ระบบการสั่งอาหารอัตโนมัติ

MENU DIGITAL

นายต่อศักดิ์ สุนทรพันธุ์ รหัส 41360306

นายวัฒนพล ชัยเนตร รหัส 41360470

นายอามร กาพย์แก้ว รหัส 41360561

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

ปีการศึกษา 2544

ใบรับรองโครงงานวิจัย

หัวข้อโครงงาน	ระบบการสั่งอาหารอัตโนมัติ		
ผู้ดำเนินโครงงาน	นายต่อศักดิ์	สุนทรพันธุ์	รหัส 41360306
	นายวัฒนพล	ชัยเนตร	รหัส 41360470
	นายอามร	กาพย์แก้ว	รหัส 41360561
อาจารย์ที่ปรึกษา	อาจารย์ประทีป	ตรีรณโอภาส	
สาขา	วิศวกรรมคอมพิวเตอร์		
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์		
ปีการศึกษา	2544		

.....

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร อนุมัติให้โครงงานฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร วิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมคอมพิวเตอร์

คณะกรรมการสอบโครงงานวิจัย

.....ประธานกรรมการ
(อาจารย์สิทธิโชค เชาวกุล)

.....กรรมการ
(อาจารย์ประทีป ตรีรณโอภาส)

.....กรรมการ
(อาจารย์วัชรีร์ พืชพันธุ์)

.....กรรมการ
(อาจารย์ศิริพร เดชะศิลารักษ์)

หัวข้อโครงงาน	ระบบการสั่งอาหารอัตโนมัติ		
ผู้ดำเนินโครงงาน	นายต่อศักดิ์	สุนทรพันธุ์	รหัส 41360306
	นายวัฒนพล	ชัยเนตร	รหัส 41360470
	นายอามร	กาพย์แก้ว	รหัส 41360561
อาจารย์ที่ปรึกษา	อาจารย์ประทีป[illegible]	ตรีรณโอภาส	
สาขา	วิศวกรรมคอมพิวเตอร์		
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์		
ปีการศึกษา	2544		

.....

บทคัดย่อ

โครงงานนี้เป็นการนำเอาเทคโนโลยีมาออกแบบให้เข้ากับร้านอาหาร โดยที่เราจะมีอุปกรณ์ชนิดหนึ่งที่มีลักษณะเป็นคีย์บอร์ด ใช้รับคำสั่งรายการอาหาร ถ้าต้องการจะสั่งอาหาร ก็ให้มากดคีย์บอร์ดที่เราได้จัดไว้ จากนั้นรายการอาหารก็จะถูกส่งให้คอมพิวเตอร์หลัก เพื่อแสดงให้ทางร้านอาหารได้รับรู้ นอกจากนี้เรายังได้สร้างฐานข้อมูล เพื่อเก็บรายการอาหาร ซึ่งสามารถเพิ่มและลดได้ด้วย

ผลที่ได้จากการทำโครงงานนี้ คือระบบการสั่งอาหารอัตโนมัติผ่านทางคีย์บอร์ด และระบบฐานข้อมูลของร้านอาหาร

Project Title	Menu Digital			
Name	Mr.Tusak	Suntonpan	ID.	41360306
	Mr.Wattanapol	Chainate	ID.	41360470
	Mr.Armorn	Kapkeaw	ID	41360561
Project Advisor	Mr.Pateep	Teeronophas		
Major	Computer Engineering			
Department	Electrical and Computer Engineering			
Academic Year	2544			

ABSTRACT

This project is designed a technology for restaurant. We'll have a one device, is a one keyboard for a one on table, use for sent order to main computer. If the customer would like to sent order then press keyboard. Next, this order 'll show on the main computer for cook.We 'll designed a database system for keep food list in the restaurant that it can add or delete data ,too.

The result of this project is an automatic menu system and a database system

กิตติกรรมประกาศ

โครงงานชิ้นนี้สำเร็จลุล่วงได้โดยความอนุเคราะห์เป็นอย่างดีจากอาจารย์ประทีป ตรีรณโอภาส ซึ่งเป็นอาจารย์ที่ปรึกษาโครงงาน ที่ให้คำปรึกษาและแนะนำแนวทางต่างๆในการทำโครงงาน และขอขอบคุณเพื่อนคนหนึ่งที่ให้คำปรึกษาและช่วยเหลือจนโครงงานนี้สำเร็จได้

สุดท้ายนี้ต้องขอขอบพระคุณ พ่อ แม่ ที่ให้การสนับสนุนและให้กำลังใจมาโดยตลอด ขอขอบพระคุณอาจารย์ทุกท่านที่ประสิทธิ์ประสาทวิชาความรู้ให้ และขอบใจเพื่อนๆทุกคนที่ให้ความช่วยเหลือและเป็นเพื่อนที่ดีมาโดยตลอด

นายต่อศักดิ์ สุนทรพันธุ์
นายวัฒนพล ชัยเนตร
นายอามร กาพย์แก้ว

สารบัญ

หน้า

สารบัญ(ต่อ)

สารบัญรูป

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของโครงงาน

ปัจจุบันโลกของเราได้มีการสร้างเทคโนโลยีใหม่ๆ อย่างมากมาย เพื่อตอบสนองความต้องการของประชากรโลก และเพื่อหลีกหนีความซ้ำซากจำเจ ของชีวิตประจำวัน และยังคงสร้างความสะดวกสบายให้กับมนุษย์ ซึ่งในอนาคตอันใกล้นี้ เทคโนโลยีคงจะเป็นอีกปัจจัยหนึ่งในการดำเนินชีวิต ของมนุษย์ ในปัจจุบัน ร้านอาหาร ได้เกิดขึ้นมากมาย แต่ยังไม่ได้มีการนำเอาเทคโนโลยีมาใช้มากนัก ดังนั้นทางผู้เสนอโครงงานจึงมีความคิดที่จะนำเอาเทคโนโลยีมาประยุกต์ ในการสั่งอาหารและสร้างฐานข้อมูล ของร้านอาหาร เพื่อใช้เป็นกรณีศึกษา

1.2 วัตถุประสงค์

1. เพื่อเป็นการนำเทคโนโลยีมาประยุกต์ใช้ให้เกิดประโยชน์
2. เพื่อตอบสนองความต้องการของผู้บริโภค
3. เพื่ออำนวยความสะดวก
4. เพื่อเป็นสื่อสันแก่วงการร้านอาหารไทย

1.3 ขอบข่ายงาน

1.3.1. ศึกษาทฤษฎีที่เกี่ยวข้องกับสิ่งต่างๆเหล่านี้

1.3.1.1 Database โดยใช้ Microsoft Access 97

1.3.1.2 การทำงานของคีย์บอร์ด

1.3.1.3 การเชื่อมต่อระหว่างคีย์บอร์ดกับคอมพิวเตอร์

1.3.1.4 โปรแกรม Visual Basic 6.0

1.3.1.5 LCD

1.3.2. ออกแบบและจัดทำสิ่งต่างๆต่อไปนี้

1.3.2.1 คีย์บอร์ด

1.3.2.2 การเชื่อมต่อระหว่างคีย์บอร์ดกับคอมพิวเตอร์

1.3.2.3 ฐานข้อมูล

1.3.3. พัฒนาให้ระบบสามารถใช้งานได้จริง

1.4 ขั้นตอนการดำเนินงาน

ขั้นตอนที่ 1 ศึกษาการทำงานของบอร์ดควบคุมไมโครคอลโทลเลอร์ CP-SB31 ว่ามีการทำงานอย่างไร จะนำมาประยุกต์ใช้ในโครงงานได้อย่างไร มีพอร์ตใดบ้างที่จะใช้ ในการติดต่อสื่อสาร จะนำมาใช้ร่วมกับคีย์บอร์ดที่สร้างขึ้นอย่างไร

ขั้นตอนที่ 2 ศึกษาการทำงานของคีย์บอร์ด ออกแบบและสร้างคีย์บอร์ดให้สำเร็จโดยศึกษาว่าคีย์บอร์ดมีการทำงานอย่างไร มีวิธีการสแกนคีย์บอร์ดอย่างไร จะต้องใช้ พอร์ตใดบ้างในการเชื่อมต่อ มีการเชื่อมต่ออย่างไร พร้อมทั้งออกแบบคีย์บอร์ด ว่าต้องการคีย์บอร์ดลักษณะใด มีกี่ปุ่ม และแต่ละปุ่มมีหน้าที่อย่างไร

ขั้นตอนที่ 3 ศึกษาการทำงานของ LCD เขียนโปรแกรมควบคุมการทำงานของ LCD และศึกษาการเชื่อมต่อ LCD เข้ากับ บอร์ด CP-SB31

ขั้นตอนที่ 4 ศึกษาและออกแบบ ลักษณะโครงงาน นำบอร์ด CP-SB31 มาทำงานร่วมกับคีย์บอร์ด และนำไปเชื่อมต่อกับ คอมพิวเตอร์ผ่านพอร์ตอนุกรม

ขั้นตอนที่ 5 เขียนโปรแกรมควบคุมการทำงานทั้งหมดของ คีย์บอร์ด,LCD และไมโครคอลโทลเลอร์ โดยใช้ภาษาซี ซึ่งภาษาซีที่เขียนไปนั้นจะถูกแปลงให้เป็นภาษาแอสเซมบลีโดยโปรแกรม Brief

ขั้นตอนที่ 6 ออกแบบและสร้างโปรแกรมฐานข้อมูลโดยใช้ Visual Basic 6.0 และสร้างฐานข้อมูลโดยใช้ Microsoft Access 97

ขั้นตอนที่ 7 ทดลองการทำงาน ตรวจสอบ และ แก้ไขข้อผิดพลาดของโครงงาน

ขั้นตอนที่ 8 พัฒนาโครงงานให้เสร็จสมบูรณ์พร้อมที่จะนำไปใช้งานจริง

1.5 ผลที่คาดว่าจะได้รับ

1.5.1 ระบบการสั่งอาหารอัตโนมัติ และระบบฐานข้อมูลของร้านอาหาร

1.5.2 ความรู้ความเข้าใจเกี่ยวกับการเชื่อมต่อคอมพิวเตอร์กับอุปกรณ์ภายนอกผ่านพอร์ตอนุกรม

1.5.3 ความรู้ความเข้าใจเกี่ยวกับหลักการทำงานของLCDและคีย์บอร์ด

1.5.4 ความรู้ความเข้าใจเกี่ยวกับโปรแกรม Visual Basic 6.0 และ Microsoft Access 97

บทที่ 2

หลักการและทฤษฎี

2.1 การทำงานของคีย์บอร์ด

คีย์บอร์ดประกอบด้วยส่วนสำคัญ 3 ส่วน คือ

1.ปุ่มกด

2.ส่วนของวงจรเข้ารหัส โดยส่วนนี้จะทำหน้าที่เข้ารหัสเพื่อรับรู้ตำแหน่งของคีย์และสามารถให้ค่าคีย์ ของแต่ละตัวเมื่อมีการกดคีย์บอร์ดได้

3.ส่วนของวงจรถอดรหัสจะเป็นส่วนมี่ทำหน้าที่เปลี่ยนรหัสคีย์ ให้เป็นรหัสที่นำไปใช้งานได้ เช่น รหัส ASCII, รหัส BCD หรือ รหัส HEX เป็นต้น

วงจรคีย์บอร์ดแบ่งได้เป็น 2 ลักษณะใหญ่ ๆ คือ

1.แบบขาร่วม

2.แบบเมตริกซ์

ซึ่งในโครงงานนี้เราใช้วงจรแบบเมตริกซ์ ซึ่งมีหลักการทำงานดังนี้

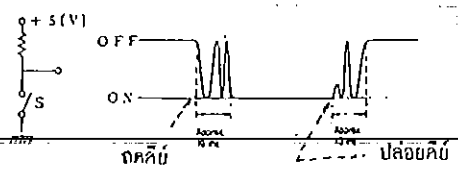
ลักษณะการสแกนแบบเมตริกซ์นี้ จะเริ่มสแกนจากแถวแรกไปยังแถวสุดท้ายและ หลักแรกไปยังหลักสุดท้ายพร้อมกัน ดังนั้นเมื่อคีย์ ใดถูกกดก็จะทำให้ทราบคีย์ที่ถูกกดว่า อยู่แถว และหลักใด

การเกิด SWITCH BOUNE

เนื่องจากเมื่อมีการกด KEY SWITCH ก็จะทำให้หน้าสัมผัสต่อ แต่การต่อหน้าสัมผัสนี้จะไม่เป็นไปตามอุดมคติดังรูปที่ 2.1

รูปที่ 2.1 ลักษณะการกดปุ่มคีย์บอร์ดตามอุดมคติ

ด้วยเหตุว่าหน้าสัมผัสจริง ๆ ไม่ได้ราบเรียบหรือ สะอาดอยู่เสมอและแรงกดของนิ้วบนแป้นกดปุ่มกด ก็ยังแตกต่างกันไป ทำให้เกิดการสะบัดหรือแกว่งของแป้นกดชั่วระยะหนึ่ง หลังจากกดปุ่มลงไปแล้วซึ่งเรียกว่าเป็นการกระเด้งของหน้าสัมผัส (BOUNCING) ซึ่งจะทำให้การต่อวงจรจริงของปุ่มกด จะเป็นดังนี้



รูปที่ 2.2 การเกิด SWITCH BOUNE

จึงทำให้ตัวตรวจจับคีย์ที่กด ตรวจว่าคีย์ดังกล่าวมีการกดหลายครั้ง ทั้ง ๆ ที่ความเป็นจริงผู้ใช้ต้องการกดเพียงครั้งเดียวซึ่งช่วงการเกิดของสัญญาณที่ไม่ต้องการนี้จะมีช่วงเวลาประมาณ 10 ms เราสามารถแก้การ DEBOUNCE ของปุ่มกด ได้โดยการเขียนโปรแกรมให้วนลูบถ้าวนลูบกลับมาเช็คแล้วยังพบว่าคีย์นั้นยังถูกกดอยู่ก็แสดงว่าคีย์นั้นถูกกดจริง แต่ถ้าวนลูบกลับมาแล้วพบว่า ปุ่มกดนั้นไม่ถูกกดแสดงว่าปุ่มกด นั้นเกิดการ boune

2.2 การสื่อสารแบบอนุกรม

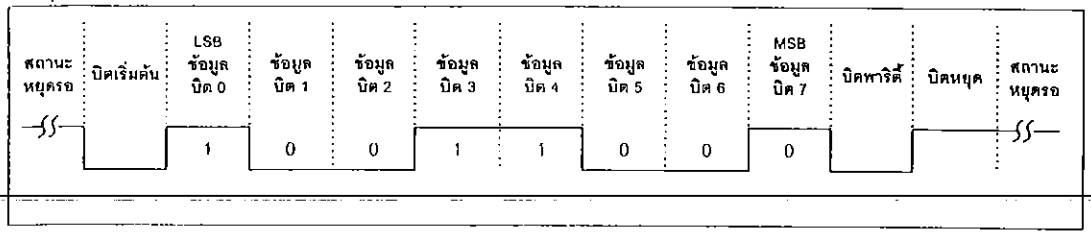
การสื่อสารแบบอนุกรมนั้นแบ่งออกได้เป็น 2 แบบ คือ การสื่อสารแบบอนุกรมแบบซิงโครนัสและการสื่อสารแบบอนุกรมแบบอะซิงโครนัส ซึ่งในโครงงานนี้ใช้การสื่อสารอนุกรมแบบอะซิงโครนัส

การสื่อสารอนุกรมแบบอะซิงโครนัส

การสื่อสารแบบอะซิงโครนัสคือการรับส่งข้อมูลไปในสายโดยไม่จำเป็นต้องมีสัญญานนาฬิการ่วมด้วย แต่จะใช้การกำหนดค่าสัญญาณทั้งภาครับและภาคส่งให้มีค่าเท่ากัน ซึ่งเรียกสัญญาณนาฬิกาที่ใช้ในการกำหนดค่าให้ภาครับและภาคส่งนี้ว่า อัตราการถ่ายทอดข้อมูล หรือ บอดเรต (baudrate) มีหน่วยเป็นบิตต่อวินาที

รูปแบบของข้อมูลที่ใช้ในการรับส่งข้อมูลแบบอะซิงโครนัสประกอบด้วย 4 ส่วนด้วยกัน คือ

1. บิตเริ่มต้น ซึ่งมีขนาด 1 บิต
2. บิตข้อมูล แบบอนุกรมจะมีขนาด 5,6,7, หรือ 8 บิต
3. บิตตรวจสอบพาริตี้ (parity bit) จะมีขนาด 1 บิตหรือไม่มี
4. บิตปิดท้าย (stop bit) จะมีขนาด 1,1.5หรือ 2 บิต



รูปที่ 2.3 รูปแบบอย่างง่ายที่สุดของข้อมูลอนุกรมแบบอะซิงโครนัส

จากรูปแสดงรูปแบบของข้อมูลอนุกรมแบบอะซิงโครนัส ซึ่งเมื่อไม่มีข้อมูลจะส่ง ขาข้อมูลจะมีสถานะ ลอจิก "1" ซึ่งจะเรียกสถานะนี้ว่า สถานะหยุดรอ (waiting state) การเริ่มต้นส่งข้อมูลจะเริ่มจากให้ขา data มีลอจิก "0" ด้วยช่วงระยะเวลา 1 บิต ซึ่งจะเรียกบิตนี้ว่าเป็นบิตเริ่มต้น จากนั้นบิตข้อมูลจะถูกส่งออกไปโดยเริ่มจากบิตที่มีนัยสำคัญต่ำสุด (lsb) ซึ่งข้อมูลในไบต์ที่จะส่งอาจจะมีจำนวนบิต 5,6,7, หรือ 8 บิตก็ได้ จากนั้นก็ตามด้วยบิตพาริตี้ ซึ่งใช้เพื่อตรวจสอบความผิดพลาดจากการส่งข้อมูล บิตสุดท้ายที่จะส่งคือบิตปิดท้าย ซึ่งจะให้ขาdata มีสถานะลอจิก 1 อีกครั้ง ด้วยระยะเวลาอย่างน้อย 1 บิต, 1.5บิต หรือ 2 บิต เพื่อเป็นการแสดงว่าสิ้นสุดข้อมูลแล้ว

