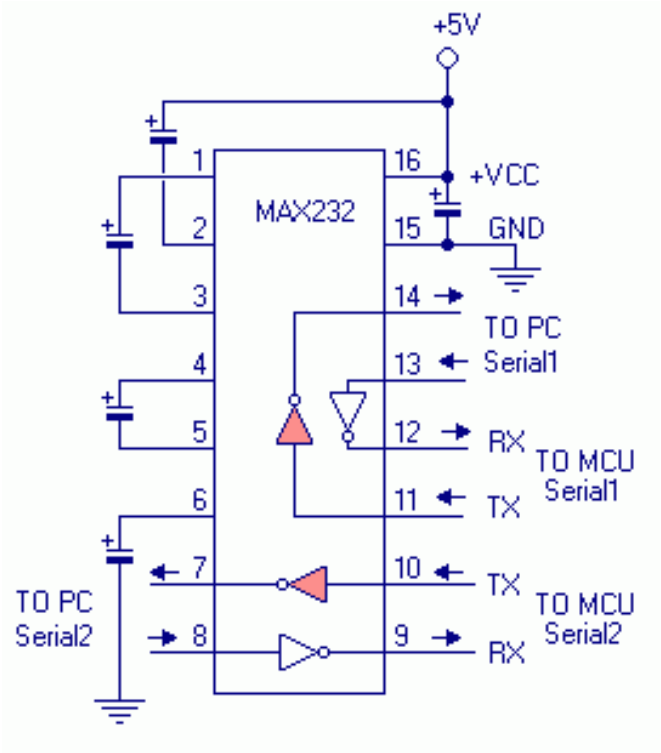
รูปที่ 3.7 การต่อโมดูลเครื่องส่งไร้สายรับค่าจาก ไมโครคอนโทรลเลอร์

3.2.3 การออกแบบหน่วยประมวลผลข้อมูลเครื่องรับ

การทำงานของภาครับเมื่อนำสัญญาณข้อมูลที่ได้มาจากภาครับของวงจร โดยจะนำค่ามาผ่าน วงจร Max 232 ซึ่งจะทำหน้าที่แปลงสัญญาณค่า Digital เป็น สัญญาณ Analog โดยผ่านสาย RS232 หรือ DB9 จะทำการแปลงเป็นระดับ TTL และในทำนองเดียวกันก็แปลงระดับสัญญาณ TTL ไปเป็นระดับสัญญาณ RS-232

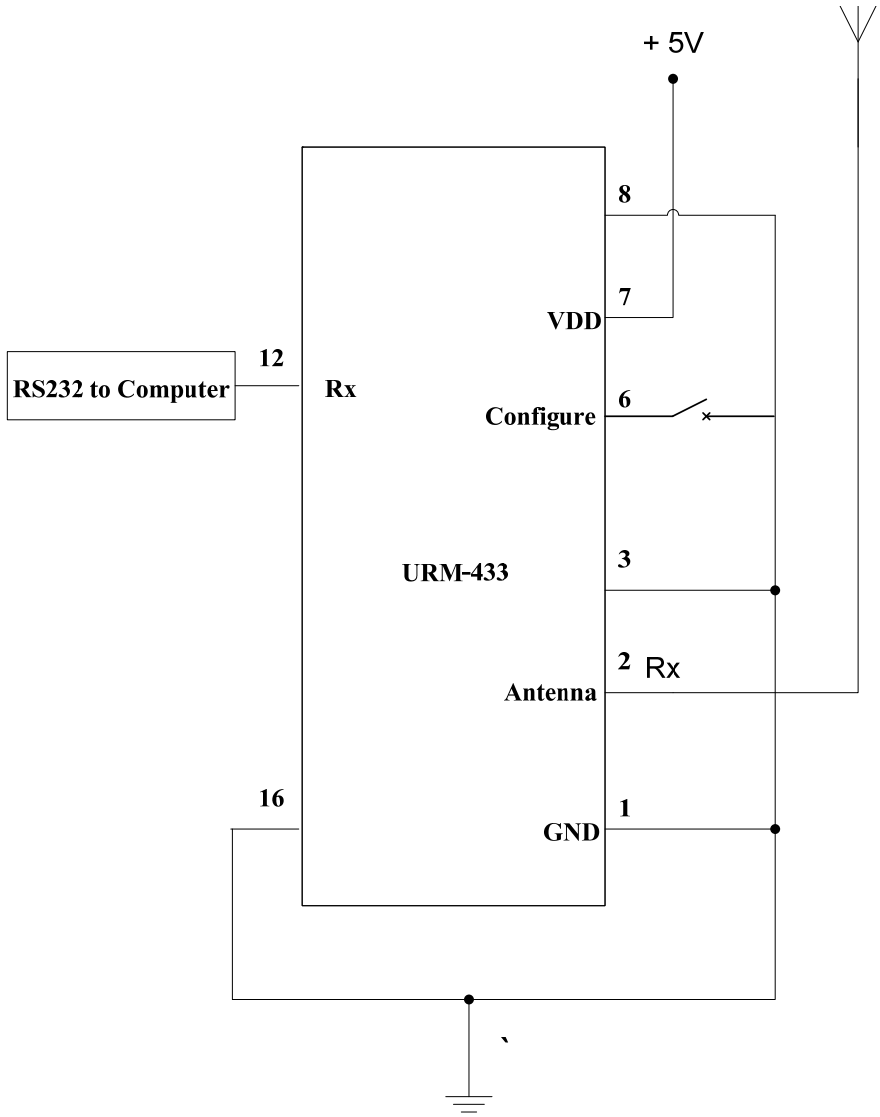
3.2.4 การทำงานของวงจรแปลงระดับแรงดัน

เริ่มต้นจากคอมพิวเตอร์เชื่อมต่อผ่านคอนเนคเตอร์ DB-9 สัญญาณจากคอมพิวเตอร์นี้จะมีระดับแรงดันตามมาตรฐาน RS232 กล่าวคือมีระดับแรงดันตั้งแต่+3 ถึง +12 V และ -3 ถึง -12 V เพื่อให้สามารถทำงานเข้ากันได้กับวงจรหลัก ซึ่งมีแรงดันเป็นแบบ TTL จึงต้องต่อผ่านไอซี MAX232CPE เพื่อแปลงแรงดันให้อยู่ในระดับทีทีแอลเสียก่อน เนื่องจากขาสัญญาณที่ใช้กับพอร์ตอนุกรมมีมากถึง 7 สัญญาณจึงต้องใช้ไอซี MAX232CPE จำนวน 2 ตัวสำหรับแปลงแรงดันด้านอินพุตและเอาท์พุต แต่ในโครงงานชิ้นนี้ใช้ขาสัญญาณเพียง 3 สัญญาณ สามารถลดไอซีดังกล่าวเหลือเพียงตัวเดียวได้



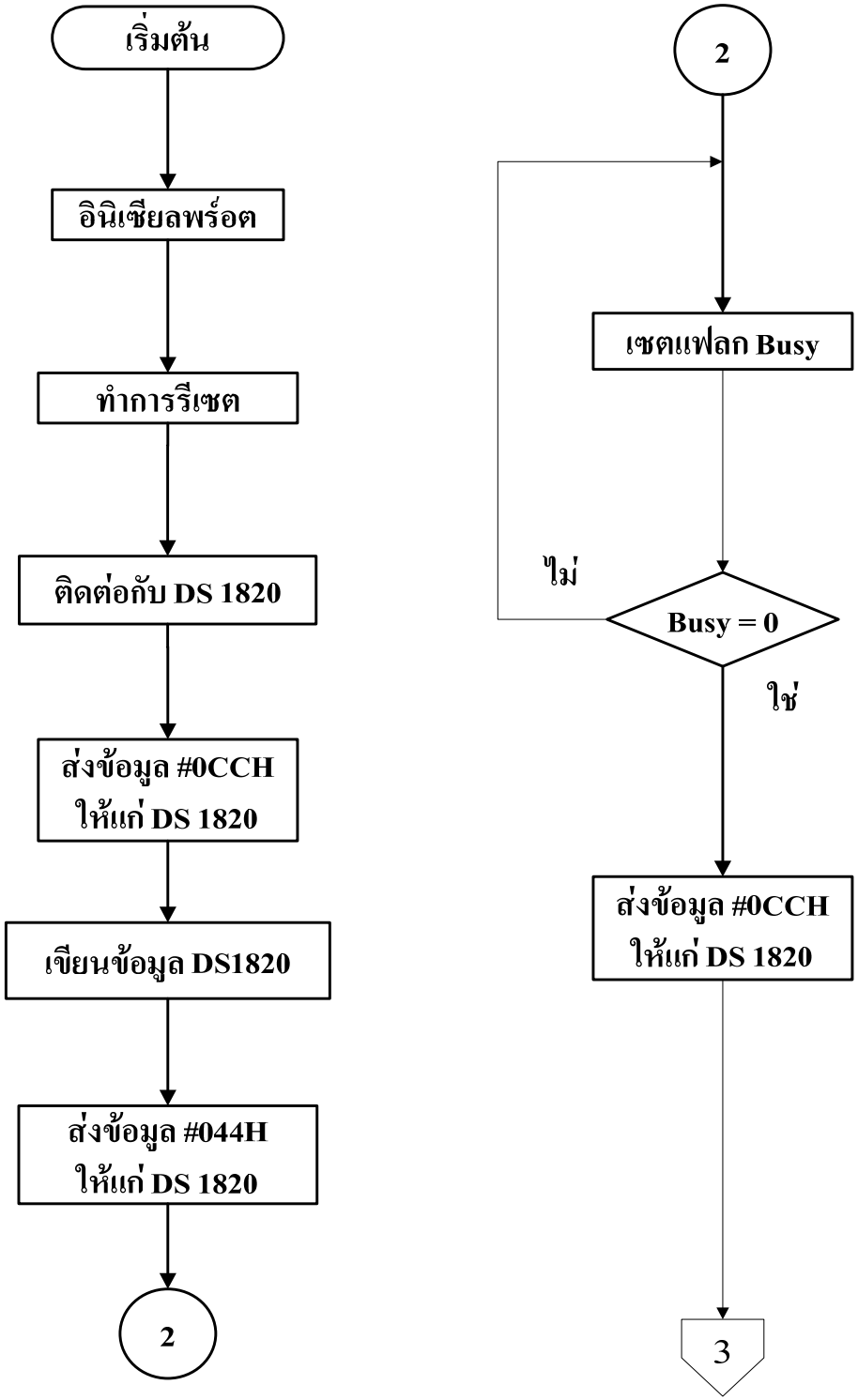
รูปที่ 3.8 วงจร Max 232

สำหรับการใช้งานการสื่อสารอนุกรมแบบ 1–Wire Bus โดยใช้สัญญาณขา DTR, RTS และ DCD เพื่อจัดสัญญาณให้ตรงตามมาตรฐานแบบ 1–Wire Bus ขา SDA ของระบบทำหน้าที่เป็นขาข้อมูล ซึ่งต้องมีการส่งและรับสัญญาณ โดยจะใช้ขา RTS ในการส่งสัญญาณ และรับสัญญาณผ่านทางขา DCD

