การใช้งานแบบ Remote ด้วยการใช้ไมโครคอนโทรลเลอร์

MICROCONTROLLER BASED REMOTE SENSING

นายพลากร พูลบำเพ็ญ

นางสาววิไลลักษณ์ กุลวัฒนาพันธ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมการวัดคุม

ภาควิชาวิศวกรรมการวัดคุม คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2548

MICROCONTROLLER-BASED REMOTE SENSING

PALAKORN POONBAMPEN

WILILUCK KUNWATTANAPUN

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

BACHELOR OF ENGINEERING IN INSTRUMENTATION ENGINEERING

DEPARTMENT OF INSTRUMENTATION ENGINEERING

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2005

ภาควิชาวิศวกรรมการวัดคุม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองปริญญานิพนธ์

หัวข้อปริญญานิพนธ์	การใช้งานแบบ Remote ด้วยการใช้ไมโครคอนโทรลเลอร์ Microcontroller Based Remote Sensing			
นักศึกษาผู้จัดทำ	นายพลากร	พูลบำเพ็ญ	**รหัสประจำตัว**	46015447
	นางสาววิไลลักษณ์	กุลวัฒนาพันธ์	**รหัสประจำตัว**	46015457
ปริญญา	วิศวกรรมศาสตรบัณฑิต			
สาขาวิชา	วิศวกรรมการวัดคุม			
ปีการศึกษา	2548			

อาจารย์ผู้ควบคุมปริญญานิพนธ์	**ลายมือชื่อ**
รศ. พิพัฒน์ เลาหสงคราม	

ภาควิชารับรองแล้ว

(รศ.ประสิทธิ์ จุลเสรีวงศ์)

หัวหน้าภาควิชาวิศวกรรมการวัดคุม

หัวข้อปริญญานิพนธ์	การใช้งานแบบ Remote ด้วยการใช้ไมโครคอนโทรลเลอร์			
	MICROCONTROLLER BASED REMOTE SENSING			
นักศึกษาผู้จัดทำ	นายพลากร	พูลบำเพ็ญ	**รหัสประจำตัว**	46015447
	นางสาววิไลลักษณ์	กุลวัฒนาพันธ์	**รหัสประจำตัว**	46015457
อาจารย์ที่ปรึกษา	รศ. พิพัฒน์ เลาหสงคราม			
ปีการศึกษา	2548			

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้ได้นำเสนอการควบคุมการรับสัญญาณ Remote ด้วยการใช้ไมโครคอนโทรลเลอร์ ในการทดลองเป็นการแปลงพอร์ตสื่อสารอนุกรม RS232 ให้เป็นสัญญาณความถี่วิทยุเพื่อใช้งานในลักษณะของการสื่อสารอนุกรมแบบไร้สาย (RF 2.4 GHz) การสื่อสารด้วยวิธีนี้มีข้อดีกว่าคือ สามารถรับส่งข้อมูลกันได้ในระยะทางที่ไกลกว่า RS232 หลายเท่าตัว ที่สำคัญคือไม่จำเป็นต้องใช้สายสัญญาณที่เป็นตัวนำสัญญาณทางไฟฟ้าในการสื่อสารข้อมูล ทำให้สามารถเปลี่ยนหรือเคลื่อนย้ายจุดรับส่งได้ง่าย การสั่งงานไปยังไมโครคอนโทรลเลอร์ และการแสดงค่าสถานะต่างๆ ของอุปกรณ์ที่ต่อร่วมกับไมโครคอนโทรลเลอร์ กระทำผ่านคอมพิวเตอร์ส่วนบุคคลในรูปแบบGraphic หรือที่เรียกว่า HMI (Human Machine Interface) และผลจากการทดลองเชื่อมต่อ (Interface) อุปกรณ์ในส่วน HMI ไมโครคอนโทรลเลอร์ และ Plant Model ปรากฏว่าสามารถใช้งานได้ดี

Thesis Title	Microcontroller Based Remote Sensing
Authors	Mr. Palakorn Poonbampen
	Miss. Wililuck Kunwattanapun
Thesis Advisor	Assoc. Prof.Phiphat Laohasongkram
Year	2005

ABSTRACT

This thesis is present about the Microcontroller based remote sensing. The project will be transform serial port communication (RS 232) to radio frequency wave communication for used in pattern wireless serial port communication (RF 2.4GHz), the wireless communication method is better than, because it is can receive or transmit in distance more than RS 232, very importance it is unnecessary cable signal for communication, and to be able change or remove through receiver / transmitter point is easy. There are commands from Microcontroller and status show each equipment interface with Microcontroller, it is make passed personal computer in graphic form, we are called HMI (Human Machine Interface). From experimental perform HMI interface with Microcontroller and Plant model equipments, this system used good working.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงได้ด้วยดีเพราะได้รับความเมตตาจาก รองศาตราจารย์ พิพัฒน์ เลาหสงคราม ที่ได้ให้คำปรึกษาและคำแนะนำแก่ผู้วิจัยตลอดมา อีกทั้งยังเอื้อเฟื้ออุปกรณ์และเครื่องมือต่างๆ ในการทำปริญญานิพนธ์ฉบับนี้ คณาจารย์ภาควิชาวิศวกรรมการวัดคุมทุกท่านที่ได้ให้คำแนะนำอันเป็นประโยชน์ ผู้ช่วยศาสตราจารย์ เชื้อ นกอยู่ ที่ให้หยิบยืมอุปกรณ์ในการทดลองผู้วิจัยรู้สึกซาบซึ้งและขอกราบขอบพระคุณเป็นอย่างสูง

และที่ลืมเสียมิได้คือ ขอกราบขอบพระคุณ คุณพ่อ คุณแม่ อันเป็นที่รักยิ่ง ที่สนับสนุนและเป็นแรงบันดาลใจในการทำปริญญานิพนธ์ฉบับนี้ ตลอดจนพี่ๆ เพื่อนๆ ทุกคนที่ให้คำแนะนำและความช่วยเหลือในด้านต่างๆ

คุณค่าและประโยชน์อันพึงมีจากปริญญานิพนธ์ฉบับนี้ ผู้วิจัยขอมอบแด่ผู้มีพระคุณทุกท่าน

คณะผู้จัดทำ

สารบัญ

สารบัญ(ต่อ)

สารบัญ(ต่อ)

สารบัญตาราง

สารบัญภาพ

สารบัญภาพ (ต่อ)

สารบัญภาพ (ต่อ)

บทที่ 1
บทนำ

1.1 ความเป็นมาและเหตุจูงใจของการวิจัย

กระบวนในปัจจุบันการสื่อสารแบบไร้สาย (Wireless) มีบทบาทเป็นอย่างมากในงานรูปแบบต่างๆ ที่เกี่ยวข้องกับอุปกรณ์อิเล็กทรอนิกส์หรือคอมพิวเตอร์ การใช้งานในส่วนของไมโครคอนโทรลเลอร์ก็เช่นกันวิธีการเชื่อมต่อส่วนใหญ่มักใช้พอร์ตสื่อสารอนุกรม RS232 แต่ในการใช้งานแบบ Remote ก็มีข้อจำกัดเรื่องของระยะทางของสายสัญญาณที่ไม่เกิน 50 ฟุต ประมาณ (15.24 เมตร) และเรื่องของความยืดหยุ่นในการใช้งาน บางครั้งส่วนที่เป็น Remote อาจต้องเคลื่อนที่อยู่ตลอดเวลา ปัญหาพวกนี้จะหมดไปถ้าหากข้อมูลที่เราส่งไปนั้นใช้อากาศเป็นตัวกลาง ด้วยเหตุนี้จึงเลือกการสื่อสารแบบไร้สายเชื่อมต่อการใช้งานไมโครคอนโทรลเลอร์

1.2 วัตถุประสงค์ของปริญญานิพนธ์

1. เพื่อศึกษาการใช้งานไมโครคอนโทรลเลอร์
2. ศึกษารูปแบบการสื่อสารแบบไร้สายและสามารถนำมาใช้งานได้จริง
3. เรียนรู้การใช้งานโปรแกรม Visual Basic เพื่อทำการมอนิเตอร์ค่าต่างๆ ที่รับมาได้

1.3 ขอบเขตของปริญญานิพนธ์

ปริญญานิพนธ์ฉบับนี้จะกล่าวถึง การสั่งงานและควบคุมไมโครคอนโทรลเลอร์ผ่านการสื่อสารแบบไร้สาย (Wireless) โดยใช้การติดต่อจากชุดรับส่งคลื่นความถี่วิทยุ 2.4 GHz ไปยังคอมพิวเตอร์ที่มีโปรแกรม Visual Basic คอยจัดการเกี่ยวกับการแสดงผลแบบกราฟฟิก

1.4 ขั้นตอนการศึกษา

การทำโครงงานวิจัยในปริญญานิพนธ์ฉบับนี้มีขั้นตอนเริ่มจากการศึกษาโปรแกรม KeilC51 ซึ่งเป็นโปรแกรมภาษา C ที่ใช้ในการเขียนเพื่อพัฒนาไมโครคอนโทรลเลอร์ในขณะเดียวกันก็ศึกษาการใช้งานไมโครคอนโทรลเลอร์ไปพร้อมกันด้วย เมื่อมีความรู้ในส่วนนี้แล้วก็ศึกษาโปรแกรม Visual Basic ในส่วนของการรับค่าผ่าน Hyper Terminal เพื่อที่จะนำค่าต่างๆ เหล่านั้นมาแสดงผลในรูปของภาพกราฟฟิก หลังจากนั้นก็ศึกษาการเชื่อมต่อแบบไร้สายเพื่อที่จะสามารถนำอุปกรณ์และโปรแกรมต่างๆ มา Interface กันได้

บทที่ 2
การศึกษาทฤษฎีและเครื่องมือต่างๆ ที่เกี่ยวข้องกับปริญญานิพนธ์

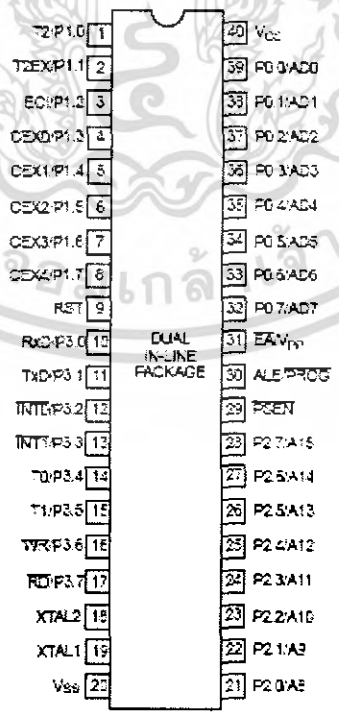
2.1 สถาปัตยกรรมและการพัฒนาไมโครคอนโทรลเลอร์

ในปัจจุบันมีไมโครคอนโทรลเลอร์มากมายหลายเบอร์ที่อยู่ในท้องตลาดแต่ละเบอร์ ก็จะมีหลายรุ่นให้เลือกใช้ตามความเหมาะสมทั้งในด้านความสามารถและราคา ไม่ว่าจะเป็นเบอร์อะไรจากบริษัทไหน ก็จะมีโครงสร้างหรือหลักการทำงานที่ไม่แตกต่างกันมากนัก ถ้าเราเข้าใจไมโครคอนโทรลเลอร์เบอร์ใดเบอร์หนึ่ง แล้วเราจะไปศึกษาไมโครคอนโทรลเลอร์อีกเบอร์หนึ่งก็ไม่ใช่เรื่องยากอีกต่อไป สิ่งที่สำคัญก็คือการเรียนรู้ไมโครคอนโทรลเลอร์สำหรับผู้เริ่มต้นนั้นควรเลือกเบอร์ที่เป็นที่นิยมใช้อยู่ในขณะนั้น และเป็นตัวที่เรียนรู้ได้ง่ายใช้ได้จริง

ในการศึกษาไมโครคอนโทรลเลอร์เพื่อทำปริญญานิพนธ์ฉบับนี้ได้เลือกใช้ไมโครคอนโทรลเลอร์ท้องตลาดและมีเครื่องมือพัฒนามากมาย

2.1.1 การออกแบบวงจรใช้งานให้กับ MCS-51

ในหัวข้อนี้จะศึกษาการทำงานของขาต่างๆ ของ MCS-51 และในขณะเดียวกันก็จะทำการศึกษาการออกแบบวงจรใช้งานให้กับ MCS-51 เพื่อให้ MCS-51 พร้อมที่จะทำงานตามคำสั่งที่เราโปรแกรมได้



ภาพที่ 2.1 แสดงรูปร่างและขาต่างๆของ MCS-51

2.1.1.1 หน้าที่การทำงานของขาต่างๆ ของ MCS-51

P0.0- P0.7 (ขา 39- 32) เป็นพอร์ตใช้งานทั่วไป, พอร์ต Data และ พอร์ต Address ไบต์ต่ำ

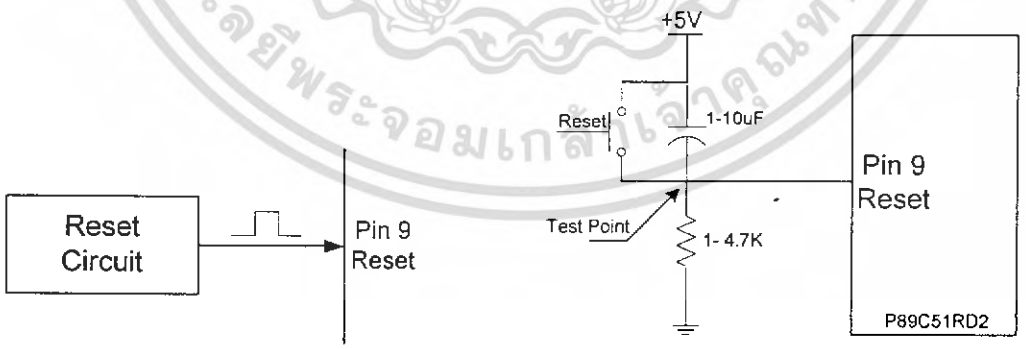
P1.0- P1.7 (ขา 1- 8) เป็นพอร์ตใช้งานทั่วไป

P2.0- P2.7 (ขา 21- 28) เป็นพอร์ตใช้งานทั่วไป และ พอร์ต Address ไบต์สูง

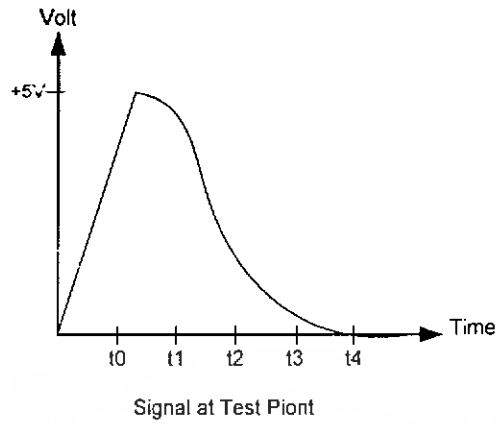
P3.0- P3.7 (ขา 10- 17) เป็นพอร์ตใช้งานทั่วไป นอกจากนั้นยังสามารถทำหน้าที่อื่นๆ ได้ดังนี้

- **P3.0, RXD** สามารถเป็นขารับสัญญาณ Serial Port
- P3.1, TXD สามารถเป็นขาส่งสัญญาณ Serial Port
- P3.2, INT0 สามารถเป็นขารับสัญญาณ Interrupt หมายเลข 0
- P3.3, INT1 สามารถเป็นขารับสัญญาณ Interrupt หมายเลข 1
- P3.4, T0 สามารถเป็นขารับสัญญาณพัลส์ หมายเลข 0 เพื่อเข้าวงจร Counter
- P3.5, T1 สามารถเป็นขารับสัญญาณพัลส์ หมายเลข 1 เพื่อเข้าวงจร Counter
- P3.6, WR สามารถเป็นขาสัญญาณเขียนข้อมูลไปยังอุปกรณ์ภายนอก
- P3.7, RD สามารถเป็นขาสัญญาณอ่านข้อมูลจากอุปกรณ์ภายนอก

ขา Reset (ขา 9) : เป็นขารีเซตการทำงานของ MCS-51 ให้เริ่มต้นทำงานใหม่ ขานี้ทำงานที่ลอจิก "1" สังเกตว่าไม่มีเครื่องหมาย – บนชื่อของขา ถ้ามีแสดงว่าทำงานที่ลอจิก "0" สิ่งที่เราต้องออกแบบวงจรก็คือ ต้องสร้างสัญญาณพัลส์บวกที่ขา Reset ในช่วงเวลาที่เราจ่ายไฟให้กับ MCS-51 (ตอนเปิดเครื่อง) เพราะฉะนั้นเมื่อเราเปิดเครื่องไมโครคอลโทรลเลอร์จะรีเซตตัวเองโดยอัตโนมัติ วงจรนี้สามารถสร้างได้โดยใช้ R และ C เพื่อสัญญาณพัลส์บวกดังภาพที่ 2.2



ภาพที่ 2.2 แสดงวงจร Reset และจุด Test Point



ภาพที่ 2.3 แสดงสัญญาณที่จุด Test Point

จากภาพที่ 2.3 เราจะใช้วงจร RC สร้างสัญญาณรีเซตให้กับ MCS-51 มาดูกันว่าวงจรนี้ทำงานกันอย่างไร โดยคุณสมบัติของ C เป็นตัวเก็บประจุไฟฟ้า เมื่อเราเริ่มจ่ายไฟเข้าไปในวงจร C จะเริ่มชาร์จตัวเองตอนนี้ C เปรียบเสมือนช็อต แรงดันตกคร่อมที่ C (Vc) = 0V แรงดันที่ตกคร่อมความต้านทาน R(Vr) = 5V จากนั้นเมื่อเวลาผ่านไป C ก็เริ่มชาร์จไฟมากขึ้น ทำให้มีแรงดัน (Vc) ตกคร่อมที่ตัวมันมากขึ้นแรงดันที่ความต้านทาน (Vr) ก็จะน้อยลง จนในที่สุด C ชาร์จไฟจนเต็มแรงดัน Vc = 5V และแรงดันตกคร่อมความต้านทาน Vr = 0V ตามภาพทางขวามือเวลาของสัญญาณพัลส์หรือความกว้างนั้นจะขึ้นอยู่กับค่า R , C ค่าที่เหมาะสมของ R ประมาณ 1K - 10K ส่วนค่าที่เหมาะ-สมของ C ประมาณ 1μF – 10μF

สามารถหาค่าเวลาที่ได้จากวงจร RC จากสูตร $t = 4RC$

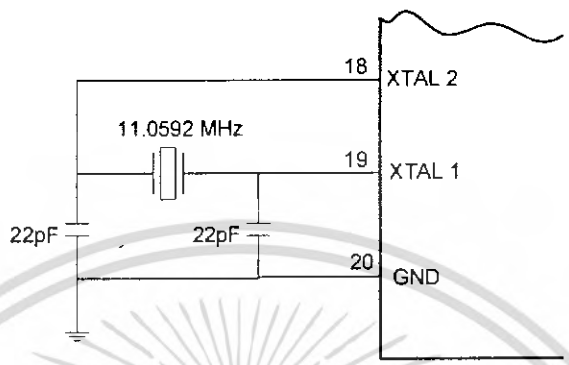
เช่นสมมติ ค่า R = 1K, C = 1μF ดังนั้น $t = 4(1000)(0.000001)$

$= 4$ mS

สวิทช์ที่ต่อคร่อม C นั้นทำหน้าที่เป็นตัว Discharge C และ Reset ให้กับ MCS-51 จะใช้ในกรณีที่เราทำการ Run โปรแกรมที่เราเขียน แล้วต้องการดูผลการ Run ใหม่ เพราะในบางครั้งที่เราต้องการดูผลการ Run ใหม่ถ้าไม่มีสวิทช์ Reset เวลาเราจะ Run โปรแกรมใหม่ก็ต้องถอดไฟเลี้ยงออกจากวงจร แล้วต่อไฟเลี้ยงให้วงจร MCS-51 ใหม่ถึงจะ Reset ได้

ขา XTAL2, XTAL1 (ขา 18, 19) : ไว้สำหรับต่อ X-TAL เพื่อสร้างสัญญาณนาฬิกาให้กับ MCS-51 เพราะว่าการทำงานของไมโครคอนโทรลเลอร์ไม่ว่าจะเป็นเบอร์อะไรก็ต้องมีวงจรสร้างสัญญาณนาฬิกา (Oscillator) เพื่อกำหนดจังหวะการทำงานของไมโครคอนโทรลเลอร์วงจรสร้างสัญญาณนาฬิกานั้นจึงต้องการ X-TAL เพื่อสร้างสัญญาณ ซึ่งค่าความถี่ X-TAL ที่เราใช้คือ 11.0592 MHz การที่เราใช้ค่านี้เพราะใน MCS-51 นั้นมีวงจร Serial Port ทำหน้าที่ในการส่งข้อมูลอนุกรม ซึ่งจะต้องมีอัตราการส่งข้อมูลหรือความถี่ในการส่งข้อมูล ให้เป็นไปตามมาตรฐานของ

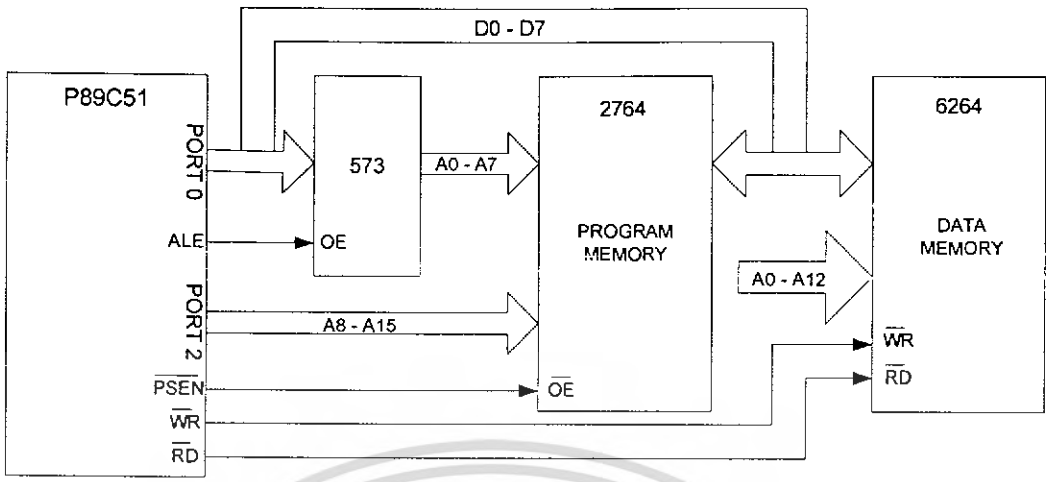
Serial Port ซึ่ง Serial Port ที่เรารู้จักก็มีอยู่ในคอมพิวเตอร์ทั่วไปที่นิยมเรียกว่า COM 1, COM 2 นั่นเอง ความถี่ในการส่งข้อมูลนั้นจะได้จากการหารค่าความถี่จาก X-TAL ซึ่งค่าความถี่จาก X-TAL ค่านี้เมื่อหารความถี่แล้ว จะได้ความถี่ที่ตรงตามความถี่มาตรฐานพอดี ดังนั้นจึงเลือกใช้ค่านี้



ภาพที่ 2.4 แสดงวงจรสร้างสัญญาณนาฬิกา

สำหรับ Capacitor ทั้งสองตัวในวงจรตามภาพที่ 2.4 ทำหน้าที่นำสัญญาณที่เราไม่ต้องการลงกราวด์ไปนั่นเอง โดยธรรมชาติของคลื่นเมื่อเกิดคลื่นที่ความถี่หลัก ก็จะเกิดสัญญาณอื่นๆ อีกหลายความถี่ ซึ่งเป็นจำนวนเท่าของความถี่หลักนั้น จำนวนเท่าของความถี่จะเป็นจำนวนคี่ คือ 3, 5, 7, 9 ไปเรื่อยๆ โดยแต่ละความถี่จะมีระดับของสัญญาณลดลงไปเรื่อยๆ สัญญาณที่เกิดจากความถี่หลักนั้นเราเรียกว่า สัญญาณ Harmonic เพราะฉะนั้น C ที่เราใส่เข้าไปในวงจรก็เพื่อเอาสัญญาณ Harmonic ลงกราวด์นั่นเอง และค่า C ที่เราใช้นั้นจะมีค่าประมาณ 6 -30 pF ซึ่งในที่นี้เราได้เลือกใช้ค่า 22 pF

ขา PSEN (Program Status Enable) (ขา 29) : ขานี้จะทำงานที่ลอจิก "0" (Active low) ขานี้ Active เมื่อ MCS-51 ต้องการติดต่อกับ Program Memory ซึ่ง Program Memory จะเป็นหน่วยความจำที่ทำหน้าที่เก็บคำสั่งที่เราเขียน หน่วยความจำส่วนนี้ส่วนใหญ่จะเป็น EPROM ที่ต่ออยู่ภายนอก หรือ FLASH ROM ที่อยู่ภายในเบอร์ของ AT89Cxx นั่นเอง ดังนั้นขานี้จะต่อเข้ากับขา OE ของ EPROM ในกรณีที่เราไม่ได้ต่อ EPROM ภายนอก เราก็ปล่อยลอยๆ ไว้อย่างนั้น แต่ถ้าเราใช้ไมโครคอนโทรลเลอร์เบอร์ 89C51RD2 ของบริษัท Phillip เราจะต้องต่อขานี้ผ่านสวิทช์ลงกราวด์ เพื่อให้สวิทช์ตัวนี้ทำหน้าที่ให้ 89C51RD2 เข้าโหมดโปรแกรมตัวเอง



ภาพที่ 2.5 แสดงการติดต่อกับหน่วยความจำแต่ละแบบ

ขา ALE (Address Latch Enable) (ขา 30) : เนื่องจาก MCS-51 มีความสามารถในการอ้างหน่วยความจำและอุปกรณ์ภายนอก จึงใช้ Port 2 เป็น Address ไบต์สูง (A8 – A15) และให้ Port 0 เป็น Data Bus และ Address Bus ไบต์ต่ำ เพื่อเป็นการประหยัดพอร์ต ดังนั้นเมื่อ Port 0 มีถึงสองหน้าที่ จึงต้องมีสัญญาณมาบอกให้รู้ว่าตอนนี้ข้อมูลที่อยู่บน Port 0 เป็น Address หรือ Data จึงใช้ ขา ALE เป็นตัวบอกว่าที่ Port 0 เป็น Address ขา ALE จะเป็น "1" ด้วยเหตุนี้ที่ Port 0 จึงมีไอซีที่ทำหน้าที่เก็บข้อมูล (Latch) เช่นเบอร์ 74HC373, 74HC573 ในกรณีที่ต่อหน่วยความจำภายนอก ขา ALE จะต่อเข้ากับขา OE เพื่อส่งสัญญาณให้กับไอซีเอาข้อมูลที่ Port 0 ซึ่งเป็น Address ไบต์ต่ำ (A0 – A7) ออกไปที่เอาท์พุทของไอซีตัวนี้ ดังนั้นเอาท์พุทของไอซี Latch จึงเป็น Address (A0 – A7) นั่นเอง

ขา EA (Enable Access) (ขา 31) : MCS-51 สามารถติดต่อกับหน่วยความจำได้ 3 แบบ คือ

1. หน่วยความจำภายใน (Internal Memory)

2. หน่วยความจำข้อมูลภายนอก (Data Memory) ส่วนที่ทำหน้าที่เก็บข้อมูลจะเป็นหน่วยความจำแบบ RAM

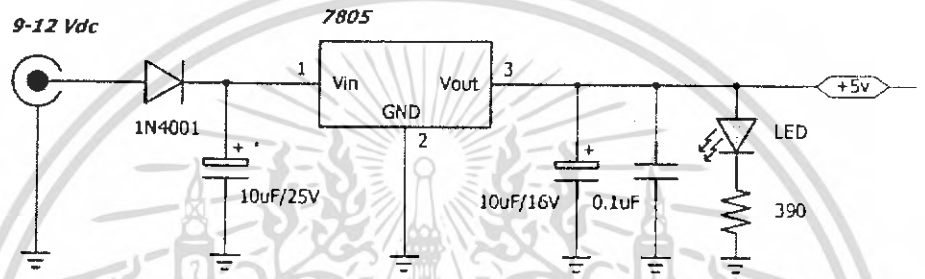
3. หน่วยความจำโปรแกรมภายนอก (Program Memory) สามารถอ้างข้อมูลได้ 64 Kbyte เป็นส่วนที่ทำหน้าที่เก็บคำสั่งที่เราเขียน ส่วนใหญ่ถูกเก็บไว้ที่ EPROM ภายนอก แต่ในปัจจุบันได้มีการพัฒนาจนสามารถนำเอา Program Memory อีกแบบหนึ่งที่เรียกว่า FLASH ROM เข้าไปในไมโครคอนโทรลเลอร์ได้แล้ว FLASH ROM นั้นมีข้อดีกว่า EPROM เพราะสามารถลบได้ด้วยไฟฟ้าและเขียนได้เร็วกว่า ด้วยเหตุนี้จึงต้องมีขา EA เป็นตัวเลือกว่าจะใช้ Program Memory ภายนอกหรือภายใน ซึ่งในการทำปริญญานิพนธ์นี้ได้เลือกใช้ไมโครคอนโทรลเลอร์เบอร์

P89C51RD2 ซึ่งมี Program Memory ภายในที่เป็น FLASH ROM เราจึงให้ขา EA ต่อ Vcc แต่ถ้าเราใช้ Program Memory ภายนอกขา EA จะต่อลงกราวด์

2.1.1.2 วงจรที่ใช้ร่วมกับไมโครคอนโทรลเลอร์

วงจร Regulator

เป็นวงจรที่ทำหน้าที่จำกัดแรงดันให้เหลือเพียง 5V เพื่อเป็นแหล่งจ่ายให้กับวงจรไมโคร-คอนโทรลเลอร์ ซึ่งตัวที่ทำหน้าที่นี้คือไอซีเบอร์ 7805

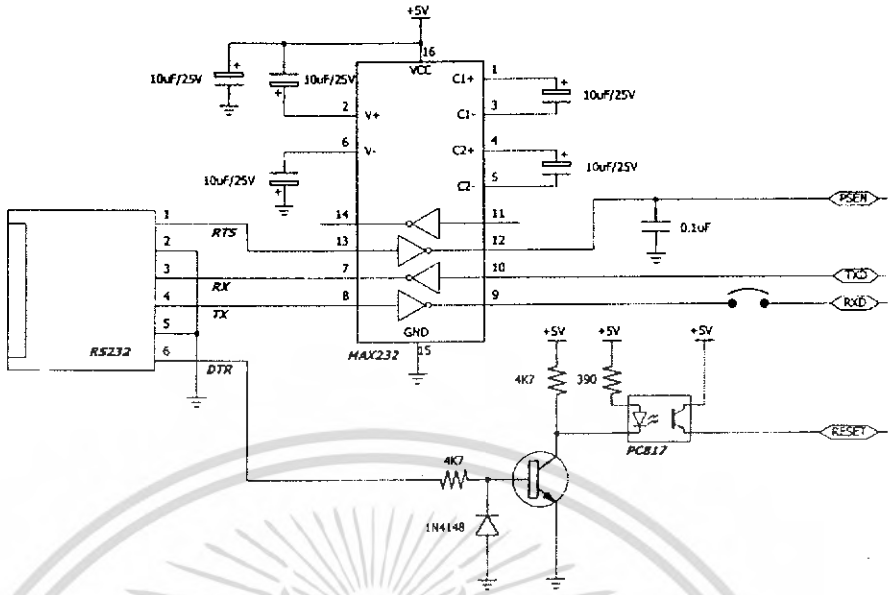


ภาพที่ 2.6 แสดงวงจร Regulator

วงจร Serial Port และ วงจรในการโปรแกรม

เป็นวงจรที่ใช้ในการติดต่อกับพอร์ตอนุกรมตามมาตรฐาน RS232 โดยใช้ไอซีเบอร์ MAX 232 ทำหน้าที่แปลงระดับแรงดันลอจิก ให้เป็นระดับแรงดันตามมาตรฐาน

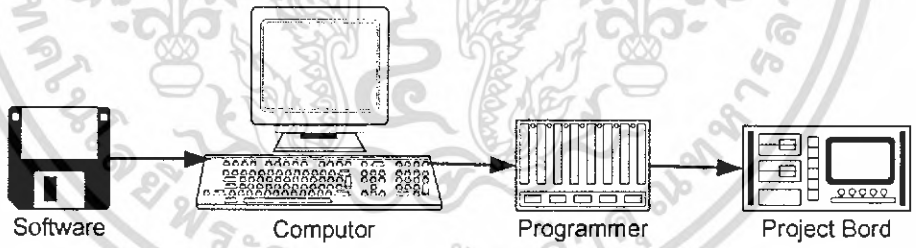
อีกส่วนหนึ่งของวงจรคือ วงจรในการโปรแกรมลงบน P89C51RD2 โดยเราจะใช้สัญญาณ RTS, DTR ของคอมพิวเตอร์มาควบคุมสัญญาณที่ขา RESET และ PSEN ให้ P89C51RD2 เข้าสู่โหมดโปรแกรม



ภาพที่ 2.7 แสดงวงจร Serial Port และ วงจรในการโปรแกรม

2.1.2 เครื่องมือในการพัฒนาไมโครคอนโทรลเลอร์

ในหัวข้อนี้เราจะมาทำความรู้จักและศึกษา วิธีการใช้งานเครื่องมือในการสร้างชิ้นงาน ก่อนที่เราจะเริ่มสร้างชิ้นงานจากไมโครคอนโทรลเลอร์ซึ่งมีองค์ประกอบที่สำคัญดังภาพที่ 2.8



ภาพที่ 2.8 แสดงเครื่องมือในการพัฒนาโปรแกรม

2.1.2.1 Software (โปรแกรม)

คือโปรแกรมที่เราใช้ในการพัฒนา ประกอบด้วย 3 โปรแกรมดังนี้

- Editor ใช้ในการเขียนโปรแกรมของเรา ซึ่งจะเป็น Editor ของใครก็ได้แล้วแต่ว่าเราจะถนัด Editor ตัวไหน เมื่อเราเขียนเสร็จแล้วให้บันทึกไฟล์โดยที่นามสกุลต้องเป็น .C

- Compiler คือโปรแกรมในการแปลง Text File ที่เราเขียนขึ้นโดยใช้โปรแกรม Editor มาให้อยู่ในรูปของ Hex File ซึ่งในไฟล์นี้ก็จะเป็นภาษาเครื่องหรือ Opcode ที่ MCS-51 เข้าใจซึ่งในการทำปริญญานิพนธ์นี้ได้เลือกใช้โปรแกรม KEIL C51 เป็นโปรแกรมสำหรับ Editor, Compiler

