เครื่องสั่งสินค้าไร้สาย

WIRELESS ORDERING MACHINE

จัดทำโดย

สุพรรษา แขกกระโทก

Supansa Khaekkratok

สุรวิชญ์ สิงโตทอง

Suravich Singtothong

ภัททิศณีย์ เหมือนเชตุ

Phatthisinee Mounchet

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรรมศาสตร์บัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2553

เครื่องสั่งสินค้าไร้สาย

WIRELESS ORDERING MACHINE

โดย

น.ส.สุพรรษา แขกกระโทก รหัส 50011736

นายสุรวิชญ์ สิงห์โตทอง รหัส 50011758

น.ส.ภัททิศณีย์ เหมือนเชตุ รหัส 50011540

อาจารย์ที่ปรึกษา

ดร.แสงระวี บัวแก้ว

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรรมศาสตร์บัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2553

WIRELESS ORDERING MACHINE

SUPANSA KHAEKKRATOK

SURAVICH SINGTOTHONG

PHATTHISINEE MOUNCHET

A THESIS SUBMITTED IN PARTIAL FUFILLMENT

OF THE REQUIREMENT FOR THE BACHELOR'S DEGREE

OF ENGINEERING IN ELECTRONICS

KINGMONGKUT'S INSTITUE OF TECHNOLOGY LADKRABANG

รายงาน ปีการศึกษา 2553

ภาควิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง เครื่องสั่งสินค้าไร้สาย

ผู้จัดทำ

1.น.ส.สุพรรษา แขกกระโทก รหัส 50011736

2.นายสุรวิชญ์ สิงห์โตทอง รหัส 50011758

3.น.ส.ภัททิศณีย์ เหมือนเชตุ รหัส 50011540

ลงชื่อ ..... อาจารย์ที่ปรึกษา

(ดร.แสงระวี บัวแก้ว)

วันที่ 21 / มี.ค. / 54

เครื่องสั่งสินค้าไร้สาย

น.ส.สุพรรษา แขกกระโทก รหัส 50011736
นายสุรวิชญ์ สิงห์โตทอง รหัส 50011758
น.ส.ภัททิศิณีย์ เหมือนเชตุ รหัส 50011540
ดร.แสงระวี บัวแก้ว อาจารย์ที่ปรึกษา
ปีการศึกษา 2553

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้เป็นการศึกษาและประดิษฐ์เครื่องสั่งสินค้าแบบไร้สาย เพื่อให้ลูกค้าสามารถสั่งสินค้าได้เองโดยไม่ต้องผ่านพนักงาน สามารถช่วยให้ร้านค้าและลูกค้าดำเนินธุรกิจได้สะดวกและรวดเร็ว เครื่องสั่งสินค้าไร้สายจะใช้วิธีการป้อนรหัสสินค้าเข้าสู่เครื่องสั่งสินค้าซึ่งมีจอแสดงผลการสั่งสินค้า จากนั้นเครื่องจะส่งข้อมูลรายการสินค้าไปยังเครื่องรับด้วยวิธีการส่งแบบไร้สาย

WIRELESS ORDERING MACHINE

Miss.Supansa Khaekkratok ID. 50011736

Mr.Suravich Singtothong ID. 50011758

Miss.Phatthisinee Mounchet ID. 50011540

Dr.Sangrawee Buakeaw Advisor

Education Year 2010

Abstract

This thesis is to study and implement the wireless ordering machine. The function of work is to allow the customer entering the code from the menu. The main objective of this thesis is to support business smoothly and rapidly. The result will be display on the LCD monitor then the machine will send this data to the receiver by wireless communication.

กิตติกรรมประกาศ

ปริญญานิพนธ์นี้ได้รับการสนับสนุนงบประมาณจากสาขาวิชาอิเล็กทรอนิกส์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีเจ้าคุณทหารลาดกระบัง

ขอขอบพระคุณอาจารย์แสงระวี บัวแก้วที่ได้ให้คำปรึกษา ตลอดจนคำแนะนำต่างๆที่เป็นประโยชน์

ขอขอบพระคุณ คณาจารย์ทุกท่าน ที่ได้พร่ำสอน ให้วิชาความรู้ต่างๆที่เป็นประโยชน์ ทั้งในการทำโปรเจคและการทำงาน

สุดท้ายนี้ขอขอบคุณเพื่อนๆในห้องโปรเจคทุกคน ที่เป็นกำลังใจ และให้คำแนะนำ

.....

(นางสาวสุพรรษา แขกกระโทก)

.....

(นายนายสุรวิชญ์ สิงห์โตทอง)

.....

(นางสาวภัททิศิณีย์ เหมือนเชตุ)

ผู้จัดทำ

สารบัญ

หน้า

สารบัญตาราง

ตารางที่ หน้า

สารบัญภาพ

รูปที่ หน้า

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มาของโครงงาน

ปัจจุบันเทคโนโลยีต่างๆในชีวิตประจำวันได้มีการพัฒนาก้าวหน้าไปอย่างรวดเร็ว เป็นผลทำให้มนุษย์แสวงหาและอาศัยเทคโนโลยีเหล่านี้ ด้วยเหตุผลนี้เองจึงทำให้เกิดแนวคิดที่จะสร้างสิ่งประดิษฐ์ที่ใช้ในชีวิตประจำวันตามคุณลักษณะข้างต้น

ปริญญานิพนธ์ฉบับนี้ได้ทำการศึกษาและพัฒนาเครื่องสั่งสินค้าไร้สาย(WIRELESS ORDERING MACHINE) โดยมีหลักการทำงานคือ เครื่องสั่งเครื่องดื่มเหล่านี้ จะถูกวางบนโต๊ะทุกโต๊ะในร้านอาหาร เมื่อลูกค้าเข้ามาถึง ก็จะเลือกรายการเครื่องดื่มจากเมนูที่วางอยู่บนโต๊ะ โดยในเมนูนี้จะมีรหัสรายการเครื่องดื่มแต่ละชนิดอยู่ จากนั้นลูกค้าจึงทำการสั่งเครื่องดื่ม โดยกดหมายเลขรหัสรายการเครื่องดื่มลงในเครื่องสั่ง ซึ่งเมื่อกดรหัสแล้ว บนหน้าจอของเครื่องสั่งเครื่องดื่มจะปรากฏชื่อรายการที่ได้ทำการกดไป และเครื่องจะให้ใส่จำนวนเครื่องดื่มที่ต้องการ เมื่อสั่งเครื่องดื่มครบทุกรายการแล้ว รายการเครื่องดื่มทั้งหมดจะถูกส่งไปสู่เครื่องรับ ทางด้านเครื่องรับใช้คอมพิวเตอร์ในการแสดงผลและประมวลผลโดยใช้โปรแกรม Visual Basic

1.2 วัตถุประสงค์ของโครงงาน

เพื่อวิเคราะห์และออกแบบโปรแกรมประยุกต์การใช้งานของPic16F877A, XBeeและ Visual Basic

1.3 ขอบเขตของโครงงาน

1.3.1 ออกแบบโปรแกรมประยุกต์สำหรับการรับส่งรายการสินค้าในร้านกาแฟ

1.3.2 นำโปรแกรมประยุกต์ที่ออกแบบมาทดลองใช้งาน

1.3.3 วิเคราะห์และแก้ปัญหาที่เกิดจากการทดลอง

1.4 วิธีดำเนินการ

1.4.1 เสนอโครงงาน

1.4.2 ศึกษาการเขียนโปรแกรมด้วย MikroC

1.4.3 ศึกษาการเขียนโปรแกรมด้วย Visual Basic

1.4.4 ศึกษาการใช้และการเซตค่า XBee

1.4.5 กำหนดปัญหาของโปรแกรม

1.4.6 วิเคราะห์และออกแบบโปรแกรม

1.4.7 ทดสอบโปรแกรม

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1.5.1 ได้รับความรู้เกี่ยวกับ microcontroller ทั้งทางด้านโครงสร้าง และภาษาที่ใช้เขียนโปรแกรม

1.5.2 ได้เรียนรู้โครงสร้างและการทำงาน ของ PIC16F877A

1.5.3 สามารถเขียนประยุกต์โดยใช้ภาษา C กับ PIC16F877A ได้

1.5.4 เรียนรู้โครงสร้างการเขียนโปรแกรมภาษา Mikro C และสามารถเขียนประยุกต์ใช้กับโครงงานนี้ได้

1.5.5 เรียนรู้การเชื่อมต่อระหว่าง Keypad, LCD กับ พอร์ตต่างๆของ PIC16F877A มีทักษะในการทำงานร่วมกับผู้อื่น รู้จักการแบ่งหน้าที่การทำงาน และการบริหารเวลาให้เกิดประโยชน์สูงสุด

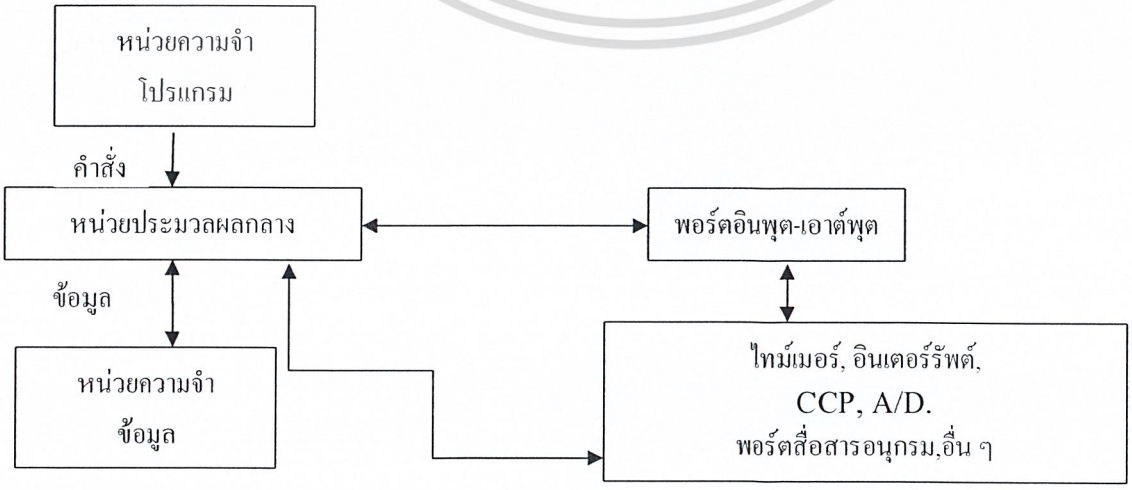
1.5.6 ฝึกให้มีความอดทน และความเพียรพยายาม ในการแก้ปัญหาต่างๆ ระหว่างการทดลอง และศึกษาค้นคว้าความรู้ใหม่ๆเพิ่มเติมเสมอ เพื่อให้ได้ผลงานดีที่สุด

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

2.1 ความหมายของ PIC

PIC ย่อมาจากคำว่า Peripheral Interface Controller เป็นไมโครคอนโทรลเลอร์ที่ผลิตโดย บริษัท ไมโครชิพ (Microchip Technology. Inc.) ชื่อนี้ถูกใช้ครั้งแรกในปี ค.ศ. 1977 ซึ่งในขณะนั้น บริษัท ไมโครชิพ ยังใช้ชื่อว่า เจนเนอรอล อินสตรูเมนท์ (General Instrument) PIC เบอร์แรกที่ผลิตออกมาก็คือ PIC1650 ซึ่งมีโครงสร้างสถาปัตยกรรมไม่ต่างจาก PIC ในปัจจุบันมากนัก แนวคิดของการสร้าง PIC คือการพยายามรวมเอาทุกอย่างไว้ในชิพตัวเดียวไม่ว่าจะเป็น หน่วยประมวลผลกลาง, หน่วยความจำโปรแกรม, หน่วยความจำข้อมูล, EEPROM, I^2C, CCP, A/D ฯลฯ โดยไม่จำเป็นต้องต่ออุปกรณ์เสริมจากภายนอก ผลที่ตามมาก็คือแผ่น PCB จะมีขนาดเล็ก และอุปกรณ์ที่ใช้จะไม่มาก บางงานอาจใช้แค่ PIC เพียงตัวเดียว โดยไม่ต้องใช้ชิพอื่น ๆ มาต่อเพิ่มเติมเลย

PIC เป็นไมโครคอนโทรลเลอร์ที่มีสถาปัตยกรรมแบบฮาวาร์ด (Harvard Architecture) หรือที่เรียกว่า ไมโครคอนโทรลเลอร์แบบ RISC (Reduced Instruction Set Computer) คือ มีการทำงานที่มีจำนวนชุดคำสั่งน้อย แต่ละคำสั่งจะทำงานแบบง่าย ๆ ทำให้ความเร็วในการทำงานแต่ละคำสั่งสูง หนึ่งคำสั่งของ PIC จะมีขนาด 14 บิต เป็นผลให้จำนวนชุดคำสั่งของ PIC มีน้อยนั่นเอง ในปัจจุบัน PIC เป็นไมโครคอนโทรลเลอร์ตระกูลที่เป็นยอดนิยมตระกูลหนึ่ง เพราะเป็นไมโครคอนโทรลเลอร์ที่ประสิทธิภาพสูง ราคาถูก และก็ยังมีให้เลือกใช้มากมายหลายเบอร์อีกด้วย ลักษณะโครงสร้างหลัก ๆ ของ PIC แสดงดังรูป



รูปที่2.1โครงสร้างหลัก ๆ ของ PIC

2.1.1 ชนิดของไมโครคอนโทรลเลอร์

ไมโครคอนโทรลเลอร์ PIC แบ่งออกตามชนิดของหน่วยงานความจำโปรแกรมได้ 3 ประเภท คือ

1. OTP (One-Time Programmable) เป็นชิพประเภทที่มีราคาถูกที่สุด แต่สามารถทำการโปรแกรมได้แค่ครั้งเดียวเท่านั้น ซึ่งชิพประเภทนี้จะใช้ในกรณีที่พัฒนาโปรแกรมจนไม่พบจุดบกพร่องของโปรแกรมอีกแล้ว และต้องการใช้งานจำนวนมาก ๆ เพราะมีราคาถูก ชิพประเภทนี้ยกตัวอย่างเช่น PIC12C509, PIC12C672, PIC19C84, PIC16C74

2. EPROM (Erasable Programmable ROM) เป็นชิพประเภทที่เมื่อเขียนโปรแกรมเข้าไปแล้วสามารถทำการลงโปรแกรมเดิมได้ด้วยแสง UV (Ultra Violet) ดังนั้นที่ด้านบนของชิพจะมีกรอบกระจกเพื่อให้แสง UV สามารถส่องผ่านเข้าไปในตัวชิพได้ ชิพแบบนี้จะมีตัวอักษร JW เขียนกำกับอยู่ ชิพประเภทนี้ยกตัวอย่างเช่น PIC16C57/JW, PIC16C72A/JW, PIC16C74/JW

3. EEPROM (electronically erasable programmable ROM) หรือที่เรียกกันทั่วไปว่าแบบแฟลช (Flash) เป็นชิพประเภทที่สามารถเขียนและลบโปรแกรมด้วยไฟฟ้า และสามารถลบและเขียนใหม่ได้หลายพันครั้ง ทำให้ชิพประเภทนี้เป็นที่นิยมใช้กันมากที่สุดใน 3 ประเภท ชิพประเภทนี้ก็ยกตัวอย่างเช่น PIC12F510, PIC16F84, PIC16F877, PIC18F1220, PIC18F4331, PIC24HF12GP201, PIC32MX340F128H

2.2 ความหมายของเบอร์ของ PIC

เคยไหมที่เวลาเราดูเบอร์ PIC แล้วงง เกิดอาการสงสัยและไม่รู้ว่าจะเลือกใช้ PIC เบอร์ไหนดี เพราะเบอร์ของ PIC มีมากมายหลายเบอร์ แต่ที่จริงแล้วเบอร์ของ PIC สามารถบอกข้อมูลได้หลายอย่างเลยทีเดียว ยกตัวอย่างเช่น PIC เบอร์ 18F452-I/P จะบอกรายละเอียดต่าง ๆ ดังนี้

18 ตัวเลขสองตัวแรกนี้คือ Family number ของ PIC ซึ่งจะมีตั้งแต่เลข 10, 12, 14, 16, 17, 18, 24 และ 32 นอกจากนั้นยังมีตัวเลข 30 และ 33 สำหรับ dsPIC (digital signal PIC) อีกด้วย ซึ่งตัวเลขเหล่านี้จะคล้ายเป็นตัวเลขบ่งบอกวิวัฒนาการของ PIC โดย PIC ที่มี Family number 10, 12, 14, 16, 18 เป็นไมโครคอนโทรลเลอร์ขนาด 8 บิต Family number 24, 30, 33 เป็นไมโครคอนโทรลเลอร์ขนาด 16 บิต และ Family number 32 เป็นไมโครคอนโทรลเลอร์ขนาด 32 บิต

F ตัวอักษรตัวนี้ หมายถึง ลักษณะของหน่วยความจำโปรแกรมว่า มีลักษณะแบบใด ซึ่งสามารถแสดงได้ดังตาราง

ตัวอักษร	ลักษณะของหน่วยความจำโปรแกรม
C	EPROM
CR	ROM
CE	One-time programmable EPROM และ EEPROM
F	Flash
HV	High Voltage (15V)
LF	Low Voltage Flash
LC	Low Voltage One-time programmable
LCR	Low Voltage ROM

ตารางที่ 2.2.1 ความหมายของตัวอักษรบอกลักษณะของหน่วยความจำโปรแกรม

452 ตัวเลขที่อยู่ถัดจากตัวอักษรนี้คือ ตัวเลขบอกประเภทของ PIC เบอร์นั้น ๆ

-XX กรณีที่มีตัวเลขต่อจากเครื่องหมายขีด (-) ตัวเลขนี้จะบอกถึงความถี่สัญญาณนาฬิกาสูงสุดที่ PIC เบอร์นั้น ๆ ทำงานได้

I ตัวอักษรตัวนี้จะบอกถึงช่วงอุณหภูมิที่ PIC เบอร์นั้น ๆ สามารถทำงานได้ ซึ่งแสดงดังตาราง

ตัวอักษร	ช่วงอุณหภูมิที่ PIC สามารถทำงานได้
C	Commercial 0 $^{\circ}$ C ถึง +85 $^{\circ}$ C
I	Industrial -40 $^{\circ}$ C ถึง +85 $^{\circ}$ C
E	Extended -40 $^{\circ}$ C ถึง +125 $^{\circ}$ C

ตารางที่ 2.2.2 ความหมายของตัวอักษรบอกช่วงอุณหภูมิที่ PIC สามารถทำงานได้

/P เป็นตัวอักษรบอกลักษณะตัวถัง (Package) ของ PIC ว่าเป็นตัวถังแบบใด ซึ่งแสดงดังตาราง

ตัวอักษร	ลักษณะตัวถัง
JW	Ceramic window (EEPROM only)
P	Plastic DIP
SN, OA, SM, SL, OD, SO, SI	SOIC-plastic small outline (surface mount)
PO	QFP-Plastic quad flat pack surface mount

SS	SSOP-plastic shrink small outline surface mount
ML	Chip scale package
ST	TSSOP-Plastic thin shrink small outline surface mount
PT	TOEP-plastic thin quad flat pack

ตารางที่ 2.2.3 ตารางแสดงอักษรบอกลักษณะ Package ของ Pic

2.3 ไมโครคอนโทรลเลอร์ (PIC16F877A)

ชิพ	หน่วยความจำโปรแกรม	หน่วยความจำข้อมูล		CAN Module	จำนวน I/O (บิต)	OSC (MHz)	Timer	PLL
		RAM	EEPROM					
16F84	1K Word	68	64	ไม่มี	13	4-10	1	ไม่มี
16F877	8K Word	368	256	ไม่มี	33	4-20	3	ไม่มี
18F442	16 KB	768	256	ไม่มี	34	40	4	มี
18F458	32 KB	1536	256	มี	34	40	4	มี

ตารางที่ 2.3 ภาพรวมของ PIC16Fxxx/18Fxxx

หมายเหตุ

1. 16F84/877 นั้น ใช้หน่วยความจำโปรแกรมขนาด 14 บิต ต่อ 1 คำ (Word)
2. PLL (Phase Lock Loop) คือ วงจรที่สร้างความถี่ของสัญญาณนาฬิกาเป็น 4 เท่าจาก XTAL ดังนั้น ถ้าเลือกโหมด PLL ก็จะใช้ XTAL ได้ไม่เกิน 10MHz

2.3.1 สัญญาณนาฬิกา

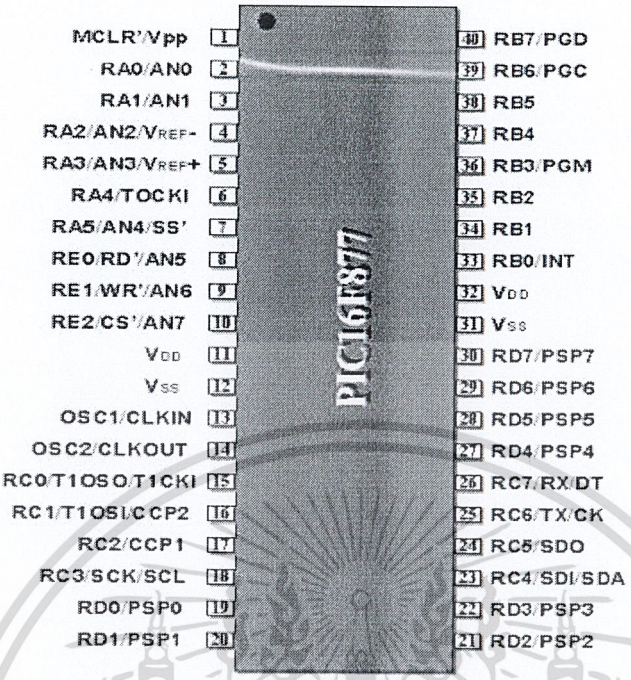
PIC จะใช้สัญญาณนาฬิกา โดยมองเป็นลักษณะของ วงรอบ (Cycle) ซึ่งระบุเอาไว้ว่า 1 คำสั่งนั้นจะประกอบไปด้วย 1-2 วงรอบ โดยแต่ละวงรอบนั้นจะแบ่งเป็น 4 ส่วน คือ Q1, Q2, Q3 และ Q4 ด้วยเหตุนี้ ความเร็วโดยรวมของ PIC จึงเท่ากับ ค่าความถี่ของสัญญาณนาฬิกา หาร ด้วย 4

$$1 \text{ cycle} = Q1+Q2+Q3+Q4 = \frac{XTAL}{4}$$

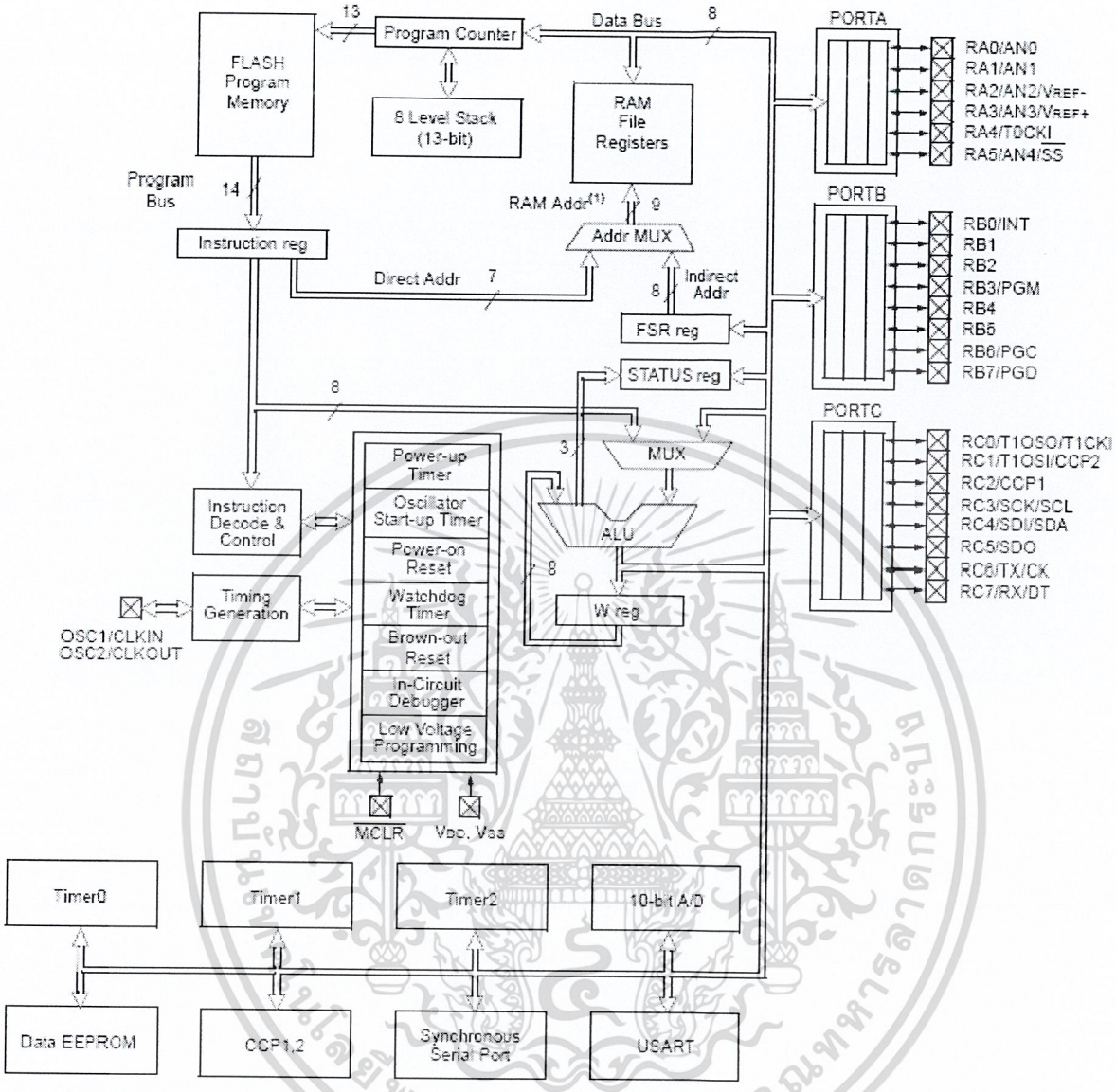
ส่วนรุ่น 18Fxxx นั้นจะมีความสามารถพิเศษคือ สามารถสร้างสัญญาณนาฬิกาเป็น 4 เท่าของ XTAL โดยใช้วงจรเฟสล็อกลูป (อยู่ในตัวไมโครคอนโทรลเลอร์) ด้วยเหตุนี้ ถ้าเราใช้ XTAL 10 MHz ความเร็วสูงสุดของชิพจึงเป็น 40MHz ซึ่งถ้าหารด้วย 4 ก็ประมาณได้ว่า ทำงานที่ความเร็วประมาณ 10 ล้านคำสั่งต่อวินาที (อย่าลืมนะครับว่าถ้าเป็นพวกคำสั่งกระโดดจะใช้มากกว่า 1 cycle

