เครื่องสั่งอาหารไร้สาย

WIRELESS FOOD ORDERING MACHINE

โดย

นายผโลทัย พฤกษ์ไพรผดุง

นายพีระพงษ์ พงษ์ทองหล่อ

นายพีระพัฒน์ สังข์สุนทร

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2553

เครื่องสั่งอาหารไร้สาย

WIRELESS FOOD ORDERING MACHINE

โดย

นายผโลทัย พฤกษ์ไพรผดุง 50010991

นายพีระพงษ์ พงษ์ทองหล่อ 50011125

นายพีระพัฒน์ สังข์สุนทร 50011128

อาจารย์ที่ปรึกษา

ผศ.ดร. สุทธิชัย นพนาคีพงษ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2553

ฝ่านการตรวจชิ้นงานแล้ว

(ลงชื่อ).....ผู้ตรวจ

ฝ่านการตรวจรูปเล่มแล้ว

(ลงชื่อ).....ผู้ตรวจ

ปริญญานิพนธ์ปีการศึกษา 2553

สาขาวิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง เครื่องสั่งอาหารไร้สาย

WIRELESS FOOD ORDERING MACHINE

ผู้จัดทำ

1. นายผโลทัย พฤกษ์ไพรผดุง 50010991
2. นายพีระพงษ์ พงษ์ทองหล่อ 50011125
3. นายพีระพัฒน์ สังข์สุนทร 50011128

..... อาจารย์ที่ปรึกษา

(ผศ.ดร.สุทธิชัย นพนาคีพงษ์)

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จได้ด้วยความกรุณาของ ผศ.ดร.สุทธิชัย นพนาคีพงษ์ อาจารย์ที่ปรึกษาปริญญานิพนธ์ซึ่งได้ให้ความรู้คำปรึกษาข้อชี้แนะให้ความกรุณาในการแก้ไขข้อบกพร่องต่างๆ และความช่วยเหลือในหลายสิ่งหลายอย่างจนกระทั่งลุล่วงไปได้ด้วยดี ผู้วิจัยขอกราบขอบพระคุณเป็นอย่างสูงมา ณ ที่นี้

นายผโลทัย พฤกษ์ไพรผดุง
นายพีระพงษ์ พงษ์ทองหล่อ
นายพีระพัฒน์ สังข์สุนทร

คณะผู้จัดทำ

เครื่องสั่งอาหารแบบไร้สาย

Wireless food ordering machine

โดย	นายผโลทัย	พฤกษ์ไพรผดุง	50010991
	นายพีระพงษ์	พงษ์ทองหล่อ	50011125
	นายพีระพัฒน์	สังข์สุนทร	50011128
อาจารย์ที่ปรึกษา	ผศ.ดร.สุทธิชัย	นพนาคีพงษ์	

ปริญญานิพนธ์นี้นำเสนอเครื่องสั่งอาหารโดยใช้หลักการสื่อสารแบบไร้สายที่ความถี่ 2.4 GHz ในการรับ–ส่งข้อมูลระหว่างชุดสั่งงานของลูกค้ากับหน่วยประมวลผลกลาง เครื่องสั่งอาหารมีการควบคุมโดยไมโครคอนโทรลเลอร์ และลูกค้าใช้การป้อนรหัสของรายการอาหารบนคีย์แพดซึ่งรายการอาหารและยอดเงินก็จะแสดงออกทางจอมอนิเตอร์

ABSTRACT

This project proposes the wireless food ordering machine .The wireless communications via 2.4 GHz TRW is used in order to send data between sets of customers and a central processing unit. The wireless food ordering machine is controlled by microcontroller and the customer places order by entering data on the keypad. The orders, as well as the balance will show on the monitor.

สารบัญ

สารบัญรูป

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญตาราง

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ปัจจุบันนี้ร้านอาหารได้มีการเปิดกิจการอย่างมากมาย ส่งผลให้เกิดการแข่งขันกันในด้านต่างๆไม่ว่าจะเป็น รสชาติอาหารที่ถูกปาก ความสะอาด ความมีระเบียบเรียบร้อยและการบริการที่ดีอันเป็นสิ่งจำเป็นที่ร้านต่างๆต้องมีเพื่อเรียกลูกค้าให้เข้าร้านได้มากที่สุด สำหรับปัญหาของร้านอาหารทั่วไปในการให้บริการลูกค้า การที่ต้องใช้พนักงานมายืนรับ Order จากลูกค้านั้นอาจทำให้เกิดความไม่สะดวกหลายประการคือ

1.ลูกค้าจำเป็นต้องสั่งอาหารทันทีในขณะนั้นอาจทำให้ลูกค้าสั่งอาหารได้ไม่อิสระตามที่ควรเนื่องจากมีพนักงานยืนรอ

2.การที่ใช้พนักงานรับ Order นั้นอาจรับรายการอาหารจากลูกค้าช้า ลูกค้าต้องรอนาน จนอาจทำให้ลูกค้าเกิดความไม่พอใจและเป็นเหตุให้ร้านอาหารเสียลูกค้าเป็นได้

ดังนั้นผู้จัดทำจึงมีแนวคิดแก้ไขปัญหานี้โดยการสร้าง เครื่องสั่งอาหารแบบไร้สายขึ้นเพื่อที่จะนำมาประยุกต์ใช้แก้ไขปัญหาดังกล่าว โดยลูกค้าไม่จำเป็นต้องสั่งอาหารในทันที สามารถคิด เมนูได้อย่างอิสระเพราะไม่มีพนักงานยืนรอ

1.2 วัตถุประสงค์

1) เพื่อใช้อำนวยความสะดวกในการสั่งอาหาร โดยไม่จำเป็นที่จะต้องมีพนักงานมารับOrder
2) เพื่ออำนวยความสะดวกให้ลูกค้ามีความสะดวกและยืดหยุ่นในการสั่งอาหารได้มากขึ้น

1.3 ขอบเขตของปริญญานิพนธ์

1. ศึกษาความเร็วในการรับส่งข้อมูล และระยะสื่อสารของ RF Module (TRW-2.4GHz) เพื่อนำไปประยุกต์ใช้กับโครงงาน
2. ศึกษาและใช้งาน โปรแกรม Visual Studio ในการสร้างฐานข้อมูลและแสดงผลผ่านจอคอมพิวเตอร์
3. ศึกษาการทำงานและคำสั่งต่างๆที่ใช้ภายในไมโครคอนโทรลเลอร์
4. สร้างเครื่องสั่งอาหารไร้สายที่แสดงผลออก LCD และ สามารถทำงานตามที่กำหนดไว้

บทที่2
ทฤษฎีและหลักการที่เกี่ยวข้อง

2.1ทฤษฎีของไมโครคอนโทรลเลอร์ตระกูล MCS-51

ไมโครคอนโทรลเลอร์ (Microcontroller)

เป็นอุปกรณ์ไอซี (IC: Integrated Circuit) ที่สามารถโปรแกรมการทำงานได้ซับซ้อน สามารถรับข้อมูลในรูปสัญญาณดิจิตอลเข้าไปทำการประมวลผล แล้วส่งผลลัพธ์ข้อมูลดิจิตอลออกมาเพื่อนำไป ใช้งานตามที่ต้องการได้

ไมโครคอนโทรลเลอร์ภายในชิพจะมีหน่วยความจำและ Port อยู่ในชิพเพียงตัวเดียวซึ่งอาจจะเรียกได้ว่าเป็น คอมพิวเตอร์ชิพเดี่ยว โดยไมโครคอนโทรลเลอร์เป็นไมโครโปรเซสเซอร์ชนิดหนึ่ง เช่นเดียวกับหน่วยประมวลผลกลาง (CPU: Central Processing Unit) ที่ใช้ในคอมพิวเตอร์ แต่ได้รับการพัฒนาแยกออกมาภายหลังเพื่อนำไปใช้ในวงจรทางด้านงานควบคุม คือ แทนที่ในการใช้งานจะต้องต่อวงจรภายนอกต่าง ๆ เพิ่มเติมเช่นเดียวกับไมโครโปรเซสเซอร์ ก็จะทำการรวมวงจรที่จำเป็น เช่น หน่วยความจำ, ส่วนอินพุท/เอาท์พุท บางส่วนเข้าไปในตัว ไอซีเดียวกัน และเพิ่มวงจรบางอย่างเข้าไปด้วยเพื่อให้มีความสามารถเหมาะสมกับการใช้ในงานควบคุม เช่น วงจรตั้งเวลา , วงจรการสื่อสารอนุกรม วงจรแปลงสัญญาณอนาลอกเป็นดิจิตอล เป็นต้น สรุปคือ

Microcontroller = Microprocessor + Memory + I/O

ไมโครคอนโทรลเลอร์สามารถนำไปประยุกต์ใช้งานอย่างกว้างขวาง โดยมักจะเป็นการนำไปใช้ฝังในระบบของอุปกรณ์อื่น ๆ (Embeded Systems) เพื่อใช้ควบคุมการทำงานบางอย่าง เช่น ใช้ในรถยนต์ , เตาอบไมโครเวฟ , เครื่องปรับอากาศ , เครื่องซักผ้าอัตโนมัติ เป็นต้น เพราะว่าไมโครคอนโทรลเลอร์มีข้อดีเหมาะสมต่อการใช้ในงานควบคุมหลายประการ เช่น

- ชิพไอซีและระบบที่ได้มีขนาดเล็ก
- ระบบที่ได้มีราคาถูกกว่าการใช้ชิพไมโครโพรเซสเซอร์
- วงจรที่ได้จะมีความซับซ้อนน้อย ช่วยลดข้อผิดพลาดที่อาจจะเกิดขึ้นได้ในการต่อวงจร
- มีคุณสมบัติเพิ่มเติมสำหรับงานควบคุมโดยเฉพาะซึ่งใช้งานได้ง่าย
- ช่วยลดระยะเวลาในการพัฒนาระบบได้

ไมโครคอนโทรลเลอร์มีหลายยี่ห้อ หลายตระกูล และหลายเบอร์ด้วยกัน ซึ่งแต่ละเบอร์ก็จะมีโครงสร้างภายในและความสามารถในการทำงานที่แตกต่างกันทำให้เลือกใช้กับงานได้อย่างเหมาะสม ในเครื่องสั่งอาหารแบบไร้สายจะมีการนำไมโครคอนโทรเลอร์มาใช้งานด้วย ดังนั้นจึงจำเป็นต้องทราบถึงทฤษฎีต่างๆของไมโครคอนโทรเลอร์ดังนี้

2.1.1 คุณสมบัติทั่วไปของไมโครคอนโทรเลอร์ตระกูล MCS-51

คุณสมบัติทั่วไปของไมโครคอนโทรเลอร์ตระกูล MCS-51มีดังต่อไปนี้

- เป็นไมโครคอนโทรเลอร์ขนาด 8บิต
- มีวงจรออสซิลเลเตอร์และวงจรผลิตสัญญาณนาฬิกาภายในไอซี
- มีขาสัญญาณอินพุตและเอาต์พุตจำนวน 32 บิต
- สามารถเชื่อมหน่วยความจำข้อมูลภายนอก (External Data Memory) โดยการอ้างอิงตำแหน่งแอดเดรสได้ถึง 64 K
- สามารถเชื่อมต่อหน่วยความจำโปรแกรมภายนอก (External Program Memory) โดยการอ้างตำแหน่งแอดเดรสได้ถึง 64 K
- มีหน่วยความจำข้อมูลภายในตัวเอง (On-Chip Data Memory) มีขนาด 128 ไบต์ โดยเฉพาะหมายเลข8032 และ M8052 จะมีหน่วยความจำในส่วนนี้ถึง 256 ไบต์

- มีหน่วยความจำโปรแกรมภายในตัว (On-Chip Program Memory) ขนาด 4 K โดยเฉพาะหมายเลข 8052 จะมีหน่วยความจำในส่วนนี้ถึง 8 K สำหรับหมายเลข 8031 และ M8032 จะไม่มีหน่วยความจำในส่วนนี้
- หน่วยความจำข้อมูลภายในบางส่วนสามารถเข้าถึงข้อมูลแบบระดับบิตได้ด้วย ทำให้สามารถควบคุมหรือตรวจสอบสถานะบิตได้ง่าย ส่งผลให้สามารถเขียนโปรแกรมได้ง่ายขึ้น
- มีไทเมอร์/เคาน์เตอร์ (Timer/Counter) ขนาด 16 บิตจำนวน 2 ตัว โดยเฉพาะหมายเลข 8032 หรือ 8052 จะมีไทเมอร์/เคาน์เตอร์จำนวน 3 ตัว
- สามารถทำการอินเตอร์รัพท์ได้จากแหล่งกำเนิด 5 แหล่ง โดยเฉพาะหมายเลข 8032 และ 8052 จะทำการอินเตอร์รัพท์ได้จากแหล่งกำเนิด 6 แหล่งพร้อมทั้งยังสามรถจัดลำดับความสำคัญของการอินเตอร์รัพท์ได้ 2 ระดับ
- มีพอร์ตสื่อสารอนุกรมภายในตัวเอง ซึ่งการทำงานจะเป็นแบบฟูลดูเพลกซ์ (Full Duplex)
- มีคำสั่งในการคำนวณทางคณิตศาสตร์และทางตรรกศาสตร์
- คำสั่งส่วนใหญ่จะใช้เวลาในการประมวลเพียง 1 ไมโครวินาที เมื่อใช้คริสตอลออสซิลเลเตอร์ความถี่ 12 เมกะเฮิร์ตซ์
- ต้องการแหล่งจ่ายไฟกระแสตรงแรงดัน 5 โวลต์

2.1.2 The AT89S8253 microcontroller

ไมโครคอนโทรเลอร์ AT89S8253 จะมีคุณสมบัติและสถาปัตยกรรมของขาใช้งานพื้นฐานดังรูปที่ 2.1 และ 2.2 โดยมีรายละเอียดขั้นต้นดังนี้

1.ใช้ร่วมกับตระกูล 8051 ได้ Compatible with 8051 family.

2.หน่วยความจำแฟลชสำหรับโปรแกรมการจัดเก็บ 12 kb

2.1 มีการโหลดโปรแกรมผ่านทางระบบ SPI (Serial Peripheral Interface)

2.2โปรแกรมอาจจะโหลด/ลบ ถึง 1000ครั้ง

3. EEPROM มีหน่วยความจำ 2 kb

4. แรงดันไฟฟ้า : 2.7 – 5.5 V

5. ความถี่สัญญาณนาฬิกาการดำเนินงาน : 0 – 24 MHz

6. 256 ไบต์ของหน่วยความจำภายในสำหรับตัวแปรจัดเก็บ

7. 32 ขาเข้า / ขาส่งออก

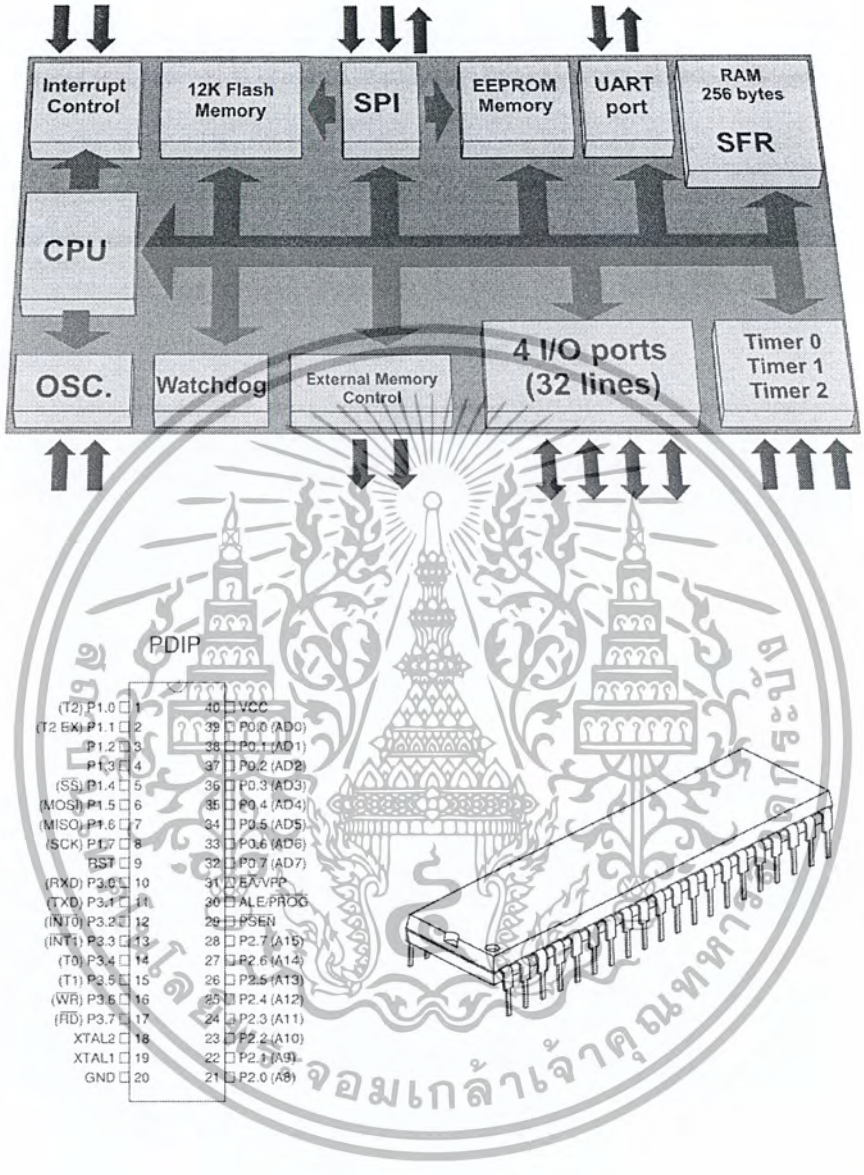
8. 16 - bit timers / เคาน์เตอร์

9. มี 9 interrupt souces

10. การสื่อสารอนุกรมโปรแกรม UART

11. ตั้งเวลาปิดโปรแกรมได้จ้องจับผิด

12. มีสามระดับล็อคหน่วยความจำโปรแกรม



รูปที่ 2.1 การจัดขามาตรฐานของไมโครคอนโทรเลอร์ AT89S8253 [6]

Pinout Description

ตารางที่ 2.1 คุณสมบัติของขาต่างๆใน ไมโครคอนโทรลเลอร์

PORT PIN	ALTERNATE FUNCTION
P1.0	T2 (Timer 2 input)
P1.1	T2EX (Timer 2 control input)
P1.4	SS (SPI system control input)
P1.5	MOSI (SPI system I/O)
P1.6	MISO (SPI system I/O)
P1.7	SCK (SPI system clock signal)
P3.0	RXD (serial input)
P3.1	TXD (serial output)
P3.2	INT0 (external interrupt 0)
P3.3	INT1 (external interrupt 1)
P3.4	T0 (Timer 0 external input)
P3.5	T1 (Timer 1 external input)
P3.6	WR (External data memory write signal)

ขา Vcc ใช้สำหรับต่อไฟเลี้ยง +5v

ขา GND เป็นขากราวด์สำหรับต่อกับกราวด์ของระบบ

ขาพอร์ท 0 (P0.0 - P0.7) มีขา 8 ขา สามรถกำหนดให้เป็นได้ทั้งอินพุตและเอาท์พุต สำหรับใช้งานทั่วไป ถ้าหากต้องการกำหนดให้ขาพอร์ท 0 ขาใดขาหนึ่งเป็นอินพุท สามารถทำได้โดยการเขียนข้อมูล "1" ไปยังแต่ละบิตของพอร์ทที่ต้องการติดต่อด้วย ส่งผลให้ขาพอร์ทนั้นมีสภานะปล่อยลอย (float) จึงมีอินพุทอิมพิแดนซ์สูง สามารถใช้งานเป็นขาพอร์ทอินพุทได้ นอกจากนั้นขาพอร์ทนี้ยังถูกใช้งานในการติดต่อกับขาแอดเดรสไบต์ต่ำของหน่วยความจำภายนอก (A0-7) และขาข้อมูล (D0-D7) โดยใช้กระบวนการมัลติเพล็กซ์เข้าช่วย เพื่อสลับการทำงานเป็นได้ทั้งขาติดต่อแอดเดรส และข้อมูล

ขาพอร์ท 1 (P1.0-P1.7) มีขา 8ขา แต่ละขาสามารถกำหนดให้เป็นทั้งอินพุทและเอาท์พุท สำหรับใช้งานทั่วไป ถ้าหากต้องการกำหนดให้ขาพอร์ทใดเป็นอินพุท สามารถทำได้โดยการเขียนข้อมูล "1" ไปยังแต่ละบิตของพอร์ทที่ต้องการติดต่อด้วย

ขาพอร์ท 2 (P2.0-P2.7) มีขา 8ขา แต่ละขาสามารถกำหนดให้เป็นทั้งอินพุทและเอาท์พุท สำหรับใช้งานทั่วไป ถ้าหากต้องการกำหนดให้ขาพอร์ทใดเป็นอินพุท สามารถทำได้โดยการเขียนข้อมูล "1" ไปยังแต่ละบิตของพอร์ทที่ต้องการติดต่อด้วย ส่งผลให้ขาพอร์ทนั้นมีสถานะปล่อยลอย (float) จึงมีอินพุทอิมพิแดนซ์สูง สามรถใช้งานเป็นขาพอร์ทอินพุทได้ นอกจากนั้นขาพอร์ทนี้ยังถูกใช้งานในการติดต่อกับขาแอดเดรสไบต์สูงของหน่วยความจำภายนอก (A8-A15)

ขาพอร์ท 3 (P3.0-P3.7) มีขา 8ขา แต่ละขาสามารถกำหนดให้เป็นทั้งอินพุทและเอาท์พุท สำหรับใช้งานทั่วไป ถ้าหากต้องการกำหนดให้ขาพอร์ทใดเป็นอินพุท สามารถทำได้โดยการเขียนข้อมูล "1"ไปยังแต่ละบิตของพอร์ทที่ต้องการติดต่อด้วย ส่งผลให้ขาพอร์ทนั้นมีสถานะปล่อยลอย (float) จึงมีอินพุทอิมพิแดนซ์สูง สามารถใช้งานเป็นขาพอร์ทอินพุทได้ นอกจากนั้นขาพอร์ท 3 นี้ยังเป็นขาที่มีหน้าที่การใช้งานเป็นพิเศษ มีรายละเอียดดังนี้

P 3.0 ใช้เป็นขาอินพุทสำหรับรับข้อมูลจากการสื่อสารแบบอนุกรม หรือขา RxD

P3.1 ใช้เป็นขาอินพุทสำหรับรับข้อมูลจากการสื่อสารแบบอนุกรม หรือขา TxD

P3.2 ใช้เป็นขาอินพุทรับสัญญาณอินเตอร์รัพท์จากภายนอกช่อง 0 หรือขา INT 0

P3.3 ใช้เป็นขาอินพุทรับสัญญาณอินเตอร์รัพท์จากภายนอกช่อง 1 หรือขา INT 1

P3.4 ใช้เป็นขาอินพุตสำหรับรับสัญญาณไทเมอร์จากภายนอกช่อง 0 หรือขา T0

P3.5 ใช้เป็นขาอินพุตสำหรับรับสัญญาณไทเมอร์จากภายนอกช่อง 1 หรือขา T1

P3.6 ใช้เป็นขาสัญญาณ WR ในกรณีที่ใช้เชื่อมต่อกับหน่วยความจำภายนอก

P3.7 ใช้เป็นขาสัญญาณ RD ในกรณีที่ใช้เชื่อมต่อกับหน่วยความจำภายนอก

ขารีเซต (Reset) ใช้ในการรีเซตการทำงานของไมโครคอนโทรเลอร์ โดยใช้การป้อนสัญญาณเพื่อรีเซตสถานะที่ขานี้ต้องอยู่ในระดับรีเซตอย่างน้อย 2 แมชชีน ไซเคิล โยที่วงจรกำเนิดสัญญาณนาฬิกายังคงทำงานต่อเนื่องไปอย่างเป็นปกติ

ขา ALE/PROG (Address Latch Enable/Program pulse input) เป็นขาที่ใช้ในการควบคุมการแลตซ์ของขาพอร์ท 0 เมื่อมีการใช้งานหน่วยความจำภายนอก นอกจากนั้นขานี้ยังเป็นชาใช้สำหรับพัลส์ของการโปรแกรมสำหรับโปรแกรมข้อมูลลงใน ไมโครคอนโทรเลอร์ MCS-51 ในรุ่นนี้มีหน่วยความจำโปรแกรมเป็นแบบอีพรอม

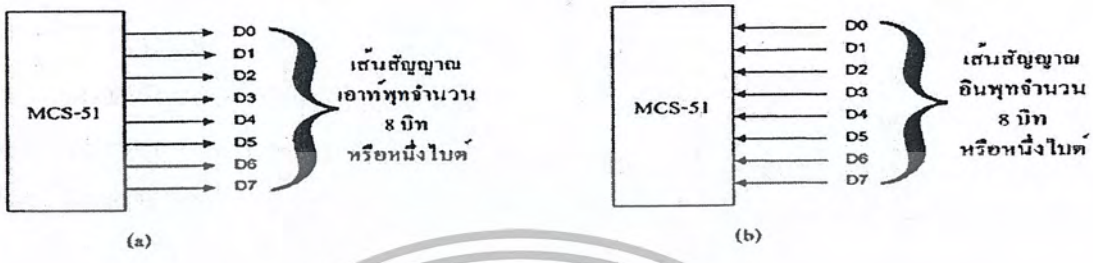
ขา PSEN (Program Store Enable) ขานี้ใช้ในการส่งสัญญาณเพื่อร้องขอติดต่อกับหน่วยความจำโปรแกรมภายนอก เมื่อไมโครคอนโทรเลอร์ต้องการอ่านข้อมูลจากหน่วยความจำโปรแกรมภายนอกตัวไมโครคอนโทรเลอร์จะส่งสัญญาณออกมาที่ขานี้ 2 ครั้ง ในแต่ละแมชชีนไซเคิล แต่ถ้าหากติดต่อกับหน่วยความจำข้อมูลภายนอกขานี้จะไม่มีการส่งสัญญาณใดๆออกมา

ขา EA/Vpp (External Access enable/Program voltage input) ใช้สำหรับเลือกการติดต่อหน่วยความจำโปรแกรมภายนอกหรือภายในตัวไมโครคอนโทรเลอร์ ถ้าหากขานี่เป็น "0" เป็นการเลือกให้ไมโครคอนโทรเลอร์ติดต่อกับหน่วยความจำโปรแกรมภายนอก แต่ถ้าหากขานี้เป็น "1" เป็นการเลือกให้ไมโครคอนโทรเลอร์ติดต่อกับหน่วยความจำภายในตัวไมโครคอนโทรเลอร์ นอกจากนี้ที่ขานี้ยังใช้เป็นขาอินพุทสำหรับแรงดันไฟสูง สำหรับการโปรแกรมหน่วยความจำภายในไมโคนคอนโทรลเลอร์ สำหรับไมโครคอนโทรเลอร์ MCS-51 แบบแฟลชต้องการแบ่งแรงดันสำหรับการโปรแกรม +5V

ขา XTAL1 และ XTAL2 เป็นขาสำหรับต่อคริสตัลเพื่อสร้างสัญญาณนาฬิกาในการกำหนดจังหวะการทำงานของไมโครคอนโทรเลอร์

2.1.3 พอร์ทอินพุทและพอร์ทเอาท์พุท

พอร์ท คือ แอดเดรสหนึ่งที่ได้รับการกำหนดไว้เพื่อการโอนหรือการย้ายข้อมูลระหว่างไมโครคอนโทลเลอร์กับอุปกรณ์ภายนอก การกำหนดประเภทของการติดตั้งขึ้นอยู่กับทิศทางของการไหลของข้อมูลเมื่อพิจารณาจากไมโครคอนโทรเลอร์เป็นหลัก จากรูปที่ 2.7(a) เป็นการใช้ไมโครคอนโทรเลอร์เป็นเอาท์พุตพอร์ท และ จากรูป 2.7(b) เป็นการใช้ไมโครคอนโทรเลอร์เป็นอินพุตพอร์ท



รูปที่ 2.2 การใช้ไมโครคอนโทรลเลอร์เป็นอินพุทและเอาท์พุทพอร์ท [5]

2.1.3.1 การใช้งานพอร์ทเป็นอินพุท

การใช้งานพอร์ทเป็นการอินพุทข้อมูลจะต้องเริ่มต้นด้วยการส่งข้อมูลที่มีค่าเป็น 1 ออกมาทางบิตของพอร์ทนั้นก่อนเป็นอันดับแรก เพื่อหยุดการทำงานของทรานซิสเตอร์ที่ทำหน้าที่ขับสัญญาณเอาท์พุทของบิตนั้น ทำให้ขาสัญญาณของบิตถูกต่อเข้ากับตัวต้านทานซึ่งทำหน้าที่ pull - up ภายในซึ่งมีผลทำให้บิตนั้นของพอร์ท 1 ,2 และ 3 เป็นสภาวะลอจิกสูง ตัวต้านทานนี้มีค่าประมาณ 50k ohm ซึ่งเป็นค่าที่สูงมาก และทำให้อุปกรณ์ภายนอกสามารถขับสัญญาณของพอร์ทเหล่านี้เป็นลอจิกต่ำได้ง่าย สำหรับบิตของพอร์ท 0 นั้นแม้ว่าจะมีหลักการทำงานที่คล้ายคลึงกันกับบิตของพอร์ทอื่นๆ แต่เนื่องจากไม่มีตัวต้านทานซึ่งทำหน้าที่ pull-up ภายในไว้ ทำให้เมื่อทรานซิสเตอร์ที่ทำหน้าที่ขับสัญญาณเอาท์พุตนั้นจะหยุดการทำงาน ซึ่งก็จะเป็นผลให้สัญญาณนี้อยู่ในสภาวะอิมพิแดนซ์สูงแทน

2.1.3.2 การใช้งานพอร์ตเป็นเอาท์พุต

เมื่อมีการส่งข้อมูลที่มีค่าเป็น 0 ให้กับแต่ละบิตของพอร์ททุกพอร์ท ข้อมูลนี้จะถูกส่งให้กับฟลิปฟลอปซึ่งจะค้างค่านี้ไว้ และมีผลให้ทรานซิสเตอร์ที่ทำหน้าที่ขับสัญญาณเอาท์พุทนั้นทำงานดังนั้นขอสัญญาณก็จะมีสถานะเป็นลอจิกต่ำด้วย

ส่วนการส่งข้อมูลที่มีค่าเป็น 1 ออกมานั้น ในกรณีที่เป็นการทำงานในแต่ละบิตของพอร์ท 1 , 2 หรือ 3 จะทำให้ทรานซิสเตอร์ที่ทำหน้าที่ขับสัญญาณเอาท์พุทนั้นหยุดทำงาน มีผลให้ขาขิงสัญญาณเป็นลอจิกสูงด้วยตัวต้านทานที่ Pull-up อยู่ภายใน แต่สำหรับการใช้งานในแต่ละบิตทางพอร์ท 0 นั้นจะมีผลแตกต่างออกไป โดยขาสัญญาณมีสภาวะอิมพิแดนซ์สูงแทน เนื่องจากไม่มีตัวต้านทานภายในเชื่อมต่ออยู่นั่นเอง ดังนั้นการใช้งานพอร์ท 0 เป็นการนำข้อมูลออกทางเอาท์พุท จึงจำเป็นต้องใช้ตัวต้านทานภายนอก Pull-up สัญญาณไว้กับลอจิกสูงแทน