- Programmer คือโปรแกรมที่มาพร้อมกับเครื่อง ซึ่งโปรแกรมนี้จะนำ Hex File ที่ได้จากการ Compiler แล้วส่งไปให้เครื่อง Programmer เพื่อทำการโปรแกรมลงบนตัวไมโครคอนโทรลเลอร์ แต่ในที่นี้เราใช้ไมโครคอนโทรลเลอร์เบอร์ P89C51RD2 ซึ่งมีความสามารถโปรแกรมตัวเองด้วยไฟฟ้าผ่านพอร์ตอนุกรม จึงไม่จำเป็นต้องใช้เครื่อง Programmer แต่อย่างใด

1. โปรแกรมFlash Magic

เป็นโปรแกรมที่ทำหน้าที่นำ Hex File ไปโปรแกรมลงบนบอร์ดทดลองของเรา ซึ่งบอร์ดทดลองที่ได้ออกแบบแล้วนั้นใช้ตัว MCU เบอร์ P89C51RD2 ของบริษัท Phillip ซึ่งมีความสามารถในการโปรแกรมตัวเองผ่านพอร์ตอนุกรม ซึ่งเป็นโปรแกรมที่นิยมใช้กันอยู่ในปัจจุบัน

ขั้นตอนในการใช้งาน Flash Magic

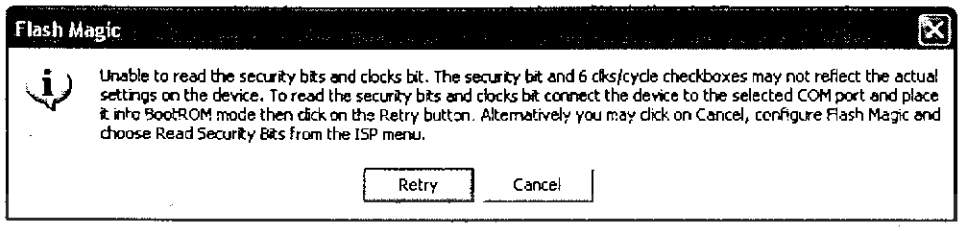
1. ทำการต่อสายสัญญาณเข้าที่ COM 1 ของคอมพิวเตอร์ อีกด้านหนึ่งต่อเข้ากับบอร์ดทดลองตรงคอนเน็คเตอร์ RS232

2. จ่ายไฟเลี้ยงให้กับบอร์ดทดลอง

3. RUN โปรแกรม Flash Magic

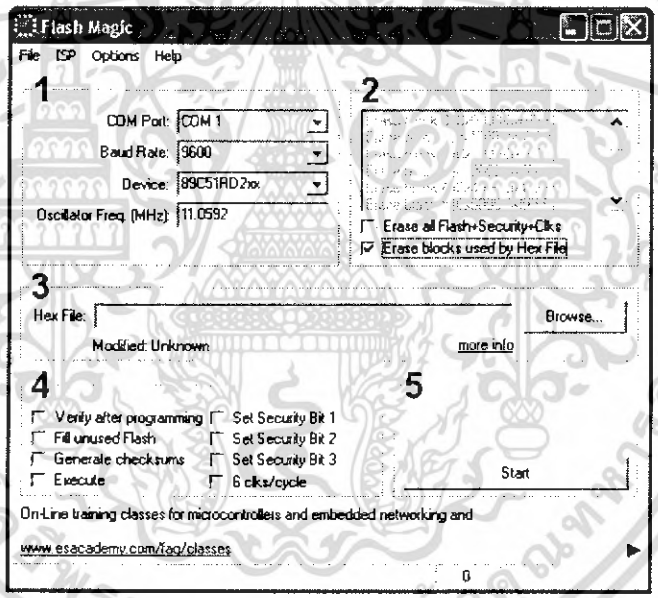
ภาพที่ 2.9 แสดงการ RUN โปรแกรม Flash Magic

4. ถ้าโปรแกรมแสดงภาพที่ 2.10 นั่นแสดงว่าโปรแกรมไม่สามารถติดต่อกับบอร์ดของเราได้ให้กดปุ่ม Cancel



ภาพที่ 2.10 แสดงว่าโปรแกรมไม่สามารถติดต่อกับบอร์ดได้

5. หลังจากนั้นโปรแกรมจะแสดงดังภาพที่ 2.11 ให้เซตค่าต่างๆ ตามภาพ



ภาพที่ 2.11 แสดงการตั้งค่าต่างๆ ของโปรแกรม Flash Magic

COM Port : เป็นพอร์ตที่ใช้ในการติดต่อกับบอร์ดของเรา

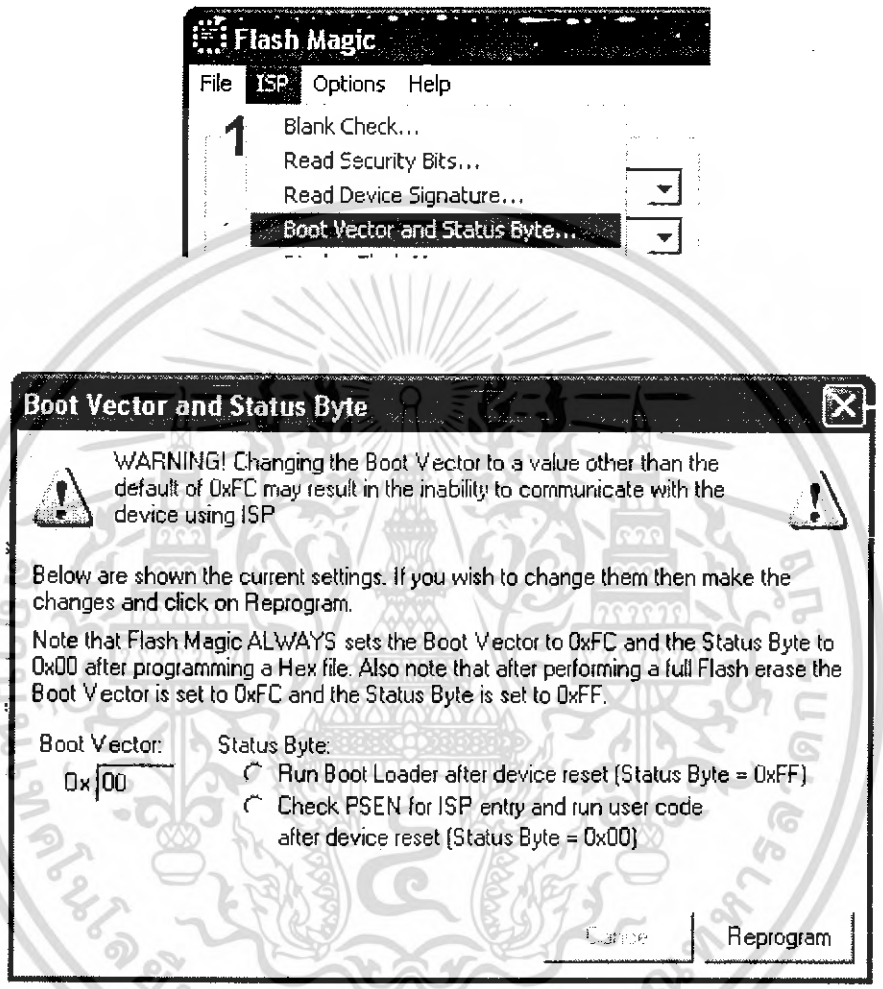
Baud Rate : ความเร็วในการส่งข้อมูลระหว่างคอมพิวเตอร์กับบอร์ด

Device : ชื่อของเบอร์ไมโครคอนโทรลเลอร์ที่เราเลือกใช้

Oscillator Freq. (MHz) : คือค่าความถี่ X-TAL บนบอร์ดทดลองของเรา

คลิกที่ Check Box : Erase block used by Hex File (เพื่อลบ Flash Rom ในตัว MCU)

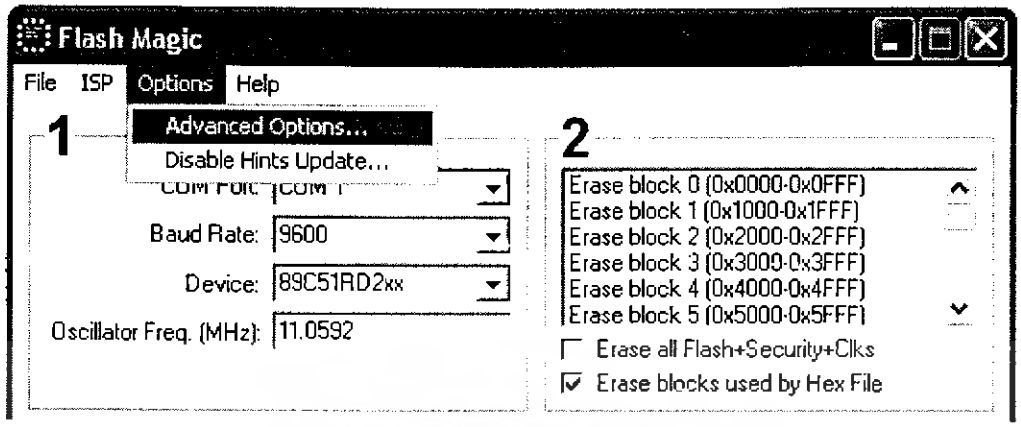
6. ในกรณีที่ซื้อไมโครคอนโทรลเลอร์ตัวใหม่มา ค่า Status Byte จะมีค่าเป็น FF ให้เราเปลี่ยนเป็นค่า 00 โดยเข้าไปที่เมนู ISP/ Boot Vector and Status Byte ตามภาพที่ 2.12



ภาพที่ 2.12 แสดงการตั้งค่า Status Byte

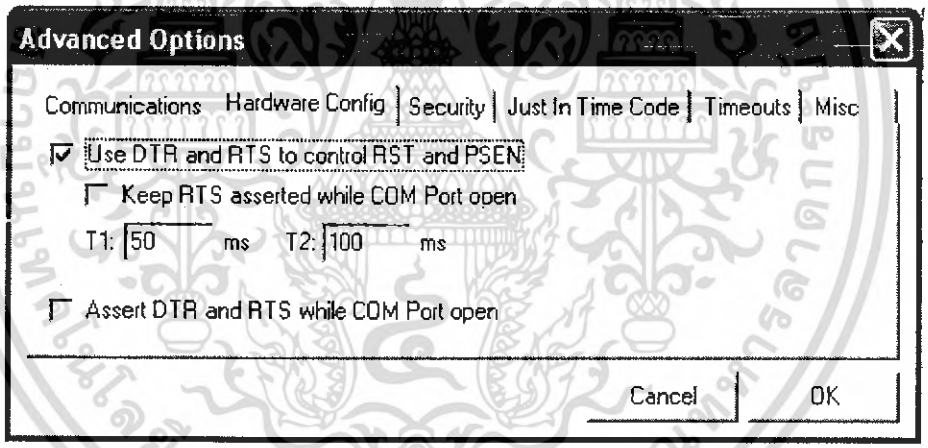
7. กดปุ่ม Reprogram เพื่อทำการโปรแกรม Status Byte

8. ทำการเซตสัญญาณ RTS , DTR ของพอร์ตอนุกรม ให้เป็นสัญญาณควบคุมโปรแกรมของบอร์ด นั่นหมายความว่าตัวบอร์ดต้องมีอุปกรณ์ที่สนับสนุนสัญญาณควบคุมนี้ด้วย ให้เราเข้าไปที่เมนู Option / Advance Option



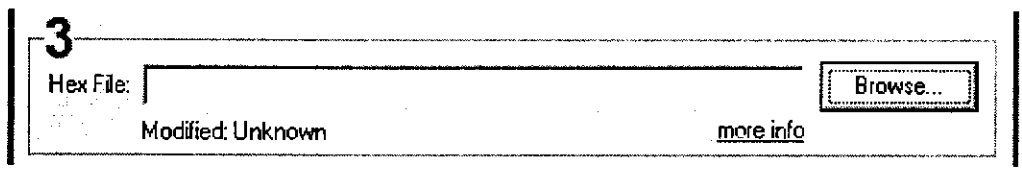
ภาพที่ 2.13 แสดงการเซตสัญญาณ RTS, DTR ของพอร์ตอนุกรม ให้เป็นสัญญาณควบคุมโปรแกรมของบอร์ด

9. โปรแกรมจะแสดงภาพที่ 2.14 ให้เราทำการเซตการใช้สัญญาณ DTR, RTS ตามนี้



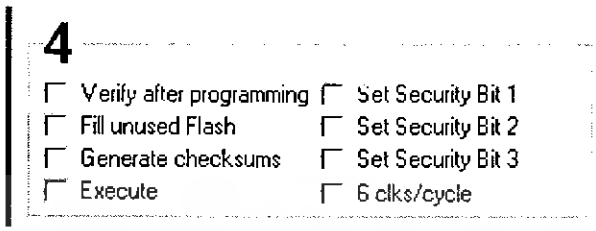
ภาพที่ 2.14 แสดง Advanced Option

10. หา Hex File ที่เราต้องการจะโปรแกรมลงบนไมโครคอนโทรลเลอร์ โดยการกดปุ่ม Browse



ภาพที่ 2.15 แสดง Hex File

11. กำหนดรูปแบบการโปรแกรมลงบนไมโครคอนโทรลเลอร์



ภาพที่ 2.16 แสดงรูปแบบการโปรแกรมลงบนไมโครคอนโทรลเลอร์

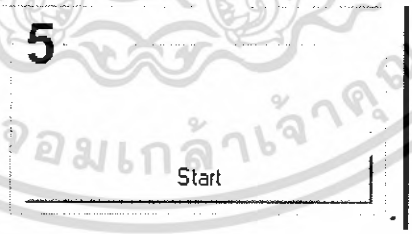
Verify after programming : หลังจากโปรแกรมแล้วให้ทำการตรวจสอบข้อมูลในไมโครคอนโทรลเลอร์กับ Hex File

Fill unused Flash : ใส่ค่า 00H ลงในส่วนที่ไม่ได้ใช้

Generate Checksum : สร้างค่า Checksum ในโปรแกรม

Set Security Bit 1- 3 : เป็นโปรแกรมการ Copy ข้อมูลทั้ง 3 ระดับ

12. กดปุ่ม Start เพื่อทำการโปรแกรมไฟล์ลงบนไมโครคอนโทรลเลอร์ในบอร์ดของเรา



ภาพที่ 2.17 แสดงการโปรแกรมไฟล์ลงบนไมโครคอนโทรลเลอร์ในบอร์ด

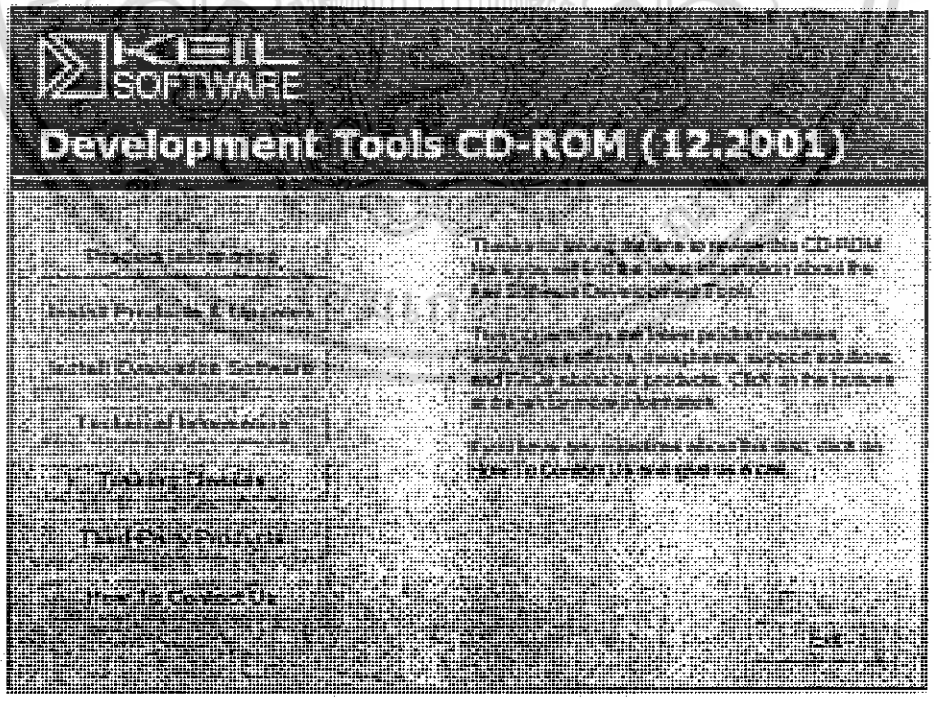
2. โปรแกรม Keil C51

โปรแกรม Keil C51 เป็นโปรแกรมที่ใช้ในการเขียนโปรแกรมภาษา C สำหรับไมโครคอนโทรลเลอร์ตระกูลต่างๆ ของแต่ละบริษัท ซึ่งสามารถทำงานบนระบบปฏิบัติการ Windows 95 ขึ้นไปได้

ภาพที่ 2.18 แสดงโปรแกรม Keil C51

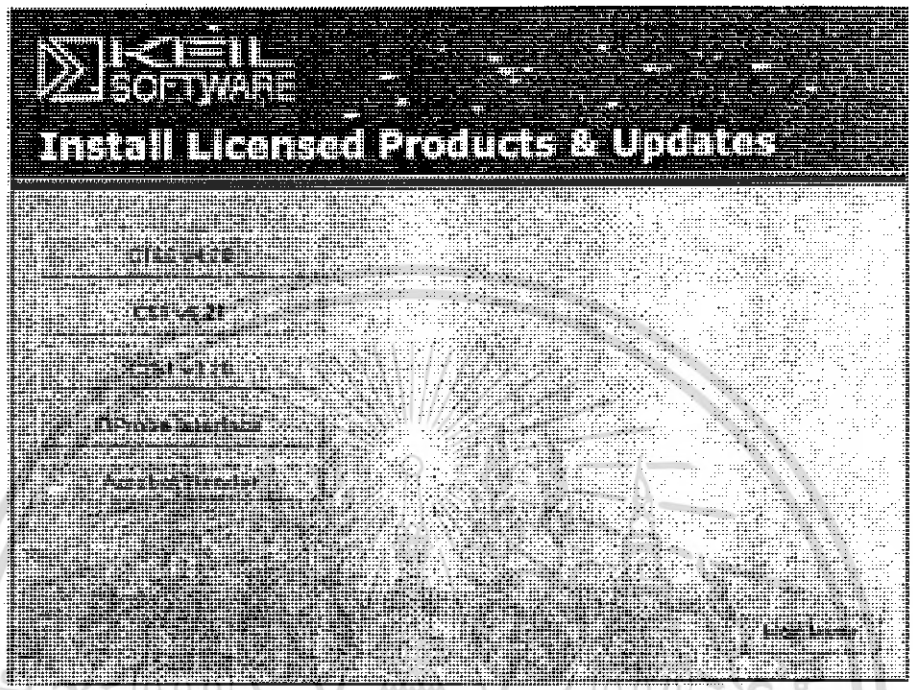
ขั้นตอนในการติดตั้งและใช้งานโปรแกรม Keil C51

1. นำแผ่น CD โปรแกรม Keil C51 ใส่เข้าไปใน CD-ROM Drive โปรแกรมจะทำงานอัตโนมัติ (Auto Run) หน้าจอจะแสดงดังภาพที่ 2.19



ภาพที่ 2.19 แสดงDevelopment Tools CD-ROM (12.2001)

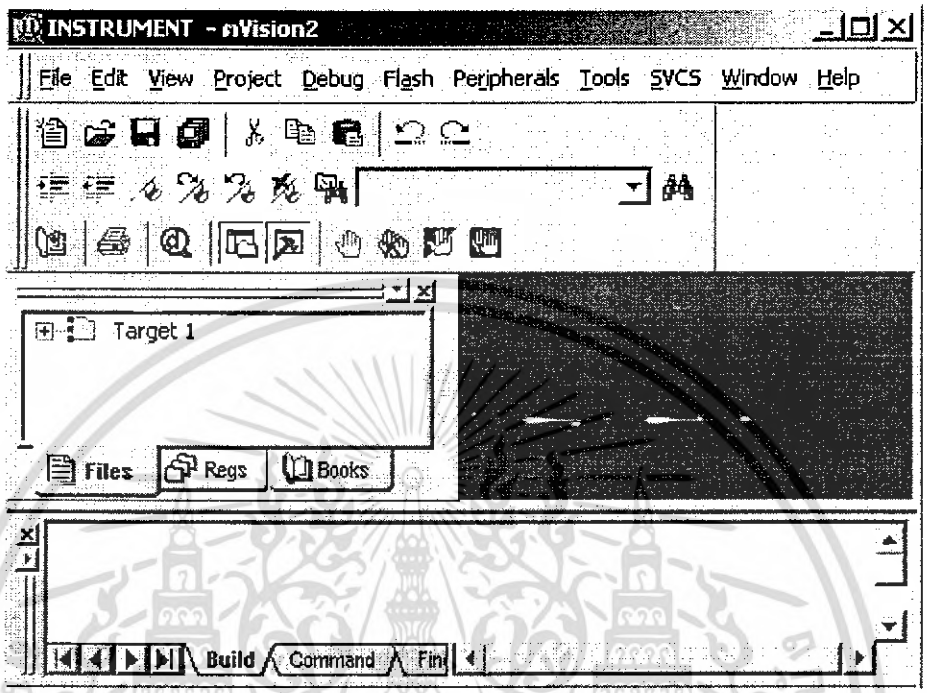
2. เลือกเมนู Install Products & Updates หน้าจอจะแสดงดังภาพที่ 2.20



ภาพที่ 2.20 แสดง Install Products & Updates

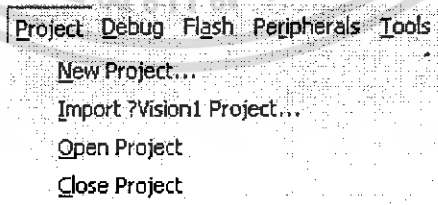
3. เลือกเมนู C51 v6.21 จะทำการติดตั้งโปรแกรม ให้เราทำตามโปรแกรมโดยการกดปุ่ม Next ไปเรื่อยๆ จนเสร็จสิ้นการติดตั้ง

4. ทำการ Run โปรแกรม จะปรากฏหน้าจอดังภาพที่ 2.21



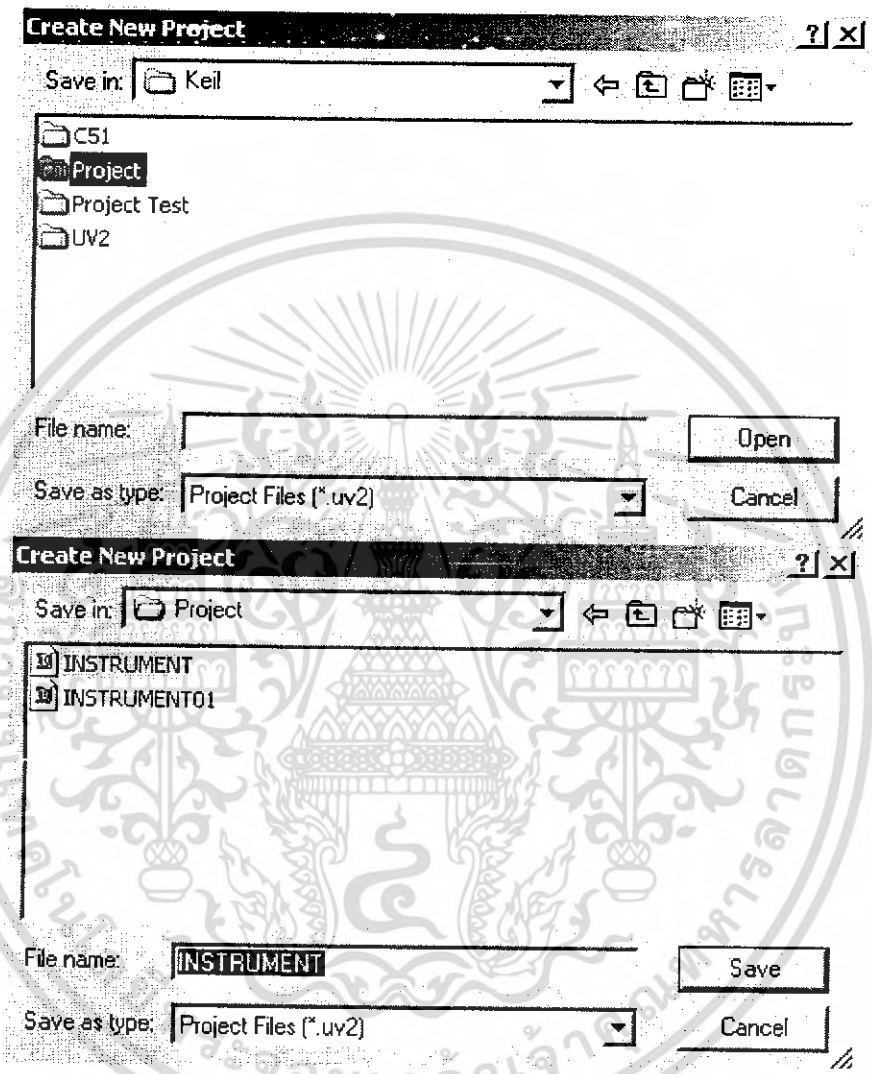
ภาพที่ 2.21 แสดงการ Run โปรแกรม

5. สร้าง Project ใหม่ โดยเข้าที่เมนู Project / New Project ดังภาพที่ 2.22



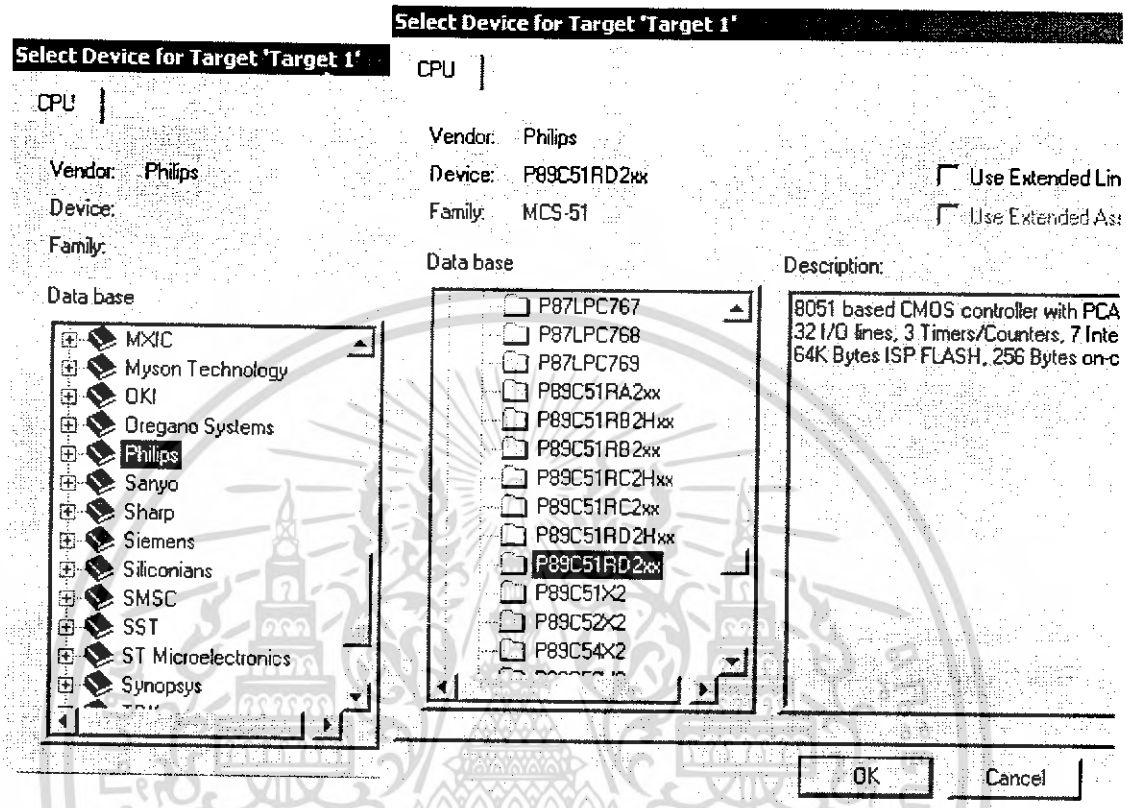
ภาพที่ 2.22 แสดงการสร้าง Project ใหม่

6. หน้าจอจะแสดงดังภาพข้างล่าง ให้เราทำการสร้างชื่อ Folder ใหม่ใน C:\Keil โดยให้ตั้งชื่อว่า Project จากนั้นให้ตั้งชื่อไฟล์ว่า INSTRUMENT แล้วคลิกปุ่ม Save ดังภาพที่ 2.23



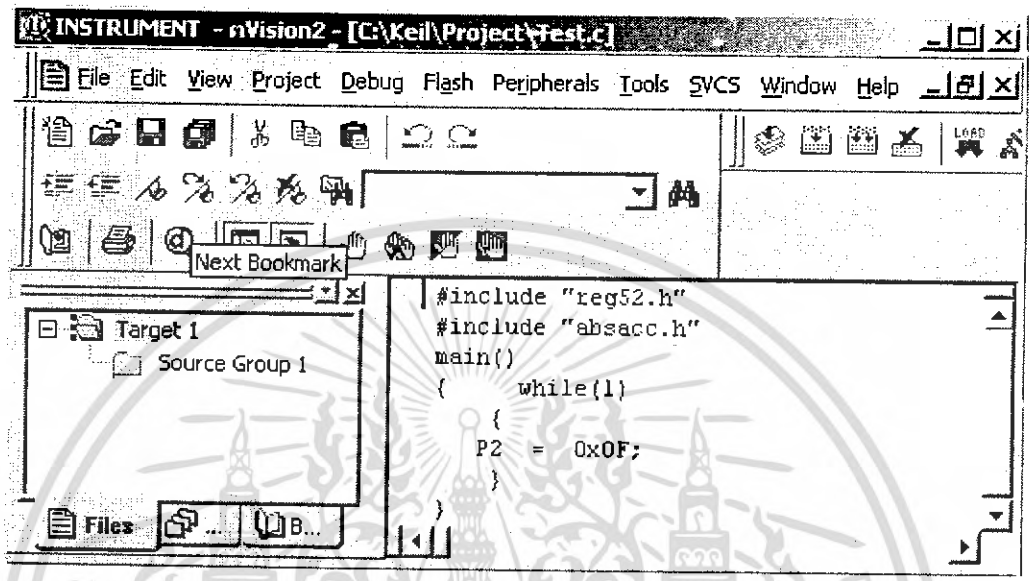
ภาพที่ 2.23 แสดงการสร้างชื่อ Folder ใหม่

7. จากนั้นให้เลือกบริษัทและเบอร์ไมโครคอนโทรลเลอร์ที่เราใช้



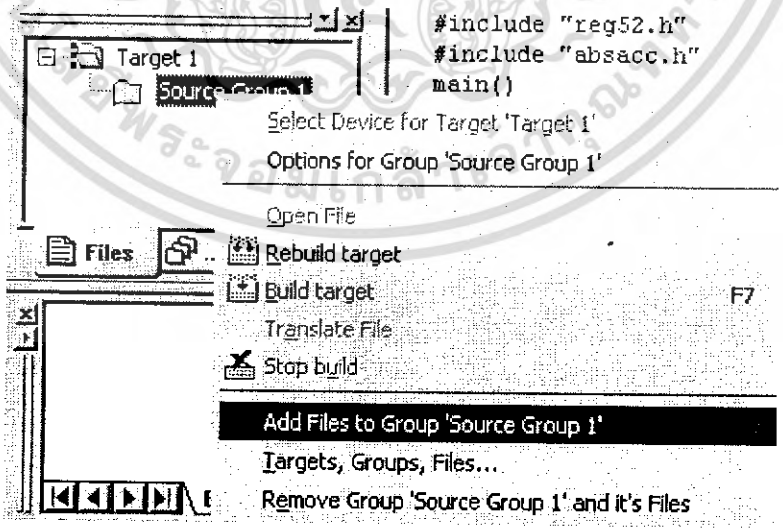
ภาพที่ 2.24 แสดงการเลือกบริษัทและเบอร์ไมโครคอนโทรลเลอร์ที่ใช้

8. สร้างไฟล์โปรแกรมใหม่โดยเลือกเมนูที่ File / New จากนั้นจะมีหน้าต่างสำหรับเขียนโปรแกรม ก็ให้เราลองเขียนโปรแกรมภาษา C ง่ายๆ สักโปรแกรมดังภาพที่ 2.25 แล้วทำการบันทึกไฟล์โดยไปที่เมนู File / Save แล้วตั้งชื่อไฟล์ว่า Test.c



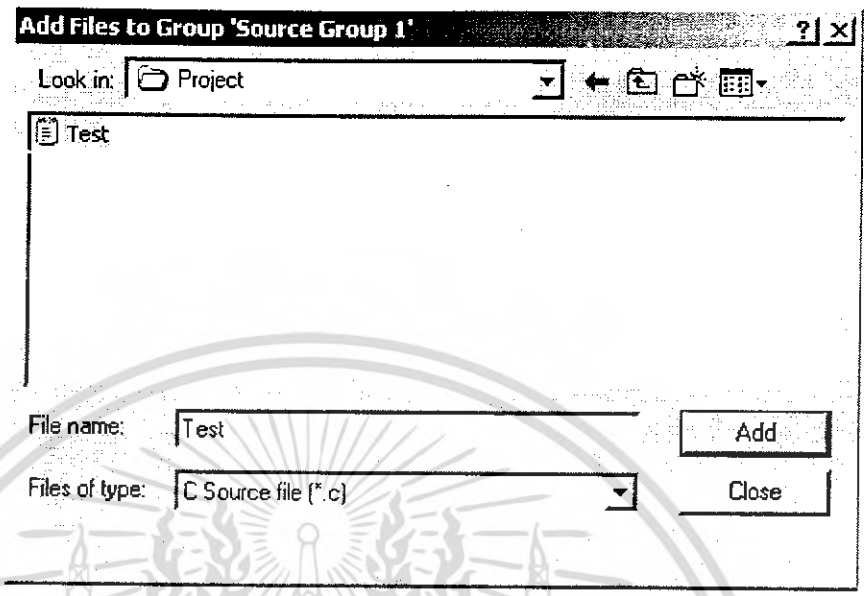
ภาพที่ 2.25 แสดงการสร้างและบันทึกไฟล์โปรแกรมใหม่

9. ทำการเพิ่มไฟล์ เข้าไปใน Project ของเราโดยการเลื่อนเมาส์ไปที่ Source Group1 แล้วคลิกขวาจากนั้นเลือก Add files to Group 'Source Group1' ดังภาพที่ 2.26