2.3.2 คุณสมบัติของPIC 16F877A

1. มีคำสั่งให้ใช้งาน 35 คำสั่ง
2. คำสั่งหนึ่งๆใช้เวลาทำงาน 1 ถึง 2 Cycle
3. ทำงานได้สูงสุดที่ 20MHz (PIC16F877-20/P)
4. ทำงานแบบ Pipe-line (มี 2 ท่อ) ทำให้ ณ เวลาหนึ่งทำงาน 2 อย่างพร้อมๆกันได้
5. หน่วยความจำโปรแกรมเป็นแบบ Flash มีขนาด 8KWord (1 word=14 บิต)
6. มี RAM ขนาด 368 ไบต์
7. มี EEPROM ขนาด 256 ไบต์
8. ตอบสนองกับอินเตอร์รัพได้ทั้งหมด 14 แหล่ง
9. มี Stack ให้ใช้ได้สูงสุด 8 ระดับ
10. มีระบบ Power On Reset, Power Up Timer, Oscillator Start-up timer
11. Watchdog timer
12. มีระบบ Code Protection
13. ประหยัดพลังงาน
14. สัญญาณนาฬิกามีหลายโหมดให้เลือกใช้งาน คือ อาจจะใช้ XTAL หรือ วงจร RC ก็ได้
15. สามารถโปรแกรมด้วยไฟ +5VDC ได้
16. ใช้การโปรแกรมแบบ In-Circuit Serial Programming
17. ทำงานที่ไฟเลี้ยง 2VDC ถึง 5.5VDC
18. Current Sink และ Current Source อยู่ที่ 25mA
19. มี Timer/Counter 3 ตัว
20. มีโมดูล Capture/Compare/PWM มี 2 ชุด
21. มี A-TO-D Converter แบบ 10 บิต จำนวน 8 ช่องนำเข้า ในตัวเอง
22. มีระบบ USART สำหรับต่อกับ การสื่อสารแบบ RS232 หรือดีกว่า
23. มีระบบตรวจระดับไปเลี้ยง (Brown-out reset)
24. มี I/O พอร์ตทั้งหมด 5 พอร์ต

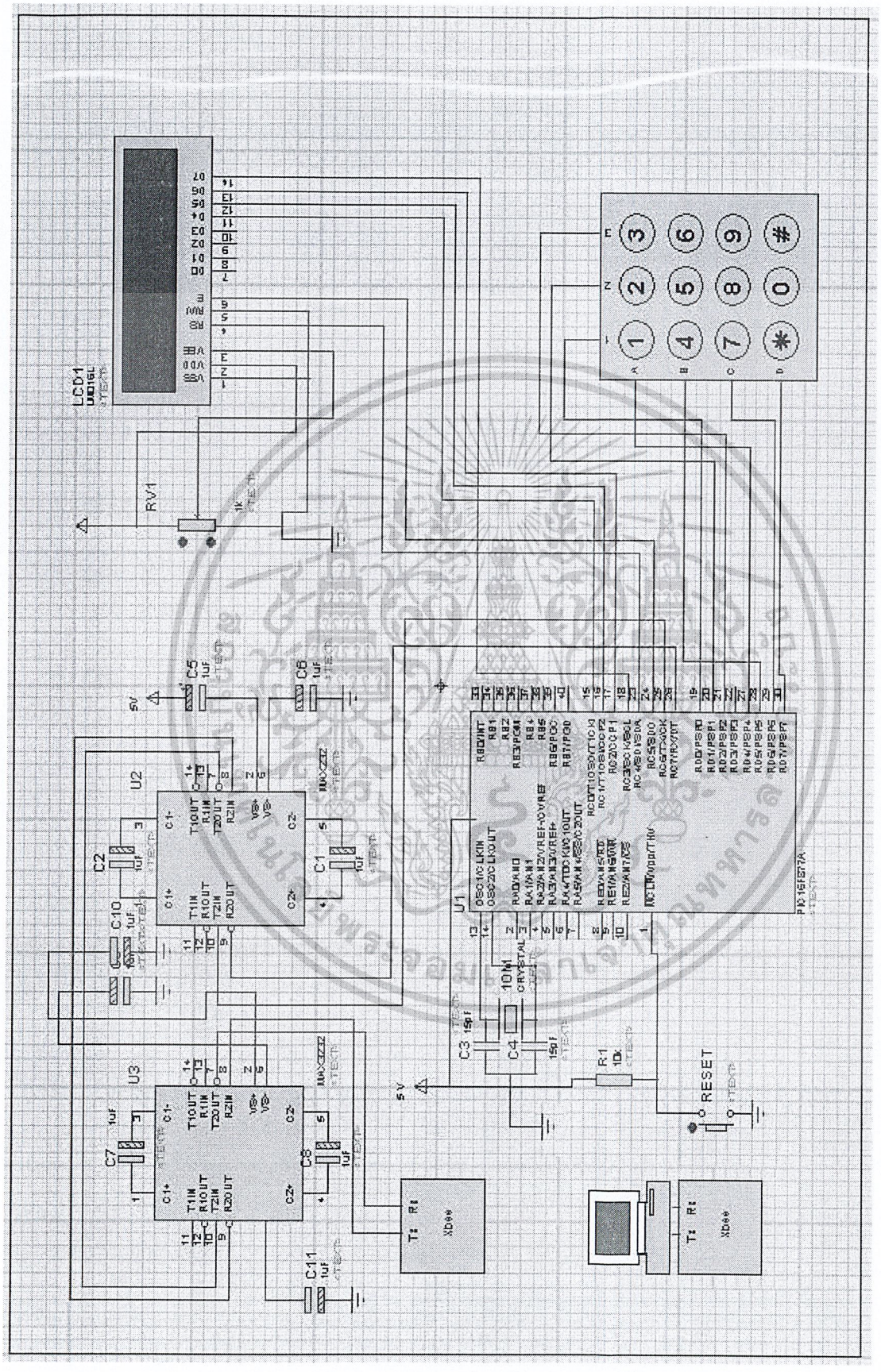


รูปที่ 2.3.2.1 ภาพแสดงโครงสร้างภายนอก



รูปที่ 2.3.2.2 แสดงโครงสร้างภายใน

จากผังจะมี Register สำคัญๆ คือ W ซึ่งเป็น Register ที่ใช้ในการทำเป็น Input ให้กับ ALU และเป็นตัวเก็บผลลัพธ์จากการทำงานของ ALU, STATUS เป็น Register ที่ใช้เก็บสถานะ การทำงานของคำสั่ง ว่าเมื่อคำสั่งทำงานเสร็จแล้วเกิดอะไรขึ้นมาบ้าง ซึ่งมีประโยชน์ในการเขียนโปรแกรมแบบมีเงื่อนไข, PC หรือ Program Counter เป็น Register อีกตัวหนึ่งที่มีความสำคัญ เนื่องจากใช้สำหรับเป็นตัวชี้ว่า คำสั่งที่จะนำมาประมวลผลนั้นอยู่ ณ ตำแหน่งใดในหน่วยความจำ



รูปที่ 2.3.2.3 แสดงการเชื่อมต่อของวงจร

2.3.3 หน้าที่ของพอร์ตที่ใช้งาน

RA0-RA3 และ RA5 จะใช้งานเป็น I/O ปกติ และทำหน้าที่เป็นขาอินพุตของสัญญาณอนาล็อก (AN0-AN4)

RA4 เป็นขา I/O

RA6/OSC2/CLKO ทำหน้าที่ในหลายส่วน คือ เป็นขา OSC2 และ CLKO จะนำมาใช้เป็นขาสัญญาณ I/O ได้ก็ต่อเมื่อใช้คริสตอลออสซิลเลเตอร์ แบบที่เป็นโมดูลสำเร็จสามารถต่อเข้ากับขา OSC1/CLKIN ได้เลยโดยที่ไม่ต้องต่อกับขา RA6/OSC2 ทำให้ ขา RA6 ว่างและนำไปใช้เป็น I/O ได้

RB0-RB7 สามารถใช้งานเป็น I/O แต่มีคุณสมบัติ พิเศษคือวงจรมีพูลอัพ (Pull-Up) ภายใน และ เป็น แหล่งกำเนิดสัญญาณอินเตอรัพท์ต่างๆ ดังนี้

- RB0/INT0 เป็นขาสัญญาณอินเตอร์รัพท์ภายนอก 0
- RB1/INT1 เป็นขาสัญญาณอินเตอร์รัพท์ภายนอก 1
- RB2/INT2 เป็นขาสัญญาณอินเตอร์รัพท์ภายนอก 2
- RB3/INT3 เป็นขาสัญญาณอินเตอร์รัพท์ภายนอก 3
- RB4-RB7 เป็นขาที่สามารถกำเนิดสัญญาณอินเตอร์รัพท์ได้

RC5/SD0 เป็นขาอินพุต-เอาต์พุตทั่วไป และเอาต์พุตข้อมูลของ SPI

RS6/TX/CX เป็นขาอินพุต-เอาต์พุตทั่วไป, เอาต์พุตข้อมูลของ USART และอินพุต-เอาต์พุตสัญญาณ Clock แบบซิงโครนัสของ USART

RS7/RX/DT เป็นขาอินพุต-เอาต์พุตทั่วไป, เอาต์พุตข้อมูลของ USART และอินพุต-เอาต์พุตสัญญาณ Clock แบบซิงโครนัสของ USART

RD0-RD7 เป็นขาอินพุต-เอาต์พุตทั่วไป และอินพุต-เอาต์พุตของ PSP บิต 0-7

-RD0/PSP0 เป็นขาอินพุต-เอาต์พุตของ PSP บิต 0

-RD1/PSP1 เป็นขาอินพุต-เอาต์พุตของ PSP บิต 1

-RD2/PSP2 เป็นขาอินพุต-เอาต์พุตของ PSP บิต 2

-RD3/PSP3 เป็นขาอินพุต-เอาต์พุตของ PSP บิต 3

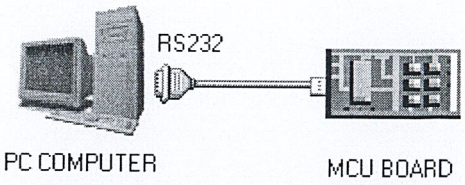
-RD4/PSP4 เป็นขาอินพุต-เอาต์พุตของ PSP บิต 4

-RD5/PSP5 เป็นขาอินพุต-เอาต์พุตของ PSP บิต 5

-RD6/PSP6 เป็นขาอินพุต-เอาต์พุตของ PSP บิต 6

-RD7/PSP7 เป็นขาอินพุต-เอาต์พุตของ PSP บิต 7

2.4 พอร์ตอนุกรม RS-232



รูปที่ 2.4 พอร์ตอนุกรม RS-232

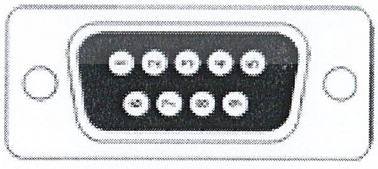
การสื่อสารแบบอนุกรม นับว่ามีความสำคัญ ต่อการใช้งาน ไมโครคอนโทรลเลอร์มาก เพราะสามารถใช้แป้นพิมพ์ และจอภาพของ PC เป็น อินพุต และ เอาต์พุต ในการติดต่อ หรือ ควบคุม ไมโครคอนโทรลเลอร์ ด้วยสัญญาณอย่างน้อย เพียง 3 เส้นเท่านั้น คือ

- สายส่งสัญญาณ TX
- สายรับสัญญาณ RX
- และสาย GND

โดยปกติพอร์ตอนุกรม RS-232 จะสามารถต่อสายได้ยาว 50 ฟุต โดยประมาณ ขึ้นอยู่กับ ชนิดของสายสัญญาณ, ระยะทาง, และ ปริมาณ สัญญาณ รบกวน

2.4.1 พอร์ตอนุกรมของ DB9

แสดงการจัดขา ของคอนเน็กเตอร์ อนุกรมแบบ DB9 และหน้าที่การใช้งานต่างๆ



รูปที่2.4.1.1 DB9 ตัวผู้ เมื่อมองจากด้านหลัง

Pin	Description	Type
1	Data Carrier Detect (DCD)	Input
2	Received Data (RXD)	Input
3	Transmitted Data (TXD)	Output
4	Data Terminal Ready (DTR)	Output
5	Signal Ground (GND)	Input
6	Data Set Ready (DSR)	Input
7	Request To Send (RTS)	Output
8	Clear to Send (CTS)	Input
9	Ring Indicator (RI)	Input

การเชื่อมต่ออุปกรณ์ภายนอกผ่าน DB9 แบบ Null modem

การต่ออุปกรณ์ภายนอกผ่าน DB9 แบบ 3 เส้น

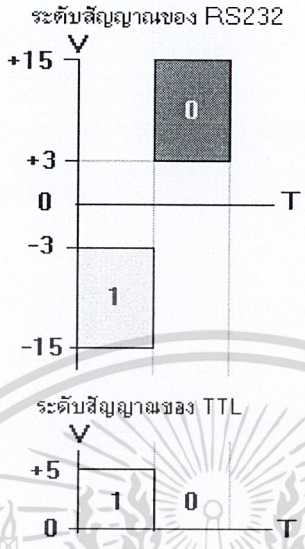
รูปที่2.4.1.2 การเชื่อมต่ออุปกรณ์อุกรณ์ภายนอกเข้ากับคอมพิวเตอร์ด้วยสาย DB9

การทำงานของขาสัญญาณ DB9

TXD เป็นขาที่ใช้ส่งข้อมูล

RXD เป็นขาที่ใช้รับข้อมูล

2.4.2 ระดับสัญญาณของ RS232



รูปที่2.4.2 ระดับสัญญาณของ RS232C และระดับสัญญาณของ TTL

สัญญาณรบกวนที่เกิดขึ้น ในสายนำสัญญาณ มักจะมีแรงดันเป็นบวก เมื่อเทียบกับ Ground

- เพื่อป้องกันสัญญาณรบกวนนี้ จึงออกแบบแรงดัน ของโลจิก "1" เป็นลบ คืออยู่ในช่วง -3V ถึง -15V ส่วนแรงดัน ของโลจิก "0" อยู่ในช่วง +3V ถึง +15V

- และเหตุที่ ระดับสัญญาณ ของ RS232 อยู่ในช่วง +15V ถึง -15V ก็เพื่อให้ต่อสายสัญญาณไปได้ไกลขึ้น

ดังนั้นจึงจำเป็นจะต้องมีวงจรเปลี่ยนระดับแรงดันของ RS232 มาเป็นระดับแรงดันของ TTL

2.4.3 อัตราการส่งข้อมูล (Baud rate)

- คือความเร็วของการรับ-ส่งข้อมูล เป็นจำนวนบิทต่อวินาทีเช่น 300, 1,200, 2,400, 4,800 , 9,600 ,14,400 ,19,200, 38,400 ,56,000 เป็นต้น

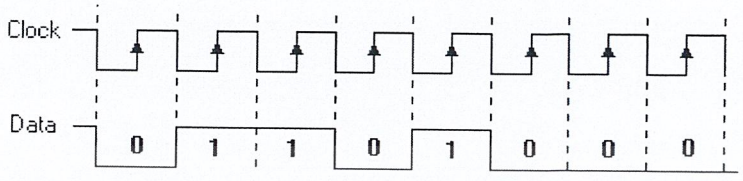
- การเลือกอัตราการส่งข้อมูลขึ้นอยู่กับ ชนิดของสายสัญญาณ, ระยะทาง,และปริมาณสัญญาณรบกวน

2.4.4 รูปแบบการสื่อสารแบบอนุกรม

มีด้วยกันอยู่ 2 แบบ คือแบบซิงโครนัส (Synchronous) และแบบอะซิงโครนัส(Asynchronous)

การสื่อสารแบบซิงโครนัส (Synchronous)

การรับส่งข้อมูล จะมีสัญญาณนาฬิกา ซึ่งเป็นตัวกำหนด จังหวะเวลา การส่งข้อมูล ร่วมอยู่ด้วย อีกเส้นหนึ่ง ใช้คู่กับสัญญาณข้อมูล ตัวอย่างเช่น การส่งสัญญาณจากคีย์บอร์ด



รูปที่ 2.4.4.1 การสื่อสารแบบซิงโครนัส (Synchronous)

การสื่อสารแบบอะซิงโครนัส (Asynchronous)

การรับส่งข้อมูล โดยที่ไม่จำเป็นต้อง มีสัญญาณนาฬิกา ร่วมด้วย แต่จะใช้ให้ตัวส่ง และตัวรับ มีอัตราส่งข้อมูล ที่เท่ากัน

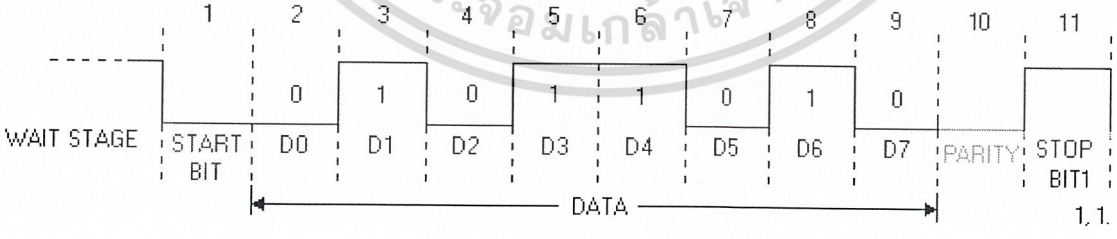
รูปแบบข้อมูลแบบอะซิงโครนัส ประกอบด้วย 4 ส่วนคือ

1 บิตเริ่มต้น (Start bit) มีขนาด1 บิต

2 บิตข้อมูล (Data) มีขนาด 5,6,7 หรือ 8 บิต

3 บิตตรวจสอบพาริตี้ (Parity bit) มีขนาด 1 บิตหรือไม่มี

4 บิตหยุด (Stop bit) มีขนาด 1, 1.5, 2 บิต



รูปที่ 2.4.4.2 การสื่อสารแบบอะซิงโครนัส (Asynchronous)

- เมื่อไม่มีการส่งข้อมูล ขา data จะมีสถานะเป็นโลจิก "1" หรือ สถานะหยุดรอ (Waiting stage)

- เมื่อเริ่มต้นส่งข้อมูลจะให้ขา data เป็นโลจิก "0" เป็นจำนวน 1 บิต เรียกว่าบิตเริ่มต้น (Start bit)

- จากนั้นก็จะเริ่มต้นส่งข้อมูล โดยส่งบิตต่ำไปก่อน (LSB)
- แล้วตามด้วยพาริตี้บิต (จะมีหรือไม่ก็ได้ ขึ้นอยู่กับการติดตั้งค่า ของทั้งสองฝ่าย)
- สุดท้ายตามด้วยโลจิก "1" อย่างน้อย 1 บิต (มีขนาด 1, 1.5, หรือ 2 บิต) เพื่อแสดงว่าสิ้นสุดข้อมูล

การรับและส่งข้อมูลแบบอนุกรมยังแบ่งออกเป็นลักษณะการใช้งานได้ 3 แบบคือ

1). แบบซิมเพลกซ์ (Simplex) เป็นการส่ง หรือรับข้อมูล แบบทิศทางเดียว เท่านั้น

2). แบบฮาล์ฟดูเพลกซ์ (Half Duplex) เป็นการส่งและรับข้อมูลแบบสลับกัน คือเมื่อด้านหนึ่งส่ง อีกด้านหนึ่ง เป็นฝ่ายรับ สลับกัน ไม่สามารถรับ-ส่งในเวลาเดียวกันได้

3). แบบฟลูดูเพลกซ์ (Full Duplex) สามารถรับ-ส่งข้อมูลในเวลาเดียวกันได้

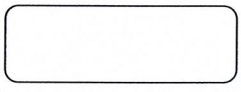
2.5 การาเขียนอัลกอริทึม

"อัลกอริทึม" คือ ลำดับขั้นตอนของกระบวนการต่าง ๆ ไม่ว่าจะเป็นการคำนวณ การเปรียบเทียบ การตัดสินใจเลือก ซึ่งถูกเขียนเรียงลำดับกันไว้ เพื่อใช้เป็นเส้นทางในการแก้ปัญหาต่าง ๆ โดยในการเขียนโปรแกรมนั้น อัลกอริทึมเปรียบเสมือนเป็นหัวใจของการเขียนโปรแกรมเลยทีเดียว ถ้าเขียนอัลกอริทึมผิดก็คือการวางแผนที่ผิดพลาด โค้ดโปรแกรมที่เขียนขึ้นมาก็จะผิดตามไปด้วย ทำให้ชิ้นงานที่สร้างขึ้นมาไม่ได้ให้เอาต์พุตตามที่ต้องการ ในการเขียนโปรแกรมบนไมโครคอนโทรลเลอร์ อัลกอริทึมก็คือตัวแทนของเหตุการณ์ที่เกิดขึ้นภายในชิพไมโครคอนโทรลเลอร์ ซึ่งมีการเขียนอัลกอริทึมอยู่ 2 แบบ คือการเขียนอัลกอริทึมแบบโฟลว์ชาร์ต และการเขียนอัลกอริทึมแบบ PDL

2.5.1 การเขียนอัลกอริทึมแบบโฟลว์ชาร์ต

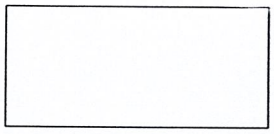
การเขียนอัลกอริทึมแบบโฟลว์ชาร์ตคือ การเขียนอัลกอริทึมโดยใช้สัญลักษณ์รูปภาพและลูกศรแทนลำดับขั้นตอนการทำงานของโปรแกรม ตั้งแต่จุดเริ่มต้นจนถึงจุดสิ้นสุดโปรแกรม โดยระหว่างทางจะแสดงให้เห็นถึงอินพุต เอาต์พุต รวมถึงการตัดสินใจทั้งหมดของโปรแกรม สัญลักษณ์ที่ใช้ในการเขียนอัลกอริทึมแบบโฟลว์ชาร์ต มีดังต่อไปนี้

จุดเริ่มต้น/สิ้นสุด (Terminator)



เป็นสัญลักษณ์แทนจุดเริ่มต้นและจุดสิ้นสุดของโปรแกรม โดยใส่คำว่า START หรือ STOP ภายในสัญลักษณ์

กระบวนการ (Process)

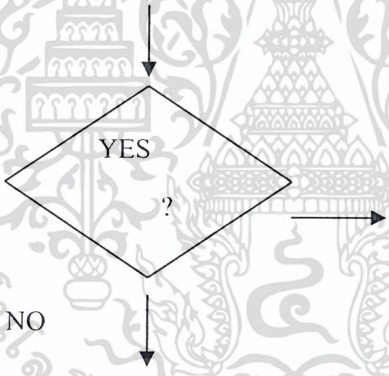


เป็นสัญลักษณ์แทนการกระทำภายในโปรแกรม โดยใส่คำสั่งการทำงานภายในสัญลักษณ์

นำเข้า/ส่งออก (Input/Output)

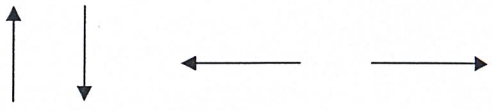
เป็นสัญลักษณ์แทนการรับข้อมูลอินพุตหรือส่งข้อมูลเอาต์พุต โดยใส่ข้อมูลภายในสัญลักษณ์

การตัดสินใจ (Decision)



เป็นสัญลักษณ์แทนการตัดสินใจจากคำถามภายในสัญลักษณ์ โดยคำตอบที่ได้คือจริง (YES) หรือเท็จ (NO) ซึ่งต้องเลือกทางเดินต่อไปจากคำตอบเพียงทางเดียวเท่านั้น

ทิศทาง (Direction)



เป็นสัญลักษณ์ลูกศรแทนทิศทางการไหลของข้อมูลหรือเส้นทางการเดินของโปรแกรม โดยหัวลูกศรเป็นตัวกำหนดทิศทาง

2.5.2 การเขียนอัลกอริทึมแบบ PDL

การเขียนอัลกอริทึมแบบ PDL (Program Description Language) คือ การเขียนอัลกอริทึมโดยใช้รูปแบบของคำภาษาอังกฤษ มาบรรยายกระบวนการลำดับขั้นตอนการทำงานของโปรแกรมเป็นบรรทัด ๆ จากบนลงล่าง ซึ่งการเขียนอัลกอริทึมแบบนี้จะใกล้เคียงกับการเขียนโค้ดโปรแกรม เพียงแต่ใช้ภาษาที่ทำความเข้าใจได้ง่ายกว่า รูปแบบของคำภาษาอังกฤษที่ใช้ในการเขียนอัลกอริทึมแบบ PDL มีดังต่อไปนี้

START-END

START

……..

END

ใช้ในการบอกจุดเริ่มต้นและจุดสิ้นสุดของโปรแกรม โดย START คือ จุดเริ่มต้นโปรแกรม และ END คือ จุดสิ้นสุดโปรแกรม

IF THEN-ELSE-ENDIF

IF...เงื่อนไข...THEN

……..……

ELSE

……..……

ENDIF

ใช้แทนการตัดสินใจเลือกเส้นทางของโปรแกรม ตามเงื่อนไขต่าง ๆ ที่อยู่หลัง IF แล้วเลือกปฏิบัติตามคำสั่งที่อยู่ใต้ IF ถ้าถูกต้องตามเงื่อนไข แต่ถ้าไม่ถูกต้องตามเงื่อนไขใด ๆ เลย ก็จะเลือกปฏิบัติตามคำสั่งที่อยู่ใต้ ELSE แทน ส่วน ENDIF ใช้บอกจุดสิ้นสุดการพิจารณาเงื่อนไข

DO-ENDDO

DO…จำนวนครั้ง…

……..……...

ENDDO

ใช้แทนการทำงานแบบวนซ้ำ (วนลูป) โดยปฏิบัติตามคำสั่งใต้ DO ตามจำนวนรอบที่อยู่หลัง DO ส่วน ENDDO ใช้บอกจุดสิ้นสุดรอบการทำงานแบบวนซ้ำ

DO UNTIL-ENDDO

DO UNTIL…เงื่อนไข…

……..…..

ENDDO

ใช้แทนการทำงานแบบวนซ้ำ โดยปฏิบัติตามคำสั่งได้ DO UNTIL จนกระทั่งเงื่อนไขที่อยู่หลัง DO UNTIL เป็นจริงจึงออกจากลูป

DO FOREVER-ENDDO

DO FOREVER

……..…..