2.1.4 การเชื่อมต่อไมโครคอนโทรลเลอร์ กับคอมพิวเตอร์ผ่านพอร์ตอนุกรม

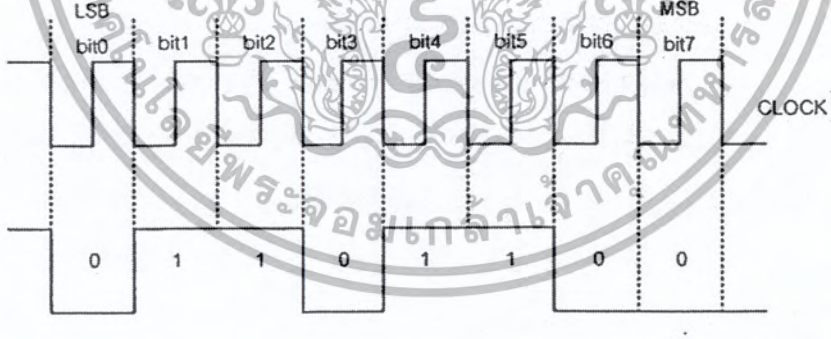
การเคลื่อนย้ายข้อมูลจากคอมพิวเตอร์ไปยังอุปกรณ์ต่อพ่วงอื่นๆ หรือคอมพิวเตอร์ด้วยกันนั้นมี 2 วิธีคือ กรรับส่งข้อมูลแบบอนุกรม การรับส่งข้อมูลแบบขนานจะเป็นการรับส่งข้อมูลคราวละ 4 บิต หรือ 8 บิต ในเวลาเดียวกันซึ่งจะทำให้การรับและส่งข้อมูลทำได้ที่ความเร็วสูง ซึ่งหมายความว่าจำนวนสายที่ใช้ในการส่งจะต้องมีมากเท่ากับจำนวนบิตของข้อมูลที่จะส่งด้วย ซึ่งอาจจะต้องใช้สายมากเป็น 2 เท่าของจำนวนบิตข้อมูลที่จะส่งก็ได้ ซึ่งก็ไม่ได้เป็นปัญหาในเรื่องของราคาของสายที่ใช้ในการเชื่อมต่อแบบขนานมักจะมีราคาแพง

ในขณะที่การรับส่งข้อมูลแบบอนุกรมเป็นการรับส่งข้อมูลอย่างละ 1 บิต แต่ก็สามารถรับส่งข้อมูลได้คราวละหลายๆบิตได้ หากแต่จะต้องมีการตกลงกันระหว่างตัวส่งและตัวรับว่า จะรับส่งข้อมูลคราวละกี่บิต ตัวรับจะต้องรอข้อมูลมาให้ครบทุกบิตเสียก่อนจึงจะทำการประมวลผล ส่งผลให้

การสื่อสารข้อมูลแบบอนุกรมอาจมีความเร็วต่ำกว่าขนาน อย่างไรก็ตาม การรับส่งข้อมูลแบบอนุกรมนั้นสามารถใช้สานสัญญาณที่มีความยาวมากกว่าแบบขนาน ทำให้ระยะทางในการสื่อสารข้อมูลแบบอนุกรมสามารถทำได้มากกว่า

2.1.4.1 การสื่อสารแบบอนุกรม

การสื่อสารแบบอนุกรมนั้นจะแบ่งออกได้เป็น 2 แบบ คือการสื่อสารอนุกรมแบบซิงโครนัสและการสื่อสารอนุกรมแบบอะซิงโครนัส การสื่อสารแบบซิงโครนัสจะมีสัญญาณนาฬิการ่วมอยู่กับการรับ และ ส่งสัญญาณด้วย ตัวอย่างการส่งข้อมูลแบบซิงโครนัสคือคีย์บอร์ดของคอมพิวเตอร์ ซึ่งสายเส้นหนึ่งจะเป็นสายของสัญญาณนาฬิกา ส่วนอีกเส้นหนึ่งจะเป็นสายของข้อมูล ดังนั้นการติดต่อแบบซิงโครนัสนี้จะต้องใช้สายในการเชื่อมต่ออย่างน้อยที่สุด 3 เส้น คือ สัญญาณนาฬิกา , ข้อมูลและกราวด์



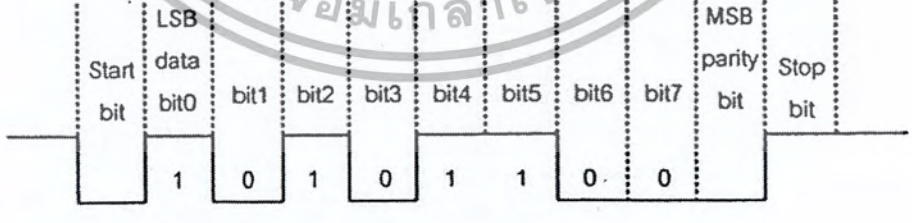
รูปที่ 2.3 รูปแบบอย่างง่ายที่สุดของข้อมูลอนุกรม [5]

2.1.4.2 การสื่อสารข้อมูลแบบอะซิงโครนัส

การสื่อสารข้อมูลแบบอะซิงโครนัส คือการรับส่งข้อมูลไปในสายสัญญาณ โดยไม่จำเป็นต้องส่งสัญญาณนาฬิการ่วมด้วยเหมือนกันกับการรับส่งข้อมูลแบบซิงโครนัส แต่จะใช้การกำหนดค่าสัญญาณนาฬิกาทั้งภาครับและภาคส่งให้มีค่าเท่ากัน ซึ่งเรียกสัญญาณนาฬิกาที่ใช้มนการกำหนดค่าให้ภาครับและภาคส่งนี้ว่า อัตราการถ่ายเทข้อมูล หรืออัตราบอด (baudrate) ที่มีหน่วยเป็นบิตต่อวินาที (bit per second : bps)

รูปแบบของข้อมูลที่ใช้ในการรับส่งแบบอะซิงโครนัสจะประกอบด้วย 4 ส่วนด้วยกัน คือ

1. บิตเริ่มต้น (Start Bit) ซึ่งจะมีขนาด 1 บิต
2. บิตข้อมูลอนุกรมจะมีขนาด 5, 6, 7 หรือ 8 บิต
3. บิตตรวจสอบพาริตี้ (Parity Bit) จะมีขนาด 1 บิตหรือไม่มี
4. บิตปิดท้าย (Stop Bit) จะมีขนาด1 , 1.5 หรือ 2 บิต



รูปที่ 2.4 รูปแบบอย่างง่ายที่สุดของข้อมูลอนุกรมแบบอะซิงโครนัส [5]

รูปที่ 2.4 แสดงรูปแบบของข้อมูลอนุกรมแบบอะซิงโครนัส ซึ่งเมื่อไม่มีข้อมูลที่จะส่ง ขาดาต้าจะมีสถานะลอจิก 1 ซึ่งจะเรียกสถานะนี้ว่าสถานะว่าง (idle stage) การเริ่มต้นส่งข้อมูลจะเริ่มจากการให้ขอดาต้ามีลอจิก 0 ด้วยช่วงเวลา 1 บิตซึ่งจะเรียกบิตนี้ว่าบิตเริ่มต้น จากนั้นบิตข้อมูลจะถูกส่งออกไปโดยเริ่มต้อนที่มีนัยสำคัญต่ำสุด (LSB) ก่อน ซึ่งข้อมูลในไบต์ที่ส่งอาจจะมีจำนวนบิต 5,6,7 หรือ 8 ก็ได้ จากนั้นจะตามด้วยบิตพาริตี้ ซึ่งใช้เพื่อตรวจสอบความผิดพลาดที่เกิดขึ้นจากการส่งข้อมูลบิตสุดท้ายที่จะส่งคือบิตปิดท้าย ซึ่งจะให้ขาดาต้ามีสถานะลอจิก 1 อีกครั้งด้วยระยะเวลาอย่างน้อย 1 บิต , 1.5บิต หรือ 2 บิตเพื่อเป็นการแสดงว่าสิ้นสุดข้อมูลแล้ว

อุปกรณ์พิเศษที่ได้รับการออกแบบมาสำหรับการรับและส่งข้อมูลแบบอะซิงโครนัส เรียกว่า Universal Asynchronous Receiver / Transmitter หรือ UART อัตราความเร็วในการรับและส่งข้อมูลของการรับส่งข้อมูลแบบอะซิงโครนัส คือค่าอัตราบอด ซึ่งก็คือค่าอัตราการเข้ารหัสที่ใช้ในการรับและส่งข้อมูล อัตราบอดมาตรฐานที่ใช้สำหรับพอร์ทอนุกรม RS-232 ได้แก่ 110 ,150 ,300, 600 ,1200 ,2400 ,4800 ,9600 ,19200 บิตต่อวินาทีและมีค่าเพิ่มมากขึ้นตามเทคโนดลยีของคอมพิวเตอร์ ซึ่งการรับส่งแบบอนุกรมโดยไม่ผ่านโมเด็มอาจจะกำหนดค่าอัตราบอดได้สูงถึง 115200 บิตต่อวินาที เนื่องจากอัตราบอดคือจำนวนบิตของข้อมูลที่สามารถถ่ายทอดได้ภายใน 1 วินาที ยกตัวอย่างข้อมูลอนุกรมถูกส่งในลักษณะ 8 บิต ไม่มีการตรวจสอบพาริตี้ มีบิตเริ่มต้น 1 บิตและบิตปิดท้าย 1 บิตความยาวของข้อมูลที่รับส่งนี้เท่ากับ 10 บิต ถ้าใช้อัตราบอดในการส่งข้อมูลเท่ากับ 9600 บิตต่อวินาที ก็จะสามารถรับส่งข้อมูลได้ด้วยความเร็ว 960 ไบต์ต่อวินาที และถ้ามีการใช้พาริตี้ความเร็วในการรับส่งข้อมูลจะเหลือเป็น 872 ไบต์ต่อวินาที

การตรวจสอบพาริตี้สามารถกำหนดให้เป็นแบบคี่ (odd) , แบบคู่ (even) หรืออาจไม่มีการตรวจสอบพาริตี้ก็ได้ การตรวจสอบพาริตี้เป็นการตรวจสอบจำนวนรวมของบิตที่เป็นลอจิก 1

ภายในข้อมูลที่ส่งไป 1 ไบต์ว่ามีจำนวนรวมเป็นเลขคู่หรือเลขคี่โดยต้องรวมบิตพาริตี้เข้าไปด้วย ยกตัวอย่างข้อมูลที่จะทำการส่งมีขนาด 8 บิตและมีค่าเท่ากับ 99 ฐานสิบหก หรือ 10011001 ฐานสอง จะเห็นว่าข้อมูลในไบต์นี้มีลอจิกเป็น 1 จำนวน 4ตัวซึ่งเป็นเลขคู่ ดังนั้นถ้ากำหนดค่าพาริตี้เป็นคู่ ค่าในบิตพาริตี้มีจำนวนบิตที่เป็นลอจิก 1 มีจำนวนรวมกันเป็นเลขคี่ ตารางที่ 2.2 แสดงตัวอย่างของบิตพาริตี้ในการรับส่งข้อมูลอนุกรม

ตารางที่ 2.2 บิตพาริตี้ของข้อมูล

ข้อมูล	บิตพาริตี้คู่	บิตพาริตี้คี่
00000000	0	1
00000001	1	0
00000010	1	0
00000011	0	1
00000100	1	0
11111110	1	0
11111111	0	1

บิตพาริตี้ถูกสร้างขึ้นจากภาคส่งข้อมูลของ UART ซึ่งทางภาครับจะต้องทำการกำหนดคุณสมบัติจากการตรวจสอบพาริตี้ให้ตรงกันว่าจะตรวจสอบพาริตี้คี่หรือพาริตี้คู่ จากนั้นภาครับของ UART จะทำการตรวจสอบค่าพาริตี้ที่เกิดขึ้นว่าเป็นคู่หรือคี่ โดยการนับจำนวนลอจิก 1 ทั้งหมดรวมทั้งบิตพาริตี้ด้วย ถ้ากำหนดพาริตี้ไว้เป็นคู่แต่อ่านค่าตัวเลขในการนับออกมาได้ตัวเลขเป็นคี่ ทางภาครับจะแสดงข้อผิดพลาดออกมาให้ผู้ใช้ทราบ นับเป็นการตรวจสอบความผิดพลาดที่เกิดขึ้นในการ

ถ่ายทอดข้อมูลที่ง่ายที่สุด แต่จะเชื่อถือได้เมื่อมีบิตข้อมูลที่ทำการส่งผิดพลาดแค่เพียงบิตเดียวเท่านั้น ถ้าข้อมูลที่ทำการส่งมีบิตผิดพลาดมากกว่า 1 บิต การตรวจสอบด้วยวิธีนี้จะไม่ได้ผล สำหรับการตั้งพาริตี้บิตเป็น NONE นั้งทั้งภาครับและภาคส่งจะไม่มีการตรวจสอบพาริตี้

คอมพิวเตอร์ในรุ่น AT เกือบทั้งหมดจะใช้ UART หมายเลข 16450 และ 16550 ส่วนคอมพิวเตอร์ในรุ่น XT ใช้ UART หมายเลข 8250 UART ชิปเหล่านี้นี้มีระดับแรงดันเป็นแบบทีทีแอล (0 และ +5 V) แต่เพื่อให้มีแรงดันเป็นไปตามมาตรฐาน RS-232 และเพื่อให้การรับส่งข้อมูลสามารถทำได้ที่ระยะทางไกลมากขึ้นระดับแรงดันทีทีแอลจะถูกแปลงเป็นระดับแรงดันที่สูงขึ้น โดยลอจิก 0 มีทั้งระดับแรงดัน +3V ถึง +12V ในขณะที่ลอจิก 1 มีระดับแรงดัน – 3V ถึง -12V

2.2 การสื่อสารข้อมูลอนุกรม

การสื่อสารข้อมูลอนุกรมเป็นการรับหรือส่งข้อมูลในลักษณะกลุ่มของบิต คราวละหนึ่งบิต เรียงลำดับเรื่อยไปจนสิ้นสุดการสื่อสารแบบนี้จะมีข้อแตกต่างจากการสื่อสารแบบขนานเป็นอย่างมาก เนื่องจากข้อมูลที่มีการโอนย้ายมาพร้อมกัน จึงมีความจำเป็นต้องใช้จำนวนเส้นสัญญาณมากขึ้นตามจำนวนบิตของข้อมูลด้วย ในขณะที่การสื่อสารแบบอนุกรมนั้นต้องการเส้นสัญญาณเพียงสองหรือสามเส้นเท่านั้น ดังนั้น ในการสื่อสารแบบขนานจึงไม่เหมาะสมในการสื่อสารกับอุปกรณ์ภายนอกในระยะไกลๆ เพราะจะทำให้สิ้นเปลืองค่าใช้จ่ายมาก

ความเร็วของการสื่อสารแบบอนุกรม

เนื่องจากการสื่อสารแบบอนุกรมเป็นการรับส่งในลักษณะกลุ่มของบิตข้อมูล (Bit Stream) ดังนั้นจึงต้องให้ความสนใจในการพิจารณา ถึงเรื่องอัตราความเร็วในการ รับ-ส่ง บิตเหล่านี้เป็นอันดับแรก โดยทั่วไปมักจะระบุกันในหน่วยของจำนวนบิตข้อมูลภายในเวลาหนึ่งวินาที เรียกว่า อัตราบอด ตามค่ามาตรฐานเหล่านี้ได้แก่ 110,150,300,1200,2400,4800,9600,19200 บอด ข้อมูลทั้ง 8 บิตนี้หากถูกส่งออกมาด้วยอัตราบอด 2400 บอด จะใช้เวลาในการส่งข้อมูลหนึ่งบิตมีค่าเท่ากับ 1/2400 หรือ 416μS และเวลาในการส่งข้อมูลทั้ง 8 บิตมีค่าเท่ากับ (8*416) หรือ 3328 μS

รูปแบบของการส่งข้อมูลอนุกรม

การสื่อสารอนุกรมแบบอะซิงโครนัสจะใช้การแปลงข้อมูลขนานให้เป็นอนุกรมแล้วเพิ่มบิตบางอย่างรวมไปกับการส่งข้อมูลจริงซึ่งได้แก่

บิตเริ่มต้น(Start Bit)

บิตเริ่มต้นมีหน้าที่สำหรับการบ่งบอกให้ทราบถึงตำแหน่งจุดเริ่มต้นก่อนบิตข้อมูล ตามปกติแล้วค่าของบิตเริ่มต้นจะเป็นระดับลอจิกต่ำ

บิตแสดงภาวะความเป็นเลขคู่หรือเลขคี่ (Parity Bit)

บิตนี้มีหน้าที่เพื่อการตรวจสอบความถูกต้องของข้อมูลโดยทั่วไปมักเรียกว่า บิตพาริตี้และจะนำไปต่อท้ายบิตข้อมูล ค่าของบิตนี้จะขึ้นอยู่กับจำนวนค่าของบิตที่เป็น 1 ซึ่งจะเป็นได้สองลักษณะคือ พาริตี้คู่ (Even Parity) หรือพาริตี้คี่ (Odd Parity)ตัวอย่างเช่นระบบที่ติดต่อกัน โดยระบุว่าใช้พาริตี้คี่ ทางด้านส่งจะนำข้อมูลที่ส่งมาพิจารณา จำนวนบิงบิตทีมีค่า 1 เป็นเลขจำนวนคู่อยู่แล้ว ค่าของบิตพาริตี้จะมีค่าเป็น 0 แต่หากว่าจำนวนบิตที่ค่าเป็น 1 เป็นจำนวนเลขคี่ ค่าของบิตพาริตี้จะมีค่าเป็น 1 การ

พิจารณาทางด้านรับเป็นการตรวจสอบจำนวนบิตที่มีค่าเป็น 1 ของข้อมูลที่ได้รับมาทั้งหมดรวมทั้งบิตพาริตี้ ถ้ามีค่าเป็นเลขจำนวนคู่ แสดงว่าข้อมูลที่รับเข้ามานี้ถูกต้อง หากไม่เป็นเลขจำนวนคู่แสดงว่าเกิดการผิดพลาดของข้อมูลขึ้น เป็นต้น

บิตสุดท้าย (Stop Bit)

เป็นบิตที่เพิ่มขึ้นเพื่อระบุถึงขอบเขตการสิ้นสุดของกลุ่มบิตสุดท้ายนี้สามารถโปรแกรมได้คือ 1 บิต 1 ½ บิตและ 2 บิต ดังนั้นกรณีของการส่งข้อมูล8 บิตหากข้อมูลถูกส่งออกไปด้วยอัตราเร็ว 2400 บอด เวลาโดยรวมในการส่งข้อมูลหนึ่งไบต์จะมีค่าเป็น (12*416) µS หรือ 4.99 mS การส่งข้อมูลอนุกรมของ 8051

พอร์ตอนุกรมของ 8051 มีโครงสร้างการทำงานในแบบที่เรียกว่า ฟูลดูเพล็กซ์ (Full Duplex) ในการรับและส่งข้อมูลอนุกรมได้ในเวลาเดียวกัน โดยทางด้านวงจรของตัวส่ง (Transmitter) ประกอบด้วยข้อมูลออกไปยังพอร์ตอนุกรมทางขาสัญญาณ TxD (พอร์ต 3.1) ส่วนวงจรด้านตัวรับ (Receiver) ประกอบด้วย SUBF เช่นเดียวกับสัญญาณข้อมูลอนุกรมที่รับเข้ามาทางขาสัญญาณ RxD (พอร์ต 3.0)พอร์ตอนุกรมของ 8051สามารถโปรแกรมการทำงานได้หลายโหมดด้วยกันโดยเลือกที่บิต SM0 และ SM1 ซึ่งอยู่ในรีจิสเตอร์ควบคุม SCON การทำงานทั้ง 4 โหมดของพอร์ตอนุกรมมีดังนี้

โหมด 0 : ใช้รับส่งข้อมูล 8 บิต โดยการส่งจะเลื่อนออกทีละบิต โดยส่งบิต 0 ออกไปก่อนทางขา RxD และไม่มีการส่ง start bit แต่จะส่ง shift clock ทางขา TxD ความเร็ว 1/12 เท่าของ CPU CLOCK

โหมด 1 : ใช้สำหรับการเชื่อมต่ออนุกรมแบบ UART (Universal Asynchronous Reciever/Transmitter)โดยส่งแบบ 10 บิต ข้อมูล 8 บิต 1 start bit และ 1 stop bit และสามารถเปลี่ยนแปลงอัตราเร็วในการส่งข้อมูลได้ โดยขึ้นกับบิต SMOD ใน PCON และอัตราโอเวอร์โฟลว์ของ TIMER 1

โหมด 2 : ใช้สำหรับการเชื่อมต่ออนุกรมแบบ UART โดยการใช้กลุ่มข้อมูลแบบ 11 บิต และกำหนดอัตราความเร็วในการส่งข้อมูลเท่ากับ 1/32 และ 1/64 ของ CPU CLOCK โดยโปรแกรมที่บิต SMOD ใน PCON

โหมด 3 : ใช้สำหรับการเชื่อมต่ออนุกรมแบบ UART โดยการใช้กลุ่มข้อมูลแบบ 11 บิต และสามารถเปลี่ยนแปลงอัตราความเร็วในการส่งข้อมูลได้ โดยควบคุมที่บิต SMOD และอัตราโอเวอร์โฟลว์ของ Timer 1 นอกจากนี้ โหมด 2 และ 3 ยังมีการดำเนินการอีกแบบหนึ่ง โดยสามารถนำมาใช้ประโยชน์ในการสื่อสารข้อมูลแบบที่มีไมโครโปรเซสเซอร์หลายตัวทำงานร่วมกันได้ซึ่งมีชื่อเรียกว่า Multiprocessor Mode

ตารางที่ 2.3 การเลือกโหมดการทำงาน

SM 0	SM 1	โหมด	การทำงาน
0	0	0	ทำงานเป็น shift register อัตราเร็วในการรับหรือส่งข้อมูลเท่ากับ 1/12 ของความถี่ออสซิลเลเตอร์
0	1	1	8 bit UART อัตราเร็วในการรับหรือส่งข้อมูลกำหนดเองได้
1	0	2	9 bit UART อัตราเร็วในการรับหรือส่งข้อมูล 1/32 หรือ 1/64 ของความถี่ออสซิลเลเตอร์
1	1	3	9 bit UART อัตราเร็วในการรับหรือส่งข้อมูลกำหนดเองได้

SM 1 บิตเลือกการทำงานแบบ

1 : เลือก Multiprocessor Mode ใช้ได้กับโหมด 2,3

2 : เลือก Single Processer Mode ใช้ได้กับทุกโหมด

REN บิตควบคุมให้รับหรือไม่รับข้อมูล

1 : ให้รับข้อมูลได้

2 : ห้ามรับข้อมูล

TB 8 (Transmit Bit D8) ข้อมูลบิตที่ 9 ที่ส่งออกไปในโหมด 2,3 ให้ใส่ในบิต TB 8

RB 8 (Receiver Bit D8) ข้อมูลบิตที่ 9 ที่รับเข้ามาจะเก็บในบิตนี้ ข้อมูลบิตที่ 9 ก็คือ ค่าใน TB 8 ทางด้านส่งนั้นเอง

TI บิต TI จะเป็น 1 เมื่อสิ้นสุดการส่งข้อมูล 1 ไบต์

RI บิต RI จะเป็น 1 เมื่อรับข้อมูลเสร็จ 1 ไบต์

2.2.1 กระบวนการรับและส่งข้อมูลอนุกรมของ 8051

การส่งข้อมูลออกทางพอร์ตอนุกรมของ 8051 จะเริ่มต้นขึ้นภายหลังเมื่อมีการเขียนข้อมูลลงใน SBUF ข้อมูลนี้จะถูกเลื่อนทีละบิตและส่งสัญญาณออกไปภายนอกโดยอัตโนมัติ เมื่อข้อมูลเหล่านี้ได้ส่งออกครบถ้วนแล้วจะทำให้ค่าของแฟล็ก TI ให้เป็น 1 เพื่อแจ้งให้ทราบว่าขณะนี้ SBUF ว่างและพร้อมที่จะส่งข้อมูลไบต์ต่อไป

แล้วในกรณีที่ใช้เขียนข้อมูลใหม่ลงในรีจีสเตอร์ SBUF โดยไม่รอให้แฟล็ก TI มีค่าเป็น 1 ก่อนจะมีผลทำให้ข้อมูลที่ส่งไปผิดพลาดได้ สำหรับการรับข้อมูลจากพอร์ตอนุกรมจะต้องเริ่มต้นโดยการกำหนดเซ็ทค่าดังนี้ REN (Receiver Enable) ให้มีค่าเป็น 1 ก่อน หลังจากนั้นเมื่อข้อมูลภายนอกถูก

ส่งเข้ามายัง 8051 ทีละบิตจนครบ และเมื่อบิตสุดท้ายถูกเลื่อนเข้ามาเรียบร้อยแล้วข้อมูลนั้นจะถูกย้ายมาเก็บไว้ยังรีจิสเตอร์ SBUF และแฟล็ก RI ก็จะมีค่าเป็น 1 หลังจากนั้นก็จะเกิดการอินเตอร์รัพต์ขึ้น

2.2.2 การสื่อสารพอร์ตอนุกรม RS-232

เป็น IC เป็นวงจรที่ทำหน้าที่เปลี่ยนแรงดันที่เข้ามาจาก Serial Port ไปเป็นแรงดันตามมาตรฐานของ RS-232 โดยเปลี่ยนระดับแรงดัน TTL เพื่อให้ใช้ได้กับไมโครคอนโทรเลอร์

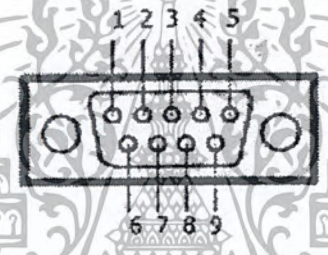
ลักษณะของการส่งข้อมูลอนุกรมนั้น ข้อมูลจะส่งออกมาทีละบิตจากตัวส่งไปตัวรับข้อมูลชิ่งสัญญาณในการส่งข้อมูลอาจใช้เพียง 1 หรือ 2ช่องสัญญาณเท่านั้น ทำให้ค่าใช้จ่ายในระบบสื่อสารจะถูกกว่าแบบขนาน แต่อัตราการรับ-ส่งข้อมูลจะช้ากว่าแบบขนาน ในการส่งข้อมูลแบบอนุกรมข้อมูลที่ต้องการส่งจะอยู่ในลักษณะเป็นไบต์จะทยอยส่งทีละบิต แล้วมารวมกันเป็นไบต์ ซึ่งทางตัวรับต้องตรวจสอบว่าบิตใดเป็นบิตเริ่มต้นหรือบิตสุดท้ายของข้อมูลการตรวจสอบนั้นจะขึ้นอยู่กับรูปแบบของรหัสของบิตข้อมูลที่ใช้ ซึ่งจะต้องมีมาตรฐานในการรับส่งข้อมูล โดยมาตรฐานที่นิยมใช้มากที่สุดคือ RS-232

มาตรฐาน RS-232

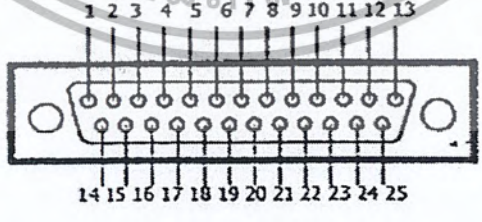
เพื่อที่จะทำให้อุปกรณ์จากผู้ผลิตต่างกันทำงานร่วมกันได้ มาตรฐานหลายชนิดจึงได้รับการออกแบบขึ้น มาตรฐานที่ใช้กันอย่างกว้างขวางที่สุดคือ RS-232 ซึ่งโดยปกติไมโครคอมพิวเตอร์จะมีพอร์ตเป็นอนุกรมอยู่ในตัวอยู่แล้วและจะทำหน้าที่รับส่งข้อมูลในแบบอนุกรม

ตามจุดประสงค์ของมาตรฐาน RS-232 นั้นเพื่อจะสามารถเชื่อมต่อกันระหว่างอุปกรณ์รับส่งปลายทาง (Data Terminal Equipment: DTE) เช่น พอร์ตของคอมพิวเตอร์หลักหรืออุปกรณ์ปลายทาง

กับอุปกรณ์สื่อสาร RS-232 เป็นข้อกำหนดของการอินเตอร์เฟสมาตรฐาน และสามารถใช้เพื่อจุดประสงค์อื่นต่างออกไป เช่น การสื่อสารแบบซิงโครนัส (synchronous communication) และรูปแบบการสื่อสารที่ต้องการสัญญาณนาฬิกา และสัญญาณกำหนดจังหวะเพิ่มเติมขึ้นมา ในความเป็นจริงแล้วเราสามารถทำให้มีการสนทนากันระหว่าง DTE และ DCE โดยการใช้สานสัญญาณเพียง 3 เส้นเท่านั้นคือใช้สาย TD สาย RD และสายกราวด์เท่านั้น



รูปที่ 2.5 คอนเน็กเตอร์ 9 ขาหรือแบบ DB9 (มองจากด้านหลังของคอมพิวเตอร์) [8]



รูปที่ 2.6 คอนเน็กเตอร์อนุกรม 25ขาหรือแบบ DB-25 (มองจากด้านหลังคอมพิวเตอร์) [8]

ตารางที่ 2.4 ข้อกำหนดของขาสัญญาณต่างๆ

หมายเลขขาสัญญาณ	ชื่อสายสัญญาณ
1	Protective Ground
2	Transmitted Data
3	Received Data
4	Request Data
5	Clear To Send
6	Data Set Ready
7	Signal Common
8	Received Line Signal Detect
9	Reserve For Testing
10	Reserve For Testing
11	Unassigned
12	Secondary Received Line Signal Detect
13	Secondary Clear To Send
14	Secondary Transmitted Data
15	Transmission Signal Element Timing
16	Secondary Received Data
17	Receiver Signal Element Timing
18	Unassigned
19	Secondary Request To Send
20	Data Terminal Ready
21	Signal Quality Detector
22	Ring Indicator
23	Data Signal Rate Detector

24	Transmitter Signal Element Timing
25	Unassigned

ตารางที่ 2.5 แสดงรายละเอียดของสัญญาณที่ต่อจาก DTE ไปยัง DCE โดยใช้คอนเน็กเตอร์แบบ DB-25

หมายเลขขาสัญญาณ	ชื่อสายสัญญาณ
1	Protective Ground
2	Transmitted Data
3	Received Data
4	Request To Send
5	Clear To Send
6	Data Set Ready
7	Signal Common
8	Data Carrier Detect
20	Data Terminal Ready
22	Ring Indicator
23	Data Signal Rate Detector

ตารางที่ 2.6 รายละเอียดการต่อคอนเน็กเตอร์แบบ DB9 มาตรฐาน RS-232

หมายเลขขาสัญญาณ	ชื่อสายสัญญาณ
1	Data Carrier Detect
2	Received Data
3	Transmitted Data
4	Data Terminal Ready
5	Signal Common
6	Data Set Ready
7	Request To Send
8	Clear To Send
9	Ring Indicator

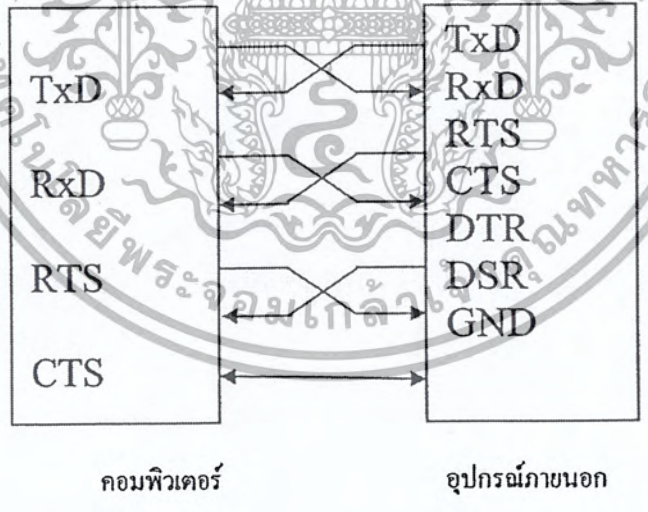
- ขา 1 (Protective Ground Circuit, AA) ขานี้จะต่อเข้ากับตัวถังของอุปกรณ์ และสามารถต่อเข้ากับกราวด์ภายนอกถ้าอุปกรณ์ใช้งานขานี้
- ขา 2 (Transmitted Data Circuit BA,TD) เป็นขาสัญญาณข้อมูลที่ออกจากอุปกรณ์ DTE กระแสบิตอนุกรมจากขานี้ คือข้อมูลที่ถูกถ่ายทอดไปโดยโมเด็ม หรือถูกถอดรหัสโดยอุปกรณ์ DTE ที่มี
- ขา 3 (Receive Data Circuit BB,RD) สัญญาณที่ขานี้จะถูกสร้างจากอุปกรณ์ DCE กระแสบิตอนุกรมนี้จะเกิดขึ้นที่อุปกรณ์ DTE ปลายทางและเป็นผลผลิตของวงจรรับข้อมูลของอุปกรณ์ DCE สัญญาณนี้มักเป็นข้อมูลที่ถูกสร้างขึ้นโดยอุปกรณ์ DCE