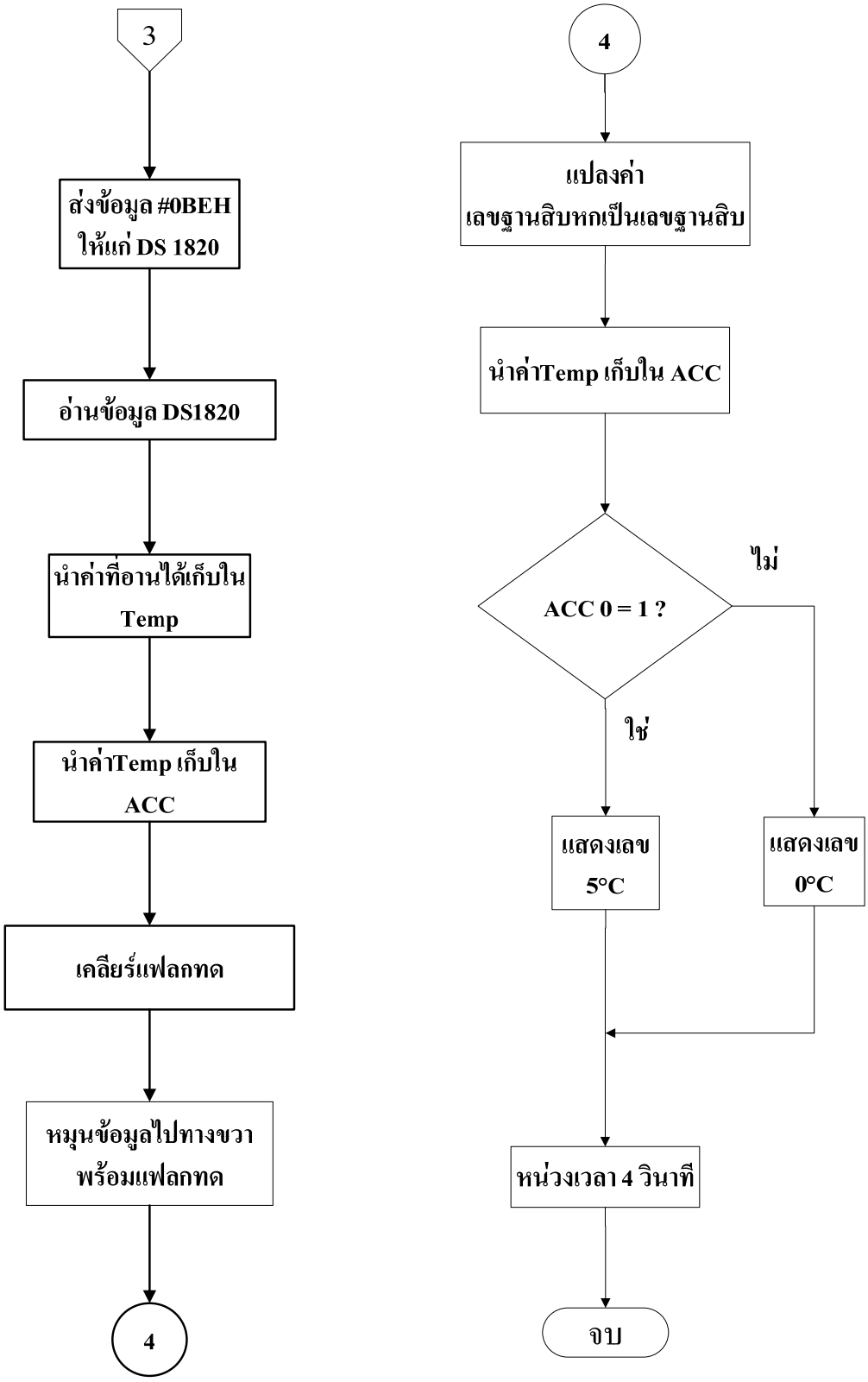


รูปที่ 3.9 โมดูลภาคเครื่องรับไร้สาย ร่วมกับ Computer

3.3 แผนผังการทำงานของ ไมโครคอลโทรเลอร์



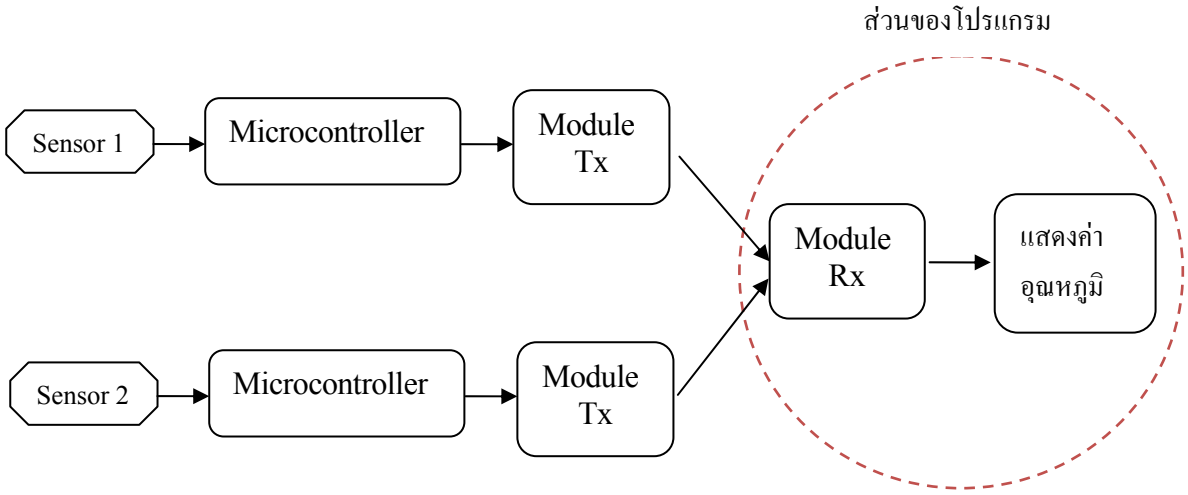
รูปที่ 2.10 ผังการทำงานของไมโครคอลโทรเลอร์



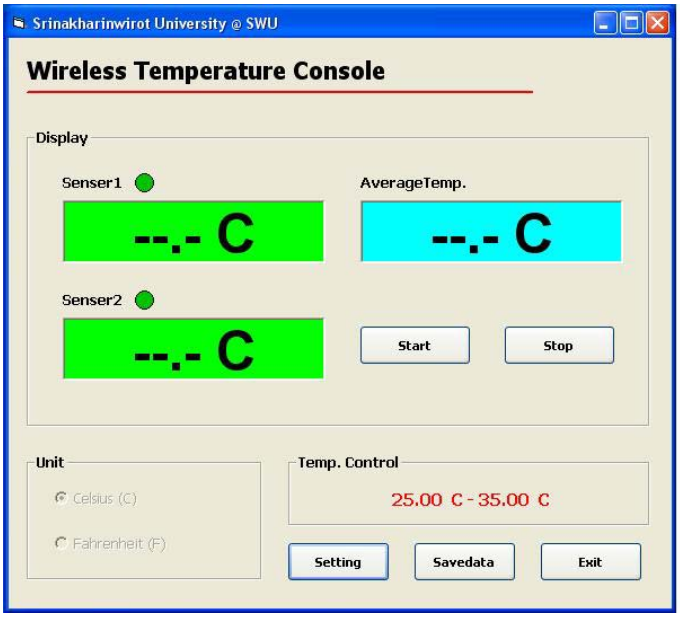
รูปที่ 2.11 ผังการทำงานของไมโครคอลโทรเลอร์ (ต่อ)

3.4 การทำงานของโปรแกรม Visual Basic

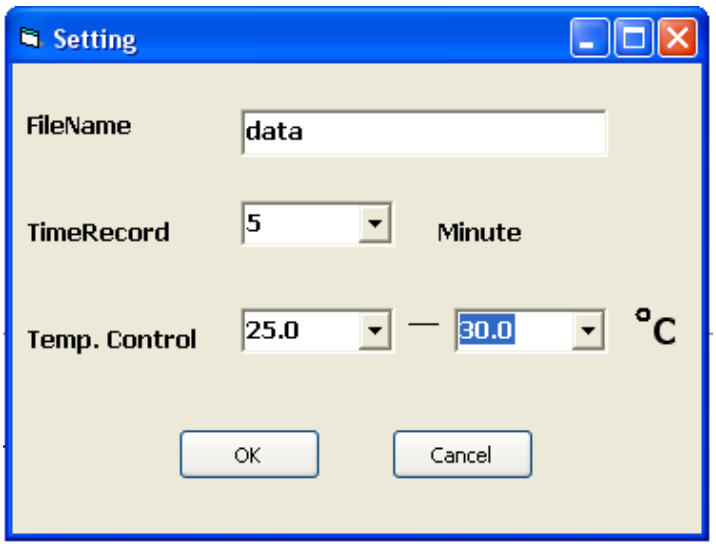
หลักในการทำงานของโปรแกรม คือจะทำการรับค่า อุณหภูมิเป็น องศาเซลเซียสจากการส่งค่าของ ตัว Microcontroller จะมี code แปะใส่มาด้วย ฉะนั้นโปรแกรมจึงมีหน้าที่ ตัด code ที่มากับอุณหภูมิในรูปที่ 3.12 จะเป็นการแสดงส่วนการรับค่าอุณหภูมิ และ หน้าตาของโปรแกรม Visual Basic ดังรูปที่ 3.14



รูปที่ 3.12 ส่วนแสดงการรับค่าอุณหภูมิ



รูปที่ 3.13 ส่วนของภาคแสดงผลโปรแกรม Visual basic



รูปที่ 3.14 ส่วนของการตั้งค่าของโปรแกรม

ในโปรแกรมก็จะมีการตั้งค่าเวลาในการบันทึกค่าอุณหภูมิได้และย่านของอุณหภูมิที่เราจะตั้งให้มีการเตือนในรูปที่ 3.14 จะตั้งไว้ระหว่าง 25-30 องศาเซลเซียส คือถ้ามีค่าต่ำกว่าหรือสูงกว่าก็จะมีการเตือนเกิดขึ้นโดยจะแสดงเป็นข้อความปรากฏที่หน้าจอ computer

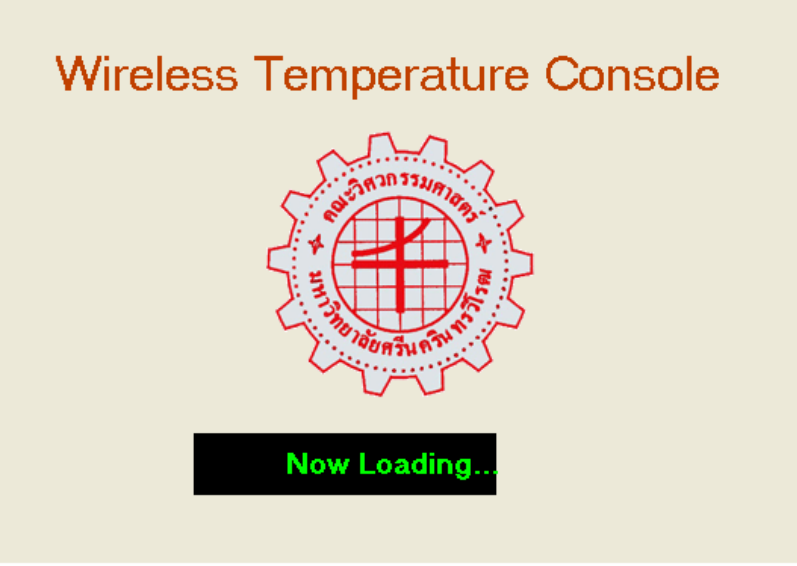
บทที่ 4
ผลการทดลอง

ในบทนี้เป็นผลการทดลองเครื่องวัดอุณหภูมิผ่านโมดูลไร้สาย จาก sensor DS1820 ค่าที่ได้จะเป็นค่า ดิจิตอล ตัวเลขอุณหภูมิ องศาเซลเซียส ซึ่งค่าจะส่งมาจากโมโดลไร้สาย 2 ตัว เราจึงนำค่าที่ได้มาแสดงโดย โปรแกรม Visual Basic 6.0 จะมีรายละเอียดดังนี้

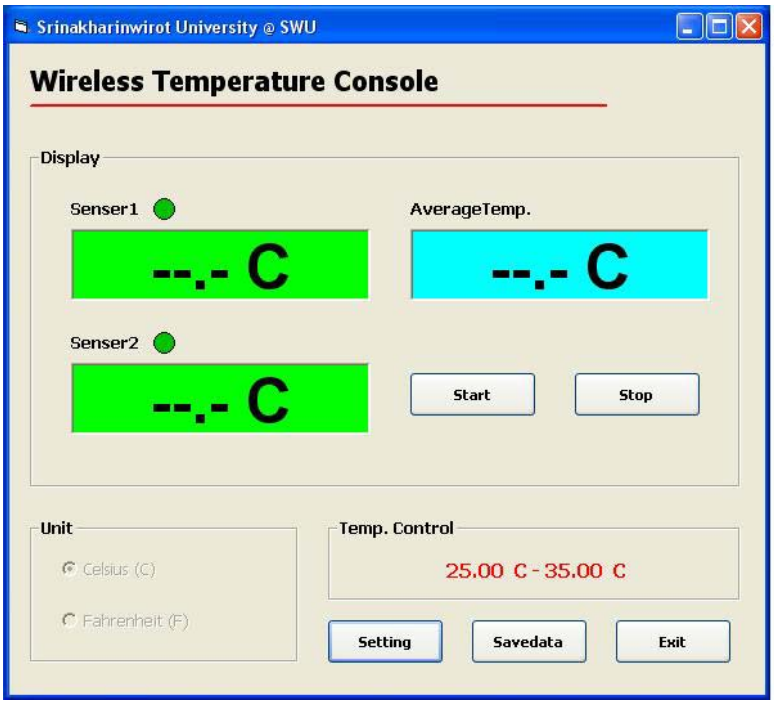
4.1 ลักษณะของโปรแกรมที่ใช้ในการทดลอง

4.2 การเปรียบเทียบค่าที่ได้จากการทดลอง

4.1 ลักษณะของโปรแกรมที่ใช้ในการทดลอง



รูปที่ 4.1 รูปการเรียกเปิดโปรแกรม

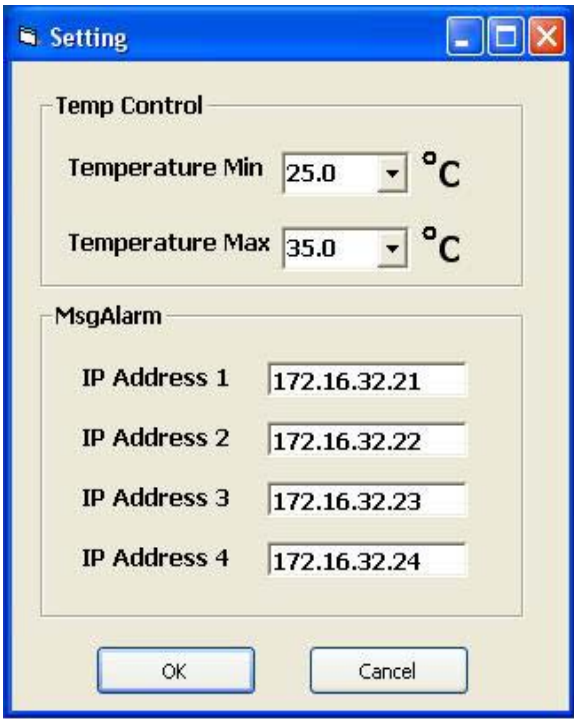


รูปที่ 4.2 โปรแกรมที่ใช้ในการทดลองขณะไม่มีการส่งค่าอุณหภูมิ

เมื่อทำการเรียกโปรแกรมขึ้นมาใช้งานจะปรากฏหน้าจอหลักของการใช้งานโปรแกรมดังแสดงในรูปที่ 4.2 ในหน้าจอหลักนั้นจะปึนตัวแสดงผลค่าของอุณหภูมิจำนวน 2 ช่อง sensor และ ค่าเฉลี่ยของอุณหภูมิทั้ง 2 จุด ที่ใช้ในการแสดงผลอุณหภูมิ