ENDDO

ใช้แทนการทำงานแบบวนซ้ำไปเรื่อย ๆ แบบไม่รู้จบ โดยปฏิบัติตามคำสั่งได้ DO FOREVER ไปเรื่อย ๆ โดยไม่มีการออกจากลูปอีก

ยกตัวอย่าง การเขียน PDL ของเครื่องเตือนกันขโมย

START

เริ่มต้นไม่ให้เสียงดัง

DO FOREVER

IF สวิตซ์ไม่ถูกกด THEN

เสียงดัง

ELSE

เสียงไม่ดัง

ENDDO

จะเห็นได้ว่า การเขียนอัลกอริทึมทั้งสองแบบมีข้อแตกต่างกันคือ การเขียนอัลกอริทึมแบบโฟลว์ชาร์ตจะมีรูปลักษณ์ที่สามารถทำความเข้าใจได้ง่ายกว่าแบบ PDL แต่ในการเปลี่ยนจากอัลกอริทึมแบบโฟลว์ชาร์ตไปเป็นโค้ดโปรแกรมนั้น จะทำได้ยากกว่าการเปลี่ยนจากอัลกอริทึมแบบ PDL ไปเป็นโค้ดโปรแกรม เพราะอัลกอริทึมแบบ PDL มีความใกล้เคียงกับโค้ดโปรแกรมมากกว่า สำหรับในหนังสือเล่มนี้จะเลือกใช้อัลกอริทึมทั้งสองแบบควบคู่กันไปในการเรียนรู้การสร้างโครงงาน PIC เพื่อแสดงให้เห็นวิธีการเขียนอัลกอริทึมแต่ละแบบให้ชัดเจนยิ่งขึ้น และเป็นการเปรียบเทียบให้เห็นถึงข้อดีข้อเสียของอัลกอริทึมแต่ละแบบอีกด้วย

2.6 โปรแกรมภาษา C

2.6.1 โครงสร้างและรูปแบบการเขียนโปรแกรมภาษา C

ตอนนี้เราได้เรียนรู้กระบวนการขั้นตอนในการใช้งานโปรแกรม mikroC for PIC ในการเขียนโปรแกรมภาษา C ให้กับ PIC ตั้งแต่เริ่มต้นจนกระทั่งได้ไฟล์ .hex ออกมากันเป็นที่เรียบร้อยแล้ว ต่อจากนี้จะเป็นการเจาะลึกโครงสร้าง รูปแบบ และไวยากรณ์ ในการเขียนโปรแกรมภาษา C สำหรับ PIC ด้วยโปรแกรม mikroC ซึ่งลักษณะการเขียนนั้นก็จะเหมือนกับการเขียนโปรแกรมภาษา C สำหรับไมโครคอนโทรลเลอร์ทั่วไป เพียงแต่อาจมีบางรายละเอียดที่มีความแตกต่างกันอยู่บ้าง

การเขียนโปรแกรมภาษา C สำหรับ PIC ด้วยโปรแกรม mikroC มีรูปแบบการเขียนดังรูป

```
1  #define Switch PORTS. FO
2  #define Pressed 0
3
4  unsigned char J, Pattern, Seed = 1 ;
5  unsigned char DICE [ ] = { 0, 0x08, 0x22, 0x2A, 0x55, 0x5D, 0x77 } ;
6
7  unsigned char Number ( int Lim, int Y )
8  {
9            unsigned char Result ;
10           static unsigned int Y ;
11
12         Y = ( Y*32719+3 ) 32749 ;
13           Result = (( Y Lim ) +1 ) ;
14           return Result ;
15 }
16
17 void main {}
18 {
19           TRISC  = 0 ;
20           TRISB  = 1 ;
21           PORTC = 0 ;
22
23           for ( ; ; )
```

เฮดเดอร์

ประกาศ

ฟังก์ชันย่อย

ฟังก์ชัน main

```
24          {
25                      if ( Switch = =Pressed )
26                      {
27                              J = Number ( 6, seed ) ;
28                              Pattern = DICE [ J ] ;
29                              PORTC = Pattern ;
30                              Delay_ms ( 3000 ) ;
31                              PORTC = 0 ;
32                      }
33              }
34  }
35
```

เฮดเดอร์ ที่มีชื่อเรียกว่าเฮดเดอร์ เพราะส่วนนี้เปรียบเสมือนส่วนหัวของโปรแกรม ซึ่งส่วนนี้จะประกอบไปด้วยคำสั่งพรีโปรเซสเซอร์ (Preprocesssor) เป็นหลัก สังเกตคำสั่งดำเนินการแบบนี้จะมีเครื่องหมาย "#" นำหน้าเสมอ ยกตัวอย่าง เช่น #define, #include เป็นต้น ซึ่ง หน้าที่ของคำสั่งพวกนี้คือ การจัดการเกี่ยวกับไฟล์ ตัวแปร ค่าคงที่ต่าง ๆ ก่อนทำการคอมไพล์โปรแกรม ซึ่งเราจะได้ศึกษารายละเอียดของแต่ละคำสั่งภายหลัง ในบางครั้งหากเราไม่ต้องการใช้คำสั่งพรีโปรเซสเซอร์ก็ไม่จำเป็นต้องมีส่วนนี้ก็ได้

ประกาศ ส่วนถัดมาเป็นส่วนที่ใช้ในการประกาศ (Declare) ตัวแปร ค่าคงที่ ฟังก์ชันย่อยต่าง ๆ ก่อนเข้าสู่ฟังก์ชัน main ซึ่งตัวแปรที่ประกาศที่ส่วนนี้ถือว่า เป็นตัวแปรแบบโกลบอล (Global Variable) คือ สามารถเรียกใช้งานตัวแปรนี้ที่ตำแหน่งใดก็ได้ในโปรแกรม แต่มีข้อเสียคือ การประกาศตัวแปรแบบนี้จะทำให้เปลืองพื้นที่ของ RAM มากกว่าการประกาศตัวแปรภายในฟังก์ชัน

ฟังก์ชันย่อย ฟังก์ชันประเภทนี้เป็นฟังก์ชันที่ถูกสร้างขึ้นเองโดยผู้เขียนโปรแกรม (User define function) ซึ่งมีลักษณะตรงกันข้ามกับไลบรารี่ฟังก์ชัน (Library function) ซึ่งเป็นฟังก์ชันที่มีอยู่แล้วในโปรแกรมสามารถเรียกใช้งานได้เลย ฟังก์ชันย่อยจะถูกเรียกใช้งานโดยฟังก์ชัน main ซึ่งการเขียนแยกเป็นฟังก์ชันย่อยนี้ก็ด้วยเหตุผลที่ว่า ทำให้สะดวกในการพัฒนาโปรแกรม ตัวอย่างของฟังก์ชันย่อยที่เขียนใช้กันบ่อย เช่น ฟังก์ชันหน่วงเวลา (Delay function),

ฟังก์ชันให้บริการอินเตอร์รัพต์ (ISR) เป็นต้น ซึ่งเราจะได้ศึกษาวิธีการสร้างฟังก์ชันย่อยเหล่านี้กันต่อไปภายหลัง

ฟังก์ชัน main ฟังก์ชัน main เป็นฟังก์ชันหลักของโปรแกรม โดยการทำงานของโปรแกรมตั้งแต่ต้นจนจบจะอยู่ภายในฟังก์ชันนี้ตั้งแต่บรรทัดแรกจนถึงบรรทัดสุดท้าย เพราะฉะนั้นในการเขียนโปรแกรมทุกครั้งต้องมีฟังก์ชัน main เสมอ ไม่เช่นนั้น จะทำให้คอมไพล์โปรแกรมไม่ผ่าน

2.6.2 อาร์เรย์

อาร์เรย์ (Arrays) มีไว้สำหรับเก็บข้อมูลที่มีความเกี่ยวข้องกันไว้ด้วยกันภายใต้ชื่อเพียงชื่อเดียว ซึ่งจะลักษณะคล้ายกับตาราง สำหรับหลักการตั้งชื่อและการระบุชนิดของอาร์เรย์จะใช้หลักการเดียวกับตัวแปร ซึ่งการประกาศอาร์เรย์แบบนี้คือ การจองพื้นที่ว่างไว้สำหรับเก็บข้อมูล ขนาดของพื้นที่จะขึ้นอยู่กับชนิดของข้อมูลที่จะเก็บและจำนวนข้อมูลที่จะเก็บ

รูปแบบการใช้งานอาร์เรย์เพื่อจองพื้นที่ว่างสำหรับเก็บข้อมูล

ชนิดของตัวแปร ชื่ออาร์เรย์ [ตำแหน่งทั้งหมดของอาร์เรย์] ;

ตัวอย่างเช่น ต้องการประกาศอาร์เรย์ชื่อ segment เก็บข้อมูลขนาด 8 บิต จำนวนทั้งหมด 5 ตำแหน่ง

เราสามารถประกาศอาร์เรย์ พร้อมกับกำหนดค่าในแต่ละตำแหน่งให้กับสมาชิกของอาร์เรย์ที่ประกาศได้ด้วย ซึ่งการประกาศอาร์เรย์ในรูปแบบนี้ ไม่จำเป็นต้องระบุจำนวนสมาชิกของอาร์เรย์ เพราะคอมไพเลอร์จะทราบจำนวนสมาชิกโดยอัตโนมัติ จากจำนวนค่าที่กำหนดให้กับอาร์เรย์

รูปแบบการใช้งานอาร์เรย์ที่มีการกำหนดค่า

ชนิดของตัวแปร ชื่ออาร์เรย์ [] = {ค่าข้อมูลที่กำหนด};

นอกจากนี้แล้ว เรายังสามารถใช้อาร์เรย์ในการเก็บข้อความได้อีกด้วย โดยให้หนึ่งสมาชิกของอาร์เรย์เก็บข้อมูลตัวอักขระ 1 ตัว (รวมถึงที่ว่างในข้อความด้วย) ซึ่งข้อมูลตัวอักขระนี้ จะถูกเก็บข้อมูลเป็นรหัสแอสกี ซึ่งมีขนาด 8 บิต ซึ่งมีรูปแบบการประกาศอาร์เรย์เก็บข้อความ

รูปแบบการใช้งานอาร์เรย์ที่เก็บข้อความ

ชนิดของตัวแปร ชื่ออาร์เรย์ [] = {“ข้อความตัวอักษร”};

ตัวอย่างเช่น

Unsigned char text [] = {“SMART LEARNING”}

2.6.3 เครื่องหมายดำเนินการ

เครื่องหมายดำเนินการในภาษา C สำหรับโปรแกรม mikroC ในการเขียนโปรแกรมให้กับ PIC สามารถแบ่งเป็นประเภทตามลักษณะหน้าที่การใช้งานได้ ดังต่อไปนี้

เครื่องหมายดำเนินการกำหนดค่า (Assignment Operators)

เครื่องหมายนี้ก็คือ เครื่องหมายเท่ากับ (=) ซึ่งใช้ในการกำหนดค่าให้กับตัวแปรต่าง ๆ ในโปรแกรมว่า ต้องการให้มีค่าเท่ากับเท่าไร เช่น ต้องการให้ตัวแปร temp มีค่าเท่ากับ 10 หรือต้องการให้ตัวแปร total มีค่าเท่ากับตัวแปร sum บวกกับ 150 แสดงโค้ดโปรแกรมของกรณีดังกล่าวดังรูป

2.6.3.1 เครื่องหมายการคำนวณทางคณิตศาสตร์ (Arithmetic Operators)

เครื่องหมายประเภทนี้ ยกตัวอย่างเช่น เครื่องหมาย บวก ลบ คูณ หาร เป็นต้น ซึ่งใช้เป็นเครื่องหมายที่ใช้ในการคำนวณทางคณิตศาสตร์ แสดงรายละเอียดและตัวอย่างการใช้งานของเครื่องหมายการคำนวณทางคณิตศาสตร์

เครื่องหมาย	ความหมาย	ตัวอย่างการใช้งาน
+	บวก	x = 8 + 6; //ผลลัพธ์ x = 8
-	ลบ	x = 8 – 6; //ผลลัพธ์ x = 2
*	คูณ	X = 8 * 6; //ผลลัพธ์ x = 48
/	หาร	x = 8 / 6; //ผลลัพธ์ x = 1
%	หาเศษจากการหาร	x = 8 % 6; //ผลลัพธ์ x = 2
++	เพิ่มค่าขึ้นหนึ่ง	x++; //ผลลัพธ์ x = x + 1
--	ลดค่าลงหนึ่ง	x--; //ผลลัพธ์ x = x - 1

ตารางที่ 2.6.3.1 เครื่องหมายการคำนวณทางคณิตศาสตร์

2.6.3.2 เครื่องหมายการเปรียบเทียบ (Relational Operators)

เครื่องหมายประเภทนี้ ยกตัวอย่างเช่น เครื่องหมายมากกว่า เท่ากับ ไม่เท่ากับ เป็นต้น ซึ่งใช้ในการเปรียบเทียบข้อมูล 2 ข้อมูล โดยให้ผลลัพธ์เป็น 1 เมื่อถูกต้องตามเงื่อนไขการเปรียบเทียบ และให้ผลลัพธ์เป็น 0 เมื่อไม่ถูกต้องตามเงื่อนไขการเปรียบเทียบ แสดงรายละเอียดและตัวอย่างการใช้งานของเครื่องหมายการเปรียบเทียบดังตาราง

เครื่องหมาย	ความหมาย	ตัวอย่างการใช้งาน
>	มากกว่า	x = 10; x > 5 //ผลลัพธ์ = 1
<	น้อยกว่า	x = 10; x < 5 //ผลลัพธ์ = 0
= =	เท่ากับ	x = 10; x = = 10 //ผลลัพธ์ = 1
! =	ไม่เท่ากับ	x = 10; x ! = 10 //ผลลัพธ์ = 0
> =	มากกว่าหรือเท่ากับ	x = 10; x > = 10 //ผลลัพธ์ = 1
< =	น้อยกว่าหรือเท่ากับ	x = 10; x < = 9 //ผลลัพธ์ = 0

ตารางที่ 2.6.3.2 เครื่องหมายการเปรียบเทียบ

2.6.3.3 เครื่องหมายดำเนินการทางลอจิก (Logical Operators)

เครื่องหมายประเภทนี้ คือ เครื่องหมาย AND, OR และ NOT ซึ่งเป็นการดำเนินการทางลอจิกของผลลัพธ์ของเงื่อนไขการเปรียบเทียบตั้งแต่ 1 เงื่อนไขขึ้นไป โดยให้ผลลัพธ์ออกมาเป็นลอจิก 0 หรือไม่ก็ลอจิก 1 แสดงรายละเอียดและตัวอย่างการใช้งานของเครื่องหมายดำเนินการทางลอจิกดังตาราง

เครื่องหมาย	ความหมาย	ตัวอย่างการใช้งาน
& &	AND	x = 10; x > 5 && x > 20 //ผลลัพธ์ = 0
| |	OR	x = 10; x > 5 | | x > 20 // ผลลัพธ์ = 1
!	NOT	x = 10; !(x > 5) //ผลลัพธ์ = 0

ตารางที่2.6.3.3.1 เครื่องหมายดำเนินการทางลอจิก

เครื่องหมาย	ความหมาย	ตัวอย่างการใช้งาน
&	AND	x = 11001010 & 01101111; //ผลลัพธ์ x = 01001010
|	OR	x = 11001010 | 01101111; //ผลลัพธ์ x = 11101111
^	EXOR	x = 11001010 ^ 01101111; //ผลลัพธ์ x = 10100101
~	คอมพลีเมนท์ (Complement)	x = ~ 11001010; //ผลลัพธ์ x = 00110101
<<	เลื่อนบิตไปทางซ้าย	x = 00011010 << 2; //ผลลัพธ์ x = 01101000
>>	เลื่อนบิตไปทางขวา	x = 10100000 >> 3; //ผลลัพธ์ x = 00010100

ตารางที่2.6.3.3.1 เครื่องหมายดำเนินการระดับบิต

2.7 คำสั่งควบคุมทางเดินโปรแกรม

โปรแกรมแทบทุกโปรแกรม จะต้องประกอบด้วยการตัดสินใจเพื่อที่จะกระทำหรือไม่กระทำสิ่งต่าง ๆ ภายในโปรแกรม ซึ่งในการเขียนโปรแกรมภาษา C ด้วยโปรแกรม mikroC แบ่งคำสั่งควบคุมทางเดินโปรแกรมออกเป็น 3 ประเภทคือ คำสั่งเลือกทางเดินโปรแกรมแบบมีเงื่อนไข, คำสั่งวนลูปทำซ้ำ และคำสั่งเลือกทางเดินโปรแกรมแบบไม่มีเงื่อนไข, คำสั่งวนลูปทำซ้ำ และคำสั่งเลือกทางเดินโปรแกรมแบบไม่มีเงื่อนไข

2.7.1 คำสั่งเลือกทางเดินโปรแกรมแบบมีเงื่อนไข

คำสั่งในประเภทนี้ได้แก่คำสั่ง if, if-else, if-else if และคำสั่ง switch-case ซึ่งมีรายละเอียดการใช้งานแต่ละคำสั่งดังต่อไปนี้

if

if เป็นคำสั่งที่ใช้ในการตัดสินใจแบบมีเงื่อนไขแบบทางเลือกเดียว โดยโปรแกรมจะเลือกกระทำตามคำสั่งภายใต้วงเล็บปีกกาของคำสั่ง if ก็ต่อเมื่อ เงื่อนไขที่อยู่ในวงเล็บท้ายคำสั่ง if เป็นจริง ไม่เช่นนั้นจะไม่เลือกกระทำแสดงตัวอย่างการใช้คำสั่ง if ได้ดังตัวอย่าง

```
if (x>10)
{x = 0;}
if (x>10) x=0;
```

จากตัวอย่าง เป็นการเขียนคำสั่ง if ซึ่งมีอยู่ 2 วิธี วิธีแรก เป็นการให้สิ่งที่ต้องกระทำเมื่อเงื่อนไขเป็นจริงอยู่ในวงเล็บปีกกาของคำสั่ง if ส่วนวิธีที่สอง ไม่จำเป็นต้องเขียนวงเล็บปีกกา และพิมพ์โค้ดเพียงแค่บรรทัดเดียว ซึ่งจากรูปทั้ง 2 วิธีให้ความหมายที่เหมือนกันคือ ถ้าค่าของ x มากกว่า 10 โปรแกรมจะกำหนดให้ x เท่ากับ 0 แต่ถ้า x น้อยกว่าหรือเท่ากับ 10 จะไม่มีอะไรเกิดขึ้น แต่วิธีเขียนวิธีที่สองนั้นสามารถใช้ได้ในกรณีที่มีสิ่งที่ต้องกระทำอันเนื่องมาจากการตัดสินใจ จากเงื่อนไขเพียงสิ่งเดียวเท่านั้น

if-else

If-else เป็นคำสั่งที่ใช้ในการตัดสินใจแบบมีเงื่อนไขแบบ 2 ทางเลือก โดยโปรแกรมจะกระทำตามคำสั่งภายใต้วงเล็บปีกกาของ if เมื่อเงื่อนไขที่อยู่ในวงเล็บท้ายคำสั่ง if เป็นจริง และกระทำตามเงื่อนไขภายใต้วงเล็บปีกกาของ else เมื่อเงื่อนไขท้ายคำสั่ง if ไม่เป็นจริง ดังตัวอย่าง

```
if (x>10)
{y=0;}
else
{y=1;}
```

ถ้าค่าของ x มากกว่า 10 โปรแกรมจะกำหนดให้ y เท่ากับ 0 แต่ถ้า x น้อยกว่าหรือเท่ากับ 10 โปรแกรมจะกำหนดให้ y เท่ากับ 1 ซึ่งจะเห็นได้ว่ามีทางเลือกทั้งหมด 2 ทาง

if-else if

if-else if เป็นคำสั่งที่ใช้ในการตัดสินใจแบบมีเงื่อนไขแบบหลายทางเลือก โดยโปรแกรมจะกระทำตามคำสั่งภายใต้วงเล็บปีกกาของคำสั่ง if หรือ else if เมื่อเงื่อนไขที่อยู่ในวงเล็บท้ายคำสั่งเหล่านั้นเป็นจริง และทำตามคำสั่งที่อยู่ภายใต้วงเล็บปีกกาของคำสั่ง else เมื่อไม่มีเงื่อนไขใดที่อยู่ท้ายคำสั่ง if หรือ else-if เป็นจริง โดยสามารถเลือกกระทำได้เพียงแค่เงื่อนไขเดียวเท่านั้น ดังตัวอย่าง

```
if (x>10&&x<=20)
{y=0;}
else if (x>20&&x<=30)
{y=1;}
else if (x>30)
{y=2;}
else
{y=3;}
```

จากตัวอย่าง ถ้าค่าของ x อยู่ในช่วง 11-20 โปรแกรมจะกำหนดให้ y เท่ากับ 0 ถ้า x อยู่ในช่วง 21-30 โปรแกรมจะกำหนดให้ y เท่ากับ 1 ถ้า x มากกว่า 30 โปรแกรมจะกำหนดให้ y เท่ากับ 2 และสุดท้ายถ้าค่าของ x ไม่สามารถทำให้เงื่อนไขใดเป็นจริงได้ โปรแกรมจะกำหนดให้ y เท่ากับ 3 ซึ่งจะเห็นว่ามีทางเลือกทั้งหมด 4 ทางเลือก และสามารถให้มีทางเลือกมากกว่านี้ก็ได้ ขึ้นอยู่กับรูปแบบอัลกอริทึมของโปรแกรม

switch-case

switch-case เป็นคำสั่งที่ในการตัดสินใจ แบบมีเงื่อนไขแบบหลายทางเลือกเหมือนคำสั่ง if-else if แต่มีรูปแบบการเขียนที่เข้าใจง่าย และซับซ้อนน้อยกว่าคำสั่ง if-else if โดยจะทำการเปรียบเทียบเงื่อนไขทั้งหมดกับตัวแปรเพียงตัวเดียวว่า มีค่าเท่ากับเท่าไหร่และตรงกับกรณี (case) ใด จากนั้น จึงเลือกกระทำตามคำสั่งของกรณีนั้น ๆ แล้วจึงกระโดดออกจากคำสั่ง ซึ่งท้ายคำสั่งที่ให้กระทำของแต่ละกรณีจะต้องตาม ด้วยคำสั่ง break เสมอ เพื่อใช้ในการกระโดดออกจากคำสั่ง แสดงดังตัวอย่าง

```
switch (op)
{
        case'+':c=a+b;break;
        case'-':c=a-b;break;
        case'*':c=a*b;break;
        case'/':c=a/b;break;
        default:c=0;
}
```

จากตัวอย่าง ถ้า op เท่ากับ '+' โปรแกรมจะกำหนดให้ c เท่ากับ a บวกกับ b ถ้า op เท่ากับ '-' โปรแกรมจะกำหนดให้ c เท่ากับ a ลบด้วย b ถ้า op เท่ากับ '*' โปรแกรมจะกำหนดให้ c เท่ากับ a คูณกับ b ถ้า op เท่ากับ '/' โปรแกรมจะกำหนดให้ c เท่ากับ a หารด้วย b ส่วนกรณีสุดท้าย ถ้า op ไม่เท่ากับ 4 กรณีที่กล่าวมา โปรแกรมจะกำหนดให้ c เท่ากับ 0 จากนั้นจึงกระโดดออกจากคำสั่ง switch-case จะเห็นได้ว่าคำสั่ง switch-case คือ การตัดสินใจแบบมีเงื่อนไขแบบหลายทางเลือกคล้ายคำสั่ง if-else if แต่มีวิธีเขียนคำสั่งที่เข้าใจง่ายกว่า และซับซ้อนน้อยกว่า

2.7.2 คำสั่งวนลูปทำซ้ำ

คำสั่งในประเภทนี้ได้แก่คำสั่ง while, do-while และคำสั่ง for ซึ่งมีรายละเอียดการใช้งานแต่ละคำสั่ง ดังต่อไปนี้

while

while เป็นคำสั่งที่สั่งให้โปรแกรมวนลูปทำซ้ำคำสั่งที่อยู่ภายใต้วงเล็บปีกกาของคำสั่ง while ไปเรื่อย ๆ ในขณะที่เงื่อนไขที่อยู่ภายในวงเล็บท้ายคำสั่ง while เป็นจริง แต่เมื่อเงื่อนไขดังกล่าวไม่เป็นจริงแล้ว โปรแกรมก็จะทำการกระโดดออกจากลูปของ while ทันที นอกจากนั้นเรายังสามารถสั่งให้โปรแกรมทำ การวนลูปไปเรื่อย ๆ แบบไม่รู้จบ ด้วยการใส่คำจำนวนเต็มบวกในวงเล็บท้ายคำสั่ง while แทนเงื่อนไข ซึ่งคำสั่งแบบนี้จะใช้บ่อยในการเขียนโปรแกรมให้กับไมโครคอนโทรลเลอร์ แสดงดังตัวอย่าง

```
while (x<=10)
{sum=sum+x;
x++;}
```

จากตัวอย่าง โปรแกรมจะทำการบวกค่า sum กับ x แล้วเก็บผลลัพธ์ไว้ที่ sum พร้อมกับเพิ่มค่าของ x รอบละหนึ่ง ทำอย่างนี้ไปเรื่อย ๆ จนกระทั่งค่าของ x มีค่ามากกว่า 10 โปรแกรมจะกระโดดออกจากลูปของคำสั่งทันที จากตัวอย่างถ้าค่าของ sum และ x เริ่มที่ 0 โปรแกรมนี้จะเป็นการบวกค่า 1+2+3+4+...+10 และเก็บผลลัพธ์ไว้ที่ sum

do-while

do-while เป็นคำสั่งที่สั่งให้โปรแกรมวนลูปทำซ้ำคล้ายกับคำสั่ง while แต่จะต่างกันตรงที่โปรแกรมจะทำตามคำสั่งภายใต้วงเล็บปีกกาของคำสั่ง do แล้วจึงมาพิจารณาเงื่อนไขที่อยู่ในวงเล็บท้ายคำสั่ง while ด้านล่าง ถ้าเงื่อนไขเป็นจริง ก็ยังคงวนลูปทำตามคำสั่งภายใต้วงเล็บปีกกาของคำสั่ง do ต่อไป แต่ถ้าเงื่อนไขไม่เป็นจริง โปรแกรมก็จะกระโดดออกจากลูปทันที แสดงตัวอย่างการใช้คำสั่ง do-while ดังรูป

```
do
{
        sum=sum+x;
        x++;
}
while (x<=10)
```

จากตัวอย่าง ลักษณะการทำงานของโปรแกรมและผลลัพธ์ที่ได้ จะเหมือนกับการใช้คำสั่ง while เพียงแต่เริ่มต้นการทำงานของคำสั่งโปรแกรมจะนำค่า sum บวกกับ x แล้วเก็บผลลัพธ์ไว้ที่ sum และเพิ่มค่า x ขึ้นมาหนึ่งก่อน แล้วจึงทำการพิจารณาเงื่อนไขที่ว่า x มากกว่า 10 หรือไม่ ถ้าค่า x มากกว่า 10 โปรแกรมก็จะกระโดดออกจากลูปทันที ซึ่งถ้าค่าของ x เริ่มต้นที่ 0 ค่าของ x ที่นำมาพิจารณาครั้งแรกในที่นี้คือ 1 ต่างจากการใช้คำสั่ง while ที่ค่า x ที่นำมาพิจารณาครั้งแรกคือ 0

for

for เป็นคำสั่งที่สั่งให้โปรแกรมวนลูปทำซ้ำ แบบที่สามารถกำหนดจำนวนรอบได้แน่นอน และมีรูปแบบการเขียนคำสั่งที่กะทัดรัดกว่าคำสั่งแบบอื่น ๆ โดยในวงเล็บเงื่อนไขท้ายคำสั่ง for จะแบ่งเป็น 3 ส่วนคือ การกำหนดค่าเริ่มต้นให้กับตัวแปร, เงื่อนไขที่จะพิจารณา และคำสั่งที่ต้องปฏิบัติเมื่อวนลูปครบหนึ่งรอบ นอกจากนั้น เรายังสามารถใช้คำสั่ง for สั่งให้โปรแกรมวนลูปทำซ้ำแบบไม่รู้จบเหมือนกับคำสั่ง while ได้โดยใส่เครื่องหมาย ‘;’ ภายในวงเล็บท้ายคำสั่ง for แทนเงื่อนไข ดังตัวอย่าง

```
loop : x++;
       if (x<=10) goto loop;
```

2.7.3 การใช้คำสั่ง goto

จากรูป เป็นการใช้คำสั่ง goto ร่วมกับคำสั่ง if ในการสั่งให้โปรแกรมวนลูปทำซ้ำ โดยถ้าค่าของ x ยังไม่มากกว่า 10 โปรแกรมก็จะวนลูปเพิ่มค่า x ทีละหนึ่งไปเรื่อย ๆ ซึ่งเป็นการใช้คำสั่ง goto กระโดดไปยังเลเบลที่ชื่อว่า “loop” โปรแกรมจะออกจากลูปนี้ได้ก็ต่อเมื่อ ค่าของ x เท่ากับ 11 นั่นเอง ซึ่งถัดจากนี้ โปรแกรมก็จะไปทำตามคำสั่งของบรรทัดถัดไป คำสั่ง goto นี้จะช่วยอำนวยความสะดวกในการที่เราจะสามารถกระโดดไปยังตำแหน่งใดของโปรแกรมก็ได้

continue

continue เป็นคำสั่งที่สั่งให้โปรแกรมข้ามการวนลูปทำซ้ำของเงื่อนไขนั้น ๆ โดยจะต้องใช้งานร่วมกับคำสั่ง if เพื่อกำหนดเงื่อนไขที่ต้องการให้ข้ามการวนลูปทำซ้ำ ดังตัวอย่าง

```
for (x=1;x<=10;x++)
{           If (x=5) continue;
            sum=sum+x;
}
```

2.7.4การใช้คำสั่ง continue

จากตัวอย่างโปรแกรมนี้จะทำการรวมค่าตัวเลข 1+2+3+...+10 และเก็บผลลัพธ์ไว้ที่ตัวแปร sum แต่ด้วยคำสั่ง continue จะทำให้โปรแกรมข้ามการบวกตัวเลข 5 ไป ดังนั้นผลลัพธ์ sum ที่ได้จะมีค่าเท่ากับ 1+2+3+4+5+6+7+8+9+10 นั่นเอง

break

break เป็นคำสั่งที่สั่งให้โปรแกรมกระโดดออกจากคำสั่งวนลูปทำซ้ำนั้น ๆ ที่มีคำสั่ง break อยู่ภายในลูป เมื่อโปรแกรมทำงานไปเรื่อย ๆ จนพบคำสั่ง break โปรแกรมก็จะกระโดดออกจากลูปทันที ในทางปฏิบัติมักจะใช้ร่วมกับคำสั่ง if เพื่อกำหนดเงื่อนไขที่ต้องการให้กระโดดออกจากลูป ดังตัวอย่าง

```
for (x=1;x<=10;x++)
{
if (x>5) break;
sum=sum+x;
}
```

จากตัวอย่าง โปรแกรมนี้จะทำการรวมค่าตัวเลข 1+2+3+...+10 และเก็บผลลัพธ์ไว้ที่ตัวแปร sum แต่ด้วยคำสั่ง break จะทำให้โปรแกรมกระโดดออกจากลูปเมื่อ x มีค่ามากกว่า 5 ดังนั้นผลลัพธ์ sum ที่ได้จะมีค่าเท่ากับ 1+2+3+4+5 นั่นเอง

2.8 ฟังก์ชัน

ฟังก์ชัน คือ ตัวแทนของกลุ่มคำโปรแกรมที่กำหนดมาเพื่อให้กระทำตามเงื่อนไขที่สร้างไว้ เมื่อใส่ค่าอินพุตเข้าไปในฟังก์ชันก็จะได้ผลเอาต์พุตตามเงื่อนไขที่กำหนดออกมา ซึ่งในการเขียนโปรแกรมภาษา C ไวยากรณ์ของฟังก์ชันส่วนใหญ่จะมีรูปแบบดังนี้

ชื่อฟังก์ชัน(ตัวแปร1, ตัวแปร2,...)