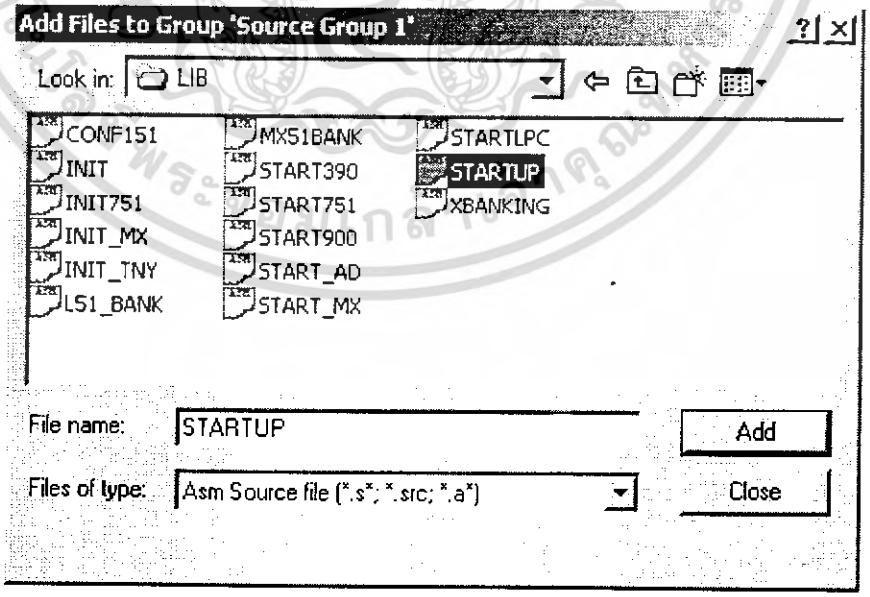
ภาพที่ 2.26 แสดง Add files to Group 'Source Group1'

10. เลือกไฟล์ชื่อ Test.c ที่เราได้สร้างไว้ แล้วกดปุ่ม Add เพียง 1 ครั้ง ข้อสังเกตก็คือหน้าต่างจะไม่ปิด



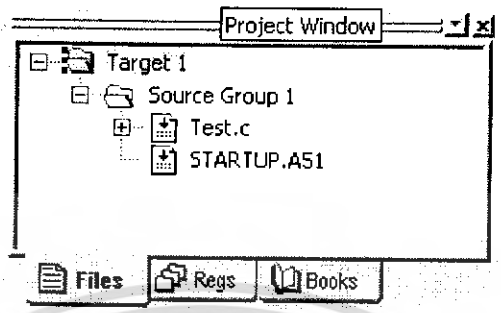
ภาพที่ 2.27 แสดงเลือกไฟล์ชื่อ Test.c ที่สร้างไว้

11. จากนั้นทำการ Add ไฟล์ต่อไป โดยให้เราเข้าไปที่ C:\Keil\C51\Lib แล้วให้เราเลือก File of Type เป็น " Asm Source file" จากนั้นเลือกไฟล์ชื่อ "STARTUP.A51" แล้วกดปุ่ม Add หนึ่งครั้ง แล้วกดปุ่ม Close



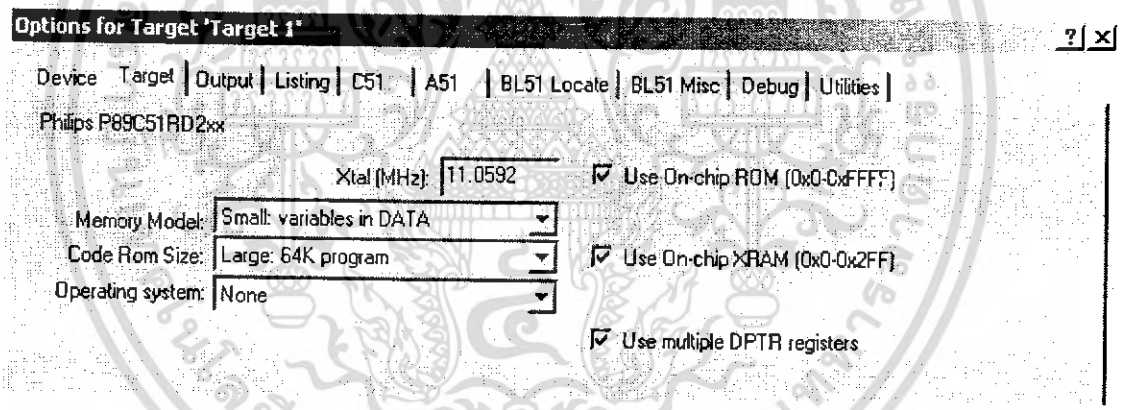
ภาพที่ 2.28 แสดงการเลือกไฟล์ชื่อ "STARTUP.A51"

12. จากนั้นหน้าต่างจะปิดให้เรากดปุ่ม + หน้า Group Source1 เราจะเห็นไฟล์ทั้งสองไฟล์ที่เราทำการ Add เข้าไป



ภาพที่ 2.29 แสดงไฟล์ทั้งสองไฟล์ที่เราทำการ Add เข้าไป

13. ทำการเซต Option ของ Project โดยให้ไปที่เมนู Project/Option for Target 'Target1' หน้าจอจะแสดงภาพที่ 2.30

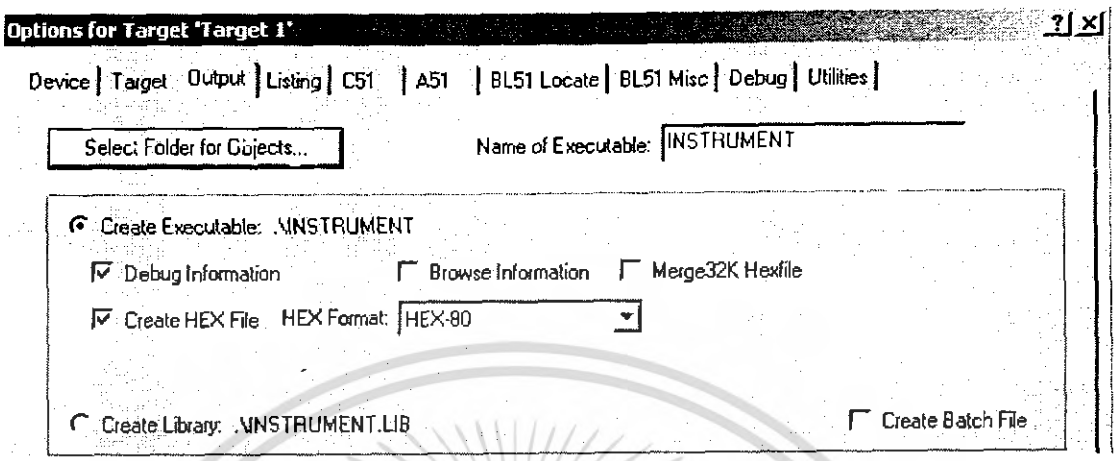


ภาพที่ 2.30 แสดงการเซต Option ของ Project

ในส่วนของ Target เราสามารถเซตค่าต่างๆ ได้ดังนี้

- ให้เราป้อนความถี่ X-TAL ของบอร์ดที่เราได้ออกแบบไว้เป็นหน่วย MHz
- เลือก Memory Model เป็น Small
- เลือก Code Rom Size เป็น Large
- Operating System เป็น None
- คลิกเลือก Use On-chip Rom , Use multiple DPTR , Use On-Chip XRAM ตามภาพอันขึ้นอยู่กับบอร์ดที่เราเลือกใช้

14. เข้ามาในส่วนของ Output ให้เราเลือก Create HEX File ดังภาพที่ 2.31

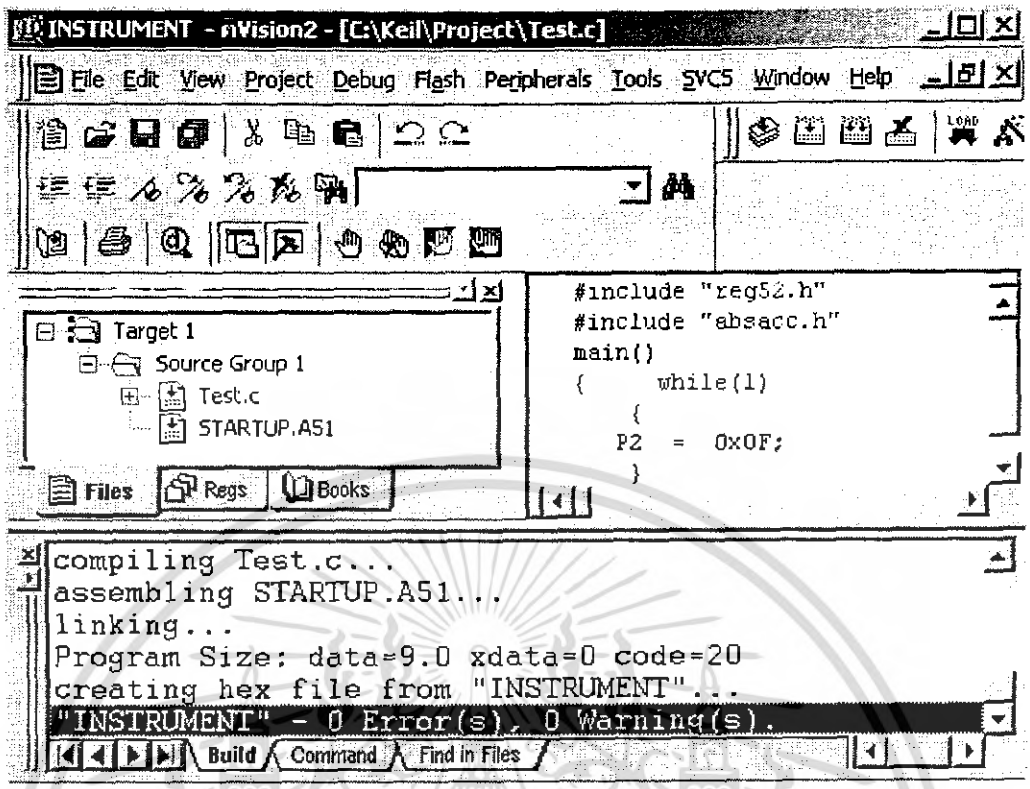


ภาพที่ 2.31 แสดงการเลือก Create HEX File

ขั้นตอนการ Compile

ตอนนี้เราก็พร้อมที่จะเขียนโปรแกรมแล้ว ในหน้าต่างของโปรแกรม หลังจากที่เราทำการเขียนโปรแกรมแล้วเราก็จะต้องทำการ Compile โปรแกรมที่เราเขียนขึ้นมา ว่าถูกต้องตามรูปแบบของคำสั่งหรือไม่โดยทำตามขั้นตอนดังนี้ไปที่เมนู Project/Bluid Target หรือกดคีย์ F7

ผลที่ได้จากการ Compile จะแสดงที่หน้าต่าง Output โปรแกรมก็จะบอกว่าคำสั่งที่เราเขียนถูกต้องหรือไม่และถ้าไม่ถูกต้องโปรแกรมก็จะบอกตำแหน่งของคำสั่งที่ผิดพลาด ผ่านทางหน้าต่าง Output ถ้าผิดพลาดก็ให้เราทำการแก้ไขให้ถูกต้อง หลังจากที่ Compile ผ่านแล้วก็จะได้ไฟล์นามสกุล HEX ใน Folder ของโปรแกรม ก็สามารถเอา HEX ไฟล์ไปโปรแกรมลงบนไมโครคอนโทรลเลอร์ของเราอีกที

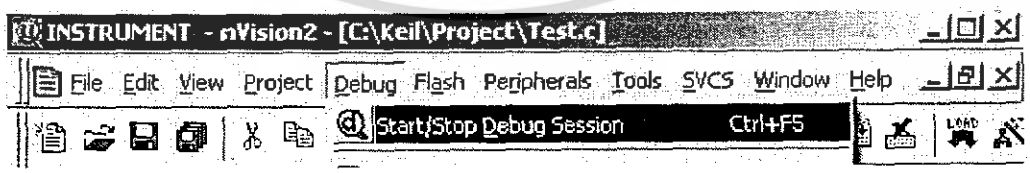


ภาพที่ 2.32 แสดงการ Compile โปรแกรม

ขั้นตอนการ Debug หรือ Simulate

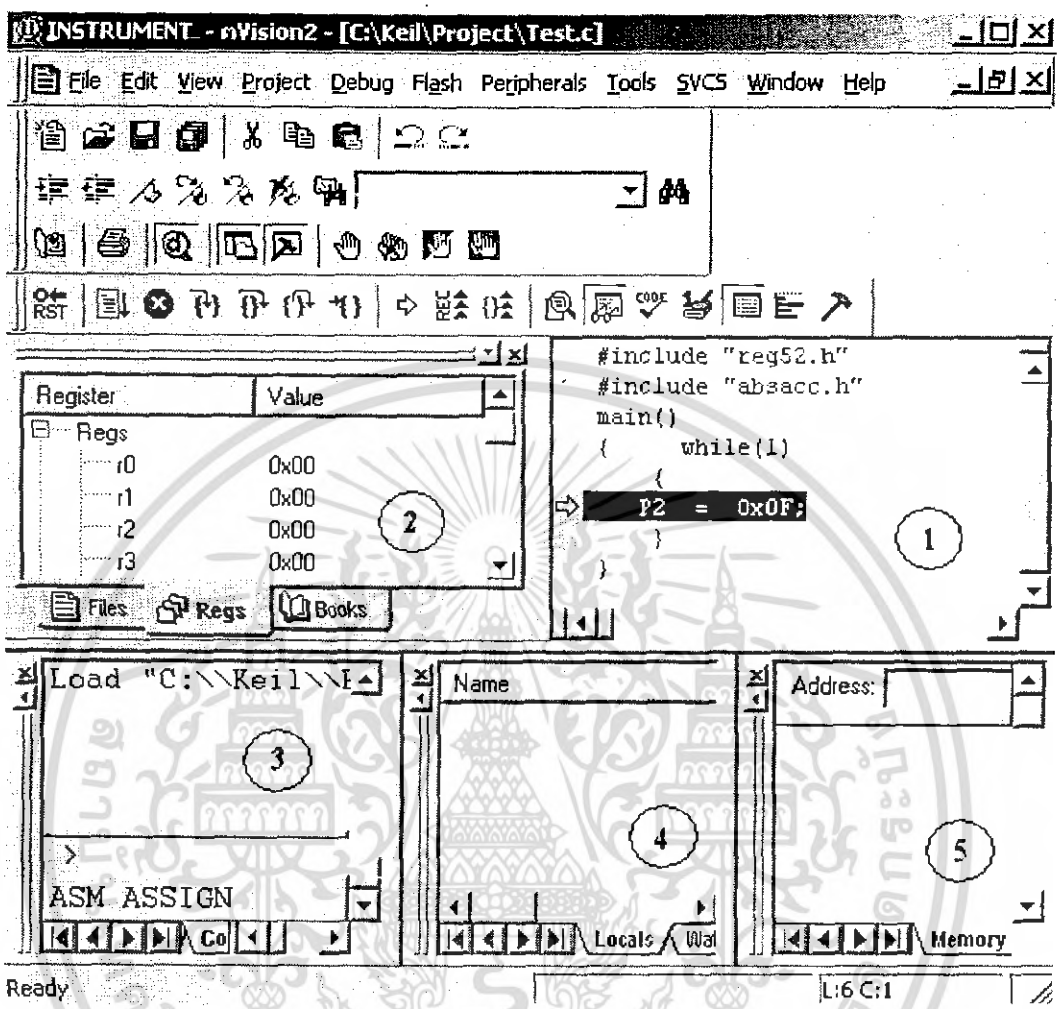
หลังจาก Compile ผ่านแล้ว เราสามารถที่จะตรวจสอบการทำงานของโปรแกรมที่เราเขียนขึ้นมาว่าเป็นไปตามต้องการหรือไม่ โดยทำตามขั้นตอนดังนี้

1. ไปที่เมนู Debug/Start/Stop Debug Session หรือกดคีย์ Ctrl+F5 เพื่อเริ่มต้นในการ Debug



ภาพที่ 2.33 แสดงการเริ่มต้นในการ Debug

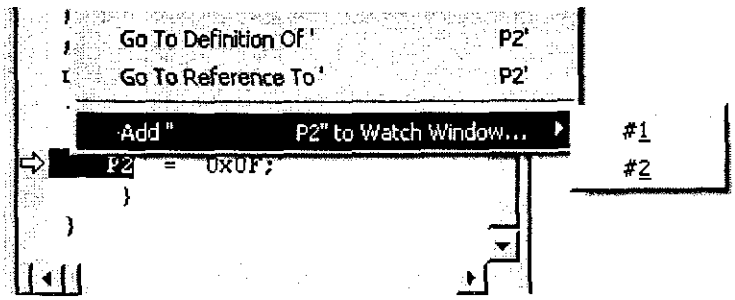
2. โปรแกรมจะเริ่มการ Debug โดยจะมีหน้าต่างหลายๆ ตัวขึ้นมาดังภาพที่ 2.34



ภาพที่ 2.34 แสดงหน้าต่างหลังจากทำการ Debug ไปแล้ว

รายละเอียดต่างๆ ของหน้าต่าง ตามภาพที่ 2.34

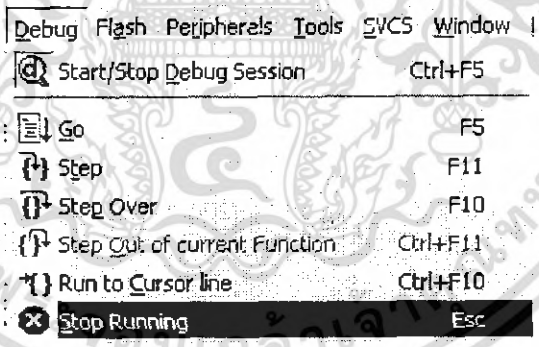
1. Work Space Window เป็นหน้าต่างที่ใช้สำหรับการเขียนโปรแกรม
2. Project Window เป็นหน้าต่างที่แสดงโครงสร้างของไฟล์และค่าของรีจีสเตอร์ต่างๆ
3. Output Window เป็นหน้าต่างทีแสดงผลของการทำงานของโปรแกรม
4. Watch / Call Stack เป็นหน้าต่างที่แสดงค่าของตัวแปรในโปรแกรม สามารถแสดงค่าของตัวแปรโดยใช้เมาส์ High Light บนตัวแปรนั้นแล้วคลิกปุ่มขวา แล้วเลือก Add to Watch Window ใช้ได้ขณะที่กำลังทำ Debug
5. Memory Window เป็นหน้าต่างที่แสดงค่าในหน่วยความจำ



ภาพที่ 2.35 แสดง Memory Window

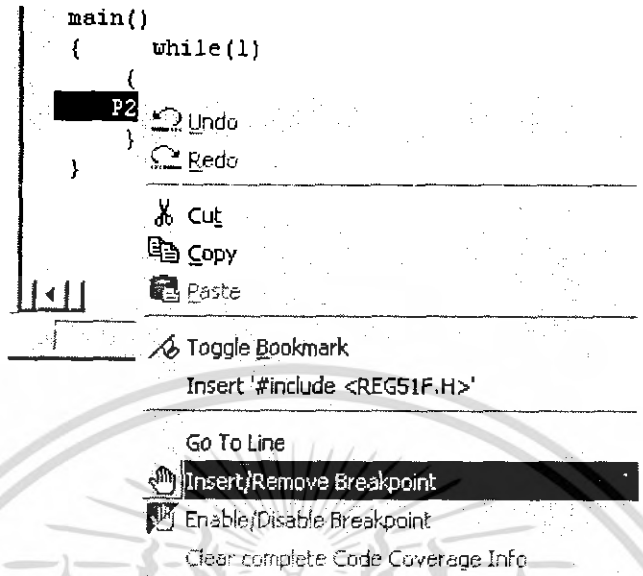
ฟังก์ชั่นต่างๆ ในการ Debug Program

- Go (F5) Run โปรแกรมแบบต่อเนื่อง
- Stop (Esc) หยุดการ Run โปรแกรม
- Step (F11) Run ที่ละครั้ง
- Step Over (F10) Run ที่ละคำสั่งและโปรแกรมย่อย
- Run to Cursor (Ctrl+F10) Run ไปจนถึงตำแหน่ง Cursor



ภาพที่ 2.36 แสดงฟังก์ชั่นต่างๆ ในการ Debug Program

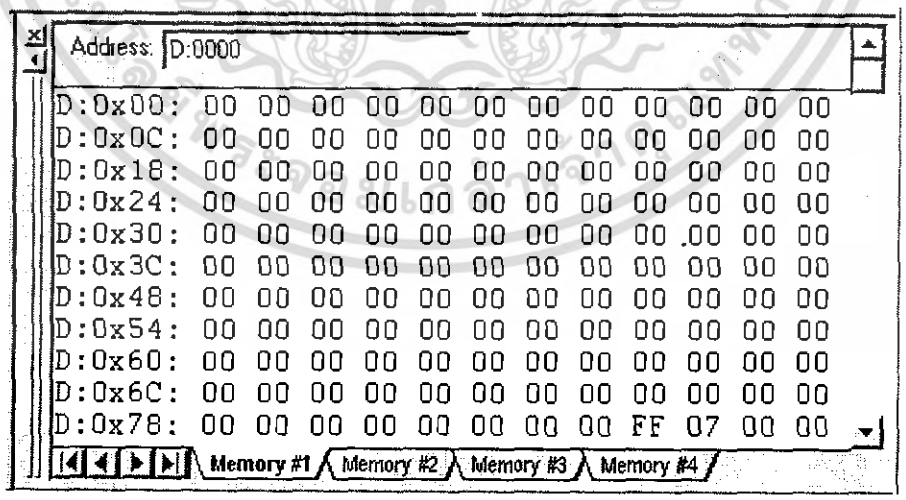
การกำหนดตำแหน่งหยุดในการ Run โปรแกรม (Break Point)



ภาพที่ 2.37 แสดงตำแหน่งหยุดในการ Run โปรแกรม (Break Point)

สามารถกำหนดตำแหน่งหยุดการ Run โปรแกรม ที่คำสั่งที่ต้องการได้โดยการเลื่อนเมาส์ไปคลิกที่หน้าคำสั่งนั้นให้ Cursor อยู่ในบรรทัดนั้น จากนั้นคลิกขวาเลือกหัวข้อ "Insert/Remove Breakpoint" ตามภาพที่ 2.37

ในขณะที่กำลังทำการ Debug โปรแกรม สามารถดูค่าตัวแปรต่างๆ โดยวิธีการดังนี้



ภาพที่ 2.38 แสดงการ Memory Window

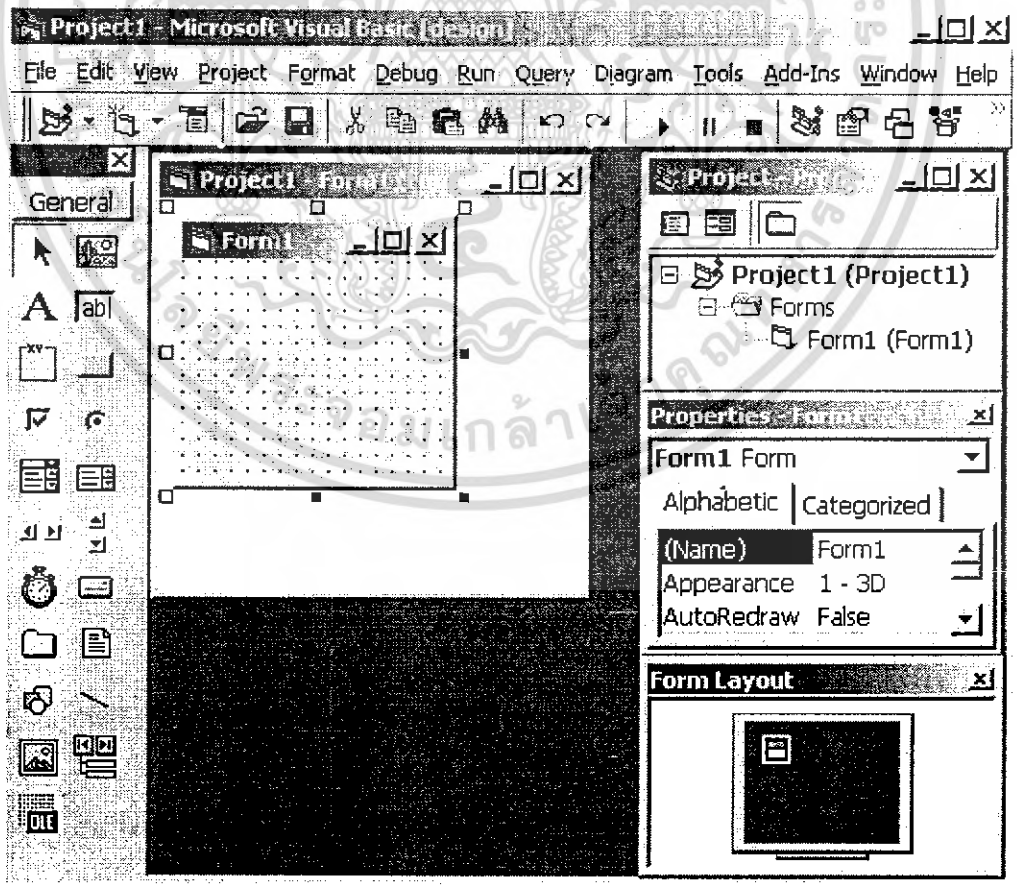
กำหนดชนิดของหน่วยความจำ และ Address เริ่มต้นที่ Memory Window ในช่อง Address ซึ่งมีรูปแบบการป้อนดังนี้ สามารถกำหนดค่าเหล่านี้ได้ 4 ช่อง

D : xx	หน่วยความจำภายใน 00H-7FH+SFR
I : xx	หน่วยความจำภายใน 00H FFH
X : xxxx	หน่วยความจำข้อมูลภายนอก 0000H-FFFFH
C : xxxx	หน่วยความจำโปรแกรมภายนอก 0000H-FFFFH
xx	คือ Address เริ่มต้น

3. โปรแกรม Visual Basic

Microsoft Visual Basic ถือว่าเป็นเครื่องมือพัฒนาแอพพลิเคชั่นที่ถูกพัฒนามาอย่าต่อเนื่อง เพื่อเป็นเครื่องมือพัฒนาแอพพลิเคชั่นที่สามารถทำงานได้มากมายและง่ายในการใช้งาน

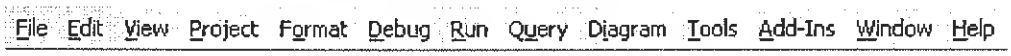
ลักษณะการทำงานของ Visual Basic จะทำงานในลักษณะ IDE (Integrated Development Environment) คือ รวบรวมเครื่องมือ , ข้อมูลที่ใช้งานต่างๆ ไว้ในหน้าจอเดียว ทำให้เรียกใช้งานได้ง่าย



ภาพที่ 2.39 แสดงส่วนประกอบต่างๆ ของโปรแกรม Visual Basic

ขั้นตอนในการใช้งาน

1. Menu Bar เป็นที่รับคำสั่งในแบบเมนู เมื่อเราทำการสร้างแอพพลิเคชั่นด้วย Visual Basic เป็นเหมือนศูนย์กลางที่ควบคุมการสร้างแอพพลิเคชั่น

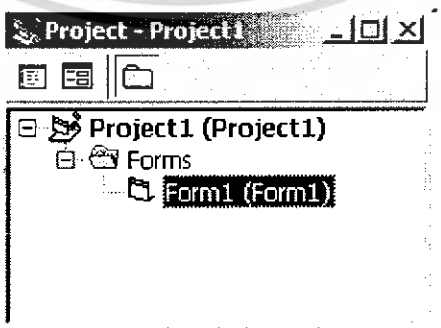


ภาพที่ 2.40 แสดง Menu Bar

2. Tool Bar ในการใช้งานในเมนูบาร์สั่งงานอาจจะมีขั้นตอนที่ยุ่งยากเพื่อขั้นตอนลง จะทำให้การคลิกที่ Tool Bar เพียงครั้งเดียวสามารถสั่งงานที่เราต้องการได้ (คีย์ลัด)

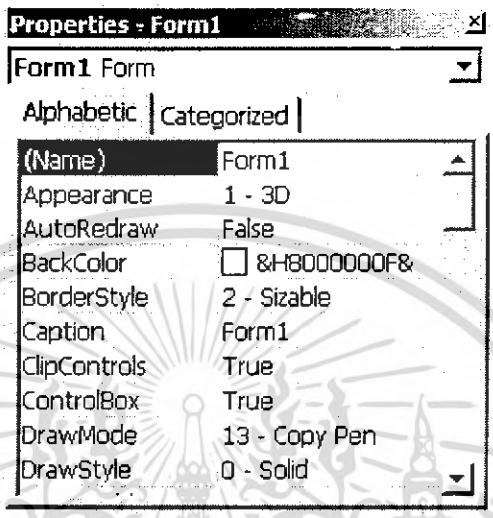
ภาพที่ 2.41 แสดง Tool Bar

3. Project Window เป็นเครื่องมือที่ใช้ควบคุมการทำงานของโปรเจ็กต์เพราะ Visual Basic สนับสนุนการสร้างแอพพลิเคชั่นได้หลายแบบ



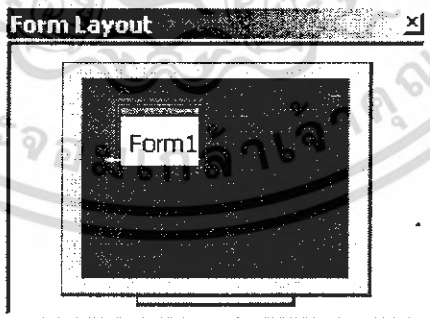
ภาพที่ 2.42 แสดง Project Window เครื่องมือที่ใช้ควบคุมการทำงานของโปรเจ็กต์

4. Properties Window เป็นส่วนที่กำหนดพรอพเพอร์ตี้ให้กับออบเจ็กต์ต่างๆ ในแอพพลิเคชั่น



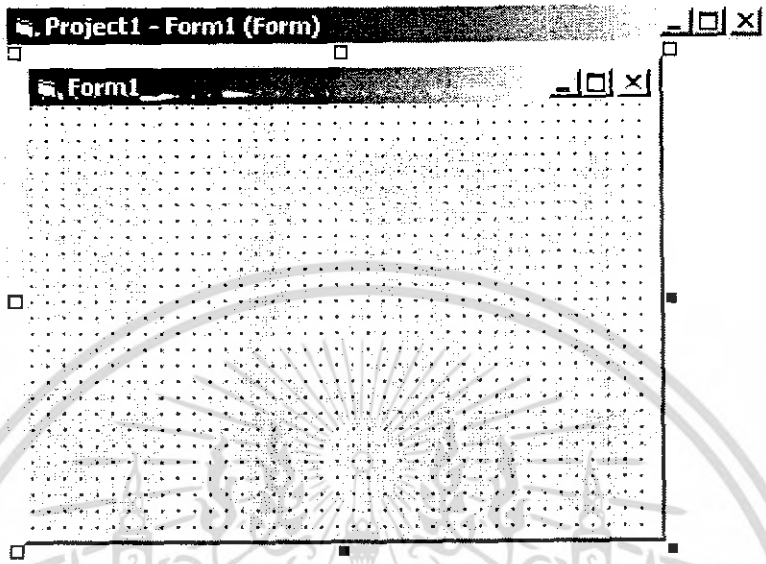
ภาพที่ 2.43 แสดงการกำหนดพรอพเพอร์ตี้ให้กับออบเจ็กต์

5. Form Layout เป็นหน้าตาจำลองของฟอร์มที่ได้จากการรันแอพพลิเคชั่น



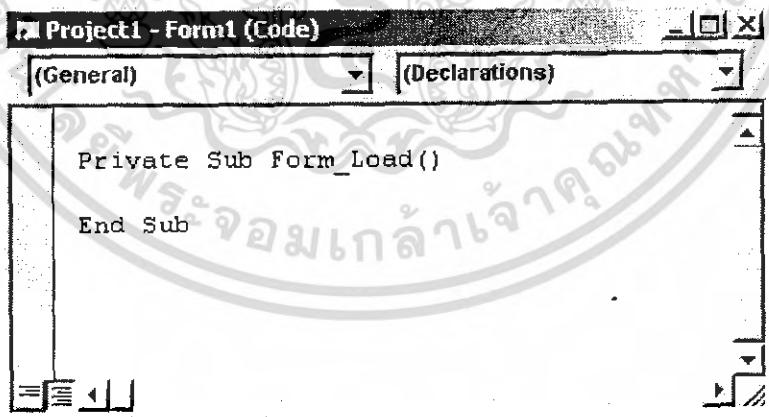
ภาพที่ 2.44 แสดง Form Layout

6. Form Designer เป็นส่วนที่มองเห็นในขณะออกแบบแอพพลิเคชั่น ของ Visual Basic เป็นส่วนที่จะใช้ติดต่อกับผู้ใช้งาน โดยการนำ ActiveX Control ต่างๆ มาวางไว้



ภาพที่ 2.45 แสดง Form Designer

7. Code Window เป็นส่วนที่ใช้เขียนโปรแกรม เพื่อควบคุมการทำงานของแอพพลิเคชั่น



ภาพที่ 2.46 แสดง Code Window

8. ข้อกำหนดและคำสั่งต่างๆ ของ Visual Basic ในการเขียนโปรแกรมด้วย Visual Basic นั้นจำเป็นที่จะต้องทราบถึงข้อกำหนดต่างๆ และคำสั่งในการทำงานต่างๆ ของ Visual Basic ซึ่งจะมีความใกล้เคียงกับภาษา Basic ทั่วไป ข้อกำหนดและคำสั่งต่างๆ ของ Visual Basic มีดังนี้

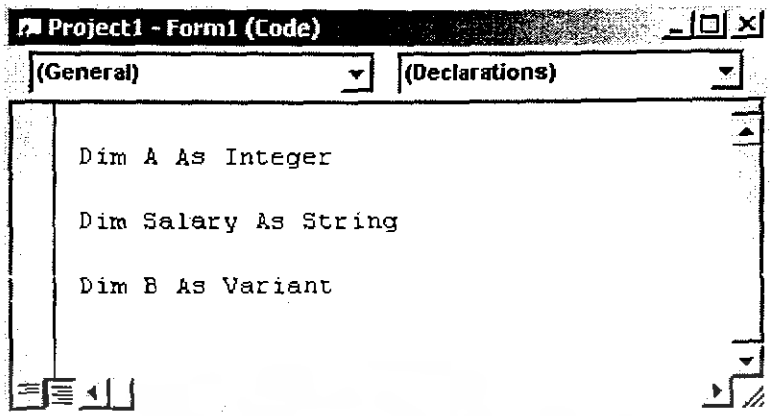
8.1 ชนิดของข้อมูลที่ใช้ใน Visual Basic ไมโครซอฟท์ได้กำหนดมาตรฐานของข้อมูลที่ใช้ในโปรแกรมหรือโค้ดสำหรับ Visual Basic ดังนี้