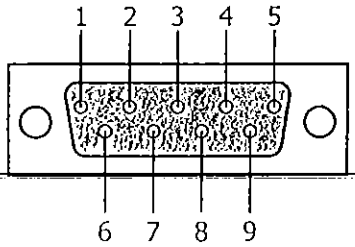
มาตรฐานพอร์ตอนุกรมแบบ RS-232

มาตรฐานการเชื่อมต่อแบบอนุกรม RS-232 เป็นมาตรฐานอุตสาหกรรมที่ออกแบบมาเพื่อใช้ในการส่งข้อมูลอนุกรมแบบอะซิงโครนัส 2 ทิศทาง

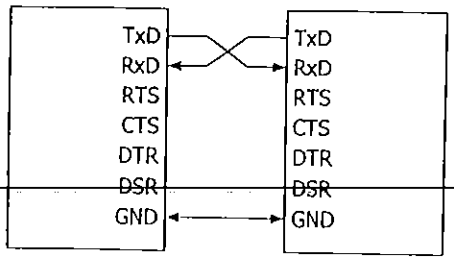
มาตรฐาน RS-232 ได้กำหนดรูปแบบของอนุกรมเชื่อมต่อข้อมูล (Data Terminal Equipment: DTE) กับวงจรข้อมูลปลายทาง (Data Circuit Terminating: DCE) ไว้ว่า อุปกรณ์ DTE จะต้องเป็นอุปกรณ์ที่มีการประมวลผลในตัวเช่น ไมโครคอนโทรลเลอร์ หรือ ไมโครคอมพิวเตอร์ ซึ่งมีความสามารถในการสร้างบิตข้อมูลแบบอนุกรมได้ ส่วนอุปกรณ์ DCE จะทำหน้าที่เป็นเพียงตัวรับข้อมูลมาจาก DTE เท่านั้น โดยการรับส่งข้อมูลระหว่างอุปกรณ์ทั้งสองจะทำผ่านมาตรฐาน RS-232

คอนเน็กเตอร์สำหรับพอร์ต RS-232 และการเชื่อมต่อ

มาตรฐานการเชื่อมต่อแบบRS-232 จะใช้คอนเน็กเตอร์แบบ DB-9 ตัวผู้ โดยมีรูปร่างและตำแหน่งขาดังรูป



รูปที่2.4 คอนเน็กเตอร์อนุกรม 9 ขา



รูปที่ 2.5 การต่ออุปกรณ์ภายนอกเข้ากับคอมพิวเตอร์แบบ RS-232 โดยใช้สายสัญญาณเพียง 3 เส้น

สำหรับการเชื่อมต่อคอมพิวเตอร์กับอุปกรณ์ภายนอกเป็นการเชื่อมต่อแบบ Null Modem ในลักษณะที่ใช้สายสัญญาณเพียง 3 เส้น โดยเส้นหนึ่ง สำหรับส่งข้อมูล อีกเส้นหนึ่งสำหรับรับข้อมูล และเส้นสุดท้ายเป็นกราวด์ สำหรับหน้าที่การทำงานในแต่ละขามีดังนี้

Receive Data : RD หรือ RxD ขานี้ใช้เพื่อรับสัญญานอนุกรม เข้ามายังคอมพิวเตอร์ โดยนำข้อมูลที่อ่านได้เก็บไว้ใน รีจิสเตอร์ บัพเฟอร์

Transmitted Data: TD หรือ TxD ขานี้ใช้เพื่อส่งข้อมูลออกจากคอมพิวเตอร์ โดยนำข้อมูลที่เก็บอยู่ในบัพเฟอร์ สำหรับส่งข้อมูลส่งออกไป

Signal Ground : GND ขากราวด์ของระบบ

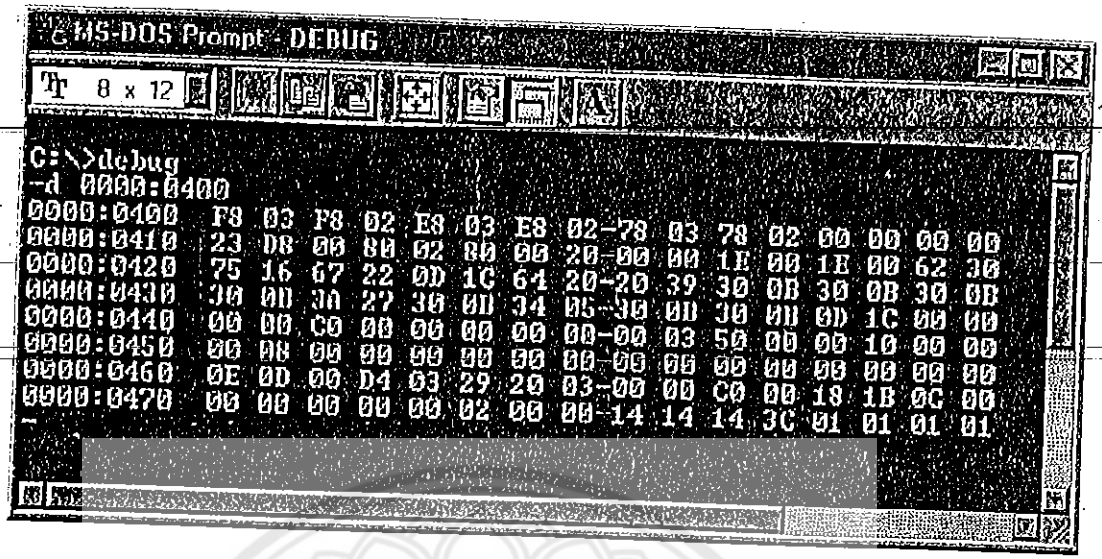
2.2.1 การเขียนโปรแกรมเพื่อใช้งานพอร์ตอนุกรม

การหาค่าตำแหน่งแอดเดรสของพอร์ตอนุกรม

การหาค่าตำแหน่งแอดเดรสของพอร์ตอนุกรมสามารถหาได้หลายวิธีดังนี้

วิธีที่1 โดยการใช้โปรแกรม Debug ไปดูค่าตำแหน่งเอกเดรสที่ตำแหน่ง 0000:0400H โดยใช้พิมพ์คำสั่งที่ดอสพร้อมพ์ดังแสดงในรูปที่ 2.6

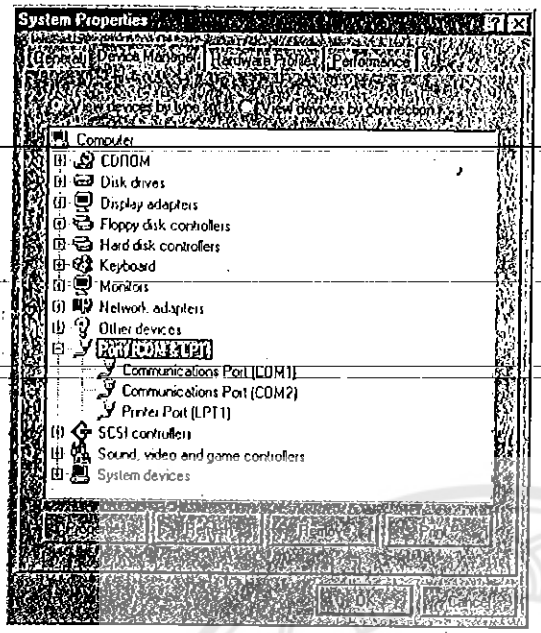
ค่าที่เห็นในรูปแสดงว่า พอร์ตอนุกรมของคอมพิวเตอร์เครื่องนี้มีถึง 4 พอร์ต มีตำแหน่งแอดเดรสที่ไล่เรียงกันตั้งแต่ตำแหน่งที่ 0000:0400H – 0000:0407H ตัวอย่างเช่นที่ตำแหน่งหน่วยความจำ 0000:0400H แสดงตัวเลข F8 03ซึ่งหมายความว่าแอดเดรสของพอร์ต COM1 คือ 03F8H นั่นเอง สำหรับจำนวนของพอร์ตอนุกรมที่ระบุอยู่ที่หน่วยความจำตำแหน่ง 0000:0411H มีค่าเท่ากับ D8Hซึ่งเมื่อแปลงเป็นเลขฐานสองจะได้ค่าเป็น 11011000 บิตที่ 1-3 มีค่าเท่ากับ 100 หมายความว่ามีจำนวนพอร์ตอนุกรม 4 พอร์ต ดังที่ได้กล่าวไปแล้วตอนต้น



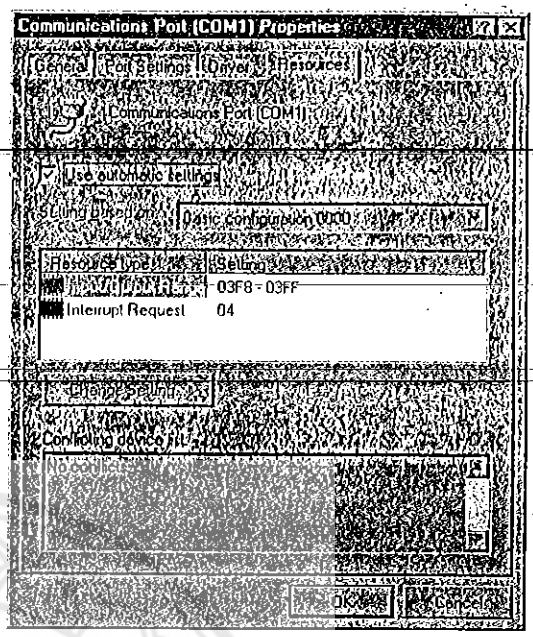
รูปที่ 2.6 หน้าต่างโปรแกรม DEBUG แสดงตำแหน่งของพอร์ตอนุกรม

วิธีที่2 สามารถดูได้จากวินโดวส์ 95 โดยไปที่ Control Panel เรียก System => Device Manager => Ports (COM and LPT) จากนั้นเลือกพอร์ตอนุกรมที่ต้องการดูแล้วดังแสดงในรูปที่ 2-2 แล้วเลือก Properties => Resources ดังแสดงในรูปที่ 2-3 ซึ่งในหน้าต่างนี้จะแสดง ทั้งตำแหน่งแอดเดรสของพอร์ตอนุกรมนั้นๆ รวมถึงตำแหน่งของอินเตอร์รัปต์ที่ใช้ด้วย

วิธีที่3 การเขียนโปรแกรมเพื่ออ่านค่า โดยสามารถใช้โปรแกรมภาษาใดๆ ก็ได้เพื่ออ่านค่า แต่ในตัวอย่างนี้จะใช้ QBASIC ในการแสดงค่าแอดเดรสของพอร์ตอนุกรม โดยจะใช้คำสั่ง PEEK ซึ่งเป็นคำสั่งที่ใช้อ่านค่าจากหน่วยความจำ นอกจากนี้ยังอ่านค่าจำนวนพอร์ตอนุกรมออกมาได้ โดยใช้คำสั่งเดียวกัน แต่จะใช้คำสั่ง AND เข้าช่วยเพื่อเลือกเอาเฉพาะบิตที่ต้องการอ่านค่าเท่านั้น จากนั้นก็ทำการเลื่อนบิตไปทางขวา 1 บิต โดยใช้วิธีการหารด้วย 2 เพื่อให้บิตที่ต้องการไปอยู่ทางด้านขวามือสุดและแสดงค่าจำนวนพอร์ตที่แท้จริงออกมา คำสั่ง HEX$ ช่วยในการแสดงผลตำแหน่งแอดเดรสของพอร์ตอนุกรมที่ออกมาเป็นค่าเลขฐานสิบหก ส่วนการคูณค่าด้วย 100H นั้นก็เพื่อที่จะเลื่อนบิตไปข้างหน้า 1 ไบต์ ทำให้เมื่อเวลานำมาบวกแล้วจะได้ค่าเป็น 2 ไบต์พอดี มีโปรแกรมแสดงตัวอย่างดังนี้



รูปที่ 2.7 ตำแหน่งของพอร์ตอนุกรมบนระบบปฏิบัติการวินโดวส์ 95/98



รูปที่ 2.8 หน้าต่างแสดงรายละเอียดของพอร์ตอนุกรมบนวินโดวส์ 95/98

```
DEF SEG = 0
CLS
PRINT "Address Of COM1 :" , HEX$ ( PEEK ( & H401 ) * &H100 ) + PEEK ( &H400 ))
PRINT "Address Of COM2 :", HEX$  (PEEK  ( & H403 ) * &H100 ) + PEEK ( &H402 ))
PRINT "Address Of COM3 :" , HEX$ ( PEEK ( & H405 ) * &H100 ) + PEEK ( &H404 ))
PRINT "Address Of COM4 :" , HEX$ ( PEEK ( & H407 ) * &H100 ) + PEEK ( &H406 ))
PRINT " Number Of RS-232 Port in This Computer :",( PEEK ( &H411 ) AND &HE) / 2
END
```

การกำหนดค่าเริ่มต้นให้กับพอร์ตอนุกรม

ก่อนการใช้งานพอร์ตอนุกรมนั้นจะต้องมีการกำหนดค่าเริ่มต้นให้กับตัวมันก่อน ซึ่งก็คือการกำหนดจำนวนบิตข้อมูลที่ต้องการส่ง,จำนวนบิตปิดท้าย,ชนิดของพาริตี้ที่ใช้ และบอดเรต

การกำหนดสามารถทำได้หลายวิธี วิธีแรกเป็นการกำหนดจากดอสพร้อมพ์ โดยใช้คำสั่ง MODE ซึ่งมีวิธีการใช้งานดังนี้

MODE COMm : baud = b , parity = p , data = d , stop = s , retry = r

หรือ MODE COMm : b , p , d , s , r

ตัวอย่าง MODE COM1: 9600, n , 8 ,1 จะเป็นการกำหนดให้พอร์ตอนุกรม COM1 มีบอดเรตเท่ากับ 9,600 บิตต่อวินาที ไม่มีการตรวจสอบพาริตี้ รับส่งข้อมูลแบบ 8 บิต และมีบิตปิดท้าย 1 บิต

วิธีที่ 2 เป็นการกำหนดโดยใช้อินเตอร์รัปต์ของดอส ตำแหน่งที่ 14H ซึ่งการใช้งานจะต้องกำหนดค่าต่างๆลงในรีจีสเตอร์ด้วย โดยจะต้องกำหนดให้รีจิสเตอร์ AH = 0 รีจิสเตอร์ DX เก็บค่าพอร์ตอนุกรมที่ต้องการกำหนดค่าเริ่มต้น โดย

DX = 0 จะกำหนดให้กับพอร์ตอนุกรม COM1

DX = 1 จะกำหนดให้กับพอร์ตอนุกรม COM2

DX = 2 จะกำหนดให้กับพอร์ตอนุกรม COM3

DX = 3 จะกำหนดให้กับพอร์ตอนุกรม COM4

รีเจสเตอร์ AL ซึ่งมีขนาด 8 บิตใช้เก็บค่าเริ่มต้นต่างๆ มีรายละเอียดดังนี้

บิต7	บิต6	บิต5	บิต4	บิต3	บิต2	บิต1	บิต0
BD2	BD1	BD0	PAR1	PAR0	STOP	DA1	DA0

BD2,BD1,BD0 ใช้สำหรับกำหนดค่าบอดเรต

"111" บอดเรตเท่ากับ 4,800 บิตต่อวินาที

"110" บอดเรตเท่ากับ 2,400 บิตต่อวินาที

"101" บอดเรตเท่ากับ 1,200 บิตต่อวินาที

"100" บอดเรตเท่ากับ 600 บิตต่อวินาที

"011" บอดเรตเท่ากับ 300 บิตต่อวินาที

"010" บอดเรตเท่ากับ 9,600 บิตต่อวินาที

"001" บอดเรตเท่ากับ 150บิตต่อวินาที

"000" บอดเรตเท่ากับ 110 บิตต่อวินาที

ตัวอย่างโปรแกรมย่อยเทอร์โบปาสคาลสำหรับการกำหนดค่าให้กับพอร์ตอนุกรม โดยกำหนดค่าเริ่มต้นให้กับพอร์ตอนุกรม COM1 มีอัตราบอดท่ากับ 9600 ไม่มีการตรวจสอบพาริตี้ บิตข้อมูล 8 บิต และบิตปิดท้าย 1 บิตเขียนได้ดังนี้

```
procedure initial ;
var regis : registers ;
begin
        with regis do begin
                ah : = 0 ;
                al = $0e3 ;
```

```
            dx : = 0 ;
            intr ( $14, regis ) ;
        end :
end :
```

PAR1,PAR0 ใช้กำหนดค่าพาริตี้โดย

"00" หรือ "10" ไม่มีการตรวจสอบพาริตี้

"01" พาริตี้คี่

"11" พาริตี้คู่

STOP ใช้กำหนดจำนวนของบิตปิดท้าย

"1" มีบิตปิดท้ายเท่ากับ 2 บิต

"0" มีบิตปิดท้ายเท่ากับ 1 บิต

DA1,DA0 ใช้กำหนดความยาวของข้อมูลโดย

"10" ความยาวข้อมูลเท่ากับ 7 บิต

"11" ความยาวข้อมูลเท่ากับ 8 บิต

การรับและการส่งข้อมูลแบบอนุกรม

มีหลากหลายวิธีในการรับส่งข้อมูลแบบอนุกรมผ่านพอร์ตอนุกรม RS-232 เช่น ใช้คำสั่งพิมพ์ออกทางเครื่องพิมพ์ เรียกอินเตอร์รัปต์ของไบออสหรือของดอส การเขียนหรืออ่านไปยังแอดเดรสของพอร์ตโดยตรง วิธีสุดทายเป็นวิธีที่ความยืดหยุ่นในการใช้งานมากที่สุด ยังตัวอย่างถ้าต้องการส่งข้อมูลไปยังพอร์ตอนุกรม COM1 สามารถเขียนข้อมูลโดยตรงไปที่รีจิสเตอร์บัฟเฟอร์ส่งข้อมูล (แอดเดรส 3F8H) โดยใช้คำสั่งภาษา QBASIC ง่ายๆ ดังนี้

```
OUT & H3F8 , X
```

ค่า X ในที่นี้หมายถึงข้อมูลที่ต้องการส่งมี 8 บิต

สำหรับการอ่านข้อมูลจากพอร์ตอนุกรม จะเป็นการอ่านข้อมูลมาจากพอร์ตรีจิสเตอร์บัฟเฟอร์สำหรับรับข้อมูล (แอดเดรส 3F8H เช่นเดียวกัน) ซึ่งสามารถเขียนโปรแกรมง่ายๆดังนี้

```
Y = INP ( &H3F8 )
```

ค่า Y ในที่นี้คือค่าที่อ่านได้จากรีจิสเตอร์บัฟเฟอ์สำหรับรับข้อมูล โดยมีขนาด 8 บิต

