ร้านอาหารอัตโนมัติและเครื่องสั่งอาหารไร้สาย

AUTOMATIC RESTAURANT WITH WIRELESS MENU

โดย

นางสาวนภา แซ่เบ๊

นายพชร บุญญลักษม์

นางสาวสุภาณี โอศิริวัฒนกุล

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าคุณทหารลาดกระบัง

ปีการศึกษา 2547

ร้านอาหารอัตโนมัติและเครื่องสั่งอาหารไร้สาย

AUTOMATIC RESTAURANT WITH WIRELESS MENU

โดย

นางสาวนภา	แซ่เบ๊	44010240
นายพชร	บุญญลักษม์	44010315
นางสาวสุภาณี	โอศิริวัฒนกุล	44010545

อาจารย์ที่ปรึกษา

ผศ. เกรียงไกร วงศ์โรจนภรณ์

รศ.ดร. สุวิพล สิทธิชีวภาค

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าคุณทหารลาดกระบัง

ปีการศึกษา 2547

ร้านอาหารอัตโนมัติและเครื่องสั่งอาหารไร้สาย

AUTOMATIC RESTAURANT WITH WIRELESS MENU

โดย	นางสาวนภา	แซ่เบ๊	44010240
	นายพชร	บุญญลักษม์	44010315
	นางสาวสุภาณี	โอศิริวัฒนกุล	44010545

อาจารย์ที่ปรึกษา	ผศ. เกรียงไกร	วงศ์โรจนภรณ์
	รศ.ดร. สุวิพล	สิทธิชีวภาค

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้ เป็นการออกแบบและสร้างระบบร้านอาหารอัตโนมัติ เพื่อเพิ่มความสะดวกรวดเร็วในการให้บริการลูกค้า และประหยัดค่าใช้จ่ายในการจ้างพนักงานของร้าน โดยใช้วิธีการให้ลูกค้าป้อนรหัสรายการอาหารเข้าสู่เครื่องสั่งอาหาร เครื่องจะทำการส่งข้อมูลรายการอาหารนี้ไปยังเครื่องรับแบบไร้สาย จากนั้นเครื่องคอมพิวเตอร์จะทำการแสดงผลออกทางจอภาพ พร้อมทั้งคิดเงิน และมีการแสดงรายการอาหารที่สั่งในครัว นอกจากนี้ยังมีระบบรอเรียกตามคิวอีกด้วย

Abstract

This project designs and invents automatic restaurant system. In system uses programmable integrated circuit (PIC) to work as wireless menu that customer can order food by entering code of food from the menu. Data will be sent by transmitter to computer then display bill on monitor and send the order into kitchen. In addition, there is queuing system for use in case of having many customer.

ปริญญานิพนธ์ปีการศึกษา 2547

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ร้านอาหารอัตโนมัติและเครื่องสั่งอาหารไร้สาย

AUTOMATIC RESTAURANT WITH WIRELESS MENU

ผู้จัดทำ

1. นางสาวนภา แซ่เบ๊ 44010240
2. นายพชร บุญญลักษม์ 44010315
3. นางสาวสุภาณี โอศิริวัฒนกุล 44010545

..... อาจารย์ที่ปรึกษา

(ผศ.เกรียงไกร วงศ์โรจนภรณ์)

..... อาจารย์ที่ปรึกษา

(รศ.ดร.สุวิพล สิทธิชีวภาค)

สารบัญ

สารบัญรูปภาพ

สารบัญตาราง

บทที่ 1 บทนำ

ในปัจจุบันธุรกิจด้านร้านอาหารมีการขยายตัวเป็นอย่างมาก เนื่องจากผู้คนออกไปทานอาหารนอกบ้านกันมากขึ้น ทำให้มีการแข่งขันในธุรกิจประเภทนี้สูง ไม่ว่าจะเป็นด้านภาพลักษณ์ของร้านอาหาร คุณภาพของอาหาร หรือรายการส่งเสริมการขายต่างๆ โดยเฉพาะอย่างยิ่งด้านการให้บริการซึ่งสามารถสร้างความประทับใจให้กับลูกค้าที่มาใช้บริการได้เป็นอย่างมาก

ปริญญานิพนธ์ฉบับนี้ได้ทำการศึกษาออกแบบ ระบบร้านอาหารอัตโนมัติและเครื่องสั่งอาหารไร้สาย โดยมีจุดประสงค์เพื่อการให้บริการที่มีความสะดวก รวดเร็วและทันสมัย ซึ่งเป็นปัจจัยสำคัญในการสร้างความประทับใจและดึงดูดลูกค้า อีกทั้งยังสามารถประหยัดค่าใช้จ่ายในการจ้างพนักงานอีกด้วย

โครงสร้างการทำงานของระบบจะแบ่งออกเป็นส่วนต่างๆตามหน้าที่ดังนี้ คือ 1.ส่วนควบคุม (Controlling) 2.ส่วนสั่งอาหาร (Ordering) 3.ส่วนครัว (Cooking) 4.ส่วนจัดคิว (Queuing) 5.ส่วนคิดเงินและออกบิล (Billing) ซึ่งมีลักษณะการทำงานคือทุกๆโต๊ะจะมีเครื่องสั่งอาหารระบบไร้สายวางไว้ เมื่อลูกค้าเข้ามาก็จะเลือกรายการอาหารจากเมนูที่วางไว้ และทำการสั่งอาหารโดยกดรหัสรายการอาหารที่เครื่องสั่งอาหาร ซึ่งเมื่อกดแล้วจอแสดงผลจะแสดงรายการอาหารและจำนวนที่สั่ง เมื่อลูกค้ากดยืนยันเครื่องก็จะทำการส่งข้อมูลไปยังส่วนควบคุมแบบไร้สาย ส่วนควบคุมก็จะทำการตรวจสอบความถูกต้องของข้อมูล แล้วส่งข้อมูลไปยังเครื่องคอมพิวเตอร์ ที่เครื่องคอมพิวเตอร์ก็จะเปิดหน้าต่างแสดงผลซึ่งจะประกอบไปด้วยรายการอาหารและจำนวนที่แต่ละโต๊ะสั่ง รวมทั้งทำการคิดเงินเมื่อลูกค้าทานอาหารเสร็จ สำหรับในส่วนครัวจะมีเครื่องรับเพื่อตรวจสอบว่ามีการสั่งอาหารอะไรที่โต๊ะใดบ้าง แล้วทำการจัดรายการอาหารที่ต้องทำตามลำดับ พร้อมทั้งแสดงผลทางจอแอลซีดีที่ติดตั้งไว้ในครัว

ทางด้านการติดต่อสื่อสารกันภายในระบบนี้นั้น จะใช้วิธีการมอดูเลตข้อมูลดิจิตอลแบบแอมปลิจูดชิฟคีย์อิ่ง (Amplitude Shift Keying) กับคลื่นพาห์ที่ความถี่ 433.92 MHz ทั้งระบบ ดังนั้นจึงต้องมีกระบวนการในการป้องกันการชนกันของข้อมูลในกรณีที่มีการส่งข้อมูลพร้อมกัน ในระบบนี้จะเลือกใช้วิธีการเข้าถึงแบบหยั่งสัญญาณ (Polling) โดยจัดให้ส่วนควบคุมทำการหยั่งสัญญาณไปยังแต่ละโต๊ะว่ามีข้อมูลที่ต้องการจะส่งหรือไม่ แล้วจึงอนุญาตให้ส่งข้อมูล ทำให้สามารถป้องกันการชนกันของข้อมูลได้อย่างสมบูรณ์

ในด้านฮาร์ดแวร์นั้นจะมีวงจรสำคัญ ๆ ดังนี้คือเครื่องส่งจะใช้ไอซี rfPIC12f675 ซึ่งเป็นไอซีที่มีคุณภาพสูงและมีราคาถูก มีทั้งไมโครคอนโทรลเลอร์และเครื่องส่งอยู่ในตัว เครื่องรับจะใช้ไอซี rfRXD0420 เป็นไอซีเครื่องรับที่ทำงานคู่กับเครื่องส่งของไอซี rfPIC12f675 วงจรบันทึก/เล่นสัญญาณเสียงจะใช้ไอซี ISD2560 ซึ่งสามารถบันทึกเสียงได้ 60 วินาที ควบคุมการเล่นเสียงโดยไมโครคอนโทรลเลอร์

ส่วนด้านซอฟแวร์นั้นจะได้มีการภาษาแอสแซมบลี(Assambly) ในการสั่งงานไมโครคอนโทรลเลอร์ ตระกูล MCS-51 และตระกูล PIC และภาษา C++ ในส่วนของการคิดเงินและออกบิล เพื่อแสดงหน้าต่างรายละเอียดการสั่งอาหาร รวมทั้งการคิดเงินและออกบิล

บทที่ 2
ทฤษฎีและหลักการ

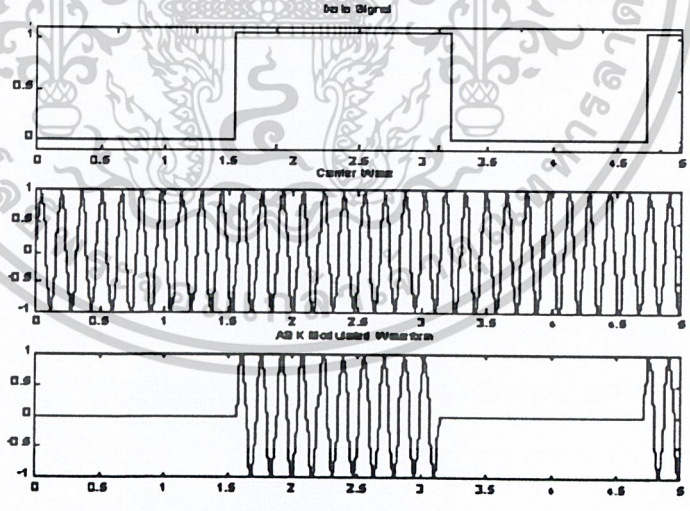
2.1 การมอดูเลตแบบแอมปลิจูดชิฟคีย์อิง (ASK)

ในการส่งสัญญาณข้อมูลดิจิตอลนั้น บางครั้งคุณสมบัติของช่องสัญญาณมีจำกัด ทำให้เราไม่สามารถส่งสัญญาณดิจิตอลผ่านช่องสัญญาณดังกล่าวไปโดยตรง เช่น เมื่อช่องสัญญาณเป็นบรรยากาศ เราจำเป็นต้องทำการมอดูเลตฝากข้อมูลสัญญาณดิจิตอลนั้นไปกับคลื่นพาห์ที่เหมาะสม ซึ่งมีเทคนิคที่ใช้กันอยู่ทั่วไป 3 แบบคือ

- Amplitude Shift Keying (ASK)
- Frequency Shift Keying (FSK)
- Phase Shift Keying (PSK)

สำหรับเทคนิคแบบ ASK นั้น ค่าไบนารีสองค่าของสัญญาณดิจิตอล จะแทนด้วยค่าแอมปลิจูดที่แตกต่างกันสองค่าของความถี่คลื่นพาห์ เช่น ไบนารี 1 แทนด้วยแอมปลิจูดคงที่ของคลื่นพาห์ ส่วนไบนารี 0 แทนด้วยคลื่นพาห์ที่ขาดหายไป ผลของสัญญาณเป็นดังนี้คือ

$$S(t) = \begin{cases} A\cos(2\pi f_c + \theta_c) & ; \text{ไบนารี 1} \\ 0 & ; \text{ไบนารี 0} \end{cases}$$



รูปที่ 2.1 การมอดูเลตแบบแอมปลิจูดชิฟคีย์อิง

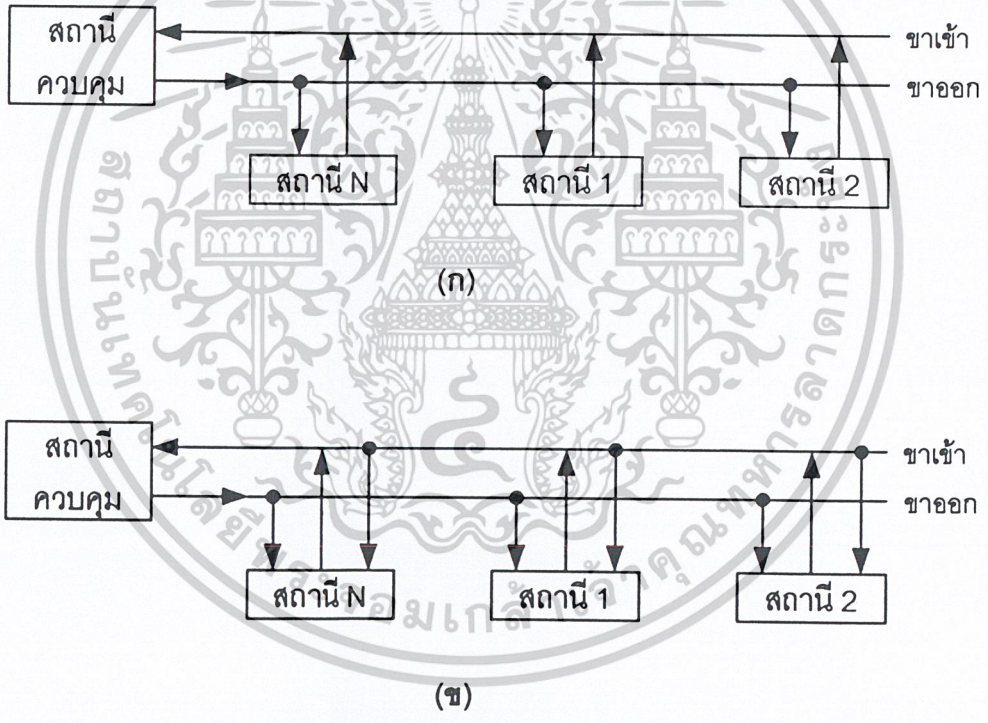
โดยที่สัญญาณคลื่นพาห์ก็คือ $A\cos(2\pi f_c + \theta_c)$ สำหรับบนสาย voice grade จะสามารถส่งข้อมูลได้สูงสุดเพียง 1200 bps

2.2 การเข้าถึงหลายทางแบบหยั่งสัญญาณ (Polling)

การเข้าถึงช่องสัญญาณแบบนี้ จะใช้กับเครือข่ายที่มีโทโปโลยีแบบดาวและบัส โดยในกรณีโทโปโลยีแบบบัสจะมีการใช้สายส่งข้อมูล 2 เส้นต่อออกจากสถานีควบคุม (Control hub) และสถานีลูกข่ายต่อ แทปออกจากสาย 2 เส้นนี้ สำหรับรูปแบบวิธีการหยั่งสัญญาณแบ่งออกเป็น 3 รูปแบบคือ

1.การหยั่งสัญญาณแบบทยอยเรียก (Roll and Poll)

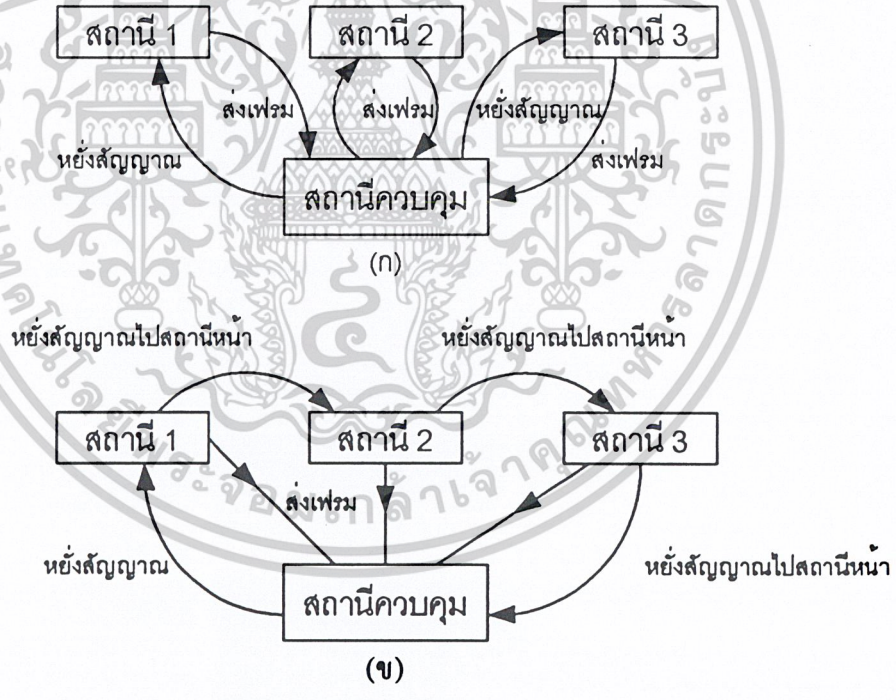
การหยั่งสัญญาณแบบนี้ สถานีควบคุมจะทำการเรียกสถานีลูกข่ายที่ต่ออยู่บนตัวกลางไปทีละสถานีจนครบและวนกลับมาหยั่งสถานีแรกใหม่ เมื่อสถานีลูกข่ายได้รับการหยั่ง สัญญาณจึงมีสิทธิส่งเฟรมข้อมูล และถ้าหากไม่มีข้อมูลที่จะส่งสถานีลูกข่ายต้องตอบกลับ สถานีควบคุมเพื่อแสดงว่าได้รับการหยั่งสัญญาณ โดยเวลาในการหยั่งสัญญาณจะเพิ่มขึ้นตามจำนวนของสถานีลูกข่าย



รูปที่ 2.2 แสดงการหยั่งสัญญาณแบบทยอยเรียก

2.การหยั่งสัญญาณแบบฮับ (Hub Polling)

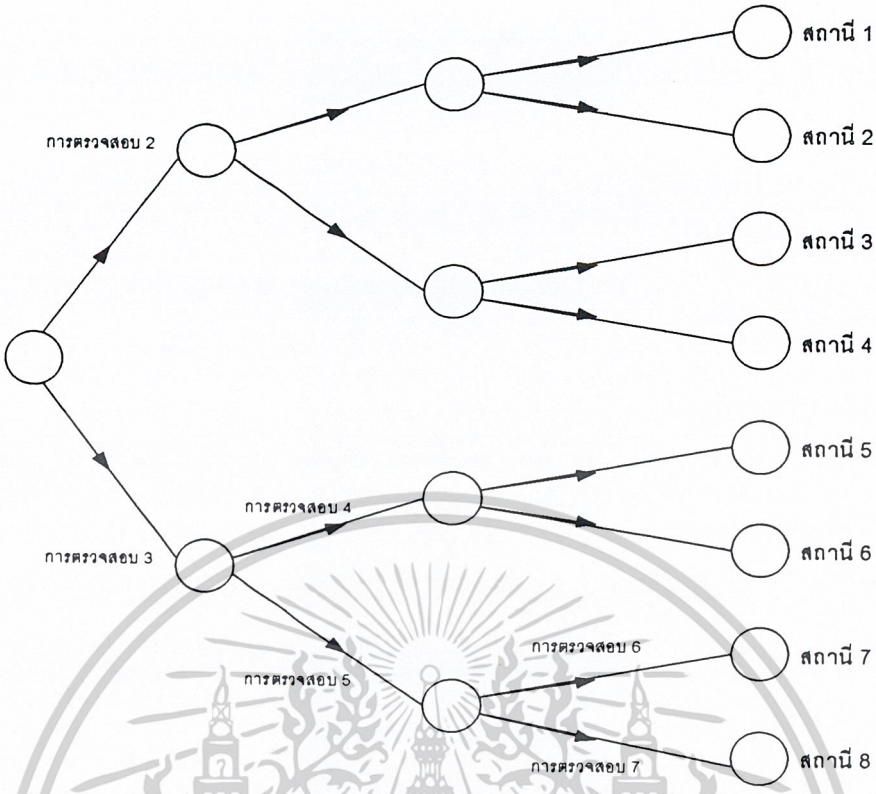
การหยั่งสัญญาณแบบฮับเป็นการปรับปรุงจากแบบทยอยเรียก โดยลดเวลาหยั่งสัญญาณในแต่ละรอบลง เมื่อสถานีควบคุมหยั่งสัญญาณไปที่สถานีที่ไกลที่สุดหรือสถานี 1 ทันทีที่สถานีที่ 1 ส่งเฟรมให้สถานีควบคุม สถานี 1 จะส่งเฟรมหยั่งสัญญาณไปให้สถานีที่อยู่ข้างเคียงคือสถานีที่ 2 ดังนั้นถ้าหากสถานีที่ 2 มีข้อมูลสามารถส่งเฟรมข้อมูลให้กับสถานีควบคุมทันที โดยไม่ต้องรอสถานีควบคุมส่งเฟรมหยั่งสัญญาณมาให้ และทันทีที่สถานี 2 ส่งเฟรมข้อมูลออกไป สถานี 2 จะส่งเฟรมหยั่งสัญญาณให้กับสถานี 3 เช่นกันตามลำดับจนถึงสถานี N ตอบกลับสถานีควบคุม ซึ่งเป็นการหมดการหยั่งสัญญาณ 1 รอบ สถานีควบคุมจึงส่งเฟรมหยั่งสัญญาณให้กับสถานี 1 ใหม่ จากรูปที่ 2.3 ข สถานีลูกข่ายคือสถานี 1, 2, N ต่อแบบสายหลายจุดต่อออก (multidrop) จากสถานีควบคุมมีการเพิ่มสายสัญญาณอีก 1 เส้น จากสายสัญญาณขาเข้า (Inbound) เพื่อรับเฟรมหยั่งสัญญาณจากสถานีข้างเคียง



รูปที่ 2.3 แสดงการหยั่งสัญญาณแบบฮับ

3.การหยั่งสัญญาณแบบปรับตัว (Adaptive Polling)

ในกรณีมีสถานีลูกข่ายมากแต่จำนวนของสถานีลูกข่ายที่ต้องการส่งเฟรมมีน้อย การหยั่งสัญญาณไปสถานีที่ไม่ต้องการส่งข้อมูลเป็นการเสียเวลา การหยั่งสัญญาณแบบปรับตัวจะทำการลดเวลาการหยั่งสัญญาณแต่ละรอบลง โดยพยายามหยั่งสัญญาณไปที่สถานีที่ต้องการส่งข้อมูลเท่านั้นและแยกสถานีที่ไม่มีข้อมูลที่จะส่งออก สำหรับวิธีการอธิบายได้จากรูปที่ 2.4 สถานีควบคุมมีสถานีลูกข่ายอยู่ 8 สถานี โดยมีสถานีที่ 7 เป็นสถานีต้องการส่งข้อมูลให้สถานีควบคุม เริ่มแรกสถานีควบคุมส่งสัญญาณตรวจสอบ (probe) เพื่อตรวจสอบว่าสถานีใดต้องการส่งข้อมูล โดยสัญญาณตรวจสอบจะมีรหัสที่ระบุกลุ่มสถานีที่ต้องการตรวจสอบ ซึ่งเมื่อมองลักษณะเครือข่ายทางลอจิกจะเป็นการเชื่อมต่อแบบกิ่งไม้ ในการหยั่งสัญญาณเป็นกลุ่มนี้ทุกสถานีที่มีข้อมูลจะตอบกลับด้วยเฟรม ACK การตอบกลับเหลื่อมซ้อนกัน ถือว่าเป็นการตอบรับว่ามีข้อมูล จากรูปที่ 2.4 เริ่มแรกสถานีควบคุมจะส่งสัญญาณตรวจสอบเรียกทุกสถานี โดยถ้ามีสถานีใดสถานีหนึ่งต้องการส่งข้อมูล สถานีควบคุมจะได้รับเฟรมตอบกลับ ในการตรวจสอบครั้งที่ 2 สถานีควบคุมจะแยกตรวจสอบเฉพาะสถานี [1, 2, 3, 4] ในกรณีนี้สถานี [1, 2, 3, 4] ไม่มีข้อมูลจึงไม่มีการตอบกลับ ดังนั้นสถานีควบคุมจะไม่ทำการหยั่งสถานี [1, 2, 3, 4] อีก การตรวจสอบครั้งที่ 3 สถานีควบคุมทำการตรวจสอบเฉพาะกลุ่ม [5, 6, 7, 8] ในกรณีนี้จะมีการตอบกลับ ดังนั้นเพื่อต้องการทราบว่าสถานีใดต้องการส่งข้อมูล สถานีควบคุมจึงแยกตรวจสอบทีละสองสถานีคือ สถานี [5, 6] ในครั้งที่ 4 ซึ่งสถานีควบคุมไม่ได้รับการตอบกลับ ดังนั้นสถานีควบคุมจึงตรวจสอบสถานี [7, 8] ในครั้งที่ 5 และพบว่ามีการตอบกลับ จึงแยกตรวจสอบสถานี 7 และ 8 ตามลำดับ การตรวจสอบในครั้งที่ 6 เมื่อสถานีควบคุมตรวจสอบสถานี 7 และสถานี 7 ต้องการส่งข้อมูล และในการตรวจสอบครั้งที่ 7 สถานี 8 ไม่มีการตอบกลับ โดยเมื่อหมดจากการตรวจสอบครั้งที่ 7 แล้วสถานีควบคุมจะเริ่มทำการตรวจสอบสถานีทั้งหมดด้วยวิธีการเช่นเดิมอีก โดยการหยั่งสัญญาณแบบปรับตัวนี้ถ้ามีสถานีลูกข่ายจำนวน 2^n สถานี จำนวนครั้งการหยั่งสัญญาณของการที่มีสถานีต้องการส่งเฟรมเพียงสถานีเดียวจะเท่ากับ $2n+1$ ครั้ง ซึ่งในกรณีที่มีจำนวนสถานีมากๆ เช่น 256 สถานี (n=8) และมีสถานีต้องการส่งข้อมูลเพียงสถานีเดียว จำนวนครั้งของการหยั่งสัญญาณมีเพียง 17 ครั้งในแต่ละรอบ แต่ถ้าหากทุกสถานีต้องการส่งข้อมูลแล้วจำนวนครั้งของการหยั่งสัญญาณเท่ากับ $2^{n+1}-1$ ครั้ง ซึ่งจะมากกว่าวิธีหยั่งสัญญาณแบบทยอยเรียก ดังนั้นการหยั่งสัญญาณแบบปรับตัว จึงเหมาะกับระบบที่มีลูกข่ายจำนวนมากแต่มีสถานีที่ต้องการส่งข้อมูลน้อยเท่านั้น



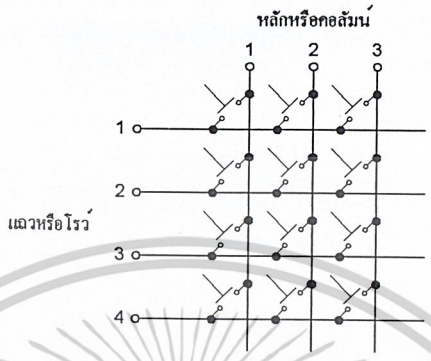
รูปที่ 2.4 การหยั่งสัญญาณแบบปรับตัว

2.3 การใช้งานคีย์แพด (Keypad)

วิธีการอ่านค่าหรือรับค่าจากสวิตช์นั้นมีด้วยกัน 2 ลักษณะใหญ่ๆคือ แบบต่อเข้ากับไฟเลี้ยงหรือกราวด์โดยตรง และต่อแบบวงจรเมทริกซ์ (Matrix Switch) ในรูปที่ 2.5 จะเห็นว่าสวิตช์จะถูกต่อในแนวแกนตั้งและแนวแกนนอน จะเรียกแนวแกนตั้งว่าหลักหรือคอลัมน์ (Column) และเรียกแนวนอนว่าแถว หรือโรว์ (Row) ดังนั้นค่าของสวิตช์ในแต่ละตำแหน่งจะต้องประกอบด้วยหลักและแถว

กระบวนการในการอ่านค่าของสวิตช์แบบนี้นั้นจะใช้วิธีการเขียนโปรแกรมในไมโครคอนโทรลเลอร์ โดยจะต้องใช้สายทั้งหมด 7 เส้น ซึ่งเป็นสายของหลัก 3 เส้นและเป็นสายของแถว 4 เส้น สำหรับสวิตช์แบบ 4 x 3 ต่อเข้ากับไมโครคอนโทรลเลอร์ และที่ขาของพอร์ทที่ต่อกับด้านแถวจะต้องต่อตัวต้านทานพลูอัพไว้เพื่อกำหนดสภาวะเริ่มต้นที่ไม่มีการกดคีย์ ไมโครคอนโทรลเลอร์จะทำการส่งข้อมูล "0" ไปยังพอร์ทที่ต่อกับด้านหลักทีละเส้นตามลำดับ ในทุกครั้งที่มีการส่งข้อมูลไปยังสายด้านหลักของคีย์แพด ไมโครคอนโทรลเลอร์จะทำการอ่านค่าที่ด้านแถวเข้ามาด้วย หากไม่มีการกดค่าของด้านแถวก็จะเป็น "1" ทั้งหมด ถ้าหากมีการกดคีย์ค่าของด้านแถวก็จะไม่เป็น "111" อีกต่อไป เป็นการแจ้งให้ทราบว่ามีการกดคีย์แพดเกิดขึ้นแล้ว จากนั้นไมโครคอนโทรลเลอร์ก็จะทำการค้นหาตำแหน่งต่อไป โดยการค้นหาตำแหน่งนั้นสิ่งที่จะได้มาอย่างแรกคือค่าตำแหน่งของคีย์นั้น จากนั้นก็จะนำค่าตำแหน่งนั้นไปเปิดตารางข้อมูลเพื่อที่จะได้ค่าที่ต้องการนำไปแสดงผลที่แท้จริง

ข้อดีของสวิตซ์แบบนี้ก็คือสามารถรองรับการเพิ่มของสวิตซ์ได้อย่างสะดวก เพียงแค่แก้ไขซอร์ฟแวร์เพียงเล็กน้อยเท่านั้น ทำให้วงจรสวิตซ์แบบเมทริกซ์นี้เป็นที่นิยมกันมากในระบบควบคุมอัตโนมัติหรือกึ่งอัตโนมัติ โดยในการใช้งานทั่วไปจะเรียกสวิตซ์แบบเมทริกซ์นี้ว่า "คีย์แพด"



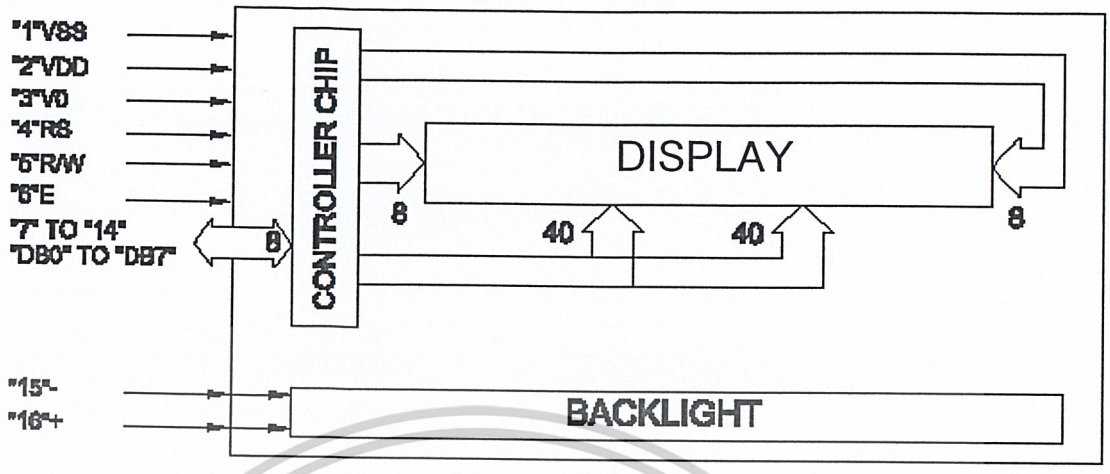
รูปที่ 2.5 วงจรสวิตซ์แบบเมทริกซ์ หรือ คีย์แพด

2.4 การใช้งานแอลซีดี

ในแอลซีดีโมดูล จะมีส่วนประกอบหลักๆ 3 ส่วนดังนี้

1. **ตัวแสดงผล (Display)** ภายในเป็นผลึกเหลวที่สามารถแสดงผลให้เห็นโดยอาศัยแสงจากภายนอก ดังนั้นจึงต้องมีมุมมองข้อมูลที่แสดงผลบนจอ
2. **ตัวควบคุม (Controller)** เป็นตัวรับข้อมูลจากอุปกรณ์ภายนอกมาควบคุมการทำงานของแอลซีดี เช่น ลบจอภาพ แสดงตัวอักษร หรือเลื่อนเคอร์เซอร์ เป็นต้น ตัวควบคุมนี้ใช้ชิปควบคุมโดยเฉพาะ ชิปที่นิยมใช้คือ เบอร์ HD44780 และ HD61830 โดย HD44780 จะใช้ควบคุมแอลซีดีแบบอักขระ ส่วน HD61830 จะใช้ควบคุมแอลซีดีแบบกราฟิก
3. **ตัวขับ (Driver)** เป็นตัวรับสัญญาณจากตัวควบคุมมาขับให้ตัวแสดงผลแสดงข้อมูลตามที่กำหนด ชิปที่ใช้ทำหน้าที่นี้ได้แก่ เบอร์ HD44100H และ MSM5259 เป็นต้น