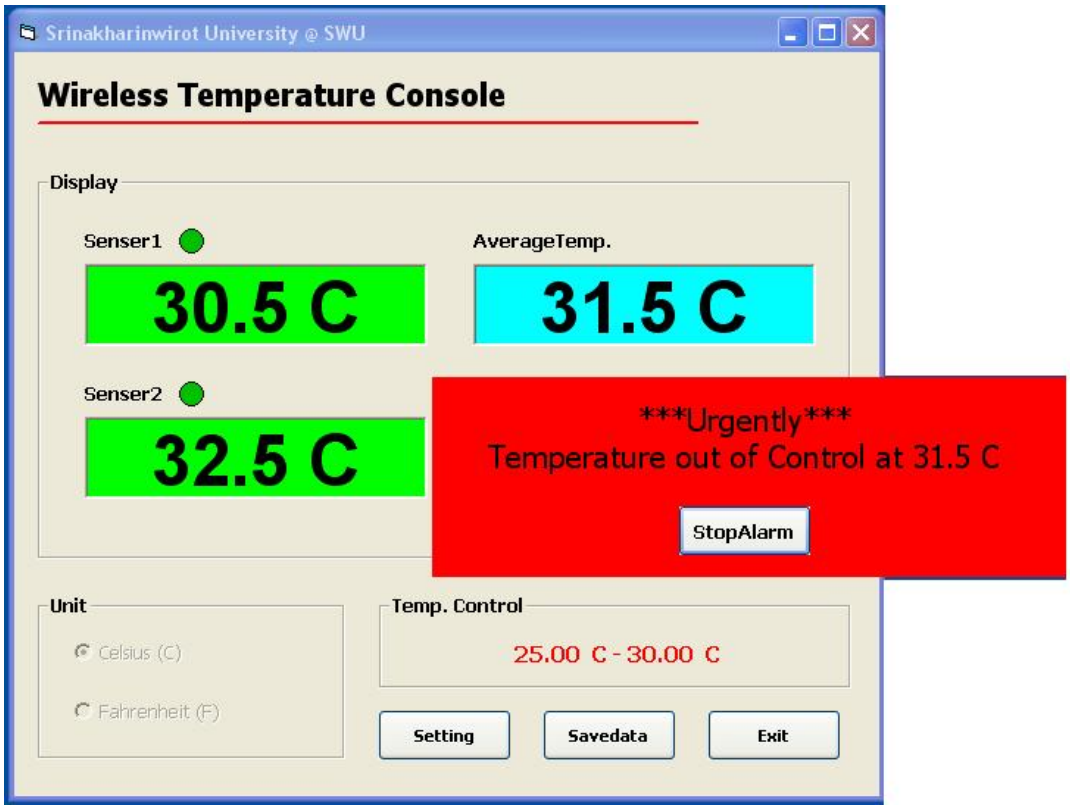
4.1.1 ทำการกำหนดค่าของการรับและส่งข้อมูล

ในการกำหนดการรับค่าการทำงานของเครื่องวัดเราจะมีการติดต่อกับเครื่อง Administrator โดยตรงคือเราสามารถระบุเครื่องที่จะมีการส่งค่าการ Alarm ของเครื่องวัดได้ โดยสามารถที่จะแจ้งข้อความเตือนผ่านโปรแกรม IP-Message โดยกำหนดการ Add IP-Address ได้มากกว่า 1 เครื่องซึ่งจะทำให้คนที่นั่ง Monitor ในการทดลองนี้จะใช้ได้ 4 เครื่อง



รูปที่ 4.3 การ Configure ค่า ของ Computer ของโปรแกรม IP-Message

เมื่อทำการกำหนดค่าของการรับส่งข้อมูลเรียบร้อยแล้วและเปิด port รับส่งข้อมูล โปรแกรมจะรับค่าอุณหภูมิเข้ามาโดยจะแสดงค่าของข้อมูลที่รับออกมาเป็นค่าตัวเลข องศาซึ่งสามารถที่จะเปลี่ยนหน่วยเป็น Celsius และ Fahrenheit ได้



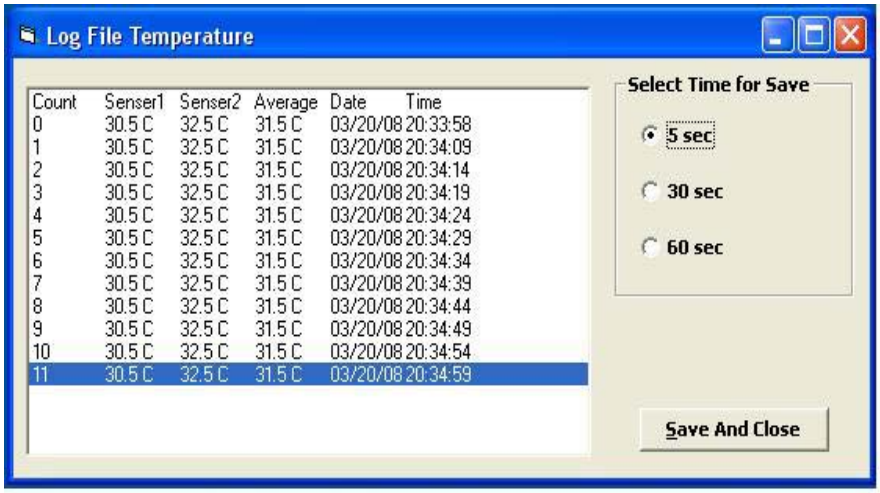
รูปที่ 4.4 รูปแสดงค่าของข้อมูลอุณหภูมิที่ได้รับเข้ามา

การทำงานของโปรแกรมขึ้นอยู่กับการตั้งค่าการ Alarm ในตัวอย่างจะตั้งไว้ที่ 25.00C°-30.00C° ถ้าอุณหภูมิสูงกว่าหรือต่ำกว่านั้นโปรแกรมก็จะมีการส่งค่า Alarm ที่จอที่ที่หน้าจอ computer และจะส่งไปยังค่าที่เราได้ตั้งค่า IP-Address ไว้ข้างต้นหลังจากนั้นโปรแกรม visual basic ก็จะส่งค่าออกเป็น message แจ้งเตือนให้ Admin ได้รู้ตัว เพราะเกิดเหตุขัดข้อง เรื่องของอุณหภูมิในห้อง server

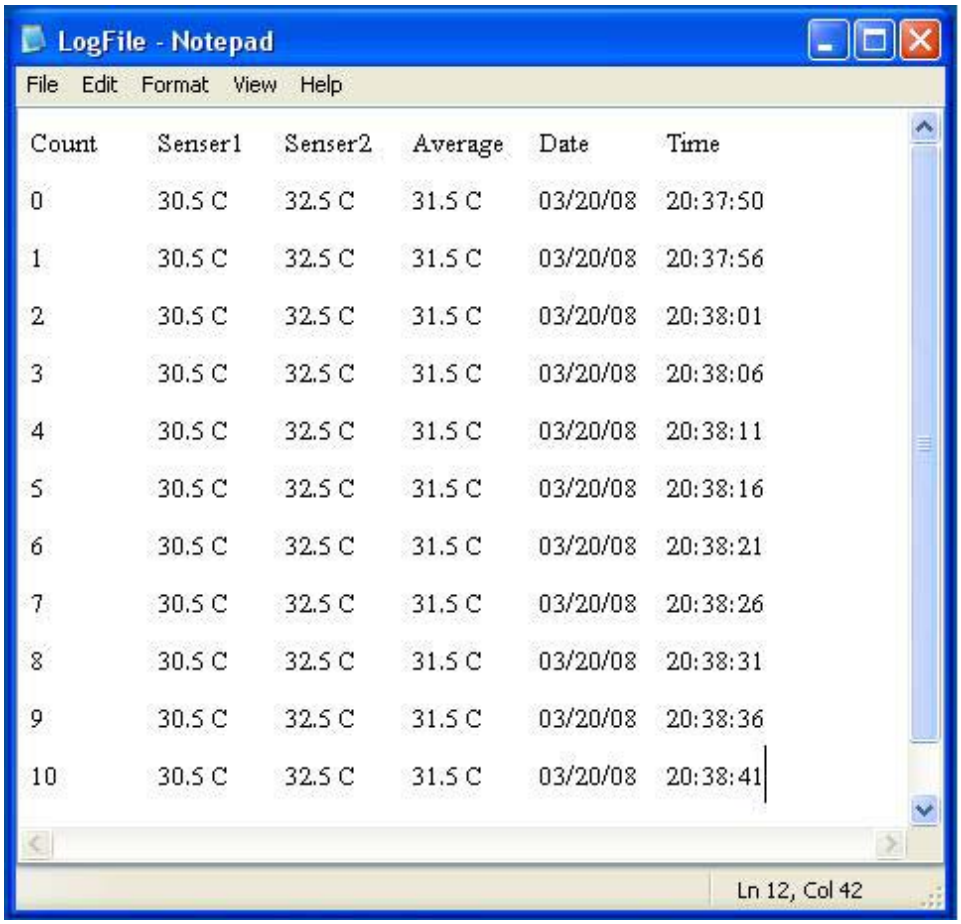
รูปที่ 4.5 แสดงการส่งข้อความเตือนของโปรแกรม IP-Message

4.1.2 โปรแกรมสามารถเก็บข้อมูลของอุณหภูมิ

โปรแกรมสามารถเก็บข้อมูลของอุณหภูมิในรูปแบบของ LogFlie ได้โดยเราสามารถที่จะตั้งค่าการเก็บผลการวัดค่าได้ทุกๆ 5 วินาที 10 วินาที และ 1 นาที โดยจะมีการบันทึก ค่าของอุณหภูมิ เวลา รวมไปถึงวันเดือนปีที่เราตรวดวัดค่าอุณหภูมิ เพื่อที่จะก็บไว้เป็น Database ได้เพื่อเป็นค่าอ้างอิงเมื่อเกิดปัญหา File ที่เก็บมีค่าไม่กี่ kb



รูปที่ 4.6 รูปแสดงข้อมูลอุณหภูมิที่ได้จากการบันทึกค่า



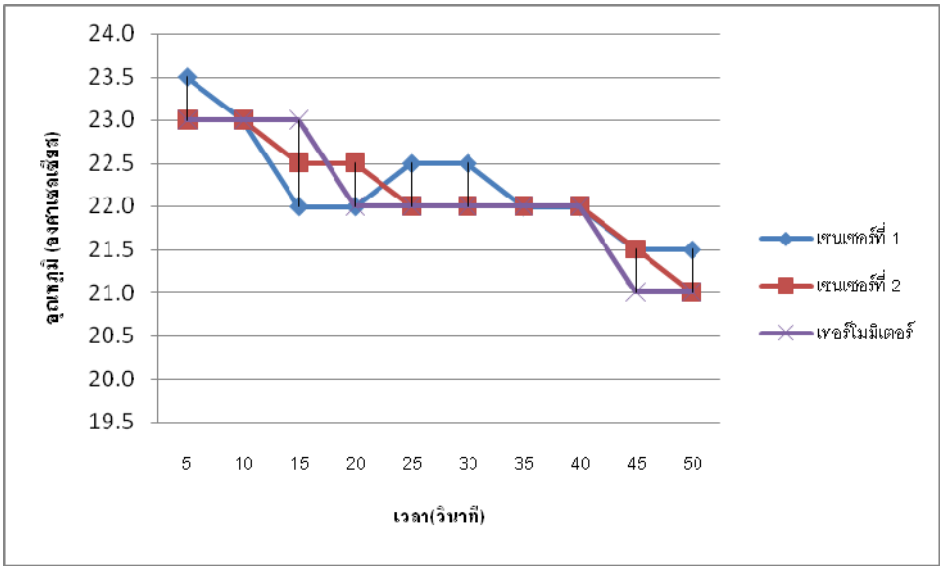
รูปที่ 4.7 รูปแสดง Log Flie ของการเก็บค่าอุณหภูมิทุกๆ 5 วินาที