- Byte ข้อมูลตัวเลขจำนวนตั้งแต่ 0 ถึง 255 ขนาดหน่วยความจำ 1 ไบต์
- Boolean ข้อมูลทางตรรก : จริง (True) , เท็จ (False) ขนาดหน่วยความจำ 2 ไบต์
- Integer -32,768 ถึง 32,767 ขนาดหน่วยความจำ 2 ไบต์
- Long จำนวนเต็มระหว่าง -2,147,483,648 ถึง 2,147,483,648 ขนาดหน่วยความจำ 4 ไบต์
- Single เลขทศนิยมระหว่าง -3.402823E38 ถึง -1.401298E-4 สำหรับ ค่าลบและ 1.401298E-4 ถึง 3.402823E38 สำหรับค่าบวก 4 ไบต์
- Double เลขทศนิยมระหว่าง - 1.79769313486232E308 ถึง -4.94065645841247E-324 สำหรับค่าลบและ 4.94065645841247E-324 ถึง 1.79769313486232E308 สำหรับค่าบวก ขนาดหน่วยความจำ 8 ไบต์
- Currency เลขที่มีค่าตั้งแต่ -922,337,203,685,477.5808 ถึง 922,337,203,685,477.5807 ขนาดหน่วยความจำ 8 ไบต์
- Date วันที่ตั้งแต่ 1 มกราคม ค.ศ. 100 ถึง 31 ธันวาคม ค.ศ. 999 ขนาดหน่วยความจำ 8 ไบต์
- String เก็บสตริงหรือข้อความที่เรียงต่อกัน ขนาดหน่วยความจำ 64 KB หรือ 2MB
- Object ข้อมูลที่อ้างอิงออปเจ็กต์ ซึ่งเก็บแอดเดรสของออปเจ็กต์ไว้ขนาดหน่วยความจำ 4 ไบต์
- Variant ข้อมูลชนิดพิเศษที่เก็บค่าได้ทุกแบบ (รวมไปถึงค่าพิเศษต่างๆ ที่มีตัวเลข) เช่น EMPTY , NULL เป็นต้น) ขนาดหน่วยความจำ 16 ไบต์

8.2 การกำหนดตัวแปรและค่าคงที่ ในการกำหนดตัวแปรให้กับโปรแกรมนั้นจะใช้คำสั่ง Dim กำหนดตัวแปรดังนี้

Dim Variable [As Type]

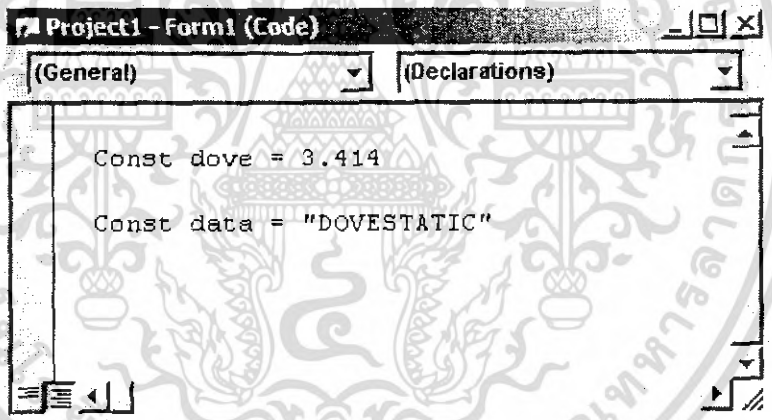
Variable หมายถึงชื่อของตัวแปร สำหรับ Type หมายถึงชนิดของข้อมูล ในส่วนที่อยู่ในวงเล็บนี้หากไม่มีจะเป็นข้อมูลแบบ Variant สำหรับการกำหนดตัวแปรมีตัวอย่างดังนี้



ภาพที่ 2.47 แสดงการกำหนดตัวแปร

สำหรับตัวแปรที่เก็บข้อมูลแบบตัวเลขซึ่งมีค่าคงที่จะใช้คำสั่ง Const กำหนดดังนี้

Const Name = Expression



ภาพที่ 2.48 แสดงการกำหนดค่าคงที่

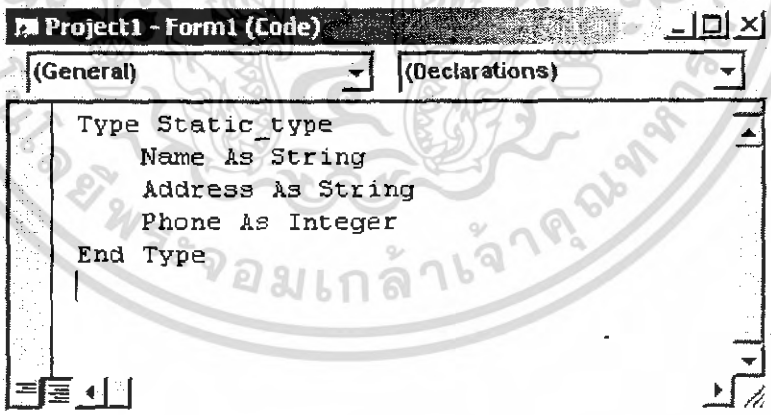
ถ้าต้องการเจาะจงแบบของข้อมูลสำหรับคำสั่งแบบ Const สามารถใส่อักขระพิเศษหลังตัวแปรหรือที่ค่าของตัวแปรเพื่อกำหนดแบบของข้อมูล ความหมายของอักขระพิเศษที่ใช้ในการกำหนดแบบของข้อมูลของ Visual Basic แสดงดังตารางที่ 2.1

ตารางที่ 2.1 แสดงอักขระพิเศษที่ใช้ในการกำหนดภาพแบบของข้อมูล

อักขระ	แบบของข้อมูล
%	Integer
&	Long
!	Single
#	Double
@	Currency
$	String

8.3 แบบข้อมูลที่ผู้ใช้กำหนดขึ้นเอง การกำหนดแบบของข้อมูลขึ้นเองนั้นสามารถทำได้โดยใช้คำสั่ง Type ซึ่งมีภาพแบบดังต่อไปนี้

```
Type Defined-type
Element [As Type]
[Element [As Type]
.....
End Type
```



ภาพที่ 2.49 แสดงการกำหนดตัวแปรขึ้นเอง

เมื่อกำหนดแบบของข้อมูลขึ้นใช้แล้ว ต้องกำหนดตัวแปรให้เป็นข้อมูลนี้เพื่อใช้ในโปรแกรมดังนี้

```
Dim Variable As Defined-type
Dim Mybook As BOOK
```

วิธีในการอ้างอิงข้อมูลซึ่งกำหนดขึ้นเองนั้น สามารถทำได้ดังตัวอย่าง

```
Mybook. Title = "Visual BASIC 4.0"
Mybook. Serial = 220
Mybook. Author = "K. Nakamura"
```

9. การตัดสินใจ (Decision) ในการเขียนโปแกรม บางครั้งจำเป็นต้องตัดสินใจให้แอพพลิเคชั่นทำงานอย่างใดอย่างหนึ่งจากทางเลือกที่จะมีให้เลือกมากกว่า 1 ทางเลือก โดยการตัดสินใจในโปรแกรม จะมี 2 ประเภท คือ

9.1 ตัดสินใจเลือก จากทางเลือก 2 ทางเลือก : If...Then...Else

```
If <ทดสอบเงื่อนไขว่าจริง หรือเท็จ> Then
    ถ้าเป็นจริงให้ทำงานหลังคำว่า Then
Else
    ถ้าเป็นเท็จให้ทำงานหลังคำว่า Else
End If
```

ในบางครั้งจะใช้ If...Then...Else เพื่อตัดสินใจ จากทางเลือกมากกว่า 2 ทางเลือกโดยการตัดสินใจเลือกทีละ 2 ทางเลือก

```
If <ทดสอบเงื่อนไขว่าจริง หรือเท็จ> Then
    ถ้าเป็นจริงให้ทำงานหลังคำว่า Then
Else If <ทดสอบเงื่อนไขว่าจริง หรือเท็จ> Then
    ถ้าเป็นจริงให้ทำงานหลังคำว่า Then เสมอ
Else If
    :
Else
    ถ้าเป็นเท็จให้ทำงานหลังคำว่า Else
End If
```

9.2 ตัดสินใจเลือก จากทางเลือกมากกว่า 2 ทางเลือก : Select...Case

```
Select Case <ทำงานตามเงื่อนไข>
Case เงื่อนไขแรก : <ทำงานตามเงื่อนไขแรก>
Case เงื่อนไขที่สอง : <ทำงานตามเงื่อนไขที่สอง>
    :
Case เงื่อนไขสุดท้าย : <ทำงานตามเงื่อนไขสุดท้าย>
```

Case Else <เมื่อไม่ตรงกับเงื่อนไขใดๆเลย ทำงานหลังคำว่า Else>

End Select

10. การวนซ้ำ (Iteration) ในการเขียนโปรแกรม เราอาจจำเป็นต้องสั่งให้แอพพลิเคชั่นทำงานซ้ำตามจำนวนครั้งที่ต้องการได้ ซึ่งภาพแบบของการวนซ้ำมี 2 ภาพแบบ ได้แก่

10.1 การวนซ้ำด้วยจำนวนรอบที่แน่นอน : For...Next เป็นภาพแบบการวนซ้ำที่เราสามารถกำหนดรอบของการวนซ้ำได้แน่นอน

For ตัวแปรใช้นับจำนวนรอบ = จำนวนรอบเริ่มต้น To จำนวนรอบสุดท้าย

<ทำงานตามคำสั่ง>

Next ตัวแปรที่ใช้นับจำนวนรอบ

จะเห็นว่าต้องใช้ตัวแปร 1 ตัวทำหน้าที่ตัวนับรอบในการวนซ้ำ ซึ่งตัวนับนี้สามารถนับได้ทั้งแบบเดินหน้า (ตัวนับมีค่าเพิ่มขึ้น) และแบบถอยหลัง (ตัวนับมีค่าลดลง)

10.2 การวนซ้ำด้วยจำนวนรอบที่ไม่แน่นอน จะมี 2 ภาพแบบ คือ

10.2.1 การวนซ้ำด้วยจำนวนรอบที่ไม่แน่นอน : While...Wend เป็นภาพแบบการวนซ้ำที่ไม่สามารถกำหนดเงื่อนไขการวนซ้ำได้ โดยอาศัยการตรวจสอบเงื่อนไขก่อนการวนซ้ำว่าต้องวนซ้ำอีกหรือไม่

While <ทดสอบเงื่อนไขว่าจริง หรือเท็จ>

<ถ้าเงื่อนไขเป็นจริง ให้ทำงานตามคำสั่ง>

Wend

สำหรับเงื่อนไขการหลุดจากการวนซ้ำ นั้นจะต้องทำให้เงื่อนไขเป็นเท็จ (False) เท่านั้นจึงจะหลุด

10.2.2 การวนซ้ำด้วยจำนวนรอบที่ไม่แน่นอน : Do/While...Unitl/Loop

Do While <ทดสอบเงื่อนไข จริงหรือเท็จ>

<ถ้าเงื่อนไขเป็นจริง ให้ทำงานตามคำสั่ง>

Loop

Do Unitl <ทดสอบเงื่อนไข จริงหรือเท็จ>

<ถ้าเงื่อนไขเป็นจริง ให้ทำงานตามคำสั่ง>

Loop

Do

<ทำงานตามคำสั่ง>

Loop While <ทดสอบเงื่อนไข จริงหรือเท็จ ถ้าเป็นจริงให้กลับไปทำงานอีกรอบ>

Do

<ทำงานตามคำสั่ง ก่อน 1 รอบ>

Loop Unitl <ทดสอบเงื่อนไข จริงหรือเท็จ ถ้าเป็นจริงให้กลับไปทำงานอีกรอบ>
จะเห็นว่าสามารถตรวจสอบเงื่อนไขได้ 2 แบบ

1. While จะหลุดจากการวนซ้ำได้เมื่อทดสอบเงื่อนไขแล้วเป็นเท็จ (False)

2. Unitl จะหลุดจากการวนซ้ำได้เมื่อทดสอบเงื่อนไขแล้วเป็นจริง (True)

นอกจากนี้สามารถวางตำแหน่งการตรวจสอบเงื่อนไขได้ ว่าจะตรวจสอบก่อนการวนซ้ำ หรือหลังจากการวนซ้ำ

10.3 การกระโดดออกจากการวนซ้ำในบางครั้งเราจำเป็นต้องกระโดดออกมา จากการวนซ้ำ ด้วยเงื่อนไขบางอย่างสามารถทำได้ โดยใช้คำสั่ง Exit For , Exit Loop

11. โปรแกรมย่อย (Procedure) ในการเขียนโปรแกรมย่อยเกิดการทำงานที่ซ้ำๆ กัน สามารถสามารถลดการทำงานซ้ำๆนั้น โดยการเขียนโปรแกรมย่อยเก็บไว้ เมื่อต้องการทำงานแบบเดิมอีกเพียงแต่เรียกโปรแกรมย่อยนั้นมาใช้งาน ทำให้ลดเวลาการเขียนโปรแกรมและลดความยุ่งยากของโปรแกรมที่เขียนด้วย

โปรแกรมย่อยแบ่งออกเป็น 2 ประเภท ตามลักษณะของการคืนค่ากลับมาหลังจากจบโปรแกรมย่อย

11.1 Sub (Sub Routine) เป็นโปรแกรมย่อยที่เขียนขึ้นมา เมื่อโปรแกรมย่อยทำงานเสร็จแล้วจะไม่มีการคืนค่าใดๆ กลับมายังผู้ที่เรียกใช้งาน ซึ่งสับรูทีนมีโครงสร้างดังนี้

```
Sub ชื่อสับรูทีน (รายการอาร์กิวเมนต์)
        :
< คำสั่งใน Visual Basic>
        :
End Sub
```

11.2 Function ฟังก์ชั่นเป็นโปรแกรมย่อยที่ต้องคืนค่ากลับมาหาผู้เรียกใช้หลังจากโปรแกรมย่อยทำงานเสร็จแล้วซึ่งฟังก์ชั่น มีโครงสร้างดังนี้

```
Function ชื่อฟังก์ชั่น (รายการอาร์กิวเมนต์) As ชนิดข้อมูลที่คืนให้ผู้เรียกใช้
        :
< คำสั่งใน Visual Basic>
        :
End Function
```

12. ขอบเขตการใช้งานตัวแปรทั้งตัวแปรและค่าคงที่ที่ได้กล่าวมาแล้ว โดยปกติจะมีขอบเขตการใช้งานเพื่อให้สะดวก เหมาะสมและมีความเป็นเหตุเป็นผล ตัวแปรหรือค่าคงที่ใน Visual Basic จะมีขอบเขตการใช้งาน 3 ระดับคือ

12.1 ตัวแปรโลคอล (Local) เป็นตัวแปรที่มีขอบเขตอยู่ในระดับโพรซีเดอร์หรือในฟังก์ชั่นที่ประกาศในตัวแปรนั้น ส่วนมากเพื่อการใช้งานเป็นส่วนย่อยๆ เพื่อเก็บค่าเพียงชั่วคราว เมื่อโพรซีเดอร์หรือฟังก์ชั่นนั้นสิ้นสุด ค่าจะถูกยกเลิกด้วย

12.2 ตัวแปรโมดูล (Module) เป็นตัวแปรที่ประกาศเพื่อใช้ในหลายโพรซีเดอร์หรือหลายฟังก์ชั่นในโมดูลนั้น รวมถึงโมดูลที่มีหลายรูทีนและโมดูลของฟอร์มที่มีรูทีนสำหรับการทำงานกับ ออปเจคในฟอร์มนั้นประกาศตัวแปรในโมดูล จะประกาศในส่วน Declaration

12.3 ตัวแปรแบบโกลบอล (Global) เป็นตัวแปรที่ประกาศได้ทุกส่วนของ โปรเจกร์ ตัวแปรนั้นสามารถนำไปใช้ได้ในทุกรูทีนและทุกส่วน การประกาศเช่นเดียวกับ ตัวแปรทั่วไป เพียงแต่เพิ่มคำว่า "Global" นำหน้าตัวแปรนั้น ภาพแบบการประกาศเป็นดังนี้

เช่น Global variable As type

Global A As Integer

Global Const Pi = 3.1415

13. ตัวแปรแบบอาร์เรย์ (Array Variable) ตัวแปรแบบอาร์เรย์เปรียบเสมือนตัวแปรซ้อนตัวแปร มีประโยชน์มากสำหรับโปรแกรมที่มีตัวแปรแบบลักษณะเดียวกัน ภาพแบบการกำหนดตัวแปรแบบอาร์เรย์เป็นดังนี้

เช่น Dim variable (size) As type

Dim X(3) As Integer หรือ X%(3)

สำหรับการกำหนดค่าหรือการรับค่าจากตัวแปรอาร์เรย์ สามารถระบุได้ดังนี้

Dim X(1) = 5

Dim X(2) = 10

Dim X(3) = 15

14. ตัวดำเนินการในการประมวลผลข้อมูลทางคณิตศาสตร์ ทางตรรกศาสตร์ การเปรียบเทียบข้อมูลและการประมวลผลข้อมูลชนิดสตริง นั้นต้องใช้ตัวดำเนินการต่างๆ ซึ่ง Visual Basic ได้กำหนดไว้ดังนี้

14.1 ตัวดำเนินการทางคณิตศาสตร์ (Arithmetic Operator)

+ การบวก (Addition)

- การลบ (Subtraction)

* การคูณ (Multiplication)

^ การยกกำลัง (Exponentiation)

/ การหารผลลัพธ์ที่ได้เป็นเลขจำนวนจริง (Floating-Point Division)

\ การหารผลลัพธ์ที่ได้เป็นเลขจำนวนเต็ม (Integer Division)

Mod การหาเศษจากการหารเลขจำนวนเต็ม 2 จำนวน (Modulus)

14.2 ตัวดำเนินการทางตรรกศาสตร์ (Logica l Operator) ตัวดำเนินการนี้ ในบางครั้งเรียกว่าเป็นตัวดำเนินการ Boolean เพราะตัวดำเนินการเหล่านี้ได้มาจากพีชคณิต ของตรรกคณิตศาสตร์ ซึ่งพัฒนาขึ้นโดย จอร์จ บูล ตัวดำเนินการเหล่านี้มีค่า Boolean เป็นโอเปอแรนด์และตัวดำเนินการจะส่งคืนผลลัพธ์ Boolean

ตารางที่ 2.2 แสดงตัวดำเนินการ AND

การดำเนินการ	ผลลัพธ์
False And False	False
False And True	False
True And False	False
True And True	True

ตารางที่ 2.3 แสดงตัวดำเนินการ OR

การดำเนินการ	ผลลัพธ์
False Or False	False
False Or True	True
True Or False	True
True Or True	True

ตารางที่ 2.4 แสดงตัวดำเนินการ XOR

การดำเนินการ	ผลลัพธ์
False Or False	False
False Or True	True
True Or False	True
True Or True	False

ตารางที่ 2.5 แสดงตัวดำเนินการ IMP

การดำเนินการ	ผลลัพธ์
False Imp False	True
False Imp True	True
True Imp False	False
True Imp True	True

ตารางที่ 2.6 แสดงตัวดำเนินการ EQV

การดำเนินการ	ผลลัพธ์
False Eqv False	True
False Eqv True	False
True Eqv False	False
True Eqv True	True

ตารางที่ 2.7 แสดงตัวดำเนินการ NOT

การดำเนินการ	ผลลัพธ์
NOT False	True
NOT True	False

14.3 ตัวดำเนินการเชื่อมสตริง (String Concatenation Operator) ตัวดำเนินการทางสตริงจะใช้เครื่องหมาย + เป็นตัวเชื่อมสตริง ตัวอย่างเช่น

กำหนดให้ Name$ = "Dumrong"

ทำการเชื่อมสตริงดังนี้ "My Name is + Name$ +"."

ผลลัพธ์คือ My Name is Dumrong

นอกจากนี้ยังสามารถใช้เครื่องหมาย & เป็นตัวเชื่อมข้อมูลแบบสตริง ข้อมูลแบบตัวแปร (Integer , Long , Currency , Single หรือ Double) เข้าด้วยกัน โดยที่ Visual Basic จะเปลี่ยนเป็น ข้อมูลแบบสตริงโดยอัตโนมัติซึ่งผลลัพธ์ที่ได้จะเป็นข้อมูลแบบสตริง

ตัวอย่างเช่น Varl = 76

Varl2 = "trombones"

MsgBox Varl& " " & Varl2

ผลลัพธ์ที่ปรากฏคือ 76 trombones

14.4 ตัวดำเนินการเปรียบเทียบ (Comparison Operator หรือ Relational Operator) ตัวดำเนินการเปรียบเทียบใช้กับข้อมูลได้ทุกชนิด ผลของการเปรียบเทียบที่ได้ คือ True และ False ซึ่งเป็น Built-in Constant ของ Visual Basic ค่าแท้จริงของ True และ False จะเป็นเลขจำนวนเต็ม -1 และ 0 ตามลำดับ ผลจากการเปรียบเทียบจะเป็นค่าตรรกตัวดำเนินการเปรียบเทียบที่ใช้ใน Visual Basic ดังตารางที่ 2.8

ตารางที่ 2.8 แสดงตัวดำเนินการการเปรียบเทียบ

ตัวดำเนินการ	ความหมายของตัวดำเนินการ
>	มากกว่า (Greater Than)
<	น้อยกว่า (Less Than)
>=	มากกว่าหรือเท่ากับ (Greater Than Or Equal To)
<=	น้อยกว่าหรือเท่ากับ (Less Than Or Equal To)
=	เท่ากับ (Equal To)
<>	ไม่เท่ากับ (Not Equal To)

ตัวอย่างเช่น 4 > 2^2 ให้ค่าเป็น False

a <= Sum

"Exit" = "E" + "x" + "I" + "t" ให้เป็นค่า True

True <= 12 ให้ค่าเป็น True เพราะ –1 น้อยกว่า 12

15. ลำดับการประมวลผลข้อมูล ในกรณีที่นิพจน์มีตัวดำเนินการหลายแบบรวมอยู่ด้วยกัน หากมีการใส่วงเล็บ Visual Basic จะประมวลผลข้อมูลในวงเล็บในก่อน จากนั้นจึงประมวลผลข้อมูลตามลำดับความสำคัญของตัวดำเนินการ ในกรณีที่ตัวดำเนินการมีความสำคัญเท่ากันจะประมวลผลนิพจน์จากซ้ายไปขวาและจากตารางที่ 2.9 แสดงรายการของชุดของตัวดำเนินการของ Visual Basic ทั้งหมด (ยกเว้นตัวดำเนินการเชื่อมต่อสตริง) เรียงลำดับความสำคัญ ของตัวดำเนินการที่ใช้ในนิพจน์ โดยเรียงลำดับตามการทำก่อนจากสูงสุดไปยังต่ำสุด เมื่อพบกับนิพจน์ย่อยนั้นเป็นนิพจน์แรก หลังจากนั้นหาว่าตัวดำเนินการใดที่มีการทำก่อนอยู่ในลำดับรองลงมา ให้ประเมินผลนิพจน์ย่อยนั้น และทำเช่นนี้ต่อเนื่องไปจนกว่าจะได้ผลลัพธ์ในขั้นสุดท้าย

ตารางที่ 2.9 แสดงลำดับของการคำนวณตัวดำเนินการ

ลำดับ	ตัวดำเนินการ	สัญลักษณ์
1	ยกกำลัง	^
2	เท่ากับ	=
3	NOT	NOT
4	ไม่เท่ากับ	<>
5	AND	AND
6	คูณหาร	*/
7	น้อยกว่า	<
8	OR	OR
9	หารเอาจำนวนเต็ม	\
10	มากกว่า	>
11	XOR	XOR
12	เศษการหาร	MOD
13	น้อยกว่าหรือเท่ากับ	<=
14	บวก ลบ	+ -
15	มากกว่าหรือเท่ากับ	>=
16	รวมข้อความ	&

16. ทูลบ็อกซ์ (Tool Box) ส่วนประกอบของแอพพลิเคชั่นที่ได้บรรจุเอาไว้บนแบบฟอร์ม เรียกว่า Object หรือตัวควบคุม ทูลบ็อกซ์เป็นที่รวม Object ต่างๆ ที่จะนำมาประกอบในแอพพลิเคชั่น โดย Object จะมีดังตารางที่ 2.10

ตารางที่ 2.10 แสดงทูลบ็อกซ์ใน Visual Basic

ออปเจ็กต์	คุณสมบัติ
ab	Text Box ใช้สำหรับข้อความซึ่งอาจยาวเกินกว่าจะแสดงในกรอบหรือ Box ให้เห็นพร้อมกันหมดได้ Text เป็นคุณสมบัติที่เก็บข้อความปรากฏอยู่ในกรอบหรือ Box เวลาจะเอาข้อความไปใช้อ้างถึงคุณสมบัติ Text นี้

ออปเจ็กต์	คุณสมบัติ
	Image ทำหน้าที่แสดงภาพ เช่นเดียวกับ Picture Box ทั้งภาพในแบบ Bitmap , Icon หรือ Metafile ข้อดีของ Image คือ ใช้ทรัพยากรของระบบน้อยกว่า Picture Box และยังทำการวาดภาพใหม่ได้เร็วกว่าด้วย หน้าที่หลักของ Image คือ แสดงภาพเท่านั้น
	Data Control ทำหน้าที่สร้างตัวควบคุมข้อมูล โดยอาศัยตัวควบคุมนี้ ในการเข้าถึงข้อมูลจำเพาะในฐานข้อมูล โดยคุณสมบัติและวิธีการใช้งานของตัวควบคุมข้อมูลนี้ จะแสดงอย่างละเอียดในหัวข้อ Data Control
	DBCombo มีลักษณะคล้ายกับคอมโบบ็อกซ์ที่เป็นมาตรของ Window 95 แต่สามารถแสดงข้อมูลฟิลด์ในเรคคอร์ดเซตที่อยู่ในดาต้า-คอนโทรลได้ โดยทั่วไปเราใช้ DBCombo คู่กับ DBGrid ในการตั้งเงื่อนไขในประโยคย่อย WHERE ของนิพจน์ SQL
	Command Button เป็นคอนโทรลพื้นฐานที่ใช้บ่อยที่สุดคือ ใช้เป็นปุ่มในการสั่งให้ทำงาน เพราะอย่างน้อยที่สุดเกือบทุกฟอร์มจะมีปุ่ม Ok และ Cancel อยู่ด้วย
	Label มีหน้าที่แสดงข้อความให้ปรากฏฟอร์ม โดยผู้ใช้ไม่สามารถแก้ไขได้เหมือน Text Box และเหมือนกับคอนโทรลอื่นๆ อีกหลายตัวที่มีคุณสมบัติ Caption ซึ่งเก็บข้อความให้เห็น
	Timer ใช้เพื่อให้ทำงานในทุกๆ ช่วงเวลาที่กำหนดได้และสามารถทำงานในแบบฉากหลังได้ ในตอนรันโปรแกรมจะไม่เห็นตัวคอนโทรลในขณะที่รันโปรแกรมขึ้นมา
	OLE Container ใช้ในการเชื่อม (Linking) และฝัง (Embedding) ออปเจ็กต์จากโปรแกรมอื่น นำความสามารถที่โปรแกรมอื่นมีอยู่แล้วมาใช้ได้โดยไม่ต้องสร้างขึ้นใหม่เอง
	Picture Box เป็นออปเจ็กต์ที่แสดงภาพในภาพของ Bitmap , Icon หรือ Metafile โดยที่แสดงภาพเท่าที่แสดงได้ในขนาดของตัวในเท่านั้น นั่นคือ หากขนาดของภาพโตกว่า Picture Boxจะเห็นภาพไม่หมด นอกจากนี้อาจใช้คำสั่งด้านกราฟิกเพื่อวาดภาพใน Picture Box รวมไปถึงการใช้ Method Print ได้ด้วย หรือจะใช้แสดงข้อความ (Text) ได้ซึ่ง Picture Box มีคุณสมบัติเกี่ยวกับ Text ไว้ให้ด้วย

ออปเจ็กต์	คุณสมบัติ
	DBList มีลักษณะคล้ายกับลิสต์บ็อกซ์ที่เป็นมาตรฐานของ Window 95 แต่สามารถแสดงข้อมูลของฟิลด์ในเรคคอร์ดเซตที่อยู่ในคาต้า-คอนโทรลได้ DBList ทำงานร่วมกับคาต้าคอนโทรล

17. Data Control ตัวควบคุม Data มีคุณสมบัติจำนวนหนึ่งที่นิยามการติดต่อระหว่างแอพพลิเคชั่น Visual Basic กับฐานข้อมูลที่ต้องการเข้าถึง แอพพลิเคชั่นสามารถมีตัวควบคุม Data มากกว่าหนึ่งตัวและชุดข้อมูลที่นิยมไว้แต่ละชุดต้องการวัดควบคุม Data ที่แยกต่างหากกันตัวควบคุม Data แต่ละตัวจะเข้าถึงได้เพียงทีละหนึ่งเรคคอร์ด ซึ่งเรียกว่า เรคคอร์ดล่าสุด (Current Record) หลังจากตั้งคุณสมบัติเหล่านี้แล้ว จะสามารถยึดเหนี่ยวตัวควบคุมอื่นๆ ของ Visual Basic เข้ากับตัวควบคุม Data เมื่อเปลี่ยนแปลงเรคคอร์ดล่าสุด ตัวควบคุมเหล่านั้น จะแสดงข้อมูลที่อยู่ในเรคคอร์ดล่าสุดออกมา Visual Basic นี้สนับสนุนการแลกเปลี่ยนข้อมูลกับ DBMS คือ Microsoft Access , Microsoft Foxpro , dBase และ Parabox

2.2 ความรู้เบื้องต้นเกี่ยวกับพอร์ตอนุกรม

การเคลื่อนย้ายข้อมูลจากคอมพิวเตอร์ไปยังอุปกรณ์ต่อพวงภายนอกหรือคอมพิวเตอร์ด้วยกันมี 2 รูปแบบ คือ รับส่งข้อมูลแบบขนานและรับส่งข้อมูลแบบอนุกรม

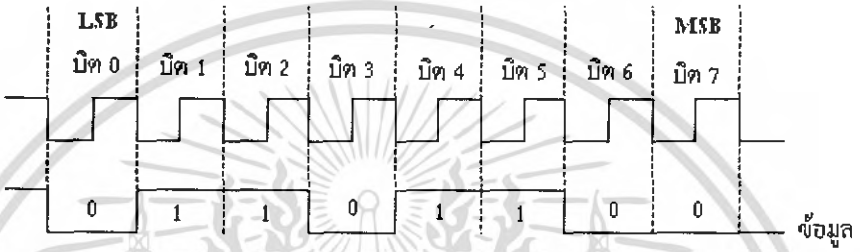
การส่งข้อมูลแบบขนาน เป็นการรับและส่งข้อมูลคราวละ 4 ถึง 8 บิตในเวลาเดียวกัน ทำให้การรับส่งข้อมูลมีความเร็วสูง แต่จำนวนสายที่ใช้ในการถ่ายทอดข้อมูลมีมากเท่ากับจำนวนบิตของข้อมูลที่ทำการถ่ายทอด นอกจากนั้นยังมีสายที่ใช้สำหรับควบคุม และตรวจสอบการรับส่งข้อมูลด้วยซึ่งอาจต้องใช้สายมากเป็น 2 เท่าของจำนวนบิตข้อมูล

ในขณะที่การับส่งข้อมูลแบบอนุกรม จะเป็นการรับส่งข้อมูลครั้งละ 1 บิต โดยมีรูปแบบการรับส่งที่เป็นมาตรฐาน ต้องมีการตรวจสอบความพร้อมในการรับและส่งข้อมูลของตัวส่งและตัวรับ การรับส่งข้อมูลแบบอนุกรมมีข้อดีในเรื่องจำนวนสายสัญญาณที่น้อยมากและไม่แปรผันตามจำนวนบิตของข้อมูล ระยะทางในการรับส่งข้อมูลสูงกว่าแบบขนานมาก

การสื่อสารแบบอนุกรมแบ่งออกเป็น 2 แบบคือ การสื่อสารอนุกรมแบบซิงโครนัสและการสื่อสารอนุกรมแบบอะซิงโครนัส

2.2.1 การสื่อสารข้อมูลแบบซิงโครนัส

การสื่อสารอนุกรมแบบซิงโครนัสจะมีสัญญาณนาฬิการ่วมอยู่กับการรับและส่งสัญญาณ ด้วยตัวอย่างการส่งข้อมูลแบบซิงโครนัสก็คือคีย์บอร์ดของคอมพิวเตอร์ ซึ่งสายเส้นหนึ่งจะเป็นสายของสัญญาณนาฬิกา ส่วนสายอีกเส้นหนึ่งจะเป็นสายของข้อมูล ดังนั้นการติดต่อกันแบบซิงโครนัสนี้จะต้องใช้สายในการเชื่อมต่ออย่างน้อยที่สุด 3 เส้นคือ สัญญาณนาฬิกา ข้อมูลและกราวด์ ภาพด้านล่างแสดงให้เห็นถึงไดอะแกรมเวลาของการสื่อสารข้อมูลแบบซิงโครนัส



ภาพที่ 2.50 แสดงแผนการทำงานเวลาของการสื่อสารข้อมูลแบบซิงโครนัส

2.2.2 การสื่อสารข้อมูลแบบอะซิงโครนัส

การสื่อสารข้อมูลแบบอะซิงโครนัส คือ การรับข้อมูลโดยไม่จำเป็นต้องมีสัญญาณนาฬิการ่วมด้วย แต่จะใช้การกำหนดค่าอัตราความเร็วในการรับและส่งข้อมูลให้มีค่าเท่ากัน ซึ่งเรียกอัตราเร็วนี้ว่า อัตราบอดเรต (Baud Rate) มีหน่วยเป็น บิตต่อวินาที

รูปแบบของข้อมูลที่ใช้ในการรับส่งแบบอะซิงโครนัสประกอบด้วย 4 ส่วน ด้วยกันคือ

1. บิตเริ่มต้น
2. บิตข้อมูลแบบอนุกรม มีขนาด 5,6,7 หรือ 8 บิต
3. บิตตรวจสอบพาริตี้(Parity Bit) มีขนาด 1 บิต หรือไม่มีบิต
4. บิตปิดท้ายหรือบิตหยุด(Stop Bit) มีขนาด 1,1.5 หรือ 2 บิต