แอลซีดีโมดูลมีอยู่หลายรุ่น และคุณสมบัติแตกต่างกันไป ซึ่งแบ่งได้เป็น 2 แบบคือ แบบคอทเมทริกซ์ และกราฟิก โดยแบบ คอทเมทริกซ์จะแสดงผลเป็นแบบ 5 x 8 คอท หรือ 5 x 10 คอท มีตั้งแต่ 1 ไลน์, 2 ไลน์ และ 4 ไลน์ ซึ่งการใช้งานแต่ละแบบจะใกล้เคียงกัน ลักษณะขาสัญญาณของแอลซีดีโมดูลแบบ 1 ไลน์ ดังรูปที่ 2.6



รูปที่ 2.6 โครงสร้างของโมดูลแอลซีดี

<u>รายละเอียดการใช้งานขาต่างๆ</u>

Vss (ขา 1) - ต่อกราวด์

VDD (ขา 2) - ต่อไฟเลี้ยง +5V

Vo (ขา 3) - เป็นขาอินพุทรับแรงดันเพื่อปรับความเข้มของการแสดงผล

RS (ขา 4) - เป็นขาอินพุทใช้ในการแยกชนิดของข้อมูลที่ทำการประมวลผลในขณะนั้นว่าเป็นคำสั่งสำหรับรีจิสเตอร์ IR หรือเป็นข้อมูลสำหรับรีจิสเตอร์ DR โดยขานี้เป็น "0" ข้อมูลที่ส่งมาจะเป็นคำสั่งแต่ถ้าขานี้เป็น "1"ข้อมูลที่ส่งมาจะเป็นข้อมูลสำหรับการแสดงผล

R/W (ขา 5) - เป็นขาที่ใช้เลือกการอ่านหรือเขียนข้อมูลจากโมดูลแอลซีดีถ้าเป็น "0" เป็นการกำหนดให้เขียนข้อมูล แต่ถ้าเป็น "1" จะเป็นการอ่านข้อมูล

E (ขา 6) - เป็นขาสำหรับรับสัญญาณพัลส์เอ็นเอเบิลโมดูลแอลซีดีให้ทำงาน

D0–D7 (ขา 7-14) - เป็นขาที่ใช้เป็นทางผ่านของข้อมูลระหว่างแอลซีดี กับอุปกรณ์ภายนอกขนาด 8 บิต

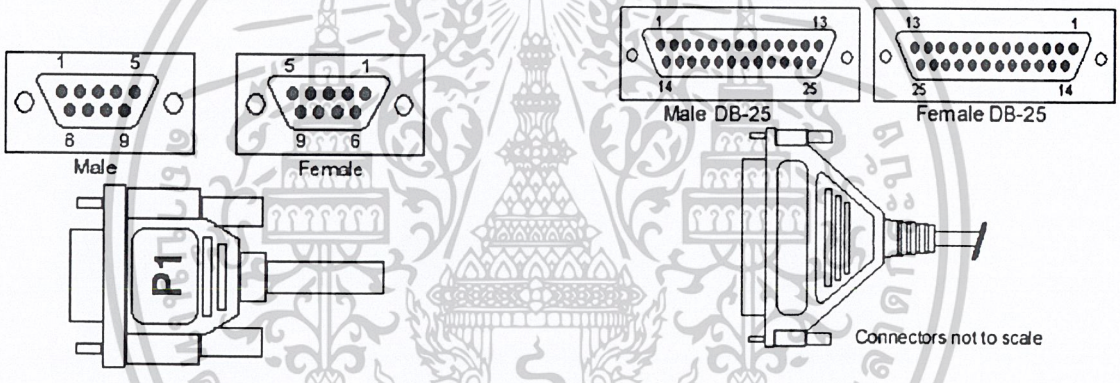
2.5 มาตรฐานการเชื่อมต่อพอร์ทอนุกรมแบบ RS-232

มาตรฐานการเชื่อมต่อแบบอนุกรม RS-232 เป็นมาตรฐานอุตสาหกรรมที่ออกแบบมาเพื่อใช้ในการส่งข้อมูลอนุกรมแบบอะซิงโครนัส 2 ทิศทาง โดยมาตรฐาน RS-232 ในอดีตนั้นถูกออกแบบมาเพื่อการส่งผ่านข้อมูลจากคอมพิวเตอร์ไปยังโมเด็มเพียงอย่างเดียว เพื่อที่จะนำข้อมูลจากโมเด็มนี้สื่อสารผ่านสายโทรศัพท์ไปยังคอมพิวเตอร์อีกชุดที่อยู่ห่างไกลกัน โดยคณะกรรมการที่เรียกว่า สมาคมอุตสาหกรรมอิเล็กทรอนิกส์ (Electronics Industries Association: EIA) ได้วางมาตรฐานที่มีชื่อเรียกกันว่า EIA RS-232

มาตรฐานนี้ในช่วงแรกจะใช้คอนเน็กเตอร์เป็นแบบ DB-25 โดยกำหนดความยาวสูงสุดของสายสัญญาณไว้ที่ 50 ฟุต มีระดับสัญญาณตั้งแต่ -3 ถึง -12V แสดงว่ามีข้อมูล (Mark) และ +3 ถึง +12V แสดงว่าเป็นช่องว่าง (Space)

มาตรฐาน RS-232 ได้กำหนดรูปแบบของอุปกรณ์เชื่อมต่อข้อมูล (Data Terminal Equipment: DTE) กับวงจรข้อมูลปลายทาง (Data Circuit Terminal: DCE) ไว้ว่าอุปกรณ์ DTE จะต้องเป็นอุปกรณ์ที่มีการประมวลผลในตัว เช่น ไมโครคอนโทรลเลอร์หรือไมโครคอมพิวเตอร์ ซึ่งมีความสามารถในการสร้างบิตข้อมูลแบบอนุกรมได้ ส่วนอุปกรณ์ DCE จะทำหน้าที่เป็นเพียงตัวรับข้อมูลที่ส่งมาจาก DTE เท่านั้น โดยการรับส่งข้อมูลระหว่างอุปกรณ์ทั้งสองจะกระทำผ่านมาตรฐาน RS-232

มาตรฐานการเชื่อมต่อแบบ RS-232 จะใช้คอนเน็กเตอร์แบบ DB-25 ตัวผู้หรือ DB-9 ตัวผู้ ซึ่งคอนเน็กเตอร์แบบ DB-25 จะมีขาต่อใช้งานเพียง 9 เส้น เช่นเดียวกับคอนเน็กเตอร์แบบ DB-9 เนื่องจากขาอื่นๆ ที่เคยใช้งานในอดีต ปัจจุบันมีการใช้งานไม่มากนัก จึงถูกยกเลิกไป โดยแสดงดังรูปร่างและตำแหน่งขาในรูปที่ 2.7



(ก) คอนเนคเตอร์แบบ DB-9 (ข) คอนเนคเตอร์แบบ DB-25

รูปที่ 2.7 แสดงคอนเนคเตอร์ที่ใช้กับพอร์ทอนุกรมแบบ RS-232

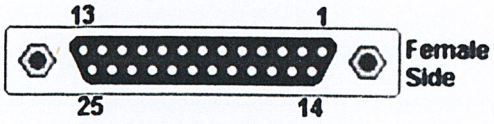
<u>รายละเอียดหน้าที่การทำงานในแต่ละขาของพอร์ทอนุกรม RS-232</u>

- **Data Carrier Detect: DCD หรืออาจเรียกว่า Carrier Detect: CD ขานี้จะแอกทีฟเมื่อมีการส่งสัญญาณพาห์จากอุปกรณ์สื่อสารข้อมูล เช่น โมเด็ม สำหรับการใช้งานปกติ ขานี้จะไม่ได้ถูกใช้งานมากนัก**
- **Receive Data: RD หรือ RxD ขานี้ใช้เพื่อรับสัญญาณอนุกรมเข้ามายังคอมพิวเตอร์โดยนำข้อมูลที่อ่านได้เก็บไว้ในรีจิสเตอร์ บัฟเฟอร์**
- **Transmitterd Data: TD หรือ TxD ขานี้ใช้เพื่อส่งข้อมูลออกจากคอมพิวเตอร์ โดยนำข้อมูลที่เก็บอยู่ในบัฟเฟอร์สำหรับส่งข้อมูลส่งออกไป**

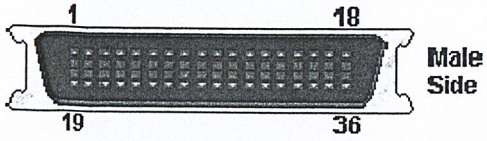
- **Data Terminal Ready: DTR** เป็นขาสัญญาณที่ส่งออกจากคอมพิวเตอร์เพื่อให้อุปกรณ์ปลายทางรับรู้ว่า ต้องการติดต่อด้วย โดยขา DTR นี้จะต้องเชื่อมต่อกับขา DSR ของอุปกรณ์ปลายทางและขา DTR ของอุปกรณ์ปลายทางจะต้องเชื่อมต่อกับขา DSR ของคอมพิวเตอร์ ถ้าใช้การเชื่อมต่อเป็นแบบโดยตรงโดยไม่ต้องผ่านโมเด็ม ซึ่งใช้สายในการเชื่อมต่อเพียง 3 เส้น จะต้องต่อขา DTR และ DSR ของตัวมันเองเข้าด้วยกันและต้องต่อกับขา DCD ด้วยในกรณีที่โปรแกรมสื่อสารที่ใช้มีการตรวจจับสัญญาณพาห์
- **Signal Ground: GND** ขากราวด์ของระบบ
- **Data Set Ready: DSR** ขานี้จะใช้คู่กับขา DTR เพื่อการตรวจสอบการเชื่อมต่อกันระหว่างคอมพิวเตอร์กับอุปกรณ์ปลายทาง ซึ่งขา DSR นี้จะเป็นขาสำหรับรับข้อมูลจากภายนอกซึ่งถูกส่งมาจากขา DTR
- **Request To Send: RTS** เป็นขาสำหรับส่งสัญญาณร้องขอให้ทางอุปกรณ์ปลายทางส่งข้อมูลกลับมายังคอมพิวเตอร์ โดยขาที่รับสัญญาณ RTS ก็คือขา CTS ในกรณีที่ใช้การเชื่อมต่อแบบโดยตรงโดยไม่ต้องผ่านโมเด็ม 3 สาย จะต้องเชื่อมต่อขา RTS และ CTS ของตัวมันเองเข้าด้วยกันเพื่อจะให้การรับและส่งข้อมูลสามารถเกิดขึ้นได้ตลอดเวลา
- **Clear To Send: CTS** ขานี้จะคอยรับสัญญาณจากขา RTS เมื่อรับสัญญาณได้ ข้อมูลที่ขา TxD จะถูกส่งออกไป ดังนั้นขานี้จึงถูกใช้เพื่อตรวจสอบอุปกรณ์ต่อพ่วงว่าพร้อมที่จะรับข้อมูลหรือไม่
- **Ring Indicator: RI** ใช้แสดงสถานะสัญญาณเรียกจากสายโทรศัพท์ ปกติในการสื่อสารโดยทั่วไป สายนี้จะถูกใช้งานก็ต่อเมื่อมีการเชื่อมต่อกับโมเด็มและโปรแกรมมีการตรวจสอบสัญญาณนี้เท่านั้น

2.6 มาตรฐานการเชื่อมต่อพอร์ทขนานและการสั่งงานพริ้นเตอร์

พอร์ทขนานที่ใช้กับเครื่องคอมพิวเตอร์ทั่วไปนั้น มีต้นกำเนิดมาจากการที่บริษัท IBM ได้ผลิตเครื่องคอมพิวเตอร์ส่วนบุคคล (PC) เครื่องแรกขึ้น แล้วมีโรงงานพริ้นเตอร์ที่มีชื่อเสียงได้ทำการพัฒนาสัญญาณควบคุมเครื่องพริ้นเตอร์โดยสร้างคอนเนคเตอร์ที่ใช้เชื่อมต่อกับเครื่องพริ้นเตอร์ ซึ่งเรียกว่า "Centronics" มี 36 พิน จากนั้นบริษัท IBM ก็ได้คิดค้นคอนเนคเตอร์ของตัวเองขึ้นมาเอง ซึ่งใช้ชื่อว่า "DB25" มี 25 พิน ทำให้เราจำเป็นต้องใช้สายแปลงระหว่างคอนเนคเตอร์ Centronics และ DB25 จนถึงทุกวันนี้



(ก) คอนเนคเตอร์แบบ DB-25 ตัวเมีย



(ข) คอนเนคเตอร์แบบ Centronics ตัวผู้

รูปที่ 2.8 คอนเนคเตอร์ที่ใช้กับพอร์ทขนานในการเชื่อมต่อพริ้นเตอร์

DB25			*Centronic 36 pin*	
Pin No.	Signal Name		Pin No.	Signal Name
1	Strobe		1	Strobe
2	Data 0		2	Data 0
3	Data 1		3	Data 1
4	Data 2		4	Data 2
5	Data 3		5	Data 3
6	Data 4		6	Data 4
7	Data 5		7	Data 5
8	Data 6		8	Data 6
9	Data 7		9	Data 7
10	Acknowledge		10	Acknowledge
11	Busy		11	Busy
12	Paper End		12	Paper End
13	Select		13	Select
14	Auto Feed		14	Auto Feed
15	Error		15	NC
16	Init		16	NC
17	Select In		17	NC
18	Ground		18	NC
19	Ground		19	Ground
20	Ground		20	Ground
21	Ground		21	Ground
22	Ground		22	Ground
23	Ground		23	Ground

24	Ground		24	Ground
25	Ground		25	Ground
			26	Ground
			27	Ground
			28	Ground
			29	Ground
			30	Ground
			31	Init
			32	Error
			33	Ground
			34	NC
			35	NC
			36	Select In

ตารางที่ 2.1 ขาของ DB25 และ Centronics

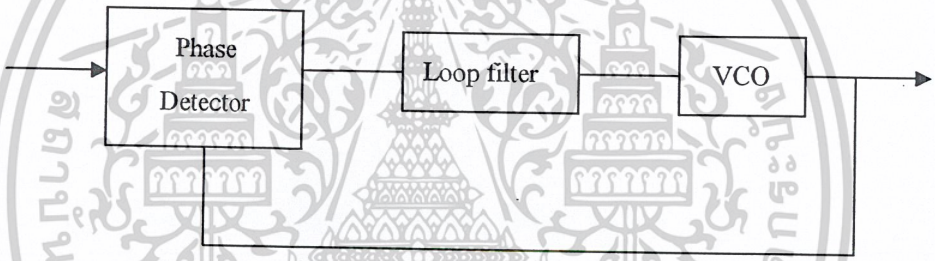
หน้าที่การทำงานของแต่ละขาสัญญาณของ DB25 มีดังนี้

- **Strobe** ใช้ในการบอกให้เครื่องพริ้นเตอร์รู้ว่าเมื่อใดจะมีการส่งข้อมูล โดยสัญญาณนี้ปกติจะเป็น high และจะเป็น low เมื่อมีการส่งข้อมูล
- **Data** เป็นสัญญาณ 8 เส้นที่ใช้ในการส่งข้อมูลของอักษรแต่ละตัว แต่ละเส้นจะส่ง 1 บิตใน 1 ไบต์ สัญญาณนี้จะเป็นไปตามมาตรฐาน TTL คือมีแรงดัน 5 โวลต์สำหรับลอจิก 1 และ 0 โวลต์สำหรับลอจิก 0
- **Acknowledge** สัญญาณนี้ใช้ในการแจ้งให้เครื่องคอมพิวเตอร์รู้ว่าข้อมูลตัวอักษรที่ส่งไปนั้นได้รับเรียบร้อยแล้ว โดยปกติสัญญาณนี้จะเป็น high และจะเป็น low เมื่อได้รับข้อมูลตัวอักษรเรียบร้อยและพร้อมที่จะรับตัวต่อไป โดยจะเป็น low ประมาณ 8 μ sec
- **Busy** โดยปกติข้อมูลแต่ละไบต์จะถูกส่งไปยังเครื่องพริ้นเตอร์ ใช้เวลาประมาณ 2 μsec หมายความว่าเครื่องพริ้นเตอร์จะต้องรับข้อมูล 500,000 ไบต์ต่อวินาที ซึ่งไม่มีเครื่องพริ้นเตอร์ที่มีความเร็วขนาดนั้น ดังนั้นเมื่อเครื่องพริ้นเตอร์ได้รับข้อมูล 1 ไบต์ สัญญาณนี้จะเป็น high เพื่อบอกให้คอมพิวเตอร์หยุดส่งข้อมูล และจะเป็น low เมื่อคอมพิวเตอร์พร้อมจะรับไบต์ถัดไป
- **Paper End** สัญญาณนี้จะแจ้งให้คอมพิวเตอร์รู้ว่าตอนนี้กระดาษหมด โดยสัญญาณนี้จะเป็น high เมื่อไม่มีกระดาษอยู่ในเครื่องพริ้นเตอร์
- **Auto Feed** สัญญาณนี้จะใช้ในการเลื่อนบรรทัด โดยจะเป็น low เมื่อมีการเลื่อนบรรทัด
- **Error** สัญญาณนี้จะเป็น high เมื่อไม่มีการตรวจพบความผิดพลาดจากพริ้นเตอร์ และจะเป็น low เมื่อมีความผิดพลาดเกิดขึ้น เช่น ไม่ได้ปิดฝาเครื่องพริ้นเตอร์ กระดาษติด เป็นต้น

- **Initialize Printer** ใช้ในการกำหนดค่าเริ่มต้นของเครื่องพริ้นเตอร์ มีประโยชน์ในการเริ่มพิมพ์งาน โดยจะเลื่อนส่วนหัวกระดาษให้อัตโนมัติ สัญญาณนี้ปกติจะเป็น high และจะเป็น low ขณะที่เลื่อนหัวกระดาษ
- **Select input** สัญญาณนี้จะแจ้งให้เครื่องคอมพิวเตอร์รู้ว่าขณะนี้เครื่องพริ้นเตอร์เชื่อมต่อ (online) อยู่หรือไม่ โดยจะเป็น high เมื่อเครื่องพริ้นเตอร์ไม่ได้เชื่อมต่ออยู่ และจะเป็น low เมื่อเครื่องพริ้นเตอร์ทำการเชื่อมต่ออยู่

2.7 ระบบเฟสล็อกลูป

เป็นระบบที่ใช้การป้อนกลับ (feed back) เพื่อเปรียบเทียบสัญญาณอินพุท แล้วสร้างสัญญาณเอาต์พุทที่มีความถี่และเฟสตรงกับสัญญาณอินพุท บล็อกไดอะแกรมของ Phase Lock Loop แสดงดังรูปที่ 2.9



รูปที่ 2.9 ระบบเฟสล็อกลูป

การทำงานของเฟสล็อกลูป

ขณะที่ยังไม่มีสัญญาณเข้าไปในระบบวงจรโวลเตจคอนโทรลออสซิลเลเตอร์ จะทำการผลิตความถี่ ω_0 ค่าหนึ่ง ซึ่งเรียกว่า “Free-running Frequency” ถ้ามีสัญญาณเข้าไปในระบบ เฟสดีเทคเตอร์จะทำการเปรียบเทียบเฟส และความถี่ของสัญญาณอินพุทกับความถี่ของวงจรโวลเตจคอนโทรลออสซิลเลเตอร์ และผลิตแรงดันคาดเคลื่อน ซึ่งสัมพันธ์กับความแตกต่างของเฟสและความถี่ระหว่างสัญญาณทั้งสอง แรงดันคลาดเคลื่อนนี้จะถูกกรอง และ ขยายส่งไปยังขาควบคุมของวงจรโวลเตจคอนโทรลออสซิลเลเตอร์ ซึ่งแรงดันควบคุมนี้ จะไปเปลี่ยนความถี่ของวงจรโวลเตจคอนโทรลออสซิลเลเตอร์ ให้เปลี่ยนไปในทิศทางที่ จะลดความแตกต่างของความถี่ระหว่าง Vo กับสัญญาณที่เข้า ถ้าความถี่ของสัญญาณที่เข้า ใกล้เคียงกับ Vo จะทำให้ เกิดการล็อกกับสัญญาณที่เข้ามา ขณะที่เกิดการล็อกนั้นความถี่ VCO จะเท่ากับสัญญาณอินพุทแต่เฟสยังต่างกันอยู่ ซึ่งเฟสที่ต่างกันนั้น มีความจำเป็นต่อการผลิตแรงดันคลาดเคลื่อน ที่จะ

สัญญาณอินพุทแต่เฟสยังต่างกันอยู่ ซึ่งเฟสที่ต่างกันนั้น มีความจำเป็นต่อการผลิตแรงดันคลาดเคลื่อน ที่จะไปคอยปรับความถี่วงจรโวลเตจคอนโทรลออสซิลเลเตอร์ จากค่า Free-running ให้เท่ากับความถี่ที่เข้ามา ดังนั้น เฟสล็อกลูปจะยังคงรักษาสภาพการล็อก การที่ระบบสามารถที่จะปรับตัวเองได้ทำให้เฟสล็อกลูปสามารถติดตามการล็อกกับสัญญาณที่เข้าไปให้อยู่ในสภาพล็อกเช่นเดิม เฟสล็อกลูปโดยทั่วไปจะสามารถจับความถี่ได้ในช่วงหนึ่ง กล่าวคือในขณะที่ เฟสล็อกลูปทำงานอย่างอิสระ (free running)แล้วเกิดมีสัญญาณความถี่อื่นเข้ามา เฟสล็อกลูปก็จะพยายามติดตามเพื่อจับความถี่นั้น ช่วงความถี่ที่เฟสล็อกลูปสามารถติดตามจับได้ตลอดเวลาเรียกว่า ช่วงความถี่ตรวจจับ (Capture Range) และเมื่อเฟสล็อกลูปติดตามจับความถี่ได้แล้วช่วงที่เฟสล็อกลูปจะติดตามจับได้ตลอดเวลา เรียกว่าช่วงความถี่ล็อก (Lock Range) โดยทั่วไปแล้วช่วงความถี่ล็อกจะกว้างกว่าช่วงความถี่ตรวจจับ

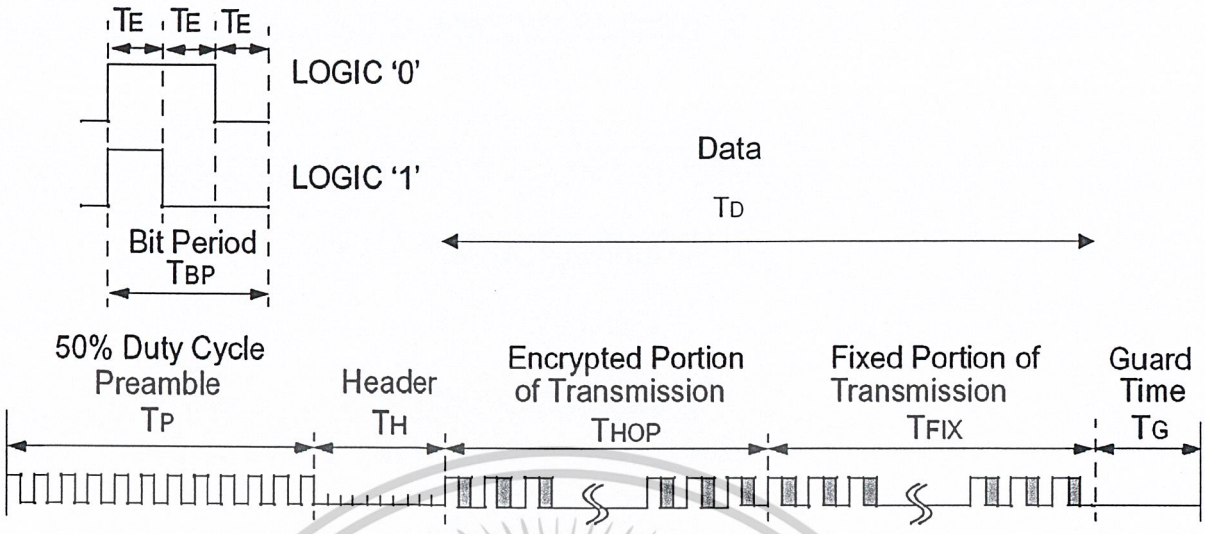
2.8 โปรโตคอล KeeLoq

KeeLoq เป็นโปรโตคอลที่กำหนดขึ้นโดยบริษัทไมโครชิป เพื่อใช้ในการเข้ารหัส (encoding) การถอดรหัส (decoding) ข้อมูลดิจิตอล และจัดรูปแบบในการรับส่งข้อมูล สำหรับไมโครคอนโทรลเลอร์ตระกูลต่างๆของบริษัทไมโครชิป

วิธีการเข้ารหัสสำหรับโปรโตคอลแบบ KeeLoq นี้ จะใช้วิธีการแทนข้อมูลดิจิตอลลอจิก “1” ด้วย “110” และแทนลอจิก “0” ด้วย “100” ดังตัวอย่างในรูปที่ 2.10 จะเห็นได้ว่า 1 บิตข้อมูลจะแทนค่าด้วยไบนารี่ 3 บิต ซึ่งแต่ละบิตจะมีคาบเท่ากับ 1 แมชชีนไซเคิล (1 T_E) ดังนั้น 1 บิตข้อมูลจะมีคาบเท่ากับ 3 T_E

รูปที่ 2.10 ตัวอย่างการเข้ารหัสสัญญาณข้อมูลดิจิตอลแบบ KeeLoq

สำหรับรูปแบบของเฟรมที่ใช้ในการส่งข้อมูลนั้นจะเริ่มต้นด้วยพรีแอมเบิล (Preamble) เป็นกลุ่มบิต “10101010101010101010101010101010” จำนวน 32 บิตซิงโครนัส เพื่อใช้ในการซิงโครนัสระหว่างด้านส่งและด้านรับ ตามด้วยส่วนเฮดเดอร์ (Header) ซึ่งมีคาบ (T_H) เท่ากับ 10 T_E เป็นช่องว่างเพื่อรอการส่งข้อมูล แล้วจึงส่งข้อมูลตามด้วยการ์ดไทม์ (Guard Time)



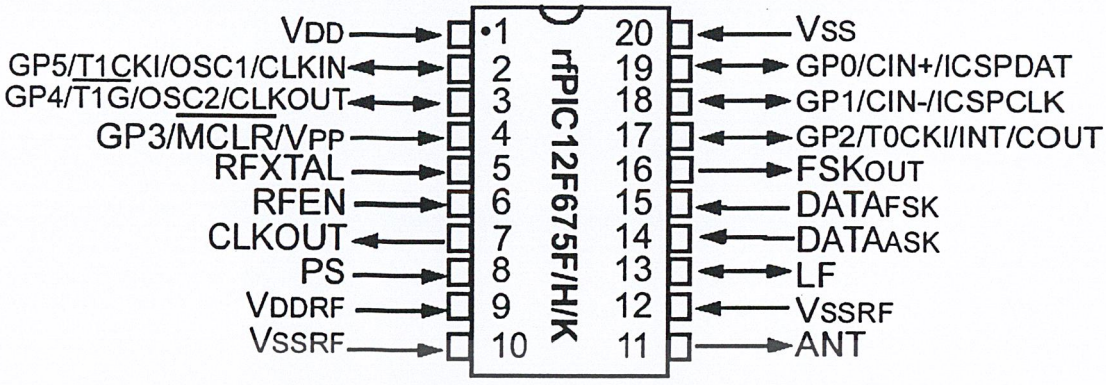
รูปที่ 2.11 สัญญาณและการจัดเฟรมแบบ KeeLoq

ข้อดีของการเข้ารหัสของโปรโตคอล KeeLoq

- จะได้สัญญาณที่มีระดับ DC component มีค่าประมาณศูนย์
- ในกรณีที่เกิดบิต 0 หรือ 1 ต่อเนื่องจะไม่เกิดการสูญเสียการซิงโครไนซ์
- กระบวนการในการตรวจจับสัญญาณสามารถเขียนโปรแกรมได้ง่าย

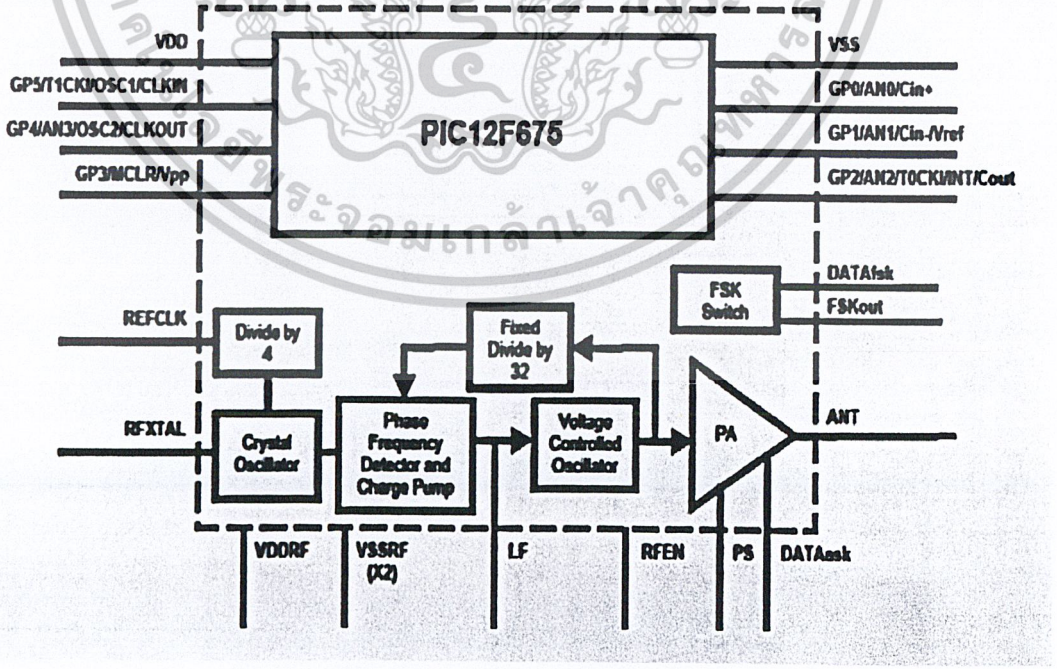
2.9 ไอซี rfPIC12f675

ไอซี rfPIC12f675 เป็นไมโครคอนโทรลเลอร์ตระกูล PIC ที่มีเครื่องส่งแบบ ASK และ FSK อยู่ในตัว มีจุดเด่นก็คือมีพอร์ทอินพุทเอาท์พุทถึง 6 ขา และยังบรรจุวงจรแปลงสัญญาณอนาลอกเป็นดิจิตอล 10 บิต ไว้ในตัวถึง 4 ช่อง ส่วนที่เป็นเครื่องส่งนั้นก็สามารถควบคุมความถี่ของคลื่นพาห์ได้ โดยการเปลี่ยนค่าคริสตัลออสซิลเลเตอร์ และตัวเก็บประจุเพียงไม่กี่ตัว อีกทั้งยังมีระบบเฟสล็อกลูป นอกจากนี้ยังมีโหมดประหยัดพลังงาน (Sleep mode) และยังมีชุดคำสั่งในการเขียนโปรแกรม เพียง 35 คำสั่งเท่านั้น จึงเหมาะเป็นอย่างยิ่งที่จะนำมาใช้กับงานควบคุมอัตโนมัติขนาดเล็กที่เป็นแบบไร้สาย



รูปที่ 2.12 ไอซี rfPIC12f675

รูปที่ 2.13 แสดงบล็อกไดอะแกรมของ ไอซี rfPIC12f675 จะเห็นว่าประกอบด้วย 2 ส่วนใหญ่ ด้านบนเป็นไมโครคอนโทรเลอร์ PIC12f675 ส่วนด้านล่างเป็นส่วนของเครื่องส่งซึ่งประกอบด้วย วงจรคริสตัลออสซิลเลเตอร์ ทำหน้าที่ผลิตคลื่นพาห์ ระบบเฟสล็อกลูป และวงจรขยายซึ่งสามารถปรับอัตราขยายได้