- ขา 4 (Request To Send Circuit CA,RTS)สัญญาณนี้จะเตรียมพร้อมอุปกรณ์ DCE สำหรับการทำงานส่งข้อมูลเมื่อสัญญาณ RTS นี้อยู่ในสภาวะ “ON” จะทำให้อุปกรณ์ DCE อยู่ในโหมดรับข้อมูล (Receive mode)ในขณะที่สัญญาณนี้อยู่ในสภาวะ “OFF”เสียก่อน สัญญาณนี้จะถูกใช้ร่วมกับสัญญาณ DTR DSR และ DCD ขาสัญญาณ RTS จะถูกใช้อย่างมากในการควบคุมการไหลของข้อมูล
- ขา 5 (Clear To Send Circuit CB,CTS)สัญญาณนี้จะตอบรับกลับไปยังอุปกรณ์ DTE เมื่อได้รับสัญญาณ RTS และข้อมูลสามารถถูกส่งออกไปได้ ข้อมูลจะถูกส่งออกไปตามตัวกลางที่ใช้สื่อสารได้ ก็ต่อเมื่อสัญญาณ CTS นี้อยู่ในสภาวะ “ON” เท่านั้น สัญญาณนี้จะใช้ร่วมกับสัญญาณ DTR DSR และ DCD ขาสัญญาณนี้จะใช้ร่วมกับขา RTS สำหรับควบคุมการไหลของข้อมูล
- ขา 6 (Data Set Ready Circuit CC,DSR)สัญญาณ DSR จะบอกต่ออุปกรณ์ DCE ได้ต่อกับตัวกลางการสื่อสารที่ถูกต้องแล้ว และในบางกรณีจะบ่งชี้ว่าสายโทรศัพท์อยู่ในสภาวะ “OFF HOOK” สภาวะ “OFF HOOK” จะเป็นตัวบ่งชี้ว่าอุปกรณ์ DCE กำลังอยู่ในโหมด dialing หรือกำลังติดต่ออยู่กับอุปกรณ์ DCE อีกตัวหนึ่งอยู่ เมื่อสัญญาณ DSR อยู่ในสภาวะ “OFF” อุปกรณ์ DTE ก็ควรจะถูกให้กำหนดให้ไม่สนใจสัญญาณอื่นๆทั้งหมดจากอุปกรณ์ DCE ถ้าสัญญาณนี้ถูกทำให้อยู่ในสภาวะ “OFF” ก่อนอุปกรณ์ DTR แล้วอุปกรณ์ DCE ก็จะสรุปว่าการสื่อสารนั้นสิ้นสุดลง
- ขา 7 (Signal Common Circuit AB) สายนี้จะให้สัญญาณอ้างอิงของกราวด์ร่วมกันสำหรับวงจรการแลกเปลี่ยนข้อมูลทั้งหมด ยกเว้นวงจร AA หรือ Protective Ground ข้อกำหนด RS-232B จะอนุญาตให้วงจรนี้ถูกตัดต่อเพิ่มเติมกับ Protective Ground ภายในอุปกรณ์ DCE ได้ถ้าจำเป็น

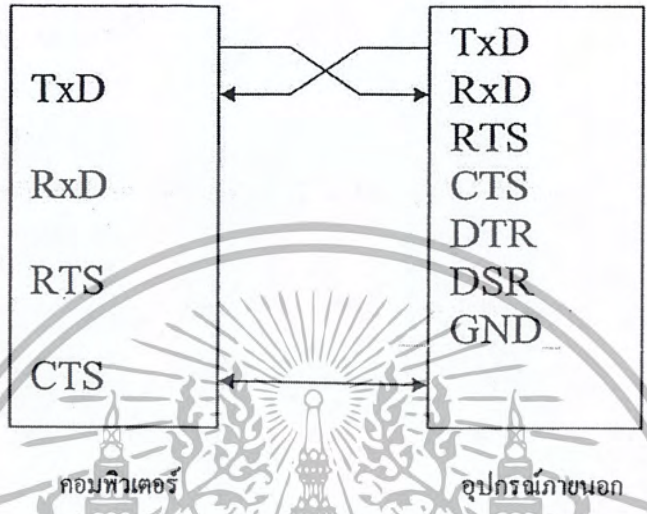
- ขา 8 (Data Carrier Detect Circuit CF ,DCD) ขานี้รู้จักกันในชื่อของ Receive Line Signal Detect (RLSD) หรือขา Carrier Detect(CD)สัญญาณนี้จะเกิด Active เมื่อสัญญาณนี้อยู่ในสภาวะ “OFF” สัญญาณที่ขา RD ควรจะถูกทำให้ค้างอยู่ในสภาวะ “Mark”(สถานะ”1”ในเลขฐานสอง)
- ขา 20 (Data Terminal Ready Circuit CD, DTR) สัญญาณ DTR ถูกใช้ในการควบคุมสวิตซ์อุปกรณ์ DCE เข้ากับตัวกลางในการสื่อสารสัญญาณ DTR ON บ่งชี้ว่าอุปกรณ์ DCE ที่กำลังเชื่อมต่อกันก็ยังต่อร่วมกัน และถ้ายังไม่มีการเชื่อมต่อกันก็สามารถทำให้มีการเชื่อมต่อกันครั้งใหม่ได้ ปกติแล้วสัญญาณ DTR จะอยู่ในสภาวะ “OFF” เพื่อกระตุ้นให้เกิดสภาวะ “ON HOOK ” (วางสาย)อุปกรณ์ DCE โดยการทำให้สัญญาณ DSR แอกทีฟ
- ขา 22 (Ring Indicator Circuit CE, RI)สภาวะ”ON” ของขานี้จะบ่งชี้ว่าได้รับสัญญาณเรียกสายโทรศัพท์จากตัวกลางในการสื่อสาร โดยปกติแล้วจะขึ้นอยู่กับโปรแกรมควบคุม ในการที่จะทำให้เกิดสัญญาณนี้ขึ้นหรือไม่
- ขา 23 (Data Signal Rate Detector Circuit CH/CI DSRD) วงจร CH เป็นส่วนประกอบของ DTE และวงจร CI เป็นส่วนประกอบของ DCE สัญญาณนี้ถูกใช้ในการเลือกใช้ค่าในการส่งอัตราข้อมูลค่าใดค่าหนึ่งในสองค่า ในกรณีที่ใช้โมเด็มที่มีอัตราการส่งข้อมูลได้ 2 ค่า (Dual Rate Modems) ถ้าสัญญาณนี้เป็น “ON” ก็จะเป็นการเลือกอัตราส่วนการส่งข้อมูลที่มีค่าอัตราสูงสุดในสองค่านั้น

2.2.3 ขั้นตอนการติดต่อระหว่างอุปกรณ์ DTE และ DCE

1. เมื่อจ่ายกำลังงานให้กับ DTE และอุปกรณ์ที่จะส่งสัญญาณ DTR ออกมา
2. อุปกรณ์ DCE ถูกเปิดขึ้นและรับรู้สัญญาณ DTR ที่ส่งมาจากอุปกรณ์ DTE
3. อุปกรณ์ DCE ส่งสัญญาณ DSR ออกมาและโมเด็มก็กระทำกระบวนการ OFF HOOK
4. ถ้าสายสัญญาณอยู่ในสภาพดีและปลายอีกด้านหนึ่งก็พร้อมที่จะรับข้อมูลแล้ว จะมีการตรวจจับสัญญาณพาหะแล้วอุปกรณ์ DCE ส่งสัญญาณ DCD ออกมา
5. อุปกรณ์ DCE จะตอบสนองด้วยการ ส่งสัญญาณ CTS ออกมา
6. การติดต่อสื่อสารก็เริ่มขึ้น โปรแกรมควบคุมจะทำการส่งหรือรับข้อมูล



(ก) การต่ออุปกรณ์ภายนอกเข้ากับคอมพิวเตอร์แบบ Null modem [8]



(ข) การต่ออุปกรณ์ภายนอกเข้ากับคอมพิวเตอร์ แบบ RS-232 โดยใช้สายสัญญาณเพียง 3 เส้น

รูปที่ 2.7 การต่ออุปกรณ์ภายนอกกับพอร์ตอนุกรมของคอมพิวเตอร์ในลักษณะต่างๆ [8]

ส่วนลำดับในการตอบรับมีดังนี้

1. อุปกรณ์ DTE จะส่งสัญญาณ DTR ออกมา
2. อุปกรณ์ DCE จะอยู่ในโหมดตอบรับอัตโนมัติ (auto answer mode) โดยมีสัญญาณออกมา
3. สถานีปลายทางส่งสัญญาณเรียกอุปกรณ์ DCE และอุปกรณ์ DCE ส่งสัญญาณ RI ออกมา
4. อุปกรณ์ DTE รับรู้ถึงอุปกรณ์ RI ที่ส่งมาจากเครื่องปลายทาง และอุปกรณ์ CDE ก็เข้าสู่สภาวะ OFF HOOK

5. อุปกรณ์ DCE ทำการแลกเปลี่ยนข้อมูลกับอุปกรณ์ DCE ที่อีกปลายทางหนึ่ง และมีการส่งสัญญาณ DCD ออกมา
6. อุปกรณ์ DTE จะส่งสัญญาณ RTS ออกมาหรืออาจรอข้อมูลก็ได้
7. อุปกรณ์ DCE จะตอบสนองด้วยการส่งสัญญาณ DTS กลับออกมา
8. การติดต่อสื่อสารก็จะเริ่มขึ้น

2.3 Single chip 2.4 GHz Transceiver nRF2401

Mode of operation (โหมดดำเนินการ)

2.3.1 Overview

ระบบย่อย nRF2401 สามารถจัดอยู่ในโหมดหลักได้ต่อไปนี้ โดยจะขึ้นกับ 3 pins ควบคุม

ตารางที่ 2.7 nRF2401 subsystem main modes

Mode	PWR_UP	CE	CS
Active	1	1	0
Configuration	1	0	1
Stand by	1	0	0
Power down	0	X	X

2.3.2 Active modes

ระบบย่อย nRF2401 มี 2 Active โหมด (Tx/Rx)

- Shock Burst
- Direct Mode (not supported by nRF24E1)

ฟังก์ชันการทำงานของอุปกรณ์ในโหมดนี้เลือกโดยเนื้อหาของ Configuration word โดย Configuration word จะแสดงในส่วนของ Configuration

2.3.3 Shock Burst

เทคโนโลยี Shock Burst ใช้บนชิป FIFO เพื่อจับเวลาข้อมูลที่อัตราข้อมูลต่ำและส่งต่อที่อัตราข้อมูลสูงมาก ด้วยเหตุนี้จะสามารถทำให้เกิดการลดทอนของกำลังสูงสุดเมื่อดำเนินการระบบย่อยnRF2401 ใน Shock Burst เราจะได้อัตราข้อมูลสูงสุด (1 Mbps) โดย band 2.4 GHz โดยไม่ต้องใช้ไมโครคอนโทรลเลอร์ความเร็วสูงและราคาสูง (MCU) สำหรับการดำเนินการทางข้อมูล

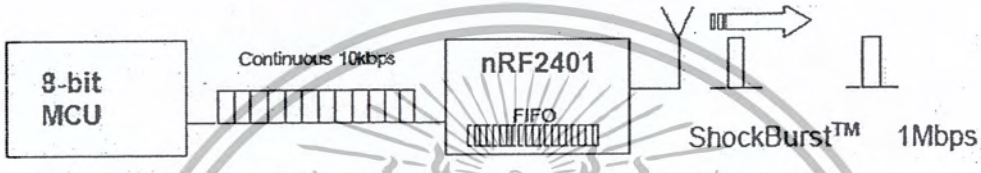
โดยการใส่การประมวลผลข้อมูลความเร็วสูงสัมพันธ์กับ RF Protocol บนชิป nRF24E1 มีประโยชน์ดังนี้

- ลดการใช้กระแส
- ระบบที่มีราคาถูก (ใช้งานได้ง่ายสำหรับ microcontroller ราคาไม่แพง)
- ลดอัตราการเสี่ยงของการชนกันในอากาศ เนื่องจากการส่งสัญญาณในช่วงเวลาสั้นๆได้ดี

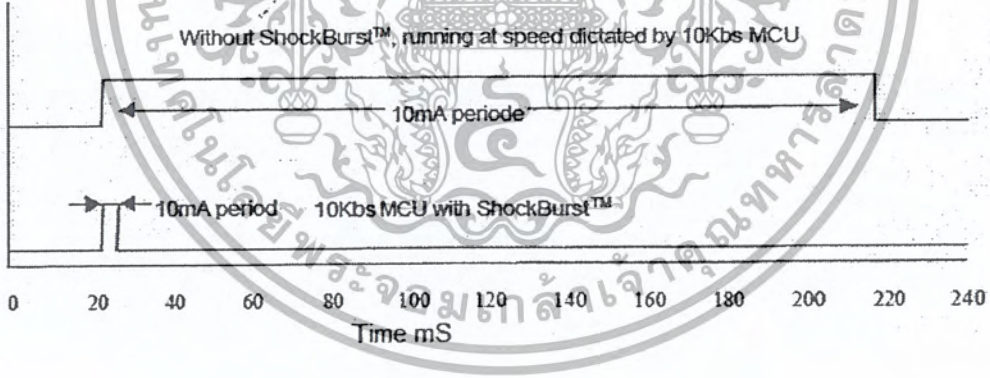
ระบบย่อย nRF 2401 สามารถโปรแกรมได้โดยใช้สาย interface พื้นฐาน 3 เส้น ซึ่งอัตราของข้อมูลจะเลือกโดยความเร็ว CPU โดยยอมรับส่วนดิจิตอลของ application เพื่อจะ run ที่ความเร็วต่ำขณะที่อัตราข้อมูลบน RF Link มีค่าเพิ่มสูงสุด โหมด Shock Burst จะลดการใช้กระแสเฉลี่ยในapplication

2.3.3.1 หลักการของ Shock Burst

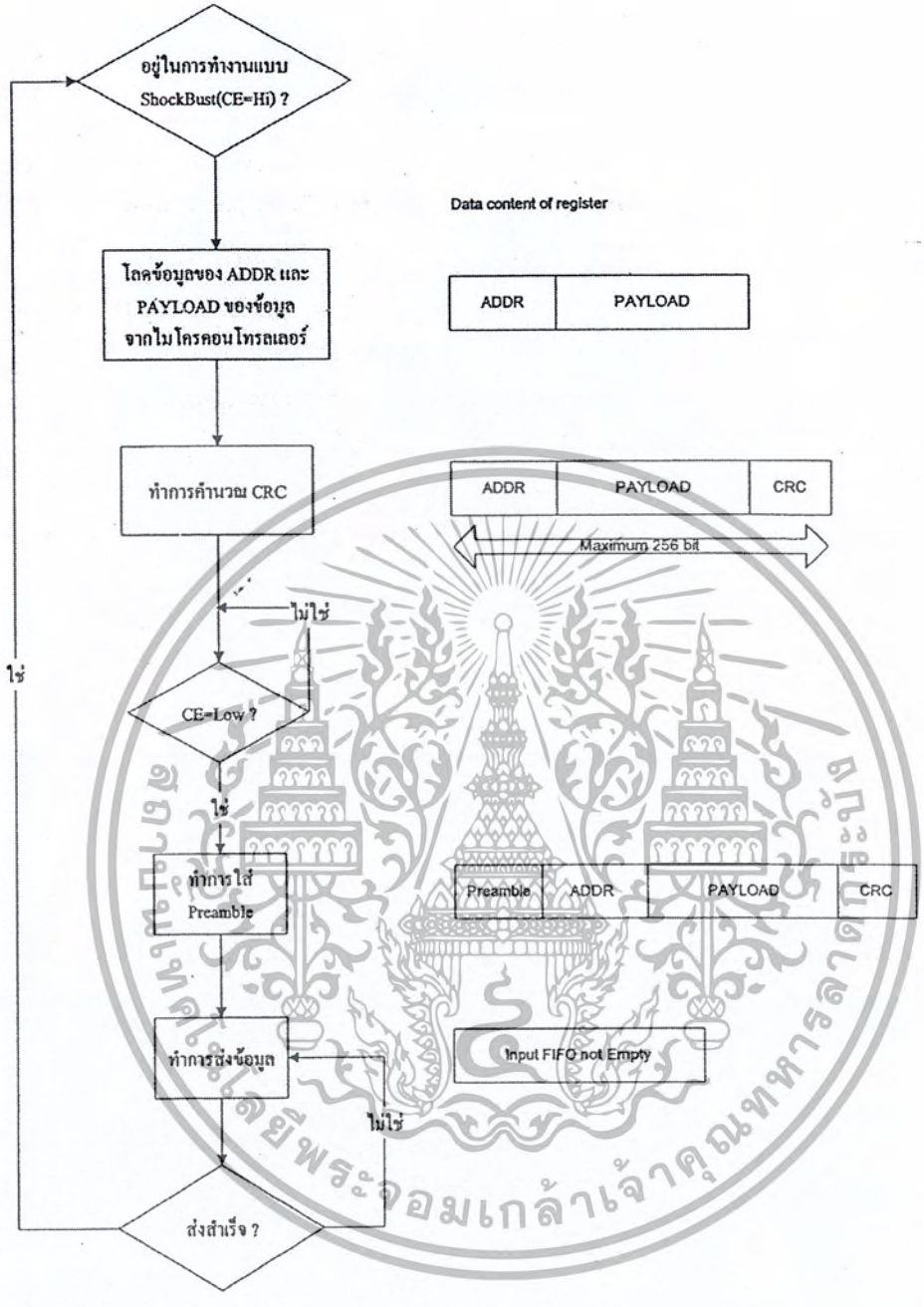
เมื่อระบบย่อย nRF2401 มีโครงร่าง Shock Burst กระบวนการ Tx จะดำเนินการดังต่อไปนี้ตัวอย่าง 10 kbps



รูปที่ 2.8 จับเวลาข้อมูลโดย CPU ส่งโดยเทคโนโลยี Shock Burst [3]



รูปที่ 2.9 การใช้กระแส RF โดยใช้และไม่ใช้เทคโนโลยี Shock Burst [3]



รูปที่ 2.10 แผนผังงาน (Flow Chart) Shock Burst การส่งของระบบย่อย nRF2401 [3]

2.3.3.2 การส่ง Shock Burst

CPU interface pins : CE,CLK1,DATA

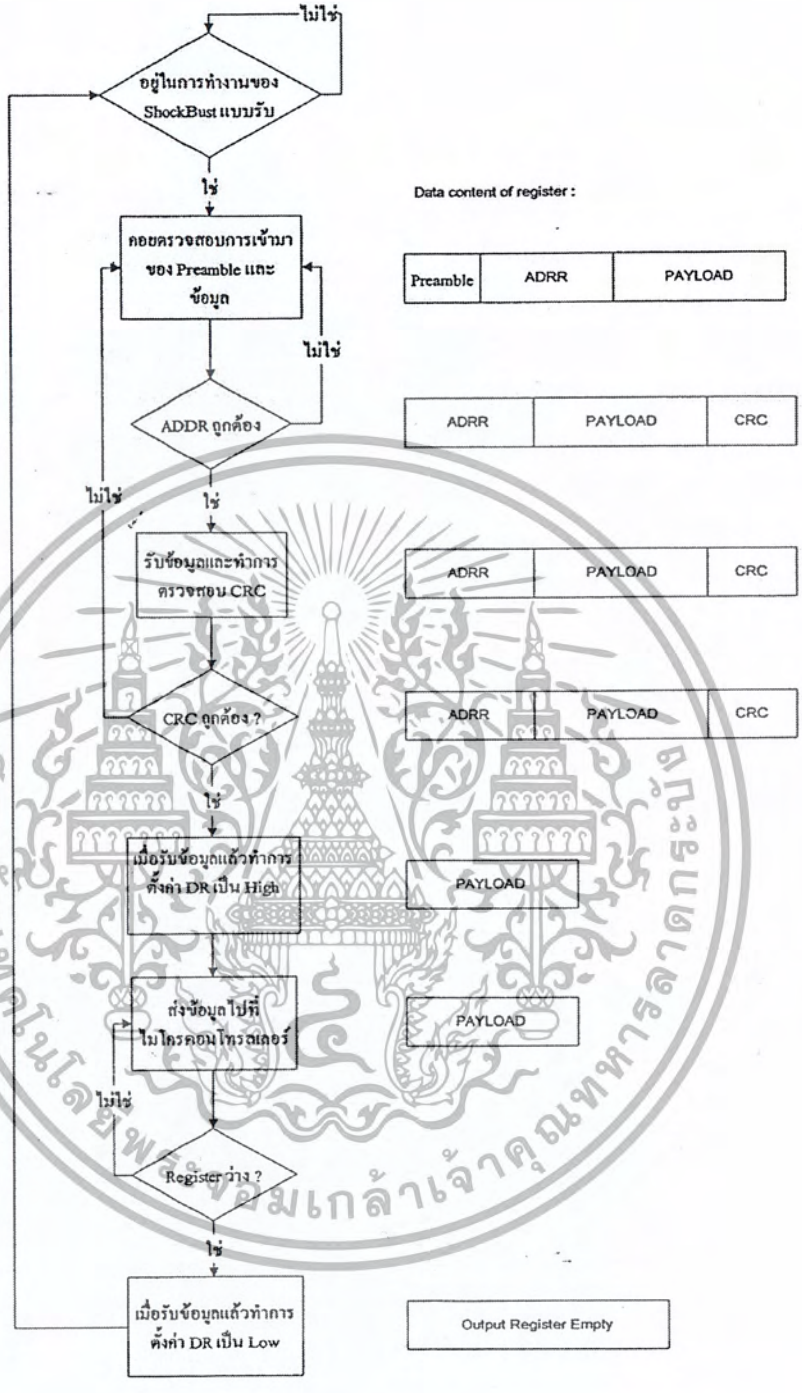
1. เมื่อ CPU มีข้อมูลที่จะส่ง set CE high คือ การกระตุ้น nRF2401 ให้ประมวลผลบนบอร์ด

2. แอดเดรสของโหมดรับ (RX) และ payload ข้อมูลจะจับเวลาเข้าสู่ช่วงเวลาระบบ ย่อย nRF2401 protocol หรือ CPU ตั้งความเร็วน้อยกว่า 1 Mbps

3. CPU ตั้ง CE low คือ กระตุ้นการส่ง Shock Burst

4. Shock Burst

- RF front end จะมีกำลังเพิ่มขึ้น
- RF Package จะสมบูรณ์
- ข้อมูลจะถูกส่งที่ความเร็วสูง (250 bps หรือ 1 Mbps แล้วแต่ผู้ใช้)



รูปที่ 2.11 แผนผังงาน (Flow Chart) Shock Burst การรับของระบบย่อย nRF2401 [3]

2.3.3.3 การรับ Shock Burst

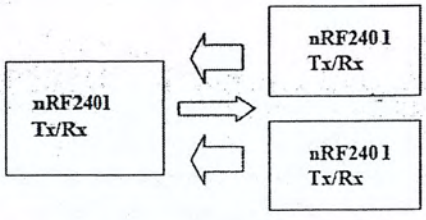
CPU interface pins : CE,DRT,CLK1,DATA (1 ช่องรับ Rx)

1. แอดเดรสถูกต้องและขนาดของ payload ของ RF package ที่เข้ามาจะ set
2. 2401 เป็นโครงร่างของ Shock Burst Rx
3. เมื่อกระตุ้น Rx ; set CE High
4. หลังจากตั้ง 200 us nRF2401 จะตรวจสอบอากาศสำหรับการสื่อสารที่จะเข้ามา เมื่อPackage ที่ถูกต้องถูกเรียกมา (แอดเดรสถูกต้องและพบ CRC) nRF2401 จะทำการย้ายบทนำแอดเดรสและบิต CRC
5. จากนั้นระบบย่อย nRF2401 จะ interrupt CPU โดยการตั้ง DR1 high
6. CPU จะ set CE low เพื่อป้องกัน RF front end (โหมดที่กระแสต่ำ)
7. CPU จะจับเวลาที่หยุด payload data ที่อัตราที่เหมาะสม
8. เมื่อ payload data ทั้งหมดถูกเรียกมา nRF2401 set DR1 low อีกครั้ง และพร้อมสำหรับข้อมูลที่จะเข้ามา ถ้า CE ยัง high ระหว่างที่ download ข้อมูล ถ้า CE set low การลำดับStart up ใหม่จะสามารถเริ่มต้นได้

2.3.4 DUO Reciever Simultaneous two channel Receive mode

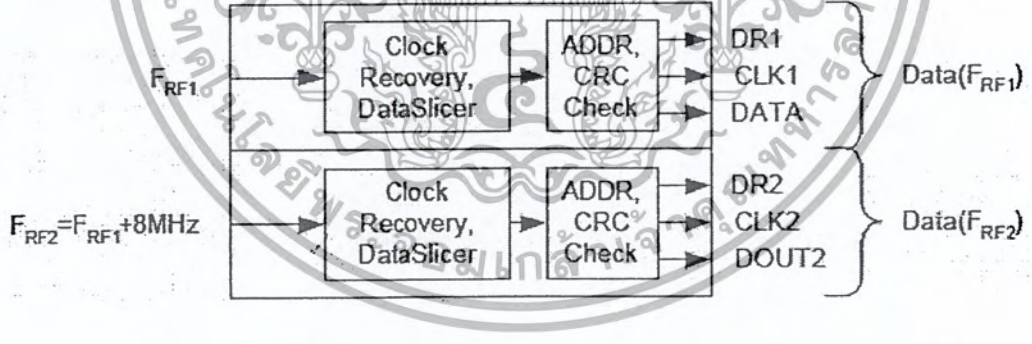
ในโหมด Shock Burst ของ nRF24E1 สามารถรองรับความถี่ 2 channel ที่ขนานกันและเป็นอิสระ ที่อัตราข้อมูลสูงสุดได้สะดวกในเวลาเดียวกันซึ่งหมายความว่า

- RF 2401 สามารถรับข้อมูลจากเครื่องส่ง 2 เครื่อง ที่ 1 Mbps และ 8 MHz ผ่าน 1 เสาอากาศ
- Output จาก 2 ช่องสัญญาณข้อมูล จะป้อนให้กับ 2 interface pins ที่แยกจากกัน
- Data Channel 1 : CLK1,DATA,DR1
- Data Channel 2 : CLK2,DATA2,DR2



รูปที่ 2.12 Simultaneous 2 Channel receiver on nRF24E1 [3]

สำหรับ nRF24E1 จะสามารถรับที่ช่องสัญญาณข้อมูลช่องที่ 2 ได้นั้น ความถี่ช่องสัญญาณต้องอยู่สูงกว่า 8 MHz ความถี่ของช่องสัญญาณข้อมูล nRF2401 ต้อง program เพื่อรับข้อมูลที่ความถี่ของช่องสัญญาณที่ 1 ไม่มีการใช้เวลา multiplexing เพื่อทำฟังก์ชันนี้ให้สำเร็จ ใน Direct Mode ถ้ามันไม่รับส่งได้พร้อมกันระหว่าง 2 ช่องสัญญาณข้อมูล CPU ต้องสามารถจัดการข้อมูลที่เข้ามาพร้อมกันได้ส่วนใน Shock Burst สามารถทำให้ CPU หยุดจับเวลาข้อมูลช่วงนึง ขณะที่ไม่มีการสูญเสีย package ข้อมูลและไม่ลดประสิทธิภาพของ CPU



รูปที่ 2.13 DUOCiever โดยมี 2 ช่องสัญญาณรับข้อมูลที่เป็นอิสระต่อกัน [3]

2.4 การเชื่อมต่อคีย์แพดเข้ากับไมโครคอนโทรลเลอร์ MCS-51

จากโครงงานจะทำการต่อคีย์แพดเข้ากับไมโครคอนโทรลเลอร์ที่พอร์ต 1 โดยใช้สายทั้งหมด 7 เส้น เป็นสาย column 3 เส้นและสาย row 4 เส้น โดยไมโครคอนโทรลเลอร์จะทำการส่งข้อมูล "0"ไปยัง P 1.4 – 1.6 ตามลำดับ ในทุกครั้งที่มีการส่งข้อมูลไปยังสาย column ของคีย์แพด ไมโครคอนโทรเลอร์จะทำการอ่านค่าที่ P1.0 – P1.3 เข้ามาด้วย หากไม่มีการกดค่าของ P1.0 – P1.3 ก็จะเป็น "1" ทั้งหมด ถ้าหากมีการกดคีย์ ค่าของ P1.0 – P1.3 ก็จะไม่เป็น "1111" อีกต่อไป เป็นการแจ้งให้ทราบว่ามีการกดคีย์แพดขึ้นแล้ว จากนั้นไมโครคอนโทรลเลอร์ก็จะทำการค้นหาตำแหน่งต่อไปโดยการค้นหาตำแหน่งนั้น สิ่งที่จะได้มาอย่างแรกคือ ค่าของตำแหน่งของคีย์นั้น จากนั้นก็จะนำค่าตำแหน่งนั้นไปเปิดตารางข้อมูลเพื่อที่จะได้ค่าที่ต้องการนำไปแสดงผลที่แท้จริง

2.5 การเชื่อมต่อกับโมดูลแอลซีดี (LCD Module)

ในโมดูลแอลซีดีนั้นจะมีส่วนประกอบ 3 ส่วนหลักดังนี้

- ตัวแสดงผล (Display) ภายในเป็นผลึกเหลวที่สามารถแสดงผลให้เห็นได้โดยอาศัยแสงจากภายนอก ดังนั้นจึงต้องมีมุมในการมองข้อมูลที่แสดงบนจอแอลซีดี
- ตัวควบคุม (Controller) เป็นตัวรับข้อมูลจากอุปกรณ์ภายนอก มาควบคุมการทำงานของโมดูลแอลซีดี เช่น ลบจอภาพ แสดงตัวอักษรหรือเลื่อนเคอร์เซอร์เป็นต้น ตัวควบคุมนี้ใช้ชิปควบคุมโดยเฉพาะชิป ที่นิยมใช้คือเบอร์ HD 44780 และ HD 61830 โดย HD 44780 จะใช้ควบคุม LCD แบบอักขระ และ HD61830 ใช้ควบคุม LCD แบบกราฟฟิก
- ตัวขับ (Driver) เป็นตัวรับสัญญาณจากตัวควบคุมมาขับให้ตัวแสดงผลแสดงข้อมูลตามที่กำหนด ชิปที่ใช้ทำงานเป็นตัวขับนี้ได้แก่เบอร์ HD44100H และ MSM5259 เป็นต้น