ภาพที่ 2.51 แสดงรูปแบบของข้อมูลอนุกรมแบบอะซิงโครนัส เมื่อไม่มีการส่งข้อมูล ขา DATA จะมีสถานะลอจิก "1" เรียกสถานะนี้ว่า สถานะหยุดรอ (Waiting Stage) การเริ่มต้นส่งข้อมูลจะเริ่มจากการให้ขา DATA มีลอจิก "0" ด้วยช่วงระยะเวลา 1 บิต เรียกบิตนี้ว่าบิตเริ่มต้น (Start Bit) จากนั้นบิตข้อมูลจะถูกส่งออกไป โดยเริ่มจากบิตที่มีนัยสำคัญต่ำสุดหรือบิต LSB ก่อน ซึ่งข้อมูลที่ต้องการส่งอาจมีจำนวน 5,6,7 บิต 8 บิตก็ได้ จากนั้นตามด้วยบิตพาริตี้(Parity Bit) ซึ่งใช้ในการตรวจสอบความผิดพลาดที่เกิดขึ้นจากการส่งข้อมูล บิตสุดท้ายที่ส่งก็คือ บิตปิดท้ายหรือ บิตหยุด (Stop Bit) โดยจะเป็นการทำให้ขา DATA จะมีสถานะลอจิก "1" อีกครั้งด้วยระยะเวลาอย่างน้อย 1,1.5 หรือ 2 บิต เพื่อเป็นการแสดงว่าสิ้นสุดข้อมูลแล้ว

อัตราความเร็วในการรับและส่งข้อมูลของการสื่อสารข้อมูลแบบอะซิงโครนัสหรืออัตราบอดหรือบอดเรดที่ใช้สำหรับพอร์ตอนุกรม RS232 มีด้วยกันหลายค่า ได้แก่ 110, 150, 300, 600, 1200, 2400, 4800, 9600 หรือ 19200 บิตต่อวินาที โดยมีค่ามากขึ้นตามเทคโนโลยีของคอมพิวเตอร์ เนื่องจากบอดเรตคือค่าของจำนวนบิตที่สามารถส่งได้ใน 1 วินาที สมมติว่าข้อมูลอนุกรมมีขนาด 8 บิต ไม่มีการตรวจสอบพาริตี้ มีบิตเริ่มต้น 1 บิต และบิตปิดท้าย 1 บิต ความยาวของข้อมูล 1 ไบต์ จะมีความยาวเท่ากับ 10 บิต ถ้าใช้บอดเรตในการส่งข้อมูลเท่ากับ 9600 บิตต่อวินาที ก็สามารถส่งข้อมูลได้ด้วยความเร็ว 960 ไบต์ต่อวินาที

ภาพที่ 2.51 แสดงรูปแบบของข้อมูลแบบอะซิงโครนัส

การตรวจสอบพาริตี้สามารถกำหนดเป็นแบบคี่ (Odd), แบบคู่ (Even) หรือไม่มีการตรวจสอบพาริตี้ก็ได้ พาริตี้คี่หรือพาริตี้คู่แสดงถึงจำนวนลอจิก "1" ทั้งหมดภายในข้อมูลที่ส่งไป 1 ไบต์รวมพาริตี้ว่ามีจำนวนเป็นเลขคู่หรือเลขคี่ ยกตัวอย่างข้อมูลที่จะทำการส่งมีขนาด 8 บิต มีค่าเท่ากับ 99 H หรือ 10011001B จะเห็นว่าข้อมูลในไบต์มีจำนวนลอจิก "1" จำนวน 4 บิต ซึ่งเป็นเลขคู่ ดังนั้นถ้ากำหนดค่าพาริตี้เป็นคู่ ค่าของพาริตี้บิตจะต้องมีลอจิกเป็น "0" แต่ถ้ากำหนดพาริตี้เป็นคี่ ค่าของบิตพาริตี้จะต้องเป็น "1" เพื่อให้ข้อมูล 1 ไบต์ รวมทั้งบิตพาริตี้เป็นคี่

บิตพาริตี้ถูกสร้างขึ้นจากการส่งข้อมูลของ UART (Universal Asynchronous Receiver Transmitter) ซึ่งทางภาครับจะต้องกำหนดคุณสมบัติการตรวจสอบพาริตี้ที่ตรงกันเอาไว้ว่าจะตรวจสอบพาริตี้คู่ โดยการนับจำนวนลอจิก "1" ทั้งหมดรวมทั้งบิตพาริตี้ด้วย ถ้ากำหนดพาริตี้ไว้เป็นคู่แต่อ่านค่าตัวเลขในการนับออกมาได้ตัวเลขเป็นคี่ ทางภาครับจะแสดงข้อผิดพลาดออกมาให้ผู้ใช้ทราบ กระบวนการดังกล่าวเป็นวิธีการตรวจสอบความผิดพลาดที่เกิดขึ้นในการรับส่งข้อมูลที่ง่ายที่สุด แต่มันสามารถตรวจสอบได้เมื่อมีบิตข้อมูลที่ทำการรับส่งผิดพลาดเพียงบิตเดียวเท่านั้น ถ้าข้อมูลที่ทำการส่งมีบิตที่ผิดพลาดมากกว่า 1 บิต การตรวจสอบด้วยวิธีนี้จะไม่ได้ผลสำหรับการตั้งพาริตี้บิตเป็น None นั้นทั้งภาครับและส่ง จะไม่มีการตรวจสอบพาริตี้

คอมพิวเตอร์ในรุ่น AT เกือบทั้งหมดจะใช้ไอซี UART เบอร์ 16450 และ 16550 ส่วนคอมพิวเตอร์ในรุ่น XT ใช้ไอซี UART เบอร์ 8250 ไอซี UART เหล่านี้มีระดับแรงดันของลอจิกเป็น

แบบทีทีแอล (+5V) แต่เพื่อให้แรงดันเป็นไปตามมาตรฐาน RS232 และเพื่อให้การรับส่งข้อมูลสามารถทำได้ที่ระยะไกลมากยิ่งขึ้น ระดับแรงดันทีทีแอลจะถูกแปลงเป็นระดับแรงดันที่สูงขึ้นโดยลอจิก "0" จะมีระดับแรงดัน –3V ถึง –12V และลอจิก "1" มีระดับแรงดัน +3V ถึง +12V

2.3 ชุดรับส่ง RF 2.4 GHz

2.3.1 ลักษณะโดยทั่วไป

ET-RF24G V1.0 เป็นชุด Signal Converter สำหรับใช้แปลงสัญญาณระหว่าง RS232 และ RF-Wireless โดยในโหมดการทำงานของการส่งข้อมูล (Transmitter) จะทำหน้าที่รอรับข้อมูลจากพอร์ตสื่อสารอนุกรม RS232 จากขา RX แล้วแปลงเป็นสัญญาณความถี่ (GFSK) ส่งออกไปในอากาศ และในทางกลับกันในโหมดการทำงานแบบรับ (Receiver) ชุด ET-RF24G V1.0 ก็จะทำหน้าที่คอยตรวจจับข้อมูลที่อยู่ในรูปของสัญญาณความถี่ (GFSK) จากด้าน RF เพื่อแปลงกลับเป็นข้อมูลแบบ RS232 ส่งออกไปทางขา TX ได้ด้วย

ซึ่งจะเห็นได้ว่าชุดแปลงสัญญาณ ET-RF24G V1.0 นั้น สามารถนำไปต่อใช้งานร่วมกับพอร์ตสื่อสารอนุกรม แบบ RS232 เพื่อใช้งานในลักษณะของการสื่อสารอนุกรมแบบไร้สาย (Wireless Transceiver) ได้โดยตรงโดยจะมีข้อดีกว่า คือ สามารถรับส่งข้อมูลกันได้ในระยะทางที่ไกลกว่า RS232 หลายเท่าตัว และประการสำคัญ คือ ไม่จำเป็นต้องใช้สายสัญญาณที่เป็นตัวนำสัญญาณทางไฟฟ้าในการสื่อสารข้อมูลกัน ทำให้สามารถเปลี่ยนแปลงหรือเคลื่อนย้ายจุดรับส่งข้อมูลได้ตลอดเวลา ซึ่งถ้าเป็นการรับส่งข้อมูลด้วยระบบ RS232 แบบที่ใช้สายสัญญาณนั้นจะเกิดความยุ่งยากในการติดตั้งสายสัญญาณเป็นอย่างมาก

แต่อย่างไรก็ตามการรับส่งข้อมูลโดยใช้อากาศเป็นตัวกลางในการสื่อสารนั้น ก็มีข้อจำกัดบางประการเหมือนกันโดยเฉพาะอย่างยิ่ง เรื่องความน่าเชื่อถือของข้อมูลที่รับส่งกัน ซึ่งมีโอกาสผิดพลาดหรือสูญหายได้เหมือนกัน เนื่องจากในการสื่อสารข้อมูลนั้นไม่ได้ใช้สายสัญญาณเป็นตัวกลางในการรับส่งข้อมูล แต่ใช้อากาศเป็นตัวกลางในการรับส่งข้อมูลแทน ซึ่งมีโอกาสที่ข้อมูลจะเกิดการรบกวนจากสัญญาณอื่นๆที่มีย่านความถี่ใกล้เคียงกันแล้วทำให้ข้อมูลผิดเพี้ยนไปได้บ้างเหมือนกัน ซึ่งระบบการจัดการข้อมูลของเครื่อง ET-RF24G V1.0 นั้น มีระบบการเข้ารหัสและถอดรหัสข้อมูลที่มีความน่าเชื่อถืออยู่ในเกณฑ์ที่จัดว่าดี โดยข้อมูลแต่ละ Byte ที่มีการรับส่งกันนั้นจะมีการตรวจสอบความถูกต้องของข้อมูลให้ด้วยแล้ว โดยข้อมูลที่รับได้จากด้าน RF นั้นรับประกันได้ว่าเป็นข้อมูลที่มีความถูกต้องแน่นอน แต่อย่างไรก็ตามการรับส่งข้อมูลนั้นมีโอกาสผิดพลาดในเรื่องของการสูญหายของข้อมูลบ้างเหมือนกัน เนื่องจากกลไกในการรับส่งข้อมูลของเครื่อง ET-RF24G V1.0 นั้น จะมีการตรวจสอบข้อมูลทุก Byte ที่รับได้จาก RF เสมอ ซึ่งถ้าพบว่ามีความผิดพลาดเกิดขึ้นจะทิ้งข้อมูล Byte นั้นไป ซึ่งผู้ใช้ควรมีกลไกในการตรวจสอบข้อมูลที่รับส่งกันว่า

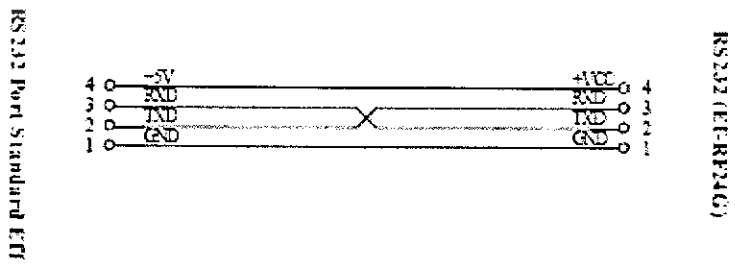
ครบถ้วนหรือไม่ด้วย ซึ่งหากพบว่ามีการสูญหายของข้อมูลเกิดขึ้นก็ให้ร้องขอให้มีการส่งข้อมูลนั้นซ้ำนั้นๆใหม่อีกครั้งหนึ่ง ก็จะสามารถแก้ไขปัญหาดังกล่าวได้

2.3.2 Power Supply

สำหรับการต่อแหล่งจ่ายไฟให้กับเครื่อง ET-RF24G V1.0 นั้น จะสามารถเลือกต่อแหล่งจ่ายไฟให้กับตัวเครื่องได้ 2 ทางด้วยกัน โดยเครื่อง ET-RF24G V1.0 นั้น ต้องการไฟเลี้ยงวงจร ซึ่งเป็นแหล่งจ่ายกระแสตรง ขนาดประมาณ +5VDC ถึง +9VDC โดยจุดเชื่อมต่อแหล่งจ่ายไฟของเครื่อง ET-RF24G V1.0 นี้ สามารถเชื่อมต่อได้ 2 จุดด้วยกัน โดยผู้ใช้สามารถเลือกต่อแหล่งจ่ายไฟให้กับเครื่อง ET-RF24G V1.0 จุดใดจุดหนึ่งก็ได้

ภาพที่ 2.52 แสดงแหล่งจ่ายไฟให้กับเครื่อง ET-RF24G V1.0

ในการนำเครื่อง ET-RF24G V1.0 ไปเชื่อมต่อกับบอร์ดไมโครคอนโทรลเลอร์ นั้นสามารถใช้แหล่งจ่ายไฟจากบอร์ดไมโครคอนโทรลเลอร์ เพื่อจ่ายให้กับตัวเครื่อง ET-RF24G V1.0 ได้ทันที โดยไม่ต้องใช้แหล่งจ่ายไฟจากภายนอก เนื่องจากขั้วต่อสัญญาณ RS232 ของบอร์ดไมโครคอนโทรลเลอร์ได้เป็นแหล่งจ่ายไฟตรง ขนาด +5V เตรียมไว้ให้ด้วยแล้ว โดยนำสายสัญญาณ RS232 ซึ่งทำการต่อสายสัญญาณครบทั้ง 4 เส้น ดังภาพมาเชื่อมต่อก็สามารถใช้งานได้แล้ว



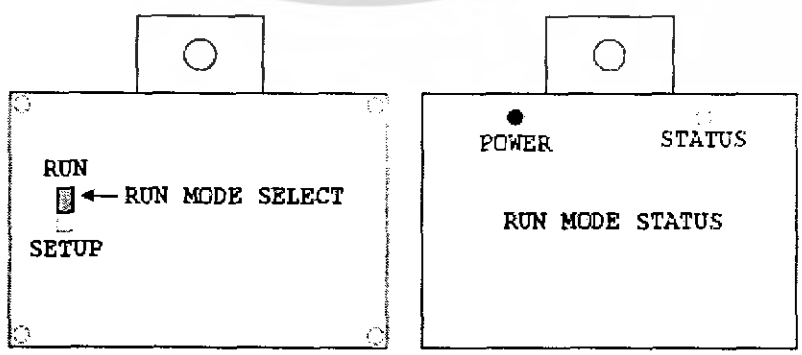
ภาพที่ 2.53 แสดงการต่อสายสัญญาณ RS232 เพื่อใช้แหล่งจ่ายไฟจากบอร์ดไมโครคอนโทรลเลอร์

2.3.3 โหมดการทำงาน

สำหรับโหมดการทำงานของ ET-RF24G V1.0 นั้นจะแบ่งออกเป็น 2 โหมดด้วยกันโดยการกำหนดโหมดการทำงานของ ET-RF24G V1.0 นั้นจะกระทำผ่าน Switch เลือกโหมด ซึ่งอยู่ด้านใต้กล่อง โดยการเลือกโหมดการทำงานนั้นจะต้องกระทำให้เสร็จเรียบร้อยก่อนการจ่ายไฟให้กับ ET-RF24G V1.0 ด้วยเสมอ เนื่องจากการทำงานของเครื่อง ET-RF24G V1.0 นั้นจะทำการตรวจสอบโหมดการทำงานของเครื่องจาก Switch เลือกโหมด เฉพาะในช่วงของการจ่ายไฟเลี้ยงให้เครื่องเริ่มต้นการทำงานครั้งแรก (Power-ON) เท่านั้น ซึ่งการเปลี่ยนแปลงตำแหน่งการทำงานของ Switch เลือกโหมด หลังจากทำการจ่ายไฟให้กับ ET-RF24G V1.0 ไปแล้ว จะไม่มีผลต่อการทำงานของเครื่องแต่อย่างใด โดยการทำงานของเครื่อง ET-RF24G V1.0 นั้นจะมี LED แสดงสถานะการทำงานของเครื่องจำนวน 2 หลอด คือ LED POWER ซึ่งเป็น LED สีแดง โดยที่ LED POWER นี้จะติดสว่างให้เห็นตลอดเวลาที่มีการจ่ายไฟเลี้ยงให้เครื่องทำงานอยู่ ส่วน LED อีกดวงหนึ่งนั้นจะเป็น LED สีเขียว ใช้แสดงสถานะการทำงานของเครื่องซึ่งเรียกว่า LED STATUS โดย LED STATUS นี้จะเกิดการกระพริบตามจังหวะของการรับส่งข้อมูลกันในแต่ละครั้งโดยในสภาวะปกตินั้น ถ้าเครื่องทำงานใน RUN MODE หลอด LED STATUS จะดับอยู่ตลอดเวลาถ้าไม่มีการรับส่งข้อมูล แต่ถ้าตัวเครื่องทำงานอยู่ใน SETUP MODE หลอด LED STATUS จะติดอยู่ตลอดเวลาถ้าไม่มีการรับส่งข้อมูล โดยโหมดการทำงานของ ET-RF24G V1.0 จะมีอยู่ด้วยกัน 2 โหมด คือ

2.3.3.1 การใช้งานเครื่อง ET-RF24G V1.0 ใน Run mode

การใช้งานใน Run Mode ซึ่งเป็นโหมดของการใช้งานตามปกติของเครื่อง โดยเมื่อเครื่อง ET-RF24G V1.0 เข้าทำงานในโหมดนี้แล้ว จะสังเกตเห็นหลอดไฟแสดงสถานะการทำงานของ หรือ LED STATUS ดับอยู่ แต่เมื่อการรับหรือส่งข้อมูลเกิดขึ้น สถานะการทำงานของ LED STATUS จึงจะกระพริบตามจังหวะของการรับส่งข้อมูลนั้นๆ แต่ถ้ายังไม่มีการรับส่งข้อมูลกัน LED STATUS จะดับอยู่ตลอดเวลา



ภาพที่ 2.54 แสดงการเลือกโหมดการทำงาน สำหรับใช้งานปกติ (Run Mode)

- การทำงานแบบ RF Receive Only

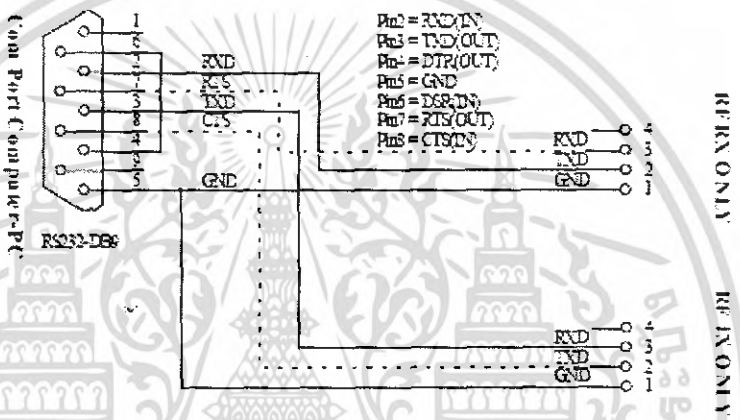
เป็นการทำงานแบบทิศทางเดียว โดยการทำงานในโหมดนี้ จะเป็นการรอรับข้อมูลความถี่แบบ GFSK จากด้าน RF แล้วเปลี่ยนเป็นข้อมูลอนุกรมส่งออกไปทางขา TX (Transmit) ของ RS232 โดยการทำงานจะวนรอบอยู่เช่นนี้ไปตลอด ซึ่งในการใช้งานเครื่อง ET-RF24G V1.0 ในโหมดนี้จะต้องนำสัญญาณ TX (Transmit) ไปต่อกับขาสัญญาณ RX (Receive) ของอุปกรณ์ด้านตรงข้าม (RS232 ของคอมพิวเตอร์ PC) โดยในโหมดนี้ การทำงานของขาสัญญาณ RX ด้าน RS232 ของเครื่อง ET-RF24G V1.0 จะถูกเปลี่ยนหน้าที่เป็นสัญญาณ CTS (Clear To Send) สำหรับใช้ตรวจสอบความพร้อมในการส่งข้อมูลไปให้อุปกรณ์ด้านตรงข้ามแทน ซึ่งในการใช้งานจะต้องนำสัญญาณนี้ไปต่อเข้ากับสัญญาณ RTS (Ready To Send) ของอุปกรณ์ด้านตรงข้าม โดยเครื่อง ET-RF24G V1.0 จะทำการตรวจสอบสถานะของสัญญาณ RX ซึ่งในโหมดนี้เปรียบเสมือน CTS ว่ามีค่าเป็น "0" หรือไม่ โดยถ้าพบว่าเป็น "0" จึงจะส่งข้อมูลออกไปให้ทางขา TX แต่ถ้าพบว่าสถานะของขาสัญญาณนี้มีค่าเป็น "1" แสดงว่าอุปกรณ์ด้านตรงข้ามยังไม่พร้อมรับข้อมูลก็จะรอจนกว่าจะพบว่าสถานะของสัญญาณดังกล่าวมีค่าเป็น "0" จึงจะส่งข้อมูลออกไปให้ โดยเครื่อง ET-RF24G V1.0 จะสามารถจัดเก็บข้อมูลไว้ใน Buffer เพื่อรอการส่งได้สูงสุด 64 Byte เท่านั้น ซึ่งถ้าในระหว่างที่รอความพร้อมอยู่นั้น มีข้อมูลด้าน RF ส่งเข้ามาเกินกว่า 64 Byte จะทำให้ข้อมูลที่เกิดมานั้นสูญหายไป

- การทำงานแบบ RF Transmit Only

เป็นการทำงานแบบทิศทางเดียว โดยการทำงานในโหมดนี้จะมีลักษณะตรงข้ามกับ RF Receive Only กล่าวคือ เครื่อง ET-RF24G V1.0 จะทำหน้าที่รอรับข้อมูลจากขา RX (Receive) ด้าน RS232 แล้วเปลี่ยนเป็นข้อมูลแบบ GFSK ส่งออกไปทางด้าน RF โดยการใช้งานเครื่องในโหมดนี้ จะต้องนำสัญญาณ TX (Transmit) ซึ่งเป็นขาส่งข้อมูลจาก RS232 ของอุปกรณ์ด้านตรงข้ามมาต่อเข้ากับขา RX (Receive) ของเครื่อง ET-RF24G V1.0 ส่วนขาสัญญาณ TX จะถูกเปลี่ยนหน้าที่เป็น RTS (Ready To Send) เพื่อใช้แสดงสถานะความพร้อมในการรับข้อมูลจากด้าน RS232 ซึ่งในการใช้งานจะต้องนำสัญญาณ TX ซึ่งในขณะนี้เปรียบเสมือนกับ RTS นำไปต่อเข้ากับสัญญาณ CTS (Clear To Send) ของอุปกรณ์ด้านตรงข้าม เพื่อใช้ในการตรวจสอบความพร้อมในการรับข้อมูล โดยอุปกรณ์ด้านตรงข้ามจะต้องทำการตรวจสอบสถานะของสัญญาณ RTS นี้ เพื่อตรวจสอบความพร้อมในการรับข้อมูลของเครื่อง ET-RF24G V1.0 ด้วย โดยถ้าเครื่อง ET-RF24G V1.0 พร้อมรับข้อมูลจาก RS232 มันจะส่งสัญญาณ RTS ให้มีค่าเป็น "0" รอไว้ และเมื่อใดก็ตามที่การรับข้อมูลทางด้านของ RS232 มีจำนวนข้อมูลที่ยังไม่สามารถเปลี่ยนเป็น GFSK เพื่อส่งออกไปทางด้าน RF ได้ทันจนเกือบจะเต็ม Buffer แล้ว เครื่อง ET-RF24G V1.0 จะทำการส่งสัญญาณ RST ให้มีค่าเป็น "1" ออกไปบอกให้อุปกรณ์ด้านตรงข้ามทราบเพื่อจะได้หยุดการส่งข้อมูลออกมา โดยอุปกรณ์ตรงข้ามจะต้องหยุดการส่งข้อมูลและรอจนกว่าสถานะของสัญญาณ RTS จะกลับเป็น "0" จึงจะเริ่มต้น

ส่งข้อมูลออกมาใหม่ ซึ่งหลังจากที่เครื่อง ET-RF24G V1.0 ส่งสัญญาณ RTS ด้วยค่า "1" ออกไปแล้ว จะยังคงสามารถรับข้อมูลได้เพิ่มเติมอีกไม่เกิน 16 Byte เท่านั้น ซึ่งถ้าอุปกรณ์ด้านตรงข้ามยังส่งข้อมูลต่อเนื่องมาอีกจนเกินขนาดของ Buffer ที่เครื่อง ET-RF24G V1.0 จะรับไว้ได้จะทำให้ข้อมูลที่เกินมานั้นเกิดการสูญหายได้

โดยเราสามารถนำเครื่อง ET-RF24G V1.0 จำนวน 4 ชุดมาต่อใช้งานร่วมกัน เพื่อใช้งานในการรับส่งข้อมูลกันแบบ Full Duplex โดยแบ่งการใช้งานออกเป็น 2 ด้าน คือ ต้นทาง และ ปลายทาง ด้านละ 2 ชุด โดยแต่ละด้านให้กำหนดหน้าที่การทำงานเป็น RF Receive Only 1 ชุด และ RF Transmit Only อีก 1 ชุด



ภาพที่ 2.55 แสดงสายสัญญาณ RS232 เพื่อใช้กับ ET-RF24G ในโหมด RF Receive Only และ RF Transmit Only

- การทำงานแบบ RF Auto Direction

เป็นการทำงานชนิด 2 ทิศทาง แบบ Half Duplex หรือ ผลัดกันรับผลัดกันส่ง ซึ่งสามารถใช้รับส่งข้อมูลระหว่างต้นทาง และ ปลายทางได้โดยใช้เครื่อง ET-RF24G V1.0 ด้านละ 1 ชุด เท่านั้น เพียงแต่การรับส่งข้อมูลแบบนี้จะไม่สามารถส่งข้อมูลสวนทางกันได้เหมือนกับแบบ Full Duplex แต่จะต้องใช้วิธีการผลัดกันรับข้อมูลและส่งข้อมูลแทน โดยเมื่อฝ่ายรับทำการรับข้อมูลได้จนครบแล้วจึงจะสลับหน้าที่เป็นฝ่ายส่งเพื่อส่งข้อมูลย้อนกลับไป

โดยในโหมดนี้ เครื่อง ET-RF24G V1.0 จะทำหน้าที่เป็นทั้ง ฝ่ายรับ และฝ่ายส่งข้อมูล แบบอัตโนมัติ โดยในสภาวะปกติจะอยู่ในสภาวะของการรอรับข้อมูล ทั้งด้าน RF และ RS232 ซึ่งถ้าพบว่ามีข้อมูลส่งเข้ามาทางด้านของ RF ก็จะนำข้อมูลนั้นส่งออกไปทางด้านขา TX ของ RS232 ทันที และในทำนองเดียวกัน ถ้าพบว่ามีข้อมูลส่งเข้ามาทางด้าน RX ของ RS232 มันก็จะทำการรับข้อมูลนั้นจาก RS232 พร้อมกับเปลี่ยนทิศทางของอุปกรณ์ RF จากการรอรับข้อมูลให้ทำหน้าที่เป็น

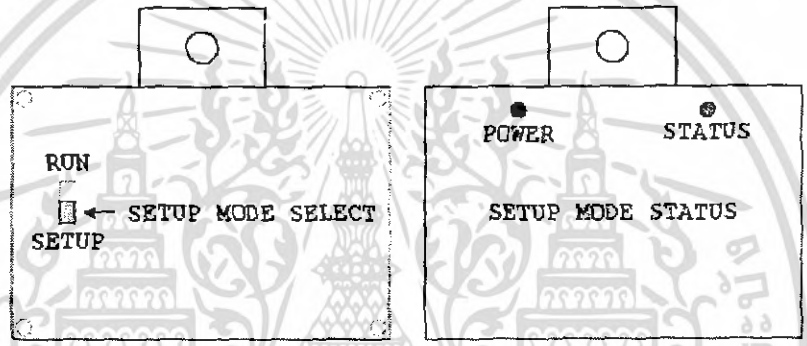
ตัวส่งข้อมูลแทน เพื่อทำการส่งข้อมูลที่รับได้จาก RS232 ออกไปทาง RF ในทันที ซึ่งหลังจากที่เครื่อง ET-RF24G V1.0 ทำการสลับโหมดการทำงานของอุปกรณ์ด้าน RF จากการรอรับเป็นการส่ง และทำการเริ่มต้นส่งข้อมูลออกไปทางด้าน RF เรียบร้อยแล้ว มันจะวนกลับไปตรวจสอบการรับข้อมูลจากด้าน RS232 อีกว่ายังมีข้อมูลส่งเข้ามาอีกหรือไม่ ถ้าพบว่ายังมีข้อมูลส่งเข้ามีอีกก็จะทำการแปลงข้อมูลนั้นเพื่อส่งออกไปยังด้าน RF ต่อไปอีกจนกว่าการส่งข้อมูลด้าน RS232 จะสิ้นสุดลง ซึ่งข้อมูลด้าน RS232 ที่ส่งเข้ามานั้น ควรส่งอย่างต่อเนื่อง โดยเมื่อเครื่อง ET-RF24G V1.0 ทำการส่งข้อมูลแต่ละ Byte ออกไปทางด้าน RF เรียบร้อยแล้วมันจะวนรอบรอรับข้อมูล Byte ถัดไปจาก RS232 ภายในเวลา 2.5 mS ถ้าไม่พบข้อมูลส่งเข้ามาอีกภายในระยะเวลาดังกล่าวมันจึงจะทำการเปลี่ยนหน้าที่ของอุปกรณ์ด้าน RF ให้กลับมาทำหน้าที่เป็นการรอรับข้อมูลตามเดิม โดยในขณะที่อุปกรณ์ด้าน RF ถูกกำหนดให้เป็นฝ่ายส่งข้อมูลอยู่นั้น จะไม่สามารถทำการรับข้อมูลจาก RF ได้ ซึ่งถ้ามีการส่งข้อมูลเข้ามาในขณะนั้นก็จะไม่สามารถรับได้ โดยค่าเวลาที่จะใช้ในการสลับโหมดการทำงานของ RF จากฝ่ายส่งข้อมูลให้เป็นฝ่ายรับข้อมูลนั้น จะมีค่าเป็น 2.5 mS ดังนั้นเมื่อฝ่ายรับสามารถรับข้อมูลได้ครบหมดแล้วก่อนที่จะทำการส่งข้อมูลเพื่อสอบกลับไปยังฝ่ายตรงข้ามนั้น ควรทำการหน่วงเวลาไว้ไม่น้อยกว่า 3 mS นับจากรับข้อมูล Byte สุดท้ายได้เรียบร้อยแล้วจึงเริ่มต้นส่งข้อมูล Byte แรกย้อนกลับไป ซึ่งถ้าฝ่ายรับทำการส่งข้อมูลตอบกลับไปยังฝ่ายตรงข้ามเร็วกว่านี้อาจทำให้ฝ่ายตรงข้ามไม่สามารถรับข้อมูล Byte แรกได้ทัน

สำหรับการใช้งานเครื่อง ET-RF24G V1.0 ในโหมด RF Auto Direction นี้ การรับและส่งข้อมูลด้าน RS232 จะไม่มีการตรวจสอบความพร้อมของฝ่ายรับและส่งด้วยสัญญาณทางไฟฟ้า (CTS/RTS) เหมือนกับการใช้งานใน 2 โหมดที่ผ่านมาแล้ว โดยเมื่อมันสามารถรับข้อมูลจาก RF ได้ก็จะทำการส่งข้อมูลนั้นออกไปทางขา TX (Transmit) ของ RS232 ในทันที โดยไม่สนใจว่าอุปกรณ์ที่ต่อไว้ด้าน RS232 จะพร้อมรับข้อมูลหรือไม่ ซึ่งถ้าด้าน RS232 ไม่พร้อมรับข้อมูลก็จะทำให้ข้อมูล Byte นั้นสูญหายไปทันที ซึ่งในการใช้งานนั้น ผู้ใช้ควรกำหนดค่าความเร็วในการรับส่งข้อมูลด้าน RS232 ที่จะใช้กับเครื่อง ET-RF24G V1.0 ทุกๆตัวด้วยค่าความเร็วที่เท่ากันด้วย เพื่อให้การรับและส่งข้อมูลเกิดความสัมพันธ์กันอย่างเหมาะสม

สำหรับความสามารถในการรอรับข้อมูลจาก RS232 ของเครื่อง ET-RF24G V1.0 ในโหมดนี้ จะสามารถรับข้อมูลได้อย่างต่อเนื่องสูงสุด ไม่เกิน 64 Byte ดังนั้นในกรณีที่มีการส่งข้อมูลจากด้าน RS232 ด้วยข้อมูลจำนวนมากกว่า 64 Byte ต่อเนื่องกันนั้น ควรทำการแบ่งข้อมูลออกเป็นชุดๆ โดยให้มีขนาดชุดละไม่เกิน 64 Byte ซึ่งหลังจากทำการส่งข้อมูลอย่างต่อเนื่องไปได้ 1 ชุด (64 Byte) แล้วควรทำการหน่วงเวลาไว้ชั่วขณะหนึ่งอย่างน้อย 1 mS แล้วจึงเริ่มส่งข้อมูลชุดถัดไปสลับกับการหน่วงเวลา อย่างนี้เรื่อยๆ เพื่อให้เครื่อง ET-RF24G V1.0 สามารถนำข้อมูลที่รับได้จากด้าน RS232 ส่งออกไปทางด้าน RF ได้ทัน ซึ่งถ้าทำการส่งข้อมูลอย่างต่อเนื่องโดยไม่มีการหน่วงเวลาเลยอาจทำให้ข้อมูลบาง Byte เกิดการสูญหายไปได้