รูปที่ 2.13 บล็อกไดอะแกรมของไอซี rfPIC12f675

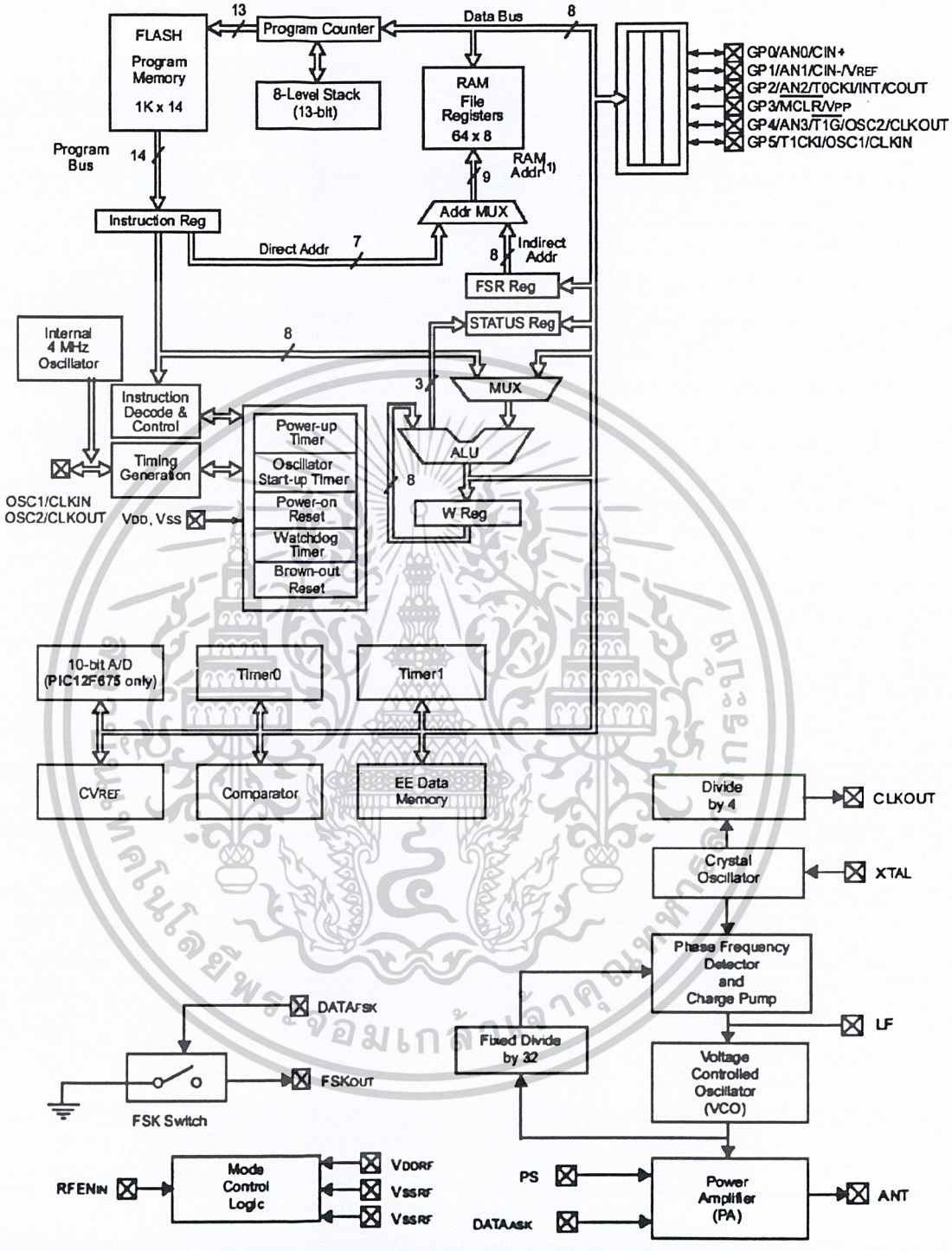
โครงสร้างของ rfPIC12f675 แสดงดังรูปที่ 2.13 ซึ่งมีส่วนประกอบหลักที่โดดเด่นคือ ไทเมอร์ที่มี 2 ตัว ซึ่งถือว่ามากเมื่อเทียบกับขนาดของมัน วงจรแปลงสัญญาณอนาลอกเป็นดิจิตอลขนาด 10 บิต โมดูลเปรียบเทียบแรงดันอนาลอก 2 ชุด (Analog Comparator) วงจรบราวเอาต์รีเซต (Brown-out reset : BOR) ส่วนของเครื่องส่งซึ่งใช้ระบบเฟสล็อกลูปซึ่งประกอบด้วย วงจรเฟสดีเทคเตอร์(Phase Detector) วงจรโวลเตจคอนโทรลออสซิลเลเตอร์ (VCO)วงจรหารความถี่ด้วย 32 และส่วนที่เป็นหน่วยความจำต่างๆ

<u>คุณสมบัติของไมโครคอนโทรลเลอร์ rfPIC12f675</u>

- ซีพียูเป็นแบบ RISC (Reduce Instruction – Set Computer) มีคำสั่งใช้งานเพียง 35 คำสั่ง
- ความถี่สัญญาณนาฬิกา ตั้งแต่ไฟตรงถึง 20 MHz
- ขนาดหน่วยความจำโปรแกรม 1 กิโลเวิร์ด
- หน่วยความจำแรมข้อมูล 64 ไบต์
- หน่วยความจำข้อมูลอีอีพรอม 128 ไบต์
- ตอบสนองแหล่งกำเนิดอินเตอร์รัปต์ได้ 10 แหล่ง
- มีสแต็ก 8 ระดับ
- มีวงจรเพาเวอร์ออนรีเซต (POR), เพาเวอร์อัปไทเมอร์ (PWRT) และออสซิลเลเตอร์สตาร์ตอัปไทเมอร์ (OST)
- มีวอตช์ด็อกไทเมอร์ (WDT) ที่มีวงจรออสซิลเลเตอร์ในตัว ทำให้มีความน่าเชื่อถือในการทำงานสูง
- เลือกป้องกันข้อมูลทั้งในหน่วยความจำโปรแกรมและหน่วยความจำข้อมูล
- สามารถโปรแกรมในวงจรได้
- ไฟเลี้ยง +3 ถึง +5.5V
- ขาพอร์ทสูงสุด 6 บิต เมื่อทำงานในโหมด INTRC และกำหนดให้ MCLR เป็นพอร์ทอินพุท
- ไทเมอร์ 2 ตัว (ไทเมอร์ 0 และไทเมอร์ 1)
- มีโมดูลเปรียบเทียบแรงดันอนาลอก 1 ชุด
- มีโมดูลสร้างแรงดันอ้างอิงสำหรับวงจรเปรียบเทียบแรงดันอนาลอก
- มีวงจรแปลงสัญญาณอนาลอกเป็นดิจิตอล 10 บิต จำนวน 4 ช่อง
- มีวงจรตรวจจับระดับแรงดันไฟเลี้ยงหรือบราวเอาต์ดีเท็กชั่น (Brown-out detection) เพื่อสร้างสัญญาณรีเซตซีพียูหรือเรียกว่า บราวเอาต์รีเซต
- การใช้พลังงานไฟฟ้าในกรณีไม่ขับโหลด
    - น้อยกว่า 2 mA ที่ +5 V และสัญญาณนาฬิกา 4 MHz, 15uA ที่ +3V และ สัญญาณนาฬิกา 32 kHz
    - น้อยกว่า 1 uA ในโหมดประหยัดพลังงานหรือสแตนด์บายที่ไฟเลี้ยง +3 V

<u>คุณสมบัติของเครื่องส่ง</u>

- ประกอบด้วยคริสตัลออสซิลเลเตอร์ วงจรโวลเตจคอนโทรลออสซิลเลเตอร์ และวงจรขยาย
- มีอัตราเร็ว 0 ถึง 40 kbps สำหรับการส่งแบบ ASK
- มีอัตราเร็ว 0 ถึง 20 kbps สำหรับการส่งแบบ FSK
- มีกำลังส่งตั้งแต่ -6 dBm ถึง -15 dBm
- สามารถปรับกำลังส่งได้
- การตั้งความถี่ทำได้โดยการคูณความถี่ของคริสตัลออสซิลเลเตอร์ด้วย 32
- มีวงจรเฟสล็อกลูป



รูปที่ 2.14 โครงสร้างภายในไอซี rfPIC12f675

รายละเอียดการใช้งานขาต่างๆ

VDD - ต่อไฟเลี้ยงบวกตั้งแต่ 2 ถึง5.5 V

VSS - ต่อกราวด์

GP0 - พอร์ทอินพุทเอาท์พุทดิจิตอล
- อินพุทวงจรแปลงสัญญาณอนาลอกเป็นดิจิตอล
- อินพุทบวกวงจรเปรียบเทียบแรงดันอนาลอก
- ขาสัญญาณข้อมูลของการโปรแกรม

GP1 - พอร์ทอินพุทเอาท์พุทดิจิตอล
- อินพุทวงจรแปลงสัญญาณอนาลอกเป็นดิจิตอล
- อินพุทลบวงจรเปรียบเทียบแรงดันอนาลอก
- อินพุทรับแรงดันอ้างอิงจากภายนอกสำหรับวงจรแปลงสัญญาณอนาลอกเป็นดิจิตอล
- ขาสัญญาณข้อมูลของการโปรแกรม

GP2 - พอร์ทอินพุทเอาท์พุทดิจิตอล
- อินพุทวงจรแปลงสัญญาณอนาลอกเป็นดิจิตอล
- อินพุทสัญญาณนาฬิกาของไทเมอร์ 0
- อินพุทรับสัญญาณอินเตอร์รัปต์จากภายนอก
- เอาท์พุทของวงจรเปรียบเทียบแรงดันอนาลอก

GP3 - พอร์ทอินพุทดิจิตอล
- ขารีเซตหลัก
- อินพุทรับแรงดันสูงสำหรับการโปรแกรมหน่วยความจำ

GP4 - พอร์ทอินพุทเอาท์พุทดิจิตอล
- อินพุทวงจรแปลงสัญญาณอนาลอกเป็นดิจิตอล
- อินพุทสัญญาณเปิดเกตสำหรับไทเมอร์ 1
- ขาต่อคริสตัลหรือเซรามิกเรโซเนเตอร์
- เอาท์พุทสัญญาณความถี่ 1/4 ของสัญญาณนาฬิกาหลัก

GP5 - พอร์ทอินพุทเอาท์พุทดิจิตอล
- อินพุทสัญญาณนาฬิกาของไทเมอร์ 1
- ขาต่อคริสตัลหรือเซรามิกเรโซเนเตอร์
- อินพุทสัญญาณนาฬิกาจากภายนอก และต่อวงจรตัวต้านทานและตัวเก็บประจุเพื่อกำหนดความถี่ของสัญญาณนาฬิกาเมื่อทำงานในโหมด RC

RFXTAL - ใช้ต่อคริสตัลออสซิลเลเตอร์

RFEN - อินพุทสัญญาณ enable เครื่องส่ง

CLKOUT - เอาท์พุทสัญญาณนาฬิกา

PS - ปรับกำลังขยายของวงจรขยายสัญญาณ

VDDRF - ขาป้อนแรงดันเปรียบเทียบบวก

VSSRF - ขาป้อนแรงดันเปรียบเทียบลบ

FSKOUT - เอาท์พุทสัญญาณที่ผ่านการมอดูเลตแบบ FSK

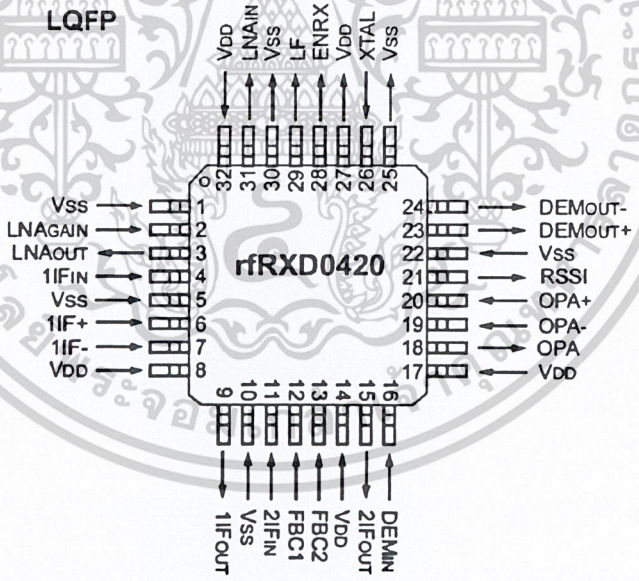
DATA FSK - อินพุทสัญญาณที่จะมอดูเลตแบบ FSK

DATA ASK - อินพุทสัญญาณที่จะมอดูเลตแบบ ASK

LF - ขาต่อวงจรลูปฟิลเตอร์

ANT - ขาสำหรับต่อสายอากาศ

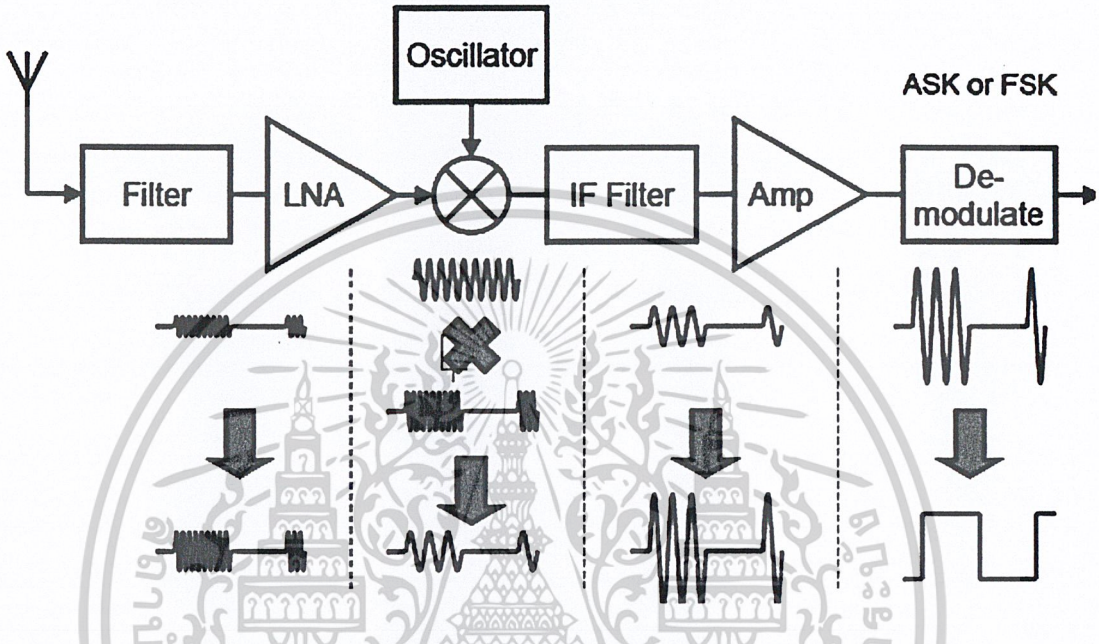
2.10 ไอซี rfRXD0420



รูปที่2.15 ไอซี rfRXD0420

ไอซี rfRXD0420 เป็นไอซีเครื่องรับที่มีขนาดเล็ก ออกแบบมาเพื่อใช้คู่กับเครื่องส่งของไมโครคอนโทรลเลอร์ตระกูล rfPIC สามารถตั้งความถี่ได้ง่าย เพียงแค่เปลี่ยนคริสตัลออสซิลเลเตอร์ และตัวเก็บประจุเพียงไม่กี่ตัวเช่นเดียวกับเครื่องส่งของไอซี rfPIC12f675

รูปที่ 2.16 แสดงบล็อกไดอะแกรมของ ไอซี rfRXD0420 ซึ่งเป็นเครื่องรับแบบซูเปอร์เฮทเธอโรไดน์ (Superheterodyne) มีส่วนประกอบต่าง ๆ ดังแสดงในรูป

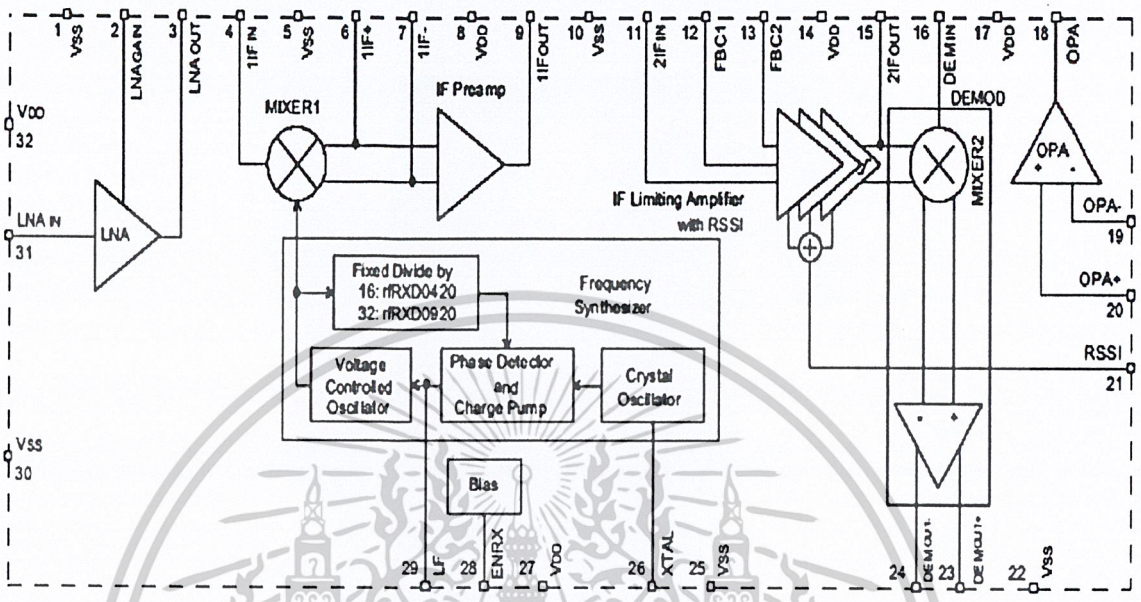


รูปที่ 2.16 บล็อกไดอะแกรมของ ไอซี rfRXD0420

<u>คุณสมบัติของไอซี rfRXD0420</u>

- สามารถปรับความถี่ได้ตั้งแต่ 300 MHz – 450 MHz โดยการเปลี่ยนคริสตัล ซึ่งสามารถคำนวณได้จาก $f_{osc} = f_{crys} \times 32$
- มีบิตเรตสูงสุด 80 kbps ในการคีมอดูเลตสัญญาณ NRZ แบบ ASK และ 40 kbps สำหรับการคีมอดูเลตแบบ FSK
- ความถี่ IF มีค่าตั้งแต่ 455 kHz – 21.4 MHz
- ใช้ระบบเฟสล็อกลูป
- สามารถปรับ Gain ของ Low Noise Amplifier (LNA) ได้
- ทนอุณหภูมิได้ตั้งแต่ –40 $^{\circ}$c ถึง 85 $^{\circ}$c

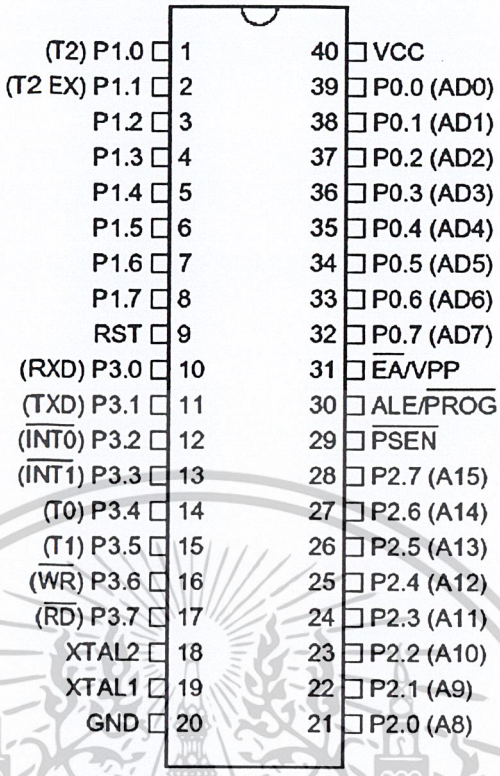
โครงสร้างของไอซี rfRXD0420 แสดงดังรูปที่ 2.17 มีส่วนประกอบที่สำคัญคือ วงจรขยายสัญญาณ วงจรมิกเซอร์ วงจรคีมอดูเลต และระบบเฟสล็อกลูป ซึ่งประกอบด้วย วงจรเฟสดีเทคเตอร์ วงจรโวลเตจคอนโทรลออสซิลเลเตอร์ และวงจรหารความถี่ด้วย 32



รูปที่ 2.17 โครงสร้างภายในของไอซี rfRXD0420

2.11 ไมโครคอนโทรลเลอร์ตระกูล MCS-51

ไมโครคอนโทรลเลอร์แบบชิปเดี่ยว (Single Chip Microcontroller) คือไมโครคอมพิวเตอร์แบบที่มีขนาดเล็กโดยบรรจุไว้ในแผงวงจรรวม (Integrated Circuit) เพียงชิปเดียวเหมาะสำหรับควบคุมอุปกรณ์อื่นๆโดยอัตโนมัติ โดยผู้ใช้สามารถเขียนโปรแกรมควบคุมได้ตามต้องการ ซึ่งไมโครคอนโทรลเลอร์ชิปเดี่ยวตระกูล 51 หรือ MCS-51 ที่เลือกใช้ในโครงงานนี้ คือเบอร์ AT89C52 ของบริษัท ATMEL ซึ่งเป็นไมโครคอนโทรลเลอร์ที่มีหน่วยความจำโปรแกรม (ROM) ภายในแบบ Flash Memory ขนาด 8Kbytes ซึ่งเป็นหน่วยความจำที่สามารถเขียนและลบข้อมูลได้ไม่ถึง 1,000 ครั้ง โดยที่ไม่ต้องใช้หน่วยความจำแบบ EPROM ภายนอก และสะดวกต่อการพัฒนาโปรแกรม



รูปที่ 2.18 ไมโครคอนโทรลเลอร์ตระกูล MCS-51

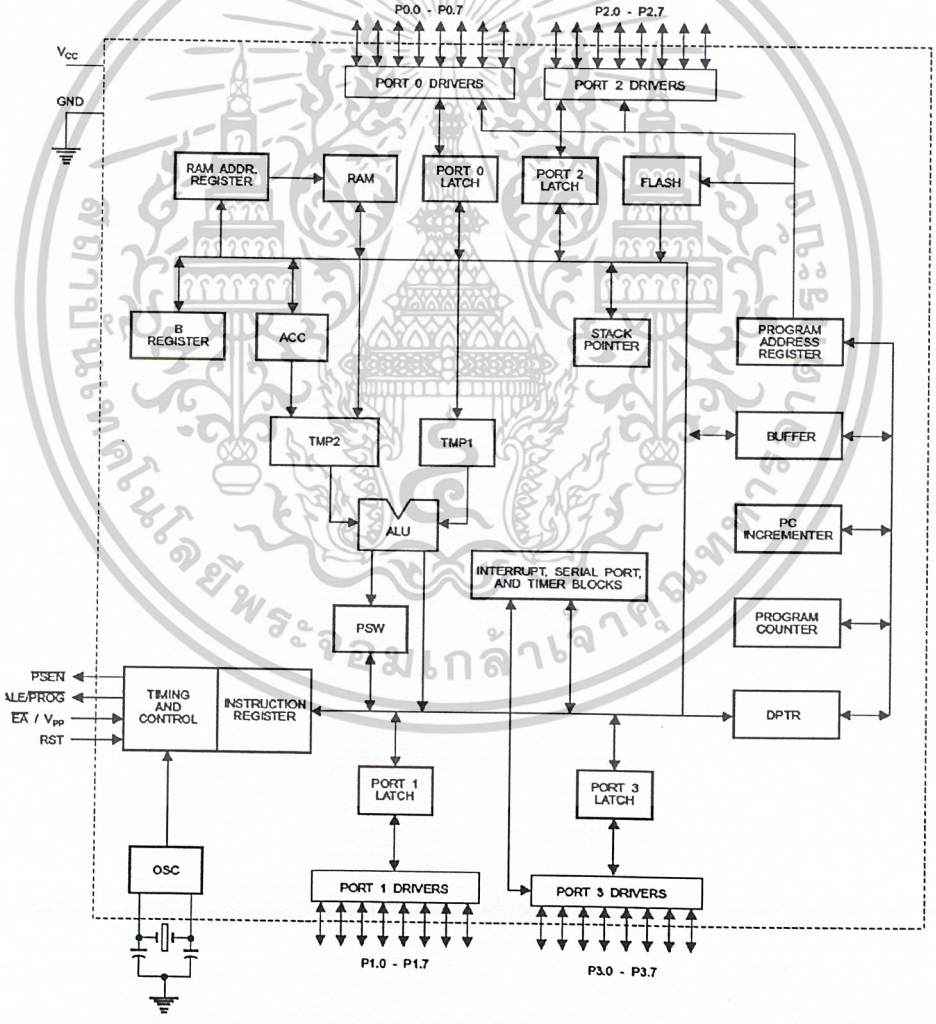
<u>คุณสมบัติของไมโครคอนโทรลเลอร์ MCS-51</u>

- มีหน่วยความจำสำหรับเก็บโปรแกรมควบคุมการทำงานอยู่ภายใน 8 Kbytes ซึ่งเพียงพอต่อการทำงานของโครงงานนี้
- สามารถต่อหน่วยความจำข้อมูล (Data Memory) ซึ่งเป็นหน่วยความจำสำหรับเก็บข้อมูลในระหว่างการทำงานของโปรแกรม (RAM) ได้ 128 bytes
- สามารถใช้กับหน่วยความจำโปรแกรมภายนอก (Program Memory) ซึ่งเป็นหน่วยความจำที่ใช้สำหรับเก็บชุดคำสั่งที่จะทำให้ MCS-51 ทำงานได้สูงสุด
- มีคำสั่งคูณและหารเลขขนาด 8 บิต
- มีการติดต่อสื่อสารข้อมูลแบบอนุกรม (Series) หรือ Universal Asynchronous Receiver Transmitter (UART)
- มีวงจร Timer/Counter ขนาด 16 บิต 2 ชุด
- มีการขอขัดจังหวะการทำงานของโปรแกรม (Interrupt)
- ต้องการแหล่งจ่ายไฟ 5 โวลต์เพียงชุดเดียว
- สามารถเลือกการทำงานให้อยู่ในโหมดของ Idle (การทำงานปกติ) และ Power down (การประหยัดพลังงานไฟฟ้า)

จากคุณสมบัติที่กล่าวถึงทำให้ MCS-51 เป็นที่นิยมใช้ในการควบคุมระบบอัตโนมัติ ซึ่งบรรจุไว้ในไอซีวงจรรวมเดี่ยว (Single Chip) ขนาด 40 ขา ดังนั้นจึงสามารถออกแบบให้ระบบมีขนาดเล็ก ทำให้ตรวจสอบข้อผิดพลาดได้ง่าย รวมถึงการลดต้นทุนการผลิตหากจะต้องมีการผลิตเป็นจำนวนมาก

โครงสร้างของ MCS-51

ภายใน MCS-51 จะประกอบขึ้นด้วยเกท (GATE) ต่างๆ เช่น AND, OR, NOT ซึ่งเกทเหล่านี้ จะถูกนำมาออกแบบให้มีหน้าที่การทำงานต่างๆ เช่นวงจรถอดรหัสคำสั่ง (Instruction Decoder) วงจรสร้างสัญญาณนาฬิกา (Clock Signal Generator) โครงสร้างภายในของ MCS-51 จะประกอบด้วยส่วนย่อยๆดังไดอะแกรมในรูป 2.19



รูปที่ 2.19 ไดอะแกรมโครงสร้างของ MCS-51

ไดอะแกรมในรูปที่ 2.19 เป็นโครงสร้างใหญ่ของ MCS-51 เนื่องจากลักษณะของ MCS-51 เป็นคอมพิวเตอร์ ประกอบด้วย 3 ส่วนหลักๆ คือ

ส่วนที่1 คือ ตัวประมวลผล (CPU:Central Processsing Unit) ส่วนนี้จะมีวงจรที่ทำหน้าที่สร้างสัญญาณควบคุมในการติดต่อกับส่วนอื่นๆ เรียกว่าวงจรควบคุม (Control Unit) สัญญาณที่สร้างจากวงจรควบคุมได้แก่สัญญาณสำหรับการติดต่อกับหน่วยความจำ อุปกรณ์รับข้อมูลเข้าหรือส่งออกจากตัว MCS-51 ซึ่งส่วนควบคุมด้วยการขัดจังหวะ (Interrupt Control) ส่วนควบคุมบัส (Bus Control) ก็เป็นส่วนหนึ่งของวงจรควบคุมด้วยการสร้างสัญญาณควบคุมจากส่วนประมวลผลนี้ จะทำการสร้างสัญญาณโดยการถอดรหัสคำสั่ง (Instruction) ตามที่มีการกำหนดไว้และสัญญาณที่สร้างขึ้นมาจะอ้างอิงกับสัญญาณนาฬิกาจากวงจรออสซิลเลเตอร์เพื่อให้ทุกๆส่วนในวงจรทำงานประสานกัน (Synchronize) อย่างถูกต้อง ในตัวซีพียูนี้ยังประกอบด้วยส่วนย่อยอีกส่วนที่เรียกว่าส่วนประมวลผล (Arithmetic Logic Unit) ส่วนนี้จะทำหน้าที่ประมวลผล เช่น การบวก, การลบ, การคูณ หรือการหารข้อมูล แล้วผลลัพธ์ไปเก็บไว้ในรีจิสเตอร์หรือหน่วยความจำที่ต้องการ

ส่วนที่2 คือ หน่วยความจำ (Memory) มีไว้สำหรับจดจำข้อมูล ถ้าจะให้เห็นภาพพจน์ของหน่วยความจำได้ดีก็คือ หน่วยความจำเปรียบเสมือนกล่องเก็บเอกสารจำนวนมากที่นำมาต่อเรียงไว้ แต่ละกล่องก็มีเอกสาร 1 แผ่น ถ้าต้องการเอกสารจากกล่องใดหรือเอาเอกสารไปเก็บที่กล่องใด จะต้องรู้หมายเลขของกล่องข้อมูลเสียก่อน ซึ่งถ้าเป็นหน่วยความจำแล้ว หมายเลขของกล่องก็คือตำแหน่งของหน่วยความจำหรือแอดเดรส (Address) นั่นเอง การเอาข้อมูลไปเก็บในหน่วยความจำเรียกว่า การเขียน (Write) ข้อมูล และการเอาข้อมูลออกจากหน่วยความจำจะเรียกว่าการอ่าน (Read) ข้อมูลซึ่งแต่ละตำแหน่งของหน่วยความจำจะเก็บข้อมูลได้เพียงค่าเดียวเท่านั้น ในไมโครโปรเซสเซอร์ทั่วไปรวมทั้ง MCS-51 นั้น ข้อมูลในแต่ละตำแหน่งของหน่วยความจำจะมีค่าได้เพียง 8 หลักของเลขฐาน 2 (8 บิตเท่ากับ 1 ไบท์) ดังนั้นแต่ละตำแหน่งของหน่วยความจำจะเก็บข้อมูลมีค่าได้ระหว่าง 0 ถึง 25 (00000000 ถึง 11111111 ในเลขฐาน 2) แต่จำนวนตำแหน่งที่จะเก็บข้อมูลได้ขึ้นกับไมโครโปรเซสเซอร์แต่ละเบอร์ การติดต่อกับหน่วยความจำจะต้องมีสัญญาณ 3 กลุ่ม คือ

1. แอดเดรส หรือค่าตำแหน่งที่ต้องการติดต่อกับหน่วยความจำใน MCS-51 จะติดต่อกับหน่วยความจำประเภทโปรแกรมเมมโมรี่ หรือ ดาต้าเมมโมรี่ ได้สูงสุดชนิดละ 65,536 ตำแหน่ง ดังนั้นการอ้างอิงแต่ละตำแหน่งของหน่วยความจำจะต้องใช้เส้นแสดงตำแหน่งในเลขฐาน 2 ทั้งหมด 16 เส้น (2^{16} เท่า 64 x 1024 = 65536)

2. ข้อมูลที่อ่านหรือเขียนกับหน่วยความจำ เพื่อบอกกับหน่วยความจำว่าต้องการอ่านหรือเขียนข้อมูล

3. สัญญาณควบคุมที่จะส่งไปยังหน่วยความจำ เพื่อบอกหน่วยความจำว่าต้องการอ่านหรือเขียนข้อมูลสัญญาณเหล่านี้จะถูกวงจรควบคุมภายใน MCS-51 สร้างมาจากวงจรถอดรหัสของคำสั่งที่ MCS-51 อ่านจากหน่วยความจำโปรแกรมเมมโมรี่เข้าไปทำงานนั้นเอง