2.5.1 โครงสร้างภายในของตัวควบคุมโมดูล LCD

ในการใช้งานโมดูล LCD จำเป็นต้องทำความเข้าใจเกี่ยวกับโครงสร้างและคำสั่งในการควบคุมให้ดีเสียก่อน โดนในที่นี้จะทำการแสดงตัวอย่าง โมดูล LCD แบบอักขระ เพราะสามารถเข้าใจง่ายโดยบล็อกไดอะแกรมภายในของชิปควบคุม LCD เบอร์ HD44780H ซึ่งใช้ในโมดูล LCD แบบอักขระประกอบด้วยบัฟเฟอร์อินพุต เอาท์พุต เป็นส่วนที่ใช้ในการติดต่อรับส่งข้อมูลกับอุปกรณ์ภายนอก เพื่อที่จะถ่ายทอดข้อมูลเข้าออกภายในตัวควบคุม

รีจิสเตอร์คำสั่ง (Instruction Register : IR) เป็นรีจิสเตอร์ที่ใช้รับข้อมูลจากอุปกรณ์ภายนอกเพื่อถ่ายทอดต่อไปยังหน่วยความจำ ที่ทำหน้าที่เก็บข้อมูลแสดงผล หรือ นำข้อมูลไปสร้างตัวอักษรเพิ่มเติมในแรมเก็บตัวอักษร

แรมเก็บข้อมูลแสดงผล (Display Data RAM : DDRAM) เป็นหน่วยความจำแบบแรมทำหน้าที่เก็บข้อมูลที่มาจากรีจิสเตอร์ DR ตัวควบคุมจะนำข้อมูลใน DDRAM นี้ไปเปิดตาราง (Look up table) ของตัวอักษรที่เก็บไว้ในหน่วยความจำรอมและแรมเก็บตัวอักษร เพื่อนำไปแสดงผลที่ตัวแสดงผล

รอมเก็บตัวอักษร (Character Generator ROM : CGROM) เป็นหน่วยความจำแบบรอมที่ใช้เก็บข้อมูลตัวอักษรหรือสัญลักษณ์ที่สามารถอ่านออกไปแสดงที่ตัวแสดงผลได้ มีขนาด 7200 บิต

แรมเก็บตัวอักษร (Character Generator RAM : CGRAM) เป็นหน่วยความจำแรมที่ใช้เก็บตัวอักษรที่มีการสร้างเพิ่มเติมขึ้นใหม่ ในกรณีที่ตัวอักษรใน CGROM ไม่เพียงพอ มีขนาด 512 บิต การเขียนและอ่านค่าไปใช้นั้นทำได้เช่นเดียวกับ CGROM คือ เขียนข้อมูลลงใน DDRAM แล้วตัวควบคุมจะมาอ่านค่าจาก CGROM เอง

แฟลกบิซซี (Busy Flag) เป็นส่วนที่ทำหน้าที่แจ้งสถานะการทำงานของตัวควบคุมให้อุปกรณ์ภายนอกทราบว่าตัวควบคุมที่พร้อมที่จะรับข้อมูลหรือคำสั่งหรือไม่

สำหรับโมดูล LCD ที่ใช้ในโครงงาน เป็นขนาด 16 ตัวอักษร 2 บรรทัด เนื่องจากเป็นขนาดที่เหมาะสมที่สุดในการใช้การเป็นจอแสดงผลสำหรับการสั่งอาหารซึ่งพิจารณาจากประโยชน์ใช้สอยและความประหยัด ซึ่งประกอบด้วยขาทั้งหมด 14 ขา

Vss (ขา 1) : ต่อกราวน์

Vdd (ขา 2) : ต่อไฟเลี้ยง +5 V

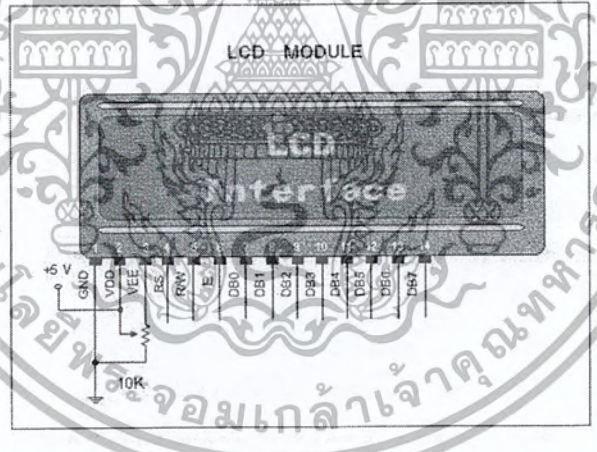
Vo (ขา 3) : เป็นอินพุตรับแรงดันเพื่อปรับความเข้มของการแสดงผล

RS (ขา 4) : เป็นขาอินพุตใช้ในการแยกชนิดของข้อมูลที่ทำการประมวลผลในขณะนั้นว่าเป็นคำสั่งสำหรับรีจิสเตอร์ IR หรือเป็นข้อมูลสำหรับรีจิสเตอร์ DR โดยขานี้เป็น “0” ข้อมูลที่ส่งมาจะเป็นคำสั่งแต่ถ้าขานี้เป็น “1” ข้อมูลที่ส่งมาจะเป็นข้อมูลสำหรับการแสดงผล

R/W (ขา 5) : เป็นขาที่ใช้ลือกการอ่านหรือเขียนข้อมูลจากโมดูล LCD ถ้าเป็น “0” เป็นการกำหนดให้เขียนข้อมูล แต่ถ้าเป็น “1” จะเป็นการอ่านข้อมูล

E (ขา 6) : เป็นขาสำหรับรับสัญญาณพัลส์ enable โมดูล LCD ให้ทำงาน

D0 – D7 : เป็นขาที่ใช้เป็นทางผ่านของข้อมูลระหว่าง LCD กับอุปกรณ์ภายนอกขนาด 8 บิต



รูปที่ 2.14 LCD Module [7]

การจัดตำแหน่งของหน่วยความจำในการแสดงผลตัวอักษรของ LCD Module ขนาด 16x2 มีดังนี้คือ

ตารางที่ 2.8 ตำแหน่งของหน่วยความจำ DDRAM

0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07	0x08	0x09	0x0A	0x0B	0x0C	0x0D	0x0E	0x0F
0x40	0x41	0x42	0x43	0x44	0x45	0x46	0x47	0x48	0x49	0x4A	0x4B	0x4C	0x4D	0x4E	0x4F

จากตารางแสดงตำแหน่งของหน่วยความจำ DDRAM จะเห็นว่า LCD Module ขนาด 16x2 จะมีตำแหน่งแสดงผลตัวอักษรบรรทัดบน 16 ตัวอักษร ตำแหน่งที่ 0x00-0x0F และบรรทัดล่าง 16 ตัวอักษร ตำแหน่งที่ 0x40-0x4F แต่ในการเขียนโปรแกรมควบคุมการทำงานของ LCD Module นั้น ถ้าเราต้องการกำหนดให้แสดงผลตัวอักษรที่ตำแหน่งใดๆ ภายในตัว LCD Module ขนาด 16x2 เรา จะต้องอ้างอิงตำแหน่งการแสดงผลตัวอักษรตามตารางนี้คือ

ตารางที่ 2.9 การกำหนดตำแหน่งแสดงผลของ LCD Module 16x2

0x80	0x81	0x82	0x83	0x84	0x85	0x86	0x87	0x88	0x89	0x8A	0x8B	0x8C	0x8D	0x8E	0x8F
0xC0	0xC1	0xC2	0xC3	0xC4	0xC5	0xC6	0xC7	0xC8	0xC9	0xCA	0xCB	0xCC	0xCD	0xCE	0xCF

2.5.2 คำสั่งควบคุมโมดูล LCD

ในการเขียนคำสั่งลงในตัวควบคุม แน่นอนว่าต้องกำหนดให้ขา RS และ R/W เป็น "0" แล้วเขียนคำสั่งตามไป คำสั่งควบคุม โมดูล LCD สรุปได้ดังตารางต่อไปนี้

ตารางที่ 2.10 คำสั่งในการควบคุมการทำงานแสดงผลของ LCD Module

คำสั่งควบคุมการแสดงผล	ความหมายของคำสั่ง
0x38	กำหนดแสดงผลขนาด 2 บรรทัด ความละเอียด 5x7 จุด ขนาด 8 บิต
0x01	เคลียร์หน้าจอ LCD และเลื่อนเคอร์เซอร์ทางด้านซ้ายสุด
0x02	เลื่อนเคอร์เซอร์ไปตำแหน่งเริ่มต้นซ้ายสุด
0x04	แสดงข้อมูลและเลื่อนเคอร์เซอร์ไปทางขวา
0x05	เลื่อนเคอร์เซอร์ไปทางขวา
0x06	แสดงข้อมูลและเลื่อนเคอร์เซอร์ไปทางซ้าย
0x07	เลื่อนเคอร์เซอร์ไปทางซ้าย
0x08	ไม่แสดงข้อความและไม่แสดงเคอร์เซอร์
0x0A	ไม่แสดงข้อความแต่แสดงเคอร์เซอร์
0x0C	แสดงข้อความแต่ไม่แสดงเคอร์เซอร์
0x0E	แสดงข้อความและแสดงเคอร์เซอร์
0x0F	แสดงข้อความและแสดงเคอร์เซอร์กระพริบ
0x10	เลื่อนตำแหน่งเคอร์เซอร์ไปทางซ้าย
0x14	เลื่อนตำแหน่งเคอร์เซอร์ไปทางขวา
0x18	เลื่อนตัวอักษรไปทางซ้าย
0x1C	เลื่อนตัวอักษรไปทางขวา
0x80	กำหนดแสดงผลเริ่มต้นที่ตำแหน่ง 0x80 ซ้ายสุดบรรทัดบน
0xC0	กำหนดแสดงผลเริ่มต้นที่ตำแหน่ง 0xC0 ซ้ายสุดบรรทัดล่าง

2.6 Visual C#

Visual C# เป็นภาษาหนึ่งในVisual Studio .NETซึ่งประกอบด้วย Visual C#, Visual Basic, Visual C++ , Visual Web Developer เวอร์ชั่นปัจจุบัน Visual Studio .NET 2008

ภาษา VC# เป็นModern Object Oriented Programming หรือภาษาเชิงวัตถุ ในVisual Studio .NET ภาษาแต่ละภาษาสามารถสร้างวัตถุขึ้นมาโดยเป็นไฟล์นามสกุล .dll แล้วสามารถนำมาใช้งานร่วมกันได้โดยไม่ต้องปรับปรุงแก้ไขใหม่ เช่น VC# สามารถสร้าง Form เก็บข้อมูลลูกค้า สร้าง Form เก็บข้อมูลคลังสินค้า เสร็จแล้วสามารถนำมาร่วมกันสร้างเป็น Application ขึ้นมาได้โดยแต่ละภาษาสามารถใช้งาน .NET Framework ได้ และติดต่อสื่อสารกันหรือแลกเปลี่ยนข้อมูลระหว่าง Platform ผ่านทาง XML (Extensible Markup Language) ซึ่งทำหน้าที่เป็นตัวกลางในการแลกเปลี่ยนระหว่าง Platformของไฟล์ฐานข้อมูล

2.6.1 หลักการทำงานของภาษา C#

ทุกภาษาใน .NET Framework เมื่อทำการ Compiler แล้วจะเปลี่ยนเป็นโค้ดภาษา MSIL แล้วจากนั้นเมื่อรัน Applicaton.NET จะทำการแปรภาษา MSIL โดยใช้CLRเพื่อนำไปสร้าง codeไบนารี่ทำให้แต่ละภาษามีความสามารถที่เท่าๆกันโดยcodeไบนารี่ที่สร้างขึ้นก็จะสามารถเข้ากันได้กับเครื่องคอมพิวเตอร์ในแต่ละแบบที่ต่างกันออกไป หลักการทำงานของโปรแกรมที่ถูกสร้างด้วย ภาษา C# สามารถนำไปประยุกต์ใช้งานในด้านต่างๆได้ ดังนี้

Console Application

เป็นการเขียนโปรแกรมด้วยภาษา C# ที่ใช้แสดงในรูปแบบ DOS Prompt ใช้ทดสอบโปรแกรมโดยไม่เน้นถึงความสวยงามและความสะดวกในการใช้งาน ใช้สำหรับสร้างแอปพลิเคชั่น ที่ไม่มี User Interface โดยจะแสดงผลลัพธ์ และป้อนค่า ผ่านหน้าต่าง Command Prompt ในรูปแบบรายการสตริง เช่น แสดงรายการไฟล์ คัดลอกและลบไฟล์

Windows Control

ใช้สำหรับสร้างคอนโทรลในรูปแบบที่เราต้องการ เพื่อนำมาใช้ในโปรเจ็คอื่นในกรณีที่คอนโทรลที่มากับ ToolBox เช่น Button control, Label control เป็นต้น ไม่สามารถให้คุณสมบัติที่เราต้องการได้

Windows Service

จะทำงานในระบบของ Windows โดยผู้ใช้จะไม่ทราบว่ามี Service นี้ทำงานอยู่ จะเริ่มทำงานก็ต่อเมื่อได้รับการร้องขอจากระบบ โดยมากจะนำมาใช้งานในระบบ Client/Server เพื่อใช้สร้างแอปพลิเคชั่นที่รันอยู่เบื้องหลังและเริ่มทำงานได้ทันที

Web Control

ใช้สร้างคอนโทรลที่ทำงานบนเว็บ เช่น คอนโทรลปฏิทิน เพียงแค่ลากคอนโทรลไปวางบนเว็บเพจในตำแหน่งที่ต้องการก็สามารถใช้งานปฏิทินในเว็บไซต์นี้ได้ โดยที่ไม่ต้องเขียนโค้ดในส่วนนี้เองเป็นต้น

ASP.NET Web Application

คือโปรแกรมที่อยู่ในWeb Server ที่จะคอยให้บริการแก่ Client ผ่านทางบราวเซอร์โดยจะแสดงผลผ่านเว็บเพจที่เป็นASP.NET

2.6.2 การเขียนโปรแกรมแบบ OOP

การเขียนโปรแกรม OOP นั้น เป็นการเลียนแบบแนวคิดของสิ่งที่ผู้เขียนโปรแกรมพบเห็นในสิ่งต่างๆที่อยู่รอบๆตัว แนวคิดของ OOP นั้นจะกำหนดให้สิ่งต่างๆเป็นวัตถุ หรือ ออบเจ็กต์ (Object) ซึ่งออบเจ็กต์แต่ละตัว จะถูกสร้างขึ้นมาจากแม่พิมพ์ที่เรียกว่า คลาส (Class) ดังนั้นออบเจ็กต์ที่สร้างจากคลาสจะถือว่าเป็นคนละตัวกันแต่มีชนิดเดียวเพราะสร้างจากคลาสเดียวกันเมื่อมามองถึงออบเจ็กต์ตัวหนึ่งๆ จะเห็นว่ามันจะต้องมีข้อมูลที่ให้กำหนดในลักษณะ เฉพาะของออบเจ็กต์ ซึ่งเรียกว่า พร็อพเพอร์ตี้ (Property) เมื่อมีพร็อพเพอร์ตี้แล้วสิ่งหนึ่งที่มักจะขาดไม่ได้ในออบเจ็กต์ก็คือความสามารถในการทำงานของออบเจ็กต์ ซึ่งถูกเรียกว่า เมธอด (Method) เมื่อผู้เขียนโปรแกรมสร้างออบเจ็กต์หนึ่งขึ้นมาจากคลาส ในภาษาของการเขียนโปรแกรม จะกล่าวได้ว่าเป็น “ออบเจ็กต์” ของคลาส

2.6.3กลุ่มออบเจ็กต์ADO.NET

ADO.NET มาจากคำว่า ActiveX Data Object .NET เป็นเครื่องมือสำหรับติดต่อกับฐานข้อมูลของ .NET Framework โดยได้รับการพัฒนามาจาก ADO โดยจัดเตรียมคลาสสำหรับพัฒนา Database Application ด้วยภาษาต่างๆของ .NET เช่น C# จุดเด่นของ ADO.NET คือต้องไม่มีการเชื่อมต่อกับฐานข้อมูลตลอดเวลาแต่จะอ่านข้อมูลเก็บไว้ในหน่วยความจำ ซึ่งรับผิดชอบในส่วนของการจัดการข้อมูลในฐานข้อมูลประเภทต่างๆ ของสถาปัตยกรรม .NET โดยจะอาศัยเทคโนโลยี OLEDB Data Provider เป็นตัวกลางในการเชื่อมต่อกับฐานข้อมูลแต่ละประเภท ADO.NET ซึ่งมีการจัดเตรียมการเข้าถึงข้อมูลได้ 2 วิธีคือ Connection เป็นการติดต่อกับฐานข้อมูลโดยสร้างการเชื่อมต่อไว้ตลอดเวลา และ Disconnection เป็นการติดต่อกับฐานข้อมูล โดยไม่ต้องทำการสร้างการเชื่อมต่อไว้กับฐานข้อมูลตลอดเวลา Connection เป็นการกระทำกับข้อมูลใน Data Source โดยตรงโดยใช้ Connection Object, Command Object และ DataReader Object ส่วน Disconnection เป็นการคัดลอกข้อมูลมาเก็บไว้บนพื้นที่ข้องหน่วยความจำ เรียกพื้นที่นี้ว่า DataSet โดยไม่ต้องสร้างการเชื่อมต่อไว้ตลอดเวลาและการดำเนินการข้อมูลจะเกิดใน Dataset ไม่ใช่ข้อมูลต้นฉบับ เมื่อสิ้นสุดการดำเนินการจึงสร้างการเชื่อมต่อกลับไปเพื่อปรับปรุงข้อมูล วิธีนี้เป็นการเพิ่มประสิทธิภาพการดำเนินงานเมื่อมีการเรียกใช้ข้อมูลจากผู้ใช้เป็นจำนวนมาก

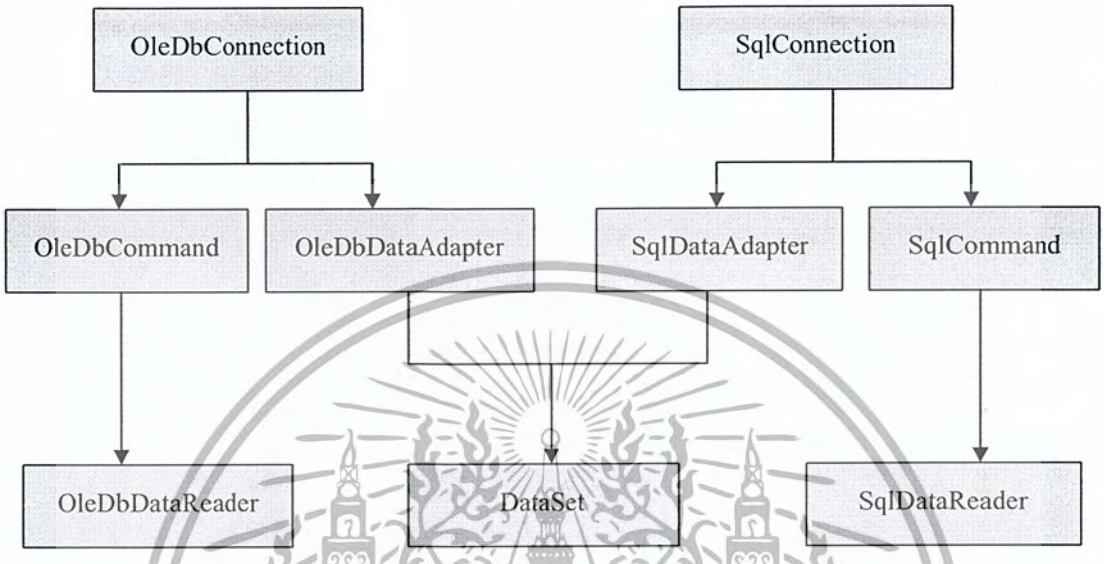
ADO.NET Data Provider ได้จัดเตรียมเครื่องมือต่างๆสำหรับการเข้าถึงฐานข้อมูลไว้ คือ Connection, Command, Data Reader และ Data Adapter ประกอบด้วย 4คลาสสำหรับสร้างอ็อบเจ็กต์ 4 ชนิดดังต่อไปนี้

ตารางที่ 2.11 อ๊อบเจ็กต์ของ .NET Data Provider

Object	คำอธิบาย
Connection	สร้างการเชื่อมต่อไปยังฐานข้อมูล
Command	ใช้เก็บคำสั่ง SQL
Data Reader	กำหนดขอจำกัดในการเข้าถึงฐานข้อมูล เช่น read-only และ forward only
Data Adapter	เป็นตัวกลางระหว่าง Command,Connectionและ Dataset Object

ตารางที่ 2.12 Data Provider ของ .NET Framework

Provider	Namespace	คำอธิบาย
SqlClient	System.Data.SqlClient	ใช้เข้าถึงฐานข้อมูล SQL Server
OleDb	System.Data.OleDb	ใช้เข้าถึงฐานข้อมูลที่รองรับเทคโนโลยี OLE DB
OleDc	System.Data.Odbc	ใช้เข้าถึงฐานข้อมูลที่รองรับเทคโนโลยี ODBC
OracleClient	System.Data.OracleClient	ใช้เข้าถึงฐานข้อมูล Oracle



รูปที่ 2.15 ความสัมพันธ์ระหว่างออบเจ็กต์ [4]

- กลุ่มออบเจ็กต์ที่อาศัย .NET OLEDB Data Provider ทำหน้าที่เข้าถึงข้อมูลในฐานข้อมูล มี 3 ตัว คือ ออบเจ็กต์ OleDbConnection , OleDbDataAdapter และ OleDbCommand
- กลุ่มออบเจ็กต์ที่อาศัย .NET MS SQL Server Data Provider ทำหน้าที่เข้าถึงข้อมูลในฐานข้อมูล มี 3 ตัว คือออบเจ็กต์ SqlConnection , SqlDataAdapter และ SqlCommand
- กลุ่มออบเจ็กต์ที่ใช้เก็บผลการทำงานมีทั้งหมด 3 ตัวคือออบเจ็กต์ DataSet ,OleDataReader และ SqlDataReade
- ออบเจ็กต์ OleDbConnection ทำหน้าที่เชื่อมต่อกับฐานข้อมูล
- ออบเจ็กต์ OleDbCommand ทำหน้าที่รันชุดคำสั่ง SQL ทั้งประเภทอ่าน (Read Operation) และเขียนข้อมูล (Write Operation)
- ออบเจ็กต์ OleDbDataAdapter ทำหน้าที่รันชุดคำสั่ง SQL ประเภทอ่านอย่างเดียว

ขั้นตอนการติดต่อกับฐานข้อมูลโดยอาศัยกลุ่มออบเจ็กต์ ADO.NET มี 2 ขั้นตอนใหญ่ ๆ

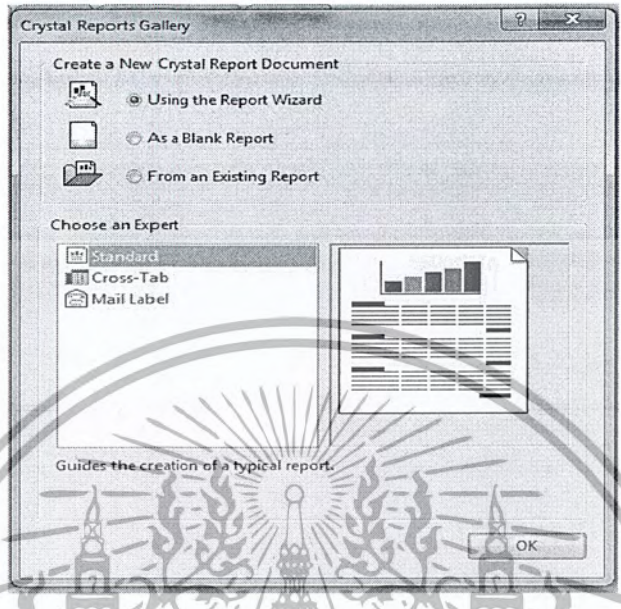
1. เชื่อมต่อกับฐานข้อมูลด้วยออบเจ็กต์ OleDbConnection
2. คิวรีข้อมูลด้วยชุดคำสั่ง sql โดยแยกได้ 2 ลักษณะคือ

2.1 ถ้าต้องการเรียกดูข้อมูลเพียงอย่างเดียว จะอาศัยออบเจ็กต์ OleDbDataAdapter ร่วมกับชุดคำสั่งประเภทเลือกดูข้อมูล คือ คำสั่ง select ทำหน้าที่คิวรีข้อมูลออกมาผลการคิวรีที่จะได้จะเก็บอยู่ในออบเจ็กต์ DataSet

2.2 ถ้าต้องการจัดการข้อมูลในฐานข้อมูล เช่น การเพิ่ม แก้ไข หรือลบข้อมูล ซึ่งรวมถึงการเรียกดูข้อมูล จะอาศัยชุดคำสั่ง sql ประเภทอ่านและจัดการข้อมูล ร่วมกับออบเจ็กต์ OleDbCommand ผลการคิวรีที่ได้จะเก็บอยู่ในออบเจ็กต์ OleDbDataReader

2.6.4 Crystal Report

เป็นเครื่องมือสำหรับสร้างรายงาน พัฒนาโดยบริษัท Seagate มีรูปแบบการนำเสนอข้อมูลชนิดหนึ่งที่มีความสำคัญ เนื่องจากเป็นการนำข้อมูลจากฐานข้อมูลมาแสดงในรูปแบบที่ง่าย ซึ่งเป็นประโยชน์สำหรับผู้ใช้โดยเฉพาะผู้บริหาร CrystalReport สามารถสร้างรายงานได้หลากหลายรูปแบบและใช้Control “CrystalReportViewer” ในการแสดงผลรายงาน



รูปที่ 2.16 การสร้างรายงานด้วย Crystal Report [4]

การสร้างรายงานด้วย CrystalReport ทำได้ 3วิธีคือ

1. Using the Report Wizard สร้างรายงาน ใหม่โดยใช้ตัวช่วยสร้าง (Wizard)
2. As a Blank Report สร้างรายงานเปล่าแล้วนำมาปรับปรุงภายหลัง
3. From an Existing Report นำรายงานเดิมที่สร้างไว้แล้วมาปรับปรุง

การแสดงรายงานมี 3 รูปแบบดังนี้

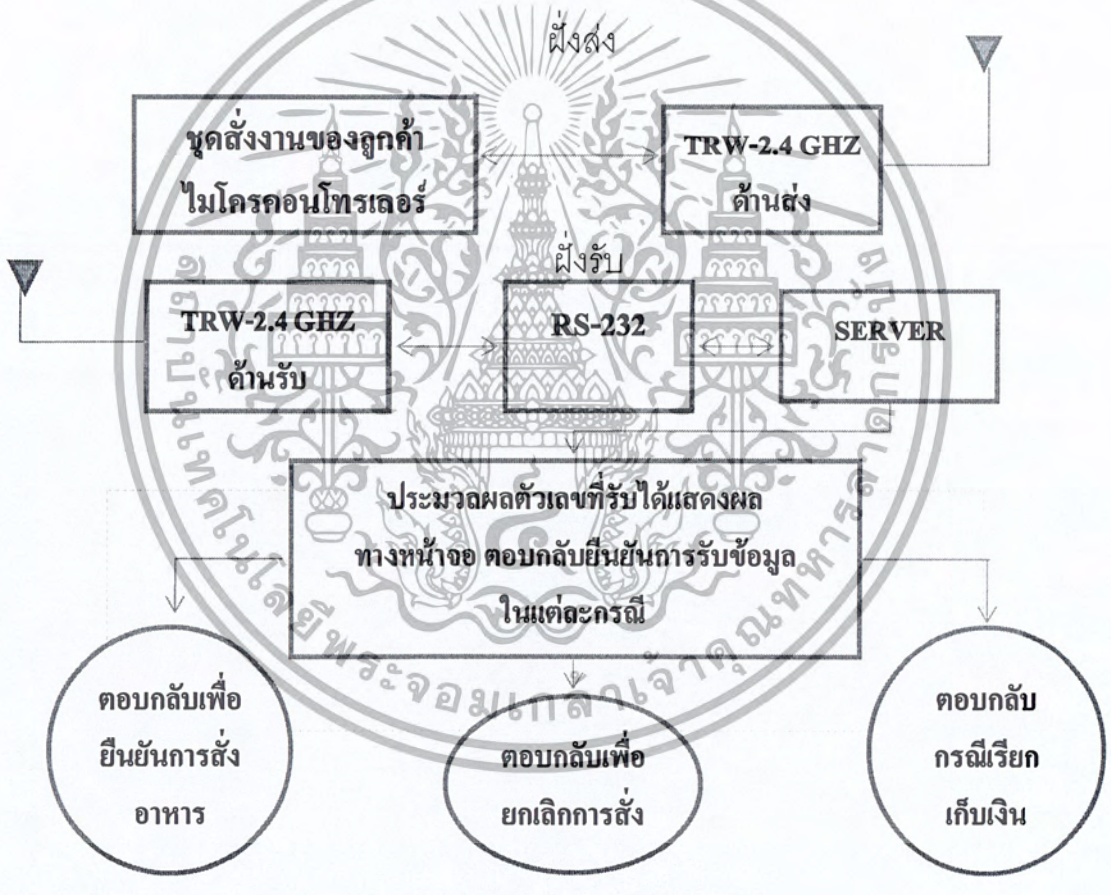
1. Standard รายงานตามมาตรฐานทั่วไป
2. Cross-Tab รายงานแบบ Cross-Tab ใช้ในการวิเคราะห์ข้อมูลที่ซับซ้อน
3. Mail Label สร้าง Label สำหรับใช้ปิดลงซองจดหมาย