4.2 การเปรียบเทียบค่าที่ได้จากการทดลอง

เมื่อทำ การทดสอบค่าความแม่นยำ ของอุณหภูมิที่อ่านได้จากโปรแกรมเทียบกับเทอร์โมมิเตอร์ที่อุณหภูมิบริเวณต่าง ๆ ได้ผลการทดลองดังตารางที่ 4.1

ตารางที่ 4.1 แสดงอุณหภูมิอ่านได้จากโปรแกรมทั้ง 2 จุดเทียบกับเทอร์โมมิเตอร์ที่อุณหภูมิต่ำ

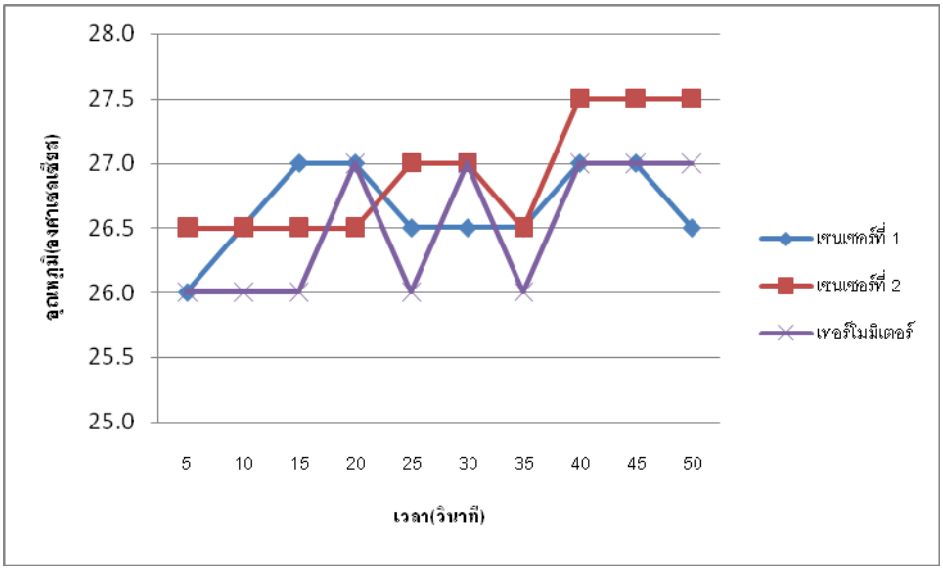
ครั้งที่	อุณหภูมิ(องศาเซลเซียส)				
	เวลา(วินาที)	เซนเซอร์ที่ 1	เซนเซอร์ที่ 2	เทอร์โมมิเตอร์	ความคลาดเคลื่อน (%)
1	5	23.5	23.0	23.0	1.09
2	10	23.0	23.0	23.0	0.00
3	15	22.0	22.5	23.0	-3.26
4	20	22.0	22.5	22.0	1.14
5	25	22.5	22.0	22.0	1.14
6	30	22.5	22.0	22.0	1.14
7	35	22.0	22.0	22.0	0.00
8	40	22.0	22.0	22.0	0.00
9	45	21.5	21.5	21.0	2.38
10	50	21.5	21.0	21.0	1.19



รูปที่ 4.8 กราฟอุณหภูมิอ่านได้จากโปรแกรมทั้ง 2 จุดเทียบกับเทอร์โมมิเตอร์ที่อุณหภูมิต่ำ

ตารางที่ 4.2 แสดงอุณหภูมิจากโปรแกรมทั้ง 2 จุดเทียบกับเทอร์โมมิเตอร์ที่อุณหภูมิห้อง

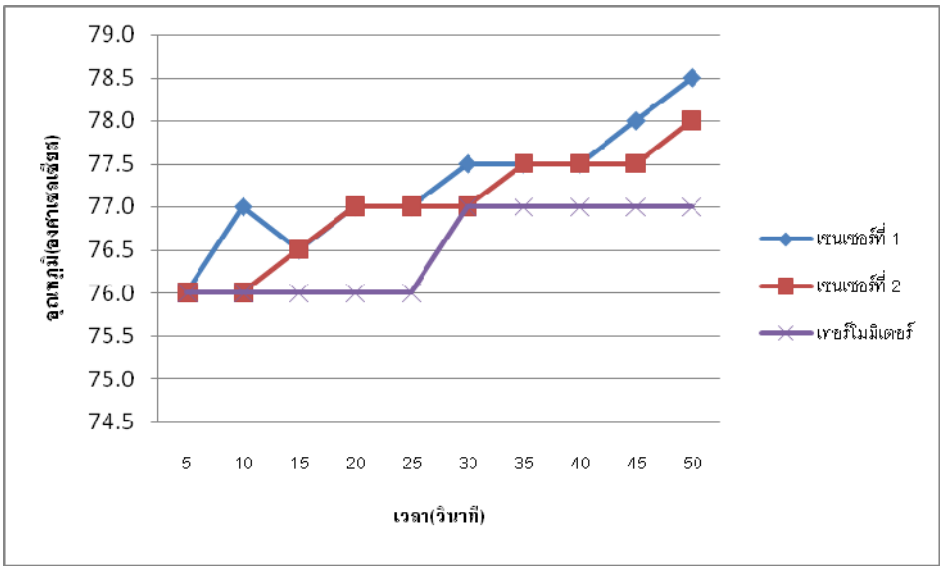
ครั้งที่	อุณหภูมิ(องศาเซลเซียส)				
	เวลา(วินาที)	เซนเซอร์ที่ 1	เซนเซอร์ที่ 2	เทอร์โมมิเตอร์	ความคลาดเคลื่อน (%)
1	5	26.0	26.5	26.0	0.96
2	10	26.5	26.5	26.0	1.92
3	15	27.0	26.5	26.0	2.88
4	20	27.0	26.5	27.0	-0.93
5	25	26.5	27.0	26.0	2.88
6	30	26.5	27.0	27.0	-0.93
7	35	26.5	26.5	26.0	1.92
8	40	27.0	27.5	27.0	0.93
9	45	27.0	27.5	27.0	0.93
10	50	26.5	27.5	27.0	0.00



รูปที่ 4.9 กราฟอุณหภูมิจากโปรแกรมทั้ง 2 จุดเทียบกับเทอร์โมมิเตอร์ที่อุณหภูมิห้อง

ตารางที่ 4.3 แสดงอุณหภูมิจากโปรแกรมทั้ง 2 จุดเทียบกับเทอร์โมมิเตอร์ที่อุณหภูมิสูง

ครั้งที่	อุณหภูมิ(องศาเซลเซียส)				
	เวลา(วินาที)	เซนเซอร์ที่ 1	เซนเซอร์ที่ 2	เทอร์โมมิเตอร์	ความคลาดเคลื่อน (%)
1	5	76.0	76.0	76.0	0.00
2	10	77.0	76.0	76.0	0.66
3	15	76.5	76.5	76.0	0.66
4	20	77.0	77.0	76.0	1.32
5	25	77.0	77.0	76.0	1.32
6	30	77.5	77.0	77.0	0.32
7	35	77.5	77.5	77.0	0.65
8	40	77.5	77.5	77.0	0.65
9	45	78.0	77.5	77.0	0.97
10	50	78.5	78.0	77.0	1.62



รูปที่ 4.10 กราฟอุณหภูมิจากโปรแกรมทั้ง 2 จุดเทียบกับเทอร์โมมิเตอร์ที่อุณหภูมิสูง

จากการทดลองค่าเมื่อ เรานำเครื่องวัดอุณหภูมิ มาทำการทดสอบเทียบกับ เทอร์โมมิเตอร์ซึ่งจะวัดที่อุณหภูมิอระดับต่างๆคือ อุณหภูมิต่ำใช้ความเย็นจากแอร์ทำความเย็น อุณภูมิห้องจากห้อง และอุณหภูมิสูง จากไดเป่าผม ซึ่งค่าที่วัดได้จะปรียบเทียบตามตารางที่ 4.1-4.3 จะสังเกตได้ว่าค่าที่วัดได้เมื่อเทียบกับเทอร์โมมิเตอร์จะมีค่าที่ไกล้เคียงกันเมื่อคิดเป็น % ความคลาดเคลื่อน

บทที่ 5
สรุป และข้อเสนอแนะ

การออกแบบ และสร้างเครื่องวัดอุณหภูมิไร้สาย เพื่อสามารถเก็บผลอุณหภูมิ ได้ถี่ขึ้นตามความเหมาะสม การเรียกดูข้อมูลทำได้สะดวกขึ้นเพื่อนำไปติดตั้งบริเวณที่ไม่สามารถเดินสายได้ หรือต้องมีการเคลื่อนย้ายบ่อยครั้ง ซึ่งในการทำงานจะสลับการส่งค่าอุณหภูมิซึ่งในบทนี้จะมีรายละเอียดดังนี้

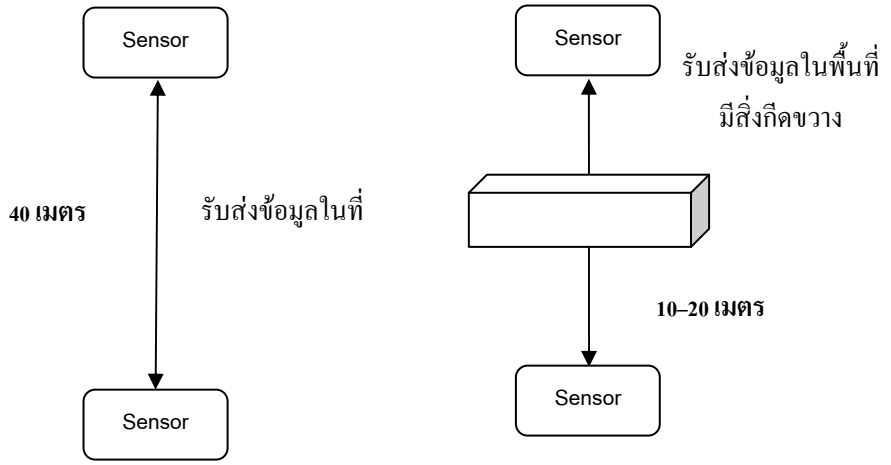
5.1 สรุปผลของโครงงาน

5.2 อุปสรรคและปัญหาของโครงงาน

5.3 ข้อเสนอแนะ

5.1 สรุปผลของโครงงาน

จากการทดลองในบทที่ 4 จะมีค่าที่ใกล้เคียงกับค่ามาตรฐานเมื่อเทียบกับ เทอร์โมมิเตอร์เราสามารถที่จะทราบค่าอุณหภูมิที่ต้องการทราบได้.เพื่อที่จะนำไปใช้ในห้อง server และสามารถนำไปใช้ในงานอื่นได้แล้วแต่ความเหมาะสม.ในส่วนของระยะทางของการรับส่งข้อมูลจะขึ้นอยู่กับสิ่งกีดขวางดังรูปที่ 5.1



รูปที่5.1 แสดงลักษณะระยะทางในการรับและส่งข้อมูล

5.2 อุปสรรคและปัญหาของโครงงาน

5.2.1 เกิดความถี่รบกวนที่ โมดูลและไมโครคอนโทรเลอร์ จากอุปกรณ์การสื่อสารต่างๆคือการทดลองในห้องทดลองต้องหาจุดที่ไม่มีการรบกวนในเรื่องความถี่

5.2.2 การส่งข้อมูลช้าเพราะโมดูลตัวส่งมีการส่งข้อมูลที่เร็วมากจนทำให้ตัวโมดูลแฮ้งแก้ปัญหาโดยการตั้งค่าการหน่วงเวลาการส่งของโกรแกรมที่ไมโครคอนโทรเลอร์ ให้มีการส่งค่าข้อมูลที่ช้ากว่าเดิม

5.2.3 ระยะในการ ส่ง-รับ ไปได้ไม่ไกลเท่าที่ควรเพราะสายอากาศคนละย่านความถี่ เพราะความถี่ที่ใช้งาน 433 MHz ซึ่งเราใช้สายอากาศที่ความถี่ 2.4 GHz ทำให้ระยะที่ได้ลดลง

5.3 ข้อเสนอแนะ

5.3.1ในรูปแบบของเตือนเมื่ออุณหภูมิเกินกำหนดควรพัฒนาโปรแกรมภาษา Visual Basic ให้มีรูปแบบการเตือนที่เหมาะกับการใช้งานเพราะความสามารถของโปรแกรมสามารถทำได้ เช่น รูปแบบเสียงเตือน ระบบ SMS