สำหรับเขียนโปรแกรมด้วย TURBO PASCALก็สามารถใช้คำสั่ง

```
PORT [$3F8] = x
```

สำหรับการเขียนข้อมูลไปยังพอร์ตอนุกรมและ

```
Y = PORT [$3F8]
```

สำหรับการอ่านข้อมูลจาดกพอร์ตอนุกรม

แต่เมื่อใช้คำสั่งนี้ในขณะที่โปรแกรมทำงานผ่านระบบปฏิบัติการวินโดวส์จะไม่สามารถใช้งานได้เนื่องจากระบบบปฏิบัติการวินโดวส์ ได้เข้าฝังตัวพอร์ตอนุกรมเข้าเป็นส่วนหนึ่งของระบบปฏิบัติการแล้ว ดังนั้นการเรียกใช้งานจึงจำเป็นต้องเรียกผ่านเครื่องมือที่ติดต่อผ่านระบบปฏิบัติการ เช่น การใช้คอนโทรล MSCOMM32.OCX ของโปรแกรม Visual BASIC

- คอนโทรล MSComm

สำหรับการใช้งาน Visual BASIC ตั้งแต่เวอร์ชั่น 2 เป็นต้นมา ใน Visual BASIC จะมีคัสตอมคอนโทรลสำหรับการสื่อสารอนุกรมของคอมพิวเตอร์มาให้โดยใน Visual BASIC เวอร์ชัน 2 และเวอร์ชัน 3 จะใช้ชื่อว่า MSCOMM.VBX ส่วนเวอร์ชัน4 ใช้ชื่อว่า MSCOMM16.OCX สำหรับการทำงานกับระบบปฏิบัติการ 16 บิต และ MSCOMM32.OCX สำหรับการทำงานกับระบบปฏิบัติการ 32 บิต สำหรับใน Visual BASIC เวอร์ชัน 5 จะมีเพียง MSCOMM32.OCX เท่านั้นเพราะถูกออกแบบมาให้ใช้งานกับระบบปฏิบัติการ 32 บิต

MSComm จัดเตรียมทางเลือกเอาไว้ 2 ทางเพื่อความสะดวกในการสื่อสารข้อมูล ทางแรกคือ การสื่อสารข้อมูลที่กระตุ้นด้วยเหตุการณ์ (event-driven communications) เป็นรูปแบบการใช้งานที่มีประสิทธิภาพมากสำหรับการตอบสนองแบบทันทีทันใด เช่น เมื่อตัวอักษรถูกส่งมาที่พอร์ตอนุกรมหรือเกิดการเปลี่ยนแปลงที่ขา Data Carrier (DCD) หรือขา Request To Send(RTS) เหตุการณ์ Oncomm ของ MSComm จะสามารถตรวจจับสัญญาณนั้นได้ทันที ซึ่งจะกล่าวรายละเอียดในหัวข้อคุณสมบัติ CommEvent ต่อไป ส่วนทางเลือกที่สองเป็นการคอยตรวจสอบค่าเหตุการณ์และความผิดพลาดที่เกิดขึ้นด้วยการดูค่าที่เปลี่ยนแปลงภายในคุณสมบัติ CommEvent หลังจากให้โปรแกรมทำงานในเวอร์ชั่นต่างๆ ไปเรียบร้อยแล้ว ซึ่งวิธีนี้ใช้งานได้ดีในกรณีที่โปรแกรมมีขนาดเล็ก

คอนโทรล MSComm 1ตัวสามารถควบคุมการทำงานของพอร์ตอนุกรมได้ 1 พอร์ต ถ้าในโปรแกรมที่ใช้งานต้องการติดต่ออับพอร์ตอนุกรมมากกว่า 1 พอร์ตจะต้องใช้คอนโทรล MSComm มากกว่า 1 ตัวเพื่อควบคุมพอร์ตอนุกรมในแต่ละพอร์ต แอดเดรสของพอร์ตอนุกรมและแอดเดรสของการเกิดอินเตอร์รัปต์สามารถเปลี่ยนได้จากการแก้ไขค่าที่ Control Panel

ถึงแม้ว่าคอนโทรล MSComm จะมีคุณสมบัติ (property) มากมาย แต่สามารถทำความเข้าใจได้ไม่ยากดังนี้

- CommPort

ใช้ในการกำหนดค่าและอ่านพอร์ตอนุกรมที่ติดต่ออยู่ (COM1,COM2,COM3,COM4,)

รูปแบบการใช้งาน

objcct . CommPort [= value]

โดย Value เป็นค่าของพอร์ตอนุกรมชนิดข้อมูลเป็น Integer ค่า Value สามารถกำหนดได้ในช่วง 1-16 (ค่าเริ่มต้นกำหนดไว้ที่ 1) เมื่อมีการกำหนดค่าแล้วจึงทำการเปิดพอร์ตโดยใช้คุณสมบัติ

PortOpen แต่ว่าพอร์ตนั้นไม่มีอยู่ในระบบ MSComm จะสร้างสัญญาณแสดงข้อผิดพลาด error 68 ขึ้นมา ซึ่งหมายถึง อุปกรณ์ตัวนี้ไม่มีอยู่ในระบบ ดังนั้นการเขียนโปรแกรมจึงจำเป็นต้องกำหนดตำแหน่งของพอร์ตอนุกรมก่อนที่จะใช้คำสั่ง OpenPort

- Setting

ใช้ในการกำหนดและอ่านค่าอัตราบอด, พาริตี้, จำนวนของบิตข้อมูล, จำนวนของบิตปิดท้าย

รูปแบบของการใช้งาน

object . Setting [= value]

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

ค่าบอดเรตมาตราฐานที่ใช้กับ MSComm มีดังนี้

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

สำหรับค่ามาตราฐานในการกำหนดค่าพาริตี้มีดังนี้

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

ค่าที่ใช้ในการกำหนดจำนวนบิตมี 5 คือ 4,5,6,7 และ 8 (เป็นค่าปกติ)

ค่าที่ระบุจำนวนบิตปิดท้ายมี 3 ค่าคือ 1 (เป็นค่าปกติ), 1.5 และ 2

ตัวอย่างการใช้งานคำสั่ง Setting โดยจะเป็นการกำหนดค่าบอดเรตเท่ากับ9600 ไม่มีพาริตี้ จำนวนบิตข้อมูล 8 บิต และบิตปิดท้าย 1 บิต สามารถเขียนโปรแกรมได้ดังนี้

MSComml . Setting = "9600 , N , 8 , 1"

หมายเหตุ สาเหตุที่ค่าที่กำหนดจะต้องอยู่ภายใต้เครื่องหมายคำพูด "" เนื่องจากค่าที่กำหนดนี้อยู่ในรูปตัวแปร String

- PortOpen

ใช้ในการกำหนดและอ่านค่าสถานะของพอร์ตอนุกรม เพื่อเปิดและปิดพอร์ตอนุกรม

รูปแบบการใช้งาน

Object . PortOpen [= value]

ค่า Value มีชนิดข้อมูลเป็นแบบบูลีน คือ True กับ False โดย True หมายถึงการเปิดพอร์ตอนุกรมและ False หมายถึงการปิดพอร์ตอนุกรม สำหรับการปิดพอร์ตนั้นจะมีการเคลียร์บัฟเฟอร์รับข้อมูลและบัฟเฟอร์ส่งข้อมูลด้วย คอนโทรล MSComm จะปิดพอร์ตอนุกรมโดยอัตโนมัติเมื่อออกจากโปรแกรม ก่อนที่จะใช้คุณสมบัติ PortOpen ต้องตรวจสอบให้แน่ใจก่อนว่าคุณสมบัติ CommPort นั้นได้ทำการกำหนดตำแหน่งของพอร์ตอนุกรมไว้ถูกต้องหรือไม่ มิเช่นนั้น MSComm จะแสดงข้อผิดพลาด Error 68 แจ้งแก่ผู้ใช้งานหรือถ้าพอร์ตอนุกรมนั้นถูกเปิดเอาไว้แล้ว โปรแกรมก็จะแจ้งข้อผิดพลาดออกมาเช่นกัน

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

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

MSComm1 . Settings = "9600 , n , 8 , 1"

MSComm1 . CommPort = 1

MSComm1 . PortOpen = True

- Input

อ่านค่าและลดค่าขบวนข้อมูลจากบัฟเฟอร์ภาครับ

รูปแบบการใช้งาน

Object . Input

คุณสมบัติ InputLen เป็นตัวกำหนดจำนวนของตัวอักษรที่จะอ่านโดยคุณสมบัติ Input การกำหนดค่าให้ InputLen เท่ากับ 0 เป็นการกำหนดคุณให้สมบัติ Input ทำการอ่านข้อมูลในบัฟเฟอร์รับข้อมูลทั้งหมด

คุณสมบัติ InputMode เป็นตัวกำหนดชนิดของข้อมูลที่คุณสมบัติ Input รับเข้ามา ถ้า InputMode ถูกกำหนดเป็น comInputModeText คุณสมบัติ Input จะส่งข้อมูลกลับมาในรูปแบบของข้อความชนิดข้อมูลเป็นแบบ Variant ถ้า InputMode กำหนดเป็น comInputModeBinary คุณสมบัติ Input จะส่งข้อมูลกลับมาในรูปแบบของไบนารี่และชนิดข้อมูลเป็นแบบ Variant

ตัวอย่างโปรแกรมแสดงให้เห็นถึงวิธีในการรับข้อมูลจากบัฟเฟอร์รับข้อมูลทั้งหมด

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

- InbufferCount

ส่งค่าจำนวนของตัวอักษรที่ในบัฟเฟอร์ภาครับ

รูปแบบการใช้งานคำสั่ง

object. InBufferCount [= value]

คำสั่ง InBufferCount จะแสดงค่าจำนวนของตัวอักษร ซึ่งรับมาจากภายนอกและยังเก็บอยู่ในบัฟเฟอร์ดภาครับ เพื่อให้ผู้ใช้งานอ่านค่าออกไป สำหรับการเคลียร์ค่าบัฟเฟอร์ภาครับทำได้โดยกำหนดให้ InBufferCount มีค่าเป็น ()

หมายเหตุ อย่าสับสันระหว่างคำสั่ง InBufferSize และ InBufferCount คำสั่ง InBufferSize นั้นใช้เพื่อกำหนดขนาดของบัฟเฟอร์ภาครับ

- InBufferSize

กำหนดและคืนค่าขนาดของบัฟเฟอร์ภาครับในหน่วยเป็นไบต์

รูปแบบการใช้งานคำสั่ง

object. InBufferSize [= value]

คำสั่ง InBufferSizc ใช้เพื่อกำหนดขนาดของบัฟเฟอร์ภาครับ โดยค่าเริ่มต้นถูกกำหนดไว้ที่ 1,024 ไบต์

หมายเหตุ การกำหนดค่าบัฟเฟอร์ภาครับขนาดใหญ่จะทำให้หน่วยความจำที่เหลือสำหรับการใช้งานส่วนอื่น ๆ จะเหลือน้อย อย่างไรก็ตามการกำหนดค่า บัฟเฟอร์ภาครับที่น้อยเกินไปจะทำให้เกิดการโอเวอร์โฟลวหรือข้อมูลล้นบัฟเฟอร์ เว้นแต่จะมีการใช้แฮนด์เช็ก ดังนั้นค่าปานกลางที่เหมาะสมก็คือค่า 1,024 ซึ่งเป็นค่าเริ่มต้นนั่นเอง แต่ถ้าโปรแกรมมีการเกิดโอเวอร์โฟลวแล้วจึงค่อยปรับเพิ่มขนาดของบัฟเฟอร์ให้มีค่ามากขึ้น

- InputLen

กำหนดค่าและคืนค่าจำนวนของตัวอักษรที่อ่านจากบัฟเฟอร์ภาครับ

รูปแบบการใช้งานคำสั่ง

object. InputLen [= value]

ค่าเริ่มต้นของคุณสมบัติ OnputLen มีค่าเท่ากับ "0" จะทำให้คำสั่ง Inpuut ของ MSComm อ่านค่าข้อมูลที่อยู่ภายในบัฟเฟอร์ภาครับทั้งหมด

ถ้าไม่มีข้อมูลอยู่ในบัฟเฟอร์ภาตับมากเท่ากับจำนวน InputLen คำสั่ง Input จะส่งค่าว่าง ("") กลับออกมา ผู้ใช้งานสามารถตรวจสอบข้อมูลในบัฟเฟอร์ภาครับได้โดยใช้คุณสมบัติ InBufferCount โดยกำหนดให้มีข้อมูลอยู่ในบัฟเฟอร์ภาครับก่อนแล้วจึงค่อยอ่านข้อมูลจากบัฟเฟอร์ภาครับ

คุณสมบัตินี้มักใช้กับการอ่านค่าข้อมูลจากเครื่องมือหรือเครื่องจักรที่มีการกำหนดค่าขนาดความยาวของข้อมูลเอาไว้แล้ว

ตัวอย่างโปรแกรมการอ่านค่าตัวอักษรออกมา 10 ตัวอักษร

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

- InputMode

กำหนดค่าและคืนค่าชนิดของข้อมูลที่รับโดยคำสั่ง Input

รูปแบบการใช้งานคำสั่ง

object. InputMode [= value]

คุณสมบัติ InputMode ใช้กำหนดว่าข้อมูลชนิดไหนที่รับเข้ามาผ่านคำสั่ง Input โดยข้อมูลจะเลือกได้ 2 ประเภทคือ

comInputMode Text สำหรับข้อมูลที่อยู่ในรูปข้อความตัวอักษรตามมาตรฐาน ANSI โดยจะต้องกำหนดค่าเป็น "0" และค่าเริ่มต้นของการรับค่าข้อมูลก็จะเป็นค่านี้

comInputModeBinary สำหรับข้อมูลอื่น ๆ ซึ่งจะเก็บในรูปไบนารี่รวมกันอยู่เป็นไบต์ข้อมูล

ตัวอย่างการใช้งาน InputMode ต่อไปนี้จะทำการอ่านค่าข้อมูล 10 ไบต์จากพอร์ตอนุกรมและเก็บข้อมูลไว้ในตัวแปรแบบอาร์เรย์ชนิดข้อมูลเป็นแบบไบต์

```
Private Sub Commandl_Click ( )
Dim Buffer as Variant
Dim Arr ( ) as Byte
MSComml. CommPort = 1          ' Set  and open port
MSComml. PortOpen = True
MSComml. InputMode = comInputModeBinary    ' Set InputMode to read binary data
Do Until MSComml. InBufferCount < 10          ' Wait until 10 byte are in the input buffer
        DoEvents
Loop
Buffer = MSComml. Input                       ' Stor binary data in buffer
Arr = Buffer                                  ' Assign to byte array for processing
End Sub
```

- Output

ใช้ในการส่งขบวนของข้อมูลไปบังบัฟเฟอร์ส่งข้อมูล

รูปแบบการใช้งาน

object. Output [= value]

ค่า Value เป็นค่าของตัวอักษรที่เขียนไปยังบัฟเฟอร์ส่งข้อมูล คุณสมบัติ Output สามารถใช้ในการส่งข้อมูลตัวอักษรหรือข้อมูลไบนารี่ก็ได้ โดยการส่งข้อมูลเป็นรูปแบบตัวอักษรจะต้องกำหนดข้อมูลเป็นแบบ Variant และมีข้อมูลภายในเป็นแบบ String สำหรับการส่งข้อมูลไบนารี่จะต้องกำหนดชนิดของข้อมูลเป็น Variant และมีข้อมูลภายในเป็นแบบ Byte

ตัวอย่างโปรแกรมการส่งค่าที่ป้อนจากคีย์บอร์ดไปบังพอร์ตอนุกรมโดยใช้คุณสมบัติ Output

```
Private Sub Form_KeyPress (KeyAscii As Integer)
    Dim Buffer as Variant
    MSComml. CommPort = 1          ' Use COM1
    MSComml. PortOpen = True       ' Open port
    Buffer = Chr $(KeyAscii)
    MSComml. Output = Buffer       ' Send DATA
End Sub
```

- OutBufferCount

คืนค่าจำนวนของข้อมูลตัวอักษรที่เก็บอยู่ในบัฟเฟอร์ภาคส่งและสามารถใช้คำสั่งนี้เพื่อเคลียร์บัฟเฟอร์ภาคส่งได้ด้วย

รูปแบบการใช้งานคำสั่ง

object. OutBufferCount [= value]

ผู้ใช้งานสามารถเคลียร์บัฟเฟอร์ภาคส่งได้โดยการกำหนดค่า OutBufferCount เท่ากับ "0"

หมายเหตุ ระวังการสับสนระหว่างคุณสมบัติ OutbufferCount กับ OutBufferSize ซึ่ง OutBufferSize ใช้เพื่อกำหนดขนาดของบัฟเฟอร์ภาคส่ง

- OutBufferSize

กำหนดค่าและคืนค่าขนาดของบัฟเฟอร์ภาคส่ง ชนิดตัวแปรเป็นแบบไบต์

รูปแบบการใช้งานคำสั่ง

object. OutBufferSize [= object]

คุณสมบัติ OutBufferSize ใช้สำหรับกำหนดขนาดของบัฟเฟอร์ภาคส่ง โดยค่าปกติที่ใช้งานจะมีค่าเท่ากับ 512 ไบต์

หมายเหตุ การกำหนดค่าบัฟเฟอร์ภาคส่งที่มากเกินไปจะทำให้มีหน่วยความจำเหลือให้ใช้งานน้อย แต่อย่างไรก็ตามถ้ากำหนดค่าน้อยเกินไป จะทำให้เกิดข้อมูลล้นบัฟเฟอร์ขึ้นได้ ยกเว้นจะมีการใช้แฮนด์เช็ค วิธีการที่ถูกต้องในการกำหนดค่าคือ ทดลองใช้ค่าเริ่มต้นคือค่า 512 ไบต์ดูก่อนถ้าโปรแกรมทำงานแล้วเกิดการล้นของมูลค่อยเพิ่มค่าของ OutBufferSize ให้มากขึ้น

- ParityReplace

กำหนดและคืนค่าตัวอักษรที่ไปวางแทนในตำแหน่งที่เกิดข้อผิดพลาดจากพาริตี้

รูปแบบการใช้งานคำสั่ง

object. ParityReplace [= value]