{.....

ส่วนของคำสั่งหรือฟังก์ชันที่กำหนด (Function body)

..... }

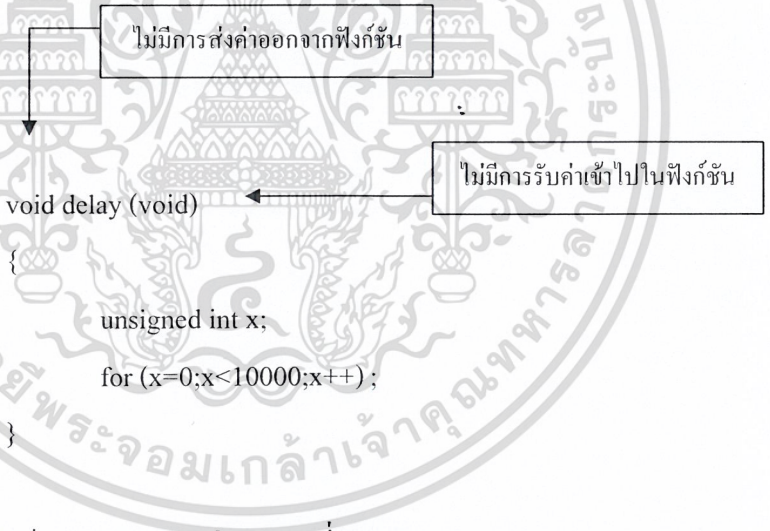
ในการเขียนโปรแกรมภาษา C ฟังก์ชันแบ่งออกได้เป็น 2 ประเภทคือ ฟังก์ชันที่ต้องเขียนขึ้นเอง (User defined function) และไลบรารี่ฟังก์ชัน (Library function) ซึ่งในหัวข้อนี้จะอธิบายลักษณะรูปแบบ และวิธีการสร้างฟังก์ชันที่ต้องเขียนขึ้นเอง ส่วนไลบรารี่ฟังก์ชันจะอธิบายในหัวข้อถัดไป

2.8.1 ชนิดของฟังก์ชัน

ฟังก์ชันที่ต้องเขียนขึ้นเอง (User defined function) แบ่งออกเป็น 2 ชนิด คือ ฟังก์ชันที่ไม่มีการส่งผ่านค่า และฟังก์ชันที่มีการส่งผ่านค่า ซึ่งฟังก์ชันแต่ละชนิดมีลักษณะการใช้งานและรูปแบบดังต่อไปนี้

ฟังก์ชันที่ไม่มีการส่งผ่านค่า

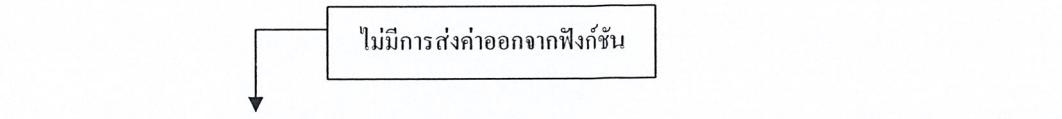
ฟังก์ชันชนิดนี้จะไม่มีการรับค่าของตัวแปรอื่น ๆ จากภายนอกฟังก์ชัน และไม่มีการส่งค่าของตัวแปรออกไปภายนอกฟังก์ชัน ฟังก์ชันจะทำงานเองโดยอิสระ เมื่อทำการเรียกใช้งานฟังก์ชันจากฟังก์ชัน main ฟังก์ชันชนิดนี้จะไม่สามารถยืดหยุ่นได้ จึงทำให้ในการใช้งานไม่สะดวกนักและปัจจุบันไม่เป็นที่นิยมใช้นัก ที่พบเห็นบ่อย ๆ เช่น ฟังก์ชันหน่วงเวลา (delay) แบบค่าเวลาคงที่ไม่สามารถเปลี่ยนแปลงได้ ฟังก์ชันชนิดนี้จะใช้คำว่า "void" ในการบอกว่าไม่มีการรับและส่งค่าใด ๆ แสดงตัวอย่างฟังก์ชันที่ไม่มีการส่งผ่านค่า ดังรูป



ฟังก์ชัน delay แบบค่าเวลาคงที่ 10000

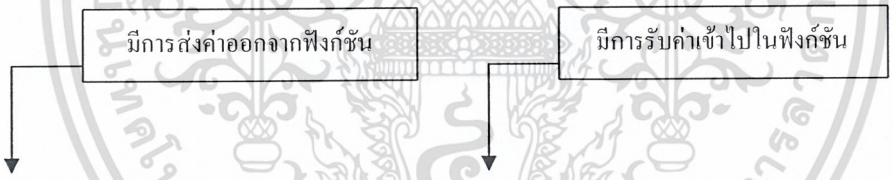
ฟังก์ชันที่มีการส่งผ่านค่า

ฟังก์ชันชนิดนี้ จะมีทั้งฟังก์ชันที่มีการรับค่าเข้ามาในฟังก์ชันอย่างเดียวไม่มีการส่งออก เช่น ฟังก์ชัน delay แบบที่สามารถกำหนดค่าเวลา delay ได้ และฟังก์ชันที่มีทั้งการรับค่าเข้ามาในฟังก์ชันและส่งค่าออกจากฟังก์ชัน เช่น ฟังก์ชันแปลงหน่วย ฟังก์ชันคำนวณต่าง ๆ ซึ่งฟังก์ชันแบบนี้จะต้องทำการส่งค่าออกกลับไปโดยใช้คำสั่ง return ในการเรียกใช้งานฟังก์ชันเหล่านี้จากฟังก์ชัน main จะต้องระบุค่าที่จะส่งให้กับฟังก์ชันด้วย แสดงตัวอย่างฟังก์ชันที่มีการส่งผ่านค่า ดัง 2 ตัวอย่างนี้




```
void delay (unsigned int z)
{
        unsigned int x;
        for (x=0;x<z;x++) ;
}
```

ฟังก์ชัน delay แบบที่สามารถกำหนดค่าเวลา delay ได้

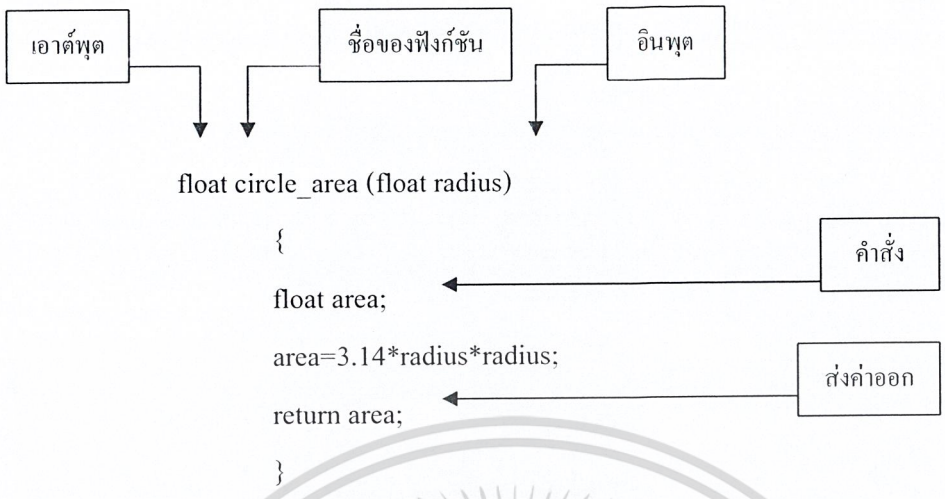



```
Int box_volume (int width, int length, int height)
{
        int volume;
        volume=width*length*height;
        return volume;
}
```

ฟังก์ชันหาปริมาตรของกล่องสี่เหลี่ยม

2.8.2 การสร้างและการใช้งานฟังก์ชัน

ในการสร้างฟังก์ชันทุกฟังก์ชัน จะต้องประกอบไปด้วยส่วนประกอบต่าง ๆ ของฟังก์ชัน แสดงดังตัวอย่าง



ฟังก์ชันหาพื้นที่ของวงกลม

ชื่อของฟังก์ชัน ชื่อของฟังก์ชันมีหลักการตั้งชื่อเดียวกับชื่อตัวแปร ซึ่งชื่อนี้จะใช้ในการอ้างอิงเพื่อเรียกใช้งานฟังก์ชันจากฟังก์ชัน main

อินพุต อินพุตของฟังก์ชันคือ การประกาศตัวแปรทางอ้อม เพื่อใช้ตัวแปรนี้ในการรับค่าอินพุตที่ส่งมาจากฟังก์ชัน main โดยจะต้องประกาศชนิดของตัวแปรอินพุตให้ตรงกับชนิดของข้อมูลที่จะรับ ตามด้วยชื่อตัวแปรอินพุต และตัวแปรอินพุตนี้ถือว่า เป็นตัวแปรภายใน (Local Variable) ไม่สามารถเรียกใช้ตัวแปรจากภายนอกฟังก์ชันได้ และถ้าหากมีอินพุตหลาย ๆ อินพุตก็จะต้องประกาศตัวแปร ณ ตำแหน่งนี้ให้มีจำนวนเท่ากับจำนวนอินพุต ส่วนถ้าต้องการให้ฟังก์ชันไม่มีการรับค่าอินพุต ข้อความที่ตำแหน่งนี้จะเป็นคำว่า “void”

เอาต์พุต เอาต์พุตของฟังก์ชันคือ การระบุชนิดของข้อมูลที่จะส่งออกจากฟังก์ชัน ซึ่งก็คือชนิดของตัวแปรที่ใช้เก็บข้อมูลที่จะส่งออกจากฟังก์ชันนั่นเอง จากรูปจะเห็นได้ว่า เอาต์พุตของฟังก์ชันคือ “float” ซึ่ง float คือ ชนิดของตัวแปรarea ที่จะส่งออกจากฟังก์ชันนั่นเอง ส่วนถ้าต้องการให้ฟังก์ชันไม่มีการส่งค่าเอาต์พุตออกจากฟังก์ชัน ข้อความที่ตำแหน่งจะเป็นคำว่า “void”

คำสั่ง เป็นคำสั่งทั่ว ๆ ไปที่อยู่ภายในฟังก์ชัน เมื่อเรียกใช้ฟังก์ชันย่อยเหล่านี้จากฟังก์ชัน main โปรแกรมก็จะปฏิบัติตามคำสั่งเหล่านี้ไปจนถึงบรรทัดสุดท้าย แล้วจึงกลับไปทำงานต่อยังฟังก์ชัน main บรรทัดที่ต่อจากการเรียกใช้งานฟังก์ชัน

ส่งค่าออก เมื่อต้องการส่งค่าใด ๆ ออกจากฟังก์ชันย่อยกลับไปยังฟังก์ชัน main จะต้องใช้คำสั่ง return ตามด้วยชื่อของตัวแปรที่เก็บข้อมูลที่ต้องการจะส่งออกจากฟังก์ชัน แล้วตามด้วยเครื่องหมายสิ้นสุดบรรทัด โดยจะสามารถส่งค่าเอาต์พุตออกจากฟังก์ชันได้เพียงค่าเดียวเท่านั้น แม้ว่าจะมีอินพุตหลายอินพุตก็ตาม

ฟังก์ชันที่เขียนขึ้นทุกฟังก์ชันถือเป็นฟังก์ชันย่อยทั้งหมด ซึ่งต้องถูกเรียกใช้โดยคำสั่งจากฟังก์ชัน main ยกเว้นฟังก์ชันให้บริการอินเตอร์รัพต์ (Interrupt Service Routine) หรือ ISR ที่จะถูกเรียกใช้งานโดยอัตโนมัติจากการอินเตอร์รัพต์ที่ CPU วิธีในการเรียกใช้งานฟังก์ชันย่อยก็เพียงแต่เขียนชื่อของฟังก์ชันที่ต้องการใช้งานตามด้วยค่าที่จะส่งให้กับฟังก์ชันนั้น ๆ ในวงเล็บและตามด้วยเครื่องหมายสิ้นสุดบรรทัดสำหรับฟังก์ชันที่มีการส่งผ่านค่า ส่วนถ้าเป็นฟังก์ชันแบบที่ไม่มีการส่งผ่านค่า ก็ให้เขียนวงเล็บปล่อยว่างไว้ แสดงตัวอย่างการเรียกใช้งานฟังก์ชันทั้งสองแบบ ดังตัวอย่าง

```
void main ()
{
    Delay () ;
}
```

(ก)

```
void main ()
{
}
```

(ข)

การเรียกใช้งานฟังก์ชัน delay แบบค่าเวลาคงที่ และแบบกำหนดค่าเวลา delay ได้

สำหรับฟังก์ชันที่มีการส่งค่าออกจากฟังก์ชันกลับมายังฟังก์ชัน main ก็สามารถเรียกใช้งานได้โดยประกาศตัวแปร เพื่อมารับค่าเอาต์พุตที่ออกจากฟังก์ชัน และเขียนโค้ดโปรแกรมให้ตัวแปรที่ประกาศขึ้นนั้นมีค่าเท่ากับชื่อของฟังก์ชัน ตามด้วยค่าอินพุตที่จะส่งเข้าไปให้กับฟังก์ชันตามลำดับของการประกาศตัวแปรอินพุต แสดงตัวอย่างการเรียกใช้งานฟังก์ชันแบบที่มีการส่งผ่านค่าเข้าและออกจากฟังก์ชัน ดังตัวอย่าง

```
void main ()
{
    int v;
    V=box_volume (3,4,5)
}
```

การเรียกใช้ฟังก์ชันหาปริมาตรของกล่องสี่เหลี่ยม

ในการเขียนโปรแกรมโดยปกติแล้ว ฟังก์ชันย่อยที่เขียนขึ้นเองเหล่านี้ จะต้องอยู่ก่อนหน้าฟังก์ชัน main หรือพูดง่าย ๆ ก็คือ ถูกคอมไพล์ก่อนฟังก์ชัน main ฟังก์ชัน main จึงจะสามารถเรียกใช้งานฟังก์ชันย่อยเหล่านี้ได้ แต่ถ้ามีฟังก์ชันย่อยจำนวนมากแล้ว เราก็จะต้องเขียนฟังก์ชันย่อยเหล่านี้ก่อนหน้าฟังก์ชัน main ซึ่งจะทำให้ไม่สะดวกต่อการแก้ไขหรือปรับปรุงโปรแกรม แต่อย่างไรก็ตาม เราสามารถที่จะเขียนฟังก์ชันย่อยเหล่านี้หลังจากฟังก์ชัน main ได้ เพียงแต่จะต้องประกาศชื่อฟังก์ชันไว้ก่อนหน้าฟังก์ชัน main คล้ายกับการประกาศตัวแปรภายนอก (Global Variable) แสดงตัวอย่างการประกาศฟังก์ชัน ดังตัวอย่าง

```
1   void delay (unsigned int z) ;
2
3   void main ( )
4   {
5           TRISB=O;
6           for (; ;)
7           {
8                   PORTB.FO=1;
9                   delay (50000) ;
10                  PORTB. FO=0;
11                  delay (30000) ;
12          }
13  }
14
15  void delay (unsigned int z)
16  {
17          unsigned int x;
18          for (x=0;x<z;x++) ;
19  }
20
```

ประกาศฟังก์ชันย่อย

ฟังก์ชัน main

เรียกใช้งานฟังก์ชันย่อย

ตัวฟังก์ชันย่อย

การเขียนฟังก์ชันย่อยหลังจากฟังก์ชัน main

การเขียนโปรแกรมที่ดี ควรมีการเขียนโค้ดโปรแกรมแยกเป็นฟังก์ชันย่อย ๆ เพราะฟังก์ชันจะเป็นตัวช่วยจัดระเบียบให้กับโค้ดโปรแกรม ทำให้เราสามารถแก้ไขโค้ดโปรแกรมได้สะดวก รวมทั้งยังทำให้ผู้ที่อ่านหรือนำโค้ดโปรแกรมของเราไปพัฒนาต่อ ทำความเข้าใจโค้ดโปรแกรมได้ง่ายยิ่งขึ้นอีกด้วย

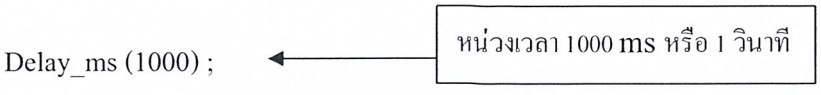
2.8.3 บิวอินฟังก์ชัน (Built-in function)

ในคอมไพเลอร์ของ mikro C สำหรับ PIC จะมีกลุ่มของบิวอินฟังก์ชัน (Built-in function) ที่สามารถเรียกได้จากโปรแกรมได้โดยตรง ซึ่งแสดงดังตาราง

ฟังก์ชัน	รายละเอียดของฟังก์ชัน
Lo	แสดงค่า Byte ล่างสุด (ตั้งแต่บิต 0 ถึงบิต 7)
Hi	แสดงค่า Byte ถัดขึ้นจาก Byte ล่างสุด (ตั้งแต่บิต 8 ถึงบิต 15)
Higher	แสดงค่า Byte ถัดลงมาจาก Byte บนสุด (ตั้งแต่บิต 16 ถึงบิต 23)
Highest	แสดงค่า Byte บนสุด (ตั้งแต่บิต 24 ถึงบิต 31)
Delay_us	หน่วงเวลาในหน่วยไมโครวินาที (µs)ตามค่าคงที่อินพุตของฟังก์ชัน
Delay_ms	หน่วงเวลาในหน่วยมิลลิวินาที (ms) ตามค่าคงที่อินพุตของฟังก์ชัน
Vdelay_ms	หน่วงเวลาในหน่วยมิลลิวินาที (ms) ตามค่าตัวแปรอินพุตของฟังก์ชัน
Delay_Cyc	หน่วงเวลาโดยใช้เวลาอ้างอิงจากสัญญาณนาฬิกาที่จ่ายให้ PIC
Clock_Khz	แสดงค่าความถี่สัญญาณนาฬิกาที่จ่ายให้ PIC ในหน่วย KHz
Clock_MHz	แสดงค่าความถี่สัญญาณนาฬิกาที่จ่ายให้ PIC ในหน่วย MHz

ตารางที่ 2.8.3 บิวอินฟังก์ชัน

ในการเรียกใช้งาน Built-in function ตามตารางสามารถเรียกใช้งานได้เลยโดยไม่จำเป็นต้องรวมไฟล์เฮดเดอร์ (.h) ใด ๆ ยกเว้นฟังก์ชัน Lo, Hi, Higher, Highest ที่ต้องทำการรวมไฟล์เฮดเดอร์ built_in.h ด้วยคำสั่ง #include<built_in.h> ที่ส่วนหัวของโปรแกรมก่อน จึงจะสามารถใช้งานได้ แสดงตัวอย่างการเรียกใช้งานฟังก์ชัน Delay_ms ดังตัวอย่าง



การเรียกใช้งานฟังก์ชัน Delay_ms

2.9 ไลบรารี่ฟังก์ชันในโปรแกรม mikroC สำหรับ PIC

ไลบรารี่ฟังก์ชัน (Library function) เป็นฟังก์ชันที่มีมาให้พร้อมกับคอมไพเลอร์ไม่จำเป็นต้องเขียนขึ้นเอง เพียงแต่ทำการเรียกใช้เท่านั้น ก็สามารถใช้งานฟังก์ชันเหล่านี้ได้เลย ทั้งนี้เพื่อช่วยอำนวยความสะดวกในการเขียนโปรแกรม ทำให้เราเขียนโปรแกรมได้ง่ายขึ้น และลดปริมาณข้อความที่เป็นโค้ดโปรแกรมลงได้อีกด้วย

ในโปรแกรม mikroC for PIC มีไลบรารี่ฟังก์ชันให้เลือกใช้มากมาย แต่จะขอยกตัวอย่างเฉพาะไลบรารี่ฟังก์ชันที่สำคัญและใช้งานบ่อย ๆ ในการเขียนโปรแกรมให้กับ PIC ซึ่งมีดังต่อไปนี้

2.9.1 LCD library functions

การเขียนโปรแกรมเชื่อมต่อระหว่าง PIC กับ LCD Module จะมีวิธีติดต่ออยู่ทั้งหมด 2 วิธีคือ การเชื่อมต่อ LCD แบบ 8 บิต และการเชื่อมต่อ LCD แบบ 4 บิต ซึ่งใน โปรแกรม mikroC นั้น มีทั้งไลบรารี่ฟังก์ชันของการเชื่อมต่อ LCD ทั้งแบบ 8 บิต และแบบ 4 บิต ซึ่งในที่นี้จะขอนำเสนอไลบรารี่ฟังก์ชันของการเชื่อมต่อ LCD แบบ 4 บิต ประกอบไปด้วยฟังก์ชันต่าง ๆ ดังต่อไปนี้

_Lcd_Config

ใช้ในการปรับแต่งหรือเลือกว่า จะต้องการใช้พอร์ตใดบิตใดบ้างของ PIC ในการเชื่อมต่อกับ LCD Module โดยมีรูปแบบเรียกใช้ฟังก์ชันคือ

_Lcd_Config (ชื่อพอร์ต, ขา RS, ขา EN, ขา R/W, ขา D7, ขา D6, ขา D5, ขา D4);

โดยให้หมายเลขบิตของพอร์ตที่จะเชื่อมต่อกับขานั้น ๆ อยู่ตรงตำแหน่งเดียวกับรูปแบบข้างต้น ยกตัวอย่างเช่น ถ้าต้องการให้ขา RS เชื่อมต่อกับ RBO, ขา EN เชื่อมต่อกับ RB1, ขา EN เชื่อมต่อกับ RB2, ขา R/W เชื่อมต่อกับ RB3, ขา D7 เชื่อมต่อกับ RB3, ขา D6 เชื่อมต่อกับ RB4, ขา D5 เชื่อมต่อกับ RB5 และขา D4 เชื่อมต่อกับ RB6 สามารถใช้ฟังก์ชัน Lcd_Config เชื่อมต่อ LCD ผ่านพอร์ต B

LCD_Config (&PORTB, 0, 1, 2, 3, 4, 5, 6) ;

_Lcd_Init

ใช้ในการเลือกพอร์ตของ PIC ที่จะใช้ในการเชื่อมต่อกับ LCD Module โดยใช้การเชื่อมต่อแบบที่ไม่มีการปรับแต่งบิตที่จะเชื่อมต่อกับ LCD

การเรียกใช้งานฟังก์ชัน Lcd_Init ก็เพียงแต่ใส่ชื่อพอร์ตที่ต้องการจะเชื่อมต่อกับ LCD ในวงเล็บต่อจากคำสั่ง Lcd_Init เท่านั้นเอง และถ้าหากใช้ฟังก์ชันนี้แล้วก็ไม่จำเป็นต้องเรียกใช้ฟังก์ชัน Lcd_Config เพื่อปรับแต่งบิตที่จะเชื่อมต่อกับ LCD อีก สามารถเขียนโปรแกรมต่อไปได้เลย แสดงตัวอย่างการเรียกใช้ฟังก์ชัน Lcd_Init เชื่อมต่อ LCD ผ่านพอร์ต B

```
Lcd_Init (&PORTB) ;
```

_Lcd_Out

เป็นฟังก์ชันที่ใช้ในการแสดงข้อความบนหน้าจอ LCD โดยมีรูปแบบการเรียกใช้ฟังก์ชันคือ

_Lcd_Out (ตำแหน่งแถว, ตำแหน่งคอลัมน์, “ข้อความที่จะแสดง”) ;

โดยตำแหน่งของแถวและตำแหน่งของคอลัมน์ คือ พิกัดบนจอ LCD ที่จะให้เป็นจุดเริ่มต้นแสดงข้อความโดยเริ่มพิกัด (1, 1) ที่มุมบนซ้ายสุดของจอ LCD แสดงตัวอย่างการเรียกใช้งานฟังก์ชัน Lcd_Out แสดงข้อความ “Hello!” ที่บรรทัดที่ 1 คอลัมน์ที่ 3 ของ LCD

```
Lcd_Out (1, 3, “Hello!”) ;
```