บทที่ 3

การออกแบบและการจัดปริญญานิพนธ์

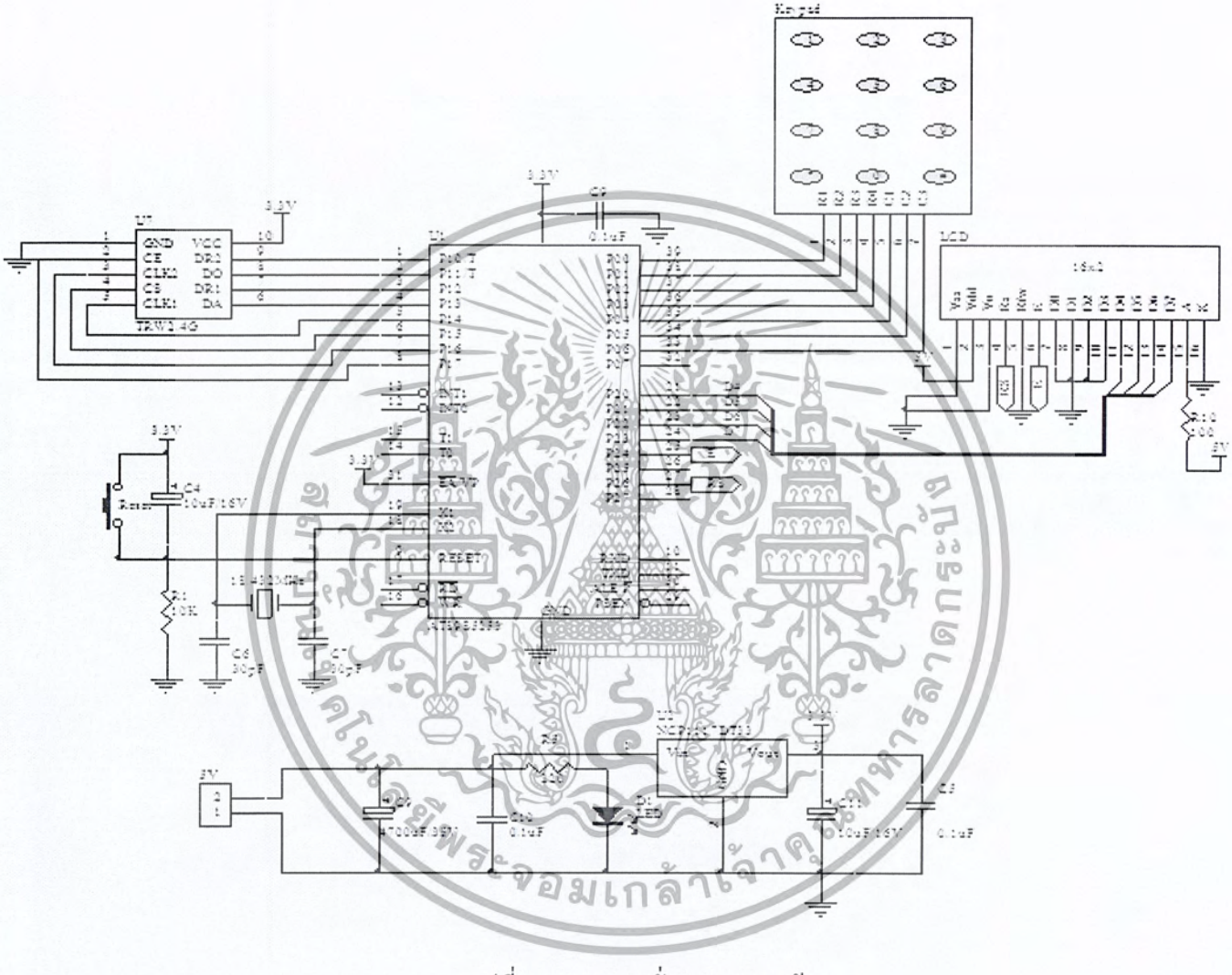
3.1 การออกแบบ

3.1.1 บล๊อกไดอะแกรมของเครื่องสั่งอาหารแบบไร้สาย



รูปที่ 3.1 บล็อกไดอะแกรมของเครื่องสั่งอาหารแบบไร้สาย

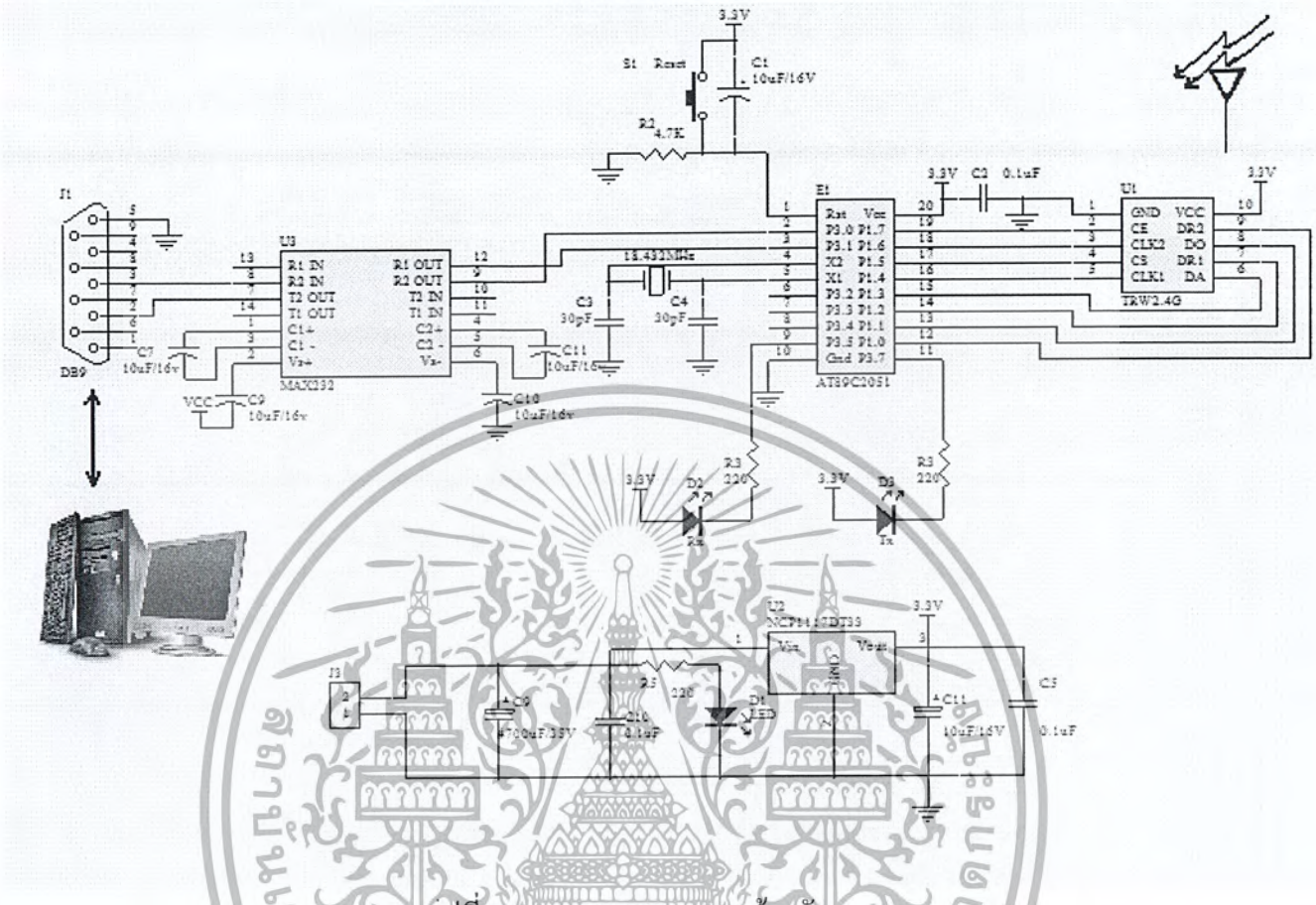
3.1.2 ชุดสั่งงานของลูกค้า



รูปที่ 3.2 วงจรชุดสั่งงานของลูกค้า

ประกอบด้วยจอ LCD คีย์แพดและไมโครคอนโทรเลอร์ เมื่อลูกค้ามาใช้บริการก็จะทำการ กดรหัสโดยดูจากเมนูอาหาร แล้วทำการกดรหัสผ่านคีย์แพดซึ่งตัวคีย์แพดจะทำการตรวจสอบที่แถวและหลักของคีย์เพดที่เชื่อมกับไมโครคอนโทรเลอร์ เพื่อนำค่าหมายเลขรหัสที่กดนั้นไปแสดงผลที่หน้าจอ LCD และเก็บค่าไว้ที่หน่วยความจำ เพื่อทำการส่งต่อไปยังวงจร TRW-2.4 GHZ

ในหน้าปัดของคีย์แพดนั้นนอกจากจะมีปุ่มกดหมายเลข 0-9 แล้วยังมีปุ่ม *ที่ทำหน้าที่ในการลบทีละตัวอักษร และปุ่ม # สำหรับใช้ส่งข้อมูล ส่วนในหน้าจอ LCD การแสดงผลมีรหัส 6 ตัวซึ่งมี 2 ช่วงๆละ 3ตัว โดยที่รหัส 3ตัวแรกจะเป็นรหัสของอาหาร อีก 3 ตัวต่อมาจะเป็นรหัสของจำนวนอาหารที่ลูกค้าสั่ง จะถูกควบคุมโดยไมโครคอนโทรเลอร์ นอกจากนี้วงจรชุดสั่งงานของลูกค้ายังสามารถแสดงผลการตอบกลับที่มาจาก Server แล้วแสดงผลที่จอ LCD ได้ด้วย



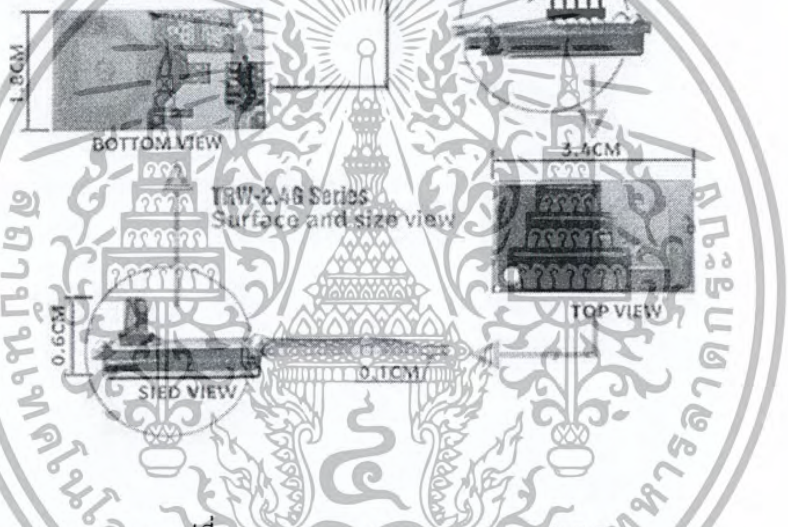
รูปที่ 3.3 วงจร TRW-2.4GHz ด้านรับ

3.1.3 วงจร TRW-2.4 GHz ฝั่งเครื่อง Server

ในส่วนของวงจรนี้จะทำการรับข้อมูลที่ส่งมาจาก TRW-2.4 GHz ด้านส่งซึ่งจำทำการส่งออกด้วย ขา Tx ของไมโครคอนโทรเลอร์ไปยังวงจรส่งผ่านพอร์ตอนุกรม(Max-232) ซึ่งไอซี Max-232 ทำหน้าที่แปลงระดับสัญญาณไฟฟ้า แล้วส่งข้อมูลผ่าน DB-9 เข้าสู่คอมพิวเตอร์เพื่อทำการตรวจสอบที่รหัสที่ลูกค้ากดมาต่อไป จากนั้นเมื่อ Server ทำการประมวลผลแล้วจะส่งข้อมูลที่แสดงการยืนยันสิ่งที่ลูกค้าต้องการกลับมาแล้วส่งต่อไปยังฝั่งลูกค้าต่อไป

3.1.4 การใช้งาน RF Module (TRW–2.4 GHz)

TRW-2.4GHz เป็น RF Module ที่ใช้ความถี่ 2.4 – 2.524 GHz สื่อสาร RF ด้วยเทคนิค GFSK มีความเร็วในการส่งข้อมูล 250 kbps ในช่วง 280 m แต่มีความเร็วสูงขึ้นเป็น 1 Mbps ในระยะทำงาน 150 m ใช้ไฟเลี้ยง Module ประมาณ 3V สามารถตั้งค่าให้เป็น transmitter หรือ receiver ก็ได้และมี mode การทำงานแบบ ShockBurst แบบ Duo Ceiver มี Antenna ภายใน Module



รูปที่ 3.4 RF Module(TRW-2.4GHz) [3]

3.1.4.1 การทำงานของ Mode ShockBurst

เมื่อต้องการส่งข้อมูล (ใช้ Pin CE , CLK1 , DATA)

- สั่ง Pin CE = 1 (เปิด ShockBurst Mode)
- สั่งข้อมูล (ADDRESS และ PAYLOAD) ทาง pin DATA โดยใช้ CLK1 เป็นตัว Synchronous Data ที่ส่งให้ TRW-2.4GHz
- TRW-2.4 GHz จะคำนวณ CRC ให้)ขนาดตามที่เรา config ไว้ (
- สั่ง pin CE = 0 (ปิด ShockBurst Mode) TRW-2.4 GHz ใส่ Preamble

และเริ่มส่ง RF จนกว่าจะเสร็จแล้วกลับมา Standby

เมื่อข้อมูลที่เข้ามาที่ตัวรับ)ใช้ Pin CE , DR1 , CLK1 , DATA)

- เมื่อมีข้อมูลเข้ามาที่ตัวรับ TRW-2.4 GHz จะทำการตรวจสอบ ADDR ที่ส่งมาค่าตรงกับของตัวมันที่เรา config ไว้หรือไม่ หากตรงจะทำงานต่อ หากไม่ตรงก็จะเพิกเฉยรอ package ใหม่
- ทำการ Check CRC ที่ส่งมาว่าถูกต้องหรือไม่ ถ้าถูกทำงานต่อหากผิดจะไม่มีการทำงานตอบกลับว่า Error ใดๆทั้งสิ้น
- TRW-2.4 GHz จะ set ขา DR1 เป็น 1 ดังนั้น microcontroller จึงคอยวน check ขา DR1 ว่าเป็น 1 เมื่อใดแสดงว่า พร้อมแล้ว
- Microcontroller ต้องสร้าง clock มาที่ CLK1 เพื่ออ่านข้อมูลจากทาง DATA ออกไป ถ้าครบแล้ว DR1 จะกลายเป็น 0 อีกครั้ง

ถ้าหากใช้ Duo Ceiver (คือ สามารถกำหนดให้ TRW-2.4 GHz ตัวนี้มีช่องรับ 2 ทาง (การแบ่งการทำงานออกเป็น 2 channel แต่ละ channel มี ADDR ของมันเอง โดยที่ใช้ขาสัญญาณไม่ตรงกัน

Rx1 ใช้ pin (DR1 , CLK1 , DATA)

Rx2 ใช้ pin (DR2 , CLK2 , DOUT2)

3.1.4.2 การ Config TRW-2.4 GHz

- สั่งให้ขา CS =1 (เริ่มการ config) แต่ขา CE ต้องเป็น 0 ด้วยเพราะจะไม่อนุญาตให้ config และรับส่งข้อมูลพร้อมกันได้
- สั่งขา DATA ที่ใช้ config 144 bit เข้าไปทาง DATA pin เริ่มจาก MSB และใช้ CLK1 ในการ Synchronous
- เมื่อเสร็จสิ้นให้ set CS = 0 เหมือนเดิม

ตารางที่ 3.1 วิธีการ Config TRW-2.4GHz

ตำแหน่ง bit	จำนวน bit	ชื่อ	คำอธิบาย
143:120	24	Test	สงวนใช้ Test
119:112	8	DATA2_1	ความยาวของPayloadคำนวนจาก
111:104	8	DATA1_W	Payload=256-ADDR_W-CRC
103:64	40	ADDR2	ค่าของ ADDR ของตัวรับ ตั้งค่า
63:24	40	ADDR1	ได้ทั้ง channel 1 และ 2 มีขนาด สูงสุดได้ 5 byte
23:18	6	ADDR_W	กำหนดความยาวของ ADDR ได้
17	1	CLC_L	ความยาวของ CRC '0'=8 bit
16	1	CRC_EN	กำหนดใช้หรือไม่ใช้ CRC
15	1	RX2_EN	กำหนดใช้กี่ channel (รับ)
14	1	CM	เปิดปิด Shockburst Mode
13	1	RFR_SB	กำหนด Data Rate '0'=250 kpbs '1'=1 Mpbs
12:10	3	XO_F	กำหนดเกี่ยวกับ CRYSTAL แต่ กำหนดตายตัวเป็น 011
7:1	7	RF_CH	กำหนดความถี่ที่ใช้ในการสั่ง คำนวณดังนี้ RF = 2400+RF_CH MHz
0	1	RE_EN	กำหนดให้ TRW เป็น Tx-Rx '0'(Tx) '1'(Rx)

ดังนั้นในวงจร TRW-2.4 GHz ที่ใช้นั้นจึงได้กำหนดค่าต่างๆไว้ดังนี้

1. ความยาวข้อมูลใน channel 1 และ channel 2 คือ 8 bit
2. Address Channel 2 ทั้ง 5 bytes คือ (0x00),(0x00),(0x00),(0x00),(0x02)
   Address Channel 1 ทั้ง 5 bytes คือ (0x00),(0x00),(0x00),(0x00),(0x00)
3. กำหนดให้ใช้ CRC ขนาด 16 bit
4. ใช้ 1 channel ในการรับ
5. เปิด ShockBurst Mode
6. Data Rate คือ 250 kbps
7. RF Power คือ 0 dbm
8. ความถี่ในการส่งคือ 2.410 MHz

3.1.5 ส่วนของเครื่อง Server

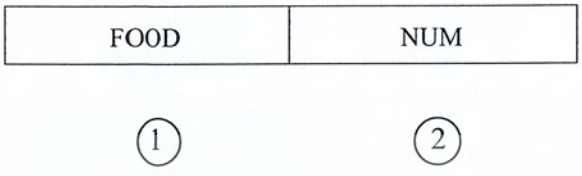
หน้าที่ของเครื่อง Server

1. รับข้อมูลจากลูกค้าแต่ละโต๊ะ โดยรับผ่านทาง Serial Port
2. แสดงรายละเอียดของข้อมูลที่ลูกค้าแต่ละโต๊ะสั่ง โดยแสดงผ่านจอแสดงผลที่สร้างขึ้น
3. ส่งค่ากลับเพื่อยืนยันการส่งข้อมูลของลูกค้าแต่ละโต๊ะ ว่าข้อมูลที่ถูกส่งมานั้นเป็นข้อมูลแบบใด
4. สามารถพิมพ์ใบเสร็จเพื่อใช้สำหรับการเรียกเก็บเงินจากลูกค้าได้

โปรแกรมของเครื่อง Server มีลักษณะการทำงานดังนี้

เมื่อโปรแกรมของเครื่อง Server อยู่ในสถานะพร้อมทำงาน โปรแกรมจะทำการรอรับข้อมูล ซึ่งเป็นส่วนของสมาชิกของลูกค้าเป็นรหัส 4 ตัว หลังจากนั้น จะรอรับข้อมูลที่ถูกส่งเข้ามาทาง Serial Port โดยลักษณะของข้อมูลที่ต้องการของโปรแกรมจะเป็นชุดตัวเลข 6 ตัว แล้วจึงนำชุดตัวเลขไปประมวลผลแล้วแสดงผลของข้อมูลที่รับมาได้ออกทางหน้าจอแสดงผลที่สร้างขึ้น และมีการตอบกลับเพื่อใช้สำหรับยืนยันว่าเครื่อง Server นั้น ได้รับข้อมูลจากลูกค้าแล้ว โดยการส่งการยืนยันไปยังโต๊ะของลูกค้าที่ส่งข้อมูลมา

รูปแบบของชุดคำสั่ง



ซึ่งลักษณะของข้อมูลที่เป็นชุดตัวเลข 6 ตัวนั้น โปรแกรมจะทำการแบ่งชุดตัวเลขออกเป็น 2 ชุด ชุดละ 3 ตัว ซึ่งจะแบ่งได้ดังนี้

- ชุดที่ 1 ใช้สำหรับแสดงรหัสอาหารที่ลูกค้าสั่ง ชุดตัวเลขนี้จะถูกนำไปเปรียบเทียบกับฐานข้อมูล แล้วดึงค่า ชื่อรายการอาหาร และ ราคาต่อหน่วย จากฐานข้อมูลมาแสดงทางจอและนำไปใช้สำหรับการคิดราคาอาหารของโต๊ะนั้น

รูปที่ 3.5 Flowchart การทำงานของ Server

- ชุดที่ 2 ใช้สำหรับแสดงจำนวนอาหารต่อหนึ่งรายการอาหารที่ลูกค้าสั่ง โดยชุดตัวเลขนี้จะถูกนำไปคูณกับราคาต่อหน่วยของรายการอาหารที่ลูกค้าสั่งเข้ามา เพื่อนำไปใช้สำหรับการคิดราคาอาหารของโต๊ะนั้น

ตัวอย่างการใช้ชุดคำสั่ง

รูปที่ 3.6 ตัวอย่างชุดคำสั่งของการสั่งอาหาร

นี่เป็นชุดคำสั่งที่ส่งมาโดยลูกค้าผ่านอุปกรณ์จากโต๊ะของลูกค้า จากภาพเป็นการส่งคำสั่งมาเพื่อสั่งอาหารที่มีรหัส 007 และจำนวนที่ต้องการคือ 1 จาน

จากนั้นโปรแกรมจะทำการพิจารณาว่ารหัสอาหารนั้นมีการใส่รหัสเกินจากที่กำหนดไว้ฐานข้อมูลของเราหรือไม่ ถ้ารหัสอาหารเกินหรือไม่มีอยู่ในฐานข้อมูลโปรแกรมก็จะไม่มีการประมวลผลในขั้นตอนต่อไป แล้วโปรแกรมจะส่งข้อความกลับไปยังอุปกรณ์ที่อยู่บนโต๊ะของของลูกค้าว่า Order Error จึงทำให้ลูกค้าต้องใส่รหัสอาหารใหม่ให้ถูกต้องโปรแกรมจึงจะทำการประมวลผลในขั้นตอนต่อไปได้

3.2 เครื่องมือที่ใช้ในการทดลอง

1. OSCILLOSCOPE

2. SPECTRUM ANALYSER

3. METER

3.3 การจัดเก็บผลการทดลอง

1. ใช้ OSCILLOSCOPE วัดสัญญาณที่ขาต่างๆของอุปกรณ์รับ-ส่งสัญญาณ TRW-2.4 GHz ที่ใช้ในชุดอุปกรณ์

2. ใช้ SPECTRUM ANALYSER วัดสเปกตรัมของอุปกรณ์รับ-ส่งสัญญาณ TRW-2.4 GHz

บทที่ 4

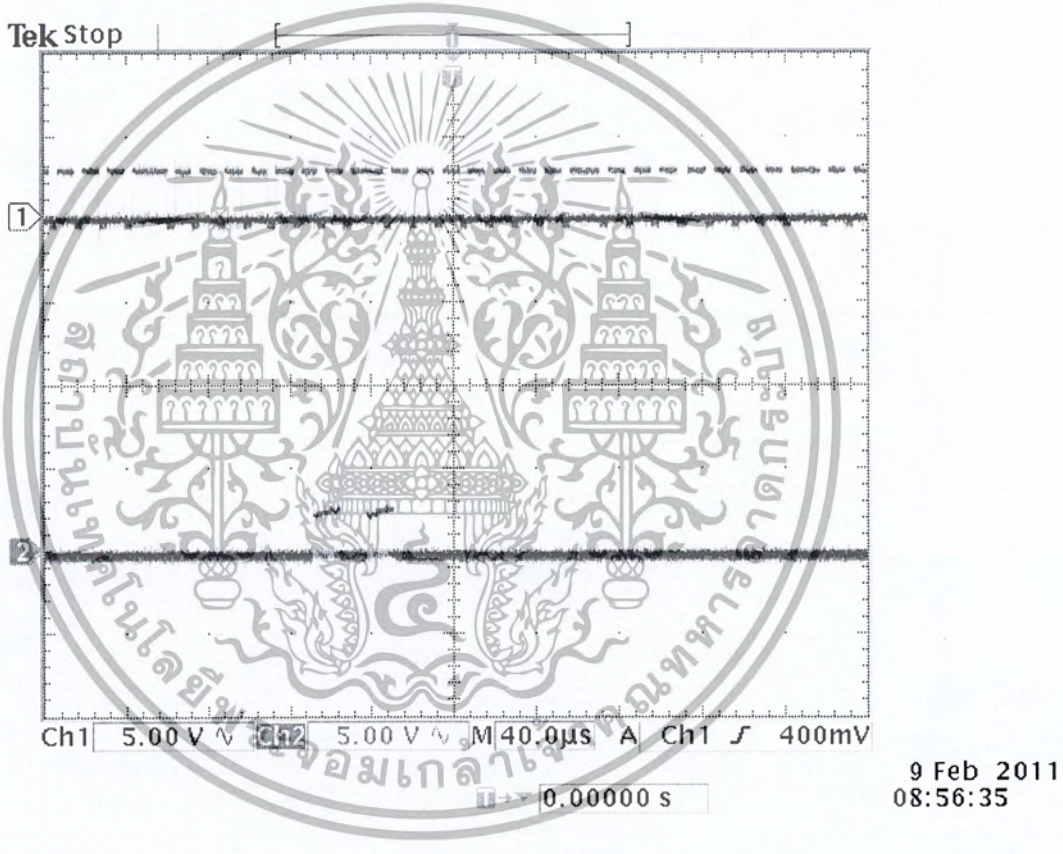
ผลการทดลอง

4.1 การทดลองเรื่อง การวัดสัญญาณการรับ – ส่งระหว่าง TRW-2.4GHz ด้านส่งและด้านรับ

1. ทำการวัดสัญญาณโดยออสซิโลสโคป ที่ขา Data เทียบกับ Clock ของ TRW-2.4GHz ฝั่งชุดสั่งงานของลูกค้า (ด้านส่ง)
2. ทำการวัดสัญญาณโดยออสซิโลสโคป ที่ขา Data เทียบกับ CS ของ TRW-2.4GHz ฝั่งชุดสั่งงานของลูกค้า (ด้านส่ง)
3. ทำการวัดสัญญาณโดยออสซิโลสโคป ที่ขา Data เทียบกับ CE ของ TRW-2.4GHz ฝั่งชุดสั่งงานของลูกค้า (ด้านส่ง)
4. ทำการวัดสัญญาณโดยออสซิโลสโคป ที่ขา Data เทียบกับ Clock ของ TRW-2.4GHz ฝั่ง Server (ด้านรับ)
5. ทำการวัดสัญญาณโดยออสซิโลสโคป ที่ขา DR1 เทียบกับ Data ของ TRW-2.4GHz ฝั่งชุดสั่งงานของลูกค้า (ด้านรับ)
6. ทำการวัดสัญญาณโดยออสซิโลสโคป ที่ขา Data เทียบกับ CS ของ TRW-2.4GHz ฝั่งชุดสั่งงานของลูกค้า (ด้านรับ)
7. ทำการวัดสัญญาณโดยออสซิโลสโคป ที่ขา Data เทียบกับ CE ของ TRW-2.4GHz ฝั่งชุดสั่งงานของลูกค้า (ด้านรับ)

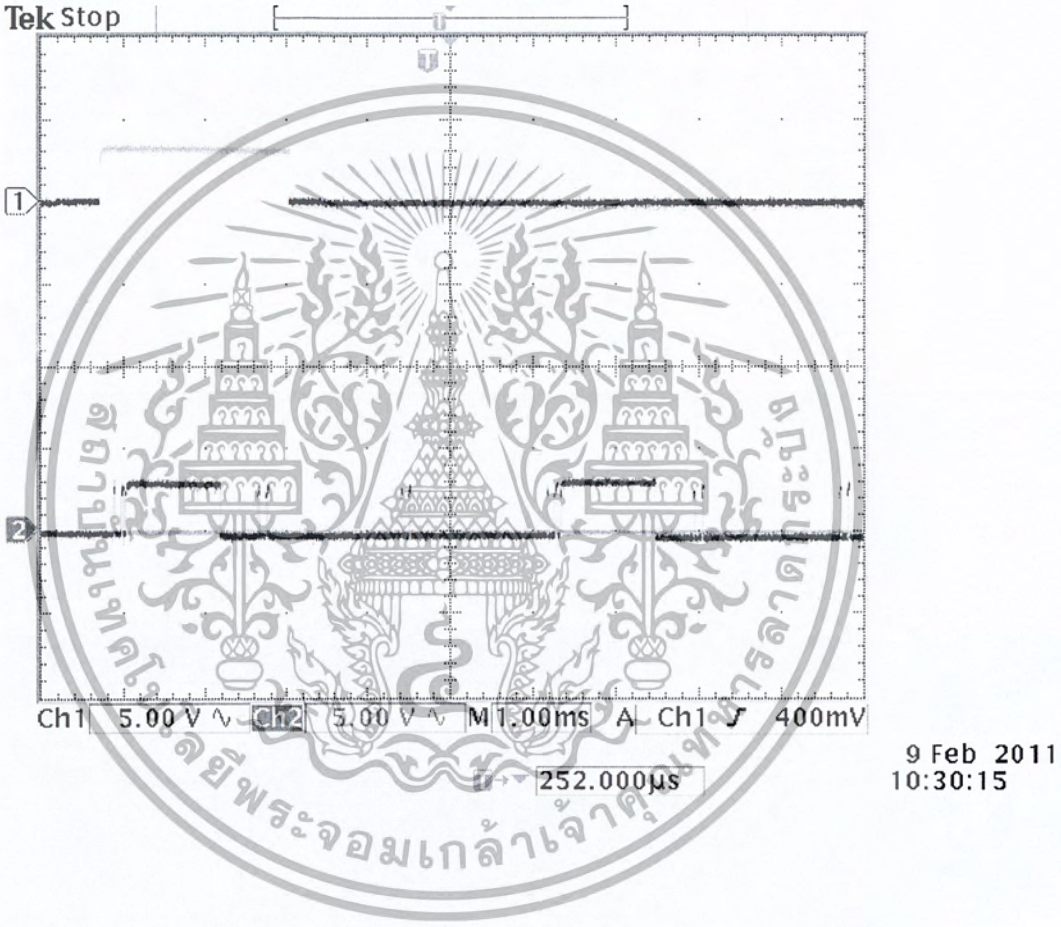
ผลการทดลอง

1. ทำการวัดสัญญาณโดยออสซิโลสโคป ที่ขา Data เทียบกับ Clock ของ TRW-2.4GHz ฝั่งชุดสั่งงานของลูกค้า (ด้านส่ง)



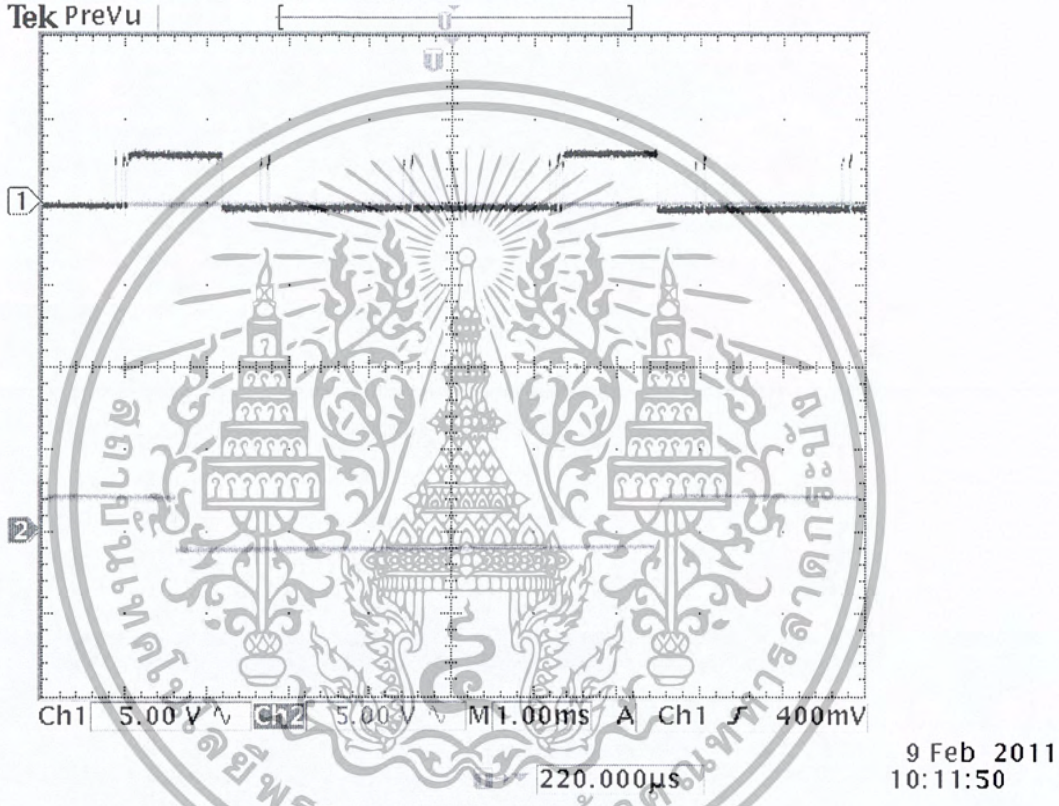
รูปที่ 4.1 สัญญาณที่ขา Clock(Channel 1) เทียบกับสัญญาณที่ขา Data (Channel 2)

2. ทำการวัดสัญญาณโดยออสซิโลสโคป ที่ขา Data เทียบกับ CS ของ TRW-2.4GHz ฝั่งชุดสั่งงานของลูกค้า (ด้านส่ง)