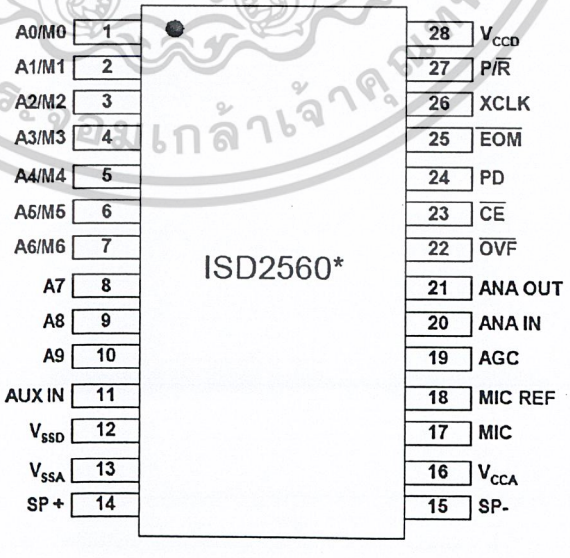
ส่วนที่3 อุปกรณ์อินพุท-เอาท์พุท (Input/Output Device) เป็นส่วนที่จะใช้ส่งข้อมูลหรือออกจาก MCS-51 ทำให้ MCS-51 ติดต่อกับภายนอกได้ ดังในไดอะแกรมรูปที่ 2.20 ซึ่งอุปกรณ์อินพุทและเอาท์พุท ได้แก่ พอร์ทอินพุท/เอาท์พุท 4 พอร์ท (4 I/O Port), ไทเมอร์0 (Timer0), ไทเมอร์1 (Timer1), พอร์ทอนุกรม (Serial Port) การทำงานของแต่ละส่วน มีดังนี้

1. พอร์ทอินพุท/เอาท์พุท 4 พอร์ท คำว่าพอร์ทหมายถึงจุดที่จะติดต่อกับส่วนที่อยู่ภายนอก พอร์ทอินพุท/เอาท์พุท 4 พอร์ท ของ MCS-51 เป็นที่ใช้สำหรับรับ-ส่งข้อมูลซึ่งเป็นสัญญาณดิจิตอลเข้าหรือออกจากตัว MCS-51 พอร์ทมีทั้งหมด 4 พอร์ท โดยแต่ละพอร์ทจะรับส่งข้อมูลได้ 8 บิต พอร์ท P0, P1, P2 และ P3 บางพอร์ทจะใช้ทำงานมากกว่าหนึ่งอย่างก็ได้ เช่น พอร์ท P0 และ P2 จะใช้สำหรับการส่งค่าตำแหน่ง (Address) ของหน่วยความจำที่ต้องการติดต่อ และพอร์ท P0 จะใช้รับส่งข้อมูลเมื่อติดต่อกับหน่วยความจำได้ด้วย แต่สิ่งเหล่านี้ไม่ได้เกิดขึ้นที่เวลาเดียวกัน แต่จะใช้วิธีทำงานตามลำดับ โดยควบคุมจากสัญญาณควบคุม (Control) ที่ถอดรหัสมาจากแต่ละคำสั่งที่ให้คอมพิวเตอร์ทำงานนั่นเอง และสัญญาณทั้งหมดจะอ้างอิงกับสัญญาณนาฬิกา

2. ไทเมอร์0 (Timer 0) และไทเมอร์1 (Timer 1) เป็นวงจรนับที่สามารถกำหนดให้ทำการนับจำนวนไซเคิลของสัญญาณที่ต่อจากภายนอก MCS-51 หรือจำนวนไซเคิลของสัญญาณนาฬิกาภายใน MCS-51 ก็ได้ ค่าจากการนับจะถูกอ่านหรือตั้งค่าเริ่มต้นของการนับได้โดย ซีพียู

3. พอร์ทอนุกรม (Serial Port) ซีพียูจะอ่านและเขียนข้อมูลกับพอร์ทอนุกรมเป็นแบบ 8 บิต แต่ข้อมูลจะถูกส่งออกจาก MCS-51 เรียงไปทีละบิตออกจากขา TXD และในการรับข้อมูลเข้า ก็รับเข้ามาทีละบิตทางขา RXD แล้วจัดเรียงใหม่เป็น 8 บิต เพื่อให้ซีพียูอ่านไปใช้งานต่อไป

2.12 ไอซี ISD2560



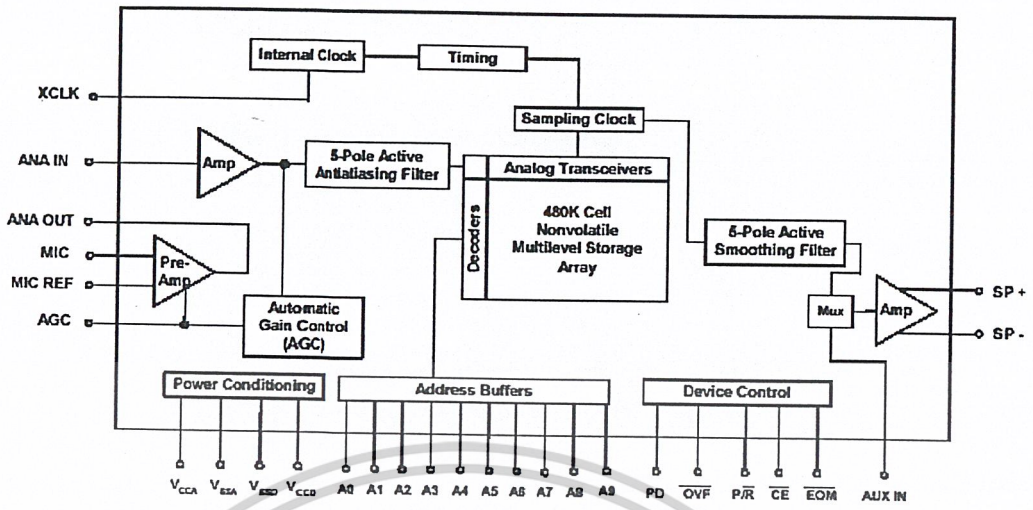
รูปที่ 2.20 ไอซี ISD2560

คุณสมบัติของ ISD2560

- ใช้ไอซีเพียงตัวเดียวก็สามารถบันทึกและเล่นกลับได้
- ไม่ต้องมีอุปกรณ์ประเภทไอซีอื่นๆ ประกอบร่วมภายนอก
- ไม่ต้องพัฒนาระบบอื่นขึ้นมาเสริมเพื่อให้ใช้งานได้
- มีประสิทธิภาพในการบันทึกและเล่นกลับโดยให้เสียงได้เหมือนต้นกำเนิดเสียง
- สามารถควบคุมการบันทึกและการเล่นกลับได้ด้วยไมโครคอนโทรลเลอร์ได้
- สามารถต่อคาดเคด (cascade) กันเพื่อเพิ่มหน่วยความจำในการเก็บข้อมูลให้ได้มากขึ้น
- การทำงานโดยอัตโนมัติเมื่อไม่มีการบันทึกหรือเล่นกลับนานเกินไป
- สามารถเก็บความจำไว้ได้นาน 100 ปี ไม่ต้องมีแบตเตอรี่สำรอง
- สามารถบันทึกซ้ำได้ถึง 100,000 ครั้ง
- วงจรกำเนิดสัญญาณนาฬิกาภายในตัว
- สามารถโปรแกรมควบคุมการเล่นกลับเพียงอย่างเดียวเพื่อพัฒนารูปแบบใช้งาน

โครงสร้างภายในของ ISD2560

จากคุณสมบัติต่างๆที่รวมอยู่ในไอซีเพียงตัวเดียว จึงทำให้ง่ายต่อการใช้งานตั้งแต่วงจรขยายสัญญาณจากไมโครโฟนจนถึงหน่วยจัดเก็บข้อมูลที่ทำการบันทึกและขับออกลำโพง ก็ถูกรวมไว้ในไอซีเพียงตัวเดียว ในโหมดการบันทึกจะจัดเก็บข้อมูลต่างๆไว้ในหน่วยความจำที่เป็นเซลแบบไม่ต้องการแรงดันสำรอง เพื่อเก็บรักษาข้อมูลไม่ให้เสียหาย (Non-volatile memory cells) สัญญาณเสียงที่อยู่ในรูปแบบของสัญญาณอนาลอกจะถูกบันทึกไว้ในหน่วยจัดเก็บความจำ โดยตรงโดยอาศัยเทคโนโลยีDAST (Direct Analog Storage Technology) และการจัดเก็บความจำก็จะจัดเก็บในลักษณะที่เป็นสัญญาณอนาลอกอยู่เช่นเดิม จึงทำให้การเล่นกลับสามารถให้สัญญาณเสียงที่เหมือนกับต้นกำเนิดเสียงมาก เพราะไม่มีขบวนการเปลี่ยนสัญญาณอนาลอกเป็นดิจิตอลเข้ามาเกี่ยวข้อง ซึ่งสามารถแสดงโครงสร้างได้ดังรูปที่ 2.21



รูปที่ 2.21 บล็อกไดอะแกรมภายในของ ISD2560

ขณะทำการบันทึก (Recording) ISD2560 จะเกิดขึ้นได้นั้น สัญญาณที่เข้ามาจะต้องตรงตามเงื่อนไข ซึ่งการทำสัญญาณให้ตรงตามเงื่อนไขที่จะสามารถทำการบันทึกได้นั้นจะต้องผ่านหลายขั้นตอนด้วยกัน ขั้นแรกคือการขยายสัญญาณอินพุท ให้มีขนาดมากพอสำหรับวงจรบันทึกข้อมูล ซึ่งกระบวนการนี้จะถูกกระทำโดยส่วนปรีแอมป์และวงจรควบคุมอัตราการขยาย AGC

ปรีแอมป์ (Preamplifier) เป็นส่วนที่ต่อกับไมโครโฟน โดยมีตัวเก็บประจุ (DC blocking capacitor) ทำหน้าที่แยกองค์ประกอบทางดีซี (DC component) ออกจากสัญญาณเอซี (AC signal) ที่มีระดับต่ำมากๆ ประมาณ 2-20 mV

วงจรควบคุมอัตราการขยาย (AGC circuit) จะทำหน้าที่ ควบคุมระดับของสัญญาณที่ออกมาจากปรีแอมป์ และส่งค่าโวลท์เตจที่ถูกควบคุมอัตราการขยายแล้วไปยังปรีแอมป์ อัตราการขยายของปรีแอมป์จะถูกปรับค่าอย่างอัตโนมัติ เพื่อที่จะคงขนาดของสัญญาณให้เหมาะสมเพียงพอที่จะส่งต่อไปในส่วนของวงจรกรองสัญญาณ (filter)

วงจรกรองความถี่ (5-Pole Active Antialiasing Filter) จะทำหน้าที่ปรับสัญญาณให้เหมาะสมและลดทอนส่วนที่ไม่สำคัญออกไป จึงได้เฉพาะเสียงที่มีคุณภาพดีในการบันทึกเป็นลำดับไป

สัญญาณที่ผ่านการปรับจนเหมาะสมแล้ว จะถูกเขียนลงอย่างต่อเนื่องลงในหน่วยความจำอนาลอก ในรูปการรวมสัญญาณ

การเล่นกลับ (Playback) อนาลอกโวลท์เตจที่ถูกบันทึกไว้จะถูกอ่านออกมาจากหน่วยความจำอย่างต่อเนื่อง จะมีส่วนทำการแยกสัญญาณให้กลับมาอยู่ในสภาพเดิม สัญญาณอนาลอกก็จะผ่านลำโพงออกมาได้ ลำโพงอาจมีกำลังขับประมาณ 125 mW RMS (25mW peak) 16โอห์มก็เพียงพอแล้วที่จะได้ยินอย่างชัดเจนในห้องขนาดธรรมดา กรณีที่ใช้ ISD2560 ต่อคาสเคดกันหลายๆตัว เราสามารถให้ลำโพงตัวเดียวกันได้โดยผ่านเข้า AUX IN

การใช้งานในโหมดที่สอง

การใช้งานจะมีอยู่ 3 แบบด้วยกันดังนี้

1. **การเขียนข้อความเรียงลำดับ** คือการบันทึกข้อความเรียงทีละข้อความตั้งแต่ข้อความที่ 1 จนถึงข้อความสุดท้าย วิธีการเขียนข้อความให้ทำตามลำดับขั้นตอนดังนี้
   - ป้อนสัญญาณพัลส์บวกที่ขา RES เพื่อรีเซตการทำงาน
   - ป้อนไฟลบ (ลอจิก 0) ให้ขา P/R เข้าสู่การบันทึกข้อความ
   - ป้อนสัญญาณพัลส์ลบที่ขา ST เพื่อเป็นสัญญาณเริ่มต้นการบันทึกข้อความที่ 1
   - พูดข้อความที่ 1 ลงไปจนจบ
   - ป้อนสัญญาณพัลส์ลบที่ขา ST เพื่อเป็นสัญญาณสิ้นสุดการบันทึกข้อความที่ 1
   - ป้อนสัญญาณพัลส์ลบที่ขา ST เพื่อเป็นสัญญาณเริ่มต้นการบันทึกข้อความที่ 2
   - พูดข้อความที่ 2 ลงไปจนจบ
   - ป้อนสัญญาณพัลส์ลบที่ขา ST เพื่อเป็นสัญญาณสิ้นสุดการบันทึกข้อความที่ 2

   เมื่อต้องการบันทึกข้อความที่ 3 ,4 ,5,...ก็ให้ทำซ้ำแบบเดิมต่อไปเรื่อยๆ
2. **การอ่านข้อความเรียงลำดับ** คือการเล่นข้อความเรียงลำดับตั้งแต่ข้อความที่ 1 จนถึงข้อความสุดท้าย วิธีการอ่านข้อความให้ทำตามลำดับขั้นตอนดังนี้
   - ป้อนสัญญาณพัลส์บวกที่ขา RES เพื่อให้รีเซตการทำงาน
   - ป้อนไฟบวก (ลอจิก 1) ให้ขา P/R เพื่อเข้าสู่การอ่านข้อความ
   - ป้อนสัญญาณพัลส์ลบที่ขา ST ข้อความที่ 1 ก็จะถูกเล่นออกมาจนจบ
   - ป้อนสัญญาณพัลส์ลบที่ขา ST ข้อความที่ 2 ก็จะถูกเล่นออกมาจนจบ

   เมื่อต้องการเล่นข้อความที่ 3 ,4 ,5,... ก็ให้ทำซ้ำแบบเดิมไปเรื่อยๆจนหมดทุกข้อความ ถ้าขณะที่ข้อความนั้นกำลังเล่นอยู่ยังไม่จบถ้ามีการป้อนสัญญาณพัลส์ลบที่ขา ST ข้อความที่กำลังเล่นอยู่ก็จะหยุดทันที เหมือนกับเป็นการ PAUSE และเมื่อป้อนสัญญาณพัลส์ลบที่ขา ST อีกครั้ง ข้อความนั้นก็จะถูกเล่นต่อจากเดิมไปจนจบ
3. **การอ่านข้อความไม่เรียงลำดับ** คือการเล่นข้อความแบบข้ามช่องไปมาหรือจะเล่นช่องไหนก่อนก็ได้ โดยที่ไม่ต้องเรียงลำดับตามช่องที่บันทึก วิธีการอ่านข้อความแบบไม่เรียงลำดับให้ทำตามลำดับขั้นตอนดังนี้
   - ป้อนสัญญาณพัลส์บวกที่ขา RES เพื่อให้รีเซตการทำงาน
   - ป้อนไฟบวก (ลอจิก 1) ที่ขา P/R เพื่อเข้าสู่การอ่านข้อความ
   - ป้อนไฟบวก (ลอจิก 1) ที่ขา SHIFT เพื่อเข้าสู่การเลื่อนข้อความ
   - ป้อนสัญญาณพัลส์ลบที่ขา ST 1 ครั้ง เพื่อสั่งเลื่อนข้อความไป 1 ช่อง
   - ป้อนไฟลบ (ลอจิก 0) ที่ขา SHIFT เพื่อเสร็จสิ้นการเลื่อนข้อความ
   - ป้อนสัญญาณพัลส์ลบที่ขา ST ข้อความที่ 2 ก็จะถูกเล่นออกมาจนจบ

จากที่กล่าวมาเป็นการอ่านข้อความที่ 2 ก่อน ถ้าต้องการกระโดดไปอ่านข้อความที่ 5 ถัดไป จะต้องทำดังนี้

- ป้อนไฟบวก (ลอจิก 1) ที่ขา SHIFT เพื่อเข้าสู่การเลื่อนข้อความ
- ป้อนสัญญาณพัลส์ลบที่ขา ST 2 ครั้ง เพื่อสั่งเลื่อนข้อความไป 2 ช่อง
- ป้อนไฟลบ (ลอจิก 0) ที่ขา SHIFT เพื่อเสร็จสิ้นการเลื่อนข้อความ
- ป้อนสัญญาณพัลส์ลบที่ขา ST ข้อความที่ 5 ก็จะถูกเล่นออกมาจนจบ

หลักการอ่านข้อความไม่เรียงลำดับ วิธีสั่งเลื่อนข้อความจะขึ้นกับการป้อนสัญญาณพัลส์ลบที่ขา ST หลังจากที่ขา SHIFT ได้รับลอจิก 1 แล้ว สมมติว่าต้องการอ่านข้อความที่ 4 ก่อน ก็จะต้องป้อนสัญญาณพัลส์ลบที่ขา ST 3 ครั้ง โดยครั้งที่ 1 จะเลื่อนไปหาข้อความที่ 2 ครั้งที่ 2 จะเลื่อนไปหาข้อความที่ 3 และครั้งที่ 3 จะเลื่อนไปหาข้อความที่ 4 นั่นเอง ถ้าต้องการเลื่อนไปหาข้อความที่ 8 หลังจากอ่านข้อความที่ 4 แล้วก็จะต้องป้อนสัญญาณพัลส์ลบที่ขา ST อีก 3 ครั้ง หากต้องการกลับไปอ่านข้อความที่ 6 หลังจากอ่านข้อความที่ 8 แล้วจะต้องไปเริ่มต้นรีเซตใหม่ แล้วเลื่อนไปยังข้อความที่ 6 เนื่องจากการเลื่อนข้อความไม่สามารถถอยหลังได้

<table>
<tr><th colspan="2">การเขียนเรียงลำดับ</th><th colspan="2">การอ่านเรียงลำดับ</th><th colspan="2">การอ่านไม่เรียงลำดับ</th></tr>
<tr><th>ขาอินพุท</th><th>สภาวะ</th><th>ขาอินพุท</th><th>สภาวะ</th><th>ขาอินพุท</th><th>สภาวะ</th></tr>
<tr><td>ขา RES</td><td>⎍</td><td>ขา RES</td><td>⎍</td><td>ขา RES</td><td>⎍</td></tr>
<tr><td>ขา P/R</td><td>“ 0 ”</td><td>ขา P/R</td><td>“ 1 ”</td><td>ขา P/R</td><td>“ 1 ”</td></tr>
<tr><td>ขา ST
ใส่ข้อความที่ 1
ขา ST</td><td>

⎍</td><td>ขา ST

ฟังข้อความที่ 1</td><td>⎍</td><td>ขา SHIFT
ขา ST
ขา SHIFT
ขา ST
ฟังข้อความที่ 2</td><td>“ 1 ”
⎍
“ 0 ”
⎍</td></tr>
<tr><td>ขา ST
ใส่ข้อความที่ 2
ขา ST</td><td>⎍

⎍</td><td>ขา ST

ฟังข้อความที่ 2</td><td>⎍</td><td>ขา SHIFT
ขา ST
ขา ST
ขา SHIFT
ขา ST
ฟังข้อความที่ 5</td><td>“ 1 ”
⎍
⎍
“ 0 ”
⎍</td></tr>
<tr><td>ขา ST
ใส่ข้อความที่ 3
ขา ST</td><td>⎍

⎍</td><td>ขา ST

ฟังข้อความที่ 3</td><td>⎍</td><td></td><td></td></tr>
</table>

ตารางที่ 2.2 วิธีควบคุมการทำงานของ ISD25xx ในโหมดที่สอง

<u>รายละเอียดการใช้งานขาต่างๆของ ISD2560</u>

- Microphone Input (MIC)

ขา 17 จะรับสัญญาณอินพุทที่ผ่านเข้ามายังไมโครโฟน แล้วส่งผ่านสัญญาณเข้าสู่วงจรปรีแอมป์ที่ประกอบอยู่ภายในตัวไอซี ภายในประกอบด้วยวงจรควบคุมอัตราการขยายอัตโนมัติ (AGC) โดยวงจรนี้จะทำหน้าที่ควบคุมอัตราการขยายวงจรปรีแอมป์ให้มีการขยายในช่วง –15 ถึง 24 เดซิเบล ไมโครโฟนจากภายนอกจะถูกคับปลิ้งผ่านตัวเก็บประจุในลักษณะอนุกรมกับขา 17 ค่าความจุของตัวเก็บประจุจะกำหนดโดยคำนึงถึงค่าความต้านทานภายนอกในช่วงไอซี ($10k\Omega$) เพื่อทำให้เกิดการคัทออฟที่ความถี่ต่ำ

- Microphone Reference Input (MIC REF)

ขา18 นี้จะต่อกับกราวด์อนาลอก (Vssa) โดยต่ออนุกรมกับตัวเก็บประจุเพื่อทำหน้าที่กำจัดสัญญาณรบกวนทางอินพุทขา 17 และเพื่อให้การชดเชยทางด้านสัญญาณรบกวนให้ดีกว่า 10 เดซิเบล

- Analog Output (ANA OUT)

ขา 20 จะรับสัญญาณผ่านวงจรปรีแอมป์ออกมาทางขา 21 โดยผ่านตัวเก็บประจุคับปลิ้งภายนอก คับปลิ้งสัญญาณที่ขา 20 นี้เพื่อผ่านสัญญาณเข้าไปบันทึกไว้ภายในตัวไอซี ตัวเก็บประจุคับปลิ้งภายนอกนี้จะต้องสัมพันธ์กับค่าความต้านทานใน 3 $k\Omega$ ซึ่งเป็นอินพุทอิมพีแดนซ์ เพื่อจะทำให้เป็นวงจรกรองความถี่แบบคัทออฟ

- Automatic Gain Control Input (AGC)

ขา 19 เป็นอินพุทเพื่อควบคุมอัตราการขยายของปรีแอมป์ไมโครโฟนทางด้านไดนามิก เพื่อให้เกิดความเหมาะสมกับระดับสัญญาณที่มีย่านความถี่กว้างมากของสัญญาณทางด้านอินพุทจากไมโครโฟน และเพื่อให้ระดับสัญญาณที่ทำการบันทึกมีความผิดเพี้ยนน้อยที่สุด ขา AGC นี้จะต่อรวมกับอุปกรณ์ RC เพื่อกำหนดค่าเวลาคงที่ โดยมีค่าความต้านทานภายใน $5k\Omega$ และจะต่อกับตัวเก็บประจุภายนอกอีกตัวหนึ่งเพื่อผ่านลงกราวด์อนาลอก ค่าที่เหมาะสมบางครั้งกำหนดไว้ที่ $R = 470\Omega$, $C = 4.7\mu F$

- Speaker Outputs (SP+, SP-)

ขา 14 , 15 เป็นขาเอาท์พุทต่อออกมาลำโพง โดยในไอซีจะมีวงจรสัญญาณความแตกต่างออกสู่ลำโพง ซึ่งความสามารถในการขับลำโพงเอาท์พุทได้ 50 mW ที่โหลด 16Ω ขาเอาท์พุทนี้ไม่สามารถต่อขนานกันได้หลายตัว ในกรณีที่ต่อคาสเคดกันหลายตัว

- Power Down Input (PD)

ขา 24 ในขณะที่ไม่มีการบันทึกหรือเล่นกลับ ที่ขา PD จะมีสภาวะเป็น "1" ก็จะเป็นการรักษาระดับการสิ้นเปลืองกำลังงานในระดับต่ำมากๆ แต่เมื่อขา $\overline{OVF}$ มีสภาวะเป็น "0" ที่แสดงถึงการเล่นกลับสิ้นสุดลงปรากฏขึ้น ขา PD ปกติจะเป็น "1" อยู่ในขณะนี้ ก็จะถูกรีเซตและจะเริ่มขบวนการบันทึกหรือเล่นกลับใหม่อีกครั้ง

- Chip Enable Input ($\overline{CE}$)

ขา 23 ขา $\overline{CE}$ จะต้องได้รับสัญญาณพัลส์ "0" เพื่อเกิดการเปลี่ยนแปลงระหว่างการเล่นกลับและการบันทึกที่ขาแอดเดรสอินพุท และขา $P / \overline{R}$ จะถูกแลตซ์จากพัลส์ขอบขาลงของพัลส์ที่ขา $\overline{CE}$

- Playback/Record Input ($P / \overline{R}$)

ขา 27 เมื่อขาอินพุทควบคุมการเล่นกลับและบันทึกได้รับพัลส์ "1" จะเป็นวงรอบของการเล่นกลับและถ้าเป็นพัลส์ "0" จะเป็นการเลือกวงรอบการบันทึก ถ้าหากได้รับพัลส์ที่ขอบขาลงของขา $\overline{CE}$ จะเป็นการแลตซ์อินพุทที่ขา $P / \overline{R}$

- Address/Mode Input (A0-A9/M0-M6)

ขา 1-10 ขาแอดเดรสและโหมดอินพุทจะมีอยู่สองฟังค์ชั่นที่ขึ้นอยู่กับระดับของ MSB สองบิทของแอดเดรส ถ้ามีบิทใดบิทหนึ่งของสองบิท MSB เป็น "0" อินพุทก็จะมาปรากฏที่แอดเดรสบิททั้งหมดและใช้เป็นแอดเดรสเริ่มต้นสำหรับวงรอบการบันทึกและการเล่นกลับ และขาแอดเดรสจะเกิดการแลตซ์โดยขอบขาลงของพัลส์ที่ขา $\overline{CE}$ และถ้า MSB ทั้งสองบิทมีสถานะเป็น "1" ขาแอดเดรสของโหมดอินพุทจะมาขึ้นอยู่ที่โหมดบิททั้งหมดและเกิดการแลตซ์เมื่อพัลส์ขอบขาลงปรากฏที่ขา $\overline{CE}$

ขา 22 สัญญาณพัลส์ "0" จะปรากฏออกทางขาเอาท์พุทนี้เพื่อเป็นการแสดงว่าสิ้นสุดการเล่นกลับหรือหน่วยความจำภายในไอซีถูกอ่านออกมาจนหมดแล้ว และจะแสดงเป็นสภาวะเล่นกลับ พัลส์จากขานี้จะจ่ายให้กับขาอินพุทจนกว่าจะได้รับพัลส์เพื่อทำการรีเซท และเริ่มวงรอบการเล่นกลับใหม่อีกครั้ง พัลส์ที่ขานี้จะสามารถใช้เริ่มต้นการทำงานของ ISD2560 ในตัวถัดไปได้เมื่อมีการต่อค่าคาสเคดกันอยู่หลายตัว

- Auxiliary Input (AUX IN)

ขา 11 จะเป็นขารับอินพุทจากภายนอกเพื่อทำการมัลติเพล็กซ์สัญญาณผ่านออกไปทางเอาท์พุทลำโพง โดยขั้นตอนการทำงานนี้จะเกิดขึ้นเมื่อขามีสถานะเป็น "1" วงรอบของการเล่นกลับก็จะสิ้นสุดลง หรือเมื่อสัญญาณที่บันทึกไว้ถูกเล่นกลับจนหมดแล้ว การต่อคาสเคด ISD2560 กันหลายๆขา AUX IN จะถูกใช้เมื่อต่อเข้ากับสัญญาณเล่นกลับที่ออกมาจากขาเอาท์พุทลำโพงของตัวก่อนหน้านี้ หรือจากตัวอันดับแรก

- Voltage Input (V_{cca} , V_{ccd})

ขา 16 และ 28 เป็นขารับแรงดันที่จะต้องแยกกันระหว่างขารับแรงดันของวงจรอนาลอกและวงจรดิจิตอลที่ปรากฏอยู่ในตัวไอซีแล้ว ขารับแรงดันต้องการแรงดันไฟเลี้ยง +5 โวลต์ และต้องเป็นแรงดันไฟเลี้ยงที่มีสัญญาณรบกวนต่ำมาก

- Ground Input (V_{ssa} , V_{ssd})

ขา 12 และ 13 โดยคุณสมบัติของไอซีในตระกูล ISD25XX จะมีการแยกกันระหว่างกราวด์ของสัญญาณอนาลอก และกราวด์ของสัญญาณดิจิตอล ขากราวด์ทั้งสองข้างนี้จะถูกต่อและปิดไว้ในตัวถังบรรจุของไอซี การใช้งานของกราวด์ทั้งสองจะเลือกต่อกับกราวด์ของเพาว์เวอร์ซัพพลายในส่วนที่มีค่าอิมพีแดนซ์เพื่อไม่ต้องการให้เกิดค่าแรงดันที่แตกต่างกันระหว่างกราวด์ทั้งสอง

บทที่ 3
การออกแบบและการสร้าง

3.1 การออกแบบระบบ

จากการสำรวจร้านอาหารขนาดกลางถึงขนาดใหญ่ในกรุงเทพฯ พบว่าแต่ละร้านมีลักษณะการทำงานที่ใกล้เคียงกัน จึงได้ทำการออกแบบระบบร้านอาหารอัตโนมัติให้มีความเหมาะสมกับร้านทั่วไป โดยคำนึงถึง ความสะดวกรวดเร็วในการให้บริการ ประหยัดค่าใช้จ่ายในการจ้างพนักงาน และมีความทันสมัย โดยได้แบ่งระบบการทำงานออกเป็นส่วน ๆ ตามหน้าที่การทำงาน ดังในรูปที่ 3.1 ดังนี้

ส่วนควบคุม (Controlling)

มีลักษณะเหมือนสถานีแม่ ทำหน้าที่ในการจัดการและควบคุมการติดต่อสื่อสารกันภายในระบบ โดยจะส่งสัญญาณโพลไปถามยังเครื่องสั่งอาหารว่ามีข้อมูลการสั่งอาหารจะส่งหรือไม่ ถ้ามีก็จะรับข้อมูลมาประมวลผลแล้วส่งต่อไปยังคอมพิวเตอร์ส่วนกลาง รอการตอบรับของคอมพิวเตอร์ แล้วส่งผลการสั่งอาหารกลับไปยังเครื่องสั่งอาหารนั้น ๆ

ส่วนสั่งอาหาร (Ordering)

มีหน้าที่หลักคือรับข้อมูลรายการอาหารผ่านทางคีย์แพด แสดงผลทางจอแอลซีดี แล้วส่งข้อมูลไปยังส่วนควบคุมแบบไร้สาย ประกอบด้วยวงจรเครื่องส่ง วงจรเครื่องรับ วงจรควบคุมการรับข้อมูลและส่วนแสดงผล

ส่วนครัว (Cooking)

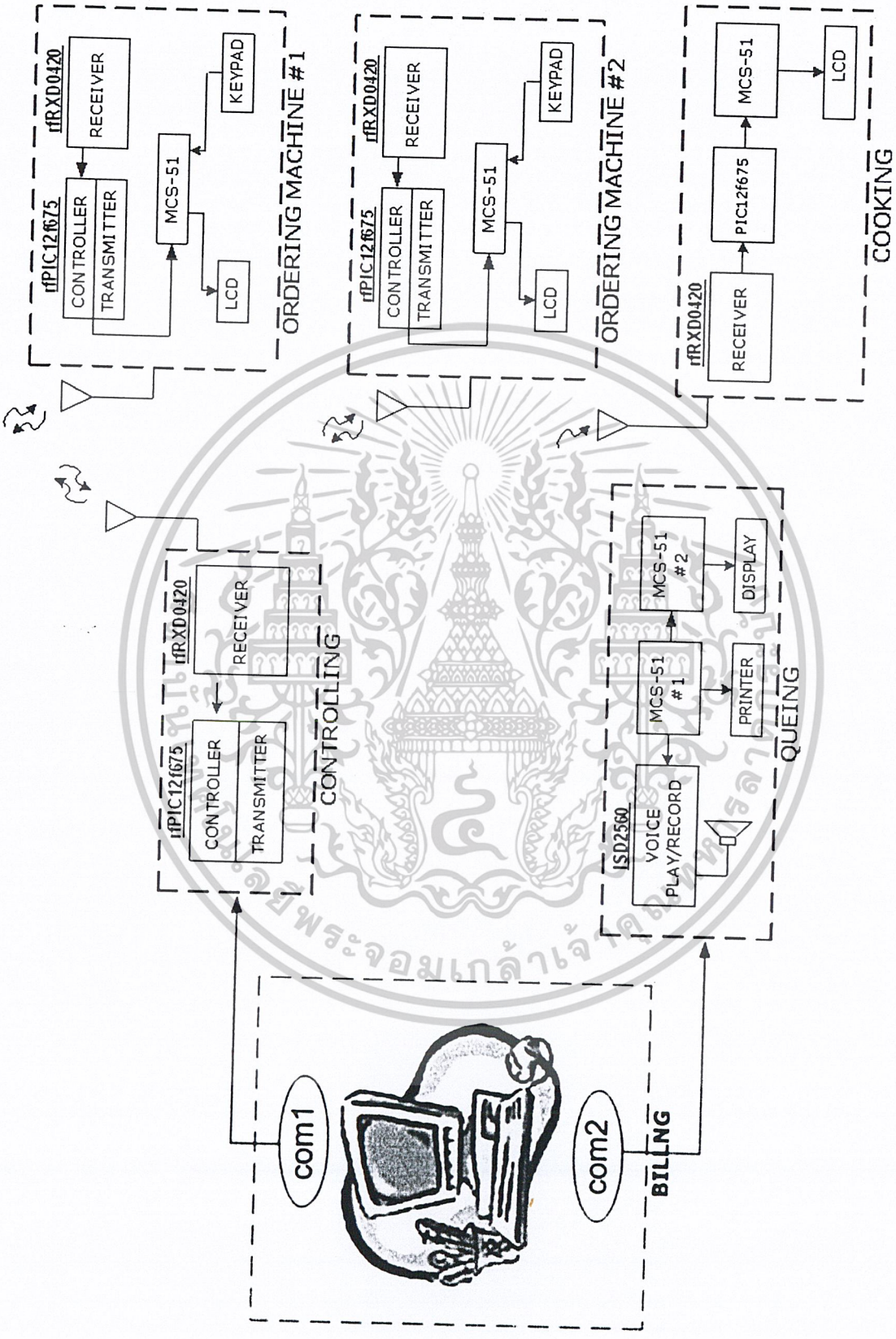
ส่วนนี้จะทำหน้าที่แสดงรายการอาหารที่ลูกค้าสั่งทางจอแอลซีดีที่อยู่ในครัว เพื่อให้พ่อครัวได้ทำตามลำดับ โดยจะตรวจจับข้อมูลการสั่งอาหาร เมื่อการสั่งอาหารเสร็จสมบูรณ์ก็จะแสดงผลทางจอแอลซีดีทีละรายการ เมื่อพ่อครัวกดปุ่ม