_Lcd_Out_Cp

ใช้ในการแสดงข้อความบนหน้าจอ LCD โดยเริ่มต้น ณ ตำแหน่งที่เคอร์เซอร์อยู่ปัจจุบัน แสดงตัวอย่างการใช้งานฟังก์ชัน Lcd_Out_Cp แสดงข้อความ “Here!”

```
Lcd_Out_Cp (“Here!”) ;
```

_Lcd_Chr

ใช้ในการแสดงอักขระเดี่ยวบนหน้าจอ LCD ณ ตำแหน่งพิกัดแถวและคอลัมน์ที่ระบุ โดยมีรูปแบบการใช้ฟังก์ชันเหมือนกับฟังก์ชัน Lcd_Out เพียงแต่ใช้ในการแสดงอักขระเพียงตัวเดียวแทน แสดงตัวอย่างการใช้ฟังก์ชัน Lcd_Chr แสดงตัวอักษร “i” บนหน้าจอ LCD ที่บรรทัดที่ 2 คอลัมน์ที่ 3

```
Lcd_Chr (2, 3, ‘I’) ;
```

Lcd_Chr_Cp

ใช้ในการแสดงอักขระเดี่ยวบนหน้าจอ LCD ณ ตำแหน่งที่เคอร์เซอร์อยู่ปัจจุบัน เหมือนกับฟังก์ชัน Lcd_Out_Cp เพียงแต่ใช้ในการแสดงอักขระเพียงตัวเดียวแทน แสดงตัวอย่างการใช้ฟังก์ชัน Lcd_Chr_Cp แสดงตัวอักษร “e”

Lcd_Chr_Cp (‘e’) ;

Lcd_Cmd

ใช้ในการควบคุมการแสดงผลของ LCD โดยประกอบไปด้วยคำสั่งต่าง ๆ แสดงดังตาราง

คำสั่ง	รายละเอียดของคำสั่ง
LCD_CLEAR	เคลียร์หน้าจอ LCD
LCD_RETURN_HOME	ย้ายเคอร์เซอร์ไปอยู่ที่พิกัดแรก (1, 1)
LCD_FIRST_ROW	ย้ายเคอร์เซอร์ไปอยู่ที่บรรทัดที่หนึ่ง
LCD_SECOND_ROW	ย้ายเคอร์เซอร์ไปอยู่ที่บรรทัดที่สอง
LCD_THIRD_ROW	ย้ายเคอร์เซอร์ไปอยู่ที่บรรทัดที่สาม
LCD_FOURTH_ROW	ย้ายเคอร์เซอร์ไปอยู่ที่บรรทัดที่สี่
LCD_BLINK_CURSOR_ON	สั่งให้เคอร์เซอร์กระพริบ
LCD_UNDERLINE_ON	แสดงเคอร์เซอร์แบบขีดเส้นใต้
LCD_CURSOR_OFF	ปิดการแสดงเคอร์เซอร์
LCD_TURN_ON	เปิดการแสดงผล LCD
LCD_TURN_OFF	ปิดการแสดงผล LCD
LCD_MOVE_CURSOR_LEFT	เลื่อนเคอร์เซอร์ไปทางซ้าย
LCD_MOVE_CURSOR_RIGHT	เลื่อนเคอร์เซอร์ไปทางขวา
LCD_SHIFT_LEFT	เลื่อนข้อความไปทางซ้าย
LCD_SHIFT_RIGHT	เลื่อนข้อความไปทางขวา

ตารางที่ 2.9.1LCD library functions

ในการเรียกใช้ฟังก์ชัน Lcd_Cmd สามารถเรียกใช้ได้โดยใส่คำสั่งควบคุมที่ต้องการลงในวงเล็บต่อท้ายจาก Lcd_Cmd แล้วตามด้วยเครื่องหมายสิ้นสุดบรรทัด (;) แสดงตัวอย่างการใช้ฟังก์ชัน Lcd_Cmd ในการเคลียร์หน้าจอ LCD

2.9.2 Keypad library functions

Keypad library functions จะช่วยให้เราไม่ต้องเขียนโปรแกรมสแกน Keypad ในการเขียนโปรแกรมเชื่อมต่อ PIC เพื่อรับคำอินพุตจาก Keypad โดยปกติแล้วไลบรารี่ฟังก์ชันนี้จะถูกใช้ร่วมกับ Keypad ขนาด 4 x 4 แต่อย่างไรก็ตาม เราสามารถนำมาประยุกต์ใช้กับ Keypad ขนาด 4 x 3, 4 x 2 หรือ 4 x 1 ได้เช่นเดียวกัน ซึ่งประกอบไปด้วยฟังก์ชันต่าง ๆ ดังต่อไปนี้

Keypad_Init

ใช้ในการเลือกพอร์ตของ PIC ที่จะใช้เชื่อมต่อจาก Keypad โดยจะต้องใช้ทั้งหมด 8 บิต ไม่ว่าจะเป็น Keypad ขนาดใดก็ตาม เพราะไลบรารี่ฟังก์ชันนี้จะอ้างอิงกับ Keypad ขนาด 4 x 4 เสมอ

สำหรับการใช้ Keypad ขนาดอื่น ๆ ที่ไม่ใช่ Keypad ขนาด 4 x 4 คอลัมน์หรือแถวใดไม่มี ก็ไม่จำเป็นต้องเชื่อมต่อให้ปล่อยว่างไว้ การเรียกใช้งานฟังก์ชันนี้ก็เพียงแต่ใส่ชื่อของพอร์ตที่ต้องการเชื่อมต่อในวงเล็บท้ายชื่อของฟังก์ชัน แล้วตามด้วยเครื่องหมายสิ้นสุดบรรทัด แสดงตัวอย่างการใช้ฟังก์ชัน Keypad_Init ในการเชื่อมต่อ Keypad ด้วยพอร์ต B

```
Keypad_Init (&PORTB) ;
```

Keypad_Read

ใช้แทนการสแกน Keypad ในการอ่านข้อมูลการกด Keypad โดยส่งค่าการกด Keypad ออกมาจากฟังก์ชันเป็นค่าตำแหน่งตัวเลขที่อ้างอิงจากคีย์แพด 4 x 4 ซึ่งจะเป็นค่าตัวเลข 1 ถึง 16 เพื่อแสดงข้อมูลว่า ปุ่มใดถูกกดหรือไม่ และจะส่งค่าตัวเลข 0 ออกมาจากฟังก์ชัน เมื่อไม่มีปุ่มใดถูกกด ซึ่งตำแหน่งของปุ่ม Keypad และ ค่าตัวเลขมีความสัมพันธ์กัน

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

ตารางที่ 2.9.2ความสัมพันธ์ระหว่างตำแหน่งปุ่มกดของ Keypad กับค่าตัวเลข

ในการเรียกใช้งานฟังก์ชัน Keypad_Read ก็จะต้องสร้างตัวแปรขึ้นมา เพื่อเก็บค่าที่ส่งออกมาจากฟังก์ชัน จากนั้นจึงนำค่าที่ได้ไปสร้างเงื่อนไขให้กับโปรแกรมต่อไป แสดงตัวอย่างการใช้ฟังก์ชัน Keypad_Read ในการสแกน Keypad เก็บค่าไว้ที่ตัวแปร kp

```
Char kp ;
Kp = Keypad_Read ( ) ;
```

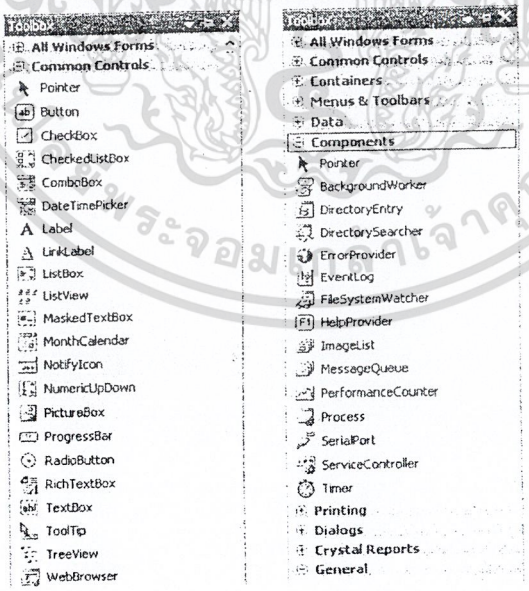
Keypad_Released

ใช้แทนการสแกน Keypad เหมือนกับฟังก์ชัน Keypad_Read แต่จะแตกต่างตรงที่ฟังก์ชันนี้หลังจากกดปุ่มแล้วจะรอจนกว่าปล่อยปุ่มจึงจะส่งค่าการกดเป็นค่าตัวเลข 1 ถึง 16 ออกมาจากฟังก์ชัน ฟังก์ชันนี้จะเป็นที่นิยมใช้กว่าฟังก์ชัน Keypad_Read เพราะสามารถป้องกันการกดปุ่มค้างได้ด้วย ความสัมพันธ์ระหว่างตำแหน่งปุ่มกดของ Keypad กับค่าตัวเลข และวิธีการเรียกใช้ฟังก์ชัน Keypad_Released มีลักษณะเดียวกันกับฟังก์ชัน Keypad_Read แสดงตัวอย่างการใช้ฟังก์ชัน Keypad_Released ในการสแกน Keypad เก็บค่าไว้ที่ตัวแปร kp

2.10 คอนโทรลที่ใช้ติดต่อและควบคุม SerialPort ใน Visual Basic 2005

สำหรับ VB 2005 นั้น ทางผู้พัฒนาโปรแกรมได้เตรียมออบเจ็กต์ไว้ให้เลือกใช้งานหลายตัว ออบเจ็กต์แต่ละตัวได้ถูกจัดกลุ่มและจัดเก็บไว้ใน Toolbox ดังรูปด้านล่างนี้

ส่วนออบเจ็กต์ที่ใช้สำหรับการติดต่อกับพอร์ตอนุกรมนั้นจะอยู่ในกลุ่มของ Components ที่มีชื่อว่า SerialPort (ออบเจ็กต์ในกลุ่มนี้จะมองไม่เห็นเวลาทำงานบนแอพพลิเคชัน)



รูปที่2.10.1คอนโทรลที่ใช้ติดต่อและควบคุม SerialPort ใน Visual Basic 2005

พร็อพเพอร์ตี้ที่สำคัญในการใช้งาน SerialPort

PortName ใช้ในการกำหนดหมายเลขของพอร์ตอนุกรมที่เราต้องการจะติดต่อ

BaudRate ใช้ในการกำหนดอัตราบอด (Baud Rate) หรือความเร็วในการส่งข้อมูล มีหน่วยเป็นบิตต่อวินาที

DataBits ใช้ในการกำหนดจำนวนของบิตข้อมูล

StopBits ใช้ในการกำหนดจำนวนของบิตปิดท้าย

Parity ใช้ในการกำหนดพาริตี้ของข้อมูล

Open ใช้สำหรับเปิดการใช้งานพอร์ตอนุกรม

Close ใช้สำหรับปิดการใช้งานพอร์ตอนุกรม

Write ใช้สำหรับส่งข้อมูลออกจากพอร์ตอนุกรม โดยมีตัวเลือก

SerialPort.Write ส่งข้อมูลออกทางพอร์ตอนุกรม

SerialPort.WriteLine ส่งข้อมูลที่ถูกระบุโดย String และค่าของ NewLine ออกทางพอร์ตอนุกรม

Read ใช้สำหรับรับข้อมูลที่เข้ามาทางพอร์ตอนุกรม แบ่งออกเป็นวิธีการย่อย ๆ ได้ดังนี้

SerialPort.ReadByte รับข้อมูล 1 Byte แบบ Synchronously จากพอร์ตอนุกรม

SerialPort.ReadChar รับข้อมูล 1 Character แบบ Synchronously จากพอร์ตอนุกรม

SerialPort.ReadLine อ่านค่าของ NewLine ที่อยู่ใน Input Buffer

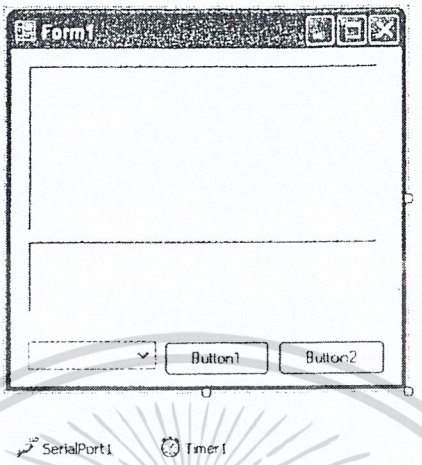
SerialPort.ReadTo อ่านค่า String ที่ถูกระบุไว้ในค่าของ Input Buffer

การเขียนโปรแกรมเพื่อรับและส่งข้อมูลผ่านพอร์ตอนุกรมด้วย Visual Basic 2005

1. เปิด Project ใหม่ โดยคลิกเมนู File > New Project ให้เลือก Windows Application เพื่อเปิดฟอร์มเปล่าขึ้นมา
2. นำคอนโทรลต่าง ๆ มาวางลงบนฟอร์ม ดังนี้

คอนโทรล Rich TextBox	2	คอนโทรล
คอนโทรล Button	2	คอนโทรล
คอนโทรล ComboBox	1	คอนโทรล
คอนโทรล Timer	1	คอนโทรล
คอนโทรล SerialPort	1	คอนโทรล

และจัดให้เป็นระเบียบ ดังรูป



รูปที่2.10.2โปรแกรมเพื่อรับและส่งข้อมูลผ่านพอร์ตอนุกรมด้วย Visual Basic 2005

3. เขียนโค้ดคำสั่งเพื่อควบคุมการทำงานของโปรแกรม ดังนี้

```
Public Class Form 1
    Private Sub From 1_Load(ByVal sender As System.Object. ByVal e As
        System.EventArgs) Handles MyBase.Load
    ComboBox1.Items.Add("COM1")
    ComboBox1.Items.Add("COM2")
    ComboBox1.Items.Add("COM3")
    ComboBox1.Items.Add("COM4")
    ComboBox1.Items.Add("COM5")
    RichTextBox2.Text = "Input text here"
    Button1.Text = "Connect"
    Button2.Text = "Send"
    Timer1.Stop()
End Sub
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles Button1.Click
        If Button.Text = "Connect" Then
                With SerialPort1
```

```
            .PortName   =   ComboBox1.Text
            .BaudRate       =       9600
            .DataBits       =       8
            .StopBits       =       10.Pots.StopBits.One
            .Parity         =       10.Ports.Parity.None
        End With
        SerialPort1.Open()
        Button1.Text = "Disconnect"
        Timer1.Interval = 500
        Timer1.Start()
    Else
        SerialPort1.Close()
        Timer1.Stop()
        Button.Text = "Connect"
    End If
End Sub
Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles Timer1.Tick Rich TextBox1.AppendText
    (SerialPort1.ReadExisting)
End Sub
Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles Button2.Click SerialPort1.Write(Rich TextBox2.Text &
    vbCrLf)
    RichTextBox2.Text = ""
End Sub
End Class
```

2.11 ระบบเครือข่ายไร้สาย

ระบบเครือข่ายไร้สาย เป็นเทคโนโลยีที่ช่วยให้การติดต่อสื่อสาร ระหว่างอุปกรณ์สื่อสาร หรือเครื่องคอมพิวเตอร์ 2 เครื่อง หรือกลุ่มของเครื่องคอมพิวเตอร์ที่สามารถสื่อสารกันได้ รวมถึงการติดต่อสื่อสารระหว่างเครื่องคอมพิวเตอร์กับอุปกรณ์เครือข่ายคอมพิวเตอร์ด้วยเช่นกัน โดยปราศจาก

การใช้สายสัญญาณในการเชื่อมต่อ แต่จะใช้คลื่นวิทยุเป็นช่องทางการสื่อสารแทน การรับ – ส่งข้อมูลระหว่างกันจะผ่านอากาศ ทำให้ไม่ต้องเดินสายสัญญาณ และติดตั้งใช้งานได้สะดวกขึ้น ระบบเครือข่ายไร้สายใช้แม่เหล็กไฟฟ้าผ่านอากาศ เพื่อรับ – ส่งข้อมูลข่าวสารระหว่างเครื่องคอมพิวเตอร์ และระหว่างเครื่องคอมพิวเตอร์กับอุปกรณ์เครือข่าย โดยคลื่นแม่เหล็กไฟฟ้านี้อาจเป็นคลื่นวิทยุ (radio) หรืออินฟาเรด (Infrared) ก็ได้

ระบบเครือข่ายไร้สาย (Wireless LANs) เกิดขึ้นครั้งแรก ในปี ค.ศ. 1971 บนเกาะฮาวาย โดย Project ของนักศึกษาของมหาวิทยาลัยฮาวาย ที่ชื่อว่า “ALOHNET” ขณะนั้นลักษณะการส่งข้อมูลเป็นแบบ Bi-directional ส่งไป-กลับง่ายๆ ผ่านคลื่นวิทยุ สื่อสารกันระหว่างคอมพิวเตอร์ 7 เครื่อง ซึ่งตั้งอยู่บนเกาะ 4 เกาะโดยรอบ และมีศูนย์กลางการเชื่อมต่ออยู่ที่เกาะๆหนึ่ง ที่ชื่อว่า Oahu

เทคโนโลยีการเข้าถึงข้อมูลความเร็วสูง หรือที่เรียกกันโดยทั่วไปว่า การเข้าถึงบรอดแบนด์ (Broadband Access) มีได้หลายประเภท ทั้งบรอดแบนด์ผ่านสาย (สายทองแดง สายเคเบิล และสายใยแก้วนำแสง) และบรอดแบนด์ไร้สายผ่านทางคลื่นวิทยุ ซึ่งปัจจุบันสามารถส่งผ่านข้อมูลด้วยความเร็วสูงยิ่งขึ้น เนื่องจากมีรูปแบบของการมอดูเลชั่นที่ใช้คลื่นวิทยุได้คุ้มค่ามากขึ้น เทคโนโลยี บรอดแบนด์ในปัจจุบัน หรือที่จะมีใช้งานในอนาคตอันใกล้

เทคโนโลยีการเข้าถึงข้อมูลเฉพาะในส่วนที่เป็นการเข้าถึงไร้สาย (Broadband Wireless Access: BWA) นั้น สามารถแยกกลุ่มออกได้ตามลักษณะของการเข้าถึงได้ดังต่อไปนี้

1. Personal Area Network (PAN) คือเทคโนโลยีการเข้าถึงไร้สายในพื้นที่เฉพาะส่วนบุคคล โดยมีระยะทางไม่เกิน 10 เมตร และมีอัตราการรับส่งข้อมูลความเร็วสูงมาก (สูงถึง 480 Mbps) ซึ่งเทคโนโลยีที่ใช้กันแพร่หลาย ก็เช่น

- Ultra Wide Band (UWB) ตามมาตรฐาน IEEE 802.15.3a
- Bluetooth ตามมาตรฐาน IEEE 802.15.1
- Zigbee ตามมาตรฐาน IEEE 802.15.4

เทคโนโลยีเหล่านี้ใช้สำหรับการติดต่อสื่อสารระหว่างคอมพิวเตอร์และอุปกรณ์ต่อพ่วง (Peripherals) ให้สามารถรับส่งข้อมูลถึงกันได้ และยังใช้สำหรับการรับส่งสัญญาณวิดีโอที่มีความละเอียดภาพสูง (highdefinition video signal) ได้ด้วย

2. Local Area Network (LAN) คือ เทคโนโลยีการเข้าถึงไร้สายในพื้นที่เฉพาะ ซึ่งมักมีระยะทางไม่เกิน 100 เมตร และมีอัตราการรับส่งข้อมูลความเร็วที่สูงถึงระดับ 100 Mbps และติดตั้งสถานีฐานที่เรียกว่า Access Point เพื่อทำหน้าที่เชื่อมต่อสัญญาณระหว่างอุปกรณ์ปลายทาง (Terminal Equipment) ในลักษณะที่เป็นเซลล์ขนาดเล็กมาก (pico cells) ที่ไม่แตกต่างจากเซลล์ของระบบโทรศัพท์เคลื่อนที่มากนัก ซึ่งเทคโนโลยีใช้กันแพร่หลาย คือ

- WiFi ตามมาตรฐาน IEEE 802.11และมาตรฐานที่พัฒนาจากมาตรฐานดังกล่าว

• ETSI HIPERLAN ตามมาตรฐานของกลุ่มประเทศยุโรป

ข้อจำกัดสำหรับการใช้งานเทคโนโลยีนี้ คือ จำนวนของผู้ใช้งานในขณะใดขณะหนึ่งพร้อมกัน ระยะห่างระหว่าง Access point กับ terminal equipment และความพอเพียงของคลื่นความถี่ เนื่องจากส่วนใหญ่จะเป็นการใช้งานในลักษณะได้รับยกเว้นใบอนุญาต (unlicensed) จึงต้องใช้คลื่นความถี่ร่วมกันกับผู้ประกอบการรายอื่น

3. Metropolitan Area Network (MAN) คือ เทคโนโลยีการเข้าถึงไร้สายในพื้นที่เขตเมืองหรือพื้นที่ขนาดใหญ่ ซึ่งมีระยะทางตั้งแต่ 10 ถึง 50 กม. ขึ้นอยู่กับคลื่นความถี่ที่ใช้งาน และมีอัตราการรับส่งข้อมูลที่ความเร็วสูงในระดับ 15 – 50 Mbps ขึ้นอยู่กับการใช้งาน ว่าเป็น non-line-of-sight (NLOS) หรือ line-of-sight (LOS) โดยเทคโนโลยีที่เป็นที่กล่าวถึงในปัจจุบัน คือ

• WiMAX ตามมาตรฐาน IEEE 802.16 และมาตรฐานที่พัฒนาจากมาตรฐานดังกล่าว

• WiBro ซึ่งเป็นมาตรฐานที่พัฒนาโดยประเทศเกาหลีใต้ ก่อนที่จะได้พัฒนาต่อเนื่องจนถือว่าเป็นส่วนหนึ่งของมาตรฐาน IEEE 802.16

• ETSI HIPERMAN ซึ่งเป็นมาตรฐานของกลุ่มประเทศยุโรป ซึ่งพัฒนาให้ทำงานร่วมกันได้กับมาตรฐาน IEEE 802.16

แต่เดิมนั้น เทคโนโลยีการเข้าถึงดังกล่าว มุ่งเน้นที่การใช้งานแบบประจำที่ (Fixed) ซึ่งอุปกรณ์ของผู้ใช้บริการจะติดตั้งอยู่กับที่ในลักษณะภายนอกอาคาร (outdoor) เป็นหลัก ก่อนที่จะพัฒนาไปเป็นการใช้งานภายในอาคาร (indoor) แล้วจึงพัฒนาออกแบบให้สามารถใช้งานแบบเคลื่อนที่ (Mobile) ได้ด้วย

อย่างไรก็ตาม ยังมีเทคโนโลยีการเข้าถึงไร้สายในลักษณะดังกล่าว ที่ไม่ได้อ้างอิงมาตรฐานหลักที่กล่าวไว้ข้างต้น และมีใช้งานอยู่ก่อนหน้าที่มาตรฐานทางเทคนิคดังกล่าวจะเป็นที่ยอมรับในวงกว้าง โดยเป็นมาตรฐานเฉพาะของผู้ผลิตแต่ละราย (Proprietary) ที่อาจมีข้อจำกัดในการใช้งานร่วมกันกับระบบของผู้ผลิตรายอื่น อุปกรณ์ที่ใช้เทคโนโลยีในลักษณะนี้ มักเรียกกันโดยทั่วไปว่า Pre-WiMAX

4. Wide Area Network (WAN) คือ เทคโนโลยีการเข้าถึงไร้สายบริเวณกว้าง ที่อาจครอบคลุมพื้นที่ทั่วประเทศ หรือเขตภูมิภาค แต่จะมีอัตราการรับส่งข้อมูลที่มีความเร็วได้ไม่เกิน 1.5 Mbps เนื่องจากมุ่งเน้นที่การใช้งานแบบเคลื่อนที่ ทั้งนี้ เทคโนโลยีการเข้าถึงที่มักระบุว่าเป็นเทคโนโลยีการเข้าถึงไร้สายบริเวณกว้าง คือ

• เทคโนโลยีโทรศัพท์เคลื่อนที่ยุคที่ 3 (3G) ซึ่งพัฒนาต่อเนื่องมาจากเทคโนโลยีโทรศัพท์เคลื่อนที่ยุคที่ 2 โดยมีสองมาตรฐานหลัก คือ

•Wideband CDMA (W-CDMA) ที่พัฒนาต่อเนื่องจากเทคโนโลยี GSM

•cdma2000 ที่พัฒนาต่อเนื่องจากจากเทคโนโลยี cdmaOne (IS-95)

• MBWA (Mobile Broadband Wireless Access) ตามมาตรฐาน IEEE 802.20 ซึ่งเป็นมาตรฐานที่เทียบเคียงกันได้กับ IEEE 802.16e (Mobile WiMAX) แต่ยังอยู่ระหว่างการจัดทำมาตรฐาน

Technology	**Standard**	**Application**	**Coverage (m)**	**Frequency (GHz)**
UWB	802.15.3a	Wireless PAN	10	Flexible
Bluetooth	802.15.1	Wireless PAN	10	2.4
ZigBee	802.15.4	Wireless PAN	10	Not identified
WiFi	802.11a	Wireless LAN	100	5
	802.11b	Wireless LAN	100	2.4
	802.11g,n	Wireless LAN	100	2.4
WiMAX	802.16d	Wireless MAN	6400 - 9600	11
	802.16e	Mobile Wireless MAN	1600 – 4800	2 – 6
WCDMA	IMT-2000 (3G)	Wireless WAN	1600 - 8000	1.8, 1.9, 2.1
cdma2000	IMT-2000 (3G)	Wireless WAN	1600 - 8000	0.4, 0.8, 0.9, 1.7, 1.8, 1.9, 2.1
MBWA	802.2	Mobile Wireless WAN	4000 - 12000	3.5

ตารางที่ 2.11 แสดงเทคโนโลยีต่างๆ ที่สำคัญ

2.11.1 คุณสมบัติของ ZigBee

อุปกรณ์ ZigBee ที่ใช้สำหรับโครงงานมีคุณสมบัติต่างๆ ดังต่อไปนี้

- อัตราการส่งข้อมูล 250 kbps (2.4GHz) ซึ่งเป็นช่วงความถี่ที่ใช้กันทั่วโลก
- High throughput และ low latency Duty Cycle ต่ำ (<0.1%)
- มีการเข้าถึง Channel แบบ Channel access using Carrier Sense Multiple Access with Collision Avoidance (CSMA – CA)
- สามารถรองรับ Address ได้ถึง 64 bit IEEE address (65535 network)
- รับประกันการส่งแบบ Full hand shake protocol
- เชื่อมต่อ Topology ได้หลายแบบ เช่น Star, Peer-to-Peer, Mesh

- ใช้พลังงานต่ำ
- ระยะทางการส่งพื้นฐาน 5-500 เมตร

Module	Antenna Type	Outdoor Distance (Visual Line-of-Sight)	Indoor Distance (Office Building)	Indoor Distance (Warehouse)
XBee	Chip	470 ft. (143 m)	80 ft. (24 m)	-
	Whip	845 ft. (258 m)	80 ft. (24 m)	84 ft. (26 m)
XBee-PRO	Chip	1690 ft. (515 m)	140 ft. (43 m)	-
	Whip	4382 ft. (1335 m)	140 ft. (43 m)	355 ft. (108 m)

ตารางที่ 2.11.1 คุณสมบัติของ ZigBee

ZigBee ได้แบ่งตามลักษณะการทำงาน 3 แบบ คือ

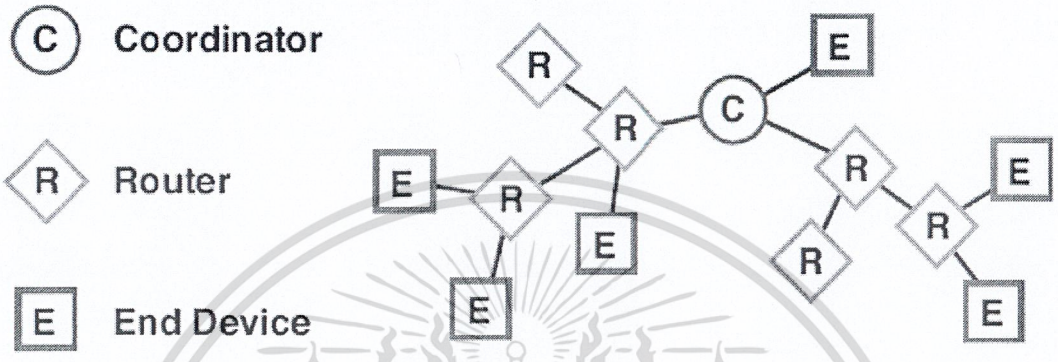
1. Coordinator มีหน้าที่สร้างการสื่อสาร เชื่อมโยงเครือข่าย ระหว่าง End Device กับ Router หรือ Coordinator กับ Coordinator ด้วยกัน หรือ Coordinator กับ Router กำหนด address ให้กับ device ที่อยู่ในวงเครือข่าย ไม่ให้ซ้ำกัน ดูแลจัดการเรื่องการ Routing เส้นทาง ซึ่งเทียบได้กับ FFD

2. End Device เป็นอุปกรณ์ปลายทางสุด ซึ่งจะใช้รับสัญญาณจาก Sensor ที่ปลายทาง โดยที่ใช้พลังงานต่ำในการทำงาน เทียบได้กับ RFD หรือ FFD บางกรณี ขึ้นอยู่กับ sensor ที่ใช้

3. Router มีหน้าที่ รับส่งข้อมูล ในเส้นทางต่าง ๆ ของเครือข่าย ซึ่งเทียบได้กับ FFD

ZigBee Nodes

The ZigBee Protocol defines three types of nodes: Coordinators, Routers and End Device, with a requirement of one Coordinator per network. While all nodes can send and receive data, there are differences in the specific roles they play.