บิตพาริตี้ เป็นบิตที่ทางภาคส่งข้อมูลทำการส่งมาพร้อมข้อมูล เพื่อตรวจสอบข้อผิดพลาดของข้อมูล โดยเมื่อมีการใช้บิตพาริตี้ คอนโทรล MSCOM จะทำการบวกบิตทุกบิตที่มีค่าลอจิก "1" ในแต่ละไบต์และทำการตรวจสอบผลลัพธ์ว่าบิตที่อ่านได้นั้นมีจำนวนลอจิก "1" เป็นเลขคู่หรือคี่ และตรงกับค่าที่กำหนดไว้แต่ต้นหรือไม่ ถ้าค่าที่นำมาบวกแล้วมีพาริตี้ไม่ตรงแสดงว่าการรับส่งข้อมูลผิดพลาด

การกำหนดค่า เริ่มต้นให้กับ ParityReplace นั้นกำหนดให้ใช้เครื่องหมาย (?)ไปวางไว้ที่ตำแหน่งที่เกิดพาริตี้ ถ้ากำหนดค่า ParityReplace ให้เป็นค่าว่าง ("") จะเป็นการยกเลิกการใช้งาน ParityReplace และไม่มีการป้อนข้อมูลแทนเมื่อตรวจพบข้อมูลแทนเมื่อตรวจพบข้อผิดพลาด

ParityReplace ใช้ชนิดข้อมูลเป็นแบบสตริง แต่จะการกำหนด จะกำหนดได้เพียงไบต์เดียวเท่านั้น ซึ่งจะสามารถใช้ค่าใด ๆ ก็ได้ที่เป็นโค้ด ANSI มีค่าอยู่ระหว่าง 0 – 255

- DTREnable

ใช้ในการกำหนดสถานะลอจิกของขา Data Terminal Ready (DTR) โดยสัญญาณของขา DTR จะส่งจากคอมพิวเตอร์ไปยังโมเด็มเพื่อแสดงว่าคอมพิวเตอร์พร้อมที่จะรับข้อมูลแล้ว ชนิดของข้อมูลเป็นแบบบูลีน

รูปแบบการใช้งาน

object. DTREnable [= value]

ค่า Value เป็นค่าสถานะ True หรือ False เพื่อกำหนดลอจิกของเขา DTR ให้เป็น "0" หรือ "1" โดย

True หมายถึง ให้เขา DTR มีลอจิก "1"

False หมายถึง ให้เขา DTR มีลอจิก "0" (เป็นค่าปกติ)

หมายเหตุ เมื่อเขา DTR ถูกกำหนดสถานะให้เป็น True ที่เขา DTR จะมีสถานะลอจิก "1" เมื่อทำการเปิดพอร์ตและจะมีสถานะเป็น "0" เมื่อมีการปิดพอร์ต เมื่อเขา DTR ถูกกำหนดสถานะเป็น False ที่เขา DTR จะมีสถานะ ลอจิก "0" ตลอดเวลาไม่ว่าจะใช้คำสั่งเปิดพอร์ตหรือปิดพอร์ต

สำหรับการใช้งานกับโมเด็ม การทำให้เขา DTR เป็นลอจิก "0" จะเป็นการวางหูโทรศัพท์หรือยกเลิกการติดต่อ

- RTSEnable

ใช้เพื่อกำหนดสถานะลอจิกให้ขา Request To Send (RTS) โดยขา RTS จะเป็นสัญญาณที่ส่งจากคอมพิวเตอร์ไปยังโมเด็มเพื่อร้องขอส่งข้อมูล ชนิดของข้อมูลเป็นแบบ Boolean

รูปแบบการใช้งาน

object. RTSEnable [= value]

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

True หมายถึง ให้ขา RTS มีลอจิก "1"

False หมายถึง ให้ขา RTS มีลอจิก "0" (เป็นค่าปกติ)

หมายเหตุ เมื่อขา RTSEnable ถูกกำหนดให้เป็น True ขา RTS จะมีสถานะลอจิก "1" เมื่อเปิดพอร์ตและมีสถานะลอจิก "0" เมื่อปิดพอร์ต

- EOFEnable

เป็นการกำหนดให้ MSComm รอสัญลักษณ์แสดงส่วนท้ายสุดของไฟล์ (End of file : EOF) ระหว่างการรับอินพุตเข้ามา ถ้าพบสัญลักษณ์ EOF ภาคอินพุตจะหยุดรับข้อมูล และเหตุการณ์ OnComm จะถูกกระตุ้นให้ทำงาน คุณสมบัติ CommEvent จะมีค่าเท่ากับ 7 หรือ ComEvEOF

รูปแบบการใช้งาน

object. EOFEnable [= value]

โดย Value เป็นค่าสถานะ True หรือ False เพื่ออีนาเบิลหรือดิสเอบิลการทำงานของเหตุการณ์ OnComm เมื่อตรวจพบสัญลักษณ์ EOF โดย

True หมายถึง เหตุการณ์ OnComm จะถูกกระตุ้นให้ทำงานด้วย EOF

False หมายถึง เหตุการณ์ OnComm จะถูกกระตุ้นให้ทำงานด้วย EOF (เป็นค่าปกติ)

เมื่อ EOFEnable กำหนดให้เป็น False ส่วนควบคุมจะไม่มีการตรวจสอบสัญลักษณ์ EOF

- CTSHolding

ผู้ใช้งานสามารถตรวจสอบการทำงานของขา Clear To Send (CTS) ได้ว่ามีมีสถานะลอจิก "0" หรือ "1" โดยค่าที่อ่านได้จะเป็นบูลีน True และ False ถ้าค่า CTSHolding เป็น True ขา CTS จะมีสถานะลอจิกเป็น "1" ถ้าค่า CTSHolding เป็น False ขา CTS จะมีสถานะลอจิกเป็น "0"

รูปแบบการใช้งาน

object. CTSHolding

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

- CDHolding

ผู้ใช้งานสามารถตรวจสอบการทำงานของขา Data Carrier Detect (DCD) ได้ว่ามีสถานะลอจิกเป็น "1" หรือ "0" โดยค่าที่อ่านได้จะเป็นบูลีน True และ False ถ้าค่า CDHolding เป็น True ขา DCD จะมีสถานะลอจิก "1" ถ้าค่า CDHolding เป็น False ขา DCD จะมีสถานะลอจิก "0"

รูปแบบการใช้งาน

object. CDHolding

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

- DSRHolding

ผู้ใช้งานสามารถตรวจสอบการทำงานของขา DSR (DSR) ได้ว่ามีสถานะลอจิก "1" หรือ "0" โดยค่าที่อ่านไจะเป็นบูลีน True และ False ถ้าค่า DSRHolding เป็น Treu ขา DSR จะมีสถานะลอจิก "1" ถ้า DSRHolding เป็น False ขา DSR จะมีสถานะลอจิก "0"

รูปแบบการใช้งาน

object. DSRHolding

เมื่อขา DSR เป็นลอจิก "1" (SDRHolding = Truc) และเกิดไทม์เอาต์ คอนโทรล MSComm จะกำหนดให้คุณสมบัติ CommEvent มีค่าเป็น comEventDSRTO (Data Set Ready Timeout) และกระตุ้นให้เกิดเหตุกษณ์ OnComm

- Handshaking

กำหนดคุณสมบัติและคืนค่ารูปแบบแฮนด์เช็กทางฮาร์ดแวร์

รูปแบบการใช้งานคำสั่ง

object. Handshaking [= value]

ค่าตัวแปร Value ที่ใช้กำหนดค่ากำหนดได้ 4 รูปแบบด้วยกันคือ

1. comNone ค่าที่กำหนดคือ 0 เป็นการกำหนดให้ไม่มีการแฮนด์เช็ก (เป็นค่าเริ่มต้น)
2. comXOnXOff ค่าที่กำหนดคือ 1 เป็นการกกำหนดให้ใช้แฮนด์เช็กแบบ XON/XOFF
3. comRTS ค่าที่กำหนดคือ 2 เป็นการกำหนดให้ใช้ขา RTS/CTS (Rtquest To Send / Clear To Send)
4. comRTSXOnOff ค่าที่กำหนดคือ 3 เป็นการกำหนดใช้ใช้ทั้งแบบ Request To Send และ XON/XOFF

คุณสมบัติ Handshaking ใช้เพื่อกำหนดรูปแบบการสื่อสารภายในระหว่างที่ข้อมูลถูกส่งไปยังบัฟเฟอร์ภาครับ เมื่อข้อมูลตัวอักษรถูกส่งมาถึงพอร์ตอนุกรม อุปกรณ์สื่อสารข้อมูลจะทำการย้ายข้อมูลไปยังบัฟเฟอร์ภาครับ เพื่อที่จะให้โปรแกรมสามารถอ่านค่าไปใช้งานได้ ถ้าไม่มีบัฟเฟอร์ภาครับ โปรแกรมที่ใช้งานจะต้องทำการอ่านค่าข้อมูลโดยตรงจากฮาร์ดแวร์ของพอร์ตอนุกรม ซึ่งผู้ใช้งานจะเกิดปัญหาข้อมูลสูญหายได้ เนื่องจากว่าการเปลี่ยนแปลงของข้อมูลที่ส่งเข้ามามีการเปลี่ยนแปลงอย่างรวดเร็ว

คุณสมบัติ handshaking ช่วยให้ผู้ใช้งานแน่ใจได้ว่าข้อมูลที่ได้รับมานั้นไม่มีการสูญหายเมื่อบัฟเฟอร์ภาครับที่รับข้อมูลนั้นเกิดข้อมูลล้นหรือโอเวอร์โฟลว (overflow) โดยใช้วิธีการตรวจสอบความพร้อมของบัฟเฟอร์ว่าพร้อมรับข้อมูลหรือไม่ก่อนที่จะส่งข้อมูลมาให้

- Break

ใช้ในการเซตและเคลียร์ค่าสัญญาณ Break ชนิดของข้อมูลเป็นแบบ Boolean

รูปแบบการใช้งาน

object. Break [= value]

โดย Value เป็นค่าบูลีน ถ้า Value = True หมายถึง การส่งสัญญาณ Break ออกไป ถ้า Value = False หมายถึงการเคลียร์สัญญาณ Break

เมื่อกำหนดให้สัญญาณ Break เป็น True จะเป็นการหยุดส่งข้อมูลชั่วคราวจนกว่าจะมีการสั่งให้สัญญาณ Break เป็น Falsc

ตัวอย่าง เป็นวิธีการส่งสัญญาณ Break ออกไปเป็นช่วงเวลาสั้น ๆ ที่ 1/10 ของวินาที

```
MSComml. Break = True          ' Set the Break condition.
Duration! = Timer + .1         ' Set duration to 1/10 second.
Do Until Timer > Duiration!    ' Wait for the duration to pass.
Dummy = DoEvents ( )
Loop
MSComml. Break = False         ' Clear the Break condition.
```

เหตุการณ์ OnComm

เหตุการณ์ OnComm จะถูกสร้างขึ้นเมื่อค่าของคุณสมบัติ CommEvent มีการเปลี่ยนแปลง เพื่อแสดงผลการเปลี่ยนแปลงเหล่านั้นแบบทันทีทันใดหรือแสดงข้อผิดพลาดที่เกิดขึ้นตัวอย่างโปรแกรมย่อย OnComm เพื่อนำเหตุการณ์ CommEvent มาแสดง

```
Private Sub MSComm OnComm ( )
        Select Case MSComml. CommEvent
        ' Handle each event or error by placing
        ' code below each case statement
        ' Errors
                Case comEventBreak          ' A Break was received.
                Case comEventCDTO           ' CD (RLSD) Timeout.
                Case comEventCTSTO          ' CTS Timeout.
                Case comFDTYp               ' DSR Timeout.
                Case comEventFrame          ' Framing Error
                Case comEvenOverrum         ' Data Lost.
                Case comEventRxOver         ' Receive buffer overflow.
                Case comEventRxParity       ' Parity Error.
Xase comEventTxFull
        ' Events
                Case comEvCD                ' Change in the CD line.
                Case comEvCTS               ' Change in the CTS line.
                Case comEvDSR               ' change in the DSR line.
                Case comEvRing              ' Change in the Ring Indicator.
                Case comEvReceive           ' Received Rthreshold # of chars.
                Case comEvSend              ' Sthreshold number in the 'transmit buffer.
Case comEvEof                               ' An Eof charater was found in the input
stream
        End Select
End Sub
```

ค่าคงที่คุณสมบัติของคอนโทรล MSComm

ค่าคงที่สำหรับคุณสมบัติ Handshake

ค่าคงที่	ค่า	รายละเอียด
comNone	0	ไม่ใช้การตรวจสอบแฮนด์เชก
comXonXoff	1	ใช้การตรวจสอบแฮนด์เชกแบบ Xon/Xoff
comRTS	2	ใช้การตรวจสอบแฮนด์เชกผ่านทางขา RTS และ CTS
comRTSXOnXOff	3	กำหนดการตรวจสอบแฮนด์เชกทั้งแบบ RTS, CTS XOn/Xoff

ค่าคงที่สำหรับคุณสมบัติ OnComm

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

ค่าคงที่สำหรับคุณสมบัติ Error

ค่าคงที่	ค่า	รายละเอียด
comEventBreak	1001	ได้รับสัญญาณ Break
comEventCTSTO	1002	ขา CTS เกิดไทม์เอาต์
comEventDSRTO	1003	ขา DSR เกิดไทม์เอาต์
comEventFrame	1004	เกิดข้อผิดพลาดที่เฟรมข้อมูล (Fdraming error)
comEventOverrun	1006	พอร์ตอนุกรมเกิดโอเวอร์รัน (Port overrun)
comEventRxOver	1007	ขา DCD เกิดไทม์เอาต์
comEventRxOver	1008	บัฟเฟอร์รับข้อมูลเกิดโอเวอร์โฟลว
comEventRxParity	1009	เกิดข้อผิดพลาดที่พาริตี้ (Parity error)
comEventFull	1010	บัฟเฟอร์ส่งข้อมูลเต็ม

ค่าคงที่สำหรับคุณสมบัติ InputMode

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

การใช้ MSComm เพื่อการติดต่อฮาร์ดแวร์

จากรายละเอียดของ MSComm ที่กล่าวไปในตอนต้นนั้น จะเห็นได้ว่าวิธีการที่จะอ่านค่าหรือเขียนค่าไปยังขาสถานะและขาควบคุมของพอร์ตอนุกรมสามารถทำได้ง่ายดายมากโดยใช้คำสั่งเหล่านี้

DTREnable สำหรับสั่งให้ขา DTR มีลอจิก "0" หรือ "1"

RTSEnable สำหรับสั่งให้ขา RTS มีลอจิก "0" หรือ "1"

CTSHolding สำหรับอ่านค่าสถานะจากขา CTS ว่ามีลอจิก "0" หรือ "1"

CDHolding สำหรับอ่านค่าสถานะจากขา DCD ว่ามีลอจิก "0" หรือ "1"

DSRHolding สำหรับอ่านค่าสถานะจากขา DSR ว่ามีลอจิก "0" หรือ "1"

Break สำหรับการสั่งให้ขา Txd มีลอจิก "0" หรือ "1"

2.2.2 การส่งและอ่านข้อมูลผ่านพอร์ตอนุกรม RS – 232

พอร์ตอนุกรมนอกจากจะมีขาสำหรบการรับและส่งข้อมูลปกติแล้วยังมีขาที่ออกแบบไว้สำหรับควบคุมการไหลของข้อมูลอีกหลายตำแหน่งด้วยกัน โดยแยกเป็น 2 ประเภทคือขาที่ทำหน้าที่เป็นเอาต์พุต และขาที่ทำหน้าที่เป็นอินพุต

ขาสัญญาณเอาต์พุตของพอร์ตอนุกรม

ขาที่ทำหน้าที่เป็นขาเอาต์พุต ได้แก่ ขา DTR, RTS และ TxD โดยรีจิสเตอร์ที่ทำหน้าที่ควบคุมขาเหล่านี้คือรีจิสเตอร์ควบคุมโมเด็ม (MCR) โดยมีแอดเดรสอยู่ถัดจากรีจิสเตอร์หลักของพอร์ตอนุกรม 4 ตำแหน่ง รีจิสเตอร์ควบคุมการทำงานของ DTR จะอยู่ที่ตำแหน่งบิต 0 ส่วนขา RTS จะอยู่ที่ตำแหน่งบิต1ส่วนขาTxDเป็นขาปกติในการส่งข้อมูลดังนั้นจึงมีแอดเดรสอยู่ที่แอดเดรสของรีจิสเตอร์หลัก

ระดับแรงดันที่ใช้งานสำหรับพอร์ตอนุกรม RS – 232

มาตรฐานการสื่อสารข้อมูลของพอร์ตอนุกรม ได้ระบุช่วงระดับแรงดันสำหรับการทำงานของพอร์ตอนุกรมไว้ว่า ที่ลอจิก "0" จะมีระดับสัญญาณ +3 ถึง +15V ส่วนลอจิก "1" จะมีระดับสัญญาณ –3 ถึง –15V ระดับสัญญาณนี้ทำให้ไม่สามารถที่จะนำขาเอาต์พุตใด ๆ ต่อเข้ากับลอจิกเกตเพื่อใช้งานได้โดยตรง จะต้องผ่านวงจรเพื่อเปลี่ยนระดับแรงดันเสียก่อน โดยปกติจะใช้ไอซีจำพวก RS – 232 transceiver ที่นิยมมากคือ MAX 232 หรือ ICL 232 ไอซีในกลุ่มนี้จะทำหน้าที่แปลงระดับแรงดันของ RS – 232 ให้อยู่ในระดับทีทีแอล โดยลอจิก "0" ซึ่งเดิมมีระดับสัญญาณ +3 ถึง

+15V จะแปลงเป็น 0V ส่วนลอจิก "1" ซึ่งมีระดับสัญญาณ +3 ถึง –15V จะแปลงเป็น +5V ทั้งนี้เพื่อให้สามารถเชื่อมต่อกับอุปกรณ์ดิจิตอลอื่นที่ใช้ระดับแรงดันทีทีแอลได้

การเขียนซอฟต์แวร์เพื่อควบคุมขาเอาต์พุต

การติดต่อกับพอร์ตอนุกรมบนระดบบปฏิบัติการวินโดวส์จะต้องเพิ่มอุปกรณ์ทางซอฟต์แวร์สำหรับการติดต่อกับพอร์ตอนุกรม นั่นคือ MSComm ซึ่งกล่าวรายละเอียดไว้แล้ว

ผู้ใช้งานสามารถเขียนโปรแกรมด้วย Visual BASIC เพื่อส่งค่าออกไปยังขาเอาต์พุตต่าง ๆ ของพอร์ตอนุกรมได้ โดยใช้คำสั่งดังนี้