ส่วนการจัดคิว (Queuing)

ส่วนนี้ทำหน้าที่ในการจัดลำดับการเข้าใช้บริการของลูกค้า โดยจะใช้เครื่องพริ้นเตอร์ในการออกบัตรคิวให้ลูกค้า และแจ้งให้ทราบเมื่อถึงคิวด้วยเสียงและจอแสดงผล มีส่วนประกอบคือวงจรบันทึกเล่น/เสียง วงจรควบคุมพริ้นเตอร์และจอแสดงผล และวงจรอินเตอร์เฟสกับพอร์ทอนุกรมของเครื่องคอมพิวเตอร์

ส่วนแสดงผลและคิดเงิน (Billing)

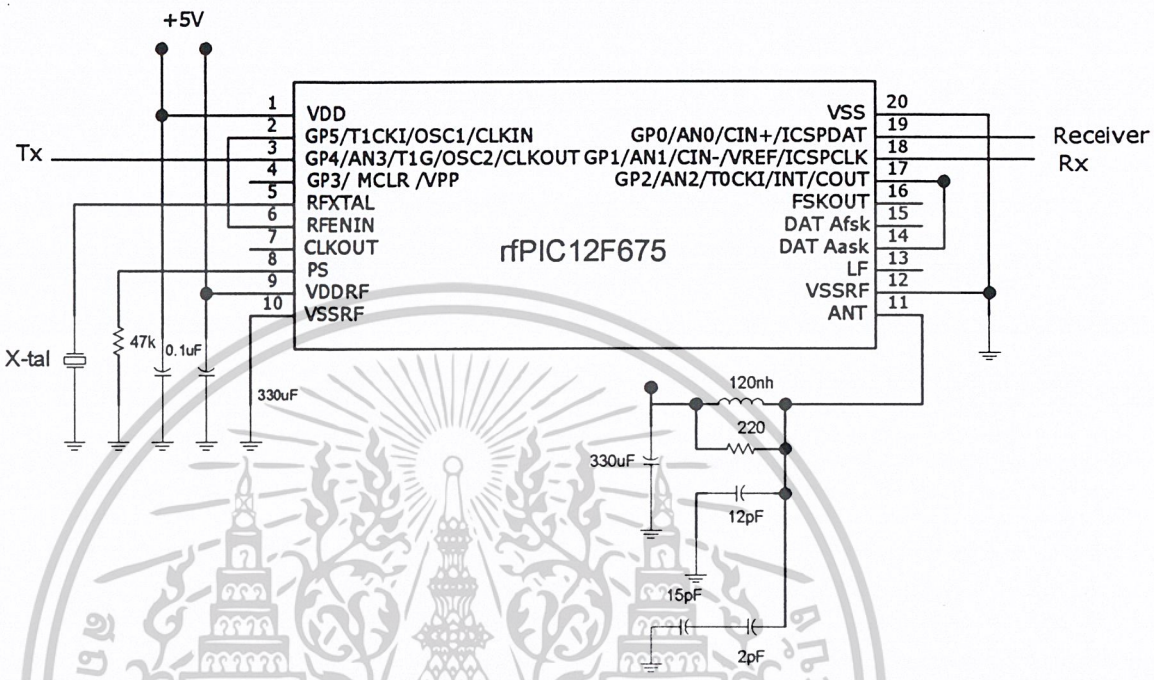
ส่วนนี้จะใช้เครื่องคอมพิวเตอร์ในการแสดงข้อมูลรายการอาหารที่ผู้ใช้บริการแต่ละโต๊ะสั่ง และทำการคิดเงินเมื่อผู้ใช้บริการทานอาหารเสร็จ



รูปที่ 3.1 แสดงภาพรวมของระบบร้านอาหารอัตโนมัติ

3.2 การออกแบบวงจรต่าง ๆ

3.2.1 วงจรเครื่องส่ง



รูปที่ 3.2 วงจรเครื่องส่ง

ประกอบด้วยไอซี rfPIC12f675 ซึ่งเป็นไอซีที่มีเครื่องส่งและไมโครคอนโทรลเลอร์อยู่ภายในตัว มีคุณสมบัติดังที่กล่าวไปแล้วในหัวข้อที่ 2.9 โดยในระบบนี้จะใช้งานเครื่องส่งที่ความถี่ 433.92 MHz มอดูเลตแบบแอมปลิจูดชิฟคีย์อิง (ASK) ค่าตัวต้านทานและตัวเก็บประจุจะใช้ตามเดต้าชีทซึ่งมีให้เลือก 2 ความถี่ คือ 315 MHz และ 433.92 MHz ซึ่งการกำหนดการใช้งานในแต่ละขาเป็นดังนี้

GP0 - ใช้รับข้อมูลจากเครื่องรับ

GP1 - ใช้รับข้อมูลจากขา Tx ของพอร์ทอนุกรม 1 หรือ MCS-51

GP2 - ใช้ส่งข้อมูลไปยังขา Data ASK

GP3 - ไม่ใช้

GP4 - ใช้ส่งข้อมูลไปยังขา Rx ของพอร์ทอนุกรม 1 หรือ MCS-51

GP5 - ใช้ส่งสัญญาณควบคุมการใช้งานเครื่องส่งไปยังขา RFEN

การคำนวณความถี่คริสตัลสามารถคำนวณได้จากสมการ ดังนี้

ความถี่คลื่นพาห์ = ความถี่คริสตัล X 32

ในโครงงานนี้จะใช้ความถี่คลื่นพาห์ 433.92 MHz

433.92 MHz = ความถี่คริสตัล X 32

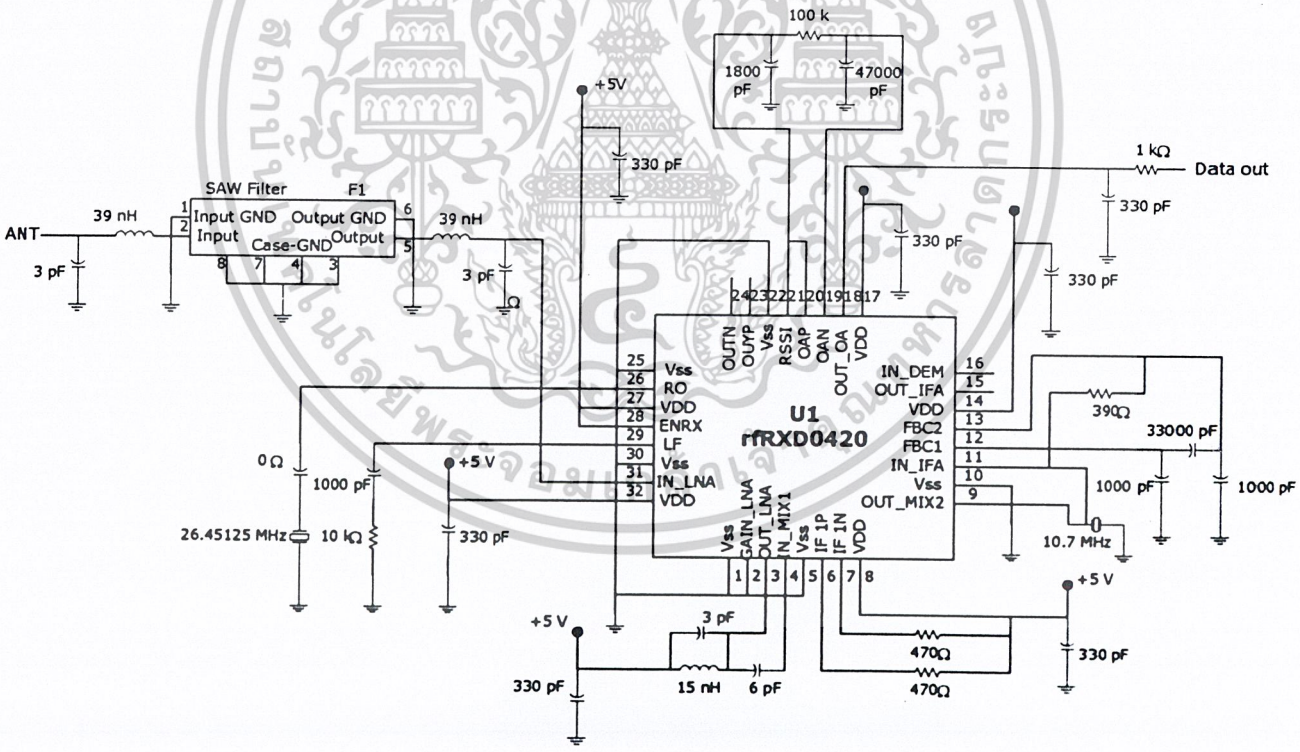
ความถี่คริสตัล = 433.92 MHz / 32

= 13.56 MHz

ดังนั้นจะใช้คริสตัลความถี่ 13.56 MHz

ส่วนสายอากาศที่ใช้จะเป็นแบบลูปแอนเทนนาตามคำแนะนำในเดต้าชีท

3.2.2 วงจรเครื่องรับ



รูปที่ 3.3 วงจรเครื่องรับ

จะใช้ไอซี rfRXD0420 ซึ่งออกแบบมาเพื่อเป็นเครื่องรับคู่กับเครื่องส่งของไอซี rfPIC12f675 ทำงานเป็นเครื่องรับที่ความถี่ 433.92 MHz และดีมอดูเลตแบบแอมปลิจูดชิฟคีย์อิง (ASK) เช่นเดียวกับส่วนควบคุม ต่อวงจรตามแบบเดต้าชีท จะได้สัญญาณที่ผ่านการดีมอดูเลตแล้วออกมาทางขา Data out

สายอากาศที่ใช้ในวงจรนี้เป็นสายอากาศแบบไดโพล สามารถทำขึ้นเองได้จากสายไฟธรรมดา โดยได้คำนวณความยาวของสายอากาศดังนี้

สายอากาศ = ความยาวคลื่น / 4

ในโครงงานนี้จะใช้ความถี่คลื่นพาห์ 433.92 MHz

$$\text{ความยาวสายอากาศ} = \left(\frac{3x10^{8}}{433.92x10^{6}}\right) / 4$$

$$= 0.1728 \text{ m}$$

ดังนั้นจะใช้สายอากาศความยาว 17.28 เซนติเมตร

<u>การหาระยะทางการส่งโดยใช้สูตรฟรีส (Friis formular)</u>

$$P_r(W) = P_t(W)\frac{G_tG_r\lambda^2}{(4\pi r)^2}$$

$$P_r(dBm) = P_t(dBm) + G_t(dBi) + G_r(dBi) + 20\log(\lambda) - 20\log(4\pi d)$$

$$P_r(dBm) = 6 + 0 + 0 + 20\log\left(\frac{3x10^{8}}{433.92x10^{6}}\right) - 20\log(4\pi d)$$

$$-109 = 6 + (-3.2058) - 20\log(4\pi d)$$

$$20\log(4\pi d) = 111.7942$$

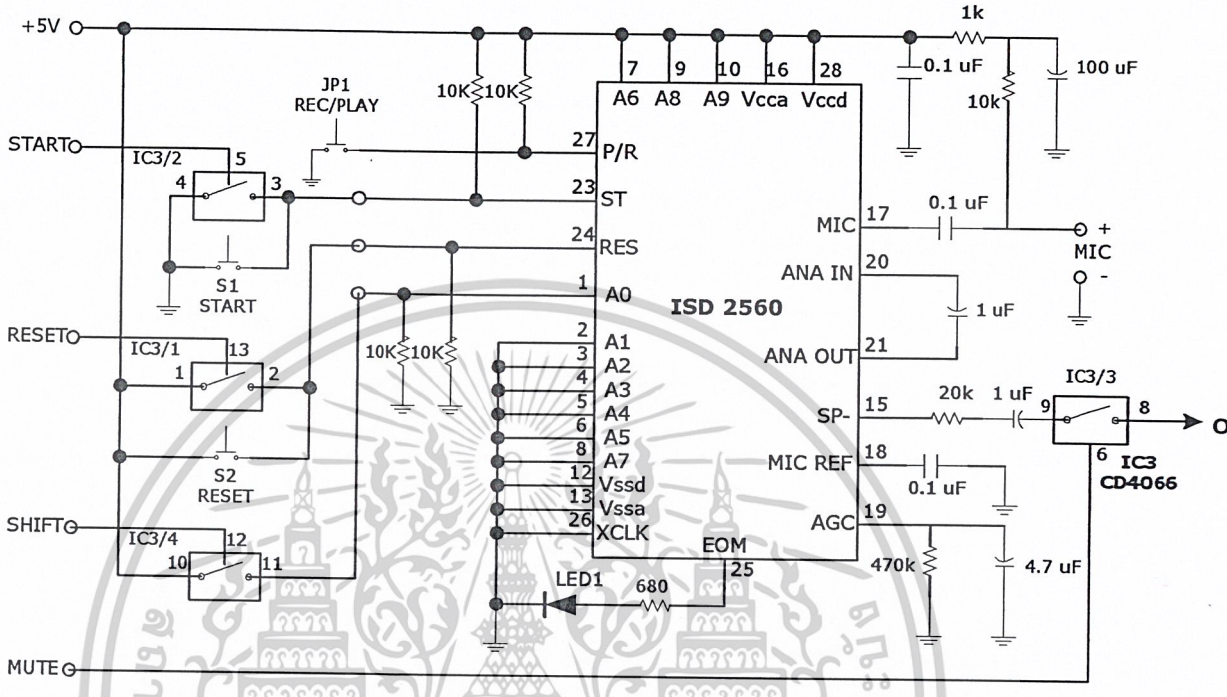
$$\log(4\pi d) = 5.58971$$

$$4\pi d = 10^{5.58971}$$

$$d = 30.954km$$

$$(P_r = -109 \ \ dBm, P_t = 6 \ \ dBm, G_t = G_r = 0 \ \ dBi, f = 433.92MHz)$$

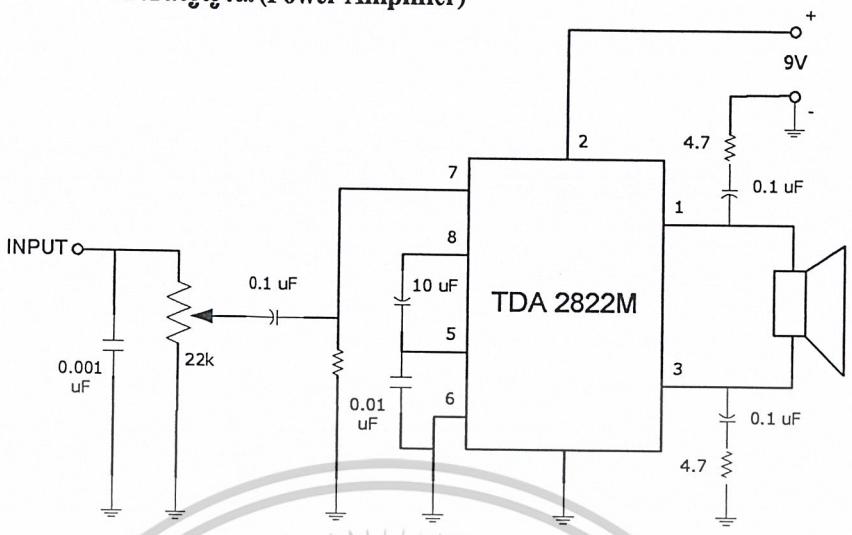
3.2.3 วงจรเล่น/บันทึกเสียง



รูปที่ 3.4 วงจรเล่น/บันทึกเสียง

ใช้ไอซี ISD2560 ทำงานในโหมดที่ 2 ซึ่งเป็นแบบบันทึกเล่นเสียงหลายๆข้อความดังที่กล่าวไปแล้วในหัวข้อ 2.12 ใช้ขาคอนโทรล 3 ขา คือ START, RESET และ SHIFT จะทำการควบคุมโดยเขียนโปรแกรมผ่านทาง ไมโครคอนโทรลเลอร์ MCS-51 และใช้ไอซี CD4066 ทำหน้าที่เป็นสวิชต์เชื่อมต่อระหว่าง MCS-51 กับขาคอนโทรลทั้ง 3 ขา

3.2.4 วงจรขยายสัญญาณ (Power Amplifier)

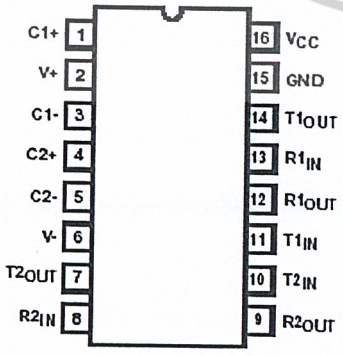


รูปที่ 3.5 วงจรขยายสัญญาณ

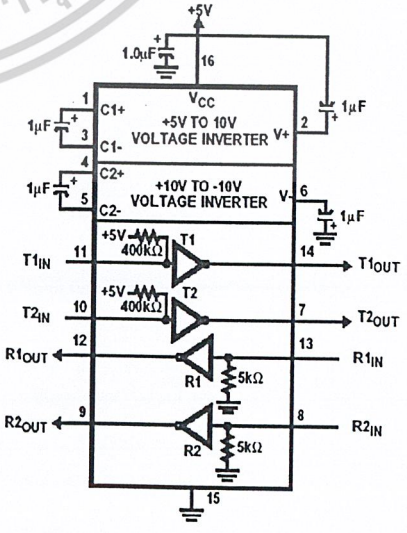
จะใช้ไอซีเบอร์ TDA2822M เป็นวงจรขยายสัญญาณย่านความถี่เสียง ในการขยายสัญญาณเอาท์พุทของวงจรเล่น/บันทึกเสียง ให้เอาท์พุทเป็นเสียงออกทางลำโพง โดยจะต่อค่าความต้านทานปรับค่าได้ที่อินพุท เพื่อปรับขนาดของสัญญาณอินพุท ให้ได้เสียงที่ดังเบาตามต้องการ การจัดวงจรจะเป็นไปตามเดต้าชีทในภาคผนวก

3.2.5 วงจรเชื่อมต่อกับพอร์ทอนุกรมของคอมพิวเตอร์

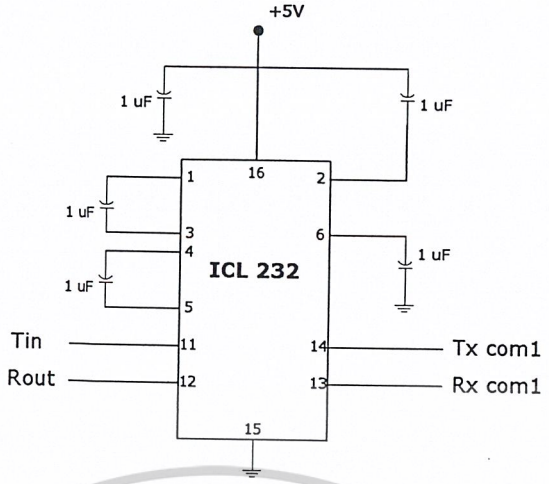
จะใช้ไอซีเบอร์ ICL232 หรือ Max 232 มาทำหน้าที่แปลงระดับสัญญาณระหว่างสัญญาณระดับทีทีแอล ของไมโครคอนโทรลเลอร์ rfPIC12f675 และ MCS-51 กับระดับสัญญาณของพอร์ทอนุกรม RS-232 เพื่อใช้เป็นส่วนอินเตอร์เฟสระหว่างไมโครคอนโทรลเลอร์ ของเครื่องคอมพิวเตอร์ ดังรูป 3.6



รูปที่ 3.6ก ไอซี ICL232



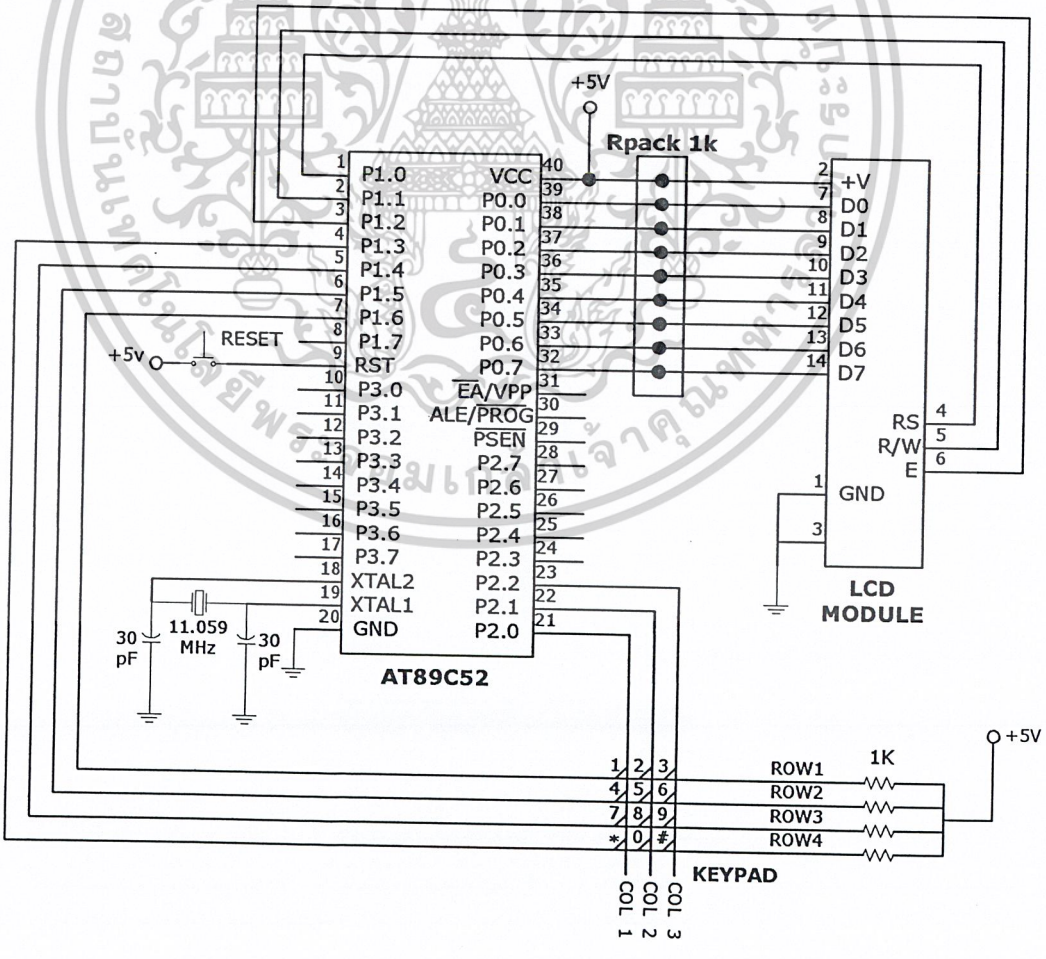
รูปที่ 3.6ข โครงสร้างภายในไอซี ICL232



รูปที่ 3.6ค การจัดวงจรเชื่อมต่อพอร์ทอนุกรม

รูปที่ 3.6 วงจรเชื่อมต่อกับพอร์ทอนุกรมของเครื่องคอมพิวเตอร์

3.2.6 วงจรรับข้อมูลจากคีย์แพดและแสดงผลทางจอแอลซีดี



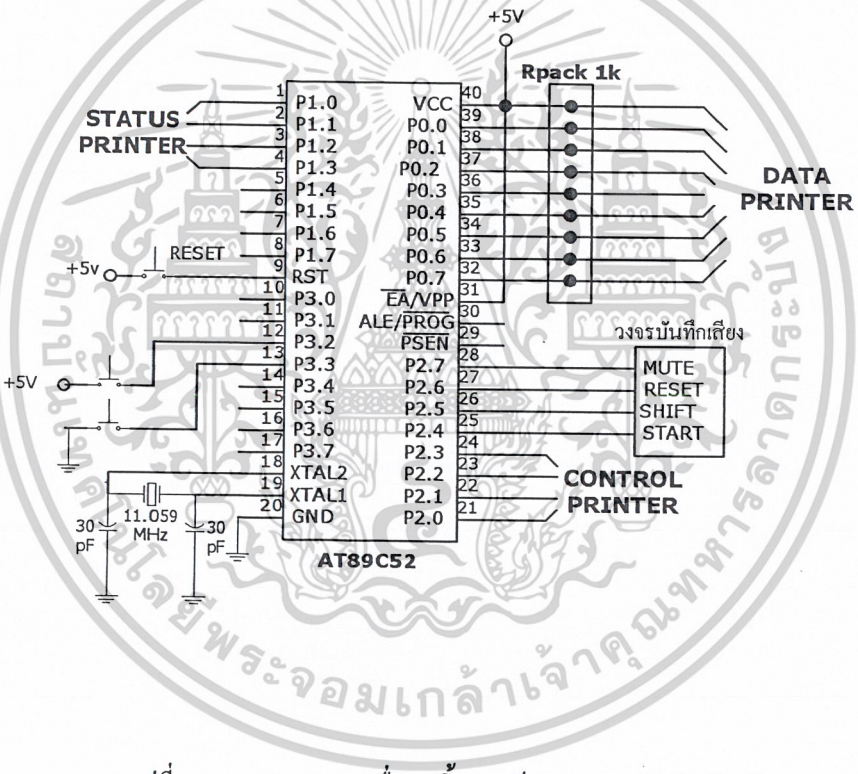
รูปที่ 3.7 วงจรรับข้อมูลจากคีย์แพดและแสดงผลทางจอแอลซีดี

วงจรนี้จะใช้ไมโครคอนโทรลเลอร์ MCS-51 ในการควบคุมการรับข้อมูลจากคีย์แพดและแสดงผลทางจอแอลซีดี

สำหรับแอลซีดีจะเลือกใช้พอร์ท 0 ซึ่งต้องต่อ Rpack เพราะไม่มีความต้านทานพูลอัพภายใน ในการส่งสัญญาณข้อมูล 8 บิท แบบขนานไปยังขา D0 - D7 ของแอลซีดี และใช้ พอร์ท P1.0 - P1.2 ในการส่งสัญญาณควบคุมไปยังจอแอลซีดี

ส่วนคีย์แพดนั้น ในโครงงานนี้ได้เลือกใช้แบบ 3 x 4 ดังนั้นจึงต้องการขาในการจ่ายข้อมูล 7 ขา เพื่อให้ง่ายต่อการเขียนโปรแกรมจะเลือกใช้ขา P1.3 – P1.6 ส่งข้อมูลไปยังคีย์แพด และเลือกใช้ขา P2.0 – P2.2 เป็นขารับข้อมูล ดังวิธีการที่อธิบายไปแล้วในหัวข้อ 2.3

3.2.7 วงจรควบคุมเครื่องพริ้นเตอร์และวงจรเล่นบันทึกเสียง



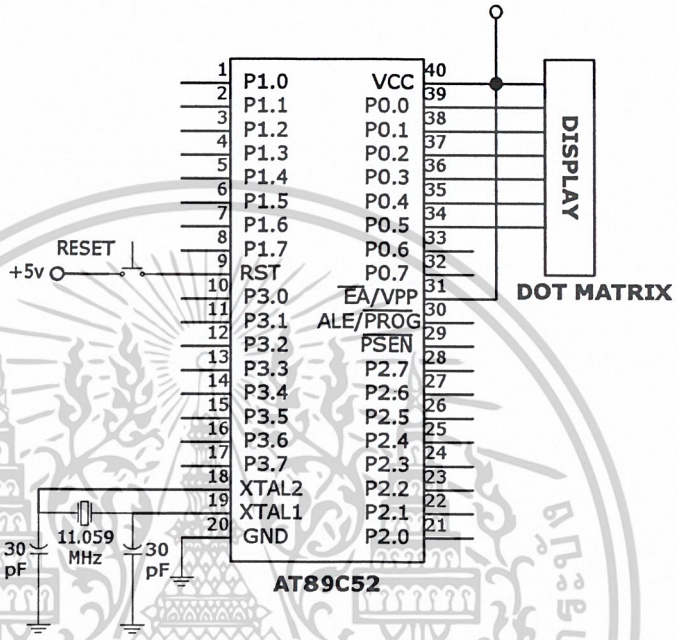
รูปที่ 3.8 วงจรควบคุมเครื่องพริ้นเตอร์และวงจรเล่นบันทึกเสียง

ในส่วนของพริ้นเตอร์ต้องใช้ขาในการรับส่งข้อมูลทั้งสิ้น 16 ขา ดังนี้

P0.0 – P0.7	ใช้ส่งข้อมูล 8 บิตให้กับเครื่องพริ้นเตอร์
P1.0 – P1.3	ใช้รับข้อมูลเกี่ยวกับสถานะ (Status) ของเครื่องพริ้นเตอร์
P2.0 – P2.3	ใช้ส่งสัญญาณควบคุม (Control) ไปยังเครื่องพริ้นเตอร์
P2.4 – P2.7	ในการควบคุมวงจรเล่น /บันทึกเสียง

3.2.8 วงจรควบคุมจอแสดงผลแบบแอลอีดีดอทเมทริกซ์

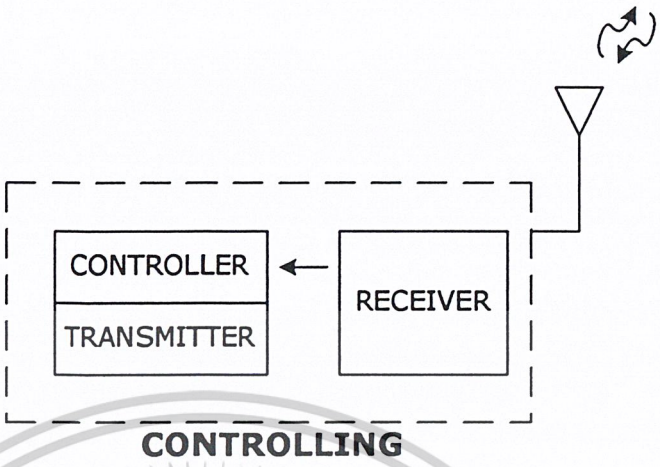
จอแสดงผลแบบแอลอีดีดอทเมทริกซ์มีทั้งหมด 11 ขา โดยจะเป็นขาที่ใช้ในการส่งข้อมูลเพื่อการแสดงผล 6 ขา จะเลือกใช้พอร์ทใช้พอร์ท P0.0 – P0.5



รูปที่ 3.9 วงจรควบคุมจอแสดงผลแบบแอลอีดีดอทเมทริกซ์

3.3 โครงสร้างและการทำงานของส่วนต่าง ๆ

3.3.1 ส่วนควบคุม



รูปที่ 3.10 โครงสร้างของส่วนควบคุม

ส่วนนี้จะต้องทำหน้าที่ในการควบคุมการสื่อสารภายในระบบ โดยจะติดต่อกับส่วนสั่งอาหารและส่วนครัวแบบไร้สาย และติดต่อกับเครื่องคอมพิวเตอร์ผ่านทางพอร์ทอนุกรม 1(COM1) ประกอบด้วย 3 วงจรดังนี้