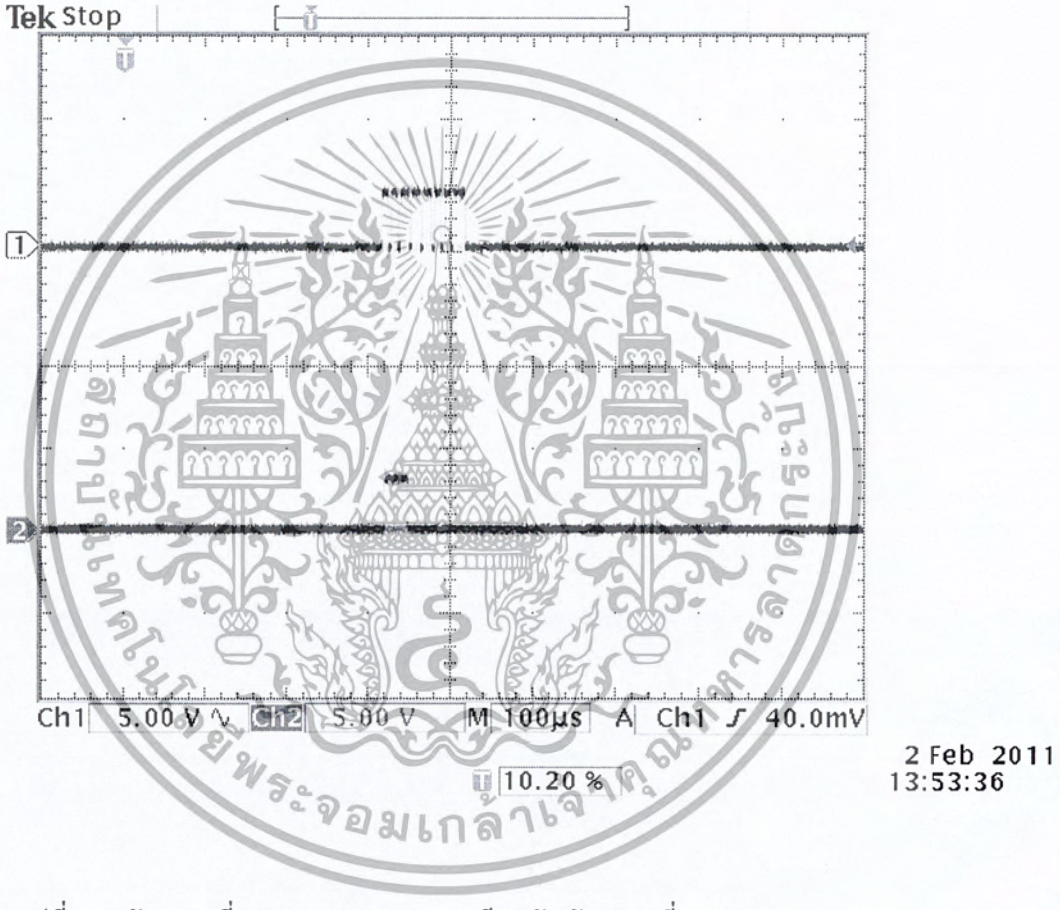
รูปที่ 4.2 สัญญาณที่ขา CS (Channel 1) เทียบกับสัญญาณที่ขา Data (Channel 2)

3. ทำการวัดสัญญาณโดยออสซิโลสโคป ที่ขา Data เทียบกับ CE ของ TRW-2.4GHz ฝั่งชุดสั่งงานของลูกค้า (ด้านส่ง)



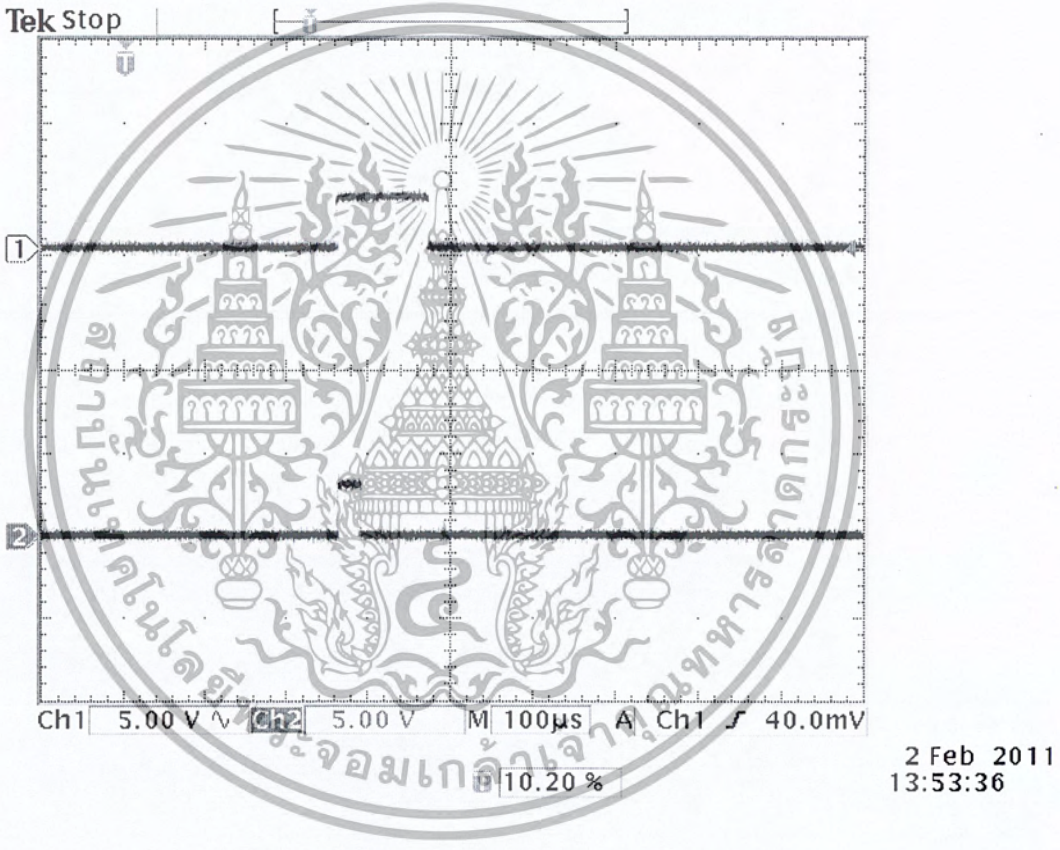
รูปที่ 4.3 สัญญาณที่ขา Data(Channel l) เทียบกับสัญญาณที่ขา CE (Channel 2)

4. ทำการวัดสัญญาณโดยออสซิโลสโคป ที่ขา Data เทียบกับ Clock ของ TRW-2.4GHz ฝั่ง Server (ด้านรับ)



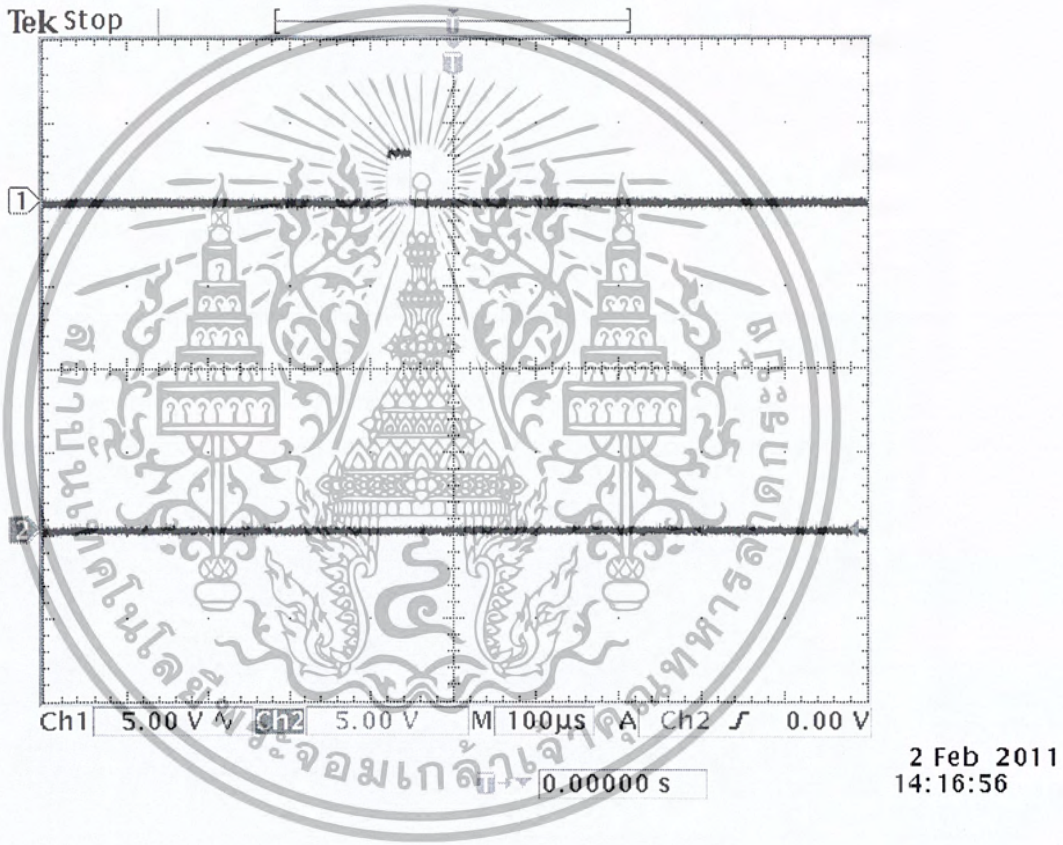
รูปที่ 4.4 สัญญาณที่ขา Clock(Channel 1) เทียบกับสัญญาณที่ขา Data (Channel 2)

5. ทำการวัดสัญญาณโดยออสซิโลสโคป ที่ขา DR1 เทียบกับ Data ของ TRW-2.4GHz ฝั่งชุดสั่งงานของลูกค้า (ด้านรับ)



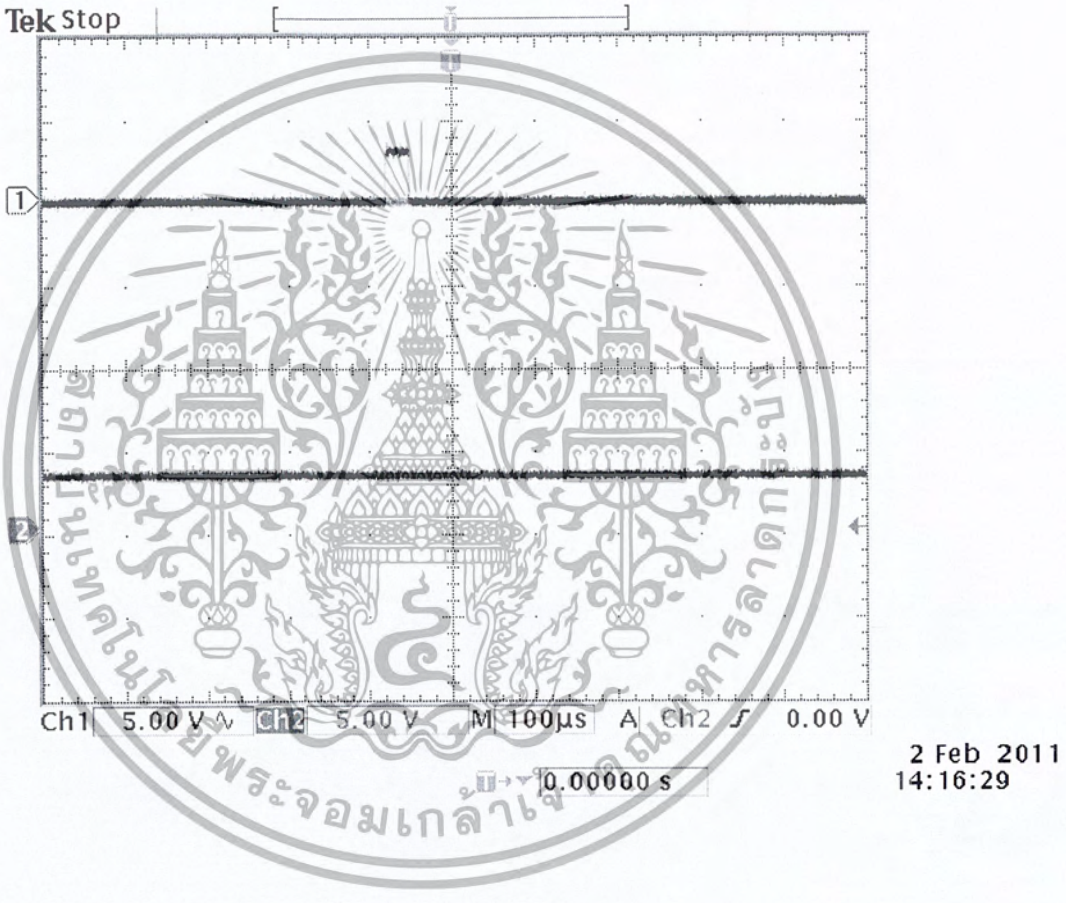
รูปที่ 4.5 สัญญาณที่ขา DR1(Channel 1) เทียบกับสัญญาณที่ขา Data (Channel 2)

6. ทำการวัดสัญญาณโดยออสซิโลสโคป ที่ขา Data เทียบกับ CS ของ TRW-2.4GHz ฝั่งชุดสั่งงานของลูกค้า (ด้านรับ)



รูปที่ 4.6 สัญญาณที่ขา Data(Channel 1) เทียบกับสัญญาณที่ขา CS (Channel 2)

7. ทำการวัดสัญญาณโดยออสซิโลสโคป ที่ขา Data เทียบกับ CE ของ TRW-2.4GHz ฝั่งชุดสั่งงานของลูกค้า (ด้านรับ)

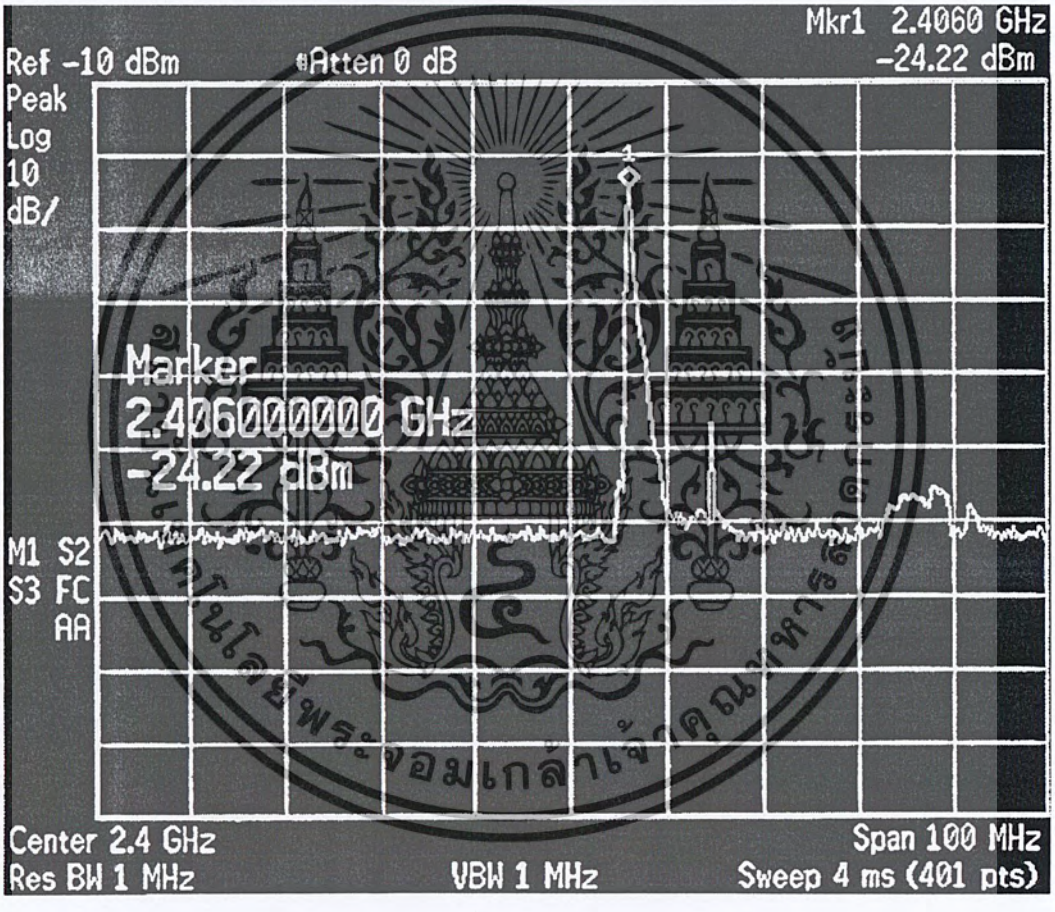


รูปที่ 4.7 สัญญาณที่ขา Data(Channel 1) เทียบกับสัญญาณที่ขา CE (Channel 2)

4.2 การทดลองเรื่อง การวัดความถี่และกำลังที่ใช้ส่งสัญญาณของ TRW-2.4GHz

ทำการทดลองวัดความถี่และกำลังส่งของสัญญาณ โดยใช้เครื่อง Spectrum Analyzer โดยทำการสังเกตและบันทึกผลการทดลอง

ผลการทดลอง



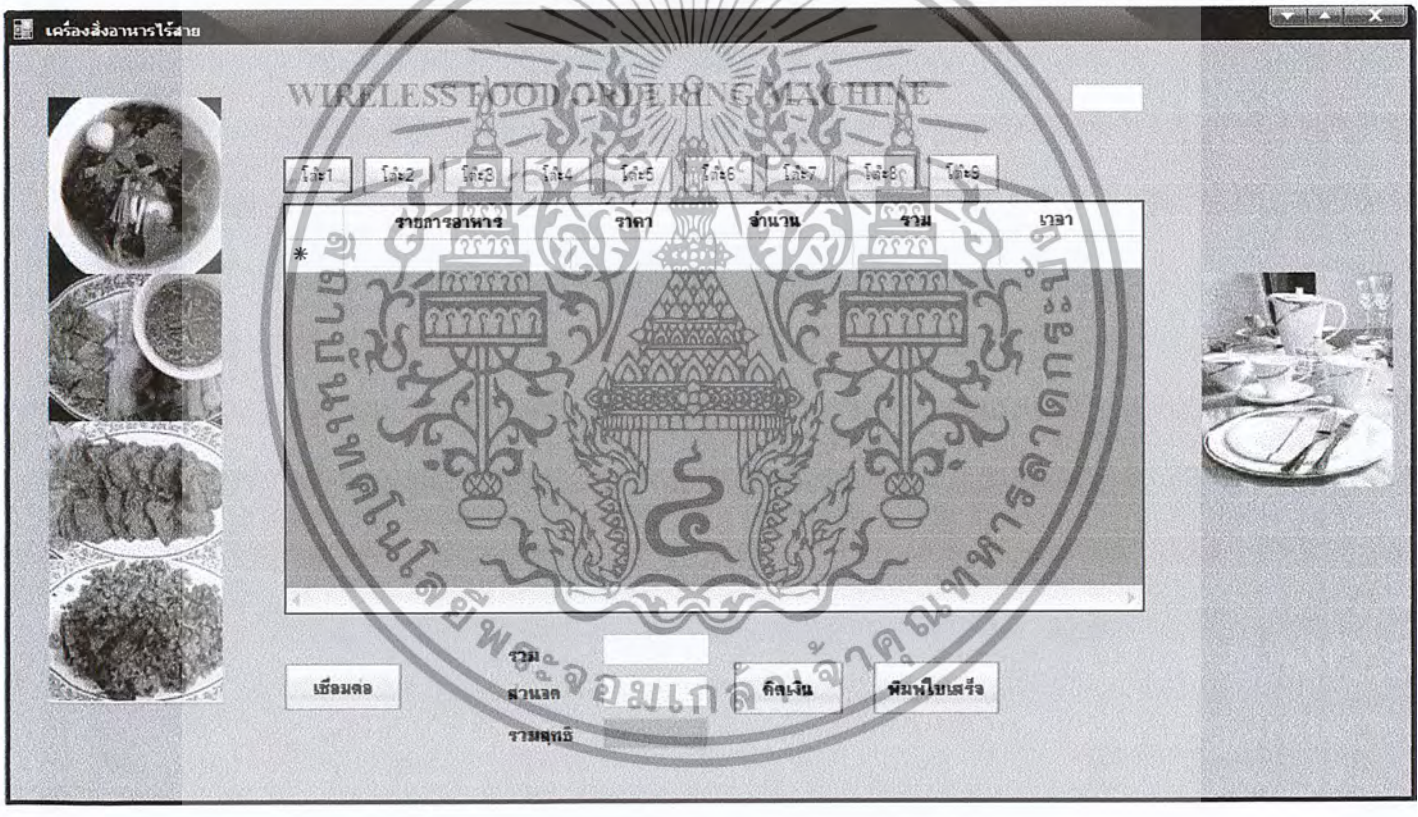
รูปที่ 4.8 ผลการวัดความถี่และกำลังของสัญญาณที่ใช้ส่งของ TRW - 2.4GHz

จากรูป 4.8 จะเห็นว่าความถี่ที่วัดได้ คือ 2.40 GHz และ วัดกำลังส่งของสัญญาณ ได้เท่ากับ -24.22 dBm

4.3 การทดลองเรื่อง การแสดงผลของเครื่อง Server และการแสดงผลตอบรับที่จอ LCD

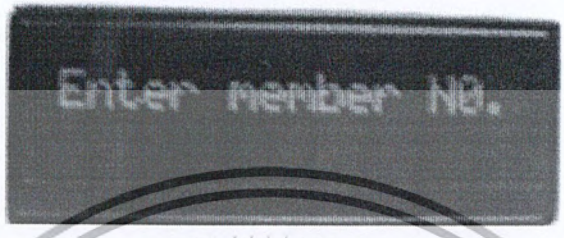
ขั้นตอนการทดลอง

1. ทำการเปิดโปรแกรม ที่สร้างโดยโปรแกรม Visual Studio ด้วยภาษา C# หน้าฟอร์มก็จะปรากฏขึ้นดังรูป จากนั้นทำการคลิกปุ่ม เชื่อมต่อ เพื่อทำการเชื่อมต่อเข้ากับ Serial Port เครื่องคอมพิวเตอร์ Server จะอยู่ในสถานะพร้อมรับข้อมูล แต่ละโต๊ะจะทำการแสดงรายการอาหาร ,ราคา ,จำนวน ,รวม และ เวลา เป็นเวลา 5 วินาทีโดยเริ่มจากโต๊ะ 1-9 ตามลำดับ หากมีโต๊ะใดสั่งอาหารมาจะแสดงรายการอาหารที่โต๊ะนั้นทันที



รูปที่ 4.9 หน้าจอแสดงผลของเครื่อง Server ในสถานะพร้อมใช้งาน

2. เปิดเครื่องทางด้านชุดสั่งงานของลูกค้า ให้อยู่ในสภาวะพร้อมทำงาน และเชื่อมต่อเครื่องด้าน Server เข้ากับ คอมพิวเตอร์ ผ่านทาง Serial Port



รูปที่ 4.10 หน้าจอ LCD เมื่อชุดสั่งงานด้านลูกค้าอยู่ในสถานะพร้อมใช้งาน

3. ทำการกดคีย์แพดทางด้านชุดสั่งงานของลูกค้าโดยทำการกดรหัสสมาชิกก่อนใช้งาน ลูกค้าที่เป็นสมาชิกจะได้รับส่วนลด 10%

รูปที่ 4.11 หน้าจอ LCD เมื่อทำการกดรหัสสมาชิก

4. กด OK แล้วป้อน ข้อมูลรายการอาหารที่มีรหัสเท่ากับ 001001

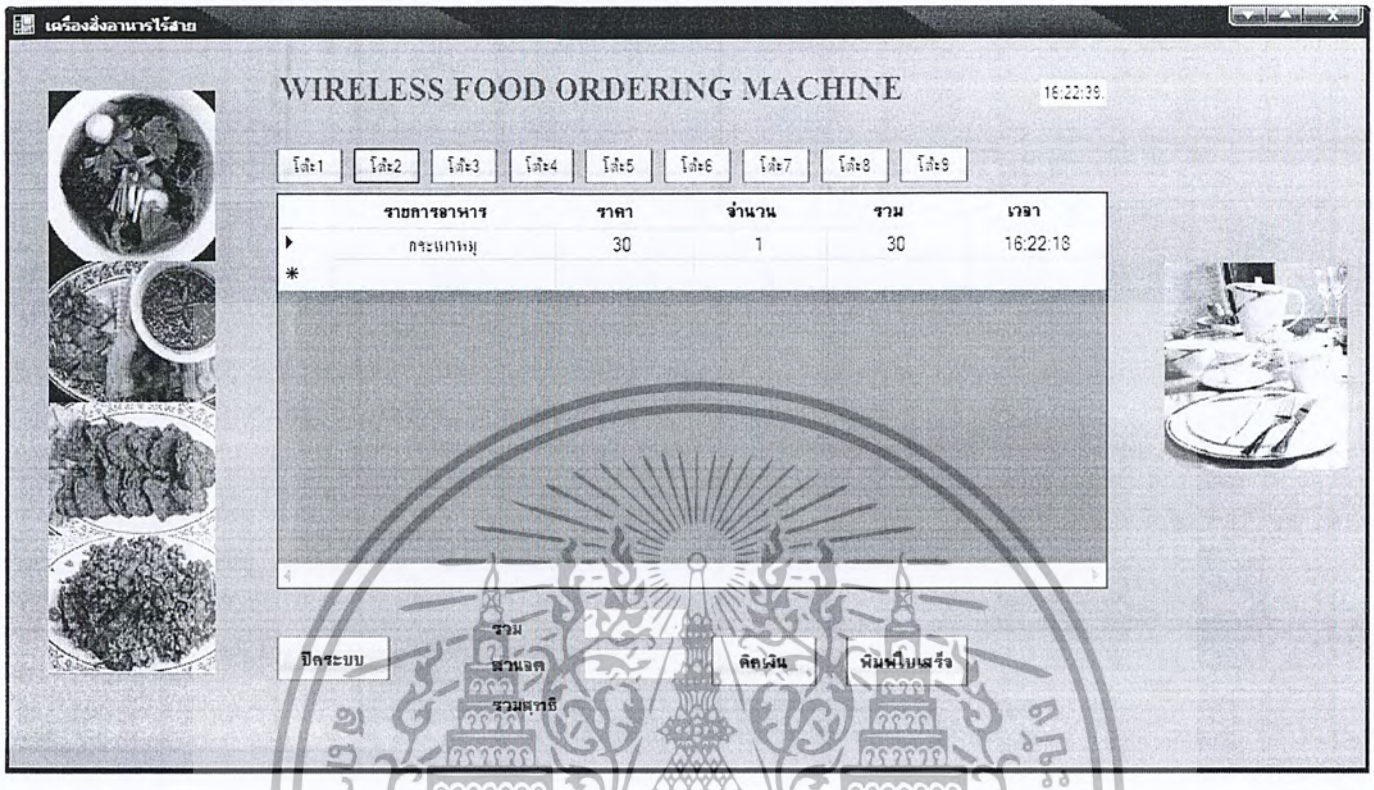
รูปที่ 4.12 หน้าจอ LCD เมื่อทำการกดรหัสอาหาร

5. ทำการกด # เพื่อทำการส่งข้อมูล หน้าจอจะปรากฏคำว่า Order Received

รูปที่ 4.13 หน้าจอ LCD เมื่อทำการส่งรหัสอาหารสมบูรณ์ถูกต้อง

6. ถ้ากดตัวเลขไม่ครบหรือรหัสไม่ถูกต้อง หน้าจอ LCD จะปรากฏคำว่า Order Error

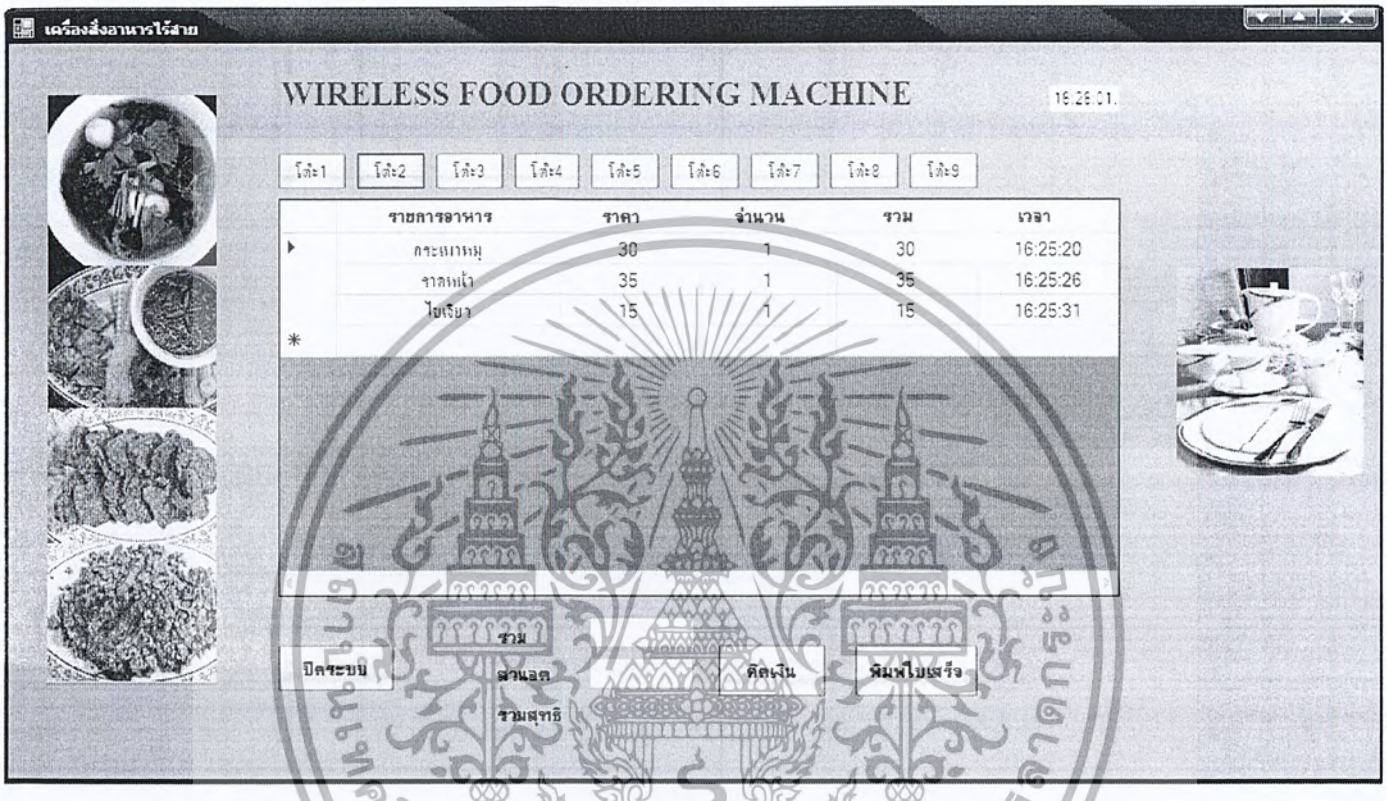
รูปที่ 4.14 หน้าจอ LCD เมื่อทำการส่งหัสอาหารไม่ถูกต้อง



รูปที่ 4.15 หน้าจอแสดงผลของเครื่องคอมพิวเตอร์ Server หลังจากรับข้อมูล 001001 อย่างถูกต้อง

จะเห็นได้ว่าหน้าจอแสดงผลของส่วน Server จะแสดงหน้ารายการอาหารของโต๊ะที่ 2 (การทดลองนี้ทดลองโดยการส่งรหัสมาจากโต๊ะที่ 2) และทำการเพิ่มรายการอาหารในฐานข้อมูลที่มีรหัสเท่ากับ 001 = กระเพราหมู และราคาต่อหน่วยเท่ากับ 30 บาท สั่งเป็นจำนวน 1 หน่วย