Coordinators are the most capable of the three node types. There is exactly one coordinator in each network and it is the device that establishes the network originally. It is able to store information about the network, including security keys.

Routers act as intermediate nodes, relaying data from other devices.

End Devices can be low-power / battery-powered devices. They have sufficient functionality to talk to their parents (either the coordinator or a router) and cannot relay data from other devices. This reduced functionality allows for the potential to reduce their cost.

ZigBee offers these advantages:

- Open standard with interoperability between vendors
- Option for lower cost, reduced function end nodes

Feature	Series1	Series2	Series1 Pro	Series2 Pro
Power Input	3.3V @ 50mA	3.3V @ 40mA	3.3V @ 215mA	3.3V @ 295mA
Max data rate (Air)	250kbps	250kbps	250kbps	250kbps
Power Output	1mW output (+0dBm)	2mW output (+3dBm)	60mW output (+18dBm)	50mW output (+17dBm)
Distance	300ft (100m) range	400ft (120m) range	1 mile (1500m) range	1 mile (1600m) range
Antenna	Wire,Chip,UFL,SMA	Wire,Chip,UFL,SMA	Wire,Chip,UFL,SMA	Wire,Chip,UFL,SMA

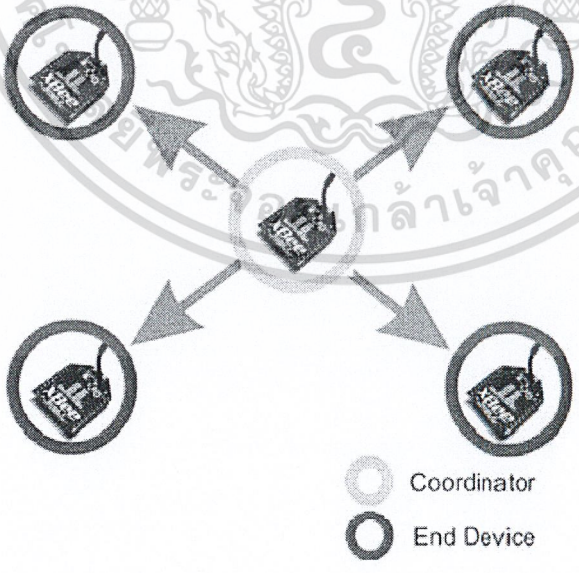
Peripheral	6 10-bit ADC input pins 8 digital IO pins	6 10-bit ADC input pins 8 digital IO pins	6 10-bit ADC input pins 8 digital IO pins	6 10-bit ADC input pins 8 digital IO pins
Upgrade Firmware	Local	Local(ZNET2.5) or over-air configuration(ZB)	Local	Local(ZNET2.5) or over-air configuration(ZB)
Network	Point to point and multi-point networks	Point to point / multi-point / Mesh Network	Point to point and multi-point networks	Point to point / multi-point / Mesh Network

ตารางที่ 2.11.2 เปรียบเทียบคุณสมบัติของ XBee แต่ละรุ่น

XBee Topology

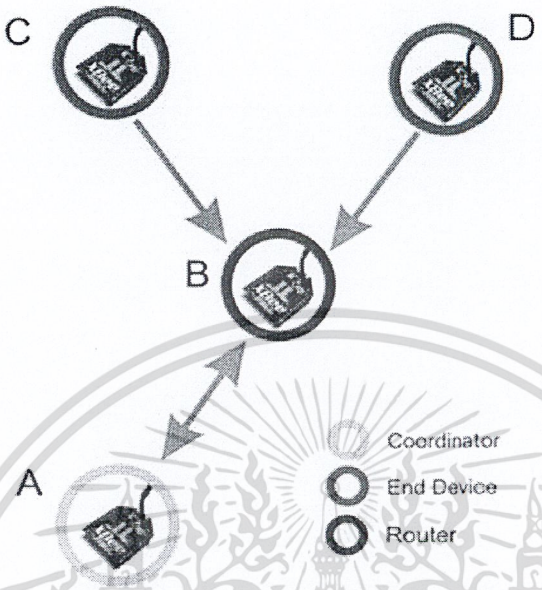
ในการสร้างโครงข่ายไร้สายของ ZigBee นั้น จะต้องประกอบด้วย Node จำนวนอย่างน้อยที่สุด 2 ชนิด คือ Coordinator node และ node ลูกข่าย ชนิดใดชนิดหนึ่ง (Router/End device) จึงจะสามารถสื่อสารและทำงานในรูปแบบของ PAN (Personal area network) ได้ โดย ZigBee สามารถแบ่งรูปแบบเครือข่ายได้เป็น 3 รูปแบบ ดังนี้

1) Star (Broadcast)



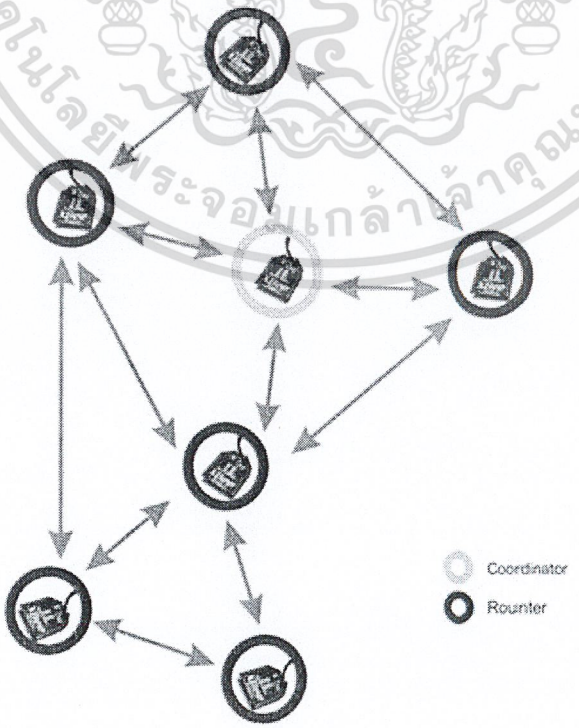
รูปที่ 2.11.1 Star (Broadcast) Network

2) Cluster Tree (Tree)



รูปที่ 2.11.2 Cluster Tree (Tree) Network

3) Mesh



รูปที่ 2.11.3 Mesh Network

XBee ทั้ง 2 Series นี้สามารถสร้าง Topology ได้ดังนี้

1. series1 (รุ่น IEEE802.15.4)

Peer-to-peer, point-to-point, point-to-multipoint (Broadcast)

2. series 2 (รุ่น ZNET2.5 / รุ่น ZB)

Mesh, Peer-to-peer, point-to-point, point-to-multipoint (Broadcast)

ข้อสังเกต สำหรับ XBee series2 ที่ทำ mesh ได้ จะมี parameter มากกว่า series 1 เยอะครับ หากใช้ Series2 ส่งแบบ point-to-point จะยากกว่า Series1

XBee Association

ในเครือข่าย ZigBee ต้องมีการทำงานในโหมดประหยัดพลังงาน ในช่วงเวลาที่ไม่มีการทำงานรับส่งข้อมูล ดังนั้นตัว XBee จึงมี Parameter ที่จะกำหนดการทำงานสำหรับ Sleep mode (Parameter A1, A2, SP, ST)

XBee Addressing

ตัว XBee จะสามารถกำหนดค่าประจำตัวอ้างอิงของมัน (Address) 2 แบบ คือ แบบ 16 bit address และ 64 bit address โดย XBee ทุกตัวจะถูกกำหนดค่ามาจากโรงงานเป็น Address 64 bit อยู่แล้ว ซึ่งจะสามารถอ่านค่าได้จาก parameter SH+SL การใช้งาน Address 64 bit สามารถทำได้โดยกำหนด parameter MY ให้มีค่า 0xFFFF หรือ 0xFFFE ส่วน การกำหนด 16 bit addressนั้นทำได้โดย กำหนด parameter MY ให้มีค่าน้อยกว่า 0xFFFE โดยจะเรียกเป็น mode การทำงาน 2 ประเภทคือ

1. Unicast Mode คือ การรับส่งข้อมูล โดยอาศัยหลักการ Acknowledgement คือหากทางด้านส่งนั้น ส่งข้อมูลไป แต่ไม่ได้รับ Ack ตอบกลับจากตัวรับ ก็จะทำการส่งข้อมูลใหม่

2. Broadcast Mode คือการส่งข้อมูลไปยังปลายทางให้ได้รับข้อมูลทุกตัว

XBee Operation Mode

XBee จะสามารถแบ่งช่วงการทำงานได้เป็น 5 แบบ คือ

1. Idle Mode โหมดนี้ จะเป็นโหมดที่ไม่ได้รับส่งข้อมูล ตัว XBee เตรียมที่จะทำงานในโหมดอื่น ๆ ต่อไปทันที หากมีเงื่อนไขบางอย่าง

2. และ 3. Transmit และ Receive Mode คือช่วงที่ XBee มีการรับ หรือ ส่งข้อมูล โดยจะแบ่งลักษณะการทำงานย่อยออกเป็น Direct กับแบบ Indirect, การกำหนด Address ต้นทางและปลายทาง, Clear Channel Assessment และ การตอบรับ Acknowledgement

4. Sleep Mode คือ ช่วงที่ XBee อยู่ในสถานการณ์ทำงานพลังงานต่ำที่สุด เมื่อไม่มีการใช้งาน

5. Command Mode คือ เป็นส่วนการปรับ parameter ของ XBee ซึ่งจะมีการกำหนด 2 แบบคือ แบบ AT command กับแบบ API Command

2.11.2 อธิบายการทำงานของโปรแกรม

1. ใน Procedure ของ Sub Form_Load นั้นเป็นการกำหนดค่าเริ่มต้นและคุณลักษณะให้กับคอนโทรลต่าง ๆ ที่เราได้สร้างไว้บนฟอร์ม

2. คำสั่งที่อยู่ใน Sub Button1_Click นั้นจะเป็นการกำหนดค่าเริ่มต้นในการสื่อสารผ่านพอร์ตอนุกรม โดยเราจะเลือกเปิดใช้งานพอร์ตอนุกรมที่ใช้ในการติดต่อสื่อสารก่อน โดยเลือกได้จากตัวเลือกที่อยู่ใน ComboBox (ในที่นี้สามารถเลือกใช้งานพอร์ตอนุกรมได้ 5 พอร์ต) และคำสั่งที่ใช้กำหนดคุณสมบัติการสื่อสารของพอร์ตอนุกรมมี ดังนี้

```
With SerialPort1
    .PortName   = ComboBox1.Text        'เลือกใช้งานพอร์ตอนุกรมดังรายการที่ปรากฏใน ComBox
    .BaudRate = 9600                    'กำหนดอัตรา Baud Rate = 9600
    .DataBits = 8                       'Data Bits = 8 Bits
    .StopBits = 10.Ports.StopBits.One '1 bit stop
    .Party = 10.Ports.Parity.None   'ไม่มี parity bit
End With
SerialPort1.Open()                  'เปิดการใช้งานพอร์ตสื่อสาร
```

3. คำสั่งที่อยู่ใน Sub Button1_Click เป็นคำสั่งที่ใช้ในการส่งข้อมูลตัวอักษรออกจากพอร์ตอนุกรม ในที่นี้จะส่งข้อความที่อยู่ใน RichTextBox2 ออกจากพอร์ตอนุกรม

4. ส่วนคำสั่งที่อยู่ใน Sub Timer1_Tick นั้น เป็นคำสั่งที่ใช้ในการรับข้อมูลจากพอร์ตอนุกรมเพื่อนำมาแสดงใน RichTextBox1

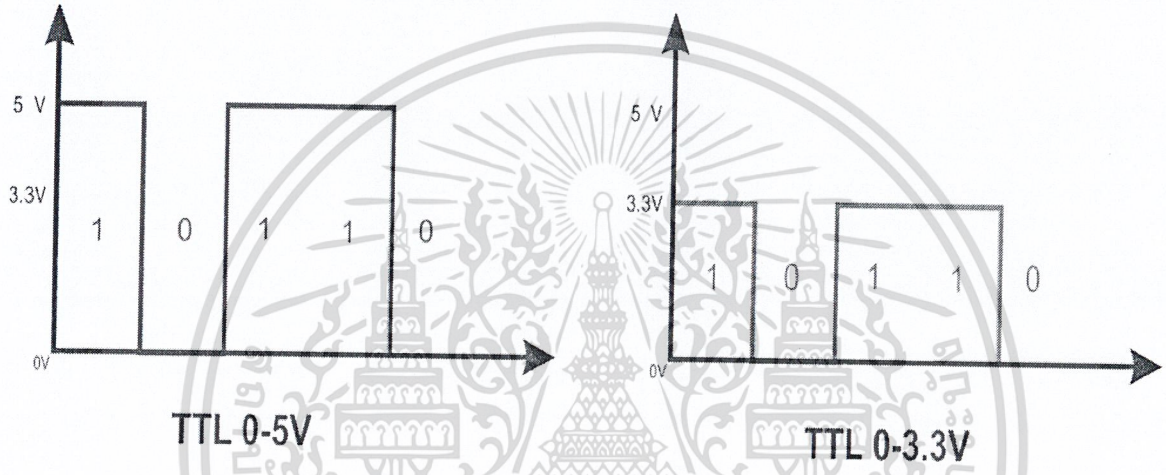
การเชื่อมต่อ

XBee เป็นอุปกรณ์ที่มี Microcontroller และ RF IC อยู่ภายใน ทำหน้าที่เป็น อุปกรณ์ transceiver (อุปกรณ์รับ-ส่งสัญญาณ) แบบ แบบ Half Duplex ย่านความถี่ 2.4 GHz มีการจัดการโดยใช้พลังงานต่ำ ใช้งานง่าย มี interface ที่ใช้รับและส่งข้อมูลกับ XBee เป็น UART (TTL) ซึ่งสำหรับทางด้านไมโครคอนโทรลเลอร์ เรานำขาที่ใช้ติดต่อสื่อสาร UART ของ XBee ต่อเข้ากับ UART ของไมโครคอนโทรลเลอร์ ได้เลย

2.12 UART / TTL / RS232 / MAX232 / MAX3232 คืออะไร

TTL (Transistor-Transistor Logic)

TTL เป็นระดับแรงดันที่ถูกกำหนดขึ้นในยุคแรกๆเพื่อใช้ระหว่าง Transistor กับ Transistor ภายในวงจรรวม(IC) ดังนั้น TTL จะใช้ระดับแรงดัน อยู่ที่ 0 – 5 V แต่ในปัจจุบันมีอุปกรณ์หลายเบอร์ที่ทำงานในช่วง 0 – 3.3 V ซึ่งควรตรวจสอบจาก Datasheet ของอุปกรณ์ที่ใช้เสียก่อนว่าเป็นระดับแรงดันแบบใด เพราะหากใช้ผิดประเภทจะทำให้อุปกรณ์เสียหาย



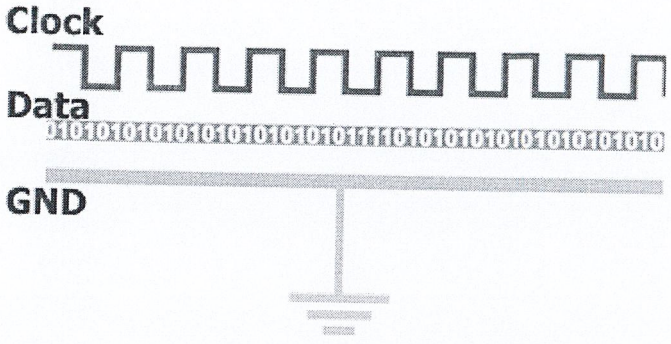
รูปที่ 2.12.1 TTL

UART

ย่อมาจากคำว่า Universal Asynchronous Receiver Transmitter หมายถึงอุปกรณ์ที่ทำหน้าที่รับและส่งข้อมูลแบบอะซิงโครนัส (Asynchronous) ซึ่งเป็นส่วนหนึ่งในการสื่อสารอนุกรม แบบ Asynchronous

การสื่อสารแบบอนุกรมแบ่งเป็น 2 แบบ คือ

1) การสื่อสารอนุกรมแบบ Synchronize เป็นรูปแบบที่ใช้วิธีส่งข้อมูล โดยใช้สัญญาณ Clock มาเป็นตัวกำหนดจังหวะ การรับส่งข้อมูล การส่งข้อมูลแบบนี้ เป็นการรับส่งที่ค่อนค่างมีคุณภาพ มีโอกาสที่ข้อมูลจะสูญหายระหว่างการส่งน้อย แต่มีข้อเสียคือ เป็นการสื่อสารแบบ Half Duplex ไม่สามารถรับและส่งข้อมูลในเวลาเดียวกัน



รูปที่ 2.12.2 การสื่อสารอนุกรมแบบ Synchronize

2) การสื่อสารอนุกรมแบบ Asynchronous เป็นการส่งข้อมูลที่ไม่ต้องใช้สัญญาณ Clock มาเป็นตัวกำหนดจังหวะการรับส่งข้อมูลแต่ ใช้วิธีกำหนด รูปแบบ Format การรับส่งข้อมูลขึ้นมาแทน และอาศัยการกำหนด ความเร็วของการรับ และ ส่ง ที่เท่ากันทั้งฝั่งรับและฝั่งส่ง ข้อดีของการใช้ Asynchronous คือสามารถสื่อสารแบบ Full Duplex รับ และ ส่งได้ในเวลาเดียวกัน แต่ Asynchronous มีโอกาสที่ข้อมูลจะสูญหายขณะรับส่งข้อมูล หรือ รับส่งข้อมูลผิดพลาด ได้มากกว่าแบบ Synchronize

สรุปกล่าวคือ UART (Universal Asynchronous Receiver Transmitter) หมายถึง รูปแบบการส่งข้อมูล ที่ถูกกำหนดขึ้นมาเพื่อใช้รับส่งข้อมูลแบบ Asynchronous โดยมีรูปแบบดังรูป

รูปที่ 2.12.3 รูปแบบการส่งข้อมูลแบบ Asynchronous

เริ่มต้นจาก Start Bit เป็น Logic 0 จากนั้นจะตามด้วย Data ที่เราส่ง แล้วจะถูกปิดด้วย STOP Bit เป็น Logic 1

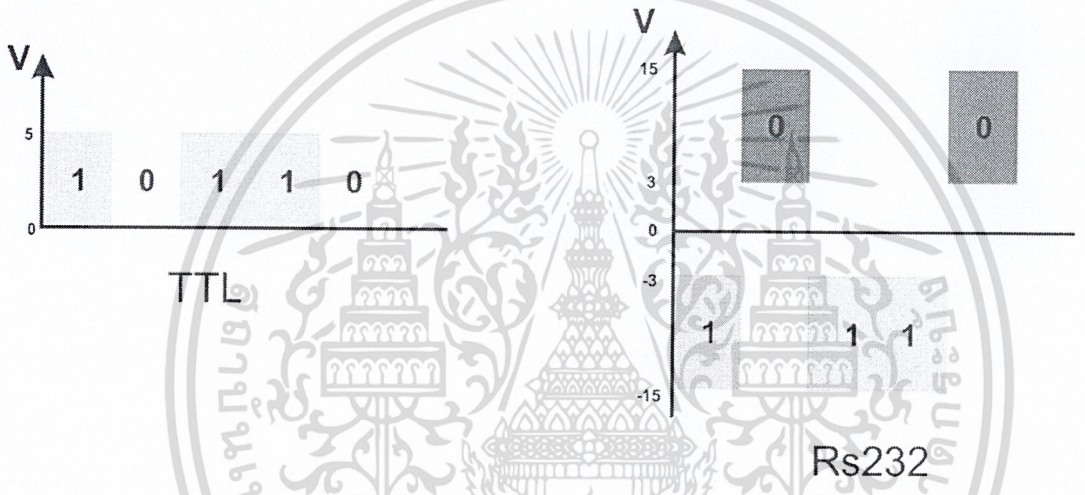
สำหรับผู้เริ่มต้นศึกษา ถ้าคุณต้องการติดต่อสื่อสาร UART ระหว่าง ไมโครคอนโทรลเลอร์ กับคอมพิวเตอร์ สามารถทำได้โดยใช้โปรแกรม HyperTerminal เข้าช่วย เพื่อการทดสอบการ รับ และ ส่งข้อมูล

RS232 (Recommended Standard 232)

RS232 กือ มาตรฐานการเชื่อมต่อข้อมูลแบบ Serial ใช้เพื่อเพิ่มระยะทางในการส่งข้อมูล แบบ Serial ให้สามารถส่งได้ระยะทางที่มากขึ้น โดยมีการเปลี่ยนระดับแรงดัน ของ Logic จากเดิมที่จะอยู่ในช่วง 0-5 V หรือ 0-3.3 V เป็นช่วง -15 ถึง 15 V โดยมีรายละเอียดดังนี้

Logic 0 ของ RS232 จะอยู่ในช่วง 3 ถึง 15V

Logic 1 ของ RS232 จะอยู่ในช่วง -3 ถึง -15V



รูปที่ 2.12.4 TTL และ RS 232

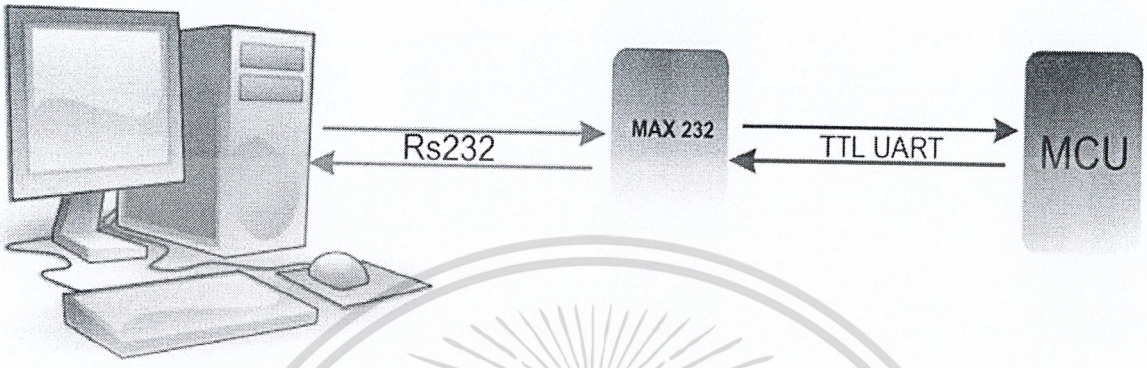
จากรูปจะเห็นได้ว่าทั้ง 2 อย่าง ส่ง Data เหมือนกันครับ แต่ ระดับแรงดันที่ใช้ต่างกันมากครับ หาก อุปกรณ์เป็น TTL แล้ว ไปต่อกับ RS232 ก็จะเกิดความเสียหายตามมาได้

Baud rate	Maximum cable length (ft)
19200	50
9600	500
4800	1000
2400	3000

ตารางที่2.12 แสดงความยาวของสาย และ อัตราความเร็วในการส่งข้อมูล

IC MAX 232

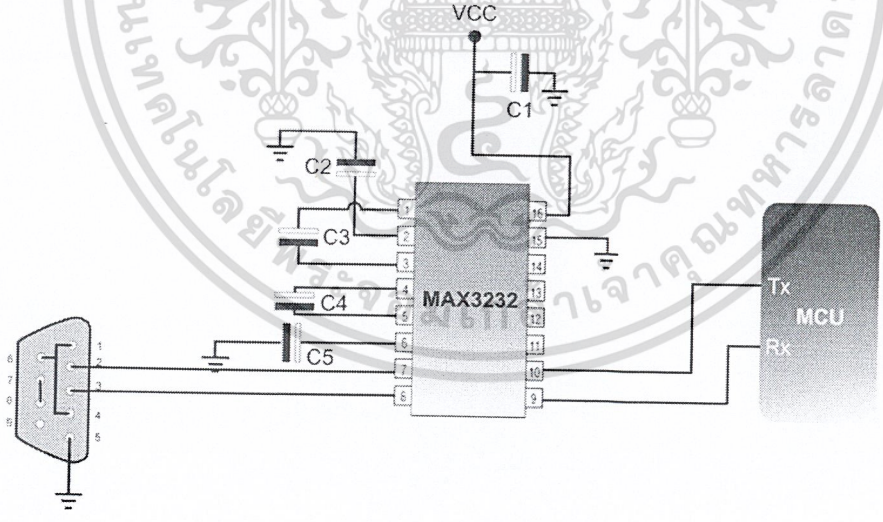
เป็น IC ที่ใช้เปลี่ยน TTL เป็น RS232 ในฝั่งส่ง และ เปลี่ยน RS232 เป็น TTL ในฝั่งรับ ดังรูป