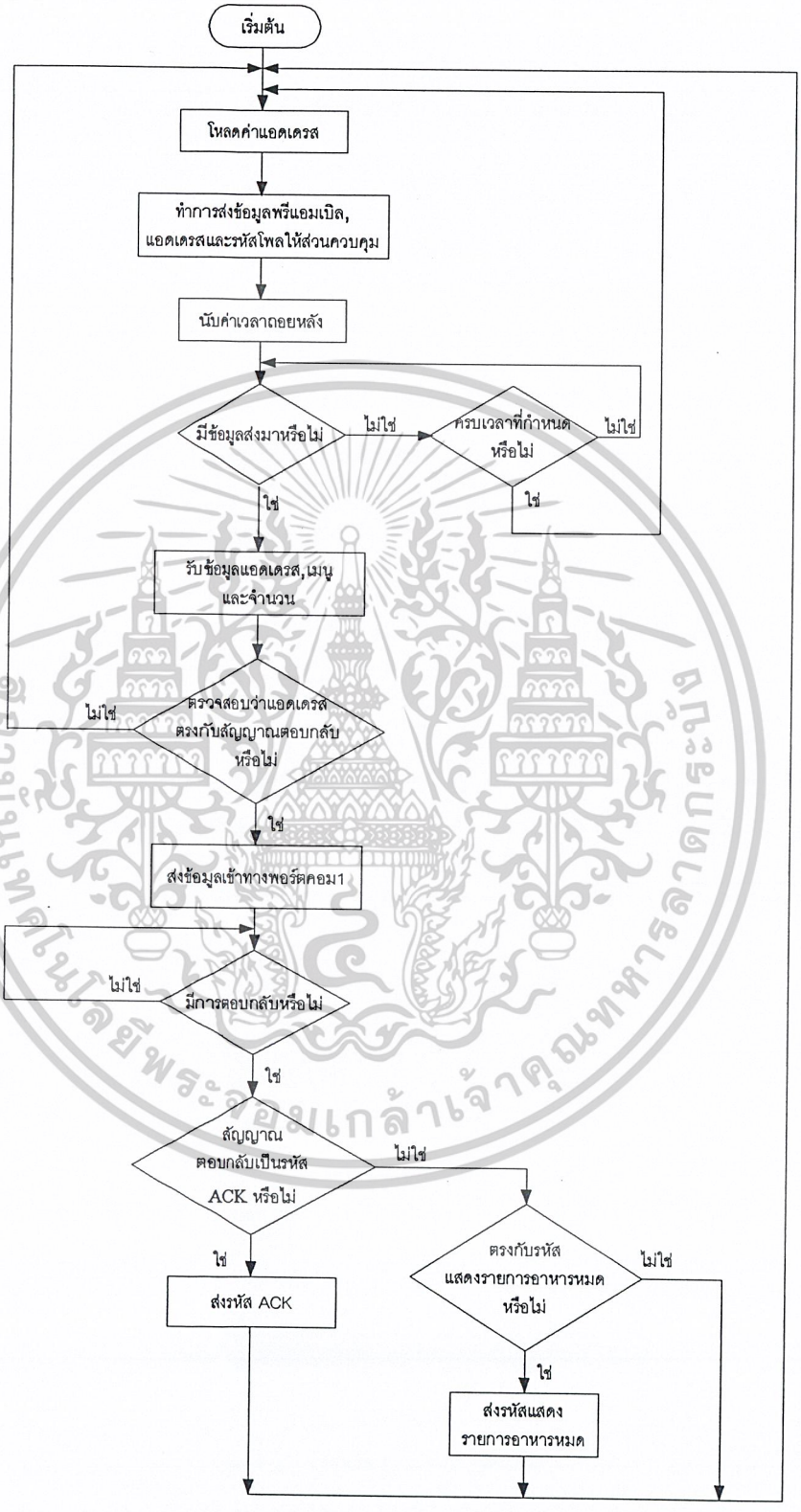
1. วงจรเครื่องส่ง
2. วงจรแครื่องรับ
3. วงจรเชื่อมต่อกับพอร์ทอนุกรม

<u>หลักการทำงาน</u>

เริ่มจากเครื่องส่ง ซึ่งมีไมโครคอนโทรลเลอร์ในตัว จะส่งสัญญาณโพลออกไป เพื่อถามเครื่องสั่งอาหารของแต่ละโต๊ะ ว่ามีข้อมูลการสั่งอาหารที่จะส่งมาหรือไม่ ถ้าหากไม่มีก็จะทำการโพลไปยังโต๊ะถัดไปเรียงตามลำดับ แต่ถ้ามีการสั่งอาหารก็จะรับข้อมูลรายการอาหารมาประมวลผล แล้วส่งต่อให้กับพอร์ทอนุกรมของคอมพิวเตอร์ โดยผ่านวงจรเชื่อมต่อกับพอร์ทอนุกรม เพื่อรอการประมวลผลของคอมพิวเตอร์ แล้วจึงส่งสัญญาณตอบรับ (ACK) ให้กับเครื่องสั่งอาหาร จากนั้นจะส่งสัญญาณโพลของโต๊ะถัดไป

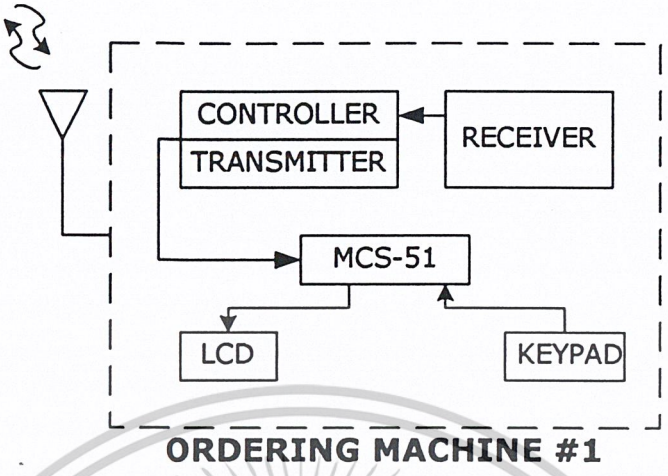
ส่วนเครื่องรับนั้นจะทำหน้าที่รับข้อมูลแบบไร้สาย เพียงอย่างเดียวแล้วส่งให้กับเครื่องส่งที่พอร์ท GP0 โดยไม่มีการประมวลผล

ผังการทำงานของส่วนควบคุม



รูปที่ 3.11 ผังการทำงานของส่วนควบคุม

3.3.2 ส่วนสั่งอาหาร



รูปที่ 3.12 โครงสร้างของส่วนสั่งอาหาร

ส่วนนี้จะทำการรับข้อมูลรายการอาหารที่ลูกค้าสั่งผ่านทางคีย์แพด โดยมีแอลซีดีเป็นส่วนแสดงผล แล้วส่งข้อมูลไปยังส่วนควบคุมแบบไร้สาย ประกอบด้วย 3 วงจร ดังนี้

1. วงจรเครื่องส่ง
2. วงจรเครื่องรับ
3. วงจรรับข้อมูลจากคีย์แพดและแสดงผลทางจอแอลซีดี

<u>หลักการทำงาน</u>

เริ่มจากรอรับข้อมูลรายการอาหาร ที่ลูกค้าสั่งผ่านทางคีย์แพด โดยลูกค้าจะต้องป้อนข้อมูล ประกอบด้วยรหัสรายการอาหาร 3 หลัก และจำนวนที่ต้องการ ซึ่งมีการแสดงผลทางจอแอลซีดี เมื่อได้รับข้อมูลรายการอาหารแล้ว ก็จะส่งข้อมูลจากไปให้ ไมโครคอนโทรลเลอร์ของเครื่องส่ง แบบอนุกรม เครื่องส่งจะทำการประมวลผล แล้วจัดเฟรมส่งไปยังส่วนควบคุมแบบแบบไร้สาย

จากนั้นจะรอสัญญาณตอบรับจากส่วนควบคุม เมื่อได้รับก็จะทำการประมวลผลโดย ไมโครคอนโทรลเลอร์ของเครื่องส่ง แล้วส่งต่อไปยัง MCS-51 เพื่อแสดงผลทางหน้าจอแอลซีดีให้ลูกค้าทราบ

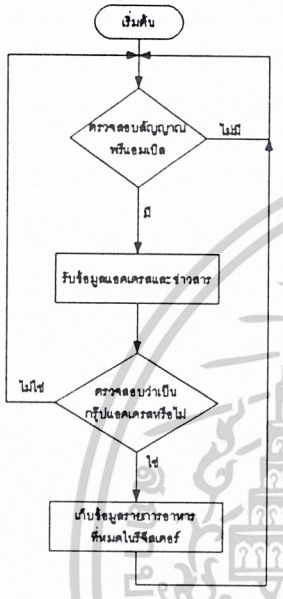
สำหรับวิธีการใช้งานเครื่องสั่งอาหารนี้โดยละเอียด สามารถดูได้ในผลการทดลอง หัวข้อที่ 4.2.2

<u>ผังการทำงานของส่วนสั่งอาหาร</u>

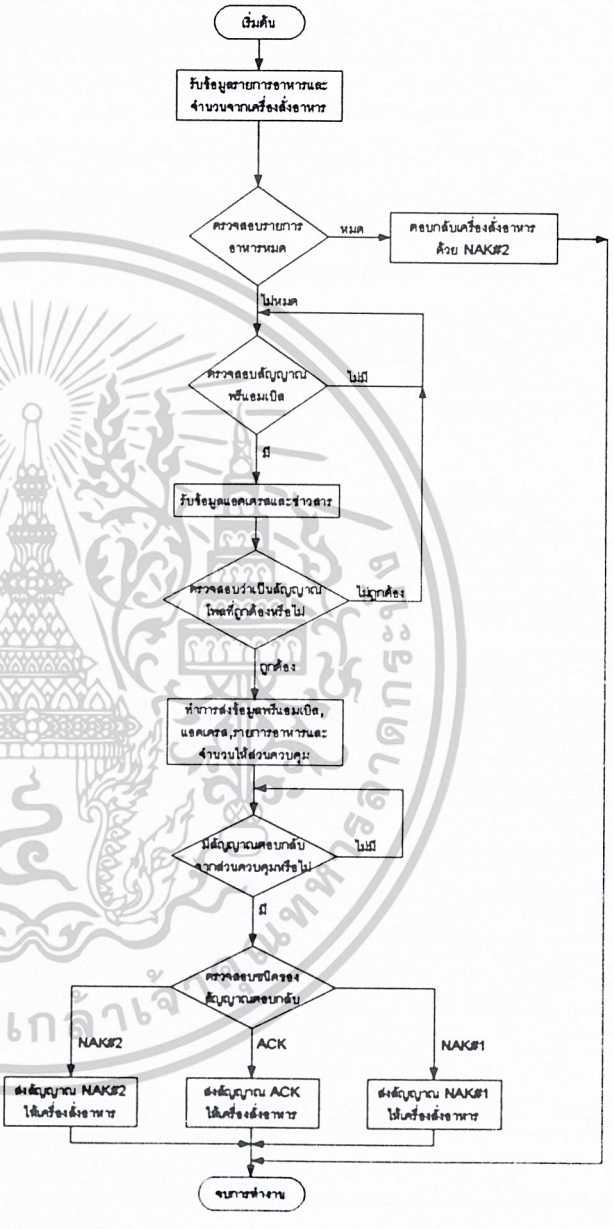
ประกอบด้วย 2 ส่วนดังนี้

1. ไอซี rfPIC12f675 - โปรแกรมหลักทำหน้าที่วนลูปรอรับรายการอาหารหมด
   - โปรแกรมอินเตอร์รัปต์ทำหน้าที่รับข้อมูลการสั่งอาหารจาก MCS-51

- โปรแกรมหลัก

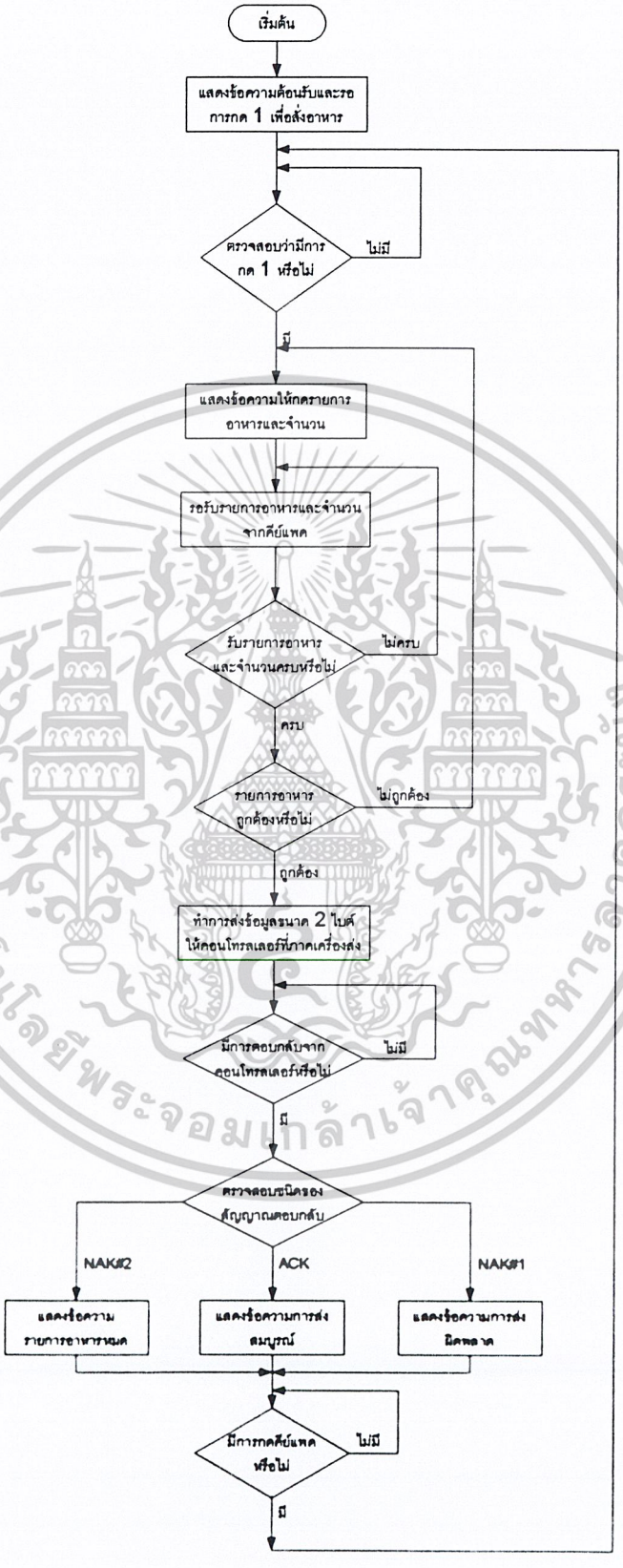


- โปรแกรมอินเตอร์รัปต์



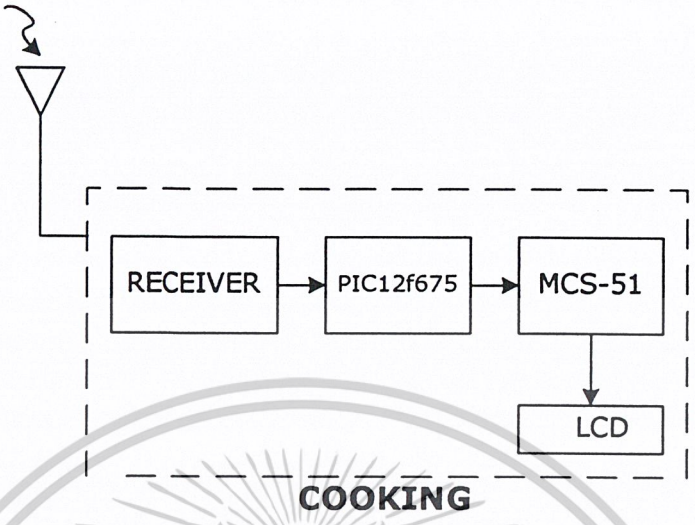
รูปที่ 3.13 ผังการทำงานของส่วนสั่งอาหาร 1

2. MCS-51 - ทำหน้าที่รอรับการสั่งอาหารจากคีย์แพดและควบคุมการแสดงผลทางจอแอลซีดี



รูปที่ 3.14 ผังการทำงานของส่วนสั่งอาหาร 2

3.3.3 ส่วนครัว



รูปที่ 3.15 โครงสร้างของส่วนครัว

ส่วนนี้จะต้องทำการตรวจจับสัญญาณในอากาศตลอดเวลา เพื่อดูว่ามีการสั่งอาหารอะไรบ้างและการสั่งนั้นสมบูรณ์หรือไม่ แล้วแสดงผลทางจอแอลซีดีของส่วนครัว ประกอบด้วยวงจรดังนี้

1. วงจรเครื่องรับ
2. วงจร ไมโครคอนโทรลเลอร์ PIC12f675
3. วงจรควบคุมการแสดงผลทางจอแอลซีดี

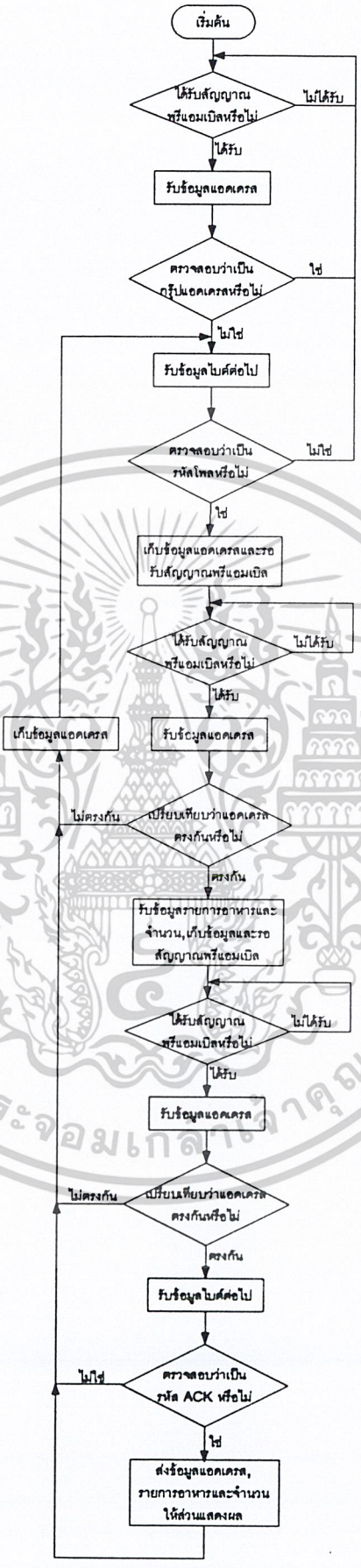
<u>หลักการทำงาน</u>

ในส่วนนี้จะใช้ไมโครคอนโทรลเลอร์ PIC12f675 ในการประมวลผลสัญญาณที่เครื่องรับตรวจจับได้ตลอดเวลา เพื่อตรวจสอบว่ามีโต๊ะใดสั่งอาหารอะไรบ้าง จำนวนเท่าไหร่ และเมื่อการสั่งนั้นเสร็จสมบูรณ์คือมีการตอบรับจากส่วนควบคุม ก็จะส่งข้อมูลรายการอาหารนั้นไปยัง MCS-51 เพื่อเก็บข้อมูลรายการอาหารไว้ และจะแสดงรายการอาหารให้พ่อครัวดูทางจอแอลซีดีทีละรายการ เมื่อพ่อครัวกดปุ่ม โดยการกดปุ่มนี้จะใช้ฟังก์ชันในการอินเตอร์รัปต์ของ MCS-51

<u>ผังการทำงานของส่วนครัว</u>

ประกอบด้วย 2 ส่วน ดังนี้

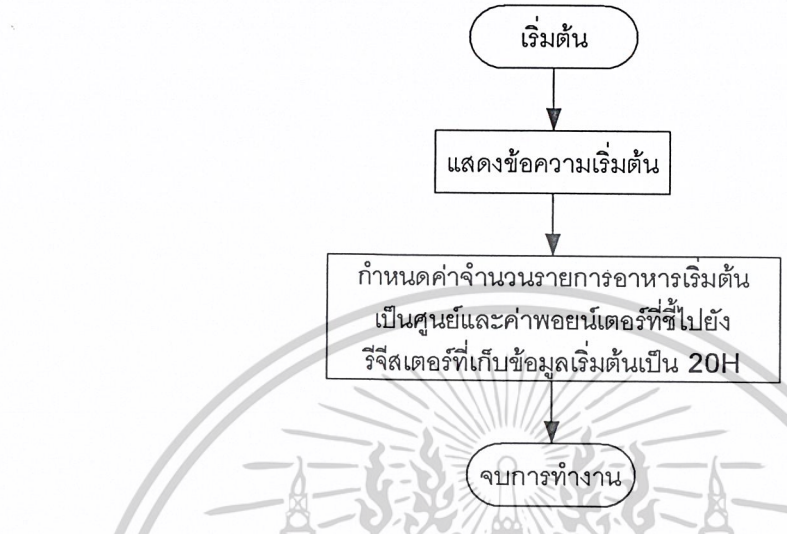
1. PIC12f675 - ทำหน้าที่จับสัญญาณการสั่งอาหาร แล้วส่งข้อมูลให้ MCS-51



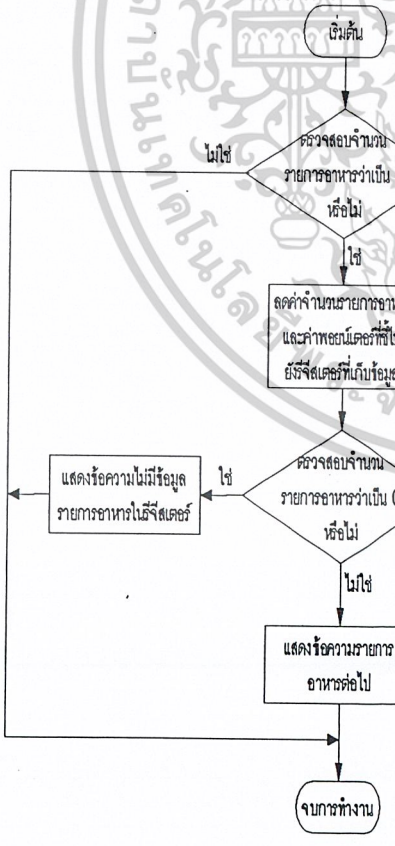
รูปที่ 3.16 ผังการทำงานของส่วนครัว 1

2. MCS-51 - ทำหน้าที่เป็นบัฟเฟอร์เก็บข้อมูลการแสดงอาหารและแสดงผลทางจอแอลซีดี

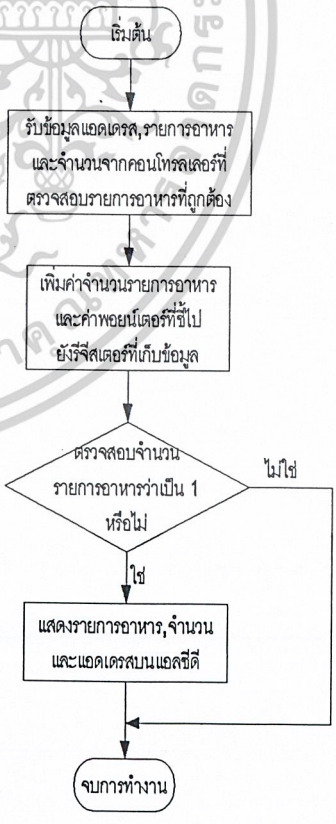
- โปรแกรมหลัก



- โปรแกรมอินเตอร์รัปต์ 1

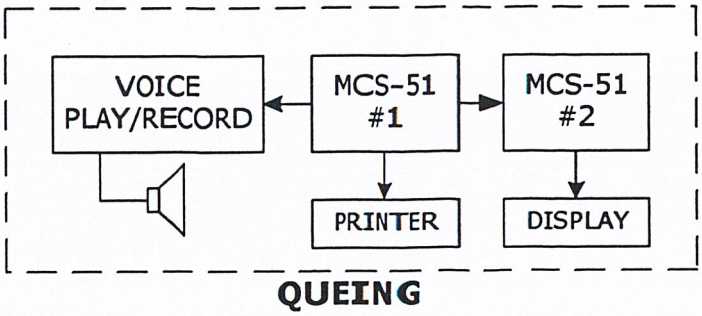


- โปรแกรมอินเตอร์รัปต์ 2



รูปที่ 3.17 ผังการทำงานของส่วนครัว 2

3.3.4 ส่วนจัดคิว



รูปที่ 3.18 โครงสร้างของส่วนจัดคิว

ส่วนนี้จะทำหน้าที่ในการจัดคิว ออกบัตรคิวและแสดงผลทางจอแสดงผลคอทเมทริกซ์ ประกอบด้วยวงจรต่างๆดังนี้

1. วงจรเล่น/บันทึกเสียง
2. วงจรขยาย
3. วงจรควบคุมเครื่องพริ้นเตอร์และวงจรเล่น/บันทึกเสียง
4. วงจรควบคุมจอแสดงผลแบบแอลอีดีคอทเมทริกซ์

<u>หลักการทำงาน</u>

จะใช้วงจรเล่น/บันทึกเสียง ในการเรียกลูกค้าตามคิวที่จัดไว้ พร้อมทั้งแจ้งหมายเลขโต๊ะให้ทราบด้วยสัญญาณเสียง โดยได้บันทึกข้อความต่อไปนี้ไว้ในไอซี ISD2560 ซึ่งสามารถเรียกใช้ได้โดย MCS-51 ตัวที่ 1

ข้อความที่ 0 “ศูนย์”
ข้อความที่ 1 “หนึ่ง”
ข้อความที่ 2 “สอง”
ข้อความที่ 3 “สาม”
ข้อความที่ 4 “สี่”
ข้อความที่ 5 “ห้า”
ข้อความที่ 6 “หก”
ข้อความที่ 7 “เจ็ด”
ข้อความที่ 8 “แปด”
ข้อความที่ 9 “เก้า”
ข้อความที่ 10 “เชิญหมายเลข”
ข้อความที่ 11 “ที่โต๊ะ”
ข้อความที่ 12 “ค่ะ”

และใช้วงจรขยายเพื่อเพิ่มความดังของเสียง สามารถปรับความดังเบาของเสียงได้ โดยใช้ค่าความต้านทานปรับค่าได้ ซึ่งต่ออยู่กับสัญญาณอินพุทที่ได้รับมาจากวงจรเล่น/บันทึกเสียง

ส่วนการออกบัตรคิวนั้นจะใช้ไมโครคอนโทรลเลอร์ MCS-51 ตัวเดียวกับที่ใช้ควบคุมวงจรเล่น/บันทึกเสียง ในการรับข้อมูลการกดปุ่มคิวผ่านทางขาอินเตอร์รัปต์ แล้วทำการสั่งงานให้เครื่องพริ้นเตอร์พิมพ์บัตรคิวเป็นหมายเลขออกมา สำหรับวิธีการสั่งงานพริ้นเตอร์สามารถดูได้จากหัวข้อที่ 2.6

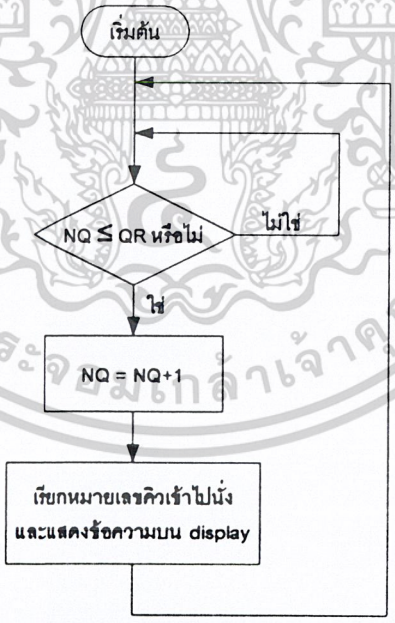
นอกจากนี้ยังมีการแสดงผลโดยจอแสดงผลแอลอีดีดอทเมทริกซ์ เพื่อแสดงผลให้ลูกค้าทราบว่าขณะนั้นมีการเรียกถึงคิวที่เท่าไหร่ และจะบอกหมายเลขโต๊ะที่ลูกค้าจะเข้าใช้บริการด้วย การควบคุมการแสดงผลจะใช้ไมโครคอนโทรลเลอร์ MCS-51 ตัวที่2

<u>ผังการทำงานของส่วนจัดคิว</u>

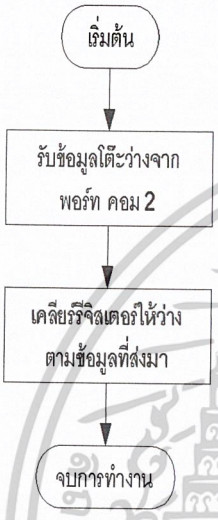
แบ่งเป็น 2 ส่วน คือ

1. MCS-51 ตัวที่ 1 -ทำหน้าที่ควบคุมวงจรเล่น/บันทึกเสียง และเครื่องพริ้นเตอร์

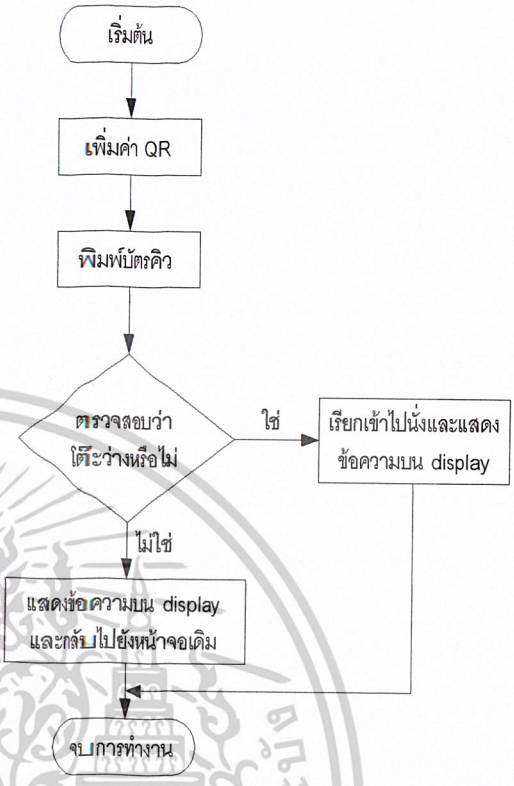
- โปรแกรมหลัก



- โปรแกรมอินเตอร์รัปต์ 1



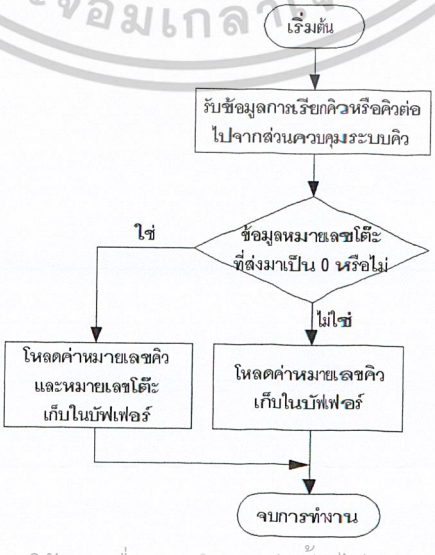
- โปรแกรมอินเตอร์รัปต์ 2



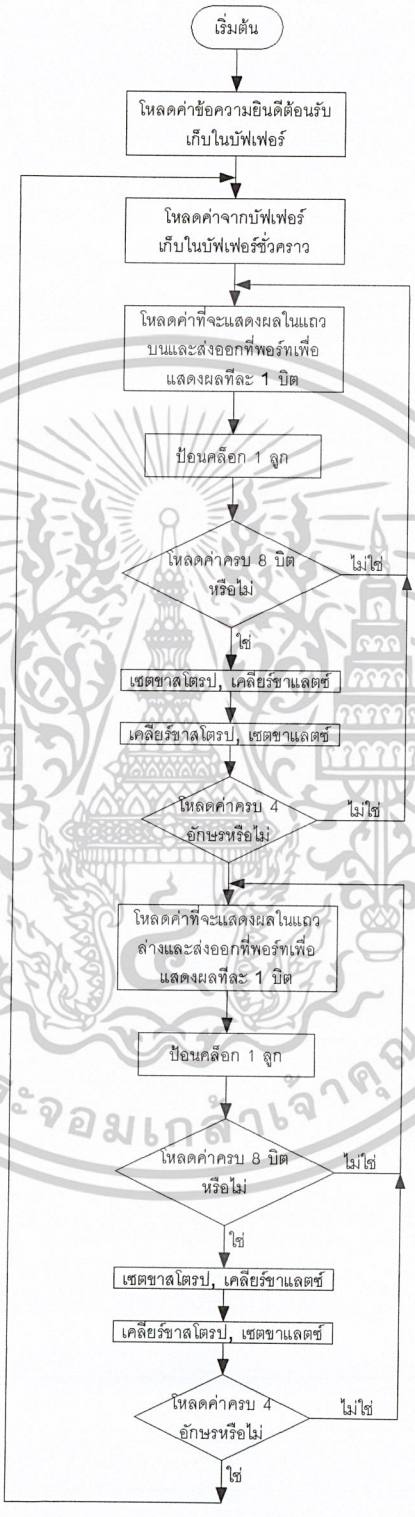
รูปที่ 3.19 ผังการทำงานของส่วนจัดคิว 1

2. MCS-51 ตัวที่ 2 -ทำหน้าที่ควบคุมการแสดงผลทางจอแสดงผลแบบแอลอีดีดอทแมทริกซ์

- โปรแกรมอินเตอร์รัปต์ 1



- โปรแกรมหลัก



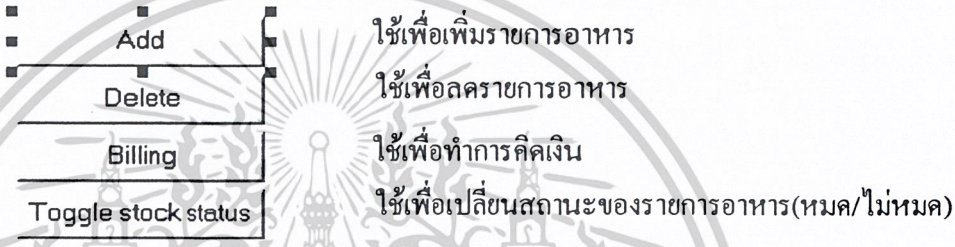
รูปที่ 3.20 ผังการทำงานของส่วนจัดคิว 2

3.3.5 ส่วนคิดเงิน

ส่วนนี้จะใช้เครื่องคอมพิวเตอร์ในการประมวลผลข้อมูลรายการอาหาร ที่ได้รับมาจากส่วนควบคุม แล้วแสดงผลรายการอาหาร จำนวน และราคา ออกทางหน้าจอ เมื่อลูกค้าใช้บริการเสร็จก็สามารถคิดเงินได้ทันที โดยโปรแกรมที่ใช้ ได้ทำการเขียนขึ้นด้วยภาษา C++ ซึ่งมีรายละเอียดและ วิธีใช้งานดังนี้