MSComml. DTREnable = True	สำหรับการกำหนดให้ขา DTR มีลอจิก "1"
MSComml. DTREnable = False	สำหรับการกำหนดให้ขา DTR มีลอจิก "0"
MSComml. RTSEnable = True	สำหรับการกำหนดให้ขา RTS มีลอจิก "1"
MSComml. RTSEnable = False	สำหรับการกำหนดให้ขา RTS มีลอจิก "0"
MSComml. Break = True	' สำหรับการกำหนดให้ขา TxD มีลอจิก "1"
MSComml. Break = True	' สำหรับการกำหนดให้ขา TxD มีลอจิก "0"

หมายเหตุ ก่อนที่ใช้งานคำสั่งของคอนโทรล MSComm จะต้องทำการเปิดพอร์ตก่อนโดยเขียนโปรแกรมดังนี้

```
Private Sub Form_Load ( )
MSComml. PortyOpen = True
End Sub
```

4400596
TX
945
M2375
2544 C.1

พร้อมกันนั้นจะต้องตรวจสอบพอร์ตที่ใช้งานให้ดีว่าพอร์ตอนุกรมที่ใช้นั้นถูกต้องหรือไม่ มิฉะนั้นโปรแกรมจะแสดงข้อผิดพลาดขึ้นมา

การอ่านค่าลอจิกจากพอร์ตอนุกรม RS – 232

พอร์ตอนุกรมมีขาที่ทำหน้าที่อินพุตได้แก่ DSR, CTS, RT และ DCD โดยรีจิสเตอร์ที่ทำหน้าควบคุมขาเหล่านี้คือรีจิสเตอร์แสดงสถานะโมเด็ม (MSR) มีแอดเดรสถัดจากรีจิสเตอร์หลักของพอร์ตอนุกรม 6 ตำแหน่ง สำหรับบิตต่าง ๆ บนรีจิสเตอร์มีรายละเอียดดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
DCD	RI	DSR	CTS	DDCD	DRI	DDSR	DCTS

4 บิตบนของรีจิสเตอร์จะแสดงสถานะการเปลี่ยนแปลงที่ขาอินพุตทั้ง 4 ขาของพอร์ตอนุกรมโดยตรง ส่วน 4 บิตล่าง จะมีสถานะเป็น "1" ก็ต่อเมื่อ 4 บิตบนมีการเปลี่ยนแปลงสถานะเมื่อเทียบกับการอ่านค่าครั้งก่อนหน้า

สำหรับการอ่านค่าจากขา DCD, CTS และ DSR โดยโปรแกรม Visual BASIC จะใช้ MSComm ร่วมกับคำสั่ง CDHol;ding, CTSHolding และ DSRHolding ตามลำดับ ซึ่งค่าที่อ่านได้นั้นจะเป็นบูลีน มีค่าเป็น True หรือ False ผู้ใช้งานสามารถตรวจสอบผลจากขาอินพุตเหล่านี้ได้ โดยใช้คำสั่ง IF THEN

เนื่องจากสัญญาณของ RS – 232 ต้องมีระดับแรงดัน ±3 ถึง ±12V แต่สัญญาณอินพุตที่เกิดขึ้นเป็นระดับทีทีแอล ดังนั้นเมื่อเกิดสัญญาณอินพุตขึ้น จะต้องส่งผ่านวงจรขับเพื่อปรับระดับแรงดันให้เหมาะสมเสียก่อน

2.3 ความรู้ทั่วไปเกี่ยวกับ LCD

ในโมดูล LCD แบบแสดงตัวอักษร มีส่วนประกอบหลักๆ 3 ส่วนดังนี้

ตัวแสดงผล ภายในเป็นผลึกเหลวที่สามรถแสดงผลให้เห็นโดยอาศัยแสงจากภายนอก ดังนั้นจึงต้องมีมุมในการมองข้อมูลที่แสดงผลบนจอ LCD

ตัวควบคุม (controller) เป็นตัวรับข้อมูลจากอุปกรณ์ภายนอกมาควบคุมการทำงานของโมดูล LCD เช่น ลบจอภาพ แสดงตัวอักษร หรือเลื่อนเคอร์เซอร์ เป็นต้น ตัวควบคุมนี้ใช้ชิปควบคุมโดยเฉพาะ ชิปที่นิยมใช้คือ เบอร์ HD44780 และ HD61830 โดย HD44780 จะใช้ควบคุม LCD แบบอักขระ ส่วน HD61830 ใช้ควบคุม LCD แบบกราฟิก

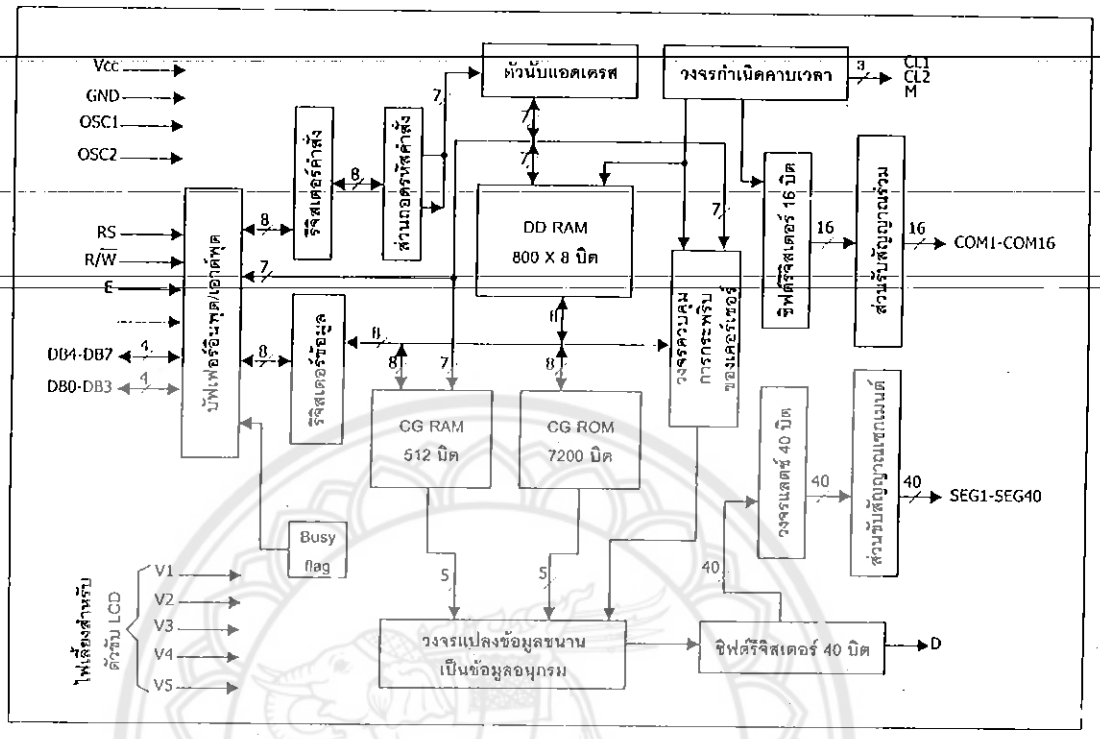
ตัวขับ (driver) เป็นตัวรับสัญญาณจากตัวควบคุมมาขับให้ตัวแสดงผลแสดงข้อมูลตามที่กำหนด ชิปที่ใช้หน้าที่เป็นตัวขับนี้ได้แก่ เบอร์ HD44100H และ MSM5259 เป็นต้น

โครงสร้างภายในของตัวควบคุมโมดูล LCD

ในการใช้ งานโมดูล LCD จำเป็นต้องทำความเข้าใจเกี่ยวกับโครงสร้างและคำสั่งที่ใช้ในการควบคุมให้ดีเสียก่อน ในที่นี้ขอยกตัวอย่างโมดูล LCD แบบอักขระ เพราะสามารถเข้าใจได้ง่าย รูปที่6-1 เป็นบล็อกไดอะแกรมภายในของชิปควบคุม LCD เบอร์ HD 44780 ซึ่งประกอบด้วย

บัฟเฟอร์อินพุตเอาต์พุต เป็นส่วนที่ใช้ในการติดต่อรับส่งข้อมูลกับอุปกรณ์ภายนอก เพื่อที่จะถ่ายทอดข้อมูลเข้าออกภายในตัวควบคุม

รีจิสเตอร์คำสั่ง (Instruction Rgister : IR)เป็นรีจิสเตอร์ใช้รับข้อมูลคำสั่งจากอุปกรณ์ภายนอกเพื่อนำไปควบคุมการแสดงผล



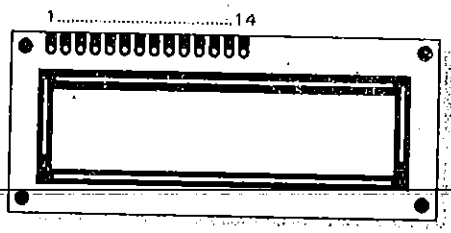
รูปที่ 2.9 ไดอะแกรมการทำงานภายในของโมดูล LCD แบบตัวอักษร

รีจิสเตอร์ข้อมูล (DATA Register : DR) เป็นรีจิสเตอร์ใช้รับข้อมูลจากอุปกรณ์ภายนอกเพื่อถ่ายทอดต่อไปยังแรมเก็บข้อมูลแสดงผล หรือนำข้อมูลนั้นไปสร้างตัวอักษรเพิ่มเติมในแรมเก็บตัวอักษร

แรมเก็บข้อมูลแสดงผล (Display Data RAM:DDRAM) เป็นหน่วยความจำแรมทำหน้าที่เก็บข้อมูล-ที่มาจากรีจิสเตอร์ DR ตัวควบคุมจะนำข้อมูลใน DDRAM นี้ไปเปิดเป็นตาราง (Look up-lable)ของตัวอักษรที่เก็บไว้ในหน่วยความจำรอมและแรมเก็บตัวอักษร เพื่อนำไปแสดงที่ตัวแสดงผล

รอมเก็บตัวอักษร (Character Generator ROM: CGROM) เป็นหน่วยความจำรอมที่ใช้เก็บข้อมูลตัวอักษรหรือสัญลักษณ์ที่สามารถอ่านไปแสดงที่ตัวแสดงผลได้ มีขนาด 7,200 บิตโดยจะถูกอ่านด้วยค่าของข้อมูลใน DDRAM

แรมเก็บตัวอักษร (Character Generator RAM : CGRAM) เป็นหน่วยความจำแรมที่ใช้เก็บคตัวอักษรที่มีการสร้างเพิ่มเติมขึ้นใหม่ ในกรณีที่ตัวอักษรใน CGROM ไม่เพียงพอ มีขนาด 512 บิต การเขียนและอ่านค่าไปใช้นั้นทำได้เช่าเดียวกับ CGROM คือเขียนข้อมูลลงไปใน DDRAM แล้วตัวควบคุมจะอ่านค่าจาก CGRAM เอง



ขา 1 : GND
ขา 2 : +V
ขา 3 : Brightness ปรับความสว่าง
ขา 4 : R/$\overline{W}$
ขา 5 : RS
ขา 6 : E
ขา 7-14 : D0-D7

RS	R/$\overline{W}$	E	การทำงาน
0	0	↓	เขียนคำสั่ง
0	1	⊓	อ่านสถานะของ LCD
1	0	↓	เขียนข้อมูล
1	0	⊓	อ่านข้อมูล

รูปที่ 2.10 ความสัมพันธ์ของขา RS , R/W และ E เพื่อการอ่านและเขียนข้อมูลกับโมดูล LCD

รูปที่ 2.11 การจัดขาของโมดูลLCD

แฟลก BUSY เป็นส่วนที่ทำหน้าที่แจ้งสถานะการทำงานของตัวควบคุมให้อุปกรณ์ภายนอกทราบว่า ตัวควบคุมพร้อมที่จะรับคำสั่งข้อมูลนั้นหรือไม่ ดังนั้นก่อนการส่งข้อมูลหรือคำสั่งมายังตัวควบคุมต้องตรวจสอบสถานะของแฟลก BUSY นี้เสียก่อน

โมดูล LCD ขนาด 16 ตัวอักษร 1 บรรทัด

สำหรับโมดูล LCD ที่ใช้ในการทดลองนี้เป็นขนาด 16 ตัวอักษร 1 บรรทัด มีขาต่อใช้งานทั้งสิ้น 14 ขา มีการจัดขาดังในรูปที่ 6-2 สำหรับรายละเอียดการทำงานของแต่ละขามีดังนี้

VSS (ขา 1) : ต่อกราวด์

VDD (ขา 2) : ต่อไฟเลี้ยง + 5

VO (ขา 3) : เป็นขาอินพุตสำหรับป้อนแรงดัน เพื่อปรับความเข้มของการแสดงผล

RS (ขา 4) :เป็นขาอินพุตใช้เลือกว่าข้อมูลที่ทำการส่งในขณะนั้นเป็นคำสั่งสำหรับรีจิสเตอร์หรือเป็นข้อมูลสำหรับรีจิสเตอร์ DR โดยถ้าขานี่เป็น 0 ข้อมูลที่ส่งมาจะเป็นคำสั่ง แต่ถ้าเป็น 1 ข้อมูลที่ส่งมาจะเป็นข้อมูลสำหรับแสดงผล

R/$\overline{W}$ (ขา 5) เป็นขาที่ใช้เลือกว่าจะอ่านหรือเขียนข้อมูลกับ LCD ถ้าเป็น 0 จะเป็นการเขียนข้อมูล แต่ถ้าเป็น 1 จะเป็นการอ่านข้อมูล

E (ขา 6) เป็นขาอินาเบิล LCD ให้ทำงาน

DBO-DB7 (ขา 7-14) : เป็นขาที่ใช้เป็นทางผ่านของข้อมูลระหว่าง LCD กับอุปกรณ์ภายนอกขนาด 8 บิต

อนึ่งขา RS, R/$\overline{W}$ และ E จะใช้งานร่วมกันโดยมีลักษณะความสัมพันธ์ดังในตารางที่ 6-1

คำสั่งควบคุมโมดูล LCD

ในการเขียนคำสั่งลงในตัวควบคุม แน่นอนว่าจะต้องกำหนดให้ขา RS และ $\overline{R/W}$ เป็น "0" ทั้งคู่ แล้วเขียนข้อมูลคำสั่งตามไป คำสั่งควบคุมโมดูล LCD ของชิปควบคุม HD44780 ที่สำคัญมี 9 คำสั่งดังต่อไปนี้

1. คำสั่งเคลียร์ตัวแสดงผล (clear display) มีข้อมูลคำสั่งเป็น ***01H*** เป็นคำสั่งที่ใช้เขียนข้อมูลช่องว่าง หรือ space เข้าไปใน DDRAM ทั้งหมดเมื่อตัวควบคุมเอ็กซิคิวต์คำสั่งนี้จะทำการกำหนดแอดเดรสของ DDRAM เป็น 0 เคอร์เซอร์จะกลับไปอยู่ที่ตำแหน่งซ้ายมือสุดของจอแสดงผล แล้วเซตบิต I/D (ซึ่งจะกล่าวถึงภายหลัง) ให้เป็น "1"

2. คำสั่ง return home ต้องกำหนดให้บิต 1 ของข้อมูลเป็น "1" เป็นคำสั่งให้เคอร์เซอร์เคลื่อนที่ไปอยู่ที่ตำแหน่งซ้ายสุดของจอแสดงผล แต่ข้อมูลบนจอแสดงผลไม่เปลี่ยนแปลง นั่นคือคำสั่งข้อมูลของคำสั่งนี้จะเป็น ***02H*** หรือ ***03H*** ก็ได้

3. คำสั่งเลือกโหมดการป้อนข้อมูล (Entry mode Set) คำสั่งนี้มีรูปแบบดังนี้

บิต7	บิต6	บิต5	บิต4	บิต3	บิต 2	บิต1	บิต0
0	0	0	0	0	0	0	0

บิต S เป็นบิตที่ใช้ในการกำหนดลักษณะของการแสดงผล เมื่อมีการป้อนข้อมูล ถ้าหากบิตเป็น "1" เมื่อเกิดข้อมูลใหม่บนจอแสดงผล ตัวเคอร์เซอร์จะอยู่กับที่ แต่ตัวอักษรข้อมูลเดิมจะถูกดันไปทางซ้าย แต่ถ้าหากบิตนี้เป็น "0" เมื่อมีการเกิดข้อมูลใหม่ตัวเคอร์เซอร์จะเลื่อนไปทางขวามือ

บิต I/D เป็นบิตที่ใช้ในการกำหนดว่า เมื่อเขียนหรืออ่านข้อมูลแล้ว ทำให้แอดเดรสของ DDRAM เพิ่มขึ้นหรือลดลงหนึ่งแอดเดรส โดยถ้าบิตนี้เป็น "1" แอดเดรสของ DDRAM จะเพิ่มขึ้น แต่ถ้าเป็น "0" แอดเดรสจะลดลง

ดังนั้น ข้อมูลคำสั่งที่เกิดขึ้นสำหรับคำสั่งนี่ได้แก่ ***04H–07H*** (4 ข้อมูลคำสั่ง) และที่ใช้บ่อยคือ ***06H*** หมายถึงเมื่อกำหนดให้เมื่อเกิดข้อมูลใหม่ เคอร์เซอร์จะเลื่อนไปทางขวามือ และแอดเดรสของ DDRAM เพิ่มขึ้น

4.คำสั่งควบคุมการแสดงผล มีรูปแบบคำสั่งดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
0	0	0	0	1	D	C	B

บิต D ใช้ควบคุมการเปิดปิดจอแสดงผล ถ้าบิตนี้เป็น "1" จะเป็นการเปิดจอแสดงผล ถ้าเป็น "0" จะเป็นการปิดจอแสดงผล

บิต C ใช้ควบคุมการแสดงจอเคอร์เซอร์บนจอแสดงผล ถ้าต้องการให้มีเคอร์เซอร์แสดงผลบนจอแสดงผลต้องกำหนดให้บิตนี้เป็น "1" ถ้ากำหนดให้เป็น "0" จะเป็นการปิดเคอร์เซอร์หรือไม่แสดงเคอร์เซอร์

บิต Bใช้ควบคุมการกระพริบของเคอร์เซอร์ ถ้าบิตนี้เป็น "1" เคอร์เซอร์จะกระพริบ

ดังนั้นจะมีข้อมูลคำสั่งได้ตั้งแต่ ***08H - OFH*** (8รูปแบบคำสั่ง) ที่ใช้บ่อยคือ 0CH เป็นการสั่งให้เปิดจอแสดงผล แต่ไม่แสดงเคอร์เซอร์ และ เป็นการสั่งให้เปิดจอแสดงผลแสดงเคอร์เซอร์และสั่งให้เคอร์เซอร์กระพริบ