รูปที่ 2.12.4 การทำงานของ IC MAX 232

วิธีต่อใช้งาน MAX 232

วงจรนี้ใช้กับ TTL 0-5V เป็น RS 232



Vcc = 3.3V

C1-C5 = 0.1uF

รูปที่ 2.12.5 วิธีต่อใช้งาน MAX 232

ตัวอย่างการนำไปใช้

จากรูปที่ 2.12.4 เป็นตัวอย่างการเชื่อมต่อ Microcontroller (MCU) กับ PC เนื่องจาก Serial Port ของ PC เป็น มาตรฐาน RS232 แต่ MCU เป็น TTL จึงต้องใช้ max 3232 ปรับระดับแรงดันให้อยู่ในระดับเดียวกัน

การติดต่อกัน ระหว่าง MCU 2 ตัว สามารถต่อ Rx -> Tx , Tx -> Rx กันโดยตรงได้เลยเนื่องจาก ทั้ง 2 ตัวมีระดับแรงดันเป็น TTL เหมือนกัน

MCU ที่ใช้กัน มี 2 ระดับ คือ 0-5V และ 0-3.3V การต่อดังรูปต้องแน่ใจว่า MCU ทั้ง 2 ตัวอยู่ในระดับแรงดันที่เท่ากัน

รูปที่ 2.12.6 ตัวอย่างการนำไปใช้

รูปนี้จะเห็นว่าจะมี MAX3232 ต่อกับ MCU ทั้ง 2 ฝั่ง วิธีต่อแบบนี้มีข้อดีคือสามารถส่งข้อมูลผ่านสายได้ไกลมากขึ้นเนื่องจาก RS 232 ใช้แรงดันในสายสัญญาณสูงทำให้สามารถส่งได้ไกลกว่าใช้ TTL และเมื่อ MCU ตัวหนึ่งส่งข้อมูลมาในรูปแบบ มาตรฐาน RS232 ทำให้อุปกรณ์ หรือ MCU อีกตัวก็จะต้อง รับ และ ส่ง ข้อมูล แบบ RS232 ด้วย

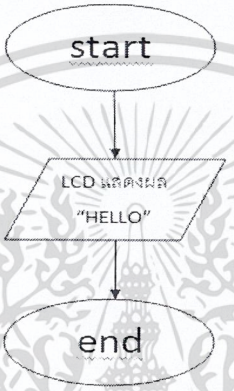
จากตัวอย่างที่ได้แสดงไปข้างต้น นั้น ได้ใช้ MCU เป็นการยกตัวอย่าง ซึ่งหากเราไปต่อใช้กับอุปกรณ์ตัวอื่นๆที่ ไม่ใช่ MCU ก็สามารถใช้หลักการข้างต้นเพื่อตัดสินใจในการต่อวงจร โดยพิจารณาจาก แรงดันเป็นหลักว่า แรงดันจากตัวที่ส่งมาอยู่ในระดับใด และ ฝั่งรับอยู่ในระดับใดและเปลี่ยนให้อยู่ในระดับเดียวกัน

บทที่ 3

การทดลอง

3.1 การแสดงผลของ LCD

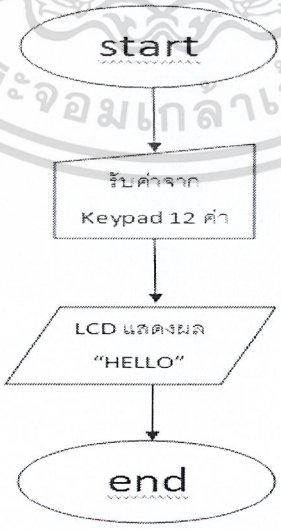
ทำการเชื่อมต่ออุปกรณ์ โดยเขียนโปรแกรมตามโฟลว์ชาร์ตดังรูปที่ 3.1 สังเกตการแสดงผลของ LCD



รูปที่ 3.1 โฟลว์ชาร์ตของโปรแกรมทดสอบการแสดงผลของ LCD

3.2 การรับค่าจาก Keypad ขนาด 4×3

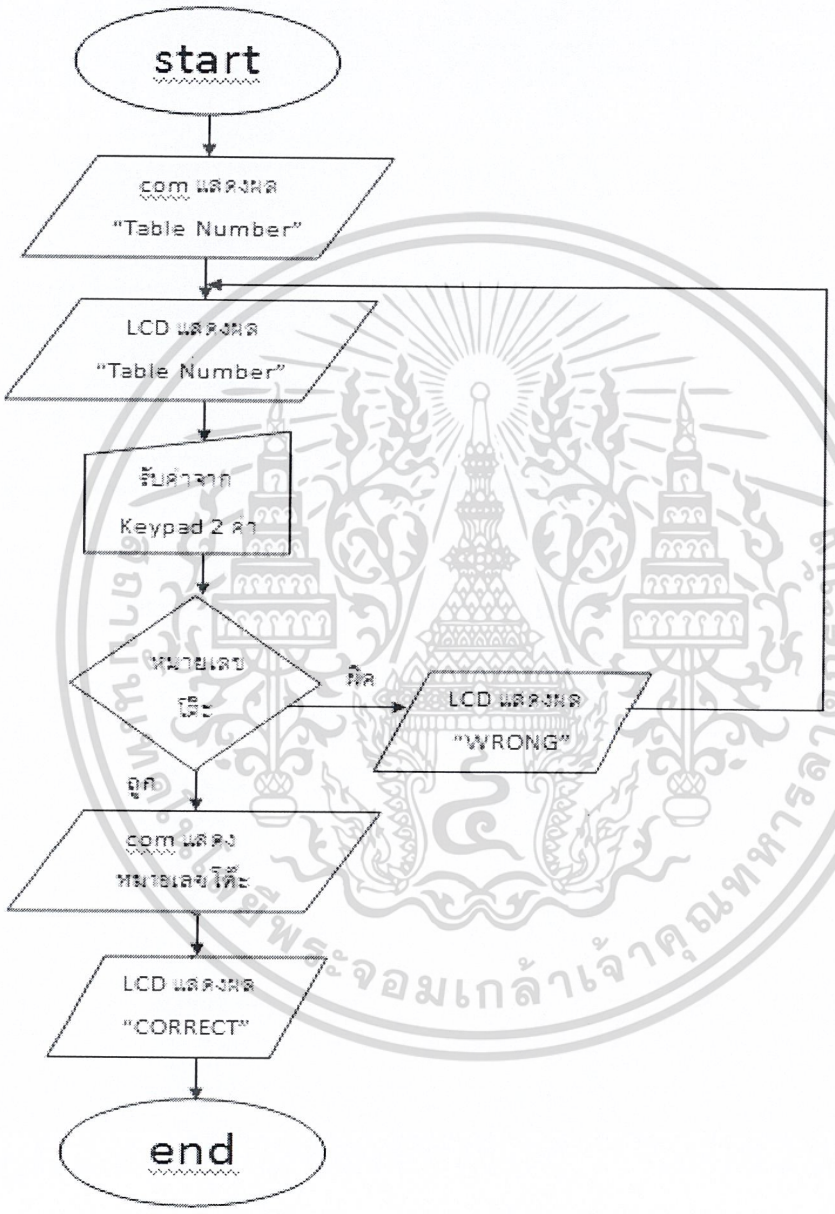
ทำการเชื่อมต่ออุปกรณ์ โดยเขียนโปรแกรมตามโฟลว์ชาร์ตดังรูปที่ 4.2 รับค่าจาก Keypad สังเกตการแสดงผลของ LCD



รูปที่ 3.2 โฟลว์ชาร์ตของโปรแกรมการรับค่าจาก Keypad ขนาด 4×3

3.3.1 ตรวจสอบและรับส่งค่าของหมายเลขโต๊ะผ่านพอร์ตอนุกรม RS 232

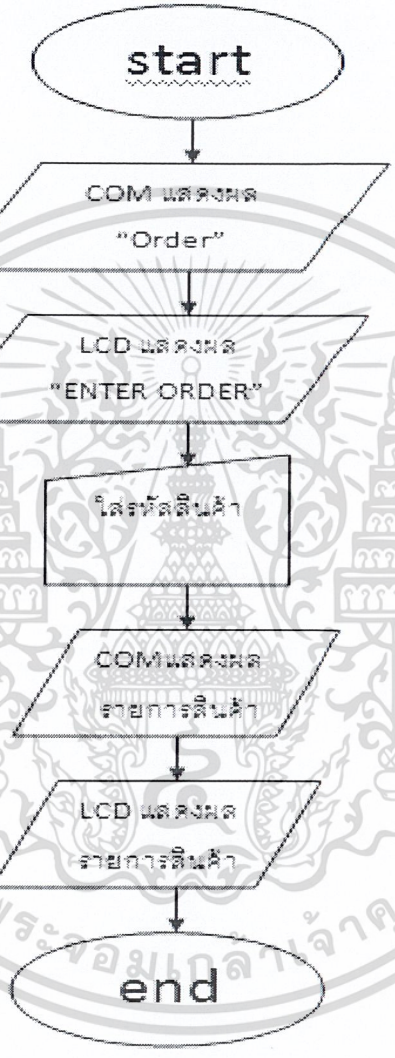
ทำการเชื่อมต่ออุปกรณ์ โดยเขียนโปรแกรมตามโฟลว์ชาร์ตดังรูปที่ 4.3 รับค่าจาก Keypad สังเกตการแสดงผลของ LCD และหน้าจอคอมพิวเตอร์



รูปที่ 3.3.1 โฟลว์ชาร์ตของโปรแกรมทดสอบการตรวจสอบและรับส่งค่าของหมายเลขโต๊ะ

3.3.2 ทดสอบการรับส่งรหัสสินค้าผ่านพอร์ตอนุกรม RS 232

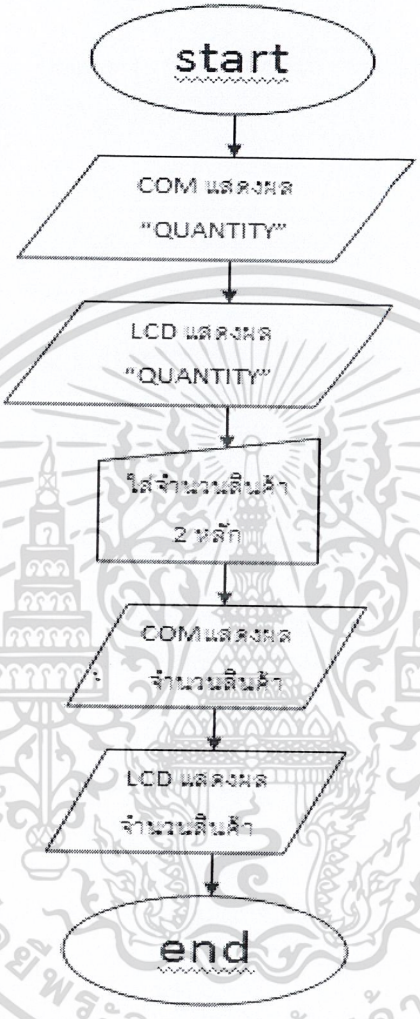
ทำการเชื่อมต่ออุปกรณ์ โดยเขียนโปรแกรมตามโฟลว์ชาร์ตดังรูปที่ 4.4 รับค่าจาก Keypad สังเกตการแสดงผลของ LCD และคอมพิวเตอร์



รูปที่ 3.3.2 โฟลว์ชาร์ตของโปรแกรมทดสอบการรับส่งรหัสสินค้า

3.3.3 ทดสอบการรับส่งจำนวนสินค้าผ่านพอร์ตอนุกรม RS 232

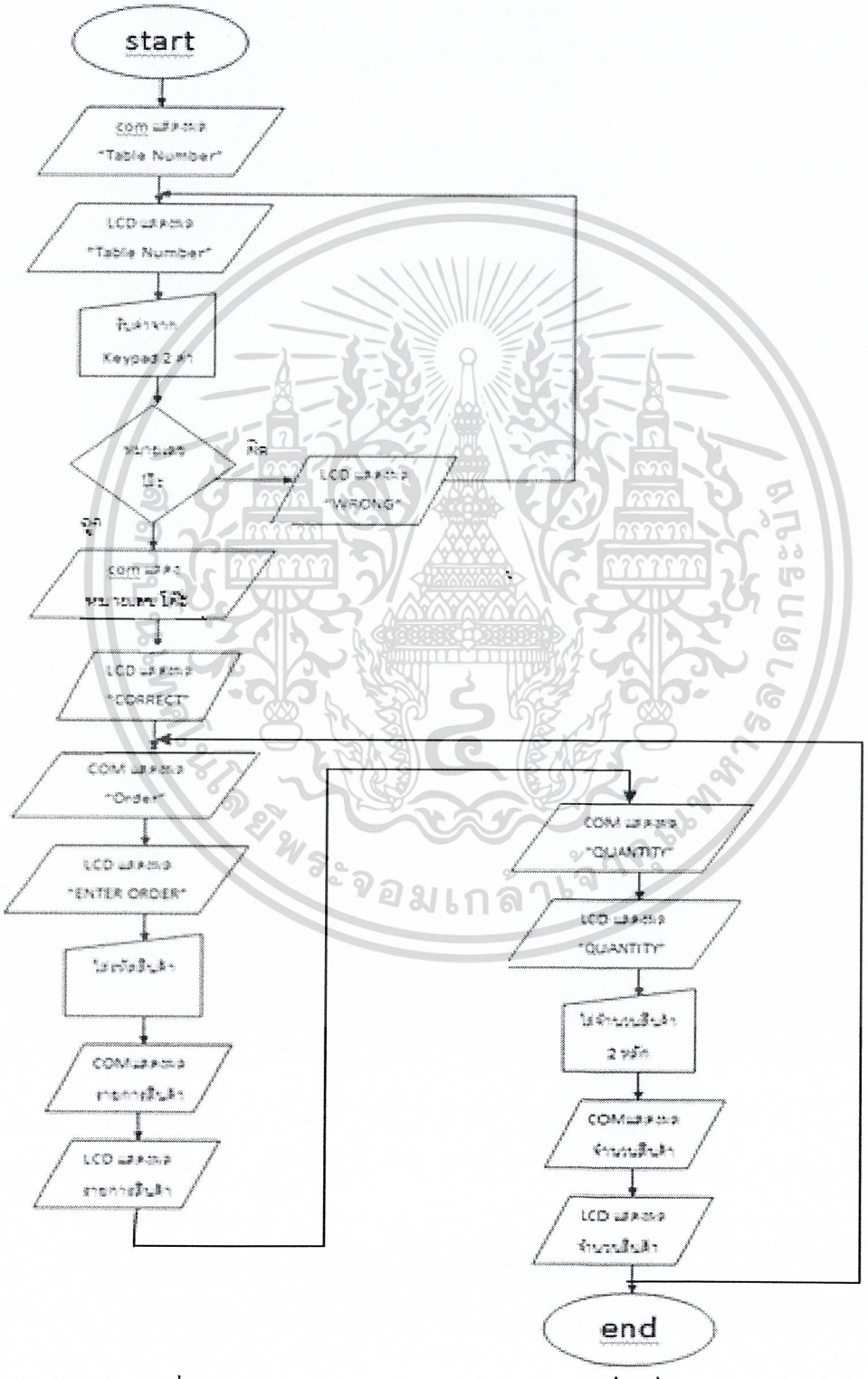
ทำการเชื่อมต่ออุปกรณ์ โดยเขียนโปรแกรมตามโฟลว์ชาร์ตดังรูปที่ 4.5 รับค่าจาก Keypad สังเกตการแสดงผลของ LCD และคอมพิวเตอร์



รูปที่ 3.3.3 โฟลว์ชาร์ตของโปรแกรมทดสอบการรับส่งจำนวนสินค้า

3.3.4 ทดสอบเครื่องสั่งสินค้าผ่านพอร์ตอนุกรม RS 232

ทำการเชื่อมต่ออุปกรณ์ โดยเขียนโปรแกรมตามโฟลว์ชาร์ตดังรูปที่ 4.6 รับค่าจาก Keypad สังเกตการแสดงผลของ LCD และหน้าจอคอมพิวเตอร์



รูปที่ 3.3.4 โฟลว์ชาร์ตของโปรแกรมทดสอบเครื่องสั่งสินค้า

3.4.1 ทดสอบการรับส่งข้อมูลระหว่างคอมพิวเตอร์ 2 เครื่อง ผ่าน XBee

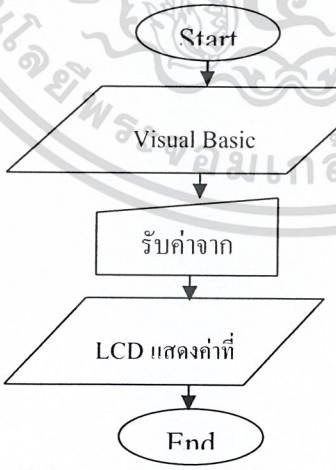
ต่อ XBee 2 ชุดกับ Computer ใช้ไฟไม่เกิน 3.6 Vตรวจสอบการต่อ port ที่ Device Manager เปิดโปรแกรม X-CTU ขึ้นมา 2 ครั้ง เลือก Comport ที่ต่ออยู่แล้วกด Test/Query จะแสดงข้อความสถานะแสดงว่าติดต่อได้ถูกต้อง ไปที่แท็บ Modem Configuration
แล้วไปที่แท็บ Read จากนั้นCheck mask ที่ Always Update firmware ที่ Function set ZIGBEE COORDINATOR AT อีกตัวหนึ่งเป็น ZIGBEE ROUTER AT ตั้ง PAN ID (ID) และ Operating Channel (CH) ให้ตรงกัน นำ Serial Number High (SH)และSerial Number Low (SL) ของตัวที่1ไปใส่ในDestination Address High (DH) และ Destination Address Low (DL) ของตัวที่ 2 ตามลำดับ โดยตัวที่ 2 ให้ทำในลักษณะเดียวกันไปที่แท็บ Write ไปที่แท็บ Terminal พิมพ์ข้อความในช่องฝั่งซ้ายและฝั่งขวาสังเกตผล

3.4.2 ทดสอบการรับส่งข้อความจาก Pic ผ่าน XBee แสดงผลด้วย X-CTU

ทำการเชื่อมต่ออุปกรณ์ โดยเขียนโปรแกรมตามโฟลว์ชาร์ตดังรูปที่ 3.1 แต่เปลี่ยนจากคำว่า HELLO เป็นคำว่า HONEY สังเกตการแสดงผลของ X-CTU

3.5.1 ทดสอบการรับและส่งค่าจาก Visual Basicผ่าน XBee แสดงผลที่หน้าจอLCD

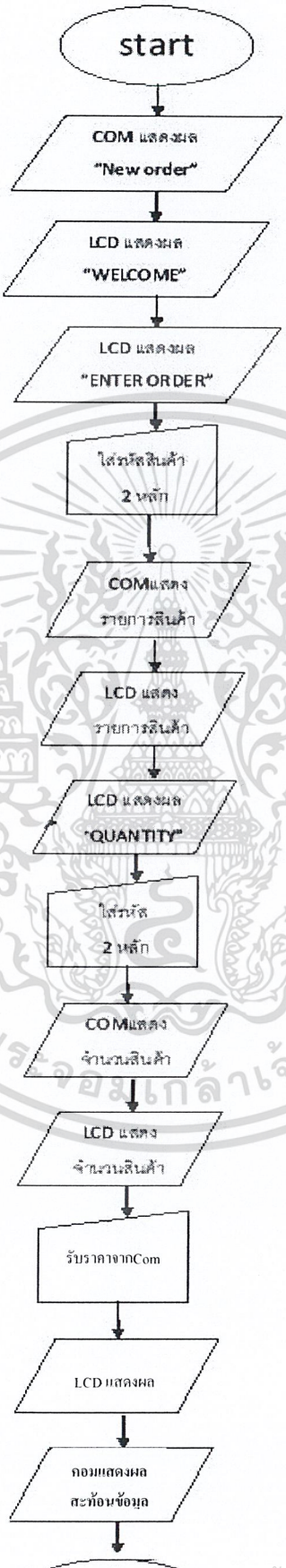
ทำการเชื่อมต่ออุปกรณ์ โดยเขียนโปรแกรมตามโฟลว์ชาร์ตดังรูปที่ 3.5.1 เมื่อมีข้อความเข้ามาที่ Visual Basic ให้คลิกที่ปุ่ม Wait a minute สังเกตการแสดงผลของหน้าจอ LCD



รูปที่ 3.5.1 โฟลว์ชาร์ตของโปรแกรมรับและส่งค่าจาก Visual Basic แสดงผลที่หน้าจอLCD

3.5.2 ทดสอบเครื่องสั่งสินค้าผ่าน XBee แสดงผลด้วย Visual Basic

ทำการเชื่อมต่ออุปกรณ์ โดยเขียนโปรแกรมตามโฟลว์ชาร์ตดังรูปที่ สังเกตการแสดงผลของ Visual Basic



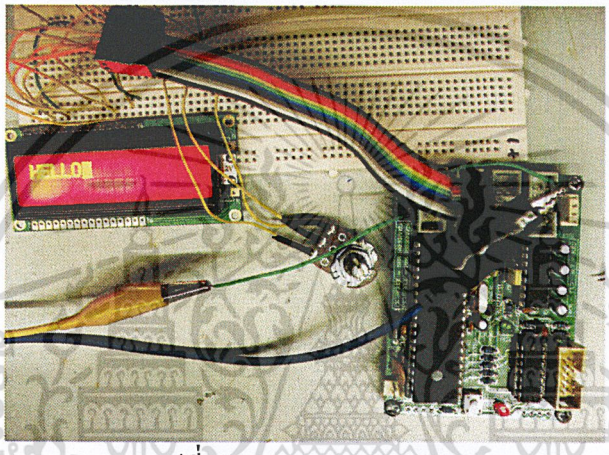
รูปที่ 3.5.2 โฟลว์ชาร์ตของโปรแกรมทดสอบเครื่องสั่งสินค้าผ่าน XBee

บทที่ 4

ผลการทดลอง

4.1 การแสดงผลของ LCD

เมื่อเชื่อมต่ออุปกรณ์ โดยเขียนโปรแกรมตามโฟลว์ชาร์ตดังรูปที่ 4.1 พบว่า LCD แสดงผลคำว่า “HELLO”



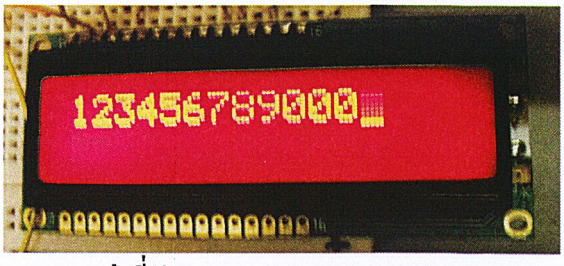
รูปที่ 4.1 การแสดงผล LCD

4.2 ผลทดสอบการรับค่าจาก Keypad ขนาด 4×3

เมื่อเชื่อมต่ออุปกรณ์ โดยเขียนโปรแกรมตามโฟลว์ชาร์ต ดังรูปที่ 4.2 พบว่าเมื่อรับค่าจาก Keypad จะแสดงตัวเลขบนหน้าจอ LCD ได้ 12 ตัว ดังตารางที่ 5.2

Keypad	1	2	3	4	5	6	7	8	9	*	0	#
LCD	1	2	3	4	5	6	7	8	9	0	0	0

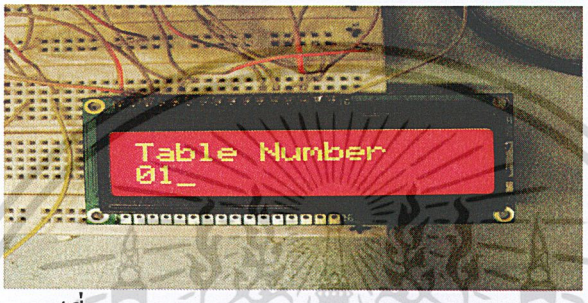
ตารางที่ 4.2 แสดงค่าที่รับจาก keypad และการแสดงผลของ LCD



รูปที่ 4.2 แสดงค่าที่รับจาก keypad และการแสดงผลของ LCD

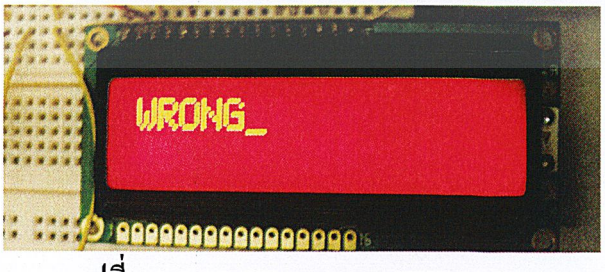
4.3.1 ผลทดสอบการตรวจสอบและรับส่งค่าของหมายเลขโต๊ะผ่านพอร์ตอนุกรม RS 232

เมื่อเชื่อมต่ออุปกรณ์ โดยเขียนโปรแกรมตามโฟลว์ชาร์ตดังรูปที่ 4.3 คอมพิวเตอร์และ LCDแสดงผล “Table Number” รอรับค่าหมายเลขโต๊ะจาก Keypad จนกว่าจำนวนจะครบ 2 หลัก และกดปุ่ม# ตรวจสอบถ้าหลักที่ 1 และ 2 ตรงกับหมายเลขที่ตั้งไว้ คอมพิวเตอร์แสดงผลหมายเลขนั้นและ LCD แสดงผล “CORRECT” แต่ถ้าไม่ใช่ LCDแสดงผล “WRONG” และวนกลับไปให้ใส่หมายเลขโต๊ะใหม่



รูปที่ 4.3.1.1 LCD แสดงผล “Table Number”

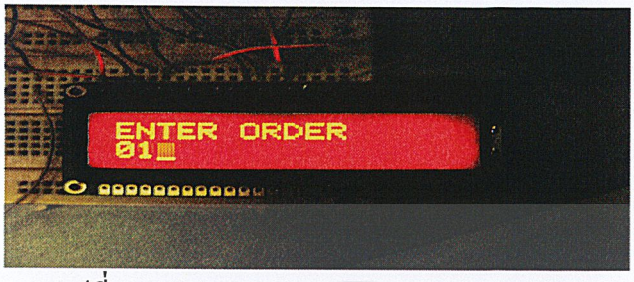
รูปที่ 4.3.1.2 LCD แสดงผล “CORRECT”



รูปที่ 4.3.1.3 LCD แสดงผล “WRONG”

4.3.2 ผลทดสอบการรับส่งรหัสสินค้าผ่านพอร์ตอนุกรม RS 232

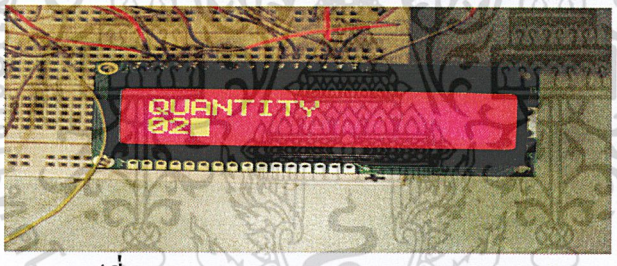
เมื่อเชื่อมต่ออุปกรณ์ โดยเขียนโปรแกรมตามโฟลว์ชาร์ตดังรูปที่ 4.4 คอมพิวเตอร์แสดงผล “Order” LCDแสดงผล “ENTER ORDER” รอรับค่าหมายเลขโต๊ะจาก Keypad จนกว่าจำนวนจะครบ 2 หลักและกดปุ่ม# คอมพิวเตอร์แสดงหมายเลขนั้น