7. ทำการสั่งอาหารเพิ่มโดยกด รหัส 002001 และ 003001 ทำการส่งตามลำดับ

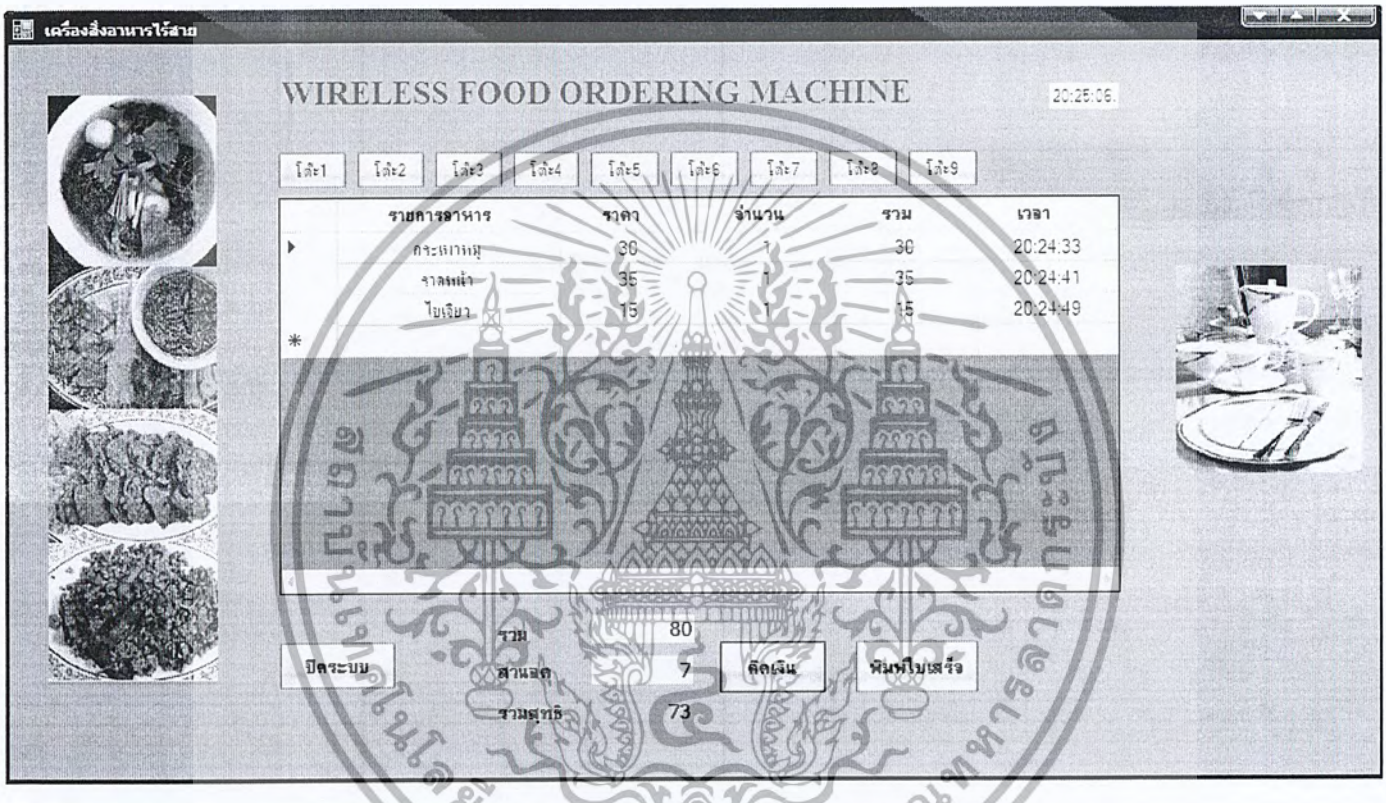


รูปที่ 4.16 หน้าจอแสดงผลของเครื่องคอมพิวเตอร์ Server หลังจากรับข้อมูล 001001 , 002001 003001

จากรูป 4.1.6 หน้ารายการอาหารของโต๊ะที่ 2 จะถูกเพิ่มรายการอาหารเข้าไปอีก 2 รายการ ตามลำดับที่ชุดสั่งงานฝั่งลูกค้าส่งมา

8. ทำการคิดเงินโดยการคลิก ปุ่ม คิดเงิน

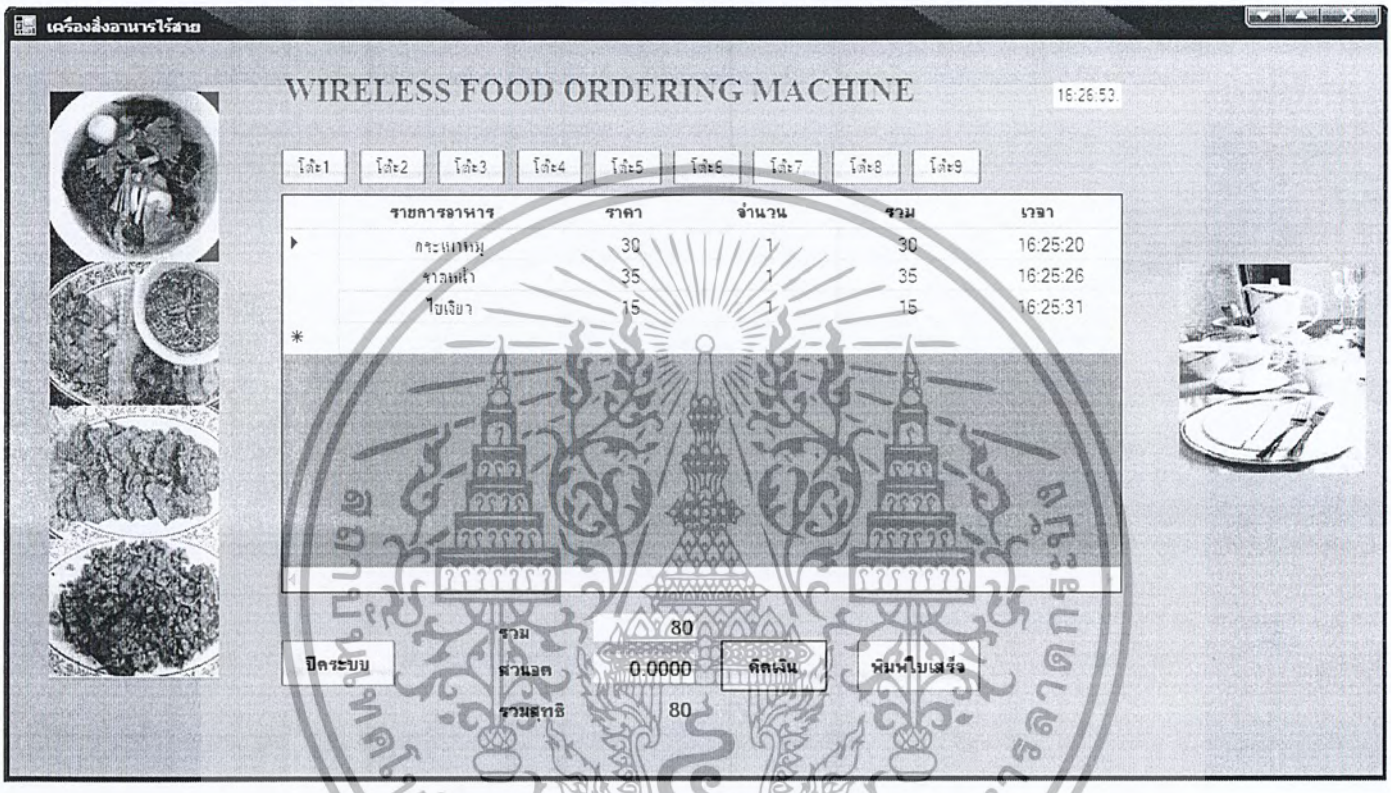
กรณีที่ 1 ลูกค้าเป็นสมาชิกจะได้รับส่วนลด 10 %



รูปที่ 4.17 หน้าจอแสดงผลของเครื่องคอมพิวเตอร์ Server หลังจากทำการกดปุ่ม“ คิดเงิน ”

จากรูป 4.17 รายการอาหารตรงส่วนช่องรวม , ส่วนลด และ รวมสุทธิ จะถูกแสดงออกมา

กรณีที่ 2 ลูกค้าที่ไม่ได้เป็นสมาชิก



รูปที่ 4.18 หน้าจอแสดงผลของเครื่องคอมพิวเตอร์ Server หลังจากทำการกดปุ่มคิดเงิน

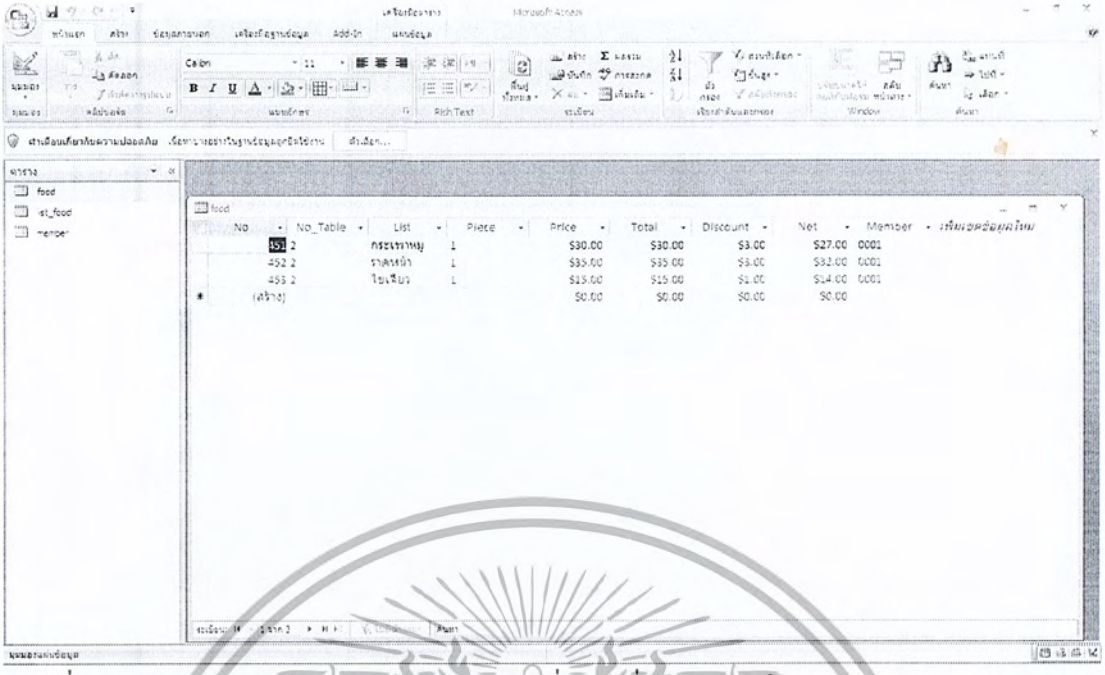
เมื่อต้องการคิดเงินให้กดรหัส 999 999 เพื่อสั่งให้พนักงานคิดเงินและทำการ “ พิมพ์ใบเสร็จ ”

ใบเสร็จรับเงิน

15/2/2554

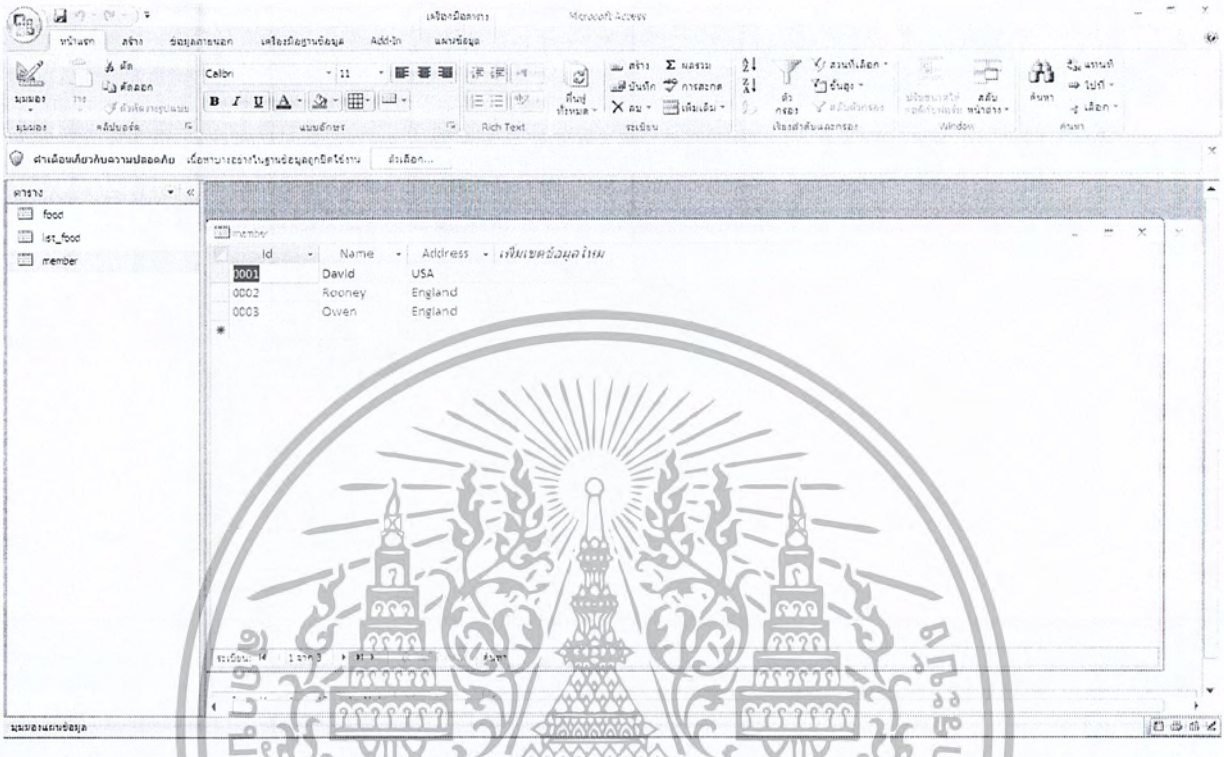
รายการอาหาร	ราคา	จำนวน	รวม
กระเพาหมู	B30.00	1	B30.00
ราดหน้า	B35.00	1	B35.00
ไขเจียว	B15.00	1	B15.00
		รวม	B80.00
		ส่วนลด	B.00
		ราคาสุทธิ	B80.00

รูปที่ 4.19 ใบเสร็จที่ใช้สำหรับเรียกเก็บเงินลูกค้า

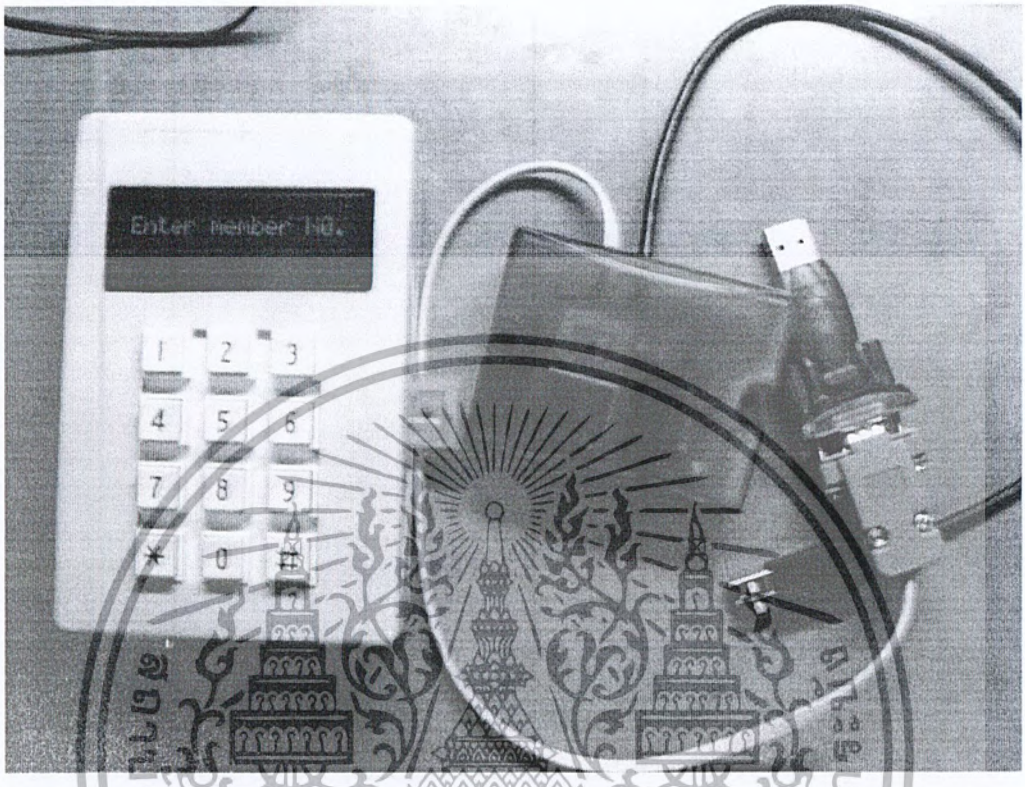


รูปที่ 4.20 ส่วนของฐานข้อมูลของรายการอาหารที่ลูกค้าสั่งจะถูกบันทึกลงในโปรแกรม Microsoft Office Access

รูปที่ 4.21 ส่วนของฐานข้อมูลของรายการอาหารซึ่งใช้อ้างอิงในการสั่งอาหารของลูกค้า



รูปที่ 4.22 ส่วนของฐานข้อมูลรายชื่อสมาชิกลูกค้า



รูปที่ 4.23 ชุดอุปกรณ์เครื่องสั่งอาหารแบบไร้สาย

บทที่ 5
สรุปผลและข้อเสนอแนะ

5.1 สรุปผล

ได้จัดทำในส่วนของชุดสั่งงานลูกค้าให้แสดงข้อความที่กำหนด และ รับค่าตัวเลขตามการสั่งงานของลูกค้า โดยเขียนโปรแกรมควบคุมการทำงานด้วยภาษา C จากการทดลองผลที่ได้ ได้ตามที่ผู้จัดทำได้ออกแบบไว้ โดย LCD แสดงค่าได้อย่างถูกต้อง KEYPAD สามารถรับค่าได้ ในส่วนของโปรแกรม Visual Studio ซึ่งเป็นโปรแกรมสำหรับสร้างฐานข้อมูลนั้น ผู้จัดทำได้ออกแบบในส่วนแสดงผลทางจอคอมพิวเตอร์ ซึ่งเป็นโปรแกรมที่มีการจัดการในส่วนต่างๆ เช่น แสดงข้อมูลที่ลูกค้าสั่งอาหารรายการต่างๆ รวมไปถึงการคิดเงินทั้งในรูปแบบราคาเต็ม และราคาที่มีส่วนลดเนื่องจากการเป็นสมาชิกของทางร้าน

5.2 ข้อเสนอแนะ

การใช้ LCD ชนิดสองบรรทัดแสดงผล ทำให้การเขียนโปรแกรมเพิ่มรายละเอียด เช่น อาหาร ไม่ใส่ผัก หรือ ไม่เผ็ดมากมีข้อจำกัด ดังนั้นถ้าหากมีงบประมาณที่สูงขึ้นการใช้ LCD ที่แสดงผลได้มากขึ้นจะทำให้ สามารถระบุความละเอียดของข้อมูลอาหารได้มากขึ้น ส่วนตัวรับ-ส่ง TRW – 2.4 GHz นั้นถ้าส่งสัญญาณโดยมีสิ่งกีดขวางต่างๆจะมีผลทำให้ประสิทธิภาพการรับ-ส่งลดลง ซึ่งการใช้งานจริงในเชิงธุรกิจ ควรใช้ตัวรับส่งที่มีประสิทธิภาพมากกว่านี้

บรรณานุกรม

[1] ทีมงานสมาร์ทเลิร์นนิ่ง.*เรียนรู้ไมโครคอนโทรลเลอร์*,กรุงเทพฯ : ห้างหุ้นส่วนสามัญสมาร์ทเลิร์นนิ่ง, 2552.

[2] Adisak Chinawong."บทความของ MCS - 51."
http://www.adisak51.com/column.html

[3] ETT. "TRW 2.4 GHz." http://www.etteam.com/product/1702.html

[4] สุธี พงศาสกุลชัย.*คัมภีร์ Visual C# 2005*.กรุงเทพฯ: หจก.ไทยเจริญการพิมพ์,2549

[5] รศ.สมยศ จุณณปิยะ , *" การประยุกต์ใช้งานไมโครคอนโทรลเลอร์ตระกูล MCS-51 "*
คณะวิศวกรรมศาสตร์ , สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง , 2546

[6] MICROELECTRONIKA . MICROCONTROLLER AT89S8253 .
http://www.mikroe.com/eng/chapters/view/67/chapter-4-at89s8253-microcontroller/

[7] อ.ขจร อนุดิตย์ . *การเขียนโปรแกรมควบคุมไมโครคอนโทรลเลอร์ MCS-51 ด้วยภาษาซี*.
นนทบุรี : สำนักพิมพ์ Corefunction ,2550

[8] ZYTRAX.Inc . RS-232 Cables,wiring and Pinouts . http://www.zytrax.com/tech/layer_1/Cables/tech_rs232.htm

ภาคผนวก

**** เมนูอาหาร ****

รหัสอาหาร	รายการอาหาร	ราคา
001	กระเพราหมู	30 บาท
002	ราดหน้า	35 บาท
003	ไข่เจียว	15 บาท
004	คะน้าหมูกรอบ	30 บาท
005	หมูพริกไทยดำ	50 บาท
006	ไข่เยี่ยวม้ากระเพรากรอบ	40 บาท
007	ข้าวผัดทะเล	40 บาท
008	ไก่ทอด	20 บาท
009	ต้มยำปลากระพง	100 บาท
010	เนื้อแดดเดียว	60 บาท
011	น้ำแข็งเปล่า 1 ถัง	15 บาท
012	น้ำอัดลมขวดละ	30 บาท

คู่มือการใช้งานเครื่องสั่งอาหารโดยไร้สาย

1. หากท่านเป็นสมาชิกกรุณากดเลขสมาชิก แล้วกด # กรณีไม่เป็นสมาชิก กรุณากดรหัส 0000 และกด # (ถ้าต้องการสมัครสมาชิกกรุณาติดต่อพนักงาน)
2. เลือกเมนูอาหารที่ท่านต้องการแล้วทำการ กดรหัสของรายการอาหารนั้น 3 หลักแรก ตามด้วยรหัสจำนวนอาหารที่ท่านต้องการ อีก 3 หลัก ตัวอย่างเช่น กรณีเมื่อท่านต้องการสั่ง กระเพราหมู 1 จาน สั่งได้โดยทำการกด รหัส 001 001
3. เมื่อท่านทำการกดรหัสครบ 6 ตัวแล้ว ให้กดปุ่ม # เพื่อทำการส่งข้อมูล
4. หากท่านทำการกดรหัสถูกต้อง หน้าจอ LCD จะปรากฏคำว่า Order received
5. หากท่านทำการกดรหัสไม่ถูกต้อง หน้าจอ LCD จะปรากฏคำว่า Order Error
6. หาก LCD ปรากฏคำว่า Order Error ให้ท่านทำการสั่งอาหารใหม่
7. เมื่อต้องการคิดเงินให้กดรหัส 999 999 เพื่อทำการปริ๊นใบเสร็จ

Source Code

โปรแกรมเครื่อง Server (C#)

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Data.SqlClient;
using System.Data.OleDb;
using System.Data.SqlTypes;
using System.Data.Sql;
using System.Data.Common;
using System.Data.Odbc;
using System.IO.Ports;
using CrystalDecisions.CrystalReports.Engine;


namespace WindowsApplication1
{
    public partial class Form1 : Form
    {

        SerialPort port = new SerialPort("COM5", 9600, Parity.None, 8,
StopBits.One);
        OleDbConnection conObj = new OleDbConnection();
        OleDbDataAdapter da;
        string strCon = "Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=C:/db_food.mdb";
        DataSet ds = new DataSet();
        OleDbCommand cm;
        OleDbDataReader dr;
        int chkTable = 0;
        int chkcount;
        int chkCal = 0;
        int chkDelete = 0;
        public Form1()
        {
            InitializeComponent();
        }

        private void button10_Click(object sender, EventArgs e)
        {
            timer2.Enabled = true;
            if (port.IsOpen == true)
            {
                button10.Text = "เชื่อมต่อ";
```

```
            timer1.Enabled = false;
            port.Close();
        }
        else
        {
            button10.Text = "ปิดระบบ";
            timer1.Enabled = true;
            port.Open();
        }
    }

    private void button1_show(object sender, EventArgs e)
    {
        button1.ForeColor = Color.Red;
        button2.ForeColor = Color.Black;
        button3.ForeColor = Color.Black;
        button4.ForeColor = Color.Black;
        button5.ForeColor = Color.Black;
        button6.ForeColor = Color.Black;
        button7.ForeColor = Color.Black;
        button8.ForeColor = Color.Black;
        button9.ForeColor = Color.Black;

        chkTable = 1;
        chkCal = 0;
        textBox1.Text = "";
        textBox3.Text = "";
        textBox4.Text = "";
        string strSQL = "SELECT * from food where No_Table = '1'";
        conObj.Open();
        cm = new OleDbCommand();
        cm.CommandText = strSQL;
        cm.CommandType = CommandType.Text;
        cm.Connection = conObj;
        dr = cm.ExecuteReader();
        DataTable dt = new DataTable();
        dt.Load(dr);
        dataGridView1.DataSource = dt;
        dr.Close();
        conObj.Close();
        chkDelete = 1;

    }


    private void Form1_Load(object sender, EventArgs e)
    {
        conObj.ConnectionString = strCon;
```

```
        string strSQL = "SELECT * from food where No_Table = '0'";
        conObj.Open();
        cm = new OleDbCommand();
        cm.CommandText = strSQL;
        cm.CommandType = CommandType.Text;
        cm.Connection = conObj;
        dr = cm.ExecuteReader();
        DataTable dt = new DataTable();
        dt.Load(dr);
        dataGridView1.DataSource = dt;
        dr.Close();
        conObj.Close();

        chkDelete = 0;
    }


    private void button2_show(object sender, EventArgs e)
    {
        button1.ForeColor = Color.Black ;
        button2.ForeColor = Color.Red ;
        button3.ForeColor = Color.Black;
        button4.ForeColor = Color.Black;
        button5.ForeColor = Color.Black;
        button6.ForeColor = Color.Black;
        button7.ForeColor = Color.Black;
        button8.ForeColor = Color.Black;
        button9.ForeColor = Color.Black;

        chkTable = 2;
        chkCal = 0;
        textBox1.Text = "";
        textBox3.Text = "";
        textBox4.Text = "";
        string strSQL = "SELECT * from food where No_Table = '2'";
        conObj.Open();
        cm = new OleDbCommand();
        cm.CommandText = strSQL;
        cm.CommandType = CommandType.Text;
        cm.Connection = conObj;
        dr = cm.ExecuteReader();
        DataTable dt = new DataTable();
        dt.Load(dr);
        dataGridView1.DataSource = dt;
        dr.Close();
        conObj.Close();
        chkDelete = 2;
    }
```

```
        private void button14_Click(object sender, EventArgs e)
        {
            chkCal = 1;

            conObj.Open();
            string strSQL = "SELECT SUM(Total) from food where
No_Table = '" + chkDelete + "'";
            da = new OleDbDataAdapter(strSQL, conObj);
            da.Fill(ds, "food");


            string a =
ds.Tables["food"].Rows[0]["Expr1000"].ToString();
            textBox1.Text = a;
            ds.Clear();
            conObj.Close();

            conObj.Open();
            strSQL = "SELECT SUM(Discount) from food where No_Table =
'" + chkDelete + "'";
            da = new OleDbDataAdapter(strSQL, conObj);
            da.Fill(ds, "food");


            a = ds.Tables["food"].Rows[0]["Expr1000"].ToString();
            textBox3.Text = a;
            ds.Clear();
            conObj.Close();

            conObj.Open();
            strSQL = "SELECT SUM(Net) from food where No_Table = '" +
chkDelete + "'";
            da = new OleDbDataAdapter(strSQL, conObj);
            da.Fill(ds, "food");


            a = ds.Tables["food"].Rows[0]["Expr1000"].ToString();
            textBox4.Text = a;
            ds.Clear();
            conObj.Close();
        }


        private void button3_Click_1(object sender, EventArgs e)
        {
            button1.ForeColor = Color.Black ;
            button2.ForeColor = Color.Black;
            button3.ForeColor = Color.Red ;
```

```
        button4.ForeColor = Color.Black;
        button5.ForeColor = Color.Black;
        button6.ForeColor = Color.Black;
        button7.ForeColor = Color.Black;
        button8.ForeColor = Color.Black;
        button9.ForeColor = Color.Black;

        chkTable = 3;
        chkCal = 0;
        textBox1.Text = "";
        textBox3.Text = "";
        textBox4.Text = "";
        string strSQL = "SELECT * from food where No_Table = '3'";
        conObj.Open();
        cm = new OleDbCommand();
        cm.CommandText = strSQL;
        cm.CommandType = CommandType.Text;
        cm.Connection = conObj;
        dr = cm.ExecuteReader();
        DataTable dt = new DataTable();
        dt.Load(dr);
        dataGridView1.DataSource = dt;
        dr.Close();
        conObj.Close();
        chkDelete = 3;
    }

    private void button4_Click(object sender, EventArgs e)
    {
        button1.ForeColor = Color.Black ;
        button2.ForeColor = Color.Black;
        button3.ForeColor = Color.Black;
        button4.ForeColor = Color.Red ;
        button5.ForeColor = Color.Black;
        button6.ForeColor = Color.Black;
        button7.ForeColor = Color.Black;
        button8.ForeColor = Color.Black;
        button9.ForeColor = Color.Black;

        chkTable = 4;
        chkCal = 0;
        textBox1.Text = "";
        textBox3.Text = "";
        textBox4.Text = "";
        string strSQL = "SELECT * from food where No_Table = '4'";
        conObj.Open();
        cm = new OleDbCommand();
        cm.CommandText = strSQL;
        cm.CommandType = CommandType.Text;
```

```
        cm.Connection = conObj;
        dr = cm.ExecuteReader();
        DataTable dt = new DataTable();
        dt.Load(dr);
        dataGridView1.DataSource = dt;
        dr.Close();
        conObj.Close();
        chkDelete = 4;
    }

    private void button5_Click(object sender, EventArgs e)
    {
        button1.ForeColor = Color.Black;
        button2.ForeColor = Color.Black;
        button3.ForeColor = Color.Black;
        button4.ForeColor = Color.Black;
        button5.ForeColor = Color.Red ;
        button6.ForeColor = Color.Black;
        button7.ForeColor = Color.Black;
        button8.ForeColor = Color.Black;
        button9.ForeColor = Color.Black;

        chkTable = 5;
        chkCal = 0;
        textBox1.Text = "";
        textBox3.Text = "";
        textBox4.Text = "";
        string strSQL = "SELECT * from food where No_Table = '5'";
        conObj.Open();
        cm = new OleDbCommand();
        cm.CommandText = strSQL;
        cm.CommandType = CommandType.Text;
        cm.Connection = conObj;
        dr = cm.ExecuteReader();
        DataTable dt = new DataTable();
        dt.Load(dr);
        dataGridView1.DataSource = dt;
        dr.Close();
        conObj.Close();
        chkDelete = 5;
    }

    private void button15_Click(object sender, EventArgs e)
    {
        timer2.Enabled = true;
        textBox1.Text = "";
        textBox3.Text = "";
        textBox4.Text = "";
        if (chkCal == 1)
```