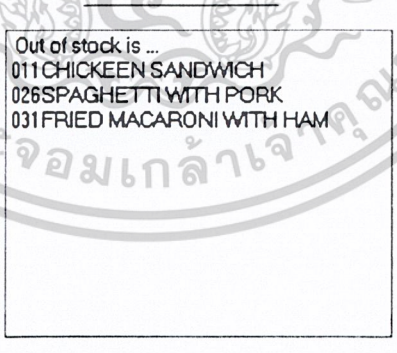
<u>รายละเอียดและวิธีการใช้งานโปรแกรมร้านอาหารอัตโนมัติ</u>

1. ฟังก์ชันการทำงานของปุ่มต่าง ๆ เป็นดังนี้

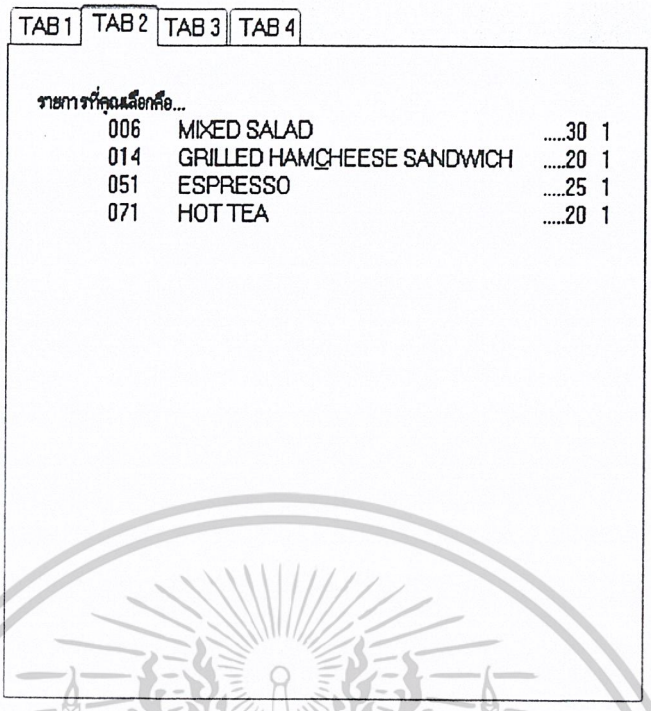


รูปที่ 3.21 ฟังก์ชันการทำงานของปุ่มต่าง ๆ

2. ข้อความที่แสดงผล ในหน้าจอต่าง ๆ

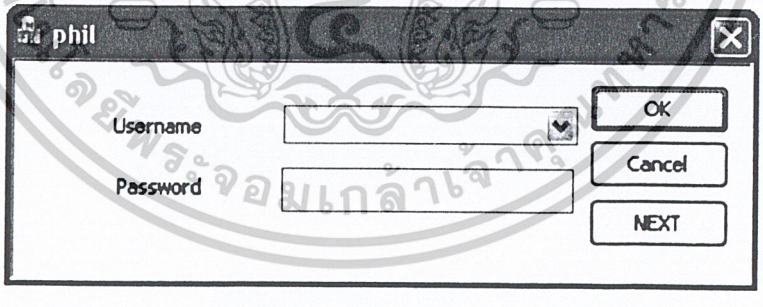


รูปที่ 3.22 หน้าต่างแสดงรายการอาหารหมด



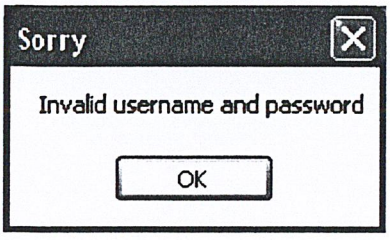
รูปที่ 3.23 หน้าต่างแสดงข้อมูลรายการอาหารของโต๊ะต่าง ๆ

3. เมื่อต้องการเข้าใช้งานโปรแกรมจะต้องมีการใส่ชื่อผู้ใช้งานและรหัสผ่านเพื่อเข้าใช้งานโดยจะแสดงหน้าต่างให้ทำการใส่ข้อมูลดังนี้



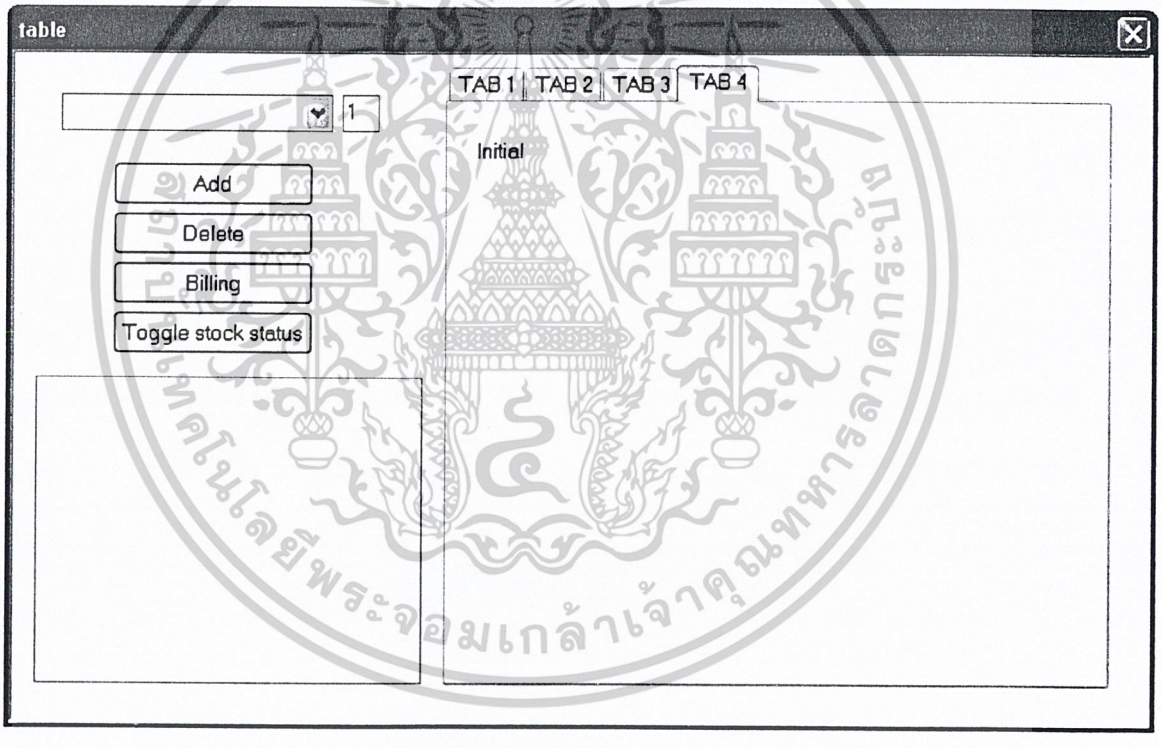
รูปที่ 3.24 หน้าต่างใส่ชื่อผู้ใช้และรหัสผ่าน

เมื่อทำการใส่ชื่อและรหัสผ่านไม่ตรงกับที่ตั้งไว้จะมีการแสดงข้อความบอกดังนี้



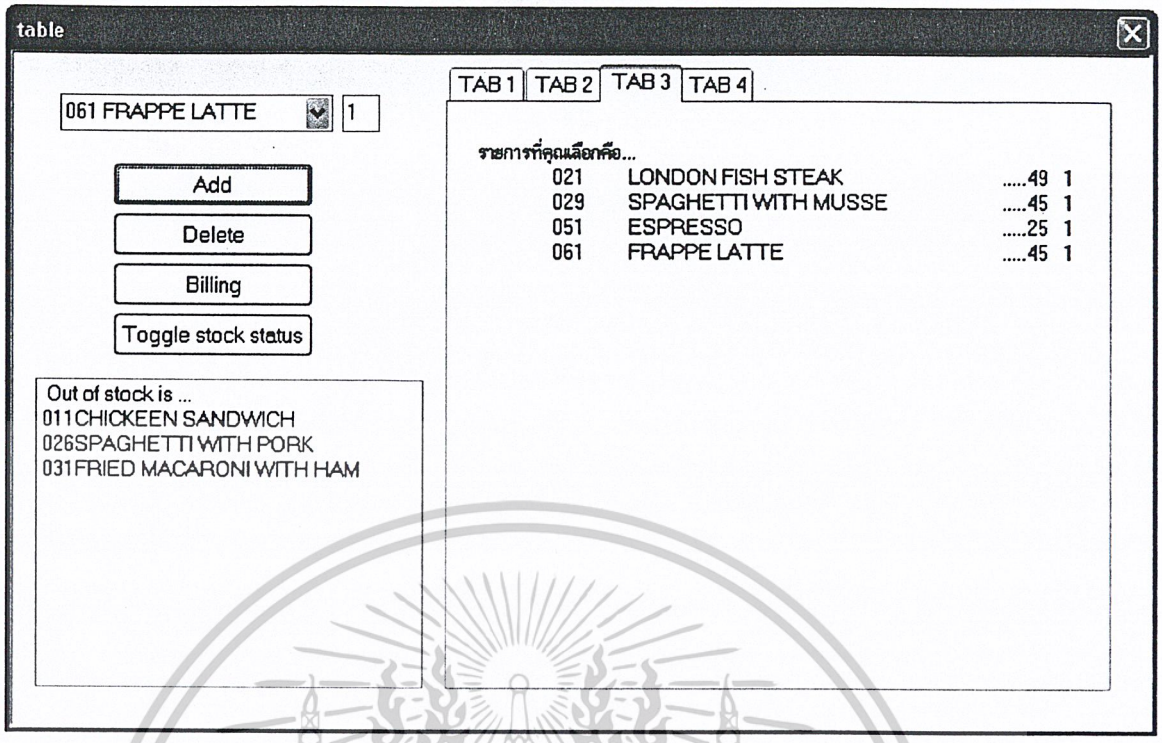
รูปที่ 3.25 หน้าต่างแสดงชื่อและรหัสผ่านผิด

เมื่อทำการใส่ชื่อและรหัสผ่านถูกต้องก็จะเข้าสู่หน้าต่างหลักของโปรแกรมร้านอาหารอัตโนมัติ ดังรูป

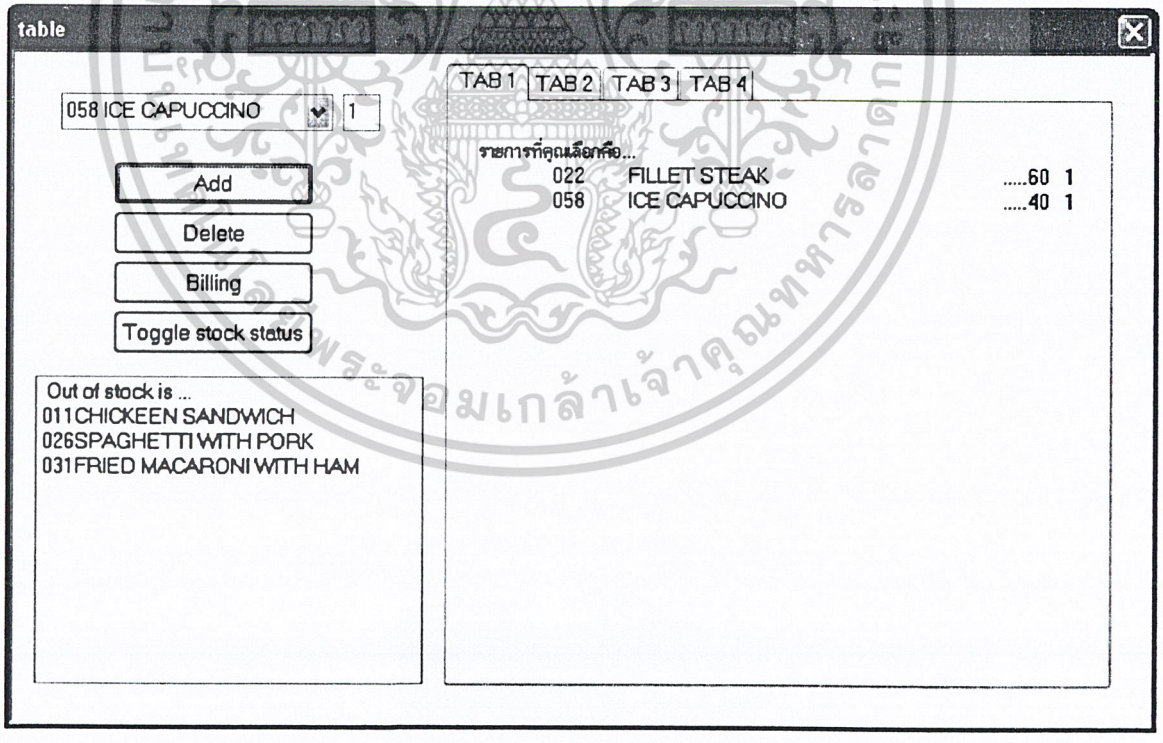


รูปที่ 3.26 หน้าต่างหลักของโปรแกรมร้านอาหารอัตโนมัติ

4. **เมื่อทำการสั่งรายการอาหารจากเครื่องสั่งอาหาร ส่วนควบคุมจะส่งรายการอาหารมาแสดงผลยังหน้าจอดังนี้**

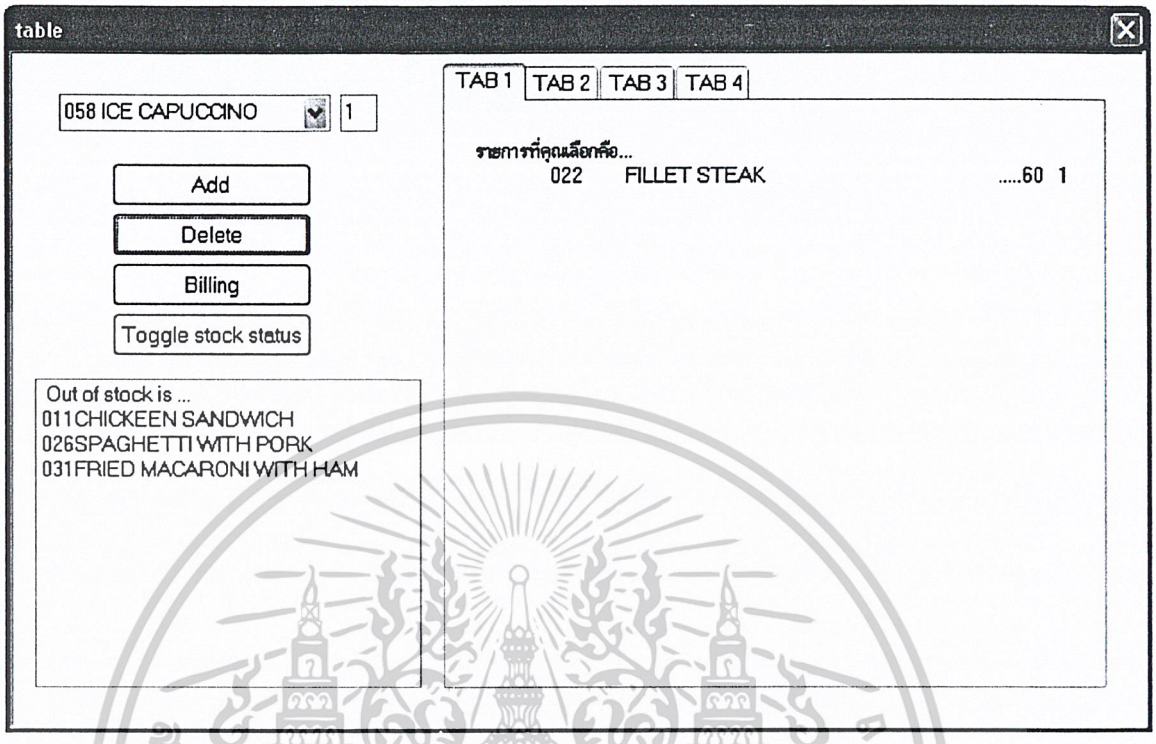


รูปที่ 3.27 หน้าต่างแสดงผลเมื่อมีการสั่งอาหารรายการที่ 21, 29, 51 และ 61 ที่โต๊ะที่ 3



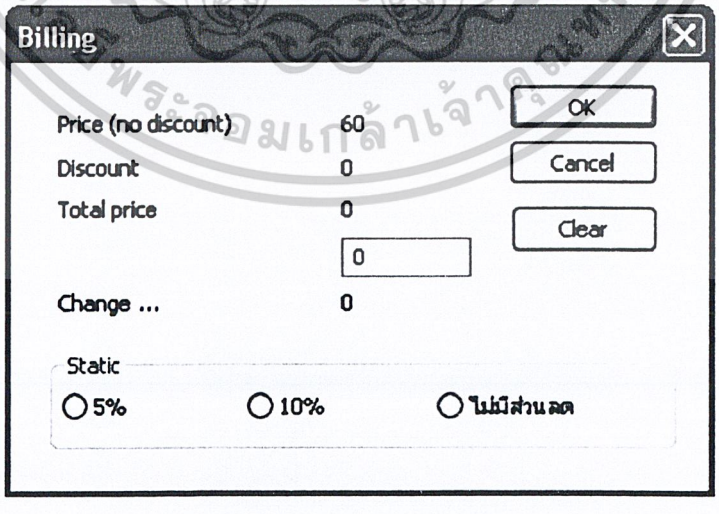
รูปที่ 3.28 หน้าต่างแสดงผลเมื่อมีการสั่งอาหารรายการที่ 22 และ 58 ที่โต๊ะที่ 1

5. เมื่อทำการลดจำนวนรายการอาหารที่สั่งก็จะทำให้รายการอาหารบนหน้าจอลดลงดังนี้



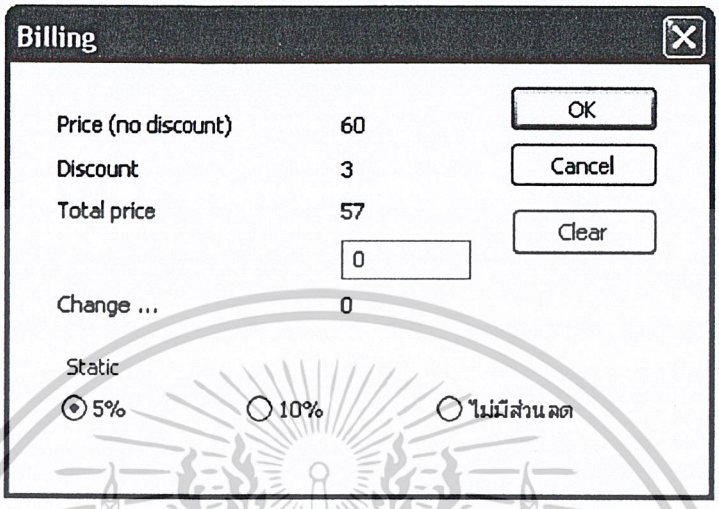
รูปที่ 3.29 หน้าต่างแสดงผลเมื่อมีการสั่งแก้ไขโดยการลบรายการที่ 58 จำนวน 1 ที่ ที่โต๊ะที่ 1

6. เมื่อมีการกดปุ่มคิดเงินจะมีหน้าจอแสดงผลดังนี้

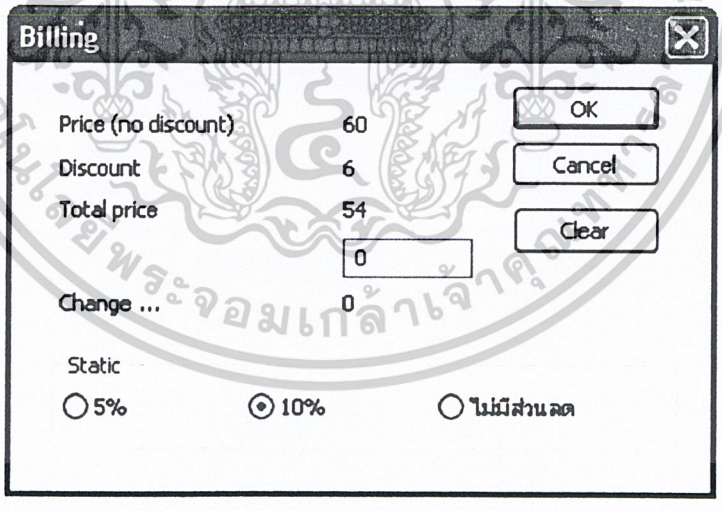


รูปที่ 3.30 หน้าต่างคิดเงิน

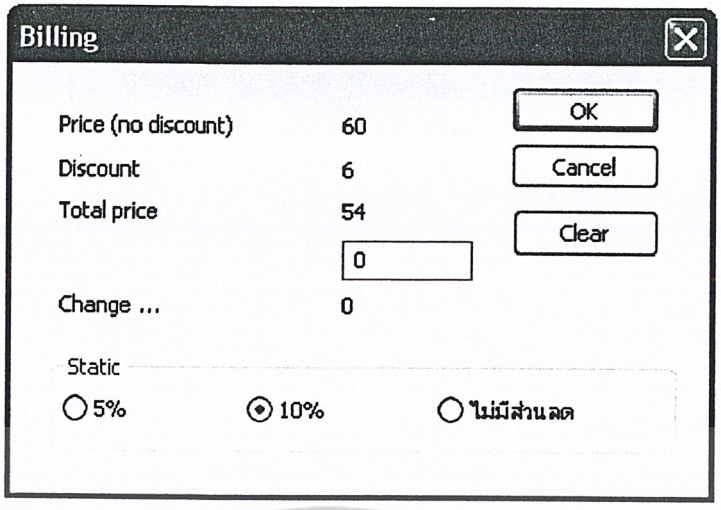
7. เมื่อมีการกดปุ่ม เพื่อคิดเงินและมีการเลือกส่วนลด 0% จะแสดงผลดังนี้



รูปที่ 3.31 หน้าต่างคิดเงินเมื่อ เลือกส่วนลด 0%

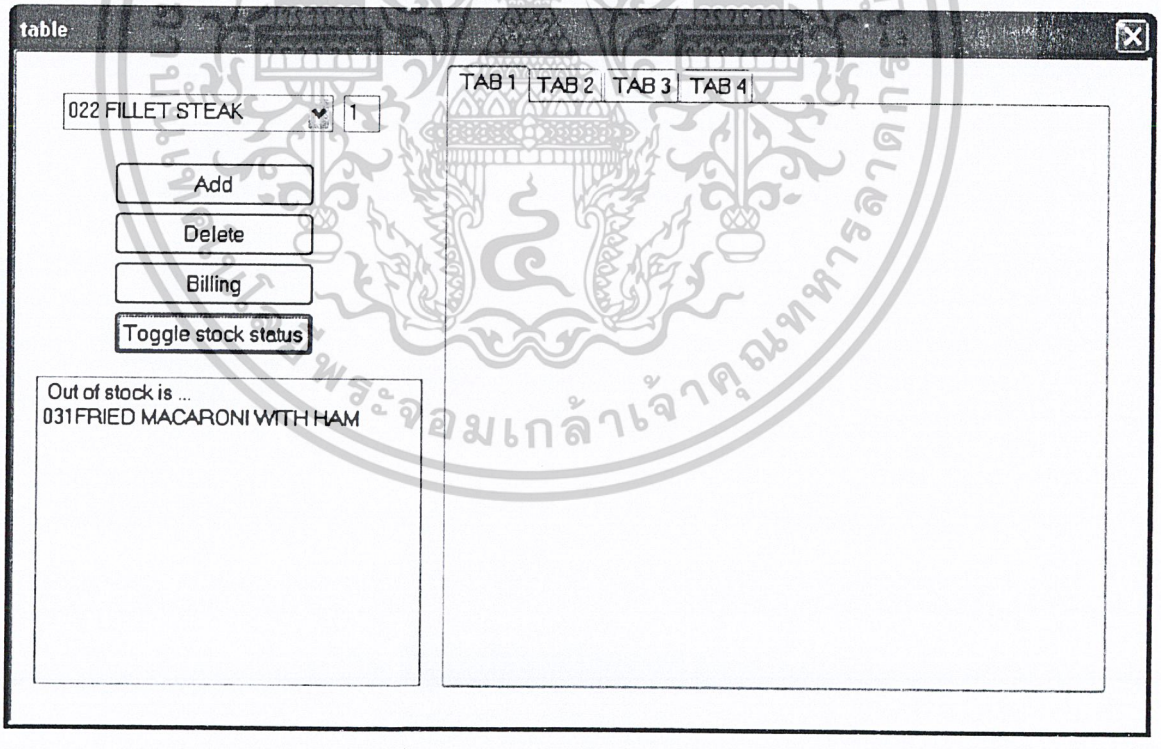


รูปที่ 3.32 หน้าต่างคิดเงินเมื่อ เลือกส่วนลด 5%



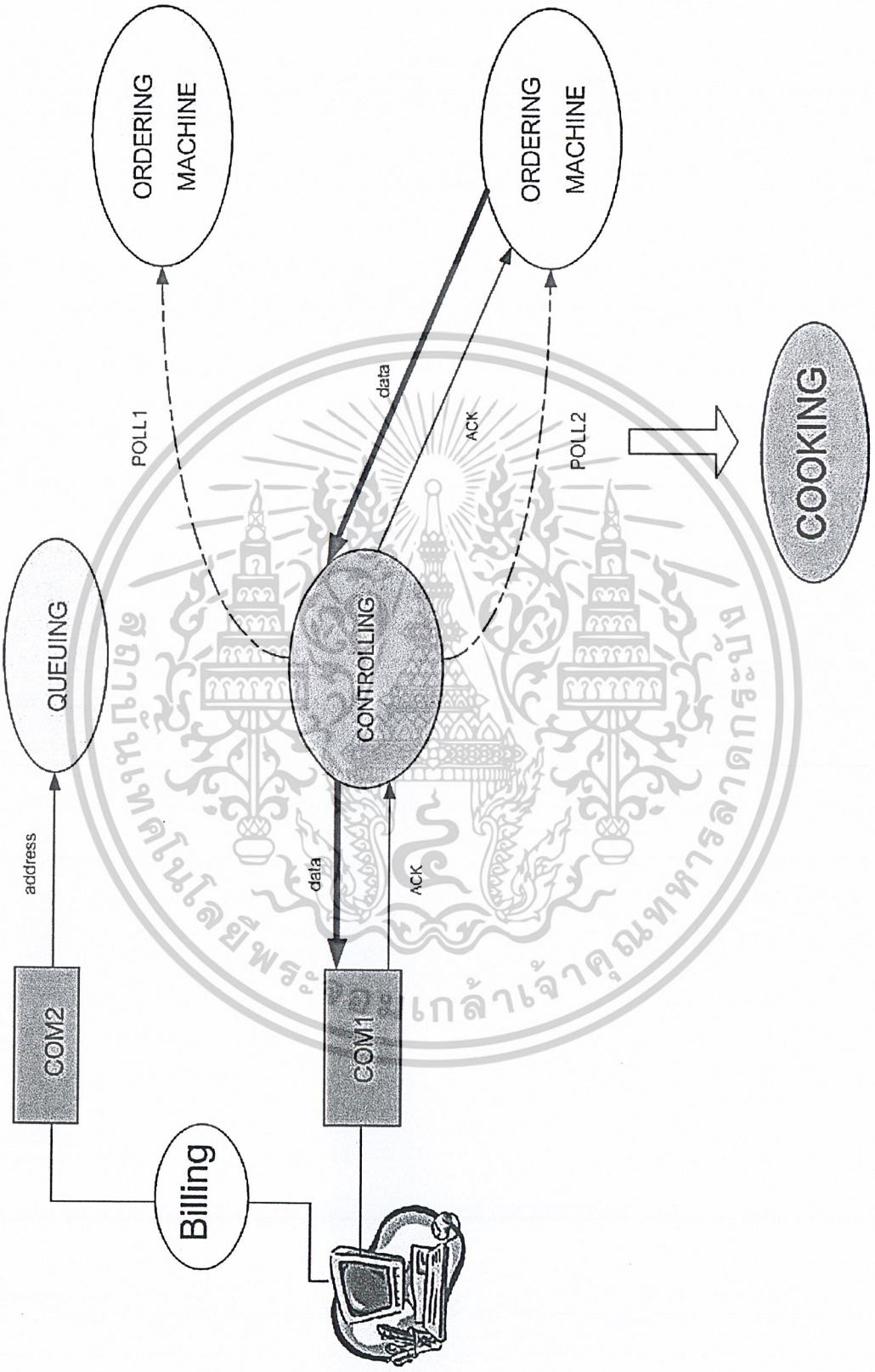
รูปที่ 3.33 หน้าต่างคิดเงินเมื่อ เลือกส่วนลด 10%

8. เมื่อมีการกดปุ่มเคลียร์เพื่อทำการลบรายการอาหารทั้งหมดออกจากรายการ จะแสดงผลทางหน้าจอดังนี้



รูปที่ 3.34 หน้าต่างหลักเมื่อกดปุ่มเคลียร์เพื่อทำการลบรายการอาหารทั้งหมด

3.4 หลักการทำงานของระบบ



รูปที่ 3.35 วิธีการติดต่อสื่อสารระหว่างส่วนต่างๆภายในระบบ

จากรูปที่ 3.35 วิธีการติดต่อสื่อสาร สามารถอธิบายการทำงานของระบบได้ดังนี้

ในสภาวะปกติขณะไม่มีผู้ใช้บริการทำการสั่งอาหาร ส่วนควบคุมจะทำการส่งสัญญาณโพล เพื่อถามส่วนสั่งอาหาร แต่ละโต๊ะว่ามีการสั่งอาหารหรือไม่ ถ้าหากไม่มีก็จะส่งสัญญาณโพลที่มีแอดเดรสของโต๊ะถัดไปเรื่อยๆ

เมื่อมีผู้ใช้บริการเข้ามาในร้านอาหาร จะต้องทำการกดคิว ส่วนจัดคิวก็จะจัดคิวพร้อมทั้งออกบัตรคิวให้ทางเครื่องพริ้นเตอร์ เมื่อถึงคิวก็จะทำการเรียกคิวพร้อมทั้งแจ้งหมายเลขโต๊ะให้ผู้ใช้บริการทราบด้วยเสียง และจอแสดงผลแบบแอลอีดีดอทเมทริกซ์ จากนั้นผู้ใช้บริการก็จะเข้าไปนั่งที่โต๊ะอาหาร