5.3.2 โมดูลรับ URM-433 มีกำลังการขยายต่ำน่าจะมีการพัฒนาให้มีการเพิ่มในส่วนของภาคขยาย ของตัวโมดูลเพื่อที่จะทำให้ประสิทธิภาพในการรับ-ส่งข้อมูลได้ไกลขึ้น

เอกสารอ้างอิง

1. Gary B. Shelly, Thomas J. Cashman, Judy A. Serwatka.(2544) **การสื่อสารข้อมูลระดับพื้นฐาน** แปล และเรียบเรียงโดย สัลยุทธ์ สว่างวรรณ. กรุงเทพมหานคร: ทอมสัน
2. กฤษดา ใจเย็น, อรรถพล บุญยะโภคา และชัยวัฒน์ ลิ้มพรจิตรวิไล. **เรียนรู้และปฏิบัติการเชื่อมคอมพิวเตอร์กับอุปกรณ์ภาพนอกผ่านพอร์ตอนุกรม.** กรุงเทพมหานคร : บริษัท อินโนเวตีฟ เอ็กเพอริเมนต์ จำกัด.
3. ธานิน สิทธะรรมชารี.(2546) **คู่มือการใช้งานโปรแกรม Visual Basic 6.0.**กรุงเทพมหานคร: บริษัท ซัลเซส มีเดีย จำกัด
4. นายศุภกิจ สักแสงโสภา , นายสุรจิต อมรพันธุ์ , นายวัชระ ขันเลื่อน **ครื่องวัดอุณหภูมิและ ความชื้นไร้สาย** (2550) ปริญญานิพนธ์ครุศาสตร์อิเล็กทรอนิกส์ และโทรคมนาคม มหาวิทยาลัยเทยโนโลยีราชมงคลธัญบุรี,
5. นายณรงค์ฤทธิ์ ธรรมประดิษฐ์ , นายพิมล ลิ้มลิขิตอักษร , นายฤทธิชัย หนุนวงษา (2547) **โปรแกรมวัดอุณหภูมิแบบเครือข่าย** คณะวิศวกรรมศาสตร์ มหาวิทยาลัยศรีนครินทรวิโรฒ

ผนวก ก

โปรแกรม Assembly

```
;       TEMP CONTROL 2 CH
;       CARV.PCB
SPAC:   DS      48
DP01:   DS      1
DP02:   DS      1
DP03:   DS      1
BUFF:   DS      1
TBUF:   DS      1
TST1:   DS      1
TST2:   DS      1
CT00:   DS      1
CT01:   DS      1
HOLD:   DS      1
KLOC    BIT     00H
CLON    BIT     01H
        ORG     0000H
        LJMP    INIT
        ORG     000BH
        LJMP    TIME
        ORG     0030H
INIT:   MOV     P0,#0FFH
        MOV     P2,#0FFH
        MOV     P1,#0FFH
        MOV     P3,#0FFH
        MOV     R0,#00          ; GP
        MOV     R1,#00          ; GP
        MOV     R2,#00          ;
        MOV     R3,#00          ;
```

```
        MOV     R4,#00              ;
        MOV     R5,#00              ; Delay
        MOV     R6,#00              ; Delay
        MOV     R7,#00              ; Delay
        MOV     DP01,#00
        MOV     DP02,#00
        MOV     DP03,#00
        MOV     TST1,#25
        MOV     TST2,#25
        MOV     CT00,#40
        MOV     CT01,#100
        MOV     HOLD,#180
        MOV     TMOD,#22H           ; Timer 0-1 8 Bit Auto Reload
        MOV     TH0,#256-231        ; Timer 0
        MOV     TL0,#256-231        ;
        SETB    TR0                 ; Start Timer
        SETB    IE.1                ;
        SETB    IE.7                ;
        MOV     SCON,#50H           ; Serial Mode 1
        MOV     TH1,#0FDH           ; 9600 Bps
        MOV     TL1,#0FDH
        SETB    TR1                 ; Start Timer
        CLR     RI                  ; Clear Receive Bit
        CLR     TI                  ; Clear Transmitt Bit
        LCALL   HHLY

;--->   Main Loop  ----------------------------------------------------
MAIN:   LCALL   HLAY
        LCALL   CVTM                ; Read Temp ( A )
        MOV     R4,A                ; Save Temp
```

```
        LCALL   DISP                ; LCD Display
        LCALL   CONT                ; Control
        LCALL   KEYB                ; Setting
        LJMP    MAIN                ; End Main Programe

;--->  Display Scan  ------------------------------------------------
DISP:   JNB     P3.4,DP10           ; Menu
        JNB     P3.5,DP20
DP00:   MOV     A,R4                ; Temp
        LCALL   DTCV
        MOV     SBUF,#'#'           ; Header
        JNB     TI,$
        CLR     TI
        MOV     R7,#00
        DJNZ    R7,$
        MOV     SBUF,#'A'           ;
        JNB     TI,$
        CLR     TI
        MOV     R7,#00
        DJNZ    R7,$
        LCALL   TXDP
        LJMP    EDIP
DP10:   MOV     SBUF,#'#'           ; Header
        JNB     TI,$
        CLR     TI
        MOV     R7,#00
        DJNZ    R7,$
        MOV     SBUF,#'B'           ; Set 2
        JNB     TI,$
        CLR     TI
```

```
        MOV     R7,#00
        DJNZ    R7,$
        MOV     A,TST1              ; HEX to BCD
        MOV     B,#10
        DIV     AB
        ORL     A,#00110000B        ; Data 1
        MOV     SBUF,A
        JNB     TI,$
        CLR     TI
        MOV     R7,#00
        DJNZ    R7,$
        MOV     A,B                 ; Data 2
        ORL     A,#00110000B
        MOV     SBUF,A
        JNB     TI,$
        CLR     TI
        MOV     R7,#00
        DJNZ    R7,$
        MOV     SBUF,#12            ; c
        JNB     TI,$
        CLR     TI
        MOV     R7,#00
        DJNZ    R7,$
        LJMP    EDIP
DP20:   MOV     SBUF,#'#'           ; Header
        JNB     TI,$
        CLR     TI
        MOV     R7,#00
        DJNZ    R7,$
        MOV     SBUF,#'C'           ; Set 2
```

```
        JNB     TI,$
        CLR     TI
        MOV     R7,#00
        DJNZ    R7,$
        MOV     A,TST2              ; HEX to BCD
        MOV     B,#10
        DIV     AB
        ORL     A,#00110000B        ; Data 1
        MOV     SBUF,A
        JNB     TI,$
        CLR     TI
        MOV     R7,#00
        DJNZ    R7,$
        MOV     A,B                 ; Data 2
        ORL     A,#00110000B
        MOV     SBUF,A
        JNB     TI,$
        CLR     TI
        MOV     R7,#00
        DJNZ    R7,$
        MOV     SBUF,#12            ; c
        JNB     TI,$
        CLR     TI
        MOV     R7,#00
        DJNZ    R7,$
        LJMP    EDIP
EDIP:   RET

;--->   Send Data to Display  ----------------------------------------
TXDP:   MOV     A,DP01
```

```
        ORL     A,#00110000B
        MOV     SBUF,A
        JNB     TI,$
        CLR     TI
        MOV     R7,#00
        DJNZ    R7,$
        MOV     A,DP02
        ORL     A,#00110000B
        MOV     SBUF,A
        JNB     TI,$
        CLR     TI
        MOV     R7,#00
        DJNZ    R7,$
        MOV     A,DP03
        ORL     A,#00110000B
        MOV     SBUF,A
        JNB     TI,$
        CLR     TI
        MOV     R7,#00
        DJNZ    R7,$
        RET

;--->   Data Convert  ------------------------------------------------
DTCV:   MOV     TBUF,A              ; Temp Buffer
        RR      A
        ANL     A,#01111111B
        LCALL   HTBC                ; HEX to BCD
        MOV     BUFF,A
        ANL     A,#11110000B
        SWAP    A
```

```
        MOV     DP01,A              ; Dig 1
        MOV     A,BUFF
        ANL     A,#00001111B
        MOV     DP02,A              ; Dig 2
        MOV     A,TBUF              ;
        JNB     ACC.0,DT20
        MOV     DP03,#05            ; 0.5
        LJMP    DT30
DT20:   MOV     DP03,#00            ; 0.0
DT30:   LJMP    ENCV
ENCV:   RET

;--->  Control  ----------------------------------------------------
CONT:   NOP
CT10:   CLR     C                   ; T < T-High
        MOV     A,R4
        RR      A
        ANL     A,#01111111B
        SUBB    A,TST1
        JNC     CT11
        SETB    P0.0                ; Off Load
        LJMP    CT12
CT11:   CLR     C                   ; T > T-Low
        MOV     A,R4
        RR      A
        ANL     A,#01111111B
        DEC     A
        SUBB    A,TST1
        JC      CT12
        CLR     P0.0                ; On Load
```

```
      LJMP    CT12
CT12:  NOP
CT20:  JB      TR0,CT22              ;
      CLR     C                     ; T < T-High
      MOV     A,R4
      RR      A
      ANL     A,#01111111B
      SUBB    A,TST2
      JNC     CT21
      SETB    P0.1                  ; Off Load
      LJMP    CT22
CT21:  CLR     C                     ; T > T-Low
      MOV     A,R4
      RR      A
      ANL     A,#01111111B
      DEC     A
      SUBB    A,TST2
      JC      CT22
      CLR     P0.1                  ; On Load
      LJMP    CT22
CT22:  NOP
ENCT:  RET

;--->  Key Setting  ------------------------------------------------
KEYB:  SETB    P3.4                  ; Key Press
      SETB    P3.5
      SETB    P3.6
      SETB    P3.7
KE10:  JB      P3.4,KE20             ; Temp 1 Set
KE11:  JB      P3.6,KE13             ; Up
```

```
        INC     TST1
KE12:   LCALL   DISP            ; Hold
        LCALL   HLAY
        JNB     P3.6,KE12
        LJMP    KE15
KE13:   JB      P3.7,KE15       ; Down
        DEC     TST1
KE14:   LCALL   DISP            ; Hold
        LCALL   HLAY
        JNB     P3.7,KE14
        LJMP    KE15
KE15:   NOP
        LJMP    EKEY
KE20:   JB      P3.5,KE30       ; Temp 2 Set
KE21:   JB      P3.6,KE23       ; Up
        INC     TST2
KE22:   LCALL   DISP            ; Hold
        LCALL   HLAY
        JNB     P3.6,KE22
        LJMP    KE25
KE23:   JB      P3.7,KE25       ; Down
        DEC     TST2
KE24:   LCALL   DISP            ; Hold
        LCALL   HLAY
        JNB     P3.7,KE24
        LJMP    KE25
KE25:   NOP
        LJMP    EKEY
KE30:   NOP
EKEY:   RET
```

```
;--->  Timer 0 Interrupt  ------------------------------------------
TIME:  DJNZ   CT00,ETIM            ; 0.01 Sec
       MOV    CT00,#40
       DJNZ   CT01,ETIM            ; 1 Sec
       MOV    CT01,#100
       CPL    CLON
       DJNZ   HOLD,ETIM
       CLR    TR0
ETIM:  RETI

;--->  Read Temp from DS1820  ---------------------------------------
IO1W   EQU    P1.0
CVTM:  MOV    A,#0CCH
       LCALL  WRTM
       MOV    A,#044H
       LCALL  WRTM
       JNB    IO1W,$
       LCALL  RSTM
       MOV    A,#0CCH
       LCALL  WRTM
       MOV    A,#0BEH
       LCALL  WRTM
       LCALL  RDTM
       LCALL  RSTM
       RET
WRTM:  MOV    R0,#08
WR00:  RRC    A
       JC     WR01
       CLR    IO1W
```

```
        MOV     R1,#15
        DJNZ    R1,$
        SETB    IO1W
        NOP
        NOP
        NOP
        NOP
        DJNZ    R0,WR00
        RET
WR01:   CLR     IO1W
        NOP
        NOP
        NOP
        NOP
        SETB    IO1W
        MOV     R1,#15
        DJNZ    R1,$
        DJNZ    R0,WR00
        RET
RDTM:   MOV     R0,#08
RD00:   CLR     IO1W
        NOP
        NOP
        NOP
        NOP
        SETB    IO1W
        NOP
        NOP
        NOP
        NOP
```

```
        MOV     C,IO1W
        MOV     R1,#15
        DJNZ    R1,$
        RRC     A
        DJNZ    R0,RD00
        RET
RSTM:   CLR     IO1W
        MOV     R1,#250
        DJNZ    R1,$
        SETB    IO1W
        MOV     R1,#10
        DJNZ    R1,$
PRTM:   JB      IO1W,$
        JNB     IO1W,$
        RET

;--->   Convert HEX to BCD  ------------------------------------------
HTBC:   MOV     R0,A                    ; Save Source
        MOV     B,#10
        CLR     C
        DIV     AB                      ; XX / 10
        MOV     B,#06
        CLR     C
        MUL     AB                      ; XX / 10 * 6
        CLR     C
        ADD     A,R0                    ; XX / 10 * 6 + XX
        RET
DLAY:   MOV     R7,#00
        DJNZ    R7,$
        RET
```

```
HLAY:   MOV     R6,#00
HL00:   MOV     R7,#00
        DJNZ    R7,$
        DJNZ    R6,HL00
        RET
HHLY:   MOV     R5,#06
HH00:   MOV     R6,#00
HH01:   MOV     R7,#00
        DJNZ    R7,$
        DJNZ    R6,HH01
        DJNZ    R5,HH00
        RET
        END
```