5. คำสั่งควบคุมการเลื่อนเคอร์เซอร์และข้อมูลตัวอักษร มีรูปแบบข้อมูลคำสั่งดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	**บิต 0**
0	0	0	0	S/L	R/L	*	*

การควบคุมการเลื่อนเคอร์เซอร์แลตัวอักษรบนจอแสดงผล ขึ้นอยู่กับการกำหนดบิต และซึ่งสามารถสรุปได้ดังนี้

บิต S/C	**บิต R/L**	ลักษณะการเลื่อน	**ข้อมูลคำสั่ง**
0	0	เลื่อนเคอร์เซอร์จากตำแหน่งเดิมไปทางซ้าย 1 ตำแหน่ง	10H – 13H
0	1	เลื่อนเคอร์เซอร์จากตำแหน่งเดิมไปทางขวา 1 ตำแหน่ง	14H – 17H
1	0	เลื่อนตัวอักษรที่เกิดขึ้นใหม่ไปทางซ้าย	18H – 1BH
1	1	เลื่อนตัวอักษรที่เกิดขึ้นใหม่ไปทางขวา	1CH – 1 FH

6. คำสั่งกำหนดฟังก์ชั่นการทำงาน มีรูปแบบคำสั่ง ดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
0	0	1	**DL**	**N**	**F**	*	*

บิต DL ใช้กำหนดจำนวนบิตที่ใช้ติดต่อส่งผ่านข้อมูล ถ้าบิตนี้เป็น "0" จะเป็นการติดต่อแบบ 4 บิต แต่ถ้าเป็น "1" จะเป็นแบบ 8 บิต

บิต N ใช้กำหนดจำนวนบรรทัดของการแสดงผล ถ้าเป็น "0 " จะแสดงผล 1 บรรทัด ถ้าเป็น "1" จะแสดงผล 2 บรรทัด ในกรณีที่จอแสดงผลสามารถแสดงได้เกิน 2 บรรทัด และต้องการให้แสดงผลมากกว่า 2 บรรทัดก็กำหนดบิต N นี้ให้เป็น "1"

บิต F ใช้เลือกความละเอียดของตัวอักษรในการแสดงผล ถ้าบิตนี้เป็น "0" จะเป็นการแสดงผลแบบ 5 x7 จุด และถ้าเป็น "1" จะเป็นการแสดงผลแบบ 5 x10 จุด

ข้อมูลคำสั่งที่ใช้บ่อยคือ ***38H*** เป็นการกำหนดให้โมดูล LCD ทำงานในแบบ 4 บิต แสดงผล 2 บรรทัดและเลือกความละเอียดเป็น 5 x 7 จุด

จุดที่น่าสังเกตคือโมดูลของ CGRAM แบบ 16 ตัวอักษร 1 บรรทัด แม้จะมีบรรทัดการแสดงผลเพียง 1 บรรทัดแต่จะต้องกำหนด N ให้เป็น "1" เนื่องจากแอดเดรสของ DDRAM แบ่งเป็น 2 ช่องคือ 0 x 00 และ 0 x 40

7. คำสั่งเลือกแอดเดรสของ CGRAM เมื่อต้องการกำหนดแอดเดรสของ CGRAM ต้องกำหนดให้บิต 7 เป็น "0" บิต 6 เป็น "1" ส่วนอีกก 6 บิตที่เหลือจะแทนด้วยค่าแอดเดรสของ CGRAM จะต้องทำการกำหนดแอดเดรสด้วยคำสั่งนี้ ก่อนที่จะอ่านหรือเขียนข้อมูลให้ CGRAM โดยแอดเดรสของ CGRAM อยู่ระหว่าง 00H – 3FH

8. คำสั่งเลือกแอดเดรสของ DDRAM ใช้ในการเลือกแอดเดรสของ DDRAM ก่อนที่จะทำการอ่านหรือเขียนข้อมูล โดยบิต 7 ต้องเป็น "1" และอีก 7 บิตที่เหลือจะเป็นค่าแอดเดรสของ DDRAM ซึ่งค่าแอดเดรสของ DDRAM จะอยู่ระหว่าง 8CH – 0FFH ทั้งนี้จำนวนแอดเดรสยังขึ้นอยู่กับการกำหนดสถานะที่บิต N ด้วย หากบิต N เป็น "0" แอดเดรสของ DDRAM จะอยู่ระหว่าง 80H – 0CFH และถ้าบิต N เป็น "1" แอดเดรสของ DDRAM จะมี 2 ช่วง คือ 80H – 87H และ C0H - C7H

9. คำส่งอ่านแฟลก busy และแอดเดรสมีรูปแบบข้อมูลคำสั่งดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
BF	A	A	A	A	A	A	A
แอดเดรสไบต์สูง				แอดเดรสไบต์ต่ำ			

เป็นคำสั่งที่ใช้อ่านแฟลก BUSY (BF) โดยแฟลกนี้จะเป็นตัวบอกสถานะของตัวควบคุม LCD ว่าพร้อมจะรับข้อมูลอยู่หรือไม่ ถ้าหากบิต BF เป็น "0" แสดงว่าตัวควบคุม LCD พร้อมรับข้อมูลหรือคำสั่ง แต่ถ้าเป็น "1" แสดงว่า ขณะนี้ตัวควบคุม LCD ยังอยู่ในกระบวนการทำงานภายในหรือกำลังประมวลผลข้อมูลอยู่ ยังไม่พร้อมรับข้อมูลหรือคำสั่ง เมื่อต้องการอ่านแฟลกต้องกำหนดให้ขา เป็น "1" ด้วย แต่สัญญาณที่ ยังต้องเป็น "0" อยู่เพราะข้อมูลนี้เป็นข้อมูลคำสั่ง

นอกจากนี้ ยังใช้เป็นคำสั่งอ่านข้อมูลแอดเดรสของ CGRAM และ DDRAM ด้วย โดยบิต 0 – บิต 6 เป็นข้อมูลของแอดเดรสที่ต้องการอ่าน

การติดต่อเพื่อเขียนคำสั่งและข้อมูลให้แก่โมดูล LCD

ในการติดต่อกับโมดูล LCD จะต้องมีการหน่วงเวลาหลังจากที่ทำการส่งรหัสคำสั่งหรือข้อมูลเนื่องจากต้องรอให้คอนโทรลเลอร์ภายในโมดูล LCD แปลรหัสความหมายของคำสั่งและทำงานตามคำสั่งให้เรียบร้อยก่อน จากนั้นจึงจะรับข้อมูลหรือคำเนินการต่อไป แต่เนื่องจากการส่งข้อมูลจากพอร์ตอนุกรมไปยังโมดูล LCD นั้นค่อนข้างช้า ผู้ใช้สามารถข้ามขั้นตอนการถ่วงเวลาไปได้ แต่

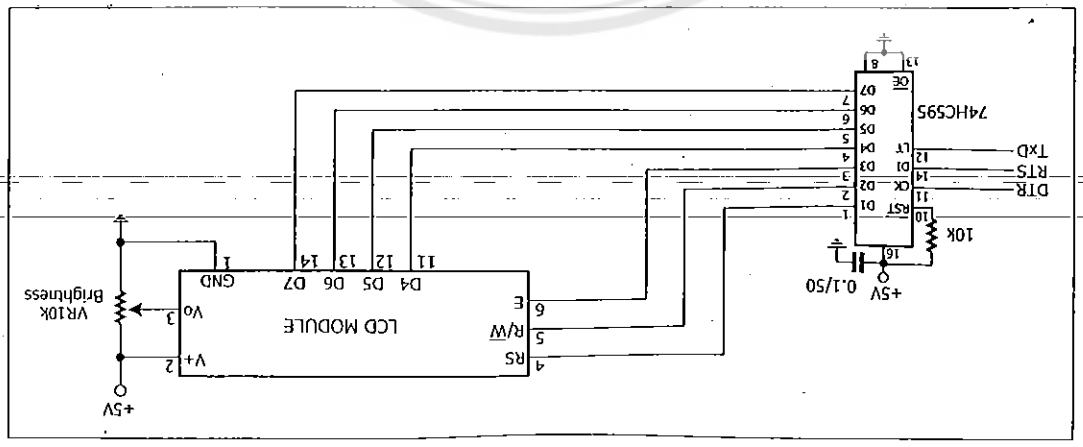
สำหรับการใช้งานโมดูล LCD กับไมโครคอลโทรลเลอร์ หรือกับคอมพิวเตอร์ที่มีความเร็วสูงมาก การหน่วงเวลาก็ยังเป็นสิ่งจำเป็น

เมื่อเริ่มต้นใช้งานจะต้องกำหนดลอจิกให้กับขา RS ของโมดูล LCD เพื่อให้คอนโทรลเลอร์ในโมดูล LCDแปลความหมายของลอจิกที่ขา RS ว่า ข้อมูลต่อไปที่จะได้รับนั้นเป็นรหัสคำสั่งหรือข้อมูลที่ต้องการแสดงผล จากนั้นจะเป็นการส่งข้อมูลมารอที่บัสข้อมูล D0-D7 (กรณีทำงานในโหมด 8 บิต) หรือ D4-D7 (กรณีการทำงานในโหมด 4 บิต)

ขั้นตอนต่อไปจะเป็นการส่งสัญญาณพัลส์ไปที่ขา E เพื่ออินาเบิลโมดูล LCD ให้รับข้อมูลจากบัสข้อมูลเข้าไปโดยพัลส์ที่ป้อนเข้าที่ขา E ของโมดูล LCD ต้องเป็นพัลส์ของขาขึ้น

ในการเขียนข้อมูลเพื่อควบคุมให้โมดูล LCD แสดงผลตามที่ผู้ใช้ต้องการ ต้องส่งคำสั่ง (instruction) แล้วกำหนดโหมดการทำงานให้แก่โมดูล LCD ก่อน จากนั้นจึงค่อยส่งข้อมูล (data) ที่ต้องการแสดงผล เนื่องจากบัสข้อมูลของโมดูล LCD มี 8 เส้นคือ D0-D7 และใช้เป็นทางผ่านของคำสั่งข้อมูล ดังนั้นในการส่งรหัสคำสั่งและข้อมูลจึงต้องอาศัยการกำหนดสัญญาณลอจิกที่ขา RS ถ้าหากที่ขา RS ได้ลอจิก “0” หมายความว่า ข้อมูลที่ป้อนให้โมดูล LCD ขณะนั้นเป็นคำสั่ง ในทางตรงข้าม หากขา RS ได้รับลอจิก “1” ข้อมูลที่ป้อนให้ขณะนั้นเป็นข้อมูลที่ใช้ในการแสดงผล

เมื่อต้องการเขียนหรืออ่านข้อมูลใน CGRAM และ DDRAM เริ่มต้นกำหนแอดเดรสที่ต้องการอ่านหรือเขียนก่อน โดยใช้คำสั่งเลือกแอดเดรส จากนั้นกำหนดให้ขา RS เป็น “1” เพื่อแจ้งให้ตัวควบคุมภายในโมดูล LCD ทราบว่าข้อมูลที่ปรากฏต่อไปนี้เป็นข้อมูลปกติไม่ใช่คำสั่ง ในกรณีที่ต้องการอ่านข้อมูลต้องกำหนดให้ขา R/$\overline{W}$ เป็น “1” ข้อมูลขนาด 8 บิต (หรือ 4 บิต) ก็จะปรากฏบนบัสข้อมูล โดยข้อมูลที่อ่านออกมาได้จะเป็นข้อมูลจากแอดเดรสของ CGRAM หรือ DDRAM ตามที่ต้องการ



รูปที่ 2.12 การขับโมดูล LCD ผ่านชิฟรีจิสเตอร์ 74HC595 ของพอร์ตอนุกรม

ในกรณีที่ต้องการเขียนข้อมูล เมื่อกำหนดแอดเดรสและป้อนลอจิก "1" ให้ขา RS แล้ว แล้วต้องกำหนดให้ขา R/W เป็น "0" ข้อมูลที่อยู่บนบัสข้อมูลจะถูกเขียนลงในรีจิสเตอร์ DR จากนั้นจึงถ่ายทอดลงใน DDRAM ต่อไป

2.4 บอร์ด CP-SB31

บอร์ด CP-SB31 ถูกสร้างขึ้นเพื่อใช้งานควบคุม ซึ่งตรงกับหน้าที่หลักของ CPU ในตระกูล MCS51 คือเป็นไมโครคอนโทรลเลอร์ โครงสร้างทางกายภาพของบอร์ด CP-SB31 มีดังนี้

ลักษณะของบอร์ด CP-SB31

- CPU 8031(ON BOARD) หรือ 8032 , 8052 , 8751

MEMORY

- มี SOCKET ขนาด 28 PIN 2 ตัว สามารถใส่หน่วยความจำได้สูงสุด 96 KB

I/O

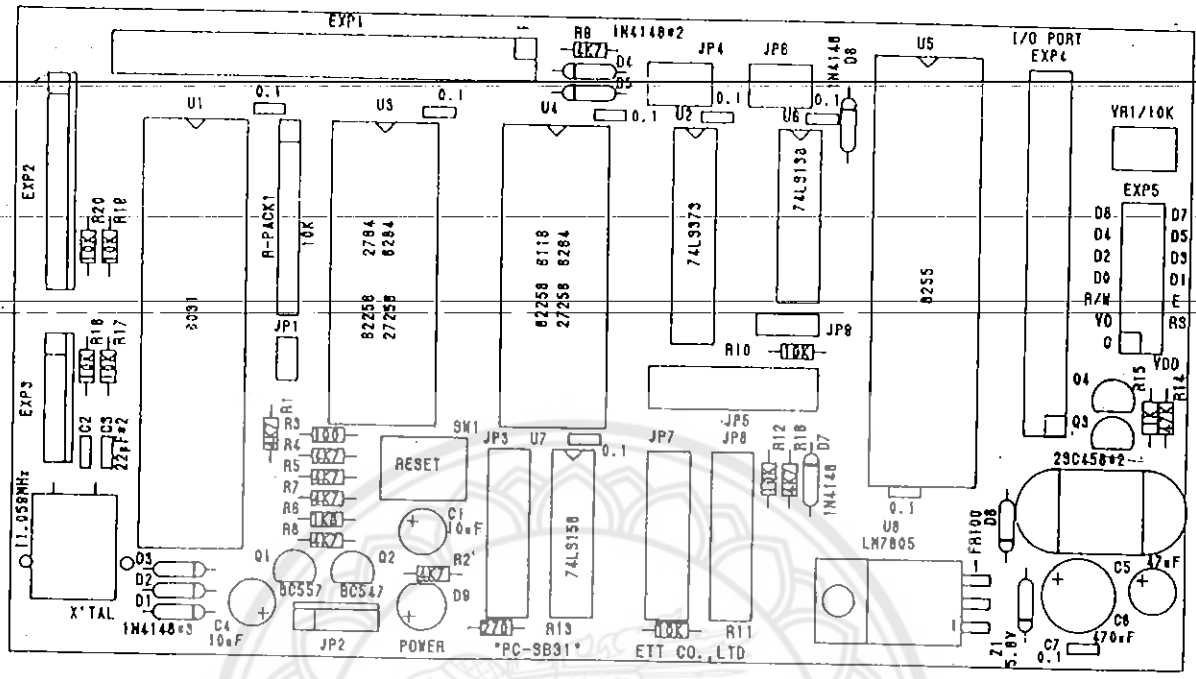
- 8 x 3 บิต INPUT/OUTPUT (8255)
- 8 x 1 บิต INPUT/OUTPUT (PORT 1)
- 1 SERIAL PORT (RS 232)

POWER

- 10 VDC POWER SUPLY JACK
- 5 VDC (REGULATE) 7805 ON BOARD

คุณลักษณะพิเศษของบอร์ด CP-SB31

1. หน่วยความจำสามารถเลือกได้ทั้งขนาด , ตำแหน่ง และลักษณะการทำงาน (DATA MEMORY , CODE MEMORY , CODE&DATA MEMORY)
2. สามารถพัฒนาโปรแกรมได้ทั้งภาษาแอสเซมบลี้ (ร่วมกับ SB31-DEBUGGER) หรือภาษาเบสิค (เพื่อใช้ 8052 AH-BASIC) หรือ ET-EPROM-EMULATOR ก็ได้
3. ต่อกับ LCD ได้ทันที โดยไม่ต้องใช้ I/O พอร์ต
4. มี I/O พอร์ต ขนาด 8 บิต ถึง 4 พอร์ต
5. ต่อร่วมกับอุปกรณ์สนับสนุนของบริษัทอีทีทีได้ทันที เช่น SSRAC , RTC , 72IO , ET-AD ฯลฯ



รูปที่ 2.13 ส่วนประกอบของบอร์ด CP-SB31

2.5 ET-EM PLUS (ET-EPROM EMULATOR)

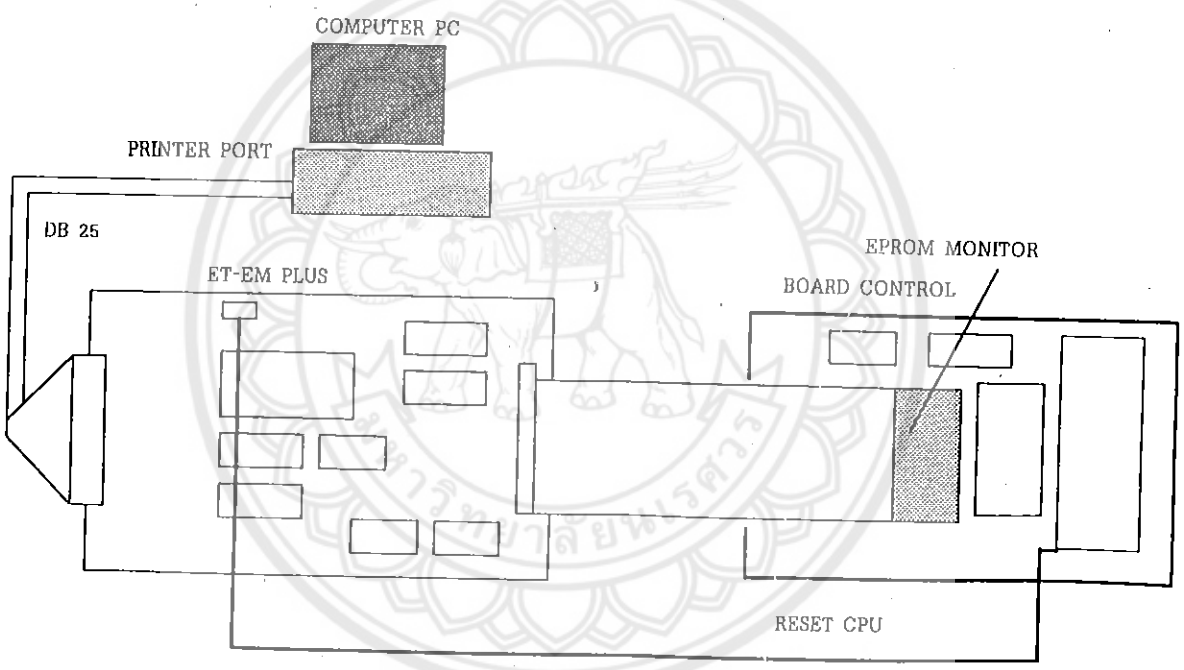
ET-EM PLUS คือบอร์ดวงจรที่ถูกสร้างขึ้นมาเพื่อใช้แทนส่วนของตัว EPROM หรือ RAM ซึ่งจะมีความง่ายและสะดวกกว่าการใช้ EPROM หรือ RAM จริงๆ ทำให้เหมาะสมกับการพัฒนาระบบไมโครต่างๆ หรือ ในการแก้ไขเปลี่ยนแปลงโปรแกรมการทำงานของเครื่องแป้นไปโดยง่าย