2.3.3.2 การใช้งานเครื่อง ET-RF24G V1.0 ใน Setup mode

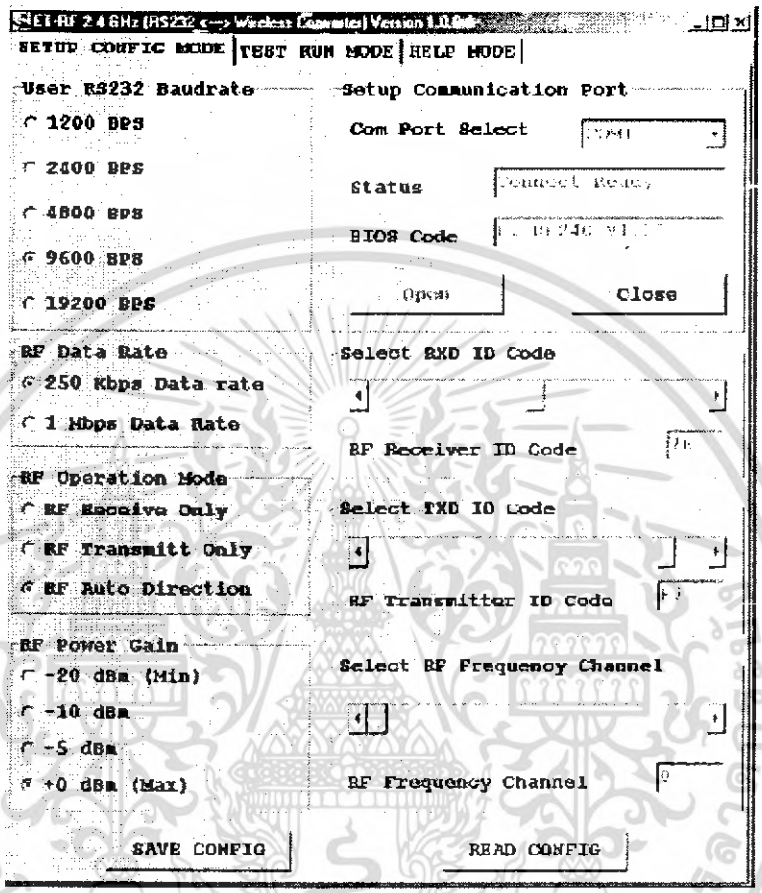
การใช้งานเครื่อง ET-RF24G V1.0 ใน Setup Mode ซึ่งเป็นโหมดสำหรับใช้กำหนดค่า Configuration ต่างๆ สำหรับควบคุมการทำงานของเครื่อง ET-RF24G V1.0 ที่จะใช้ในขณะที่เครื่องทำงานอยู่ใน Run Mode โดยในการ Setup ค่า Configuration ต่างๆนั้นจะกระทำร่วมกับโปรแกรม "ET_RF24G_V1.EXE" ของ อีทีที ซึ่งเมื่อเครื่อง ET-RF24G V1.0 เข้าทำงานในโหมด Setup แล้ว จะสังเกตเห็นหลอดไฟแสดงสถานะการทำงาน หรือ LED STATUS ติดสว่างค้างอยู่ตลอดเวลา แต่เมื่อมีการสั่งอ่านหรือเขียนข้อมูลกับบอร์ด สถานะการทำงานของ LED STATUS จึงจะกระพริบตามจังหวะของการรับส่งข้อมูล แต่ถ้ายังไม่มีการรับส่งข้อมูล LED STATUS จะติดค้างอยู่ตลอดเวลา



ภาพที่ 2.56 แสดงการเลือกโหมดการทำงาน สำหรับกำหนดค่า Configuration (Setup Mode)

ซึ่งการกำหนดค่า Configurationให้กับ ET-RF24G V1.0 นั้นจะต้องกระทำในขณะที่ตัวเครื่องทำงานอยู่ใน Setup Mode เท่านั้น (เลือก Switch กำหนดโหมดไว้ทางด้าน Setup แล้วจ่ายไฟให้เครื่องเริ่มต้นทำงาน) โดยค่าของ Configuration ต่างๆนั้นจะถูกใช้สำหรับเป็นเงื่อนไขในการทำงานของ ET-RF24G V1.0 ในขณะที่อยู่ใน Run Mode ดังนั้น ก่อนการเริ่มต้นใช้งานเครื่องในครั้งแรกนั้น จึงจำเป็นอย่างยิ่งที่จะต้องทำการกำหนดค่าของ Configuration ต่างๆให้ถูกต้องและตรงกับความต้องการที่จะใช้งานเสียก่อน โดยเมื่อทำการกำหนดค่าตัวเลือกต่างๆของ Configuration เรียบร้อยแล้ว ก็สามารถเปลี่ยนโหมดการทำงานของตัวเครื่องกลับเป็น Run Mode พร้อมกับการปิดไฟที่จ่ายให้กับตัวเครื่อง (Power-OFF) ชั่วขณะหนึ่ง จากนั้นจึงเริ่มต้นจ่ายไฟให้กับตัวเครื่องใหม่ (Power-ON) ก็สามารถใช้งาน ET-RF24G V1.0 ตามค่าของ Configuration ที่กำหนดไว้แล้วได้ทันที โดยค่าตัวเลือกต่างๆของ Configuration ที่กำหนดไว้แล้วจะถูกเก็บไว้ภายในตัวเครื่องอย่างถาวร ถึงแม้ว่าจะไม่ได้ทำการจ่ายไฟให้กับตัวเครื่องแล้วก็ตาม ดังนั้นเมื่อทำการกำหนดค่า Configuration ต่างๆเรียบร้อยแล้ว ถ้าไม่มีการเปลี่ยนแปลงเงื่อนไขการทำงานของตัวเครื่องต่างไปจากเงื่อนไขเดิมที่ได้กำหนดไว้แล้ว ก็ไม่จำเป็นต้องทำการกำหนดค่า Configuration ใหม่อีกแต่อย่างใด โดยทุกๆ

ครั้งที่เริ่มต้นจ่ายไฟเข้าเครื่องในครั้งแรกนั้น การทำงานของ ET-RF24G V1.0 จะเป็นไปตามเงื่อนไขที่กำหนดไว้ใน Configuration เสมอทุกๆครั้ง โดยคุณสมบัติของ Configuration ต่างๆนั้นมีดังนี้



ภาพที่ 2.57 แสดงโปรแกรมที่ใช้สำหรับกำหนดค่า Configuration ของ ET-RF24G V1.0

- User RS232 Baudrate ใช้สำหรับกำหนดค่าความเร็วในการรับส่งข้อมูลทางด้าน RS232 ของตัวเครื่อง ในขณะที่ทำงานอยู่ใน Run Mode ซึ่งสามารถกำหนดได้ 5 ค่าคือ

1. 1200 BPS
2. 2400 BPS
3. 4800 BPS
4. 9600 BPS
5. 19200 BPS

- RF Data Rate ใช้สำหรับกำหนดความเร็วในการรับส่งข้อมูลทางด้าน RF ของET-RF24G V1.0 ซึ่งจะต้องกำหนดให้เครื่อง ET-RF24G V1.0 ทุกๆตัว ที่จะนำมาใช้ติดต่อสื่อสารกัน มีค่าอัตราความเร็วในการรับส่งข้อมูลด้าน RF หรือ RF Data Rate นี้มีค่าเท่ากันทั้งหมด ซึ่งถ้ากำหนดค่า

กำหนดค่าความเร็วต่างกันจะไม่สามารถรับส่งข้อมูลกันได้ ซึ่งค่าอัตราความเร็วในการส่งข้อมูลนี้จะมีผลต่อระยะทางการรับส่งข้อมูลด้วย ซึ่งถ้าใช้ความเร็วในการส่ง (1 Mbps) จะทำให้รัศมีการรับส่งข้อมูลได้ระยะทางสั้นลง แต่ถ้าใช้ความเร็วในการรับส่งข้อมูลที่ช้าลง (250 Kbps) จะทำให้รัศมีการรับส่งไกลขึ้น โดยค่า RF Data Rate สามารถกำหนดได้ 2 ค่า คือ

1. 250 Kbps
2. 1 Mbps

- RF Operation Mode ใช้สำหรับกำหนดการทำงานของ ET-RF24G V1.0 ซึ่งสามารถกำหนดหน้าที่การทำงานได้ 3 แบบ ด้วยกันคือ

1. RF Receive Only เป็นการกำหนดให้ ET-RF24G V1.0 ทำหน้าที่เป็นฝ่ายรอรับข้อมูลทางด้าน RF เพื่อเปลี่ยนเป็นข้อมูลแบบ RS232 และส่งออกไปทางด้านขา TX ของ RS232 ตลอดเวลา

2. RF Transmit Only เป็นการกำหนดให้ ET-RF24G V1.0 ทำหน้าที่เป็นฝ่ายรอรับข้อมูลทางด้าน RS232 จากขา RX เพื่อเปลี่ยนเป็นข้อมูลแบบ GFSK และส่งออกไปทางด้าน RF ตลอดเวลา

3. RF Auto Direction เป็นการกำหนดโหมดการทำงานแบบ Half Duplex 2 ทิศทาง ซึ่งสามารถสลับโหมดการทำงานระหว่างการรับและส่งข้อมูลได้เองอัตโนมัติ โดยในโหมดการทำงานนี้เครื่อง ET-RF24G V1.0 จะรอตรวจสอบข้อมูลทั้งจากด้าน RS-232 และด้าน RF อยู่ตลอดเวลา โดยถ้าได้รับข้อมูลจาก ด้าน RF กลับมาเป็นฝ่ายรอรับข้อมูลตามเดิม และเมื่อได้รับข้อมูลจากด้าน RF ก็จะกำหนดให้ด้าน RF กลับมาเป็นฝ่ายรอรับข้อมูลตามเดิม และเมื่อได้รับข้อมูลจากด้าน RF ก็จะแปลงเป็นข้อมูลแล้วส่งออกไปทางด้าน RS-232 โดยอัตโนมัติ

- RF Power Gain เป็นการกำหนดกำลังส่งของวงจร RF Power ที่ใช้ในการส่งข้อมูล โดยค่า +0dBm เป็นค่ากำลังส่งสูงสุด ส่วน –20dBm เป็นค่ากำลังส่งต่ำสุด โดยสามารถกำหนดได้ 4 ระดับ คือ

1. -20dBm (กำลังส่งต่ำสุด)
2. -10dBm
3. -5dBm
4. +0dBm (กำลังส่งสูงสุด)

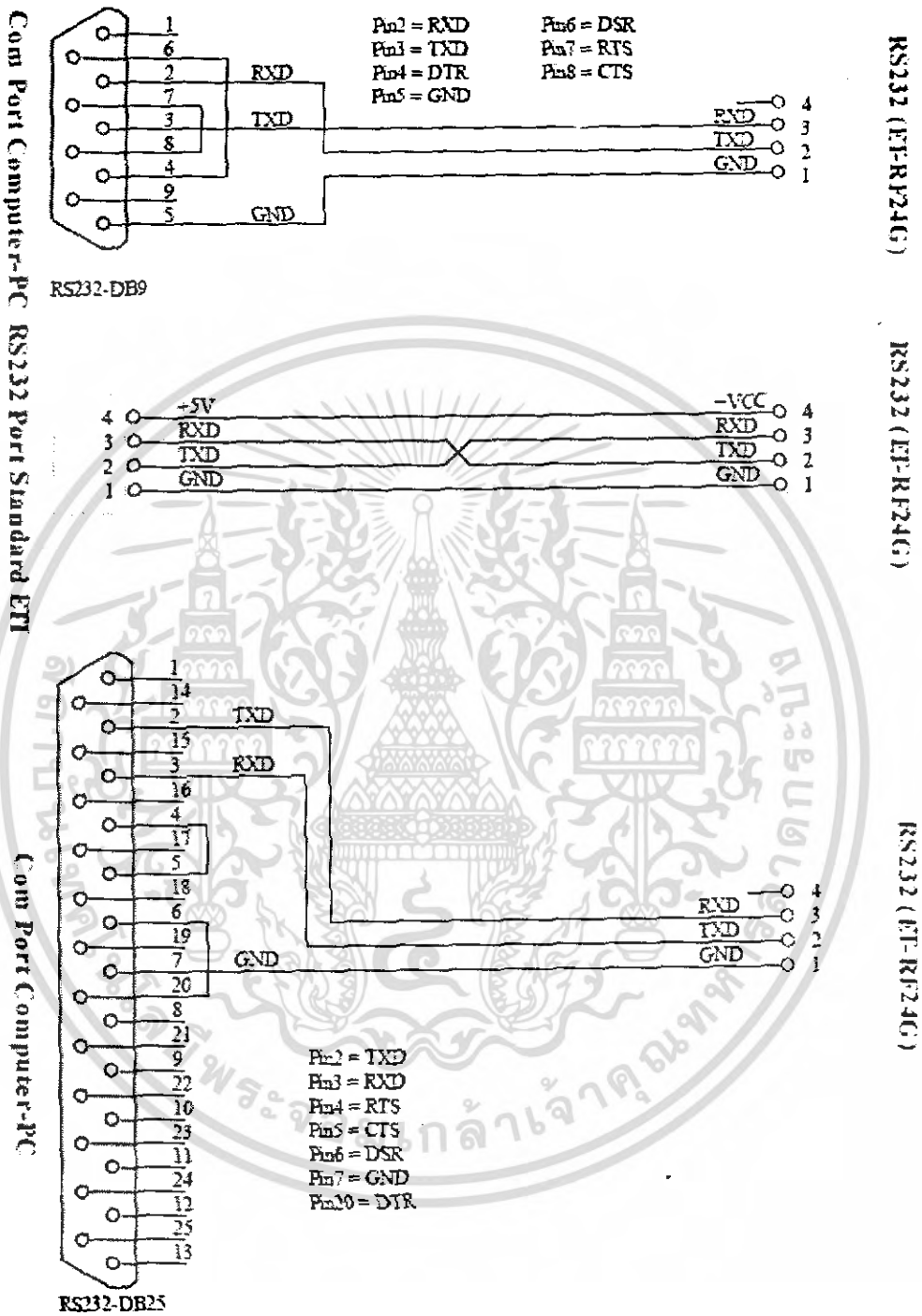
- RXD ID Code เป็นรหัส ID Code ของเครื่อง ET-RF24G V1.0 ในโหมดของการรับข้อมูลจาก RF โดยเมื่อเครื่อง ET-RF24G V1.0 ด้านส่งจะทำการส่งข้อมูลออกไปทาง RF นั้นจะมีการระบุหมายเลข ID Code ของด้านรับรวมไปกับชุดข้อมูลด้วยเสมอ โดยเมื่อเครื่อง ET-RF24G V1.0 ที่อยู่ทางด้านรับทำการรับข้อมูลจากด้าน RF ได้ อันดับแรกมันจะทำการเปรียบเทียบรหัส ID Code ที่รวมมากับข้อมูลที่รับมาได้ว่าตรงกับรหัสของ RXD ID Code ที่กำหนดไว้ในตัวมันหรือไม่ ซึ่งถ้า

ถูกต้องก็จะแยกเอาเฉพาะส่วนของข้อมูลที่รับเข้ามาได้เพื่อเปลี่ยนเป็นข้อมูลแบบ RS232 แล้วส่งออกไปทางด้าน TX ของ RS232 แต่ถ้ารหัส ID Code ที่รับมาได้ไม่ตรงกับรหัส RXD ID Code ที่กำหนดไว้ เครื่อง ET-RF24G V1.0 จะทิ้งข้อมูลชุดนั้นไปทันที โดยค่า RXD ID Code นั้นสามารถกำหนดได้ 256 ค่าในรูปแบบของเลขฐานสิบหก (00H-FFH)

- TXD ID Code เป็นรหัส ID Code ปลายทางที่จะส่งข้อมูลไปหา โดยที่เครื่อง ET-RF24G V1.0 ที่ถูกกำหนดให้ทำหน้าที่เป็นฝ่ายส่งข้อมูล เมื่อมันสามารถรับข้อมูลจาก RS232 ได้แล้ว มันจะทำการนำเอาข้อมูลนั้นไปเข้ารหัสรวมกับ TXD ID Code ที่กำหนดไว้ แล้วส่งออกไปทางด้าน RF โดยรหัสของ TXD ID Code นี้หมายถึง รหัส RXD ID Code ของฝ่ายรับที่ต้องการส่งข้อมูลไปหานั่นเอง โดยค่า TXD ID Code นั้นสามารถกำหนดได้ 256 ค่าในรูปแบบของเลขฐานสิบหก (00H-FFH)

- RF Frequency Channel เป็นการกำหนดค่าช่วงความถี่ที่จะใช้ในการรับส่งข้อมูล โดยสามารถเลือกกำหนดช่องความถี่ได้สูงสุดมากถึง 125 ช่อง (0-124) โดยการที่เครื่อง ET-RF24G V1.0 จะทำการรับส่งข้อมูลกันได้นั้นจะต้องกำหนดช่องความถี่ที่ตรงกัน และใช้อัตราความเร็ว RF Data Rate ที่เท่ากันด้วย ซึ่งที่สามารถเลือกกำหนดช่องความถี่ RF Frequency Channel ได้นั้น จะมีประโยชน์เป็นอย่างมากในกรณีที่มีการใช้งานเครื่อง ET-RF24G V1.0 จำนวนหลายๆกลุ่ม ในบริเวณพื้นที่ใกล้เคียงกัน โดยให้กำหนดช่องความถี่ของ ET-RF24G V1.0 กลุ่มที่จะสื่อสารข้อมูลร่วมกันไว้ที่ช่องความถี่เดียวกัน ส่วนกลุ่มอื่นๆก็ให้เลือกกำหนดช่องความถี่ที่แตกต่างกันออกไปเพื่อลดปัญหาการรบกวนกัน

2.3.4 การเชื่อมต่อสัญญาณ RS232



ภาพที่ 2.58 แสดงการต่อสาย RS232 เพื่อใช้งานกับ ET-RF24G V1.0 ในโหมด Auto Direction

2.3.5 ข้อแนะนำในการกำหนดค่า Configuration

การกำหนดค่า Configuration ให้กับเครื่อง ET-RF24G V1.0 นั้น สามารถเลือกกำหนดได้ตามความต้องการและจุดประสงค์ของการใช้งาน โดยแต่ละโหมดของการใช้งานนั้นจะมีค่า Configuration ที่เหมาะสมต่างกันซึ่งขอแนะนำวิธีการกำหนดค่า Configuration ดังแนวทางต่อไปนี้

- ความเร็วในการรับส่งข้อมูลด้าน RS232 หรือ User RS232 Baudrate ที่ความเร็ว 19200 Bps นั้นเหมาะกับการใช้งาน ET-RF24G V1.0 แบบ Receive Only หรือ Transmit Only ซึ่งมีการตรวจสอบความพร้อมของสัญญาณในการรับส่งข้อมูลกันด้วย แต่ถ้าต้องการใช้งานเครื่อง ET-RF24G V1.0 ในโหมด Auto Direction นั้น ควรกำหนดค่า User RS232 Baudrate ไว้ที่ความเร็วไม่เกิน 9600 Bps จะดีที่สุด และควรกำหนดค่า Baudrate ของทั้งสองฝ่ายให้มีค่าเท่ากันด้วย

- ค่าความเร็วของการรับส่งข้อมูลด้าน RF หรือ RF Data Rate ที่สามารถรับส่งข้อมูลกันได้ระยะทางไกลมากที่สุดและ มีโอกาสผิดพลาดน้อยที่สุด คือ 250 Kbps

- ค่า RF Power Gain ที่ดีที่สุดคือ 0dBm ซึ่งเป็นค่ากำลังสูงสุด ซึ่งจะทำให้สามารถส่งข้อมูลได้ระยะทางไกลที่สุด แต่ถ้าระยะการรับส่งข้อมูลไม่ไกลกันมาก และมีการใช้งานเครื่อง ET-RF24G V1.0 จำนวนหลายๆกลุ่มในพื้นที่ใกล้เคียงกัน ก็อาจทำการลดกำลังส่งให้ต่ำลงเพื่อลดปัญหาการรบกวนกันหรือกำหนดช่องความถี่ RF Frequency Channel ให้ห่างกันมากๆ

- ในกรณีที่มีการใช้เครื่อง ET-RF24G V1.0 หลายๆกลุ่มในพื้นที่ใกล้เคียงกัน ควรกำหนดช่องความถี่ในการใช้งาน หรือ RF Frequency Channel ให้ห่างกันด้วยเพื่อป้องกันการรบกวนกัน

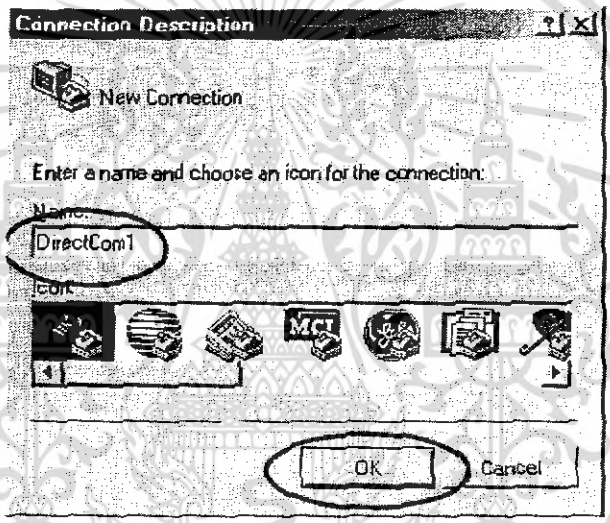
- การใช้งานเครื่อง ET-RF24G V1.0 แบบ Auto Direction นั้น ถ้ามีการส่งข้อมูลจำนวนมากๆ ควรจัดแบ่งข้อมูลออกเป็นชุดๆ โดยให้มีขนาดข้อมูลชุดละไม่เกิน 64 Byte โดยในการส่งข้อมูลแต่ละชุดนั้นให้ทำการส่งข้อมูลอย่างต่อเนื่องโดยให้ข้อมูลแต่ละ Byte มีระยะเวลาห่างกันไม่เกิน 2.5 mS เนื่องจากถ้าข้อมูลขาดหายไปนานกว่านี้ เครื่อง ET-RF24G V1.0 จะทำการเปลี่ยนโหมดของการส่งข้อมูลกลับเป็นโหมดของการรับส่งข้อมูลแทน ซึ่งเมื่อมีการส่งข้อมูล Byte ถัดไปมาอีกก็จะต้องเสียเวลาในการสลับโหมดจากฝ่ายรอรับข้อมูลให้เป็นฝ่ายส่งข้อมูลอีก ซึ่งจะทำให้ประสิทธิภาพในการจัดส่งข้อมูลลดลงเนื่องจากต้องเสียเวลาในการสลับโหมดการทำงานของวงจรภาค RF อยู่ตลอดเวลา โดยที่เมื่อทำการจัดส่งข้อมูลครบ 64 Byte แล้ว ให้ทำการหน่วงเวลาไว้ชั่วขณะหนึ่ง ประมาณ 1mS-2mS แล้วจึงส่งข้อมูลชุดถัดไปอีกอย่างนี้เรื่อยๆ จะทำให้การรับส่งข้อมูลมีประสิทธิภาพสูงสุด

- การใช้งานเครื่อง ET-RF24G V1.0 แบบ Auto Direction นั้นควรหน่วงเวลาในการสลับโหมดจากฝ่ายของการรอรับข้อมูลเป็นฝ่ายส่งข้อมูล อย่างน้อยที่สุด 3mS-5mS ซึ่งถ้าส่งข้อมูลย้อนกลับด้วยเวลาที่เร็วกว่านี้อาจทำให้ฝ่ายตรงข้ามไม่สามารถรับข้อมูล Byte แรกได้ทัน

2.3.6 ตัวอย่างการใช้งาน

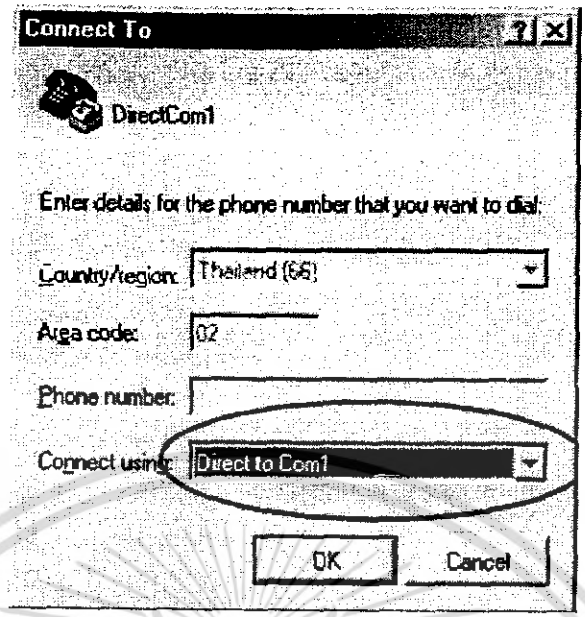
สำหรับตัวอย่างการใช้งานนั้น จะขอแสดงให้เห็นโดยใช้คอมพิวเตอร์ PC เป็นอุปกรณ์การทดลอง โดยในที่นี้จะขอเลือกใช้โปรแกรมสำเร็จรูปสำหรับใช้ในการสื่อสารของ Windows ซึ่งก็คือ Hyper Terminal โดยใน 2 ตัวอย่าง แรกนั้นจะใช้งานกับเครื่อง ET-RF24G V1.0 ในโหมด Auto Direction ซึ่งมีวิธีการใช้งานดังต่อไปนี้

1. เรียกใช้โปรแกรม Hyper Terminal ของ Windows โดยเรียกจาก Start →Programs→ Accessories→Communication→Hyper Terminal ซึ่งจะได้ผลดังภาพที่ 2.59



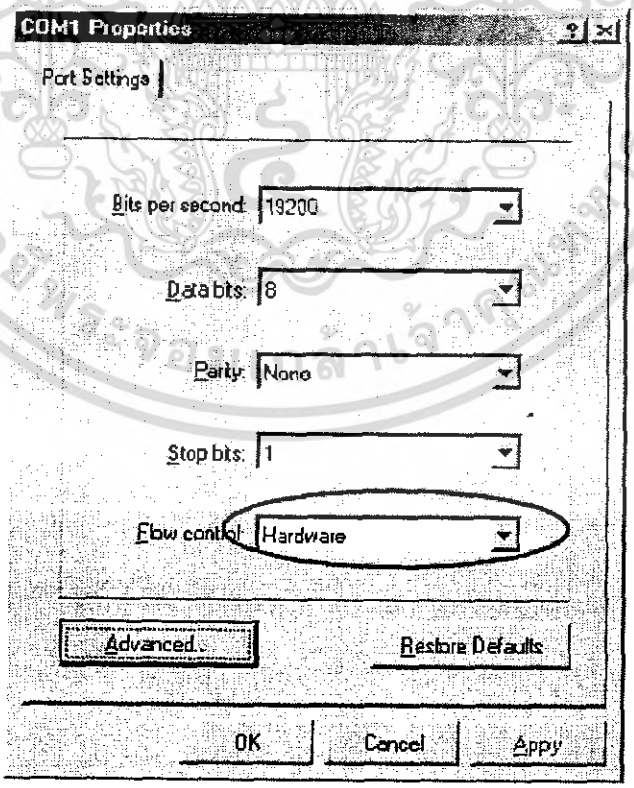
ภาพที่ 2.59 แสดงการใช้โปรแกรม Hyper Terminal

2. ให้เลือกกำหนดชื่อสำหรับใช้ในการเชื่อมต่อ ซึ่งสามารถกำหนดได้เองตามต้องการ โดยในตัวอย่างจะกำหนดเป็น DirectCom1 จากนั้นให้เลือก OK เพื่อข้ามไปยังขั้นตอนถัดไป



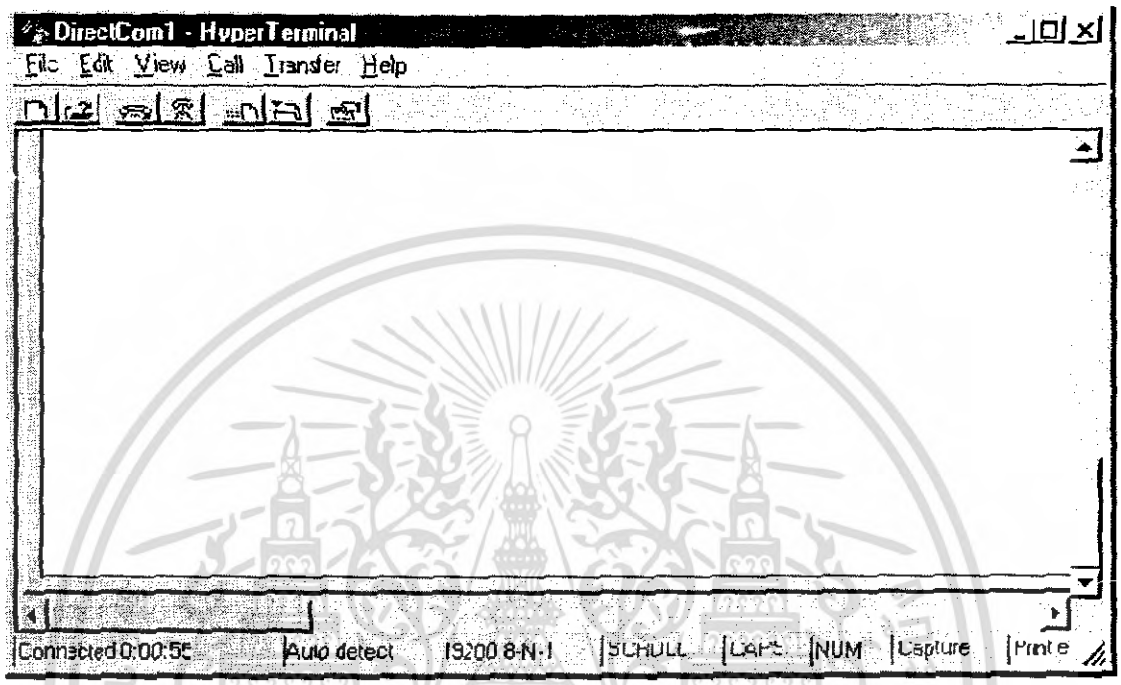
ภาพที่ 2.60 แสดงการกำหนดการเชื่อมต่อเป็น Direct to Com1

3. ให้เลือกกำหนดการเชื่อมต่อเป็น Direct to Com1 ซึ่งถ้าเครื่องคอมพิวเตอร์ที่ใช้เป็น Comport อื่นที่ไม่ใช่ Com1 ก็ให้เลือกให้ตรงกับความเป็นจริง จากนั้นเลือก OK เพื่อข้ามไปยังขั้นตอนถัดไป



ภาพที่ 2.61 แสดงการกำหนดกำหนดคุณสมบัติของพอร์ตอนุกรม RS232

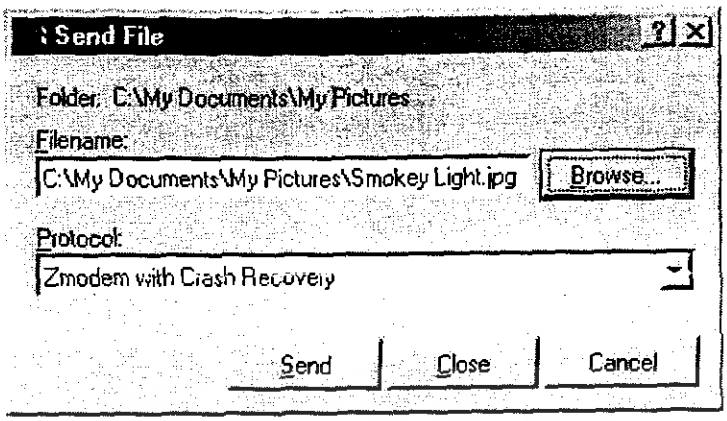
4. ในขั้นตอนนี้ จะใช้สำหรับกำหนดคุณสมบัติของพอร์ตอนุกรม RS232 โดยให้เลือก Bit per second = 9600 , Data Bit = 8 , Parity = None , Stop Bit = 1 ส่วน Flow Control ให้เลือกเป็น None จากนั้นเลือก OK ซึ่งจะเข้าสู่หน้าต่างโปรแกรมหลักของ Hyper Terminal ดังภาพที่ 2.62



ภาพที่ 2.62 แสดง Hyper Terminal

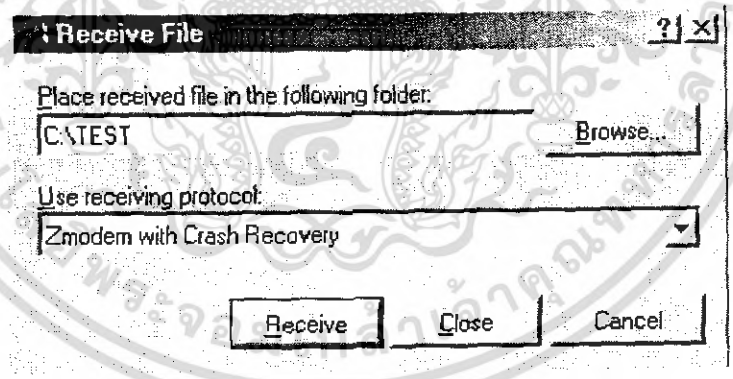
5. ในขั้นตอนนี้สามารถทำการรับส่งข้อมูลระหว่างทั้ง 2 ฝ่ายได้แล้ว ซึ่งสามารถทดสอบได้โดยการกดคีย์ใดๆจากฝ่ายหนึ่ง ซึ่งตัวอักขระบนคีย์นั้นๆจะถูกส่งไปแสดงผลยังฝั่งตรงข้ามทันที แต่ในที่นี้เราจะทำการทดสอบการรับและส่งไฟล์ โดยใช้ Protocol สำเร็จรูปของ Hyper Terminal ซึ่งมีให้เลือกใช้มากมายหลาย Protocol โดยต้องกำหนด Protocol ให้ตรงกันทั้งฝ่ายรับและฝ่ายส่ง ซึ่งในขั้นตอนของการทดสอบนั้นต้องกำหนดให้ฝ่ายหนึ่งเป็นฝ่ายรับและให้อีกฝ่ายหนึ่งเป็นฝ่ายส่ง ซึ่งในที่นี้จะเลือกใช้ Protocol ของ Zmodem with Crash Recovery ซึ่งมีวิธีการทดสอบการรับส่งข้อมูลดังนี้

ทางด้านฝ่ายส่งให้ทำการเลือกกำหนดไฟล์ที่จะส่งจากเมนูคำสั่ง Transfer→Send File... จากนั้นให้เลือกกำหนดชื่อและที่อยู่ของไฟล์ที่ต้องการจะส่ง โดยคลิกเมาส์ที่ปุ่ม Browse พร้อมกับกำหนดชื่อและที่อยู่ของไฟล์ตามต้องการ จากนั้นให้เลือกกำหนด Protocol ของการรับส่งข้อมูลเป็น Zmodem with Crash Recovery แล้วเลือกคลิกเมาส์ที่ปุ่ม Send เพื่อทำการเริ่มต้นส่งข้อมูลดังภาพ