โปรแกรม Visual Basic

```
Dim intCount As Integer
Dim strTemp1 As Single
Dim strTemp2 As Single
Private Sub cmdExit_Click()
   End
End Sub

Private Sub cmdSavedata_Click()
   frmRecord.Show
End Sub

Private Sub cmdSetting_Click()
   With frmSetting
      .cmbTempLow = ""
      .cmbTempHigh = ""
      .txtIP1 = ""
      .txtIP2 = ""
      .txtIP3 = ""
      .txtIP4 = ""
   End With
   frmSetting.Show
   frmMain.Hide
End Sub

rivate Sub cmdStart_Click()
If frmSetting.cmbTempLow = "" Or frmSetting.cmbTempHigh = "" Then
   MsgBox "Setting incompleted", vbExclamation + vbOKOnly
   Exit Sub
```

```
 Else
    If Not MSComm1.PortOpen Then MSComm1.PortOpen = True
    timeTime.Enabled = True
    optCelsius.Enabled = True
    optFahrenheit.Enabled = True
    cmdSetting.Enabled = False
End If
End Sub

Private Sub cmdStop_Click()
    TimerAlarm.Enabled = False
    timeTime.Enabled = False
  If MSComm1.PortOpen = True Then MSComm1.PortOpen = False
    optCelsius.Enabled = False
    optFahrenheit.Enabled = False
    cmdSetting.Enabled = True
  If optCelsius.Value = True Then
     txtTemp1.Text = "--.- C"
     txtTemp2.Text = "--.- C"
     txtAverage.Text = "--.- C"
  Else
     txtTemp1.Text = "--.- F"
     txtTemp2.Text = "--.- F"
     txtAverage.Text = "--.- F"
  End If
End Sub

Private Sub Command1_Click()
    AlarmMSG
End Sub
```

```
Private Sub Form_Load()
Dim fs As New FileSystemObject
Dim i, j As Integer
  If fs.FileExists("C:\ipmsg.exe") = False Then fs.CopyFile (App.Path & "\ipmsg.exe"), ("C:\")
  If fs.FileExists("C:\windows\system32\MSCOMM32.OCX") = False Then fs.CopyFile
(App.Path & "\MSCOMM32.OCX"), ("C:\windows\system32\")
  If fs.FileExists("C:\windows\system32\CCXPButton.ocx") = False Then fs.CopyFile (App.Path
& "\CCXPButton.ocx"), ("C:\windows\system32\")
    With frmSetting
            .lblDegree.Caption = "C"
          For i = 5 To 50 Step 5
                .cmbTempLow.AddItem (i) & ".0"
                .cmbTempHigh.AddItem (i) & ".0"
          Next i
    End With
    With MSComm1
          .Settings = "9600,n,8,1"
          .CommPort = 1
          .InputLen = 1
          .RThreshold = 1
    End With
    optCelsius.Enabled = False
    optFahrenheit.Enabled = False
    lblTime.Caption = ""
    cmdSetting.TabIndex = 0
End Sub

Private Sub MSComm1_OnComm()
Dim strInput As String
Dim strInputTemp As String
```

```
Dim strInputTemp1 As String
Dim i As Integer
Dim j As Integer
   DoEvents
   strInput = MSComm1.Input
   If strInput = "A" Then
      For i = 1 To 3
         strInputTemp = strInputTemp & MSComm1.Input
      Next i
      Debug.Print strInputTemp
         If Not IsNumeric(strInputTemp) Or Len(strInputTemp) <> 3 Then Exit Sub
         strTemp1 = Format(Mid(strInputTemp, 1, 2) & "." & Mid(strInputTemp, 3, 3), "00.0")
         timeTime.Enabled = True
   End If
   If strInput = "B" Then
      For j = 1 To 3
         strInputTemp1 = strInputTemp1 & MSComm1.Input
      Next j
      Debug.Print strInputTemp1
         If Not IsNumeric(strInputTemp1) Or Len(strInputTemp1) <> 3 Then Exit Sub
         strTemp2 = Format(Mid(strInputTemp1, 1, 2) & "." & Mid(strInputTemp1, 3, 3), "00.0")
         timeTime.Enabled = True
   End If
End Sub
Private Sub TimerAlarm_Timer()
      intCount = intCount + 1
      If intCount = 60 Then Call AlarmMSG
      If Second(Time) Mod 2 = 0 Then
         frmAlarm.Hide
      Else
```

```
            frmAlarm.Show
        End If
End Sub

Private Sub AlarmMSG()
Dim strInfo As String
Dim CMMD            As String
intCount = 0
strInfo = frmAlarm.lblAlarm & vbCrLf
strInfo = strInfo & "By Admin server"
'    CMMD = "C:/ipmsg.exe /msg /log " & "192.168.1.3" & " " & strInfo
    If frmSetting.txtIP1 <> "" Then
        CMMD = "C:/ipmsg.exe /msg /log " & txtIP1.Text & " " & strInfo
        dwProcessId = Shell(CMMD, 0)
    End If
    If frmSetting.txtIP2 <> "" Then
        CMMD = "C:/ipmsg.exe /msg /log " & txtIP1.Text & " " & strInfo
        dwProcessId = Shell(CMMD, 0)
    End If
    If frmSetting.txtIP3 <> "" Then
        CMMD = "C:/ipmsg.exe /msg /log " & txtIP1.Text & " " & strInfo
        dwProcessId = Shell(CMMD, 0)
    End If
    If frmSetting.txtIP4 <> "" Then
        CMMD = "C:/ipmsg.exe /msg /log " & txtIP1.Text & " " & strInfo
        dwProcessId = Shell(CMMD, 0)
    End If

    DoEvents
End Sub
```

```
Private Sub timeTime_Timer()
    lblTime.Caption = Time
    DoEvents
      If optCelsius.Value = True Then
      'Show data
          txtTemp1.Text = Format(strTemp1, "00.0") & " C"
          txtTemp2.Text = Format(strTemp2, "00.0") & " C"
          txtAverage.Text = Format((CSng(Mid(txtTemp1.Text, 1, 4)) +
CSng(Mid(txtTemp2.Text, 1, 4))) / 2, "00.0") & " C"
          lblTempControl.Caption = frmSetting.cmbTempLow.Text & " C - " &
frmSetting.cmbTempHigh.Text & " C"
      'check for alarm
          If txtAverage.Text > frmSetting.cmbTempHigh Or txtAverage.Text <
frmSetting.cmbTempLow Then
             TimerAlarm.Enabled = True
             timeTime.Enabled = False
             MSComm1.PortOpen = False
          End If
      Else
      'Show data
          txtTemp1.Text = Format(((9 * strTemp1) / 5) + 32, "00.0") & " F"
          txtTemp2.Text = Format(((9 * strTemp2) / 5) + 32, "00.0") & " F"
          txtAverage.Text = Format((CSng(Mid(txtTemp1.Text, 1, 4)) +
CSng(Mid(txtTemp2.Text, 1, 4))) / 2, "00.0") & " F"
          lblTempControl.Caption = Format((9 * frmSetting.cmbTempLow.Text / 5) + 32,
"00.0") & " F - " & Format((9 * frmSetting.cmbTempHigh.Text / 5) + 32, "00.0") & " F"
      'check for alarm
```

```
            If Format(CSng(Mid(txtAverage.Text, 1, 4)), "00.0") < Format((9 *
frmSetting.cmbTempLow.Text / 5) + 32, "00.0") Or Format(CSng(Mid(txtAverage.Text, 1, 4)),
"00.0") > Format((9 * frmSetting.cmbTempHigh.Text / 5) + 32, "00.0") Then
                TimerAlarm.Enabled = True
                timeTime.Enabled = False
                MSComm1.PortOpen = False
            End If
        End If
End Sub
```

ผนวก ข.

URM-433

Multi-Channel FSK Transceiver Module

Features:

- Addressable Point-to-Point and Point-to-Multipoint RF networks.
- Broadcast Multidrop mode.
- Selectable RF 38 channels
- RS232 interface at TTL level format 8N1.
- DTE speeds 600-115,200bps.
- Air data speeds 4,800-19,200bps.
- Flow control of hardware.
- On –air error checking and data acknowledgements.
- Supply voltage 3.3 to 5v and low operating current.
- Built-in AT command line configuration.
- Remote over-air configuration.

Applications:

- Active RF-ID systems
- Alarm & security systems
- Home automation
- Telemetry
- Wireless payment systems

Absolute Ratings:

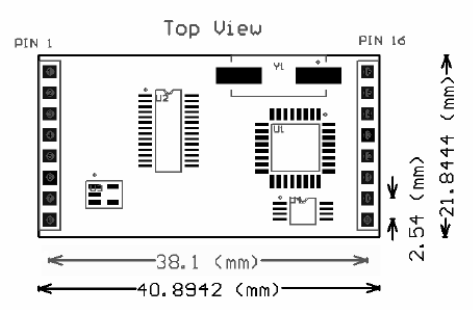
- Operating Temperature -20 ºC to +70ºC
- Storage Temperature -40 ºC to +100ºC
- Supply Voltage +5V
- Input Voltage -1.0 to Vcc+0.3V
- Output Voltage -1.0 to Vcc+0.3V

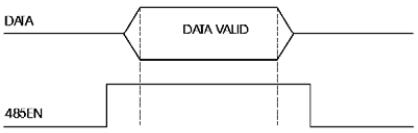
Technical Features:

	Min.	Typ.	Max.	Unit
DC Levels				
Supply voltage	2.7	3.3	5	V
Supply Current (rx mode)		15		mA
Supply Current (tx mode @ -5dBm)		14		mA
Supply Current (tx mode @ 0dBm)		16		mA
Supply Current (tx mode @ 10dBm)		33		mA
Supply Current (power down mode)		<1		mA
Input/output logic level "1"	0.7xVcc		Vcc	V
Input/output logic level "0"	0		0.3xVcc	V
RF				
Frequency band	418 - 438			MHz
Modulation type		FSK		
Receiver sensitivity		-104		dBm
RF output power(tx)	-5		10	dBm
Performance				
Input Bit Rate*	600-115,200			bps
Out door range		>100		m
Number of channels			38	
Channel spacing		524		kHz
Switching times				
PWRDN →RX			10	ms
TX↔RX			10	ms
Default values				
Channel		0		
RF output power(tx)		10		dBm

* Input signal has to be up of 1 start bit 8 data bit,1 stop bit and no parity.