เราสามารถเขียนข้อมูลเข้าไปในส่วนของ ET-EM PLUS ได้โดยตรงซึ่งต่างกับการที่เราต้องนำ EPROM นั้นออกมาเปลี่ยนแปลงแล้วจึงนำกลับเข้าไปใส่ในวงจรใหม่

คุณสมบัติของ ET-EM PLUS

1. สามารถส่งผ่านข้อมูลจากเครื่อง PC ได้ทาง PRINT PORT
2. ใช้กับไฟล์ได้หลายรูปแบบ เช่น BINARY FILE , INTEL HEX FILE , MOTOLORA FILE(S FORMAT)
3. สามารถเปลี่ยนแปลงข้อมูลใน ET-EM PLUS ทั้งหมดหรือบางไบต์ก็ได้โดยไม่จำเป็นต้องโหลดไฟล์ใหม่หมด
4. สามารถตั้ง OFSET คือค่าตำแหน่งที่อยู่ของข้อมูลจากไฟล์ได้โดยตรงอย่างอิสระ

5. สามารถเลือกรูปแบบการส่งข้อมูลจากไฟล์ได้หลายรูปแบบ เช่น ส่งเต็มข้อมูล , ส่งข้อมูลเฉพาะไบต์คู่ , ส่งข้อมูลเฉพาะไบต์คี่ ซึ่งทำให้เราสามารถนำ ET-EM PLUS 2 ตัวมาต่อกับ PRINTER PORT 2 ชุด ใช้พัฒนาระบบเครื่องที่จำเป็นต้องใช้ข้อมูล 16 บิต
6. สามารถต่อกับ PRINTER PORT ได้ทั้ง LPT1 , LPT2 หรือ MONO CARD ได้โดยอิสระต่อกัน
7. มีความเร็วในการส่งผ่านข้อมูลสูงมาก
8. สามารถ RESET CPU ได้เมื่อโหลดข้อมูลแล้ว โดยมีแบบ RESET HIGH และ RESET LOW



รูปที่2.14 การเชื่อมต่อ ET-EM PLUS เข้ากับ PRINTER PORT ของเครื่องPC

บทที่ 3
วิธีการดำเนินงานวิจัย

จะเห็นว่าข้อมูลที่ได้ได้จากการทำการทดลอง และการศึกษาจากเอกสารหรือหนังสือ ซึ่งข้อมูลที่ได้จากการทำการทดลอง จะเป็นข้อมูลที่อยู่ในรูปของความเข้าใจและทักษะของความชำนาญ ส่วนข้อมูลที่ได้จากการศึกษาเอกสารหรือหนังสือนั้น จะอยู่ในรูปของบทความ ที่ใช้เป็นข้อมูลในการทำโครงงาน และใช้เป็นแหล่งอ้างอิง วิธีการที่ใช้ในการศึกษาค้นคว้าวิจัยจะใช้วิธีการแบบเอกสาร และการวิจัยแบบทดลอง ซึ่งในการวิจัยแบบเอกสารนั้นทางคณะผู้จัดทำโครงงาน จะศึกษาจากเอกสารหรือหนังสือต่างๆที่มีข้อมูลที่เกี่ยวข้องกับโครงงาน ส่วนการวิจัยแบบทดลองนั้นทางคณะผู้จัดทำโครงงานทำการทดลองในส่วนของการออกแบบคีย์บอร์ด ที่ใช้ในการป้อนข้อมูล, การเชื่อมต่อระหว่างคีย์บอร์ดกับคอมพิวเตอร์, ระบบฐานข้อมูลของร้านอาหาร ซึ่งข้อมูลที่ได้จากการศึกษาจะถูกรวบรวมเพื่อนำมาใช้ในการจัดทำโครงงาน และใช้เป็นแหล่งข้อมูลในการอ้างอิงต่อไป

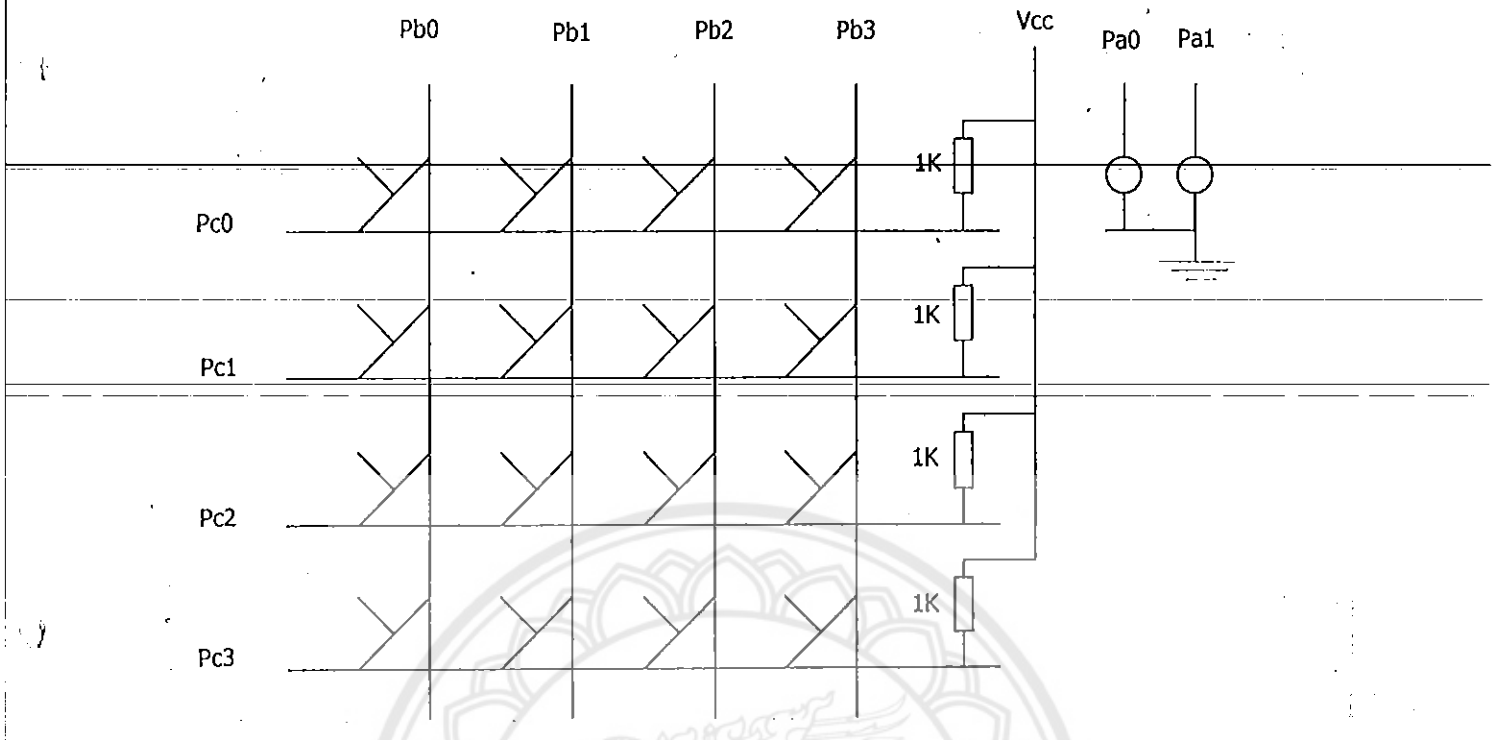
3.1 การสร้างคีย์บอร์ด

อุปกรณ์ที่ใช้

1. แผ่นปริ้น ขนาด 3x5 นิ้ว
2. ปุ่มกด จำนวน 16 ตัว
3. LED จำนวน 3 ตัว สีแดง,สีเขียว,สีเหลือง อย่างละ 1 ตัว
4. พิน ขนาด 34 พิน
5. หัวแร้ง
6. โวลท์มิเตอร์
7. ตะกั่ว
8. ตัวต้านทาน ขนาด 1 กิโลโอห์ม

วิธีการทดลอง

1. ให้บัดกรี คีย์บอร์ดดังรูป 3.1



รูปที่ 3.1 วงจรคีย์บอร์ด

2. ตรวจสอบการทำงานของคีย์บอร์ด ว่าสามารถทำงานได้หรือไม่

3.2 การเขียนโปรแกรมควบคุมการทำงานของคีย์บอร์ด และLCD

อุปกรณ์ที่ใช้

1. บอร์ด CP-SB31, บอร์ด EP-ROM
2. คีย์บอร์ดที่สร้างในการทดลองที่ 1
3. LCD
4. หม้อแปลงไฟ จาก AC 220 V เป็น DC 9 V
5. คอมพิวเตอร์ที่มีการเซตโปรแกรม Brief ไว้เรียบร้อยแล้ว

วิธีการทดลอง

1. ติดตั้งบอร์ด CP-SB31, บอร์ด EP-ROM, คอมพิวเตอร์, LCD และหม้อแปลงไฟเข้าด้วยกัน
2. เขียนโปรแกรมในการรับค่าที่ได้มาจากคีย์บอร์ด และกำหนดค่าที่ได้รับมาเป็นตัวอักษร
3. เขียนโปรแกรมในการแสดงผลตัวอักษรที่ได้กำหนดไว้บน LCD
4. รันโปรแกรม และสังเกตผลการทดลอง
5. แก้ไขปัญหาที่เกิดขึ้น
6. สรุปผล

3.3 การเขียนโปรแกรมเชื่อมต่อระหว่างบอร์ด CP-SB31 กับ คอมพิวเตอร์

อุปกรณ์ที่ใช้

1. สายสัญญาณที่ใช้ในการเชื่อมต่อระหว่างบอร์ด CP-SB31 กับ คอมพิวเตอร์ผ่านพอร์ตอนุกรม
2. คอนเน็กเตอร์ อนุกรม
3. คอนเน็กเตอร์ 4 ขา
4. โปรแกรม Visual Basic 6.0

วิธีการทดลอง

1. ทำการเชื่อมสายสัญญาณเข้ากับคอนเน็กเตอร์อนุกรม 9 ขา โดยมีการเชื่อมต่อดังนี้
   - เชื่อมสายสัญญาณเส้นที่ 1 เข้ากับขาที่ 2 (ขาอินพุต)
   - เชื่อมสายสัญญาณเส้นที่ 2 เข้ากับขาที่ 3 (ขาเอาต์พุต)
   - เชื่อมสายสัญญาณเส้นที่ 3 เข้ากับขาที่ 5 (ขากราวด์)
2. ทำการเชื่อมปลายสายสัญญาณอีกข้างหนึ่งเข้ากับคอนเน็กเตอร์ 4 ขา โดยมีการเชื่อมต่อดังนี้
   - เชื่อมสายสัญญาณเส้นที่ 1 เข้ากับขาที่ 3 (ขาเอาต์พุต)
   - เชื่อมสายสัญญาณเส้นที่ 2 เข้ากับขาที่ 2 (ขาอินพุต)
   - เชื่อมสายสัญญาณเส้นที่ 3 เข้ากับขาที่ 5 (ขากราวด์)
3. ทำการเขียนโปรแกรมในการควบคุมการส่งข้อมูลระหว่างคอมพิวเตอร์กับบอร์ด CP-SB31 โดยใช้ Visual basic 6.0 และภาษาซี
4. ทำการทดลองโปรแกรมที่เขียนขึ้นโดยลองส่งข้อมูลจากคอมพิวเตอร์ไปยังบอร์ด CP-SB31เพื่อให้ไปแสดงผลยังLCDและส่งข้อมูลจากบอร์ดCP-SB31(โดยการกดคีย์บอร์ด) ไปยังคอมพิวเตอร์
5. ปรับปรุงและแก้ไขโปรแกรมให้สมบูรณ์

3. 4 ระบบฐานข้อมูล

อุปกรณ์ที่ใช้

1. Microsoft Access 97
2. Microsoft Visual Basic 6.0

วิธีการทดลอง

1.นำข้อมูลที่อ้างอิงได้จากร้านอาหารจริงมาสร้างเป็นฐานข้อมูลโดยใช้ Microsoft Access 97 โดยในฐานข้อมูลนั้นจะแบ่งเป็น 3 ฟิลด์ คือ

- id ใช้เก็บรหัสอาหารที่สร้างขึ้น โดยเก็บเป็นเลข 3 หลัก เริ่มตั้งแต่ 101 เป็นต้นไป โดยจะเก็บเป็นแบบ Number

- f_100 ใช้เก็บชื่อรายการอาหาร โดยจะเก็บเป็น Text

- cost ใช้เก็บราคาอาหาร โดยเก็บเป็น Currency

ซึ่งเราจะใช้ส่วนของ id เป็นคีย์หลัก นั่นคือไม่สามารถมีค่าซ้ำกันได้

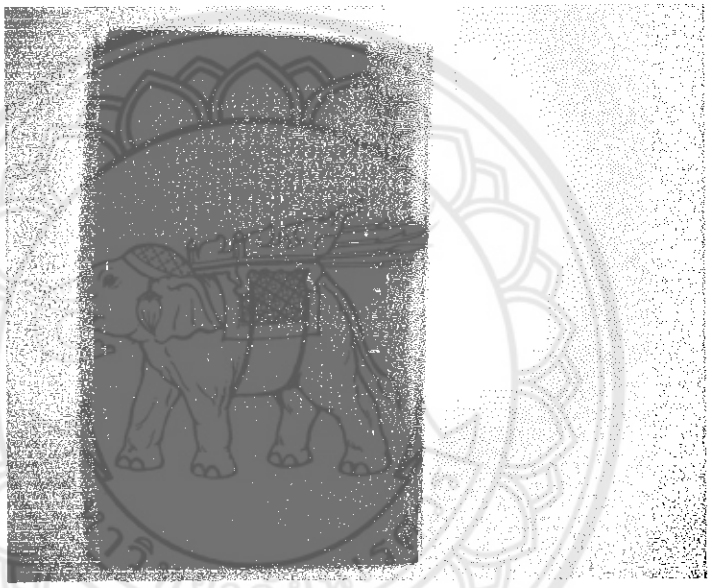
2. สร้างโปรแกรมระบบฐานข้อมูลขึ้นมาโดยใช้ Microsoft Visual Basic 6.0 และใช้ฐานข้อมูลที่ได้จากการสร้างในขั้นตอนแรก

บทที่ 4
ผลการทดลองและผลการวิเคราะห์

จากการทดลองจากบทที่แล้ว ได้แบ่งการทดลองออกเป็นการทดลอง 5 การทดลองในบทนี้ก็จะกล่าวถึงการผลการทดลอง ซึ่งก็จะแบ่งเป็น4 การทดลอง ด้วยกัน คือ

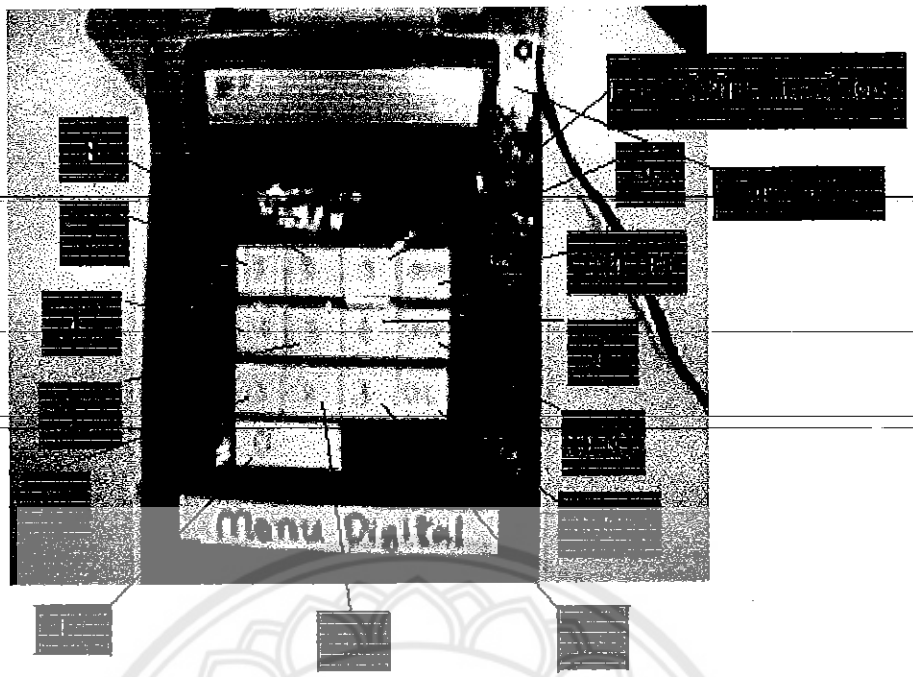
4.1 การสร้างคีย์บอร์ด

คีย์บอร์ดที่สร้างจะมีลักษณะดังรูป 4.1 และ 4.2



รูปที่4.1 ลักษณะทั่วไปของคีย์บอร์ด

คีย์บอร์ดที่เราสร้างขึ้นมานี้ ยังไม่สามารถทำงานได้ ถ้าจะให้ทำงานได้นั้นจะต้องมีการเขียนโปรแกรมควบคุม โดยจะกำหนดว่าปุ่มนั้นว่ามีหน้าที่อย่างไร เช่น ลบข้อมูล ส่งข้อมูล หรือเปิด-ปิดการทำงานของคีย์บอร์ด เป็นต้น หรือจะกำหนดค่าของปุ่มนั้นเป็นตัวเลขใดๆ