รูปที่ 4.3.2 LCDแสดงผล “ENTER ORDER”

4.3.3 ผลทดสอบการรับส่งจำนวนสินค้าผ่านพอร์ตอนุกรม RS 232

เมื่อเชื่อมต่ออุปกรณ์ โดยเขียนโปรแกรมตามโฟลว์ชาร์ตดังรูปที่ 4.5 คอมพิวเตอร์และ LCD แสดงผล “QUANTITY” รอรับค่าหมายเลขโต๊ะจาก Keypad จนกว่าจำนวนจะครบ 2 หลัก และกดปุ่ม# คอมพิวเตอร์แสดงหมายเลขนั้น



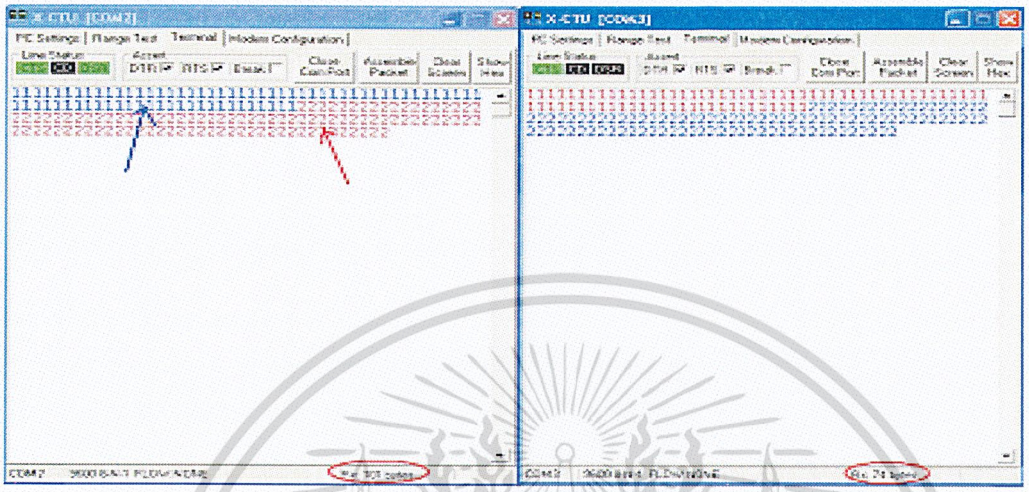
รูปที่ 4.3.3 LCDแสดงผล “QUANTITY”

4.3.4 ทดสอบเครื่องสั่งสินค้าผ่านพอร์ตอนุกรม RS 232

เมื่อเชื่อมต่ออุปกรณ์ โดยเขียนโปรแกรมตามโฟลว์ชาร์ตดังรูปที่ 4.6 คอมพิวเตอร์และ LCDแสดงผล “Table Number” รอรับค่าหมายเลขโต๊ะจาก Keypad จนกว่าจำนวนจะครบ 2 หลัก และกดปุ่ม# ตรวจสอบถ้าหลักที่ 1 และ 2 ตรงกับหมายเลขที่ตั้งไว้ คอมพิวเตอร์แสดงผลหมายเลขนั้นและLCDแสดงผล “CORRECT” แต่ถ้าไม่ใช่ LCDแสดงผล “WRONG” และวนกลับไปให้ใส่หมายเลขโต๊ะใหม่ เมื่อใส่หมายเลขโต๊ะถูกต้องคอมพิวเตอร์แสดงผล “Order” LCDแสดงผล “ENTER ORDER” รอรับค่าหมายเลขโต๊ะจาก Keypad จนกว่าจำนวนจะครบ 2 หลักและกดปุ่ม# คอมพิวเตอร์แสดงหมายเลขนั้น คอมพิวเตอร์และ LCD แสดงผล “QUANTITY” รอรับค่าหมายเลขโต๊ะจาก Keypad จนกว่าจำนวนจะครบ 2 หลักและกดปุ่ม# คอมพิวเตอร์แสดงหมายเลขนั้น

4.4.1 ผลทดสอบการรับส่งข้อมูลระหว่างคอมพิวเตอร์ 2 เครื่อง ผ่าน XBee

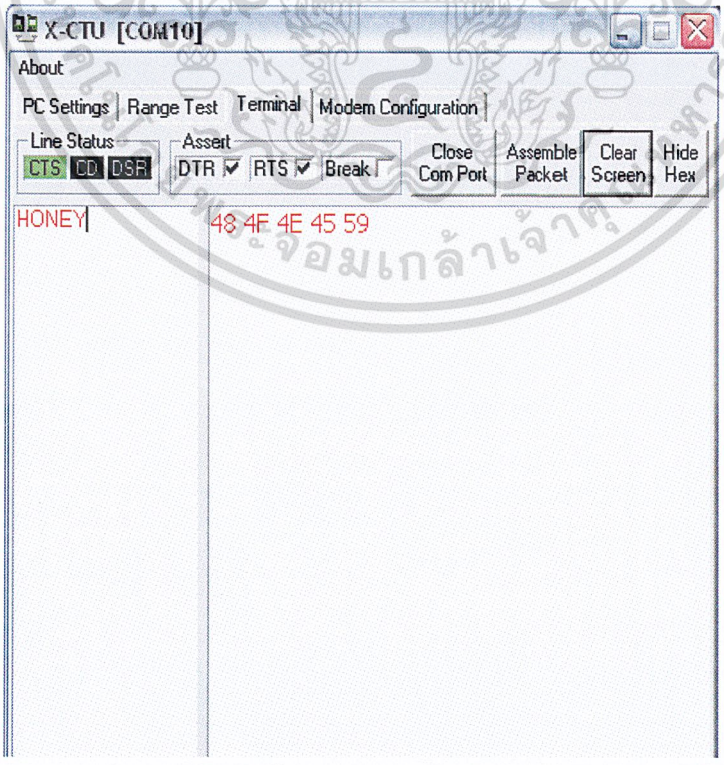
เมื่อ Set ค่า XBee แล้วพิมพ์ข้อความในช่องฝั่งซ้าย (สีน้ำเงิน) จะเห็นข้อความนั้นมาแสดงที่ฝั่งขวา (สีแดง) แต่ถ้าพิมพ์ข้อความในช่องฝั่งขวา (สีน้ำเงิน) จะเห็นข้อความนั้นมาแสดงที่ฝั่งซ้าย (สีแดง) แสดงว่าสามารถติดต่อกันได้



รูปที่ 4.4.1 ผลทดสอบการรับส่งข้อมูลระหว่างคอมพิวเตอร์ 2 เครื่อง ผ่าน XBee

4.4.2 ผลทดสอบการรับส่งข้อความจาก Pic ผ่าน XBee แสดงผลด้วย X-CTU

เมื่อเชื่อมต่ออุปกรณ์ Pic จะส่งข้อความคำว่า “HONEY” เข้ามาที่คอมพิวเตอร์ จากนั้นคอมพิวเตอร์จะแสดงค่าที่ส่งมา ซึ่งเป็นไปตามโฟลว์ชาร์ตที่ 3.1



รูปที่ 4.4.2 ผลทดสอบการรับส่งข้อความจาก Pic ผ่าน XBee แสดงผลด้วย X-CTU

4.5.1 ผลทดสอบการรับและส่งค่าจาก Visual Basicผ่าน XBee แสดงผลที่หน้าจอLCD

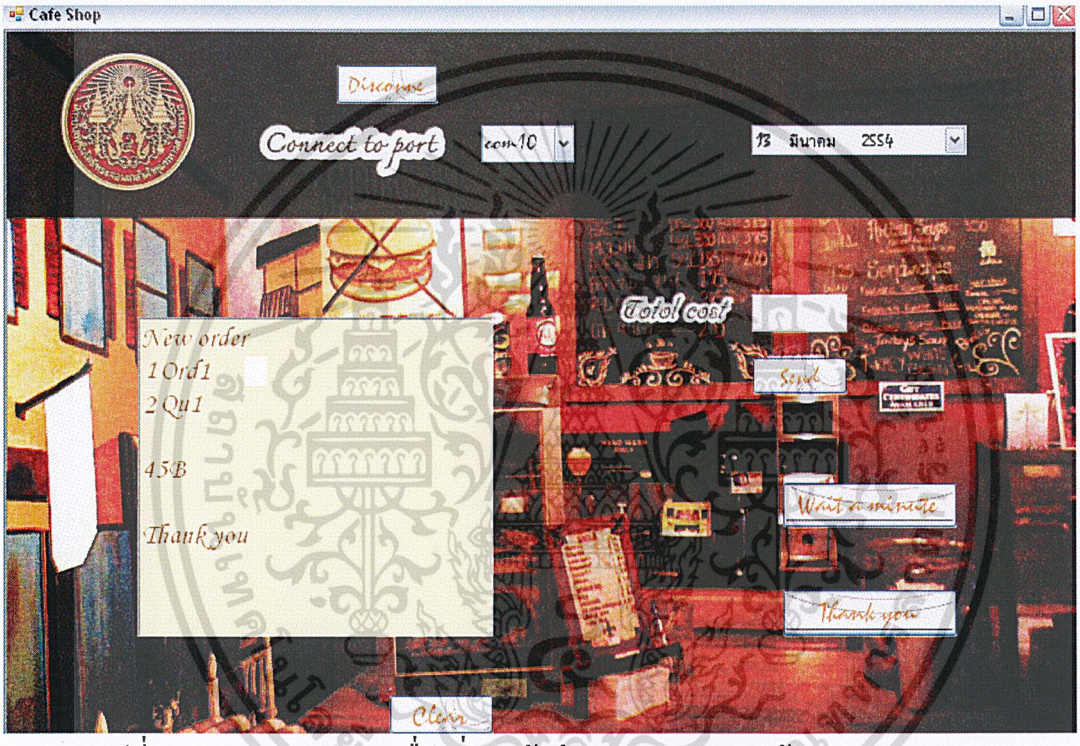
เมื่อเชื่อมต่ออุปกรณ์ Pic จะส่งข้อความคำว่า “Start” เข้ามาที่คอมพิวเตอร์ จากนั้นคอมพิวเตอร์จะแสดงค่าที่ส่งมา เมื่อกดปุ่ม “Wait a minute” บนVisual Basic แล้ว LCDจะแสดงคำว่า “Wait a minute” ซึ่งเป็นไปตามโฟลว์ชาร์ตที่ 3.9

รูปที่ 4.5.1.1 รูปการทดสอบการรับและส่งค่าจาก Visual Basic

รูปที่ 4.5.1.2 ค่าที่รับจาก Visual Basic แสดงผลที่หน้าจอLCD

4.5.2 ผลทดสอบเครื่องสั่งสินค้าผ่าน XBee แสดงผลด้วย Visual Basic

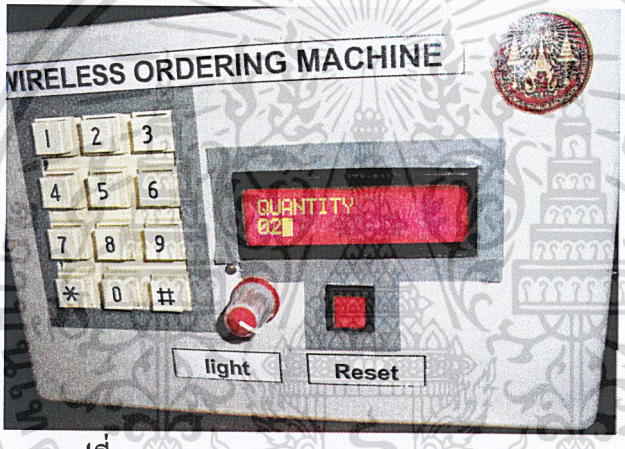
LCDและคอมพิวเตอร์แสดงผลตามโฟลว์ชาร์ตนั่นคือคอมพิวเตอร์แสดงผล “New order” LCD แสดงผล “WELCOME” จากนั้นLCDแสดงผล “ENTER ORDER” รอรับค่ารหัสสินค้าจาก Keypad จนกว่าจำนวนจะครบ 2 หลักและกดปุ่ม# คอมพิวเตอร์และ LCD แสดงรายการสินค้านั้น จากนั้น LCD แสดงผล “QUANTITY” รอรับค่าจำนวนสินค้าจาก Keypad จนกว่าจำนวนจะครบ 2 หลัก และกดปุ่ม# คอมพิวเตอร์และ LCD แสดงหมายเลขนั้น จากนั้นรอรับค่า(ราคาสินค้า)จากคอมพิวเตอร์ กด Send จะปรากฏราคาสินค้าที่ LCD และสะท้อนข้อมูลกลับมาที่คอมพิวเตอร์ เพื่อให้ผู้ส่งทราบว่าข้อมูลได้ส่งเรียบร้อยแล้ว



รูปที่ 4.5.2.1 ผลทดสอบเครื่องสั่งสินค้าผ่าน XBee แสดงผลด้วย Visual Basic

รูปที่ 4.5.2.2 LCDแสดงผล “WELCOME”

รูปที่ 4.5.2.3 LCDแสดงผล “ENTER ORDER”



รูปที่ 4.5.2.4 LCDแสดงผล “QUANTITY”

รูปที่ 4.5.2.5 LCDแสดงราคาที่ส่งมาจากคอมพิวเตอร์

บทที่ 5

สรุปและวิจารณ์

การสั่งสินค้าโดยใช้ โปรแกรมMikroC คอมไพล์PIC16F877A ในการป้อนรหัสสินค้าโดยแสดงผลออกทางหน้าจอ LCD และคอมพิวเตอร์เป็นไปตามโฟล์วชาร์ทที่ตั้งไว้ คือ สั่งรายการสินค้า และจำนวนของสินค้า โดยส่งผ่านข้อมูลผ่าน XBee และแสดงผลด้วย Visual Basic นั่นคือ เมื่อเชื่อมต่ออุปกรณ์ LCDและคอมพิวเตอร์แสดงผลตามโฟลว์ชาร์ตโดยคอมพิวเตอร์แสดงผล “New order” LCDแสดงผล “WELCOME” จากนั้นLCDแสดงผล “ENTER ORDER” รอรับค่ารหัสสินค้าจาก Keypad จนกว่าจำนวนจะครบ 2 หลักและกดปุ่ม# คอมพิวเตอร์และ LCD แสดงรายการสินค้านั้น จากนั้นLCD แสดงผล “QUANTITY” รอรับค่าจำนวนสินค้าจาก Keypad จนกว่าจำนวนจะครบ 2 หลักและกดปุ่ม# คอมพิวเตอร์และ LCD แสดงหมายเลขนั้น จากนั้นรอรับค่า(ราคาสินค้า)จากคอมพิวเตอร์ กด Send จะปรากฏราคาสินค้าที่ LCD และสะท้อนข้อมูลกลับมาที่คอมพิวเตอร์เพื่อให้ผู้ส่งทราบว่าข้อมูลได้ส่งเรียบร้อยแล้ว ซึ่ง โปรแกรมดังกล่าวได้มาจากการรวมโค๊ด โปรแกรมจะโปรแกรมทดสอบย่อย เพื่อใช้ตรวจสอบการทำงานของอุปกรณ์และโปรแกรมย่อยแต่ละส่วนว่าสามารถทำงานได้ปรกติหรือไม่ จากการทดลองพบว่าการแสดงผลของ LCD ,การรับค่าจากคีย์แพด,การแสดงผลของ Visual Basic และการเชื่อมต่ออุปกรณ์โดยรับส่งข้อมูลผ่าน XBee สามารถทำงานได้ปรกติ

วิจารณ์ผล

จากการทดลองพบว่าผู้จัดทำได้ใช้ XBee Pro seriese2 ในการส่งข้อมูลแบบไร้สาย โดยต่อและเซตค่าอุปกรณ์ตามคู่มือการใช้ แต่ระหว่างการทดลองการรับส่งข้อมูลระหว่าง Pic กับคอมพิวเตอร์ผ่าน XBee ค่าที่รับไม่ตรงกับค่าที่ส่งทำให้ต้องใช้เวลาในการหาสาเหตุที่ค่าในการรับส่งไม่เท่ากัน จากการศึกษาพบว่าการต่อ MAX3232 เพิ่มระหว่าง Pic กับ XBeeตัวส่งก่อ จึงจะทำให้ค่าที่ได้ไม่เพี้ยน เนื่องจาก pic16F877 ใช้ TTL ที่ 5 V (จากBoard มีการต่อ MAX232 ไว้แล้ว) จึงต้องปรับให้เข้ากับ XBee ที่ใช้ TTL 3.3 V

บรรณานุกรม

[1] Advance Pic Microcontroller in C ประยุกต์ใช้งาน Pic ขั้นสูงด้วยภาษา C กรุงเทพ: สมาร์ทเลิร์นนิ่ง 2553: 417หน้า

[2] เครื่องสั่งอาหารไร้สาย (WIRELESS FOOD ORDERING MANCHINE) ปริญญานิพนธ์ปีการศึกษา 2544 ภาควิชาวิศวกรรมโทรคมนาคม สถาบันเทคโนโลยี-พระจอมเกล้าเจ้าคุณทหารลาดกระบัง

[3] www.etteam.com

[4] www.alldatasheet.com/

[5] www.ThaiEasyElec.com

ภาคผนวก

โค๊ดโปรแกรมภาษาC with MikroC Pro 3.2

```
unsigned short kp;
char  keypadPort at PORTD;
sbit LCD_RS at RC4_bit;
sbit LCD_EN at RC5_bit;
sbit LCD_D4 at RC0_bit;
sbit LCD_D5 at RC1_bit;
sbit LCD_D6 at RC2_bit;
sbit LCD_D7 at RC3_bit;


sbit LCD_RS_Direction at TRISC4_bit;
sbit LCD_EN_Direction at TRISC5_bit;
sbit LCD_D4_Direction at TRISC0_bit;
sbit LCD_D5_Direction at TRISC1_bit;
sbit LCD_D6_Direction at TRISC2_bit;
sbit LCD_D7_Direction at TRISC3_bit;
void main() {
  unsigned char password[2]={0,1};
  unsigned char input[2];
  unsigned char keypad[17]={0,1,2,3,0,
                            4,5,6,0,
                            7,8,9,0,
                            0,0,0,0};
  unsigned int kp,j,d,l;
  unsigned char text[]="New order";
  unsigned char i,b;
  unsigned char receive[]="1 Qu1";
  unsigned char receive2[]="2 Qu1";
  unsigned char latte[2]={0,1};
   unsigned char capucino[2]={0,2};
  unsigned char mocca[2]={0,3};
```

```
  unsigned char ESP[2]={0,4};
 unsigned char in[2];
 unsigned char text1[]="1 Ord1";
 unsigned char text2[]="2 Ord1";
 unsigned char text3[]="3 Ord1";
 unsigned char text4[]="4 Ord1";
 unsigned char text6[]="hot1";
 unsigned char text7[]="cold1";
 unsigned char k,c,m;
 unsigned char pukan[2];
 char uart_rd;
 unsigned char q2[2]={0,2};
UART1_Init(9600);
 Delay_ms(1000);
for(i=0;i<9;i++)UART1_Write(text[i]);          //คอมแสดง"New order"
UART1_Write(0x0D);                             //คอมขึ้นบรรทัดใหม่
    Lcd_Init();
  Keypad_Init();
   kp = Keypad_Key_Press();
     switch (kp) {
  case  1: kp = 49; break; // 1
  case  2: kp = 50; break; // 2
  case  3: kp = 51; break; // 3
  case  4: kp = 52; break; // 4
  case  5: kp = 53; break; // 5
  case  6: kp = 54; break; // 6
  case  7: kp = 55; break; // 7
  case  8: kp = 56; break; // 8
  case  9: kp = 57; break; // 9
  case 10: kp = 42; break; // *
  case 11: kp = 48; break; // 0
```

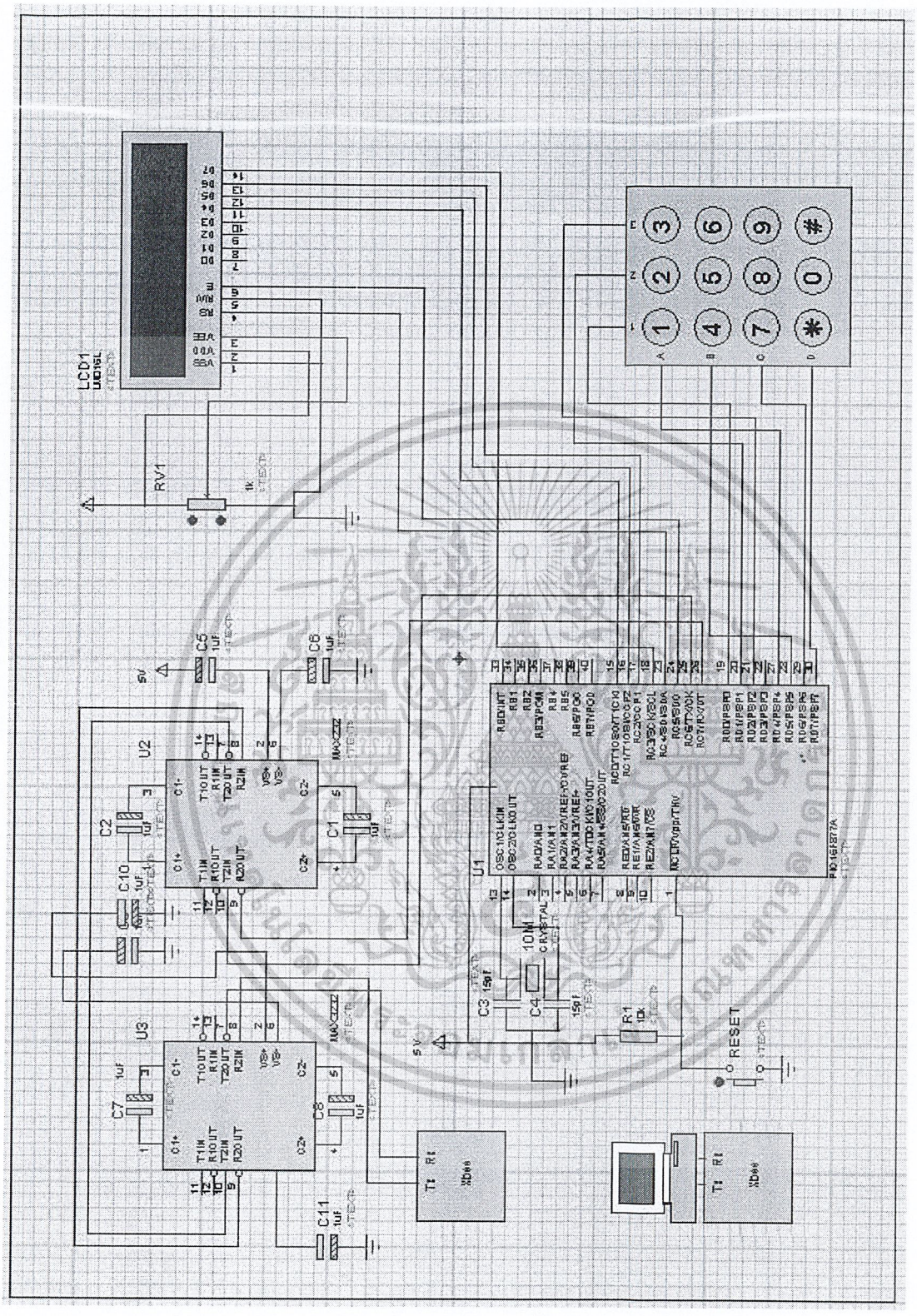
```
    case 12: kp = 35; break; // #
    default:kp +=48;

  }
   RETURN:Lcd_Cmd(_LCD_CLEAR);
   Lcd_Out(1,1,"WELCOME");                    //Lcdแสดง"WELCOME"

   delay_ms(1000);
ADDITIONAL:Lcd_Cmd(_LCD_CLEAR);
   Lcd_Out(1,1,"ENTER ORDER");                //Lcdแสดง"ENTER ORDER"
  for(d=0;d<2;d++)
  {
  while(!Keypad_Key_Press());
  kp=Keypad_Key_Press();                      //รับค่าจากKeypad
  in[d]=keypad[kp];
  Lcd_Chr(2,d+1,in[d]+48);                    //Lcdแสดงค่าที่รับ
  while(Keypad_Key_Press());
  }
  while(Keypad_Key_Press()!=15);
  if(in[0]==latte[0]&&in[1]==latte[1])        //เปรียบเทียบค่า=01
  {
       for(k=0;k<6;k++)UART1_Write(text1[k]);  //คอมแสดง"1 Ord1"
       Lcd_Cmd(_LCD_CLEAR);
       Lcd_Out(1,1,"LATTE");}                 //Lcdแสดง"LATTE"
  if(in[0]==capucino[0]&&in[1]==capucino[1])  //เปรียบเทียบค่า=02
  {
       for(c=0;c<6;c++)UART1_Write(text2[c]);  //คอมแสดง"2 Ord1"
       Lcd_Cmd(_LCD_CLEAR);
       Lcd_Out(1,1,"CAPUCINO");}              //Lcdแสดง" CAPUCINO "
   if(in[0]==mocca[0]&&in[1]==mocca[1])       //เปรียบเทียบค่า=03
```

```
  {
      for(c=0;c<6;c++)UART1_Write(text3[c]);          //คอมแสดง"3 Ord1"
      Lcd_Cmd(_LCD_CLEAR);
      Lcd_Out(1,1,"MOCCA");}                          //Lcdแสดง" MOCCA "
  if(in[0]==ESP[0]&&in[1]==ESP[1])                    //เปรียบเทียบค่า=04
  {
      for(c=0;c<6;c++)UART1_Write(text4[c]);          //คอมแสดง"4 Ord1"
      Lcd_Cmd(_LCD_CLEAR);
      Lcd_Out(1,1,"ESPRESSO");}                       //Lcdแสดง" ESPRESSO "
  UART1_Write(0x0D);                                  //คอมขึ้นบรรทัดใหม่
    delay_ms(1000);
    Lcd_Cmd(_LCD_CLEAR);
  Lcd_Out(1,1,"QUANTITY");                            //Lcdแสดง" QUANTITY "
  for(l=0;l<2;l++)
 {
 while(!Keypad_Key_Press());                          //รับค่าจากKeypad
 kp=Keypad_Key_Press();
 pukan[l]=keypad[kp];
 Lcd_Chr(2,l+1,pukan[l]+48);                          //Lcdแสดงค่าที่รับ
 while(Keypad_Key_Press());
 }
 while(Keypad_Key_Press()!=15);{
 if(pukan[0]==password[0]&&pukan[1]==password[1])      //เปรียบเทียบค่า=01
  for(b=0;b<5;b++)UART1_Write(receive[b]);            //คอมแสดง"1 Qu1"
 if(pukan[0]==q2[0]&&pukan[1]==q2[1])                 //เปรียบเทียบค่า=02
  for(b=0;b<5;b++)UART1_Write(receive2[b]);           //คอมแสดง"2 Qu1"
 UART1_Write(0x0D);                                   //คอมขึ้นบรรทัดใหม่
  }
Lcd_Cmd(_LCD_CLEAR);
  while (1) {               // Endless loop
if (UART1_Data_Ready()) {                             // If data is received,
```

```
    uart_rd = UART1_Read();                    // read the received data,
    Lcd_Chr_CP (uart_rd);                      //Lcdแสดงค่าที่รับ
    UART1_Write(uart_rd);                      // and send data via UART
   }
}
}
```



รูปการเชื่อมต่อของวงจร