Pin Description



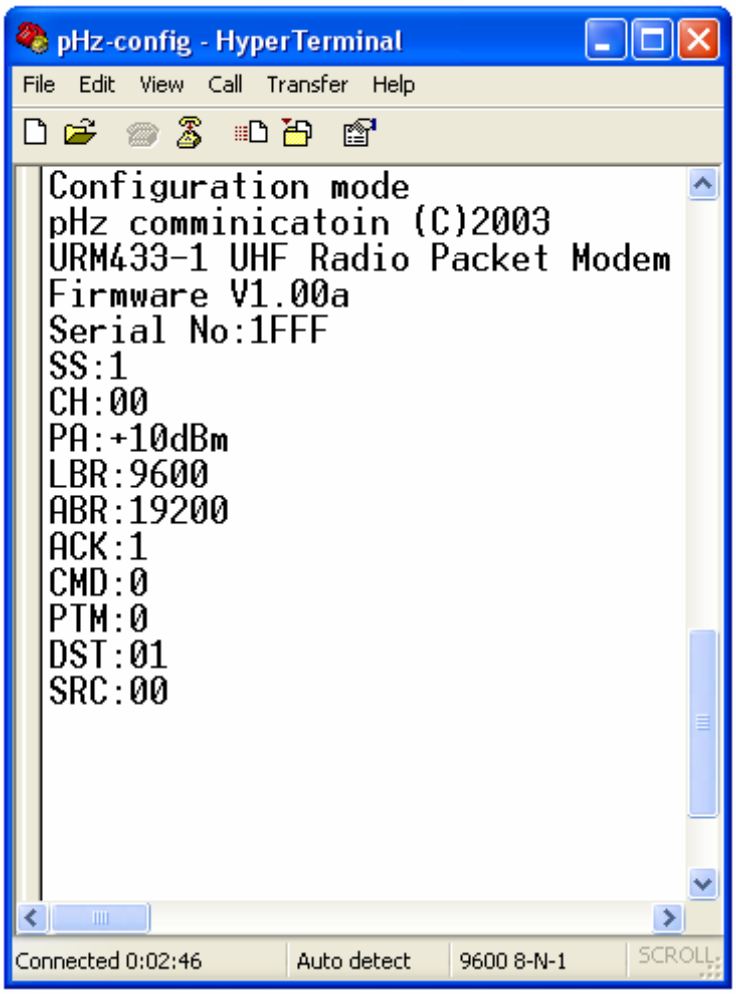
Pin Name	Pin Num.	Description
RF GND	1,3	Connection for the ground plane of the RF part.
ANT	2	Connection for the antenna 50 ohm impedance
G1	4	General pin1 **input** for Power Down Active **LOW**
G0	5	General pin0 **output** for status LED Active **LOW**
CONF	6	Configuration AT commands pin **input** Active **LOW**
+5V	7	DC supply Voltage 3.3 to 5V
GND	8,16	Ground connection
CD	9	Carrier Detect **output** Active **LOW**
G2	10	No connection
RST	11	Reset on-board **input** Active **LOW**
RXD	12	Received data **input** RS232 (TTL)
TXD	13	Transmitted data **output** RS232 (TTL)
CTS	14	Enable transmitted data on RS485 pin **output** 
RTS	15	Flow control **output** to Host (DTE)

<table>
<tr>
<td>ATCMD[n]</td>
<td>Set Enables remote over-air configuration (CMD)
<table>
<tr><th>n</th><th>Mode</th></tr>
<tr><td>0</td><td>Disable</td></tr>
<tr><td>1</td><td>Enable</td></tr>
</table>
</td>
<td>OK</td>
</tr>
<tr>
<td>ATPA[n]</td>
<td>Set RF output power levels (PA)
<table>
<tr><th>n</th><th>Output power</th></tr>
<tr><td>0</td><td>-5</td></tr>
<tr><td>1</td><td>0</td></tr>
<tr><td>2</td><td>5</td></tr>
<tr><td>3</td><td>10</td></tr>
</table>
</td>
<td>OK</td>
</tr>
<tr>
<td>ATSS[n]</td>
<td>Slave Select (SS)
<table>
<tr><th>n</th><th>Mode</th></tr>
<tr><td>0</td><td>Master</td></tr>
<tr><td>1</td><td>Slave</td></tr>
</table>
</td>
<td>OK</td>
</tr>
<tr>
<td>ATPTM[n]</td>
<td>Set Point-to- Multipoint mode (PTM)
<table>
<tr><th>n</th><th>Mode</th></tr>
<tr><td>0</td><td>Disable</td></tr>
<tr><td>1</td><td>Enable</td></tr>
</table>
</td>
<td>OK</td>
</tr>
</table>

Configuration AT commands

Following AT commands are supported upon successful implementation of each command. The module will return either a value or OK.

<table>
<tr><th>Command</th><th>Action</th><th>Response</th></tr>
<tr><td>ATZ</td><td>Reset the module</td><td>OK</td></tr>
<tr><td>ATCH[nn]</td><td>Set channels frequency (CH) nn = 00-37 (Dec)</td><td>OK</td></tr>
<tr><td>ATDA[nn]</td><td>Set Destination address (DST) nn = 00-FF (Hex)</td><td>OK</td></tr>
<tr><td>ATSA[nn]</td><td>Set Source address (SCR) nn = 00-FF (Hex)</td><td>OK</td></tr>
<tr><td>ATLBR[n]</td><td>Set DTE baud rate (LBR)
<table>
<tr><th>n</th><th>baud rate</th></tr>
<tr><td>0</td><td>600</td></tr>
<tr><td>1</td><td>1200</td></tr>
<tr><td>2</td><td>2400</td></tr>
<tr><td>3</td><td>4800</td></tr>
<tr><td>4</td><td>9600</td></tr>
<tr><td>5</td><td>19200</td></tr>
<tr><td>6</td><td>38400</td></tr>
<tr><td>7</td><td>57600</td></tr>
<tr><td>8</td><td>115200</td></tr>
</table></td><td>OK</td></tr>
<tr><td>ATABR[n]</td><td>Set Air speed bits rate (ABR)
<table>
<tr><th>n</th><th>Bits rate</th></tr>
<tr><td>0</td><td>4800</td></tr>
<tr><td>1</td><td>9600</td></tr>
<tr><td>2</td><td>19200</td></tr>
</table></td><td>OK</td></tr>
<tr><td>ATACK[n]</td><td>Set Enables transfer acknowledgments (ACK)
<table>
<tr><th>n</th><th>Mode</th></tr>
<tr><td>0</td><td>Disable</td></tr>
<tr><td>1</td><td>Enable</td></tr>
</table></td><td>OK</td></tr>
</table>



Configuration using HyperTerminal

Note: The HyperTerminal should be set Flow control disable. Default baud rate 9600bps and set pin CONF to LOW for Configuration AT commands.

;Command Aspect Script. AT command		
;Carriage Return: <CR> ASCII 13		
พิมพ์คำสั่ง	“ATDS01”<CR>	;ตั้งค่าชื่อผู้รับของชุดข้อมูลเป็นชื่อ 01
รอตอบกลับ	“OK”	;ยืนยันการตั้งค่า
พิมพ์คำสั่ง	“ATACK1”<CR>	;ตั้งค่าโมดูลรอการตอบกลับ
รอตอบกลับ	“OK”	;ยืนยันการตั้งค่า

อธิบายการทำงาน

การควบคุมการไหลของข้อมูล (Flow control)

เนื่องจากโมดูลมีหน่วยความจำบัฟเฟอร์รองรับรับข้อมูลจำกัด ดังนั้นจึงต้องมีกลไกควบคุมการไหลของข้อมูล

ให้สัมพันธ์กัน เพื่อป้องกันไม่ให้รับข้อมูลผิดพลาด โดยใช้ ขา RTS Flow control (Hardware)
RTS : Request to Send หากขา RTS มีลอจิกเป็น “1” ให้อุปกรณ์ โฮสต์ (Host) หยุดการส่งข้อมูลชั่วคราว
การจัดสรรการใช้ช่องสัญญาณ (Carrier Sense Multiple Access/Collision Avoidance)
เป็นการจัดสรรการเข้าใช้ช่องสัญญาณของแต่ล่ะช่องสัญญาณ ซึ่งการทำงานของกลไกนี้เรียกย่อๆว่า CSMA/CA คือเมื่อโมดูลหนึ่งต้องการเข้าใช้ช่องสัญญาณ จะต้องตรวจสอบช่องสัญญาณก่อนว่ามีการใช้ช่องสัญญาณ
อยู่หรือไม่ และรอจนกว่าช่องสัญญาณว่าง เมื่อช่องสัญญาณว่างจะต้องรอต่อไปอีกระยะหนึ่ง (Random Back-off)
ด้วยการสุ่มค่าเวลา สุ่มได้ระยะเวลาน้อยกว่า ก็มีสิทธิในการใช้ช่องสัญญาณก่อน

รูปแบบการจัดการสื่อสาร

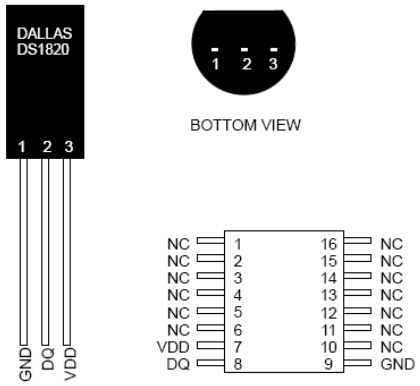
โมดูลตัวนี้รองรับการทำงาน 4 โหมด คือ
1 การสื่อสารระหว่างจุดต่อจุด(Point-To-Point) ส่งข้อมูลเป็นแพ็กเกต รอการตอบกลับ (Acknowledge) และส่งซ้ำหากไม่ได้รับการตอบกลับ
2 การสื่อสารระหว่างจุดต่อหลายจุด(Point-To-Multipoint) พร้อมรอการตอบกลับ ควบคุมด้วยชุดคำสั่งเศษโดยตรง
3 การสื่อสารแบบแพร่กระจาย(Broadcast Multi-drop) ไม่มีการรอตอบกลับ (Unacknowledged)
4 สื่อสารด้วยคำสั่งระยะไกล (Remote configured)

DS1820

FEATURES

- Unique 1–Wire™ interface requires only one port pin for communication
- Multidrop capability simplifies distributed temperature sensing applications
- Requires no external components
- Can be powered from data line
- Zero standby power required
- Measures temperatures from –55°C to +125°C in 0.5°C increments. Fahrenheit equivalent is –67°F to +257°F in 0.9°F increments
- Temperature is read as a 9–bit digital value.
- Converts temperature to digital word in 200 ms (typ.)
- User–definable, nonvolatile temperature 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



DS1820
PR35 PACKAGE
See Mech. Drawings Section

DS1820S
16–PIN SSOP
See Mech. Drawings Section

PIN DESCRIPTION

GND	–	Ground
DQ	–	Data In/Out
V_{DD}	–	Optional V_{DD}
NC	–	No Connect

DESCRIPTION

The DS1820 Digital Thermometer provides 9–bit temperature readings which indicate the temperature of the device.

Information is sent to/from the DS1820 over a 1–Wire interface, so that only one wire (and ground) needs to be connected from a central microprocessor to a DS1820. Power for reading, writing, and performing temperature conversions can be derived from the data line itself with no need for an external power source.

Because each DS1820 contains a unique silicon serial number, multiple DS1820s can exist on the same 1–Wire bus. This allows for placing temperature sensors in many different places. Applications where this feature is useful include HVAC environmental controls, sensing temperatures inside buildings, equipment or machinery, and in process monitoring and control.

DETAILED PIN DESCRIPTION

PIN 16–PIN SSOP	PIN PR35	SYMBOL	DESCRIPTION
9	1	GND	**Ground.**
8	2	DQ	**Data Input/Output pin.** For 1–Wire operation: Open drain. (See "Parasite Power" section.)
7	3	V_{DD}	**Optional V_{DD} pin.** See "Parasite Power" section for details of connection.

DS1820S (16–pin SSOP): All pins not specified in this table are not to be connected.

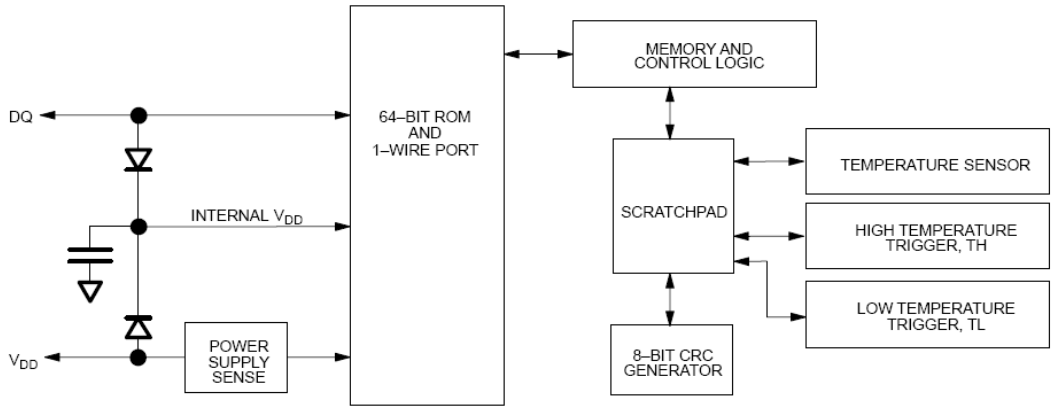
OVERVIEW

The block diagram of Figure 1 shows the major components of the DS1820. The DS1820 has three main data components: 1) 64–bit lasered ROM, 2) temperature sensor, and 3) nonvolatile temperature alarm triggers TH and TL. The device derives its power from the 1–Wire communication line by storing energy on an internal capacitor during periods of time when the signal line is high and continues to operate off this power source during the low times of the 1–Wire line until it returns high to replenish the parasite (capacitor) supply. As an alternative, the DS1820 may also be powered from an external 5 volts supply.

Communication to the DS1820 is via a 1–Wire port. With the 1–Wire port, the memory and control functions will not be available before the ROM function protocol has been established. The master must first provide one of five ROM function commands: 1) Read ROM, 2) Match ROM, 3) Search ROM, 4) Skip ROM, or 5) Alarm Search. These commands operate on the 64–bit lasered ROM portion of each device and can single out a specific device if many are present on the 1–Wire line as well as indicate to the Bus Master how many and what types of devices are present. After a ROM function sequence has been successfully executed, the memory and control functions are accessible and the master may then provide any one of the six memory and control function commands.