รูปที่ 4.2 คีย์บอร์ดที่ทำสำเร็จแล้ว

การใช้งานคีย์บอร์ดมีดังนี้

1. กดปุ่ม ON\OFF (จะเห็นไฟเขียวติด)
2. กดหมายเลขโต๊ะ 2 ตัว
3. กดรหัสอาหาร 3 ตัว โดยดูรหัสอาหารจากเมนูที่มีให้
4. กดจำนวนของอาหารที่สั่ง ซึ่งสามารถกดเป็นเลขหนึ่งหลัก หรือ สองหลักก็ได้
5. กดปุ่ม SEND
6. ถ้าต้องการสั่งอาหารต่อก็ให้ทำตามข้อ 2 ถึง 5
7. ถ้าต้องการลบตัวเลขที่กดไป ให้กดปุ่ม DEL
8. ถ้าสั่งอาหารเสร็จแล้วให้กดปุ่ม ON\OFF หรือ ปล่อยไว้ก็ได้เพราะคีย์บอร์ดจะทำการตัดการทำงานของตัวมันเอง

4.2 การเขียนโปรแกรมควบคุมการทำงานของคีย์บอร์ด และLCD

เป็นการเขียนโปรแกรมในการควบคุมให้คีย์บอร์ดทำงานได้ และสามารถนำไปแสดงบนหน้าจอได้ ในการทดลองนั้นจะมีปัญหาเกิดขึ้นมากมาย เช่น

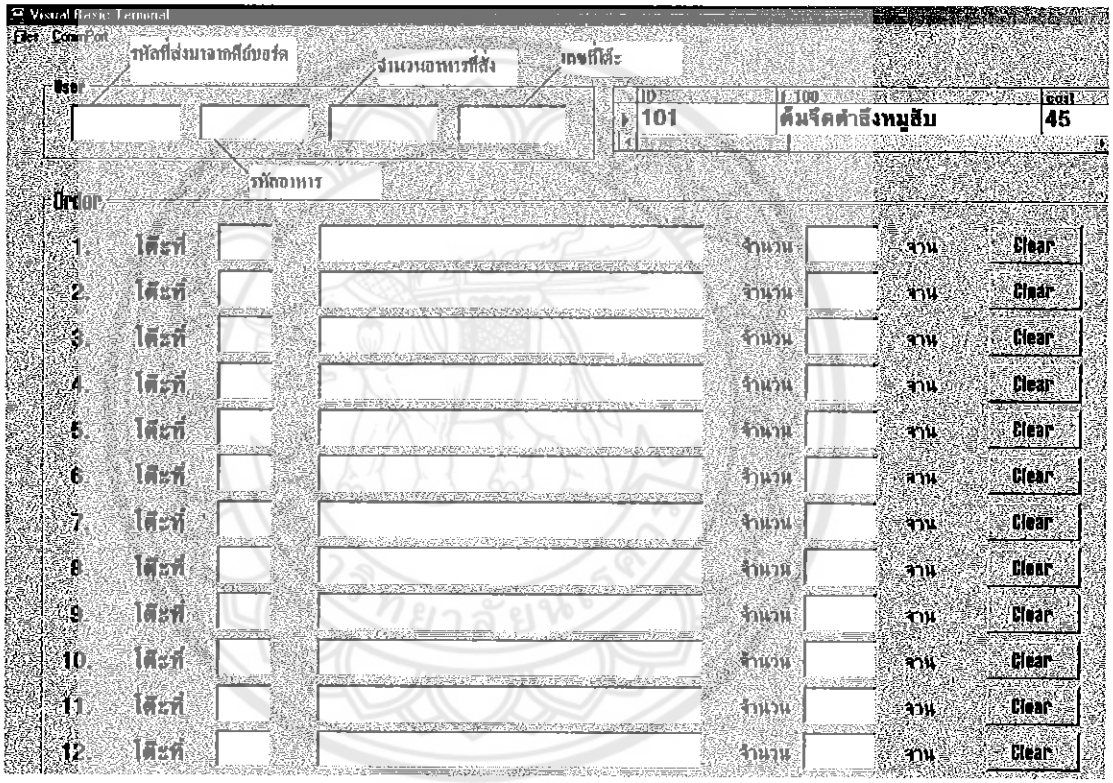
- ต่อบอร์ดผิดพลาด ทำให้ไฟไม่เข้าบอร์ด
- คีย์บอร์ดทำงานไม่ได้

- โปรแกรมที่ทดสอบรันไม่ผ่าน
- และอื่นๆอีกมากมาย

ซึ่งจะต้องพยายามแก้ปัญหาไปเรื่อยๆ จนสามารถ ได้โปรแกรมสำเร็จรูปที่สามารถควบคุมการทำงานของคีย์บอร์ดและสามารถนำไปแสดงผลบนหน้าจอ LCD ได้

4.3 การเขียนโปรแกรมเชื่อมต่อระหว่างบอร์ด CP-SB31 กับ คอมพิวเตอร์

ได้โปรแกรมที่ใช้ในการรับส่งข้อมูลระหว่างคอมพิวเตอร์ กับ บอร์ด CP-SB31 แสดงได้ดังรูปที่4.2



รูปที่ 4.3 หน้าจอรับข้อมูลจากบอร์ด CP-SB31

รูปฟอร์มที่เห็นอยู่ข้างบนนี้คือ หน้าจอที่จะแสดงบนคอมพิวเตอร์หลักที่จะบอกให้พ่อครัวทราบว่า มีลูกค้าสั่งอาหารอะไร จากโต๊ะไหนและมีจำนวนเท่าไร

ในการใช้งานจริงๆนั้น เมื่อมีลูกค้าเข้ามาในร้านอาหาร ไม่ว่าจะนั่งโต๊ะใด ก็จะมีเมนูอาหารและคำอธิบาย ในการสั่งอาหารให้ทุกโต๊ะ เมื่อลูกค้าเลือกอาหารก็จะสั่งอาหารผ่านทางคีย์บอร์ดที่เตรียมให้ โดยมีลักษณะของการกดคือ

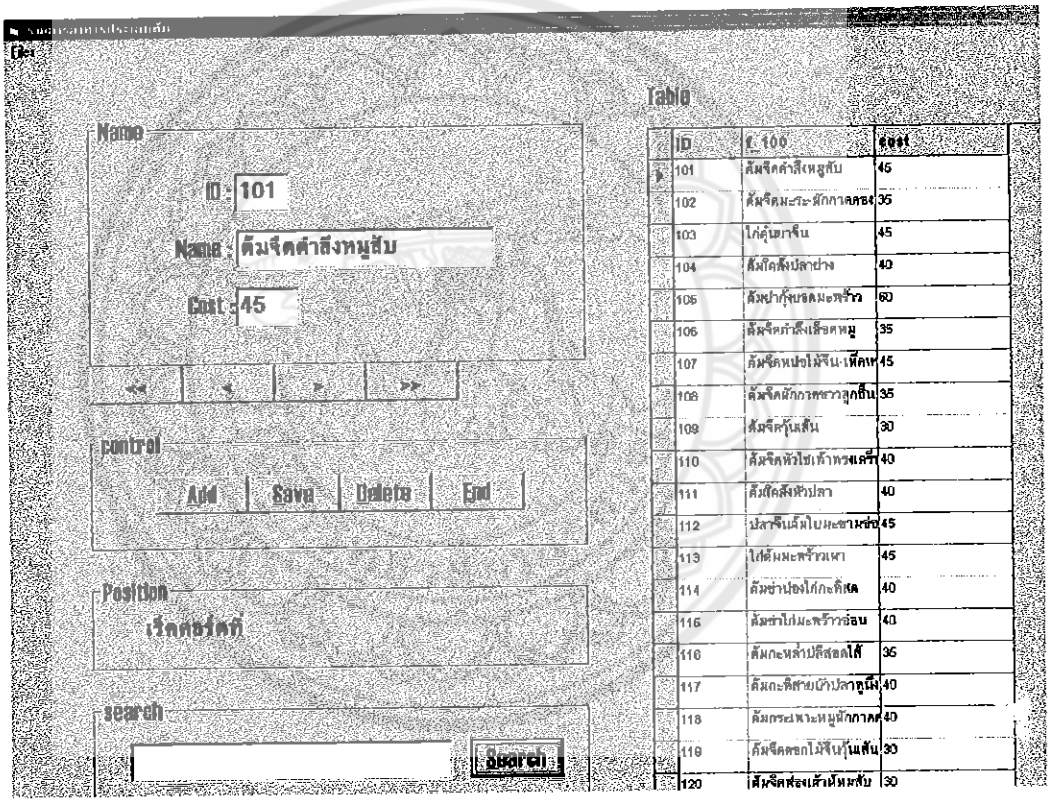
01102 = 01

โดยสองตัวแรกคือหมายเลขโต๊ะที่สั่งอาหาร สามตัวต่อมาคือ รหัสอาหารที่ลูกค้าเลือก ซึ่งก็จะไปเป็นชื่อและราคาของอาหารที่ตรงกับรหัสนี้ออกมาแสดงให้พ่อครัวทราบว่าลูกค้าสั่งอะไรมา สองตัวสุดท้ายจะบอกจำนวนอาหารชนิดนั้นที่ลูกค้าสั่งมา ซึ่งก็จะแสดงให้พ่อครัวทราบด้วยเช่นกัน

ซึ่งจะเห็นได้ว่ามีการให้ลำดับความสำคัญของการสั่งคือ ถ้าชนิดใดสั่งก่อน อาหารชนิดนั้นก็จะอยู่ในลำดับต้นๆ ถ้าพ่อครัวทำอาหารชนิดใดเสร็จแล้ว ก็จะมากดเคลียร์ ลำดับอาหารที่อยู่ถัดลงไปก็จะถูกเลื่อนขึ้นมาให้มีลำดับความสำคัญมากขึ้น

4.4 ระบบฐานข้อมูล

เราจะได้โปรแกรมจัดการฐานข้อมูล ซึ่งประกอบด้วยรูปฟอร์มหลายๆฟอร์มดังนี้



รูปที่ 4.4 หน้าจอแสดงฐานข้อมูลรายการอาหาร

หน้าจอนี้เป็นหน้าจอที่แสดงรายการอาหารทั้งหมด โดยจะมีปุ่มต่างๆดังนี้

- Add: ใช้สำหรับการเพิ่มรายการอาหารเมื่อกดปุ่มนี้แล้วจะมีการเลื่อนรหัสอาหารไปอีก 1 ลำดับ และตั้งให้ไม่สามารถที่จะแก้ไขในส่วนนี้ได้ (เพื่อป้องกันการมีรหัสซ้ำกัน) และจะมีปุ่ม OK และจะมีปุ่ม Cancel แสดงขึ้นมา ถ้าต้องการยกเลิกข้อมูล ก็ให้กดปุ่ม Cancel แต่ถ้าต้องการจะเพิ่มข้อมูลจริงๆ ก็ให้พิมพ์ชื่ออาหารและราคาลงไป (ถ้ามีการกรอกไม่ครบแล้วกดปุ่ม OK ก็จะ

มีคำเตือนให้ไปกรอกข้อมูลให้ครบ) หลังจากกดปุ่ม OK จะปรากฏปุ่ม Save ขึ้นมา ให้กดปุ่ม Save ก็จะเสร็จสิ้นในส่วนของการเพิ่มข้อมูล

- Delete: ใช้สำหรับการลบข้อมูลที่ต้องการนำออกไปจากเมนู
- Search: จะเป็นตัวค้นหารายการอาหารโดยใช้รหัสอาหารเป็นตัวค้นหา

ID	C_100	cost
101	ต้มจืดตำลึงหมูสับ	$45.00
102	ต้มจืดมะระผักกา	$35.00
103	ไก่ตุ๋นยาจีน	$45.00
104	ต้มโคล้งปลาช่าง	$40.00
105	ต้มยำกุ้งยอดมะพ	$60.00
106	ต้มจืดตำลึงเลือด	$35.00
107	ต้มจืดหน่อไม้จีน-	$45.00
108	ต้มจืดผักกาดขาว	$35.00
109	ต้มจืดวุ้นเส้น	$30.00
110	ต้มจืดหัวไชเท้าห	$40.00
111	ต้มโคล้งหัวปลา	$40.00
112	ปลาจีนต้มใบมะข	$45.00
113	ไก่ต้มมะพร้าวเผ	$45.00
114	ต้มข่าน้องไก่กะทิ	$40.00
115	ต้มข่าไก่มะพร้าว	$40.00
116	ต้มกะหล่ำปลีคล	$35.00
117	ต้มกะทิลาซัญปะ	$40.00
118	ต้มกระเพาะหมูบ้	$40.00

รูปที่ 4.5 ฐานข้อมูลที่เก็บอยู่ใน Microsoft Access 97

บทที่ 5
บทสรุป

จากการทดลองและผลการทดลองในบทที่ผ่านมาก็จะสามารถนำมาสรุปไว้ในบทนี้ ได้ดังนี้ คือ

5.1 การสร้างคีย์บอร์ด

เราจะได้คีย์บอร์ดที่สามารถใช้งานได้เพื่อใช้ในการทดลองอื่นอีกต่อไป

5.2 การเขียนโปรแกรมควบคุมการทำงานของคีย์บอร์ดและ LCD

ได้โปรแกรมในการควบคุมการทำงานของคีย์บอร์ด และสามารถนำไปแสดงผลบน LCD ได้อย่างไม่มีข้อผิดพลาด

5.3 การเขียนโปรแกรมเชื่อมต่อระหว่างบอร์ด CP-SB31 กับ คอมพิวเตอร์

จากการทดลองเราจะได้โปรแกรมที่ใช้ในการติดต่อระหว่างบอร์ด CP-SB31 กับคอมพิวเตอร์ ซึ่งสามารถทำให้บอร์ด CP-SB31 ส่งข้อมูลไปยังคอมพิวเตอร์ได้ และ คอมพิวเตอร์ก็สามารถส่งข้อมูลไปยังบอร์ด CP-SB31 ได้ตามที่ต้องการโดยผ่านพอร์ตอนุกรม

5.4 ระบบฐานข้อมูล

ได้ระบบฐานข้อมูลที่เก็บรายการอาหาร และสามารถเพิ่มหรือลดรายการอาหารได้ นอกจากนี้ยังมีฐานข้อมูลที่เก็บรายรับของร้านอาหาร

เอกสารอ้างอิง

[1] อรรถนพ พีรชาติ "การใช้งานพอร์ตอนุกรม." **Hobby Electronic.** ฉบับที่ 107, กุมภาพันธ์ 2544. 93 หน้า หน้า 48-51

[2] พิพัฒน์ หิรัณยวณิชชากร **ระบบการสื่อสารข้อมูลและเครือข่ายคอมพิวเตอร์.** กรุงเทพมหานคร : โรงพิมพ์ซีเอ็ดยูเคชั่น จำกัด. 2542.

[3] ยืน ภู่อรวรรณ. **ไมโครโพรเซสเซอร์ไมโครคอมพิวเตอร์.** กรุงเทพมหานคร : โรงพิมพ์ซีเอ็ดยูเคชั่น จำกัด. 2521.

[4] กฤษดา ใจเย็น, อรรถพล บุญยะโภคา, ชัยวัฒน์ ลิ้มพรจิตรวิไล. **เรียนรู้และปฏิบัติการเชื่อมต่อคอมพิวเตอร์กับอุปกรณ์ภายนอกผ่านพอร์ตอนุกรม.** กรุงเทพมหานคร : บริษัท อินโนเวตีฟ เอ็กเพอริเมนต์ จำกัด.

[5] ศุภชัย สมพานิช **Databaseprogramming กับ Visual Basic ฉบับมืออาชีพ,** กรุงเทพมหานคร :อินโฟเพรส, 2543

ประวัติผู้จัดทำโครงงาน

นายต่อศักดิ์ สุนทรพันธุ์
เกิดเมื่อ วันที่ 25 กุมภาพันธ์ 2523 ที่ กรุงเทพมหานคร
ภูมิลำเนา 2/2 หมู่ที่ 2 ต.ห้วยถั่วใต้ อ.หนองบัว จ.นครสวรรค์ 60110
จบการศึกษาระดับประถมศึกษาจาก โรงเรียนบ้านห้วยถั่วใต้ จังหวัดนครสวรรค์
จบการศึกษาระดับมัธยมศึกษาจาก โรงเรียนชุมแสงชนูทิศ จังหวัดนครสวรรค์
ปัจจุบันกำลังศึกษาในระดับปริญญาตรีชั้นปีที่ 4 คณะวิศวกรรมศาสตร์ สาขาวิชาวิศวกรรมคอมพิวเตอร์
มหาวิทยาลัยนเรศวร จังหวัดพิษณุโลก

นายวัฒนพล ชัยเนตร
เกิดเมื่อ วันที่ 23 เมษายน 2523 ที่ จังหวัดพิษณุโลก
ภูมิลำเนา 26/20 ถ.พญาเสือ ซอย 4 ต.ในเมือง อ.เมือง จ.พิษณุโลก 65000
จบการศึกษาระดับประถมศึกษาจาก โรงเรียนผดุงราษฎร์ จังหวัดพิษณุโลก
จบการศึกษาระดับมัธยมศึกษาจาก โรงเรียนผดุงราษฎร์ จังหวัดพิษณุโลก
ปัจจุบันกำลังศึกษาในระดับปริญญาตรีชั้นปีที่ 4 คณะวิศวกรรมศาสตร์ สาขาวิชาวิศวกรรมคอมพิวเตอร์
มหาวิทยาลัยนเรศวร จังหวัดพิษณุโลก

นายอามร กาพย์แก้ว
เกิดเมื่อ วันที่ 6 พฤศจิกายน 2522 ที่ จังหวัดเพชรบูรณ์
ภูมิลำเนา 98 หมู่ที่ 7 ต.ท่าแดง อ.หนองไผ่ จ.เพชรบูรณ์ 67140
จบการศึกษาระดับประถมศึกษาจาก โรงเรียนบ้านนาทุ่ง จังหวัดเพชรบูรณ์
จบการศึกษาระดับมัธยมศึกษาจาก โรงเรียนเพชรละครวิทยา จังหวัดเพชรบูรณ์
ปัจจุบันกำลังศึกษาในระดับปริญญาตรีชั้นปีที่ 4 คณะวิศวกรรมศาสตร์ สาขาวิชาวิศวกรรมคอมพิวเตอร์
มหาวิทยาลัยนเรศวร จังหวัดพิษณุโลก