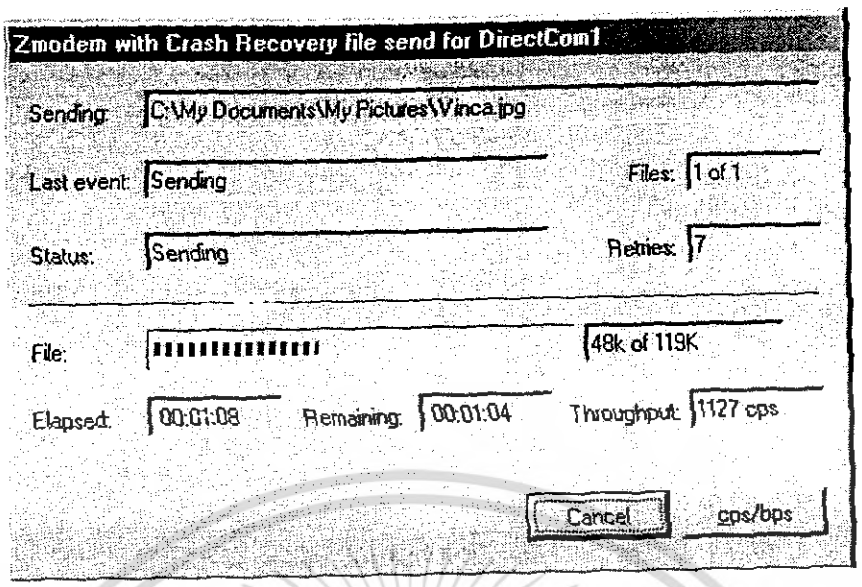
ภาพที่ 2.63 แสดงการ Send File

สำหรับในด้านที่เป็นฝ่ายรับข้อมูลนั้นก็ให้เลือกกำหนดการทำงานให้เป็นฝ่ายรับ โดยกำหนดจากเมนูคำสั่งของ Transfer→Receive File... จากนั้นให้เลือกกำหนดตำแหน่งของ Folder สำหรับใช้บันทึกไฟล์ที่รับได้จากฝ่ายส่ง โดยการเลือกจากปุ่ม Browse แล้วเลือกกำหนด Folder ที่ต้องการ ส่วนชื่อนั้นไม่ต้องกำหนด โดยโปรแกรม Hyper Terminal จะตั้งให้เองตามชื่อไฟล์จริงที่ส่งมา และในส่วนของ Protocol ที่ใช้นั้นก็ต้องกำหนดให้ตรงกับทางด้านส่ง คือ Zmodem with Crash Recovery จากนั้นให้เลือก Receive เพื่อให้โปรแกรมรอรับไฟล์จากด้านส่ง



ภาพที่ 2.64 แสดงการ Receive File

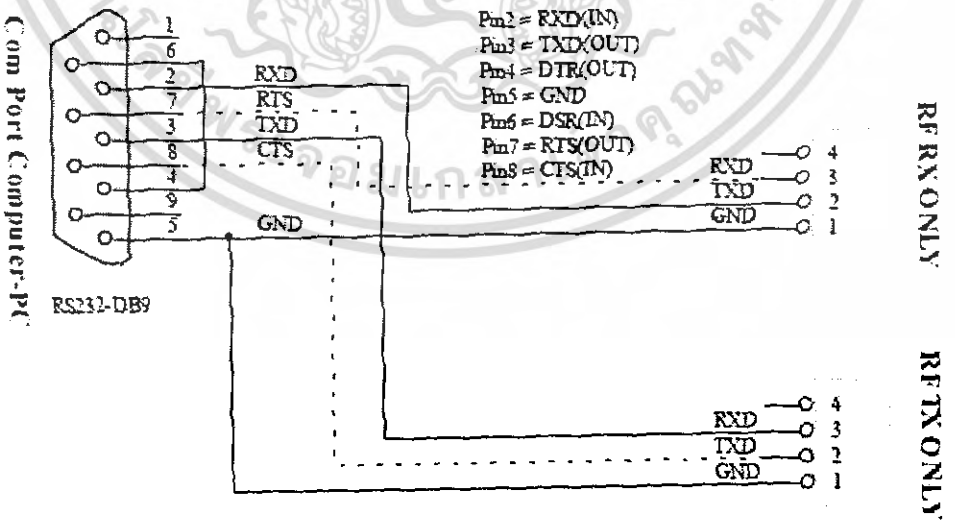
โดยในขณะที่มีการรับข้อมูลกันอยู่นั้น โปรแกรม Hyper Terminal ทั้ง 2 ด้านจะแสดงสถานะการทำงานให้ทราบอยู่ตลอดเวลา ดังภาพ



ภาพที่ 2.65 แสดงการ Zmodem with Crash Recovery file send for DirectCom1

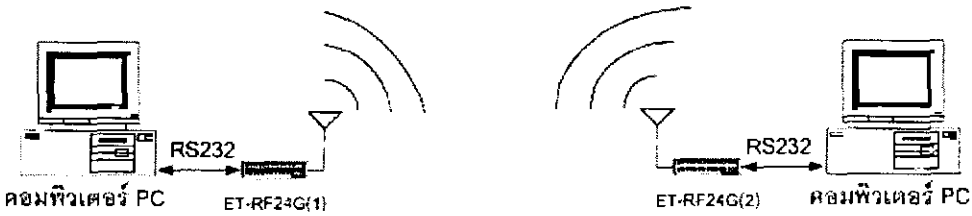
โดยให้รอจนกว่าการทำงานจะเสร็จสมบูรณ์ ซึ่งหน้าต่างที่แสดงสถานะการทำงานของโปรแกรมจะถูกปิดไปเองโดยอัตโนมัติหลังจากทำการรับส่งข้อมูลเสร็จเรียบร้อยแล้ว

โดยในการทดสอบการทำงานของโปรแกรมตามตัวอย่างนี้ จะต้องกำหนดรูปแบบการสื่อสารของ RS232 ให้มีการตรวจสอบความพร้อมในการรับส่งข้อมูลกันด้วยสัญญาณทาง Hardware ด้วย โดยเลือกกำหนดรูปแบบการสื่อสารของ RS232 ในหัวข้อ Flow Control เป็น Hardware พร้อมกับต่อสายสัญญาณดังวงจรต่อไปนี้



ภาพที่ 2.66 แสดงวงจรของสายที่ใช้สำหรับทดสอบการรับส่งข้อมูลแบบ Full Duplex

2.3.6.1 การรับส่งข้อมูล 2 ทิศทาง (Half Duplex) แบบจุดต่อจุด (Point to Point)



ภาพที่ 2.67 แสดงการรับส่งข้อมูล 2 ทิศทาง (Half Duplex) แบบจุดต่อจุด (Point to Point)

ตัวอย่างนี้เป็นการรับส่งข้อมูลระหว่างอุปกรณ์ที่มีการสื่อสารอนุกรมแบบ RS232 จำนวน 2 ชุด โดยต้องใช้รูปแบบการสื่อสารแบบ Half Duplex หรือผลัดกันรับผลัดกันส่ง กล่าวคือ ด้านรับจะต้องทำการรอรับข้อมูลจากด้านส่งจนครบทั้งหมด แล้วจึงจะส่งกลับไปได้ ซึ่งจะไม่สามารถส่งข้อมูลสวนทางกลับไปในขณะที่กำลังรับข้อมูลอยู่ได้ โดยการสื่อสารแบบนี้ฝ่ายรับข้อมูลจะต้องรอให้รับข้อมูลได้ครบทั้งหมดเสียก่อน จากนั้นจึงจะส่งข้อมูลตอบกลับไปได้ โดยให้กำหนดค่า Configuration ของตัวเครื่อง ET-RF24G V1.0 เป็นดังนี้

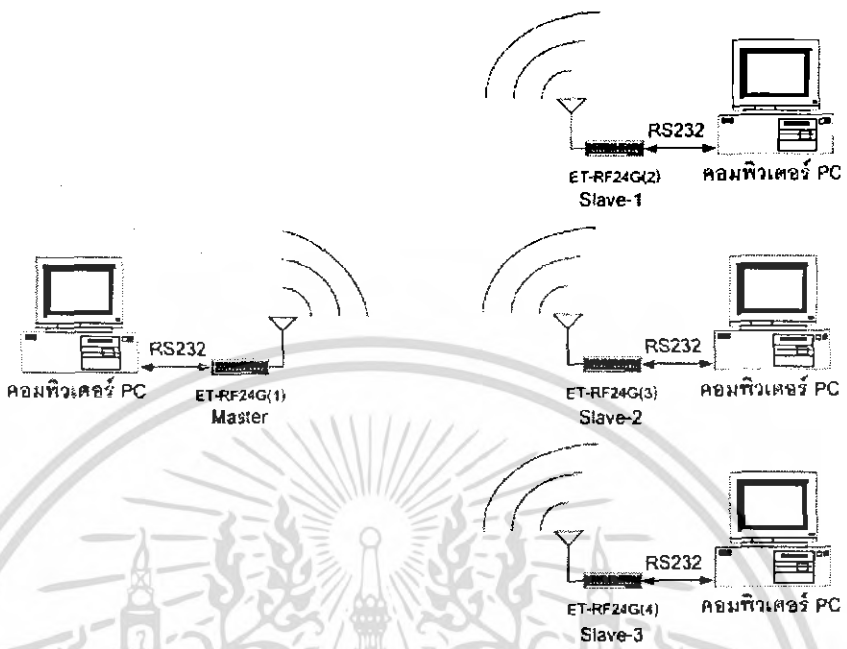
ตารางที่ 2.11 แสดงค่า Configuration ของการรับส่งข้อมูล (Half Duplex) แบบจุดต่อจุด (Point to Point)

ค่า Configuration	ET-RF24G V1.0 ตัวที่1	ET-RF24G V1.0 ตัวที่2
User RS232 Baudrate	9600 Bps	9600 Bps
RF Data Rate	250 Kbps	250 Kbps
RF Operation Mode	Auto Direction	Auto Direction
RF Power Gain	+0dBm	+0dBm
RXD ID Code	01	02
TXD ID Code	02	01
RF Frequency Channel	0	0

ข้อสังเกตในการกำหนด Configuration

- ค่า RF Frequency Channel ต้องกำหนดให้ตรงกันทั้ง 2 ตัว
- ค่า RF Data Rate ต้องกำหนดให้ตรงกันทั้ง 2 ตัว
- ค่า RXD ID Code ของตัวที่1 ต้องตรงกับ TXD ID Code ของตัวที่2
- ค่า TXD ID Code ของตัวที่1 ต้องตรงกับ RXD ID Code ของตัวที่2

2.3.6.2 การรับส่งข้อมูล 2 ทิศทาง (Half Duplex) แบบหลายๆจุด (RF Network)



ภาพที่ 2.68 แสดงการรับส่งข้อมูล 2 ทิศทาง (Half Duplex) แบบหลายๆจุด (RF Network)

ตัวอย่างนี้เป็นการรับส่งข้อมูลระหว่างอุปกรณ์ที่มีการสื่อสารอนุกรมแบบ RS232 จำนวนหลายๆ ตัวต่อร่วมกัน โดยหลักการสื่อสารแบบนี้จะใช้ตัว Master เป็นตัวควบคุมการสื่อสารกับ Slave แต่ละตัวในระบบ โดยเมื่อ Master จะทำการส่งข้อมูลออกไปจะมีการใส่รหัส ID code ของ Slave ที่ต้องใช้การสื่อสารรวมไปในชุดข้อมูลนั้นๆด้วย ซึ่ง Slave ทุกๆ ตัวจะรับข้อมูลจาก Master ได้เหมือนๆ กัน แต่จะมี Slave เพียงตัวเดียวที่ตอบสนองต่อข้อมูลนั้นๆ โดยให้กำหนดค่า Configuration ของตัวเครื่อง ET-RF24G V1.0 เป็นดังนี้

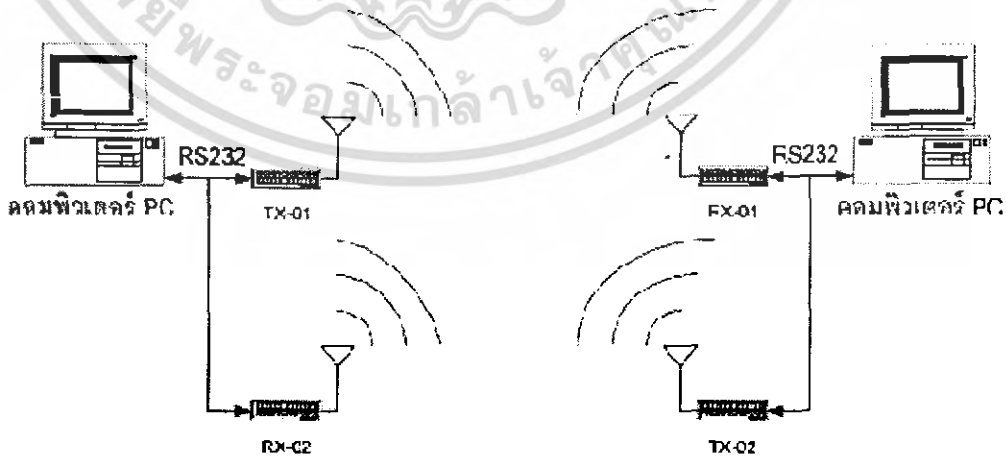
ตารางที่ 2.12 แสดงค่า Configuration ของการรับส่งข้อมูล 2 ทิศทาง (Half Duplex) แบบหลายๆจุด (RF Network)

ค่า Configuration	ET-RF24G V1.0 ฝ่ายต้นทาง		ET-RF24G V1.0 ฝ่ายปลายทาง	
	ตัวที่1 (RF RX1)	ตัวที่2 (RF TX1)	ตัวที่3 (RF RX2)	ตัวที่4 (RF TX2)
User RS232 Baudrate	19200 Bps	19200 Bps	19200 Bps	19200 Bps
RF Data Rate	250 Kbps	250 Kbps	250 Kbps	250 Kbps
RF Operation Mode	RF RX Only	RF TX Only	RF RX Only	RF TX Only
RF Power Gain	+0dBm	+0dBm	+0dBm	+0dBm
RXD ID Code	01	-	02	-
TXD ID Code	-	02	-	01
RF Frequency Channel	0	124	124	0

ข้อสังเกตในการกำหนด Configuration

- ค่า RF Frequency Channel ต้องกำหนดให้ตรงกันทั้งหมดทุกตัว
- ค่า RF Data Rate ต้องกำหนดให้ตรงกันทั้งหมดทุกตัว
- ค่า TXD ID Code ของตัวที่1 (Master) ต้องตรงกับ RXD ID Code ของตัวที่2-4 (Slave1-3)
- ค่า RXD ID Code ของตัวที่1 (Master) ต้องตรงกับ TXD ID Code ของตัวที่2-4 (Slave1-3)

2.3.6.3 การรับส่งข้อมูลแบบ Full Duplex



ภาพที่ 2.69 แสดงการรับส่งข้อมูลแบบ Full Duplex

ในตัวอย่างนี้เป็นการประยุกต์ใช้งานเครื่อง ET-RF24G V1.0 สำหรับทำการรับส่งข้อมูลแบบ Full Duplex โดยกำหนดโหมดการใช้งานเป็นแบบ RF Receive Only และ RF Transmit Only ด้านละชุด โดยให้กำหนดค่า Configuration ของตัวเครื่อง ET-RF24G V1.0 เป็นดังนี้

ตารางที่ 2.13 แสดงค่า Configuration ของการรับส่งข้อมูลแบบ Full Duplex

ค่า Configuration	ET-RF24G V1.0 ฝ่ายต้นทาง		ET-RF24G V1.0 ฝ่ายปลายทาง	
	ตัวที่1 (RF RX1)	ตัวที่2 (RF TX1)	ตัวที่3 (RF RX2)	ตัวที่4 (RF TX2)
User RS232 Baudrate	19200 Bps	19200 Bps	19200 Bps	19200 Bps
RF Data Rate	250 Kbps	250 Kbps	250 Kbps	250 Kbps
RF Operation Mode	RF RX Only	RF TX Only	RF RX Only	RF TX Only
RF Power Gain	+0dBm	+0dBm	+0dBm	+0dBm
RXD ID Code	01	-	02	-
TXD ID Code	-	02	-	01
RF Frequency Channel	0	124	124	0

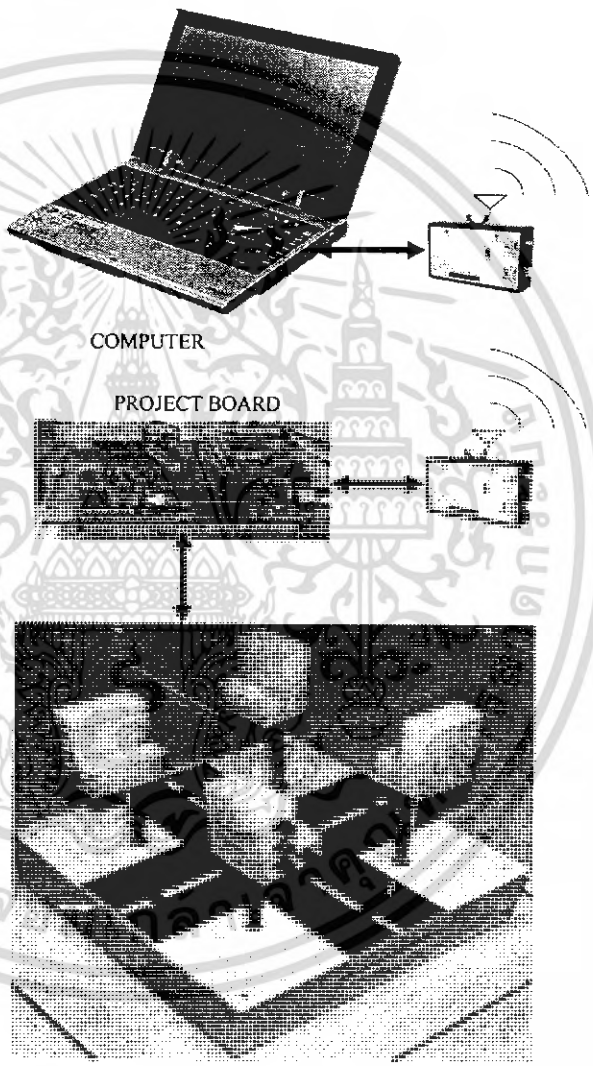
ข้อสังเกตในการกำหนด Configuration

- ค่า RF Data Rate ต้องกำหนดให้ตรงกันทั้งหมดทุกตัว
- ค่า RF Frequency Channel ของตัวรับด้านต้นทางต้องกำหนดให้ตรงกับตัวส่งด้านปลายทาง
- ค่า RF Frequency Channel ของตัวส่งด้านต้นทางต้องกำหนดให้ตรงกับตัวรับด้านปลายทาง
- ค่า RXD ID Code ของตัวรับด้านต้นทางต้องตรงกับ TXD ID Code ของตัวส่งด้านปลายทาง
- ค่า TXD ID Code ของตัวส่งด้านต้นทางต้องตรงกับ RXD ID Code ของตัวรับด้านปลายทาง

บทที่ 3

การออกแบบและการทำงาน

3.1 ผังการทำงานของ Microcontroller Based Remote Sensing



ภาพที่ 3.1 แสดงผังการทำงานของ Microcontroller Based Remote Sensing

Microcontroller Based Remote Sensing ประกอบด้วย

1. COMPUTER
2. RS-232 to RF-Wireless CONVERTER
3. PROJECT BOARD
4. PLANT MODEL

3.1.1 COMPUTER

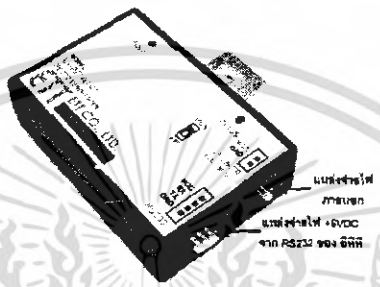
คอมพิวเตอร์เป็นอุปกรณ์พื้นฐานที่สำคัญมากสำหรับงานทางด้าน ENGINEERING สำหรับโครงงาน Microcontroller Based Remote Sensing นี้ เราใช้คอมพิวเตอร์ในหน้าที่ต่างๆ ดังต่อไปนี้

1. ใช้เขียนโปรแกรมต่างๆในส่วนของไมโครคอนโทรลเลอร์ การเซตค่าให้กับอุปกรณ์ที่เกี่ยวข้องกับการรับส่งข้อมูล
2. ใช้ในการแสดงการทำงานของ PLANT MODEL แบบ REAL TIME หรือที่เรียกว่า การ MONITORING ซึ่งการเขียนกราฟิกในส่วนนี้ใช้โปรแกรม VISUAL BASIC 6.0
3. ประมวลผลและบันทึกค่ากระบวนการ ซึ่งค่าต่างๆ ที่บันทึกนี้สามารถเรียกดูในภายหลังได้

ภาพที่ 3.2 แสดงภาพคอมพิวเตอร์ที่ใช้ในการทดลอง

3.1.2 RS232 to RF-Wireless CONVERTER

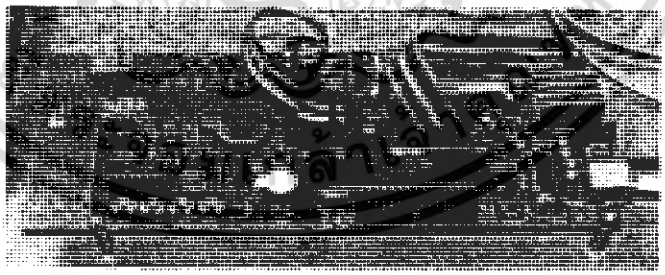
อุปกรณ์ตัวนี้ (ทางด้านส่ง) ทำหน้าที่แปลงสัญญาณจาก RS232 ให้เป็นสัญญาณความถี่วิทยุส่งออกไปในอากาศ โดยที่ทางด้านรับก็จะแปลงสัญญาณความถี่วิทยุที่รับได้กลับเป็นสัญญาณแบบ RS232 ดังเดิม ในโครงงานนี้ใช้วิธีการสื่อสารแบบ Haif Duplex ซึ่งได้กล่าวไว้โดยละเอียดแล้วในบทที่ 2 ดังนั้นเมื่อเราออกแบบโปรแกรมก็ควรพิจารณาในส่วนนี้ด้วยเพราะการสื่อสารแต่ละแบบมีขั้นตอนและวิธีการรับส่งข้อมูลที่ต่างกัน



ภาพที่ 3.2 แสดงชุดรับส่ง RS232 to RF-Wireless CONVERTER

3.1.3 PROJECT BOARD

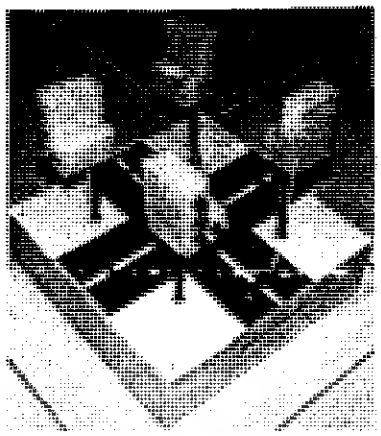
PROJECT BOARD นี้เป็นที่อยู่ของอุปกรณ์ที่สำคัญตัวหนึ่งคือไมโครคอนโทรลเลอร์ซึ่งเป็นที่เก็บโปรแกรมที่เราเขียนขึ้นเพื่อสั่งการให้ PLANT MODEL แสดงผลหรือทำงานได้ตามคำสั่งที่รับมา



ภาพที่ 3.3 แสดง PROJECT BOARD

3.1.4 PLANT MODEL

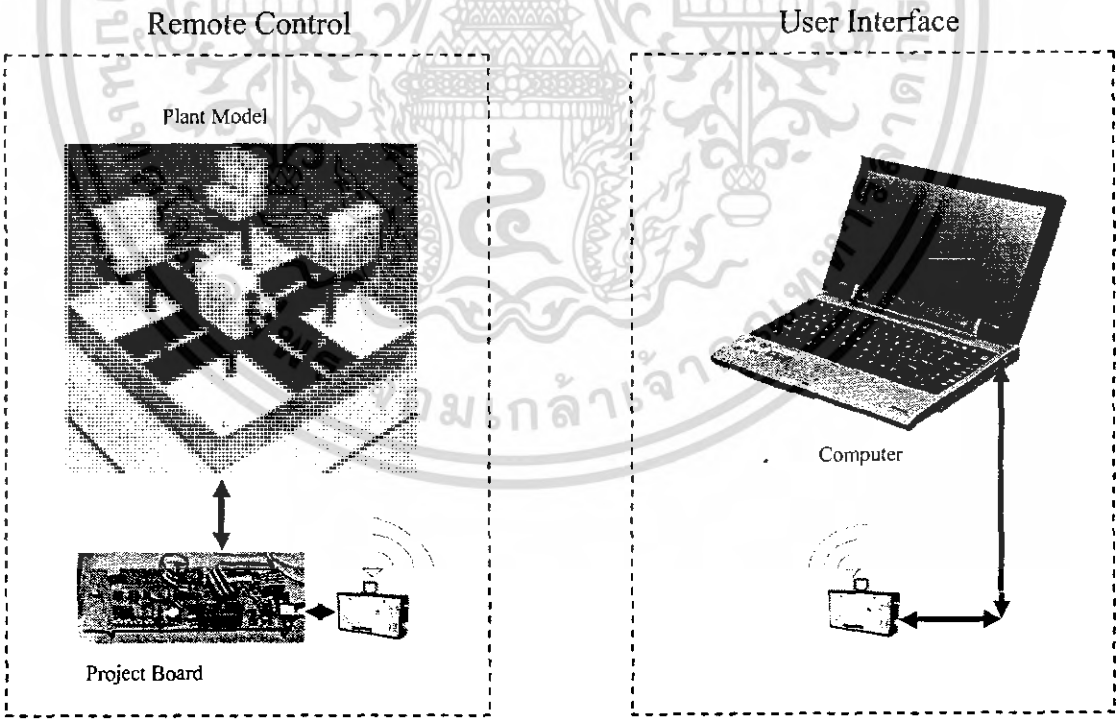
เป็นอุปกรณ์ที่เราออกแบบขึ้นตามวัตถุประสงค์ของการควบคุมสิ่งที่เราต้องการจะควบคุม ซึ่ง PLANT MODEL นี้โดยมากแล้วจะไม่ซับซ้อนมากนักเพราะต้องการสื่อให้ผู้รับชมเข้าใจได้โดยง่าย



ภาพที่ 3.4 แสดง PLANT MODEL

3.2 การเชื่อมต่ออุปกรณ์ในการใช้งานจริง

การเชื่อมต่อในการใช้งานจริงนั้นจะเห็นว่าอุปกรณ์ถูกแยกออกเป็นสองส่วนหลักๆ คือ ส่วนของผู้ใช้งาน (User Interface) และส่วนที่ต้องการควบคุม (Remote Control) ดังแสดงตามภาพที่ 3.5

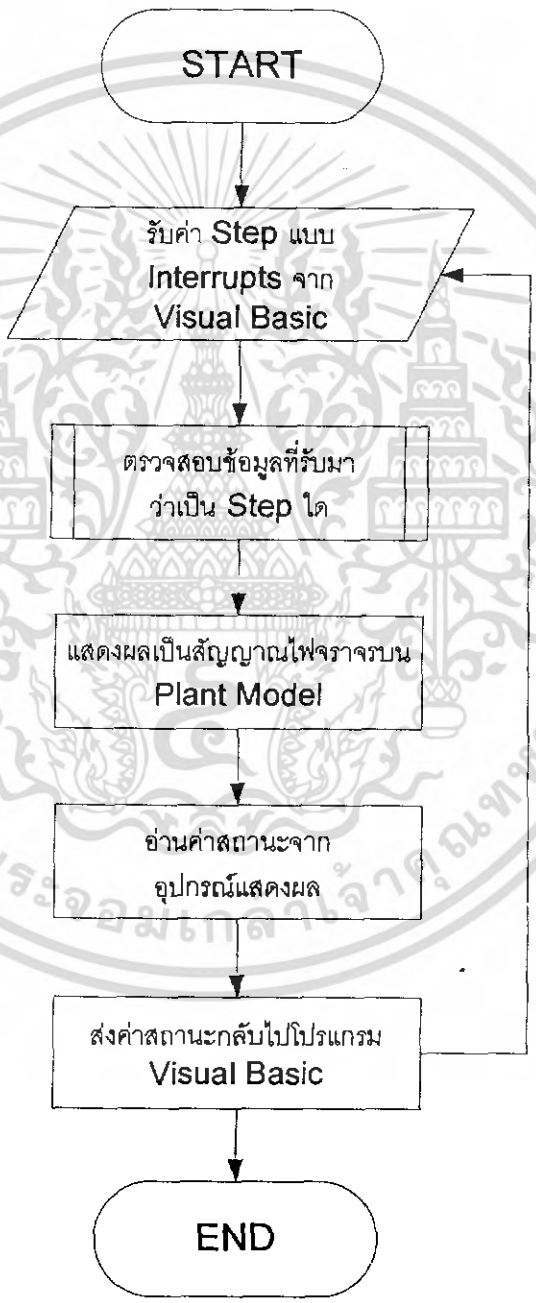


ภาพที่ 3.5 การเชื่อมต่ออุปกรณ์ในการใช้งานจริง

3.3 ผังการออกแบบโปรแกรมควบคุมการทำงาน

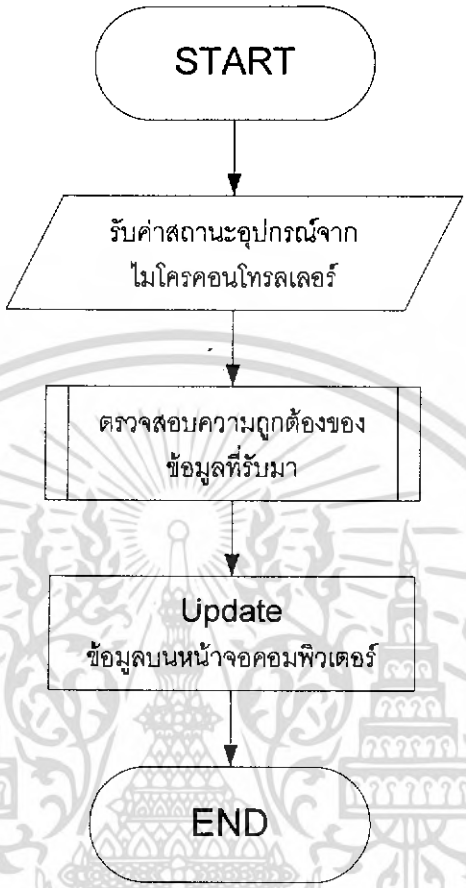
จากที่ได้กล่าวมาแล้วว่าอุปกรณ์จะถูกแยกออกเป็นสองส่วนหลักๆ คือส่วนของผู้ใช้งาน (User Interface) และส่วนที่ต้องการควบคุม (Remote Control) ดังนั้นในส่วนของโปรแกรมก็แบ่งเป็นสองส่วนเช่นกัน

3.3.1 ผังโปรแกรมควบคุมการทำงานส่วน Remote Control



ภาพที่ 3.6 แสดงผังโปรแกรมควบคุมการทำงานส่วน Remote Control

3.3.2 ผังโปรแกรมควบคุมการทำงานส่วน User Interface



ภาพที่ 3.7 แสดงผังโปรแกรมควบคุมการทำงานส่วน User Interface

3.4 โปรแกรมควบคุมการทำงาน

3.4.1 โปรแกรมควบคุมการทำงานส่วน Remote Control

```
#include "reg52.h"
#include "absacc.h"
#include "Tx_Rx.c"
#include "Delay.c"
unsigned  int StepLED[16]={0xEEEE,0xDEEE,0x3EEE,0xDEEE,0xEDEE,0xE3EE,0xEDEE,
                           0xEEDE,0xEE3E,0xEEDE,0xEEED,0xEEE3,0xEEED};
```

```
unsigned  int temp;
unsigned char Step;
unsigned char InputCnt;
unsigned char Buffer[3];
unsigned char LED[16];
sbit LineA = P2^0;
sbit LineB = P2^1;
sbit LineC = P2^2;
sbit LineD = P2^3;
void ShowLED(void);
void TX_Status(void);
void Chk_Status(void);
/*****************************************************/
void main()
 {
        SCON=0x50;              //Serial Port RxTx Mode
        T2CON=0x34;             //Baud rate mode
        RCAP2H=0xFF;            //18432000/2/16=576000:576000/9600=60 =3CH
        RCAP2L=0xC4;            //10000H-3CH = FFC4H
        Step=0;
        P2=0xFF;                //Input Port
        EA=1;
        ES=1;
                while(1)
                {
                        ShowLED();
                        Chk_Status();
                }
}
```

```
void Chk_Status(void)
 {      unsigned char i;
        bit Out;
        for(i=0;i<16;i++)
        {       Out=temp&0x8000;
                temp=temp<<1;
                LED[i]=0x30|Out;
        }
        if(LineA==0){ LED[3] = 0x35;}
        if(LineB==0){ LED[7] = 0x35;}
        if(LineC==0){ LED[11]= 0x35;}
        if(LineD==0){ LED[15]= 0x35;}
 }
void TX_Status(void)
 {      unsigned char i;
        TX_BYTE('S');
        TX_BYTE(Step);
        for(i=0;i<16;i++) { TX_BYTE(LED[i]); }
        TX_BYTE('P');
 }
void ShowLED()
 {      unsigned int *dptr;
        dptr=&StepLED[Step];
        temp=*dptr;
        P0=temp>>8;
        P1=temp&0x00FF;
 }
```

```
void Serial_Int() interrupt 4
 {
        if(RI)
        {       RI=0;
                Step=SBUF;
                Delay(20);
                TX_Status();
        }
 }
void TX_BYTE(unsigned char Data_TX)
 {
        TI=0;
        SBUF=Data_TX;
        while(TI==0){}
        TI=0;
 }
void TX_2BYTE(unsigned int Data_TX2)//TX Data 16 bit to wireless module/
 {
        TX_BYTE(Data_TX2>>8);           //High byte
        TX_BYTE(Data_TX2&0xFF);         //Low byte
 }
void TXB_ASC(unsigned char Data)
 {
        unsigned char temp;
        unsigned char Disp[3];          //for LCD display
        temp=Data;
        Disp[0]=temp/100;               //123 => 1__
        temp=temp%100;                  //123 => _23
        Disp[1]=temp/10;                //23  => 2_
        Disp[2]=temp%10;                //23  => _3
```

```
        if(Disp[0]==0)
                {       TX_BYTE(0x20);          }
                else
                {       TX_BYTE(0x30+Disp[0]);          }
                if((Disp[0]==0)&&(Disp[1]==0))
                {       TX_BYTE(0x20);          }
                else
                {       TX_BYTE(0x30+Disp[1]);          }
                TX_BYTE(0x30+Disp[2]);
                TX_BYTE(' ');
 }
void TX2B_ASC(unsigned int Data)
 {
        unsigned int temp;
        unsigned char Disp[5];
        temp=Data;
        Disp[0]=temp/10000;                     //12345 => 1____
        temp=temp%10000;                        //12345 => _2345
        Disp[1]=temp/1000;                      //2345  => 2___
        temp=temp%1000;                         //2345  => _345
        Disp[2]=temp/100;                       //345   => 3__
        temp=temp%100;                          //345   => _45
        Disp[3]=temp/10;                        //45    => 4_
        Disp[4]=temp%10;                        //45    => _5
                if(Disp[0]==0)
                {       TX_BYTE(0x20);          }
                else
                {       TX_BYTE(0x30+Disp[0]);          }

                if((Disp[0]==0)&&(Disp[1]==0))
                {       TX_BYTE(0x20);          }
```

```
        else
        {       TX_BYTE(0x30+Disp[1]);      }
        if((Disp[0]==0)&&(Disp[1]==0)&&(Disp[2]==0))
        {       TX_BYTE(0x20);      }
        else
        {       TX_BYTE(0x30+Disp[2]);      }
        if((Disp[0]==0)&&(Disp[1]==0)&&(Disp[2]==0)&&(Disp[3]==0))
        {       TX_BYTE(0x20);      }
        else
        {       TX_BYTE(0x30+Disp[3]);      }
        TX_BYTE(0x30+Disp[4]);
        TX_BYTE(' ');
}
void Delay(unsigned int x)
 {      unsigned int i,j;
        for(i=0;i<x;i++)
        {       for(j=0;j<300;j++){}    }
 }
```