One control function command instructs the DS1820 to perform a temperature measurement. The result of this measurement will be placed in the DS1820's scratchpad memory, and may be read by issuing a memory function command which reads the contents of the scratchpad memory. The temperature alarm triggers TH and TL consist of one byte EEPROM each. If the alarm search command is not applied to the DS1820, these registers may be used as general purpose user memory. Writing TH and TL is done using a memory function command. Read access to these registers is through the scratchpad. All data is read and written least significant bit first.

DS1820 BLOCK DIAGRAM Figure 1



PARASITE POWER

The block diagram (Figure 1) shows the parasite powered circuitry. This circuitry "steals" power whenever the I/O or V_{DD} pins are high. I/O will provide sufficient power as long as the specified timing and voltage requirements are met (see the section titled "1–Wire Bus System"). The advantages of parasite power are two–fold: 1) by parasiting off this pin, no local power source is needed for remote sensing of temperature, and 2) the ROM may be read in absence of normal power.

In order for the DS1820 to be able to perform accurate temperature conversions, sufficient power must be provided over the I/O line when a temperature conversion is taking place. Since the operating current of the DS1820 is up to 1 mA, the I/O line will not have sufficient drive due to the 5K pull–up resistor. This problem is particularly acute if several DS1820's are on the same I/O and attempting to convert simultaneously.

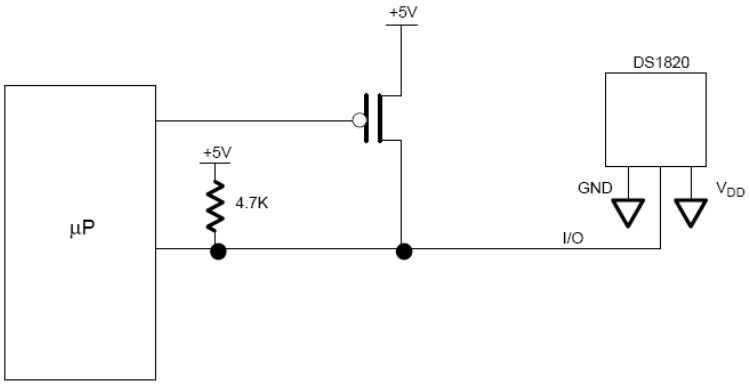
There are two ways to assure that the DS1820 has sufficient supply current during its active conversion cycle. The first is to provide a strong pull–up on the I/O line whenever temperature conversions or copies to the E^2 memory are taking place. This may be accomplished by using a MOSFET to pull the I/O line directly to the power supply as shown in Figure 2. The I/O line must be switched over to the strong pull–up within 10 µs maximum after issuing any protocol that involves copying to the E^2 memory or initiates temperature conversions. When using the parasite power mode, the V_{DD} pin must be tied to ground.

Another method of supplying current to the DS1820 is through the use of an external power supply tied to the V_{DD} pin, as shown in Figure 3. The advantage to this is that the strong pull–up is not required on the I/O line, and the bus master need not be tied up holding that line high during temperature conversions. This allows other data traffic on the 1–Wire bus during the conversion time. In addition, any number of DS1820's may be placed on the 1–Wire bus, and if they all use external power, they may all simultaneously perform temperature conversions by issuing the Skip ROM command and then issuing the Convert T command. Note that as long as the external power supply is active, the GND pin may not be floating.

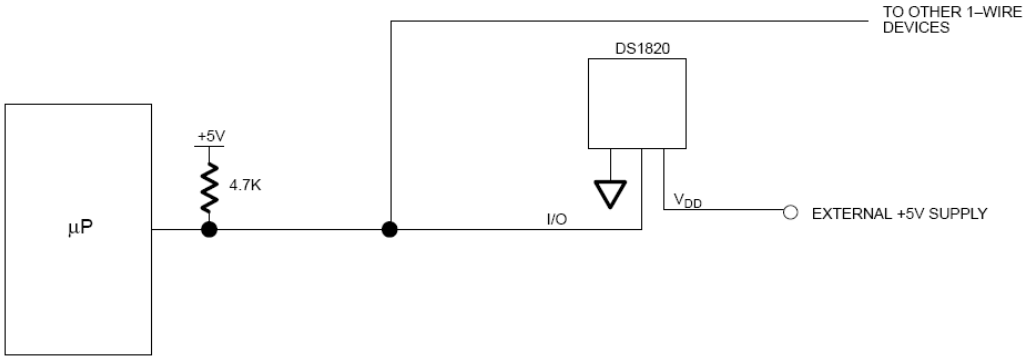
The use of parasite power is not recommended above 100°C, since it may not be able to sustain communications given the higher leakage currents the DS1820 exhibits at these temperatures. For applications in which such temperatures are likely, it is strongly recommended that V_{DD} be applied to the DS1820.

For situations where the bus master does not know whether the DS1820's on the bus are parasite powered or supplied with external V_{DD}, a provision is made in the DS1820 to signal the power supply scheme used. The bus master can determine if any DS1820's are on the bus which require the strong pull–up by sending a Skip ROM protocol, then issuing the read power supply command. After this command is issued, the master then issues read time slots. The DS1820 will send back "0" on the 1–Wire bus if it is parasite powered; it will send back a "1" if it is powered from the V_{DD} pin. If the master receives a "0", it knows that it must supply the strong pull–up on the I/O line during temperature conversions. See "Memory Command Functions" section for more detail on this command protocol.

STRONG PULL–UP FOR SUPPLYING DS1820 DURING TEMPERATURE CONVERSION Figure 2



USING V_{DD} TO SUPPLY TEMPERATURE CONVERSION CURRENT Figure 3



OPERATION – MEASURING TEMPERATURE

The DS1820 measures temperature through the use of an on–board proprietary temperature measurement technique. A block diagram of the temperature measurement circuitry is shown in Figure 4.

The DS1820 measures temperature by counting the number of clock cycles that an oscillator with a low temperature coefficient goes through during a gate period determined by a high temperature coefficient oscillator. The counter is preset with a base count that corresponds to –55°C. If the counter reaches zero before the gate period is over, the temperature register, which is also preset to the –55°C value, is incremented, indicating that the temperature is higher than –55°C.

At the same time, the counter is then preset with a value determined by the slope accumulator circuitry. This circuitry is needed to compensate for the parabolic behavior of the oscillators over temperature. The counter is then clocked again until it reaches zero. If the gate period is still not finished, then this process repeats.

The slope accumulator is used to compensate for the non–linear behavior of the oscillators over temperature, yielding a high resolution temperature measurement. This is done by changing the number of counts necessary for the counter to go through for each incremental degree in temperature. To obtain the desired resolution, therefore, both the value of the counter and the number of counts per degree C (the value of the slope accumulator) at a given temperature must be known.

Internally, this calculation is done inside the DS1820 to provide 0.5°C resolution. The temperature reading is provided in a 16–bit, sign–extended two's complement reading. Table 1 describes the exact relationship of output data to measured temperature. The data is transmitted serially over the 1–Wire interface. The DS1820 can measure temperature over the range of –55°C to +125°C in 0.5°C increments. For Fahrenheit usage, a lookup table or conversion factor must be used.

Note that temperature is represented in the DS1820 in terms of a $^1/_2$°C LSB, yielding the following 9–bit format:

MSB								LSB
1	1	1	0	0	1	1	1	0

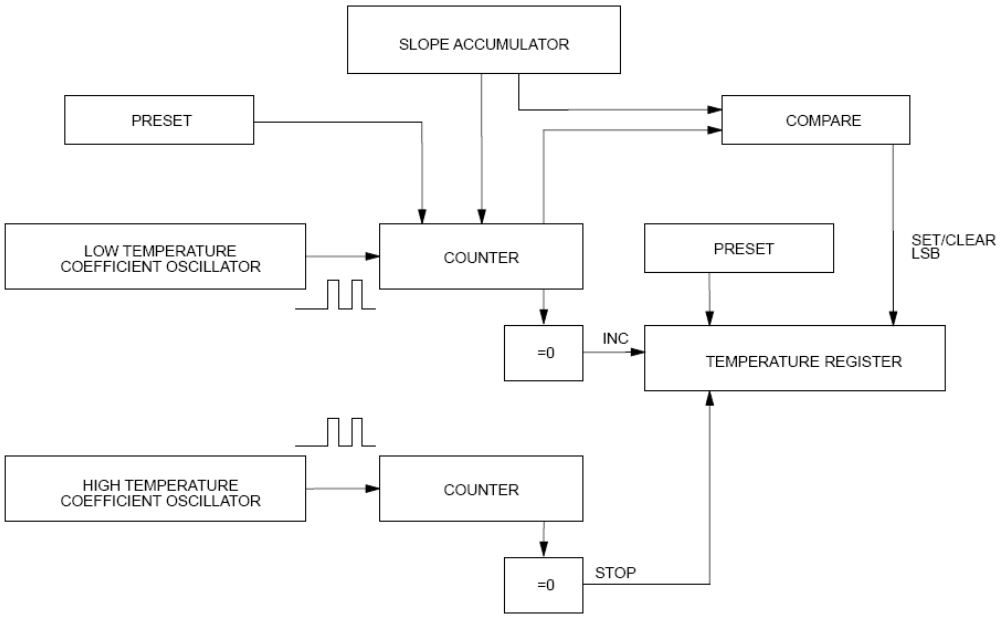
= –25°C

The most significant (sign) bit is duplicated into all of the bits in the upper MSB of the two–byte temperature register in memory. This "sign–extension" yields the 16–bit temperature readings as shown in Table 1.

Higher resolutions may be obtained by the following procedure. First, read the temperature, and truncate the 0.5°C bit (the LSB) from the read value. This value is TEMP_READ. The value left in the counter may then be read. This value is the count remaining (COUNT_REMAIN) after the gate period has ceased. The last value needed is the number of counts per degree C (COUNT_PER_C) at that temperature. The actual temperature may be then be calculated by the user using the following:

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

TEMPERATURE MEASURING CIRCUITRY Figure 4



TEMPERATURE/DATA RELATIONSHIPS Table 1

TEMPERATURE	DIGITAL OUTPUT (Binary)	DIGITAL OUTPUT (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

OPERATION – ALARM SIGNALING

After the DS1820 has performed a temperature conversion, the temperature value is compared to the trigger values stored in TH and TL. Since these registers are 8–bit only, the 0.5°C bit is ignored for comparison. The most significant bit of TH or TL directly corresponds to the sign bit of the 16–bit temperature register. If the result of a temperature measurement is higher than TH or lower than TL, an alarm flag inside the device is set. This flag is updated with every temperature measurement. As long as the alarm flag is set, the DS1820 will respond to the alarm search command. This allows many DS1820s to be connected in parallel doing simultaneous temperature measurements. If somewhere the temperature exceeds the limits, the alarming device(s) can be identified and read immediately without having to read non–alarming devices.

ประวัติย่อนิสิตผู้ทำโครงงาน

นายกิตติพงศ์ ทองหอม
เกิดวันที่ 16 กรกฎาคม 2527
ที่อยู่ปัจจุบัน 46 หมู่.2 ต.บ่อหิน อ.สิเกา จ.ตรัง 92150
การศึกษา ระดับประกาศนียบัตรวิชาชีพชั้นสูงจากมหาวิทยาลัยเทคโนโลยีราชมงคลสุวรรณภูมิ วิทยาเขตนนทบุรี
เข้าศึกษา กำลังศึกษาระดับ ปริญญา.ตรี ที่มหาวิทยาลัยศรีนครินทรวิโรฒ องครักษ์ แขนงวิชาโทรคมนาคม สาขาไฟฟ้า คณะวิศวกรรศาสตร์ เมื่อปีการศึกษา 2548

นายวรุฒน์ อุปศรี
เกิดวันที่ 21 มีนาคม 2522
ที่อยู่ปัจจุบัน 60/3 หมู่ 13 ต.โคกภู อ.ภูพาน จ.สกลนคร 47180
จบการศึกษา ระดับประกาศนียบัตรวิชาชีพชั้นสูงจากวิทยาลัยเทคนิคสกนคร
เข้าศึกษา กำลังศึกษาระดับ ปริญญา.ตรี ที่มหาวิทยาลัยศรีนครินทรวิโรฒ องครักษ์ แขนงวิชาโทรคมนาคม สาขาไฟฟ้า คณะวิศวกรรศาสตร์ เมื่อปีการศึกษา 2548

นางสาวสกุณา มากมูล
เกิดวันที่ 11 ตุลาคม 2527
ที่อยู่ปัจจุบัน 98/17 หมู่บ้านกรุงศรีธานี หมู่ 5 ต.วัดตูม อ.พระนครศรีอยุธยา จ.พระนครศรีอยุธยา 13000
จบการศึกษา ระดับประกาศนียบัตรวิชาชีพชั้นสูงจากวิทยาลัยเทคนิคสระบุรี
เข้าศึกษา กำลังศึกษาระดับ ปริญญา.ตรี ที่มหาวิทยาลัยศรีนครินทรวิโรฒ องครักษ์ แขนงวิชาโทรคมนาคม สาขาไฟฟ้า คณะวิศวกรรศาสตร์ เมื่อปีการศึกษา 2548