```
            {
                System.Reflection.Assembly exe =
System.Reflection.Assembly.GetEntryAssembly();
                string exePath =
System.IO.Path.GetDirectoryName(exe.Location);
                string strSQL = "SELECT * from food where No_Table =
'" + chkDelete + "'";
                conObj.Open();
                cm = new OleDbCommand();
                cm.CommandText = strSQL;
                cm.CommandType = CommandType.Text;
                cm.Connection = conObj;
                dr = cm.ExecuteReader();
                DataTable dt = new DataTable();
                dt.Load(dr);
                dr.Close();
                conObj.Close();
                ReportDocument rptDoc = new ReportDocument();
                string rptFile = exePath + "\\";
                rptFile = rptFile + "CrystalReport1.rpt";
                rptDoc.Load(rptFile);
                rptDoc.SetDataSource(dt);
                rptDoc.PrintToPrinter(1, true, 1, 1);

                if (conObj.State == ConnectionState.Open)
                {
                    conObj.Close();
                }

                conObj.Open();
                string sql = "delete from food where No_Table = '" +
chkDelete + "'";
                da = new OleDbDataAdapter(sql, conObj);
                ds.Tables.Clear();
                da.Fill(ds, "MyQuery");
                conObj.Close();
            }
            else
            {
                MessageBox.Show("กรุณาคิดเงินก่อน","พิมพ์ใบเสร็จ",
                    MessageBoxButtons.OK ,MessageBoxIcon.Exclamation
);
            }
        }

        private void button6_Click(object sender, EventArgs e)
        {
            button1.ForeColor = Color.Black ;
            button2.ForeColor = Color.Black;
```

```
        button3.ForeColor = Color.Black;
        button4.ForeColor = Color.Black;
        button5.ForeColor = Color.Black;
        button6.ForeColor = Color.Red ;
        button7.ForeColor = Color.Black;
        button8.ForeColor = Color.Black;
        button9.ForeColor = Color.Black;

        chkTable = 6;
        chkCal = 0;
        textBox1.Text = "";
        textBox3.Text = "";
        textBox4.Text = "";
        string strSQL = "SELECT * from food where No_Table = '6'";
        conObj.Open();
        cm = new OleDbCommand();
        cm.CommandText = strSQL;
        cm.CommandType = CommandType.Text;
        cm.Connection = conObj;
        dr = cm.ExecuteReader();
        DataTable dt = new DataTable();
        dt.Load(dr);
        dataGridView1.DataSource = dt;
        dr.Close();
        conObj.Close();
        chkDelete = 6;
}

private void button7_Click(object sender, EventArgs e)
{
        button1.ForeColor = Color.Black ;
        button2.ForeColor = Color.Black;
        button3.ForeColor = Color.Black;
        button4.ForeColor = Color.Black;
        button5.ForeColor = Color.Black;
        button6.ForeColor = Color.Black;
        button7.ForeColor = Color.Red ;
        button8.ForeColor = Color.Black;
        button9.ForeColor = Color.Black;


        chkTable = 7;
        chkCal = 0;
        textBox1.Text = "";
        textBox3.Text = "";
        textBox4.Text = "";
        string strSQL = "SELECT * from food where No_Table = '7'";
        conObj.Open();
        cm = new OleDbCommand();
```

```
        cm.CommandText = strSQL;
        cm.CommandType = CommandType.Text;
        cm.Connection = conObj;
        dr = cm.ExecuteReader();
        DataTable dt = new DataTable();
        dt.Load(dr);
        dataGridView1.DataSource = dt;
        dr.Close();
        conObj.Close();
        chkDelete = 7;
    }

    private void button8_Click(object sender, EventArgs e)
    {
        button1.ForeColor = Color.Black ;
        button2.ForeColor = Color.Black;
        button3.ForeColor = Color.Black;
        button4.ForeColor = Color.Black;
        button5.ForeColor = Color.Black;
        button6.ForeColor = Color.Black;
        button7.ForeColor = Color.Black;
        button8.ForeColor = Color.Red ;
        button9.ForeColor = Color.Black;

        chkTable = 8;
        chkCal = 0;
        textBox1.Text = "";
        textBox3.Text = "";
        textBox4.Text = "";
        string strSQL = "SELECT * from food where No_Table = '8'";
        conObj.Open();
        cm = new OleDbCommand();
        cm.CommandText = strSQL;
        cm.CommandType = CommandType.Text;
        cm.Connection = conObj;
        dr = cm.ExecuteReader();
        DataTable dt = new DataTable();
        dt.Load(dr);
        dataGridView1.DataSource = dt;
        dr.Close();
        conObj.Close();
        chkDelete = 8;
    }

    private void button9_Click(object sender, EventArgs e)
    {
        button1.ForeColor = Color.Black ;
        button2.ForeColor = Color.Black;
        button3.ForeColor = Color.Black;
```

```
        button4.ForeColor = Color.Black;
        button5.ForeColor = Color.Black;
        button6.ForeColor = Color.Black;
        button7.ForeColor = Color.Black;
        button8.ForeColor = Color.Black;
        button9.ForeColor = Color.Red;

        chkTable = 9;
        chkCal = 0;
        textBox1.Text = "";
        textBox3.Text = "";
        textBox4.Text = "";
        string strSQL = "SELECT * from food where No_Table = '9'";
        conObj.Open();
        cm = new OleDbCommand();
        cm.CommandText = strSQL;
        cm.CommandType = CommandType.Text;
        cm.Connection = conObj;
        dr = cm.ExecuteReader();
        DataTable dt = new DataTable();
        dt.Load(dr);
        dataGridView1.DataSource = dt;
        dr.Close();
        conObj.Close();
        chkDelete = 9;
    }

    private void timer1_Tick(object sender, EventArgs e)
    {
        string sBuffer;
        int valPrice;
        int valTotal;
        int valDiscount;
        int valNet;
        int i;
        string rxtime;
        string rxTable;
        string rxList;
        string rxPiece;
        string rxPrice;
        string rxTotal;
        string rxDiscount;
        string rxNet;
        string rxMember;


        if (chkcount == 1)
        {
            timer2.Enabled = true;
```

```
            }

            sBuffer = port.ReadExisting();
            if (sBuffer .Length > 10)
            {
                timer2.Enabled = false;
                chkcount = 1;
            rxTable = sBuffer.Substring(0, 1);
            chkTable = Convert.ToInt32 (rxTable);
            chkTable = chkTable - 1;
            rxList   = sBuffer.Substring(1, 3);
            rxPiece  = sBuffer.Substring(4, 3);
            rxMember  = sBuffer.Substring(7, 4);
            rxtime = textBox5.Text.Substring(0, 8);

            if (rxList == "999")
            {

                chkcount = 0;
                chkCal = 1;
                chkDelete = Convert .ToInt32 ( rxTable);

                conObj.Open();
                string strSQL_ = "SELECT SUM(Total) from food where
No_Table = '" + chkDelete + "'";
                da = new OleDbDataAdapter(strSQL_, conObj);
                da.Fill(ds, "food");


                string a =
ds.Tables["food"].Rows[0]["Expr1000"].ToString();
                textBox1.Text = a;
                ds.Clear();
                conObj.Close();


                conObj.Open();
                strSQL_ = "SELECT SUM(Discount) from food where
No_Table = '" + chkDelete + "'";
                da = new OleDbDataAdapter(strSQL_, conObj);
                da.Fill(ds, "food");


                a = ds.Tables["food"].Rows[0]["Expr1000"].ToString();
                textBox3.Text = a;
                ds.Clear();
                conObj.Close();

                conObj.Open();
```

```
                strSQL_ = "SELECT SUM(Net) from food where No_Table =
'" + chkDelete + "'";
                da = new OleDbDataAdapter(strSQL_, conObj);
                da.Fill(ds, "food");


                a = ds.Tables["food"].Rows[0]["Expr1000"].ToString();
                textBox4.Text = a;
                ds.Clear();
                conObj.Close();

                strSQL_ = "SELECT * from food where No_Table = '" +
rxTable  + "'";
                conObj.Open();
                cm = new OleDbCommand();
                cm.CommandText = strSQL_;
                cm.CommandType = CommandType.Text;
                cm.Connection = conObj;
                dr = cm.ExecuteReader();
                DataTable dt = new DataTable();
                dt.Load(dr);
                dataGridView1.DataSource = dt;
                dr.Close();
                conObj.Close();

                switch (chkDelete)
                {
                    case 1: button1.ForeColor = Color.Red;
                        button2.ForeColor = Color.Black;
                        button3.ForeColor = Color.Black;
                        button4.ForeColor = Color.Black;
                        button5.ForeColor = Color.Black;
                        button6.ForeColor = Color.Black;
                        button7.ForeColor = Color.Black;
                        button8.ForeColor = Color.Black;
                        button9.ForeColor = Color.Black;
                        break;
                    case 2: button1.ForeColor = Color.Black;
                        button2.ForeColor = Color.Red;
                        button3.ForeColor = Color.Black;
                        button4.ForeColor = Color.Black;
                        button5.ForeColor = Color.Black;
                        button6.ForeColor = Color.Black;
                        button7.ForeColor = Color.Black;
                        button8.ForeColor = Color.Black;
                        button9.ForeColor = Color.Black;
                        break;
                    case 3: button1.ForeColor = Color.Black;
                        button2.ForeColor = Color.Black;
```

```
            button3.ForeColor = Color.Red;
            button4.ForeColor = Color.Black;
            button5.ForeColor = Color.Black;
            button6.ForeColor = Color.Black;
            button7.ForeColor = Color.Black;
            button8.ForeColor = Color.Black;
            button9.ForeColor = Color.Black;
            break;
        case 4: button1.ForeColor = Color.Black;
            button2.ForeColor = Color.Black;
            button3.ForeColor = Color.Black;
            button4.ForeColor = Color.Red;
            button5.ForeColor = Color.Black;
            button6.ForeColor = Color.Black;
            button7.ForeColor = Color.Black;
            button8.ForeColor = Color.Black;
            button9.ForeColor = Color.Black;
            break;
        case 5: button1.ForeColor = Color.Black;
            button2.ForeColor = Color.Black;
            button3.ForeColor = Color.Black;
            button4.ForeColor = Color.Black;
            button5.ForeColor = Color.Red;
            button6.ForeColor = Color.Black;
            button7.ForeColor = Color.Black;
            button8.ForeColor = Color.Black;
            button9.ForeColor = Color.Black;
            break;
        case 6: button1.ForeColor = Color.Black;
            button2.ForeColor = Color.Black;
            button3.ForeColor = Color.Black;
            button4.ForeColor = Color.Black;
            button5.ForeColor = Color.Black;
            button6.ForeColor = Color.Red;
            button7.ForeColor = Color.Black;
            button8.ForeColor = Color.Black;
            button9.ForeColor = Color.Black;
            break;
        case 7: button1.ForeColor = Color.Black;
            button2.ForeColor = Color.Black;
            button3.ForeColor = Color.Black;
            button4.ForeColor = Color.Black;
            button5.ForeColor = Color.Black;
            button6.ForeColor = Color.Black;
            button7.ForeColor = Color.Red;
            button8.ForeColor = Color.Black;
            button9.ForeColor = Color.Black;
            break;
        case 8: button1.ForeColor = Color.Black;
```

```
                    button2.ForeColor = Color.Black;
                    button3.ForeColor = Color.Black;
                    button4.ForeColor = Color.Black;
                    button5.ForeColor = Color.Black;
                    button6.ForeColor = Color.Black;
                    button7.ForeColor = Color.Black;
                    button8.ForeColor = Color.Red;
                    button9.ForeColor = Color.Black;
                    break;
                case 9: button1.ForeColor = Color.Black;
                    button2.ForeColor = Color.Black;
                    button3.ForeColor = Color.Black;
                    button4.ForeColor = Color.Black;
                    button5.ForeColor = Color.Black;
                    button6.ForeColor = Color.Black;
                    button7.ForeColor = Color.Black;
                    button8.ForeColor = Color.Black;
                    button9.ForeColor = Color.Red;
                    break;
            }
        }
        conObj.Open();
        string strSQL = "SELECT count(*) from list_food where Id =
'" + rxList + "'";
        da = new OleDbDataAdapter(strSQL, conObj);
        da.Fill(ds, "list_food");


        string chk =
ds.Tables["list_food"].Rows[0]["Expr1000"].ToString();
        int chk2 = Int32.Parse(chk);
        ds.Clear();
        conObj.Close();

        if (chk2 > 0)
        {
            conObj.Open();
            strSQL = "SELECT Name, Price from list_food where Id =
'" + rxList + "'";
            da = new OleDbDataAdapter(strSQL, conObj);
            da.Fill(ds, "list_food");


            string a =
ds.Tables["list_food"].Rows[0]["Name"].ToString();
            string b =
ds.Tables["list_food"].Rows[0]["Price"].ToString();
            rxList = a;
            rxPrice = b;
```

```
                ds.Clear();
                conObj.Close();

                i = Int32.Parse(rxPiece);
                valPrice = Int32.Parse(rxPrice);
                rxPiece  = Convert.ToString(i);
                valTotal = valPrice * i;
                rxTotal = Convert.ToString(valTotal);

                // check สมาชิก //
                conObj.Open();
                strSQL = "SELECT count(*) from member where Id = '" +
rxMember + "'";
                da = new OleDbDataAdapter(strSQL, conObj);
                da.Fill(ds, "member");
                a =
ds.Tables["member"].Rows[0]["Expr1000"].ToString();
                i = Int32.Parse(a);
                ds.Clear();
                conObj.Close();
                if (i > 0)
                {
                    valDiscount = valTotal / 10;
                    rxDiscount = Convert.ToString(valDiscount);
                    valNet = valTotal - valDiscount;
                    rxNet = Convert.ToString(valNet);
                }
                else
                {
                    rxDiscount = "0";
                    rxNet = rxTotal;
                }

                string sqlAdd;
                conObj.Open();
                cm = new OleDbCommand();
                sqlAdd = "INSERT INTO food
(No_Table,List,Piece,Price,Total,Discount,Net,Member,time_)";
                sqlAdd += " VALUES
(@No_Table,@List,@Piece,@Price,@Total,@Discount,@Net,@Member,@time_)";
                cm.CommandText = sqlAdd;
                cm.CommandType = CommandType.Text;
                cm.Connection = conObj;
                cm.Parameters.Clear();
                cm.Parameters.Add("@No_Table",
OleDbType.VarChar).Value = rxTable.Trim();
                cm.Parameters.Add("@List", OleDbType.VarChar).Value =
rxList.Trim();
```

```
                cm.Parameters.Add("@Piece", OleDbType.VarChar).Value =
rxPiece.Trim();
                cm.Parameters.Add("@Price", OleDbType.VarChar).Value =
rxPrice.Trim();
                cm.Parameters.Add("@Total", OleDbType.VarChar).Value =
rxTotal.Trim();
                cm.Parameters.Add("@Discount",
OleDbType.VarChar).Value = rxDiscount.Trim();
                cm.Parameters.Add("@Net", OleDbType.VarChar).Value =
rxNet.Trim();
                cm.Parameters.Add("@Member", OleDbType.VarChar).Value
= rxMember.Trim();
                cm.Parameters.Add("@time_", OleDbType.VarChar).Value =
rxtime.Trim();
                cm.ExecuteNonQuery();
                conObj.Close();
                port.Write("1");
            }
            else
            {
                port.Write("0");
            }
            }

            textBox5.Text =
DateTime.Now.TimeOfDay.ToString().Substring (0,8);


        }

        private void timer2_Tick(object sender, EventArgs e)
        {

            chkTable = chkTable + 1;
            if (chkTable > 9)
                chkTable = 1;

            switch (chkTable){
                case 1: button1.ForeColor = Color.Red ;
                    button2.ForeColor = Color.Black;
                    button3.ForeColor = Color.Black;
                    button4.ForeColor = Color.Black;
                    button5.ForeColor = Color.Black;
                    button6.ForeColor = Color.Black;
                    button7.ForeColor = Color.Black;
                    button8.ForeColor = Color.Black;
                    button9.ForeColor = Color.Black ;
                    break;
                case 2: button1.ForeColor = Color.Black;
```

```
            button2.ForeColor = Color.Red ;
            button3.ForeColor = Color.Black;
            button4.ForeColor = Color.Black;
            button5.ForeColor = Color.Black;
            button6.ForeColor = Color.Black;
            button7.ForeColor = Color.Black;
            button8.ForeColor = Color.Black;
            button9.ForeColor = Color.Black ;
            break;
        case 3: button1.ForeColor = Color.Black;
            button2.ForeColor = Color.Black;
            button3.ForeColor = Color.Red ;
            button4.ForeColor = Color.Black;
            button5.ForeColor = Color.Black;
            button6.ForeColor = Color.Black;
            button7.ForeColor = Color.Black;
            button8.ForeColor = Color.Black;
            button9.ForeColor = Color.Black ;
            break;
        case 4: button1.ForeColor = Color.Black;
            button2.ForeColor = Color.Black;
            button3.ForeColor = Color.Black;
            button4.ForeColor = Color.Red ;
            button5.ForeColor = Color.Black;
            button6.ForeColor = Color.Black;
            button7.ForeColor = Color.Black;
            button8.ForeColor = Color.Black;
            button9.ForeColor = Color.Black;
            break;
        case 5: button1.ForeColor = Color.Black;
            button2.ForeColor = Color.Black;
            button3.ForeColor = Color.Black;
            button4.ForeColor = Color.Black;
            button5.ForeColor = Color.Red ;
            button6.ForeColor = Color.Black;
            button7.ForeColor = Color.Black;
            button8.ForeColor = Color.Black;
            button9.ForeColor = Color.Black;
            break;
        case 6: button1.ForeColor = Color.Black;
            button2.ForeColor = Color.Black;
            button3.ForeColor = Color.Black;
            button4.ForeColor = Color.Black;
            button5.ForeColor = Color.Black;
            button6.ForeColor = Color.Red ;
            button7.ForeColor = Color.Black;
            button8.ForeColor = Color.Black;
            button9.ForeColor = Color.Black;
            break;
```

```
                case 7: button1.ForeColor = Color.Black;
                    button2.ForeColor = Color.Black;
                    button3.ForeColor = Color.Black;
                    button4.ForeColor = Color.Black;
                    button5.ForeColor = Color.Black;
                    button6.ForeColor = Color.Black;
                    button7.ForeColor = Color.Red ;
                    button8.ForeColor = Color.Black;
                    button9.ForeColor = Color.Black;
                    break;
                case 8: button1.ForeColor = Color.Black;
                    button2.ForeColor = Color.Black;
                    button3.ForeColor = Color.Black;
                    button4.ForeColor = Color.Black;
                    button5.ForeColor = Color.Black;
                    button6.ForeColor = Color.Black;
                    button7.ForeColor = Color.Black;
                    button8.ForeColor = Color.Red;
                    button9.ForeColor = Color.Black;
                    break;
                case 9: button1.ForeColor = Color.Black;
                    button2.ForeColor = Color.Black;
                    button3.ForeColor = Color.Black;
                    button4.ForeColor = Color.Black;
                    button5.ForeColor = Color.Black;
                    button6.ForeColor = Color.Black;
                    button7.ForeColor = Color.Black;
                    button8.ForeColor = Color.Black;
                    button9.ForeColor = Color.Red;
                    break;

            }
            string strSQL = "SELECT * from food where No_Table = '" +
chkTable + "'";
            conObj.Open();
            cm = new OleDbCommand();
            cm.CommandText = strSQL;
            cm.CommandType = CommandType.Text;
            cm.Connection = conObj;
            dr = cm.ExecuteReader();
            DataTable dt = new DataTable();
            dt.Load(dr);
            dataGridView1.DataSource = dt;
            dr.Close();
            conObj.Close();
        }
```

```
    }
}
```

ชุดสั่งงานลูกค้า (C)

```
#include <reg51.h>
#include <string.h>
#include <intrins.h>
#include "Lcd.h"

sbit TRW_ce = P1^0;
sbit TRW_cs = P1^2;
sbit TRW_clk1 = P1^3;
sbit TRW_data1 = P1^4;
sbit TRW_dr1 = P1^5;

sbit Row1 = P0^3;
sbit Row2 = P0^2;
sbit Row3 = P0^1;
sbit Row4 = P0^0;
sbit Colum1 = P0^6;
sbit Colum2 = P0^5;
sbit Colum3 = P0^4;

#define  Table '2' //('1'-'9')

#define on                  0
#define off                 1
#define Ok                  0
#define Back                0xAA
#define null                10
#define HIGH                1
#define LOW                 0
#define Key_none            0xFF
#define Key_ok              0x0B
#define Key_cancel          0x0A
#define Recived             0x31
#define Error               0x30
#define     dl                      10

unsigned char MemberNo[5];
unsigned char Key_buff[10];
unsigned char point = 0;
unsigned char Key_press;

code unsigned char reserved_test[] = {0x8E,0x08,0x1C};
code unsigned char data2_w = 0x08;
code unsigned char data1_w = 0x08;
code unsigned char addr2[] = {0x00,0x00,0x00,0x00,0x22};
```

```
code unsigned char addr1[] = {0x00,0x00,0x00,0x00,0x00};
code unsigned char addr_w_crc = 0x0A3;
code unsigned char rf_pgm = 0x4F;
#define ID_TX 0x10
#define ID_RX 0x10
#define Fq_CH 10
unsigned char buff_out,buff_in,F_TI,cnt_rx;

void delay_10usec (void){
_nop_();
_nop_();
_nop_();
_nop_();
_nop_();
_nop_();
_nop_();
_nop_();
_nop_();
_nop_();
_nop_();
_nop_();
_nop_();
_nop_();
_nop_();
_nop_();
}

void delay_200usec(void){
int i;
for(i=0;i<20;i++)
delay_10usec();
}

void delay_msec (int x){
int i,j;
for(i=0;i<=x;i++)
for(j=0;j<1540; j++) _nop_();
}

void WR_TRW (void){
int i;
for (i=0;i<8;i++)
{
TRW_clk1 = 0;
if((buff_out&0x80)==0x80) TRW_data1 =1;
else TRW_data1 = 0;
_nop_();
TRW_clk1 = 1;
_nop_();
```

```
buff_out<<=1;
}
}

void RD_TRW (void){
int j;
buff_in=0;
for (j=0;j<8;j++)
{
TRW_clk1 =0;
_nop_();
buff_in<<=1;
TRW_clk1 =1;
_nop_();
if(TRW_data1==1) buff_in|=0x01;
else buff_in|=0x00;
}
}

void CONFIG_SEND (void){
unsigned char F_TX_CH;
TRW_clk1=0;
TRW_ce=0;
TRW_cs=1;
delay_200usec();
F_TX_CH = Eq_CH;
F_TX_CH<<=1;
F_TX_CH&=0xFE;
buff_out=reserved_test[0];
WR_TRW();
buff_out=reserved_test[1];
WR_TRW();
buff_out=reserved_test[2];
WR_TRW();
buff_out=data2_w;
WR_TRW();
buff_out=data1_w;
WR_TRW();
buff_out=addr2[0];
WR_TRW();
buff_out=addr2[1];
WR_TRW();
buff_out=addr2[2];
WR_TRW();
buff_out=addr2[3];
WR_TRW();
buff_out=addr2[4];
WR_TRW();
buff_out=addr1[0];
```

```
WR_TRW();
buff_out=addr1[1];
WR_TRW();
buff_out=addr1[2];
WR_TRW();
buff_out=addr1[3];
WR_TRW();
buff_out=ID_TX;
WR_TRW();
buff_out=addr_w_crc;
WR_TRW();
buff_out=rf_pgm;
WR_TRW();
buff_out=F_TX_CH;
WR_TRW();
_nop_();
TRW_ce=1;
TRW_cs=0;
TRW_clk1=0;
}

void CONFIG_RECV (void){
unsigned char F_RX_CH;
TRW_clk1=0;
TRW_ce=0;
TRW_cs=1;
delay_200usec();
F_RX_CH = Fq_CH;
F_RX_CH<<=1;
F_RX_CH|=0x01;
buff_out=reserved_test[0];
WR_TRW();
buff_out=reserved_test[1];
WR_TRW();
buff_out=reserved_test[2];
WR_TRW();
buff_out=data2_w;
WR_TRW();
buff_out=data1_w;
WR_TRW();
buff_out=addr2[0];
WR_TRW();
buff_out=addr2[1];
WR_TRW();
buff_out=addr2[2];
WR_TRW();
buff_out=addr2[3];
WR_TRW();
buff_out=addr2[4];
```

```
WR_TRW();
buff_out=addr1[0];
WR_TRW();
buff_out=addr1[1];
WR_TRW();
buff_out=addr1[2];
WR_TRW();
buff_out=addr1[3];
WR_TRW();
buff_out=ID_RX;
WR_TRW();
buff_out=addr_w_crc;
WR_TRW();
buff_out=rf_pgm;
WR_TRW();
buff_out=F_RX_CH;
WR_TRW();
_nop_();
TRW_ce=1;
TRW_cs=0;
TRW_clk1=0;
}

void Send_Data_to_TRW(unsigned char buff_send){
TRW_ce=1;
TRW_cs=0;
_nop_();
buff_out=addr1[0];
WR_TRW();
buff_out=addr1[1];
WR_TRW();
buff_out=addr1[2];
WR_TRW();
buff_out=addr1[3];
WR_TRW();
buff_out=ID_TX;
WR_TRW();
buff_out=buff_send;
WR_TRW();
_nop_();
TRW_ce=0;
TRW_cs=0;
TRW_clk1=0;
}

unsigned char read_Data_from_TRW(void){
unsigned char tmp;
tmp=0xff;
TRW_data1=1;
```

```
TRW_dr1=1;
TRW_ce=1;
TRW_cs=0;
_nop_();
if(TRW_dr1==1)
{
RD_TRW();
tmp=buff_in;
}
TRW_cs=0;
TRW_ce=1;
TRW_cs=0;
TRW_clk1=0;
_nop_();
return(tmp);
}

void delay(int times){
      int i,j;
      for(i=0;i<400;i++)
      for(j=0;j<times;j++);

}

void DSP_main(void){
lcd_puts(0x80,"Enter member NO.");
lcd_puts(0xC0,"                ");
lcd_command(0xC6);
}

void DSP_getorder(void){
char i;
lcd_puts(0x80,"FOOD  NUM  MEMNO");
lcd_puts(0xC0,"                ");
lcd_command(0xCC);
for(i=0;i<4;i++)
lcd_text(MemberNo[i]);
lcd_command(0xC0);
}

unsigned char Scankey(void)
{
      unsigned char Key = 0xFF;
      Colum1 = 0;
      if(Row1==0){
            delay(dl);
            if(Row1==0){
                        Key = '1';
            }
```

```
    }
    if(Row2==0){
                delay(dl);
          if(Row2==0){
                      Key = '4';
          }
    }
    if(Row3==0){
          delay(dl);
                if(Row3==0){
                      Key = '7';
          }
    }
    if(Row4==0){
          delay(dl);
                if(Row4==0){
                      Key = '*';
          }
    }

Colum1 = 1;
    Colum2 = 0;
    if(Row1==0){
          delay(dl);
                if(Row1==0){

                      Key = '2';
          }
    }
    if(Row2==0){
          delay(dl);
                if(Row2==0){
                      Key = '5';
          }
    }
    if(Row3==0){
    delay(dl);
          if(Row3==0){
                      Key = '8';
          }
    }
    if(Row4==0){
          delay(dl);
                if(Row4==0){
                      Key = '0';
          }
    }

    Colum2 = 1;
```

```
	Colum3 = 0;
	if(Row1==0){
		delay(dl);
			if(Row1==0){
				Key = '3';
		}
	}
	if(Row2==0){
		delay(dl);
			if(Row2==0){
				Key = '6';
		}
	}
	if(Row3==0){
	delay(dl);
		if(Row3==0){
				Key = '9';
		}
	}
	if(Row4==0){
		delay(dl);
			if(Row4==0){
				Key = '#';
		}
	}
	Colum3 = 1;

	return(Key);

}

void Recive_trw(void){
	unsigned char Buff_recive;
	while(1){
	if(TRW_dr1==1){
	 Buff_recive = read_Data_from_TRW();
	 delay(20);
	 if(Buff_recive == Recived){
		lcd_puts(0x80,"     Order      ");
		lcd_puts(0xC0,"    Recived     ");
		}
	 else if(Buff_recive == Error){
		lcd_puts(0x80,"     Order      ");
		lcd_puts(0xC0,"     Error      ");
			}
		 delay(500);
		 DSP_getorder();
		 break;
		}
```

```
        }
}

void Send_trw(void){
char i;
                CONFIG_SEND();
                delay_10usec();
                Send_Data_to_TRW(Table);
                delay_200usec();
                delay_200usec();
                delay_200usec();
                for(i=0;i<6;i++){
                Send_Data_to_TRW(Key_buff[i]);
                delay_200usec();
                delay_200usec();
                delay_200usec();
                }
                for(i=0;i<4;i++){
                Send_Data_to_TRW(MemberNo[i]);
                delay_200usec();
                delay_200usec();
                delay_200usec();
                }
                CONFIG_RECV();

}

unsigned char Get_memno(void){
        char i;
        Key_press = Scankey();
        if(Key_press != 0xFF && Key_press != '*' && Key_press != '#'){

                        delay(20);
                        lcd_text(Key_press);
                        Key_buff[point] = Key_press;
                        point++;
                }
                if(Key_press == '*'){
                        delay(20);
                        if(point == 1){
                                lcd_command(0xC6);
                                lcd_text(' ');
                                lcd_command(0xC6);
                        }
                        if(point == 2){
                                lcd_command(0xC7);
                                lcd_text(' ');
                                lcd_command(0xC7);
                        }
```

```
                if(point == 3){
                      lcd_command(0xC8);
                      lcd_text(' ');
                      lcd_command(0xC8);
                }
                if(point == 4){
                      lcd_command(0xC9);
                      lcd_text(' ');
                      lcd_command(0xC9);
                }
                if(point == 0){
                      return(Back);
                }
                 point --;


          }

          if(point == 4){
                lcd_puts(0xC0,"Del");
                lcd_puts(0xCE,"Ok");
                }
           if(point >= 4 && Key_press == '#' ){
                for(i=0;i<4;i++)
                MemberNo[i]=Key_buff[i];
                return(Ok);
                }
           else
                return(~Ok);

}


unsigned char Get_order(void){

      Key_press = Scankey();
      if(Key_press != Key_none & Key_press != '*' & Key_press != '#'){

                delay(20);
                lcd_text(Key_press);
                Key_buff[point] = Key_press;
                point ++;
          }
          if(Key_press == '*'){
                delay(20);
                if(point == 1){
                      lcd_command(0xC0);
                      lcd_text(' ');
                      lcd_command(0xC0);
```

```
            }
            if(point == 2){
                  lcd_command(0xC1);
                  lcd_text(' ');
                  lcd_command(0xC1);
            }
            if(point == 3){
                  lcd_command(0xC2);
                  lcd_text(' ');
                  lcd_command(0xC2);
            }
            if(point == 4){
                  lcd_command(0xC6);
                  lcd_text(' ');
                  lcd_command(0xC6);
            }
            if(point == 5){
                  lcd_command(0xC7);
                  lcd_text(' ');
                  lcd_command(0xC7);
            }
            if(point == 6){
                  lcd_command(0xC8);
                  lcd_text(' ');
                  lcd_command(0xC8);
            }

            if(point == 0){
                  return(Back);
            }
             point --;

      }

      if(point == 3){
            lcd_command(0xC6);
            }

      if(point == 6){
            lcd_command(0xCF);
            }
      if(point >= 6 && Key_press == '#' )
            return(0k);
      else
            return(~0k);

}
unsigned char Send_order(void){
```

```
unsigned char Ret;
DSP_getorder();
      point=0;
      memset(Key_buff,'0',10);
      while(1){
      if((Ret=Get_order())==Ok){
        Send_trw();
        Recive_trw();
        point=0;
        lcd_command(0xC0);
        }
        else if(Ret==Back){
            point=0;
            memset(Key_buff,'0',10);
            DSP_main();
            return 0;
            }
       }
}
void main (void){
P3=0xFF;
TRW_ce=0;
TRW_cs=0;
TRW_clk1=0;
TRW_data1=1;
TRW_dr1=1;
delay_msec(15);
lcd_init();
delay(50);
point=0;
memset(Key_buff,'0',10);
DSP_main();
while(1){
        if(Get_memno()==Ok){
            Send_order();
            }
      }
}
```