3.4.2 โปรแกรมควบคุมการทำงานส่วน User Interface

```
Option Explicit
Public StepMCU As String
Public Step As Byte
Public TimeSet As Byte
Public Cnt As Double
Public Cnt2 As Double
Public Cnt3 As Double
Dim Status(15) As String
Public TimG1 As Byte
Public TimG2 As Byte
Public TimG3 As Byte
```

```
Public TimG4 As Byte
Public Run As Boolean
Dim a As Double

    Private Sub CmdExit_Click()
        TimSend.Enabled = False
        Step = 0
        TxStep
        End
    End Sub
'*******************************************************
    Private Sub Form_Load()
        Dim i As Byte
        MSComm1.CommPort = 3
        MSComm1.Settings = "9600,n,8,1"
        MSComm1.RThreshold = 1
        MSComm1.InputLen = 0
        MSComm1.PortOpen = True
        Step = 0
    TimSend.Enabled = True
    OptMan.Value = True
        TxtGreen1.Text = "2"
        TxtGreen2.Text = "2"
        TxtGreen3.Text = "2"
        TxtGreen4.Text = "2"
        TimG1 = 2
        TimG2 = 2
        TimG3 = 2
        TimG4 = 2
        Run = False
    End Sub
'*******************************************************
```

```
Private Sub TxStep()
        If MSComm1.PortOpen = True Then
         Dim Buffer As Variant
        Buffer = Chr(Step)
        MSComm1.Output = Buffer
         MSComm1.OutBufferCount = 0
        End If
    End Sub
'*****************************************************
    Private Sub MSComm1_OnComm()
        If MSComm1.CommEvent = comEvReceive Then        'Event Receive Only
            Dim Buffer As Variant
            On Error GoTo ErrorConnect
        If MSComm1.InBufferCount >= 19 Then
            Buffer = MSComm1.Input
            MSComm1.InBufferCount = 0
              Text1.Text = Buffer
              Text2.Text = Len(Buffer)
        If (Mid(Buffer, 1, 1) = "S") And (Mid(Buffer, 19, 1) = "P") And (Len(Buffer) =
19) Then

            Dim i As Byte
            Dim Buffer2 As String
            StepMCU = Asc(Mid(Buffer, 2, 1))
            Buffer = Mid(Buffer, 3, 16)
                For i = 0 To 15
                    Buffer2 = Buffer2 + Mid(Buffer, 16 - i, 1)
                    Next i
                For i = 0 To 15
                    Status(i) = Mid(Buffer2, i + 1, 1)
                    Next i
                For i = 0 To 15
                    led(i).FillColor = CheckStatus(i)
```

```
                    Next i
                    End I
                End If
                Text3.Text = StepMCU
            End If
            ErrorConnect: Exit Sub
        End Sub
'*****************************************************
        Function CheckStatus(i As Byte) As String
        Dim Colours As String
        Select Case Status(i)
            Case "0"
                GoTo CheckColour
            Case "1"
                CheckStatus = vbBlack
            Case "5"
                CheckStatus = vbWhite
        Case Else
        End Select
        Exit Function
            CheckColour:
            Select Case i
                Case 0
                    CheckStatus = vbRed
                Case 1
                    CheckStatus = vbYellow
                Case 4
                    CheckStatus = vbRed
                Case 5
                    CheckStatus = vbYellow
                Case 8
                    CheckStatus = vbRed
```

```
                Case 9
                        CheckStatus = vbYellow
                Case 12
                        CheckStatus = vbRed
                Case 13
                        CheckStatus = vbYellow
                Case Else
                        CheckStatus = vbGreen
        End Select
End Function
'*****************************************************
Private Sub TimSend_Timer()
        Dim Buffer As Variant
                Cnt3 = Cnt3 + 1
                TxStep
        If OptAuto.Value = True Then
                If Cnt3 >= 20 Then
                        Cnt3 = 0
                        Cnt = Cnt + 1
                        CheckStep
                End If
        End If
        If OptMan.Value = True Then
                If Cnt3 >= 20 Then
                        Cnt3 = 0
                        Cnt2 = Cnt2 + 1
                End If
        End If
End Sub
'*****************************************************
```

```
Private Sub CheckStep()
    Select Case Step
        Case 0
            If Cnt = 1 Then
                Step = 1
                Cnt = 0
            End If
        Case 1
            If Cnt = 1 Then
                Step = 2
                Cnt = 0
            End If
        Case 2
            If Cnt = TimG1 Then
                Step = 3
                Cnt = 0
            End If
        Case 3
            If Cnt = 1 Then
                Step = 4
                Cnt = 0
            End If
        Case 4
            If Cnt = 1 Then
                Step = 5
                Cnt = 0
            End If
        Case 5
            If Cnt = TimG2 Then
                Step = 6
                Cnt = 0
            End If
```

```
Case 6
        If Cnt = 1 Then
            Step = 7
            Cnt = 0
        End If
Case 7
        If Cnt = 1 Then
            Step = 8
            Cnt = 0
        End If
Case 8
        If Cnt = TimG3 Then
            Step = 9
            Cnt = 0
        End If
Case 9
        If Cnt = 1 Then
            Step = 10
            Cnt = 0
        End If
Case 10
        If Cnt = 1 Then
            Step = 11
            Cnt = 0
        End If
Case 11
        If Cnt = TimG4 Then
            Step = 12
            Cnt = 0
        End If
```

```
                Case 12
                    If Cnt = 1 Then
                        Step = 1
                        Cnt = 0
                    End If
            Case Else
            End Select
        End Sub
'*****************************************************
        Private Sub Cmd1_Click()
            If (Run = False) And (Step <> 2) Then
                Run = True
                Chk (Step)
                Manual (2)
                Run = False
            End If
        End Sub
'*****************************************************
        Private Sub Cmd2_Click()
            If (Run = False) And (Step <> 5) Then
                Run = True
                Chk (Step)
                Manual (5)
                Run = False
            End If
        End Sub
'*****************************************************
        Private Sub Cmd3_Click()
            If (Run = False) And (Step <> 8) Then
                Run = True
                Chk (Step)
                Manual (8)
```

```
                Run = False
            End If
        End Sub
'*******************************************************
        Private Sub Cmd4_Click()
            If (Run = False) And (Step <> 11) Then
                Run = True
                Chk (Step)
                Manual (11)
                Run = False
            End If
        End Sub
'*******************************************************
        Private Sub Command1_Click()
            Chk (Step)
            Cnt2 = 0
            Do Until Cnt2 >= 1
                DoEvents
                    Loop
                        Step = 0
                        Cnt2 = 0
            Do Until Cnt2 >= 1
                DoEvents
                        Loop
        End Sub
'*******************************************************
        Private Sub OptAuto_Click()
            If OptAuto.Value = True Then
                Cmd1.Enabled = False
                Cmd2.Enabled = False
                Cmd3.Enabled = False
                Cmd4.Enabled = False
```

```
                Command1.Enabled = False
                Frame3.Enabled = False
                Frame2.Enabled = True
                Label1.Enabled = True
                Label2.Enabled = True
                Label3.Enabled = True
                Label4.Enabled = True
                TxtGreen1.Enabled = True
                TxtGreen2.Enabled = True
                TxtGreen3.Enabled = True
                TxtGreen4.Enabled = True
                CmdAct.Enabled = True
                CmdDefault.Enabled = True
            End If
        End Sub
'*****************************************************
        Private Sub OptMan_Click()
            If OptMan.Value = True Then
                Cmd1.Enabled = True
                Cmd2.Enabled = True
                Cmd3.Enabled = True
                Cmd4.Enabled = True
                Command1.Enabled = True
                Frame3.Enabled = True
                Frame2.Enabled = False
                Label1.Enabled = False
                Label2.Enabled = False
                Label3.Enabled = False
                Label4.Enabled = False
                TxtGreen1.Enabled = False
                TxtGreen2.Enabled = False
                TxtGreen3.Enabled = False
```

```
                TxtGreen4.Enabled = False
                CmdAct.Enabled = False
                CmdDefault.Enabled = False
                Step = 0
            End If
        End Sub
'*********************************************************************
        Function Manual(S As Byte)
            Cnt2 = 0
                Do Until Cnt2 >= 1
                    DoEvents
                Loop
                Step = 0
                Cnt2 = 0
                Do Until Cnt2 >= 1
                    DoEvents
                Loop
                Step = S
        End Function
'*********************************************************
        Function Chk(S As Byte)
            If S = 2 Then
                Step = 1
            End If
            If S = 5 Then
                Step = 4
            End If
            If S = 8 Then
                Step = 7
            End If
```

```
            If S = 11 Then
                Step = 10
            End If
        End Function
'*********************************************************
        Private Sub CmdAct_Click()
            If CmdAct.Caption = "Set" Then
                On Error GoTo ErrorText
            If (CInt(TxtGreen1.Text) <= 180) And (CInt(TxtGreen2.Text) <= 180) _
             And (CInt(TxtGreen3.Text) <= 180) And (CInt(TxtGreen4.Text) <= 180) Then
                CmdAct.Caption = "Reset"
                TxtGreen1.Enabled = False
                TxtGreen2.Enabled = False
                TxtGreen3.Enabled = False
                TxtGreen4.Enabled = False
                TimG1 = CInt(TxtGreen1.Text)
                TimG2 = CInt(TxtGreen2.Text)
                TimG3 = CInt(TxtGreen3.Text)
                TimG4 = CInt(TxtGreen4.Text)
            Exit Sub
        End If
    End If
            If CmdAct.Caption = "Reset" Then
                CmdAct.Caption = "Set"
                TxtGreen1.Enabled = True
                TxtGreen2.Enabled = True
                TxtGreen3.Enabled = True
                TxtGreen4.Enabled = True
            Exit Sub
        End If
```

```
                ErrorText:
            Exit Sub
        End Sub
'*********************************************************
        Private Sub CmdDefault_Click()
            TimG1 = 2
            TimG1 = 2
            TimG1 = 2
            TimG1 = 2
            TxtGreen1.Text = "2"
            TxtGreen2.Text = "2"
            TxtGreen3.Text = "2"
            TxtGreen4.Text = "2"
        End Sub
'*********************************************************
```

บทที่ 4

การทดลองและผลการทดลอง

4.1 การทดลอง

ในการทดลองควบคุมสัญญาณไฟจราจรให้เป็นไปตาม Sequent Table ตามตารางที่ 4.1 ได้ทดลองรับส่งที่ระยะทางต่างๆ ในสถานที่โล่งแจ้งและในสถานที่ๆ มีสิ่งกีดขวางทั้งโหมด Auto และโหมด Manual

ตารางที่ 4.1 แสดงลำดับการควบคุมไฟจราจร

	แยกที่ 1				แยกที่ 2				แยกที่ 3				แยกที่ 4			
Step	ข$_1$	ข$_2$	ห	ด	ข$_1$	ข$_2$	ห	ด	ข$_1$	ข$_2$	ห	ด	ข$_1$	ข$_2$	ห	ด
0	●	●	●	☺	●	●	●	☺	●	●	●	☺	●	●	●	☺
1	●	●	☺	●	●	●	●	☺	●	●	●	☺	●	●	●	☺
2	☺	☺	●	●	●	●	●	☺	●	●	●	☺	●	●	●	☺
3	●	●	☺	●	●	●	●	☺	●	●	●	☺	●	●	●	☺
4	●	●	●	☺	●	●	☺	●	●	●	●	☺	●	●	●	☺
5	●	●	●	☺	☺	☺	●	●	●	●	●	☺	●	●	●	☺
6	●	●	●	☺	●	●	☺	●	●	●	●	☺	●	●	●	☺
7	●	●	●	☺	●	●	●	☺	●	●	☺	●	●	●	●	☺
8	●	●	●	☺	●	●	●	☺	☺	☺	●	●	●	●	●	☺
9	●	●	●	☺	●	●	●	☺	●	●	☺	●	●	●	●	☺
10	●	●	●	☺	●	●	●	☺	●	●	●	☺	●	●	☺	●
11	●	●	●	☺	●	●	●	☺	●	●	●	☺	☺	☺	●	●
12	●	●	●	☺	●	●	●	☺	●	●	●	☺	●	●	☺	●

หมายเหตุ ข$_1$ = ไฟเขียวเลี้ยวขวา, ข$_2$ = ไฟเขียวเลี้ยวซ้าย, ห = ไฟเหลือง, ด = ไฟแดง

☺ = ติด, ● = ดับ, Step 0 = เริ่มต้น, วน Loop ระหว่างStep 1 ถึง Step 12

4.2 ผลการทดลอง

4.2.1 การทดลองส่งข้อมูลที่ระยะทางน้อยกว่า 50 เมตร ในสถานที่โล่ง

ในการทดลองการรับส่งข้อมูลในที่โล่ง พบว่าระยะทางที่ไกลที่สุดที่สามารถรับส่งข้อมูลกันได้และยังคงความถูกต้องของการแสดงผลอยู่นั้น มีระยะทางประมาณ 50 เมตร และเมื่อเพิ่มระยะทางรับส่งให้มากกว่านี้จะเริ่มเกิดความผิดพลาดของข้อมูลมากขึ้นทำให้การแสดงผลที่ภาพกราฟฟิกบนหน้าจอคอมพิวเตอร์ไม่ตรงกับ Plant Model อนึ่งความผิดพลาดของข้อมูลนี้ก็อาจจะเกิดขึ้นได้ตลอดเวลาไม่ว่าเราจะทำการส่งที่ระยะทางเท่าใดก็ตาม แต่การที่เรายังเห็นภาพกราฟฟิกบนหน้าจอคอมพิวเตอร์ถูกต้องอยู่นั่นเป็นเพราะว่าเราส่งข้อมูลของแต่ละ Sequent ไปทุกๆ 50 mS แต่ระยะเวลาในการเปลี่ยน Sequent ที่น้อยที่สุดคือ 1 วินาทีนั่นก็หมายความว่าใน 1 วินาทีเรารับข้อมูลชุดเดิม 20 ครั้ง ดังนั้นใน1 วินาทีเราต้องการข้อมูลที่ถูกต้องเพียงครั้งเดียวภาพกราฟฟิกบนหน้าจอคอมพิวเตอร์ก็ยังคงถูกต้องตรงกับ Plant Model อยู่เสมอ

ตารางที่ 4.2 แสดงการทดลองส่งข้อมูลที่ระยะทางน้อยกว่า 50 เมตร ในที่โล่ง

	ระยะทางรับส่ง 5, 10, 15, 20, 25, 30, 35, 40, 45, 50 เมตร																ความถูกต้องของข้อมูล
	แยกที่ 1				แยกที่ 2				แยกที่ 3				แยกที่ 4				
Step	ข$_1$	ข$_2$	ห	ด	ข$_1$	ข$_2$	ห	ด	ข$_1$	ข$_2$	ห	ด	ข$_1$	ข$_2$	ห	ด	
0	☻	☻	☻	☺	☻	☻	☻	☺	☻	☻	☻	☺	☻	☻	☻	☺	OK
1	☻	☻	☺	☻	☻	☻	☻	☺	☻	☻	☻	☺	☻	☻	☻	☺	OK
2	☺	☺	☻	☻	☻	☻	☻	☺	☻	☻	☻	☺	☻	☻	☻	☺	OK
3	☻	☻	☺	☻	☻	☻	☻	☺	☻	☻	☻	☺	☻	☻	☻	☺	OK
4	☻	☻	☻	☺	☻	☻	☺	☻	☻	☻	☻	☺	☻	☻	☻	☺	OK
5	☻	☻	☻	☺	☺	☺	☻	☻	☻	☻	☻	☺	☻	☻	☻	☺	OK
6	☻	☻	☻	☺	☻	☻	☺	☻	☻	☻	☻	☺	☻	☻	☻	☺	OK
7	☻	☻	☻	☺	☻	☻	☻	☺	☻	☻	☺	☻	☻	☻	☻	☺	OK
8	☻	☻	☻	☺	☻	☻	☻	☺	☺	☺	☻	☻	☻	☻	☻	☺	OK
9	☻	☻	☻	☺	☻	☻	☻	☺	☻	☻	☺	☻	☻	☻	☻	☺	OK
10	☻	☻	☻	☺	☻	☻	☻	☺	☻	☻	☻	☺	☻	☻	☺	☻	OK
11	☻	☻	☻	☺	☻	☻	☻	☺	☻	☻	☻	☺	☺	☺	☻	☻	OK
12	☻	☻	☻	☺	☻	☻	☻	☺	☻	☻	☻	☺	☻	☻	☺	☻	OK

4.2.2 การทดลองส่งข้อมูลที่ระยะทางมากกว่า 50 เมตร ในสถานที่โล่ง

ตารางที่ 4.3 แสดงการทดลองส่งข้อมูลที่ระยะทางมากกว่า 50 เมตร ในที่โล่ง

	ระยะทางรับส่งมากกว่า 40 เมตร																ความถูกต้องของข้อมูล
	แยกที่ 1				แยกที่ 2				แยกที่ 3				แยกที่ 4				
Step	ข$_1$	ข$_2$	ห	ค	ข$_1$	ข$_2$	ห	ค	ข$_1$	ข$_2$	ห	ค	ข$_1$	ข$_2$	ห	ค	
0	☻	☻	☻	☺	☻	☻	☻	☺	☻	☻	☻	☺	☻	☻	☻	☺	OK
1	☻	☻	☺	☻	☻	☻	☻	☺	☻	☻	☻	☺	☻	☻	☻	☺	NON
2	☺	☺	☻	☻	☻	☻	☻	☺	☻	☻	☻	☺	☻	☻	☻	☺	OK
3	☻	☻	☺	☻	☻	☻	☻	☺	☻	☻	☻	☺	☻	☻	☻	☺	OK
4	☻	☻	☻	☺	☻	☻	☺	☻	☻	☻	☻	☺	☻	☻	☻	☺	NON
5	☻	☻	☻	☺	☺	☺	☻	☻	☻	☻	☻	☺	☻	☻	☻	☺	OK
6	☻	☻	☻	☺	☻	☻	☺	☻	☻	☻	☻	☺	☻	☻	☻	☺	OK
7	☻	☻	☻	☺	☻	☻	☻	☺	☻	☻	☺	☻	☻	☻	☻	☺	NON
8	☻	☻	☻	☺	☻	☻	☻	☺	☺	☺	☻	☻	☻	☻	☻	☺	OK
9	☻	☻	☻	☺	☻	☻	☻	☺	☻	☻	☺	☻	☻	☻	☻	☺	OK
10	☻	☻	☻	☺	☻	☻	☻	☺	☻	☻	☻	☺	☻	☻	☺	☻	OK
11	☻	☻	☻	☺	☻	☻	☻	☺	☻	☻	☻	☺	☺	☺	☻	☻	OK
12	☻	☻	☻	☺	☻	☻	☻	☺	☻	☻	☻	☺	☻	☻	☺	☻	NON

4.2.3 การทดลองส่งข้อมูลที่ระยะทางน้อยกว่า 20 เมตร ในสถานที่ๆ มีสิ่งกีดขวาง

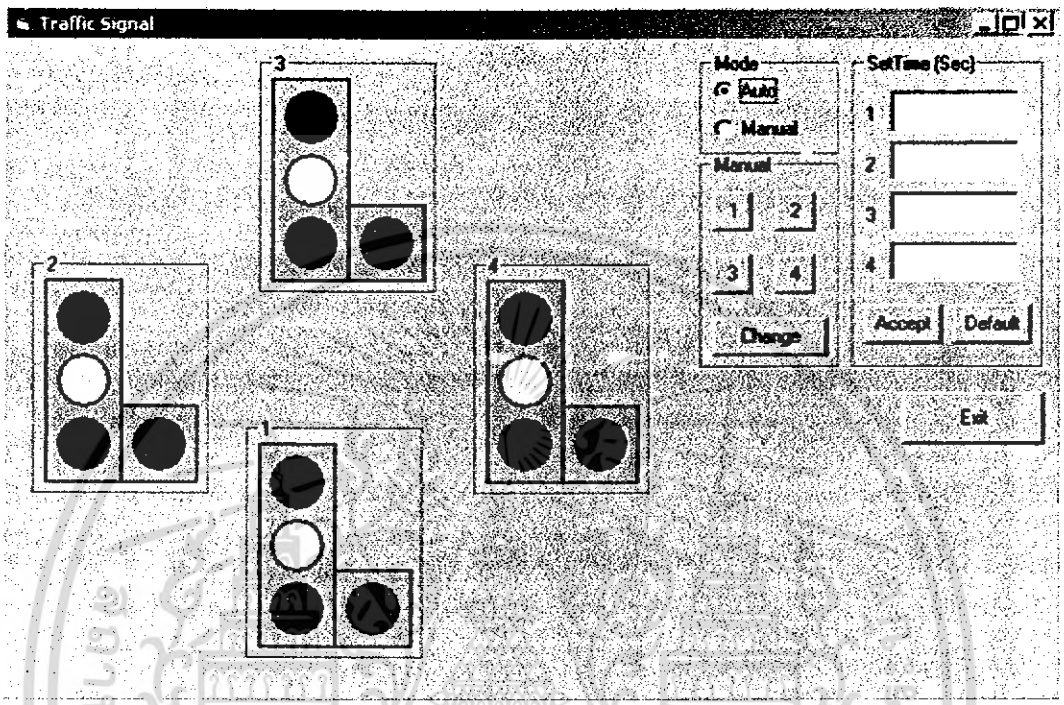
ในการทดลองการรับส่งข้อมูลในที่ๆ มีสิ่งกีดขวาง เช่นภายในบ้าน หรือสำนักงาน พบว่าระยะทางที่ไกลที่สุดที่สามารถรับส่งข้อมูลกันได้และยังคงความถูกต้องของการแสดงผลอยู่นั้น มีระยะทางประมาณ 20 เมตร เมื่อเพิ่มระยะทางรับส่งให้มากกว่านี้การแสดงผลภาพกราฟฟิกบนหน้าจอคอมพิวเตอร์จะไม่ตรงกับ Plant Model

ตารางที่ 4.4 แสดงการทดลองส่งข้อมูลที่ระยะทางต่างๆ ในสถานที่ๆ มีสิ่งกีดขวาง

	ระยะทางรับส่ง 5, 10, 15, 20 เมตร																ความถูกต้องของข้อมูล
	แยกที่ 1				แยกที่ 2				แยกที่ 3				แยกที่ 4				
Step	ข$_1$	ข$_2$	ห	ด	ข$_1$	ข$_2$	ห	ด	ข$_1$	ข$_2$	ห	ด	ข$_1$	ข$_2$	ห	ด	
0	●	●	●	☺	●	●	●	☺	●	●	●	☺	●	●	●	☺	OK
1	●	●	☺	●	●	●	●	☺	●	●	●	☺	●	●	●	☺	OK
2	☺	☺	●	●	●	●	●	☺	●	●	●	☺	●	●	●	☺	OK
3	●	●	☺	●	●	●	●	☺	●	●	●	☺	●	●	●	☺	OK
4	●	●	●	☺	●	●	☺	●	●	●	●	☺	●	●	●	☺	OK
5	●	●	●	☺	☺	☺	●	●	●	●	●	☺	●	●	●	☺	OK
6	●	●	●	☺	●	●	☺	●	●	●	●	☺	●	●	●	☺	OK
7	●	●	●	☺	●	●	●	☺	●	●	☺	●	●	●	●	☺	OK
8	●	●	●	☺	●	●	●	☺	☺	☺	●	●	●	●	●	☺	OK
9	●	●	●	☺	●	●	●	☺	●	●	☺	●	●	●	●	☺	OK
10	●	●	●	☺	●	●	●	☺	●	●	●	☺	●	●	☺	●	OK
11	●	●	●	☺	●	●	●	☺	●	●	●	☺	☺	☺	●	●	OK
12	●	●	●	☺	●	●	●	☺	●	●	●	☺	●	●	☺	●	OK

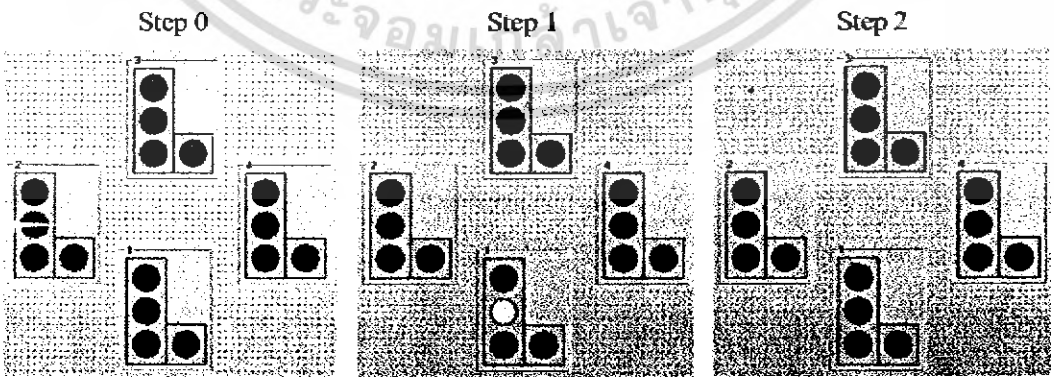
4.3 ภาพกราฟฟิกบนหน้าจอ

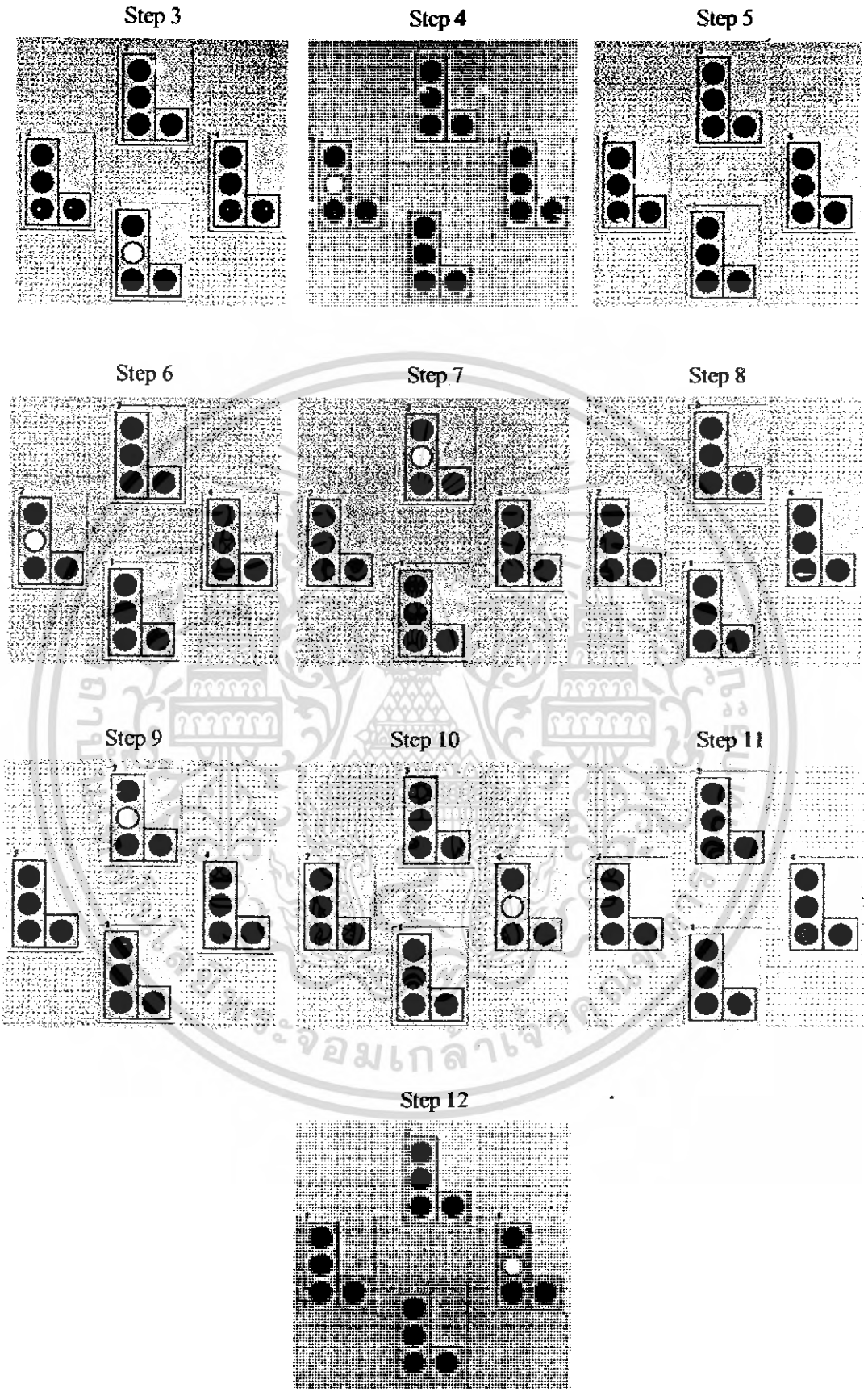
จากภาพกราฟฟิกด้านล่างสามารถสั่งงานได้ 2 แบบ คือ แบบ Auto แบบ Manual



ภาพที่ 4.1 แสดงกราฟฟิกการควบคุมไฟจราจร

4.3.1 ภาพผลการทดลองโปรแกรมในโหมด Auto จาก Step 0 ถึง Step 12





ภาพที่ 4.2 แสดงกราฟฟิกการควบคุมไฟจราจรในโหมด Auto จาก Step 0 ถึง Step 12

บทที่ 5
สรุปผลการดำเนินงานและการวิจารณ์

5.1 สรุปผลการดำเนินงาน

จากผลการดำเนินงานและการทดลองพบว่าวิธีที่จะทำให้การส่งข้อมูลได้ระยะทางไกลและมีความถูกต้องมากที่สุด คือส่งข้อมูลออกไปในที่โล่งแจ้ง อย่างไรก็ตามโอกาสที่ข้อมูลจะเกิดการรบกวนจากสัญญาณอื่นๆที่มีย่านความถี่ใกล้เคียงกันแล้วทำให้ข้อมูลผิดเพี้ยนไปอาจเกิดขึ้นได้บ้างเหมือนกัน ซึ่งระบบการจัดการข้อมูลของเครื่องรับส่งนั้น มีระบบการเข้ารหัสและถอดรหัสข้อมูลที่มีความน่าเชื่อถืออยู่ในเกณฑ์ที่จัดว่าดี โดยข้อมูลแต่ละ Byte ที่มีการรับส่งกันนั้น จะมีการตรวจสอบความถูกต้องของข้อมูลให้ด้วยแล้ว โดยข้อมูลที่รับได้จากด้าน RF นั้นรับประกันได้ว่าเป็นข้อมูลที่มีความถูกต้องแน่นอน แต่อย่างไรก็ตามการรับส่งข้อมูลนั้นมีโอกาสผิดพลาดในเรื่องของการสูญหายของข้อมูลบ้างเหมือนกัน เนื่องจากกลไกลในการรับส่งข้อมูลของเครื่อง ET-RF24G นั้น จะมีการตรวจสอบข้อมูลทุก Byte ที่รับได้จาก RF เสมอ ซึ่งถ้าพบว่ามีความผิดพลาดเกิดขึ้นจะทิ้งข้อมูล Byte นั้นไป ซึ่งในการใช้งานผู้วิจัยก็ได้สร้างกลไกลในการตรวจสอบข้อมูลที่รับส่งกันแล้ว ว่าครบถ้วนหรือไม่ซึ่งหากพบว่ามีการสูญหายของข้อมูลเกิดขึ้นก็ร้องขอให้มีการส่งข้อมูลนั้นซ้ำใหม่อีกครั้งหนึ่ง ก็จะสามารถแก้ไขปัญหาดังกล่าวได้

ความผิดพลาดอีกประการหนึ่งที่อาจเกิดขึ้นได้และไม่ควรมองข้ามคืออุปกรณ์ในส่วนสัญญาณไฟจราจรที่อาจจะมีการดับหรือเสียหาย เราก็ควรสร้างระบบการตรวจสอบความผิดพลาดของอุปกรณ์ในส่วนนี้ด้วย

5.2 วิจารณ์

จากการทดลองส่งข้อมูลที่ระยะทางต่างๆกันทั้งในที่โล่งแจ้งและสถานที่ๆ มีสิ่งกีดขวาง ความแตกต่างที่ได้จากการทดลองคือความถูกต้องของข้อมูลที่รับได้จากการส่งในระยะทางต่างๆจะไม่เท่ากันความผิดพลาดที่เกิดขึ้นนี้เราก็ต้องสร้างวิธีการตรวจสอบข้อมูลที่รับส่งกันขึ้นมาเอง ถ้าหากเกิดความผิดพลาดของข้อมูลก็ต้องร้องขอให้มีการส่งข้อมูลชุดนั้นๆ ซ้ำใหม่อีกครั้ง

ในส่วนของสัญญาณไฟจราจรก็เช่นกันถ้าหากตัวตรวจรู้จับได้ว่ามีความผิดปกติเกิดขึ้นก็ต้องนำไปเป็นเงื่อนไขของโปรแกรมด้วยเพื่อความปลอดภัยของการจราจร

บรรณานุกรม

[1] อรรถพล บุญยะโสภา, วรพจน์ กรแก้ววัฒนกุล,
ชัยวัฒน์ ลิ้มพรจิตรวิไล " เรียนรู้และปฏิบัติการเชื่อมต่อคอมพิวเตอร์กับอุปกรณ์ภายนอกผ่านพอร์ตอนุกรม " บริษัท อินโนเวตีพ เอ็กเพอริเมนต์ จำกัด

[2] คู่มือการใช้งาน RS 232 to RF-Wireless(2.4 GHz)CONVERTOR รุ่น ET-RF24G V1.0
บริษัท อีทีที จำกัด, กรุงเทพฯ

[3] สัจจะ จรัสรุ่งรวีวร " คู่มือการเขียนโปรแกรมและใช้งาน Visual Basic 6.0 "
บริษัท อินโฟเพรส จำกัด, กรุงเทพฯ, 2544

[4] จักกริช พฤษการ " การสื่อสารข้อมูลและเครือข่ายคอมพิวเตอร์ "
บริษัท สำนักพิมพ์ท้อป จำกัด, กรุงเทพฯ, 2548

ภาคผนวก