และเมื่อผู้ใช้บริการสั่งอาหารโดยกดหมายเลขรายการอาหารตามเมนูที่วางไว้ เครื่องสั่งอาหารก็จะเก็บข้อมูลไว้เพื่อรอสัญญาณโพลจากส่วนควบคุม เมื่อมีสัญญาณโพลมา ก็จะส่งข้อมูลรายการอาหารไปให้ส่วนควบคุม ส่วนควบคุมก็จะทำการตรวจสอบความถูกต้องของรายการอาหาร ถ้าถูกต้องก็จะส่งข้อมูลพร้อมกับแอดเดรสของโต๊ะที่สั่งอาหารไปยังเครื่องคอมพิวเตอร์ เครื่องคอมพิวเตอร์จะตรวจสอบว่าอาหารนั้นมีหรือหมด ถ้ามีก็จะส่งสัญญาณตอบรับกลับไปให้ส่วนควบคุมระบบพร้อมกับสร้างหน้าต่างขึ้นที่หน้าจอว่าที่โต๊ะนี้ได้สั่งรายการอะไรมาบ้าง เป็นจำนวนเท่าไร เป็นราคาเท่าไร จากนั้นส่วนควบคุมก็จะส่งสัญญาณตอบรับกลับไปยังเครื่องสั่งอาหาร เมื่อเครื่องสั่งอาหารได้รับก็จะถือว่าการสั่งอาหารหนึ่งอย่างนั้นเสร็จสมบูรณ์

และเมื่อผู้ใช้บริการทานอาหารเสร็จก็จะต้องลุกไปจ่ายเงินที่แคชเชียร์ ซึ่งอยู่หน้าคอมพิวเตอร์ หลังจากจ่ายเงินเสร็จแคชเชียร์ก็จะกดปุ่มเพื่อแจ้งให้ส่วนจัดคิวทราบ ส่วนจัดคิวก็จะเรียกคิวต่อไปที่รออยู่พร้อมกับบอกเบอร์โต๊ะ

ส่วนของครัวนั้นจะมใช้จอแอลซีดีในการแสดงรายการอาหาร โดยรายการอาหารนี้จะได้จากการตรวจจับสัญญาณในอากาศ เมื่อมีการสั่งอาหารที่เสร็จสมบูรณ์ก็จะเก็บรายการอาหารนั้นไว้ และโชว์ขึ้นที่จอแอลซีดีทีละรายการเมื่อพ่อครัวกดปุ่ม

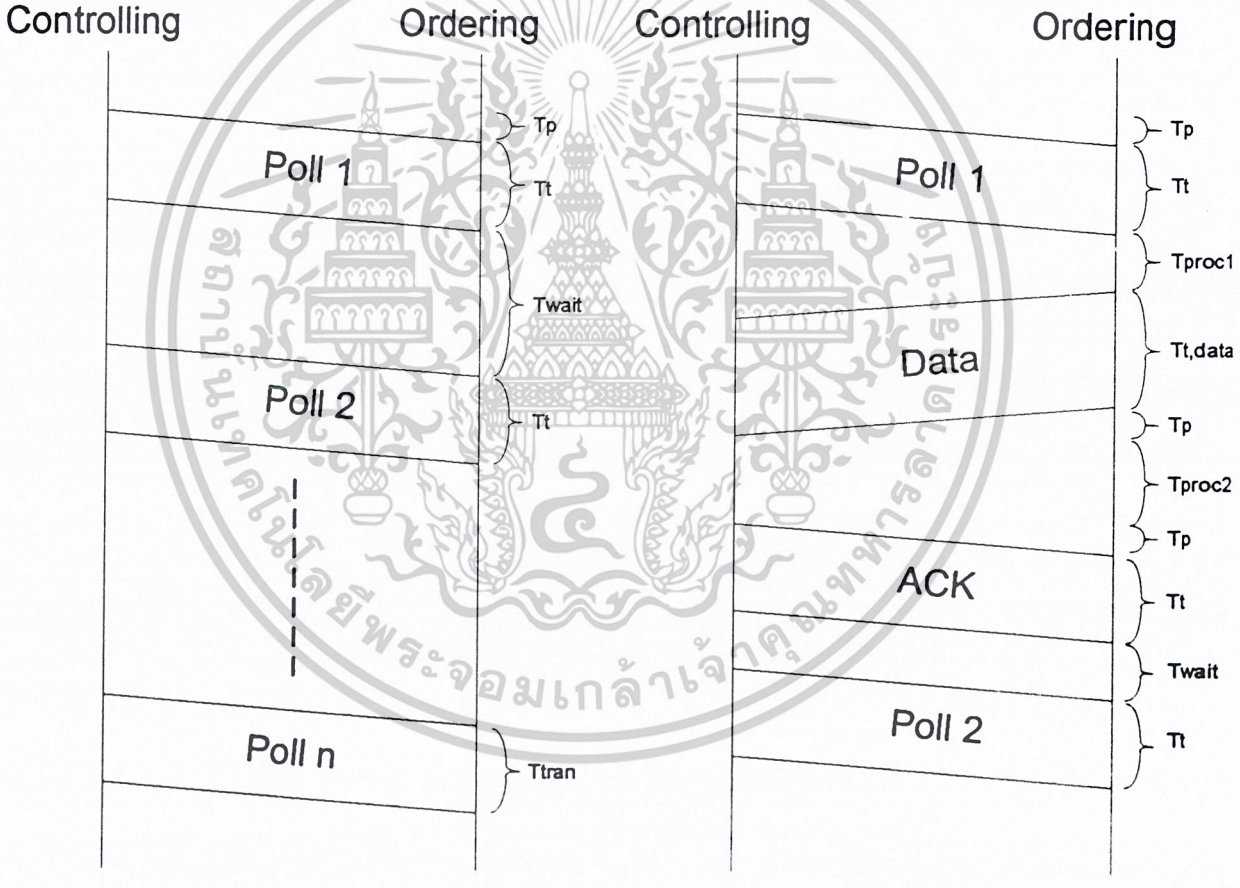
นอกจากนี้ในกรณีที่รายการอาหารบางอย่างหมด ก็จะสามารถใส่ข้อมูลลงไปในคอมพิวเตอร์ได้ คอมพิวเตอร์จะเก็บข้อมูลไว้และแจ้งให้ส่วนควบคุมทราบ ส่วนควบคุมก็จะใช้แอดเดรสพิเศษ เพื่อบอกเครื่องสั่งอาหารทุกเครื่องว่ามีรายการอาหารนั้นหมด

สำหรับการจัดเฟรมข้อมูล การเข้ารหัส และการส่งข้อมูลในแต่ละส่วนจะใช้รูปแบบที่ดัดแปลงมาจากโปรโตคอล KeeLoq ซึ่งจะอธิบายในหัวข้อถัดไป

3.5 วิธีการติดต่อสื่อสารภายในระบบ

3.5.1วิธีการติดต่อสื่อสารระหว่างส่วนควบคุมกับส่วนสั่งอาหาร

สามารถอธิบายได้จากแผนผังเวลาในรูปที่ 3.36 แบ่งออกเป็น 2 สถานะ คือสถานะแรกไม่มีการสั่งอาหารจากผู้ใช้บริการ ส่วนควบคุมก็จะส่งสัญญาณโพล โดยเปลี่ยนแอดเดรสไปตามหมายเลขโต๊ะ เรียงตามลำดับ เพื่อถามส่วนสั่งอาหารแต่ละโต๊ะว่ามีการสั่งอาหารหรือไม่ ส่วนอีกสถานะหนึ่งก็คือเมื่อมีการสั่งอาหาร ส่วนควบคุมก็จะหยุดส่งสัญญาณโพล เพื่อประมวลผลข้อมูลการสั่งอาหาร เมื่อประมวลผลเสร็จก็จะส่งสัญญาณตอบกลับ เพื่อแจ้งผลการสั่งอาหารไปยังส่วนสั่งอาหาร แล้วจึงส่งสัญญาณโพลของโต๊ะถัดไป



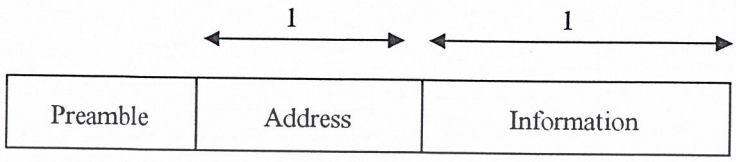
สถานะไม่มีการสั่งอาหาร สถานะมีการสั่งอาหารที่โต๊ะ 1

รูปที่ 3.36 แผนผังเวลาแสดงวิธีการติดต่อสื่อสารระหว่างส่วนควบคุมกับส่วนสั่งอาหาร

3.5.2 การจัดเฟรมข้อมูล

รูปแบบการจัดเฟรมข้อมูลของส่วนควบคุมและส่วนสั่งอาหารจะต่างกัน ตามความเหมาะสมของข้อมูลดังนี้

<u>รูปแบบเฟรมของส่วนควบคุม</u>



<u>พรีแอมเบิล (Preamble)</u>

เนื่องจากวิธีการเข้าถึง (Access) ที่ใช้ในระบบนี้ ไม่ได้เตรียมการซิงโครนัสไว้ระหว่างด้านส่งและด้านรับ ดังนั้นการรับส่งเฟรมจะทำแบบอะซิงโครนัส โดยที่จะต้องเพิ่มสัญญาณภายในเฟรมสำหรับบิตซิงโครนัสก็คือพรีแอมเบิล ซึ่งเป็นกลุ่มบิต

"10101010101010101010101010101010"

<u>แอดเดรส (Address)</u>

ระบบนี้ทำการสื่อสารกันในรูปแบบใช้สัญญาณความถี่เดียว ดังนั้นทุกสถานีจึงได้รับข้อมูลเช่นเดียวกัน ภายในเฟรมจึงต้องมีส่วนแอดเดรสผู้รับ ซึ่งในระบบนี้จะใช้แอดเดรสแทนเหมายเลขโต๊ะแต่ละตัว และได้แบ่งแอดเดรสออกเป็น 2 ประเภทคือ

- แอดเดรสอิสระ (Identify address) ใช้ในการส่งข้อมูลให้กับโต๊ะแต่ละตัว จะใช้รหัสคือ

0000 0001 – 0111 1111

- กรุ๊ปแอดเดรส (Group address) ใช้ในการส่งข้อมูลไปยังโต๊ะทุกโต๊ะ เพื่อแจ้งข่าวสารเมื่อมีอาหารชนิดใดชนิดหนึ่งหมด จะใช้รหัสคือ

1111 1111

ข่าวสาร (Information)

เป็นฟิลด์ที่ใช้ในการเก็บข่าวสาร ซึ่งระบบนี้จะแบ่งฟิลด์นี้ออกเป็น 2 ประเภทคือ ส่วนของข้อมูล และส่วนของการควบคุม ดังนี้

ข้อมูล จะใช้เป็นรหัสของรายการอาหารแต่ละชนิด โดยจะมีบิตแรกเป็น 0 ดังนั้นจะมีรายการอาหารได้สูงสุด 255 ชนิด คือ

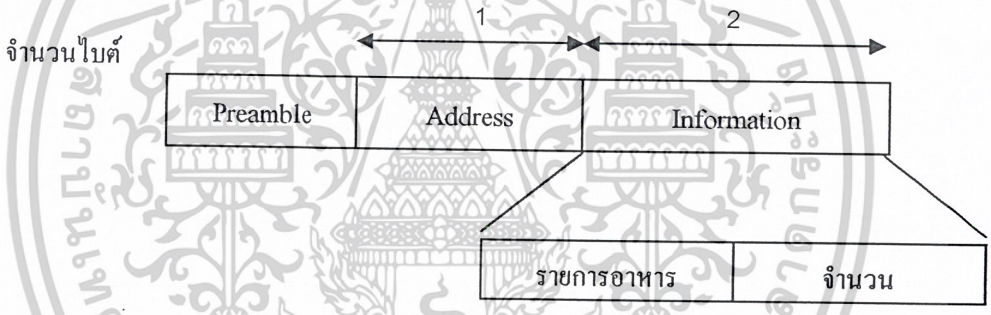
รหัส 0000 0001 ถึง 0111 1111

ควบคุม จะใช้เป็นรหัสแสดงสัญญาณควบคุมต่างๆ โดยจะมีบิตแรกเป็น 1 จะใช้รหัสดังนี้

สัญญาณ ACK – 1000 0000

สัญญาณ Poll – 1000 1000

รูปแบบเฟรมของเครื่องสั่งอาหาร



พรีแอมเบิล (Preamble)

เป็นกลุ่มบิต "10101010101010101010101010101010" เช่นเดียวกับการจัดเฟรมของส่วนควบคุมระบบ

แอดเดรส (Address)

เป็นแอดเดรสของแต่ละโต๊ะเช่นเดียวกับการจัดเฟรมของส่วนควบคุมระบบ

ข่าวสาร (Information)

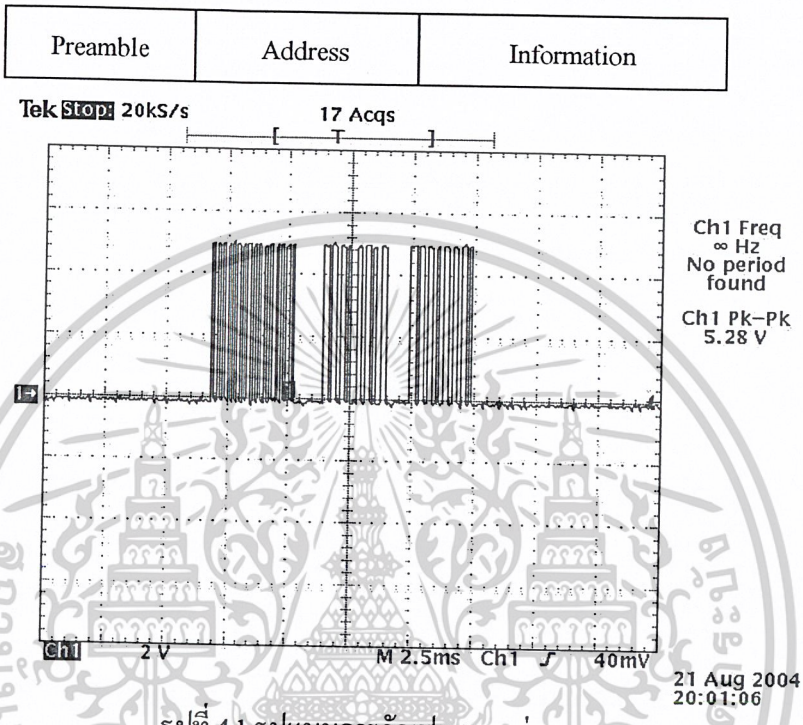
ในส่วนนี้จะมีการจัดข่าวสารออกเป็น 2 ไบต์ โดยไบต์แรกจะเป็นส่วนของรายการอาหารที่มีการเข้ารหัสเหมือนกับส่วนควบคุมระบบ ส่วนไบต์หลังจะเป็นจำนวนของอาหารที่สั่ง

บทที่ 4

ผลการทดลอง

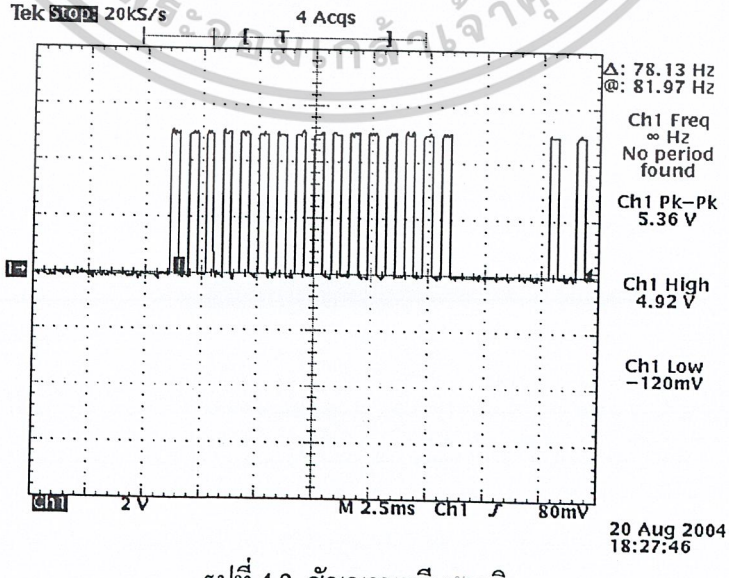
4.1 ส่วนควบคุม (Controlling)

4.1.1 รูปแบบการจัดเฟรม ประกอบด้วยส่วนของพรีแอมเบิล ส่วนของแอดเดรส 1 ไบต์ และส่วนของข่าวสาร 1 ไบต์ ดังนี้



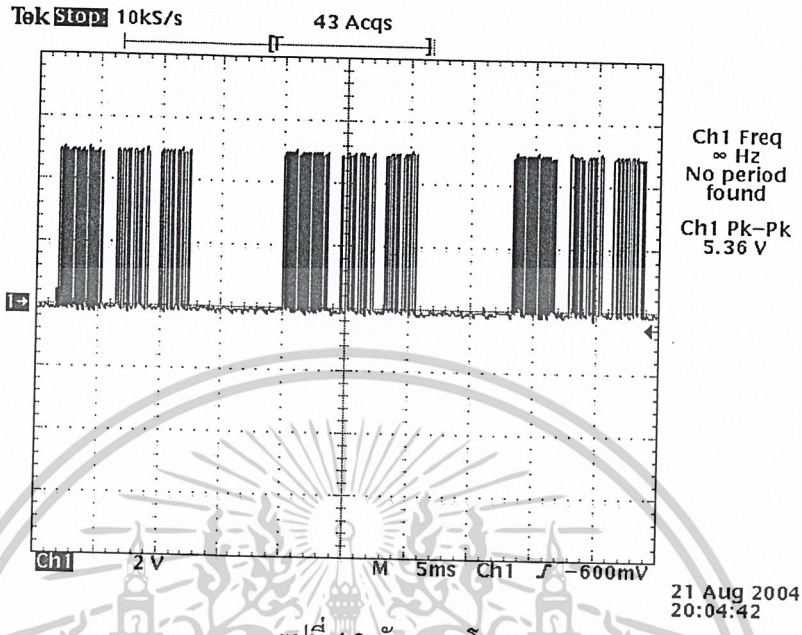
รูปที่ 4.1 รูปแบบการจัดเฟรมของส่วนควบคุม

4.1.2 สัญญาณพรีแอมเบิล เป็นกลุ่มบิต “ 10101010101010101010101010101010 ” ใช้สำหรับการซิงโครนัสของด้านส่งและด้านรับ



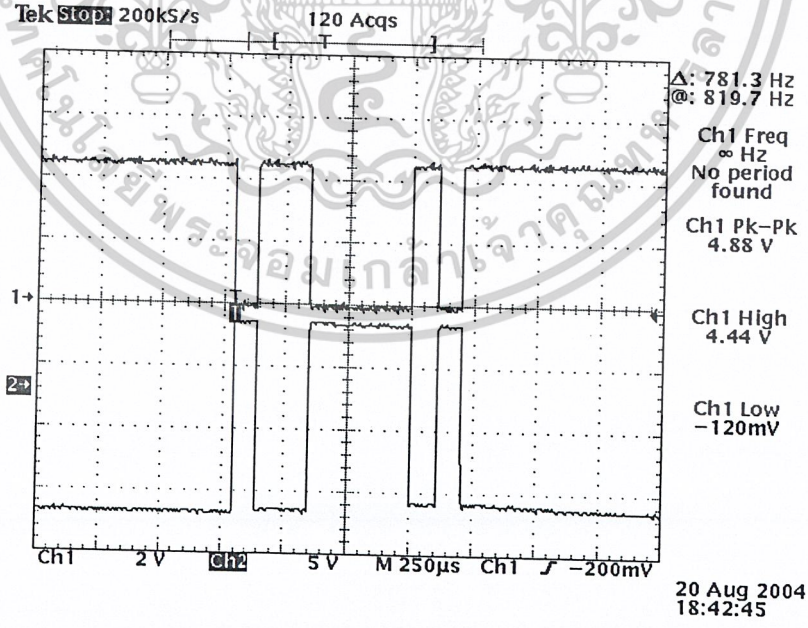
รูปที่ 4.2 สัญญาณพรีแอมเบิล

4.1.3 สัญญาณโพล ในสภาวะปกติที่ไม่มีการสั่งอาหารส่วนควบคุมจะส่งสัญญาณโพล โดยเปลี่ยนค่าแอคเดรสไปเรื่อยๆทีละโต๊ะ



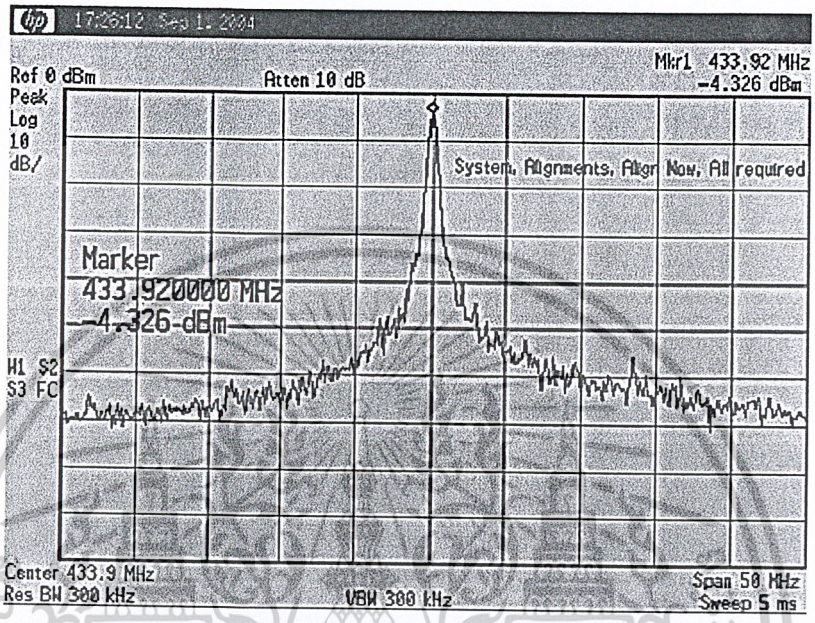
รูปที่ 4.3 สัญญาณโพล

4.1.4 เปรียบเทียบการแปลงระดับสัญญาณแบบทีทีแอล กับสัญญาณอาร์เอส232 ของวงจรเชื่อมต่อกับพอร์ทอนุกรมของคอมพิวเตอร์

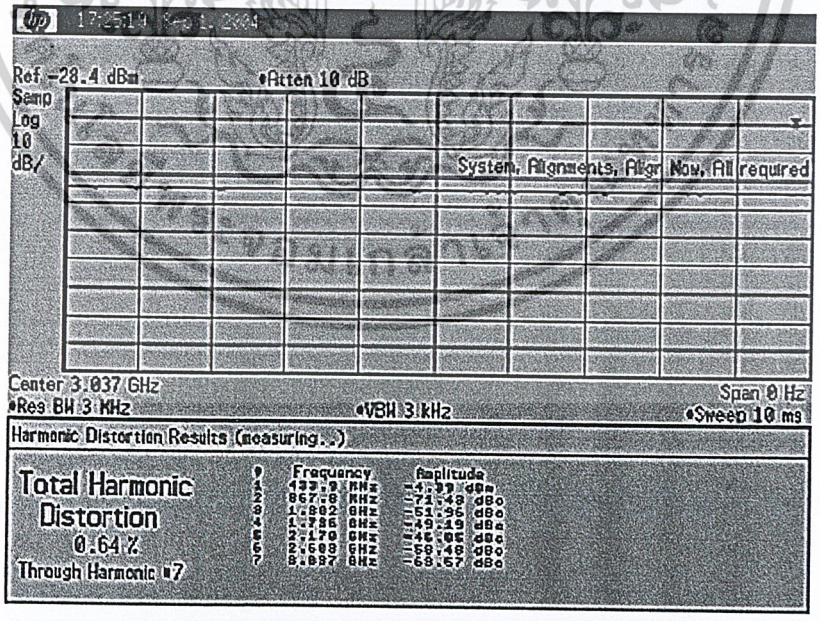


รูปที่ 4.4 การเปรียบเทียบการแปลงระดับสัญญาณของวงจรเชื่อมต่อกับพอร์ทอนุกรมของคอมพิวเตอร์

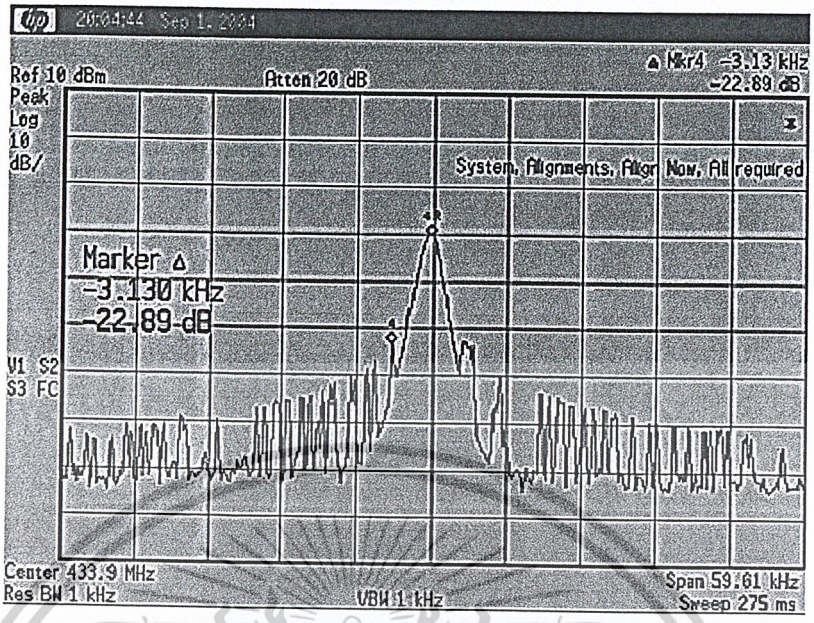
4.1.5 สเปกตรัมของสัญญาณพาห์ ค่าฮามอร์นิกดิสทอร์ชั่น และสัญญาณที่ผ่านการมอดูเลต ที่ความถี่ 433.92 MHz จากผลการทดลองพบว่า เครื่องส่งมีกำลังส่งเท่ากับ -4.326 dBm สัญญาณพาห์มีค่าฮามอร์นิกดิสทอร์ชั่นที่ต่ำมากเพียง 0.64 %



รูปที่ 4.5 สเปกตรัมของสัญญาณพาห์ความถี่ 433.92 MHz

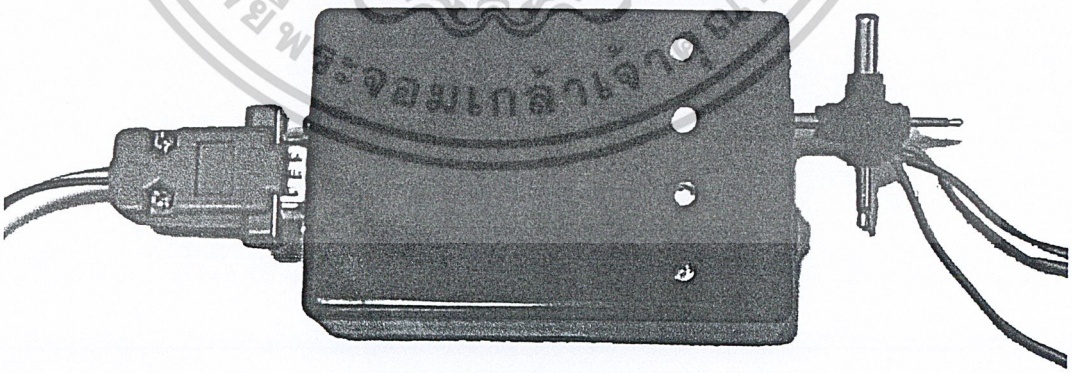


รูปที่ 4.6 การวัดคุณภาพของสัญญาณพาห์ด้วยค่าฮามอนิกดิสทอร์ชั่น



รูปที่ 4.7 สเปกตรัมของสัญญาณที่ผ่านการมอดูเลต

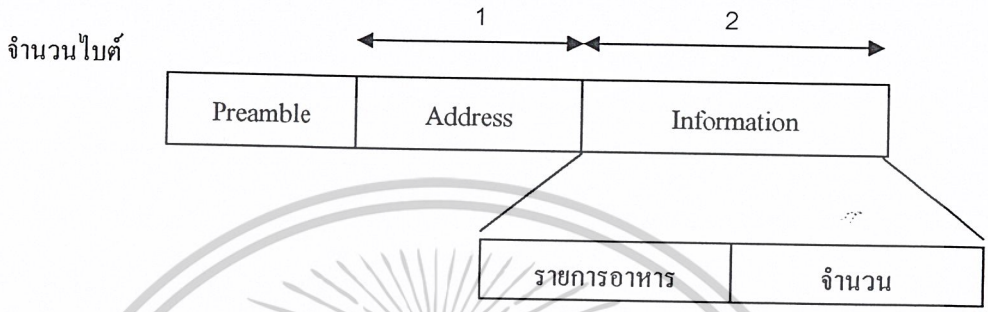
4.1.6 ชิ้นงานของส่วนควบคุม ประกอบด้วยช่องเสียบหม้อแปลง 9 โวลต์ หลอดแอลอีดีแสดงสถานะการทำงาน และช่องเสียบพอร์ทอนุกรม

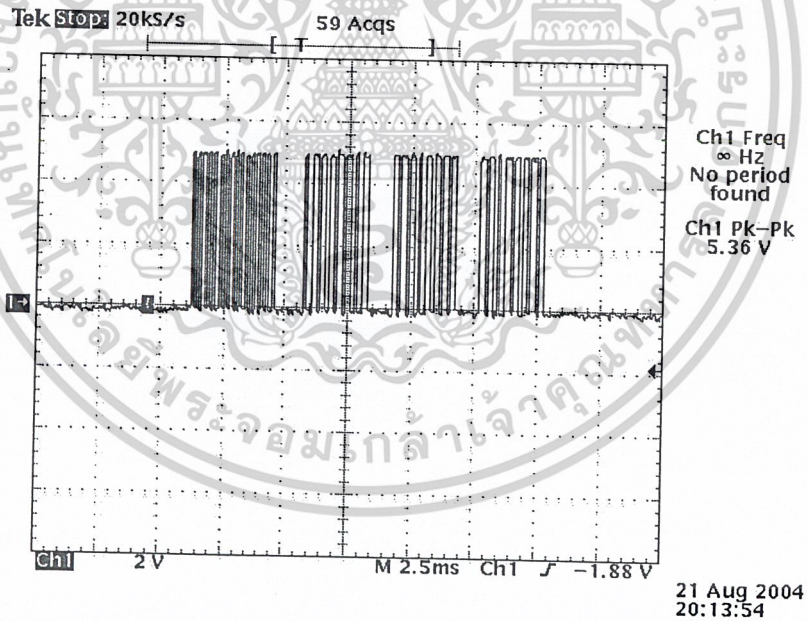


รูปที่ 4.8 ชิ้นงานของส่วนควบคุม

4.2 ส่วนสั่งอาหาร (Ordering)

4.2.1 รูปแบบการจัดเฟรม ประกอบด้วยส่วนของพรีแอมเบิล 2 ไบต์ ส่วนของแอดเดรส 1 ไบต์ และส่วนของข่าวสาร 2 ไบต์ ดังนี้



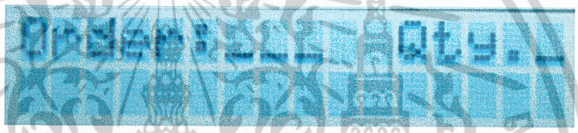


รูปที่ 4.9 รูปแบบการจัดเฟรมของส่วนสั่งอาหาร

4.2.2 วิธีการใช้งานเครื่องสั่งอาหารและการแสดงผลทางจอแอลซีดี
แบ่งเป็น 6 ขั้นตอน ดังนี้

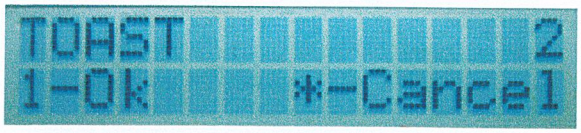
1. เมื่อกดปุ่มเริ่มต้น จะแสดงข้อความต้อนรับ และให้กด 0 เมื่อต้องการสั่งอาหาร

2. ใส่รหัสรายการอาหาร 3 หลัก ที่ต้องการสั่ง ซึ่งดูได้จากเมนู

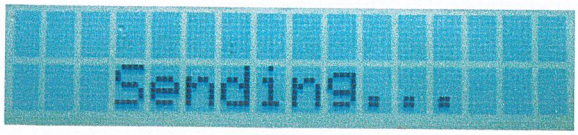


3. ใส่จำนวนที่ต้องการสั่ง 1 หลัก หรือกด * เพื่อแก้ไขรหัสรายการอาหาร

4. แสดงรายการอาหารและจำนวนที่สั่ง กด 1 ถ้ารายการอาหารและจำนวนที่ต้องการสั่งถูกต้อง
หรือ กด * เพื่อแก้ไขรายการอาหารหรือจำนวนที่ต้องการสั่ง



5. แสดงข้อความให้ผู้ใช้ทราบว่า กำลังส่งข้อมูลการสั่งอาหาร

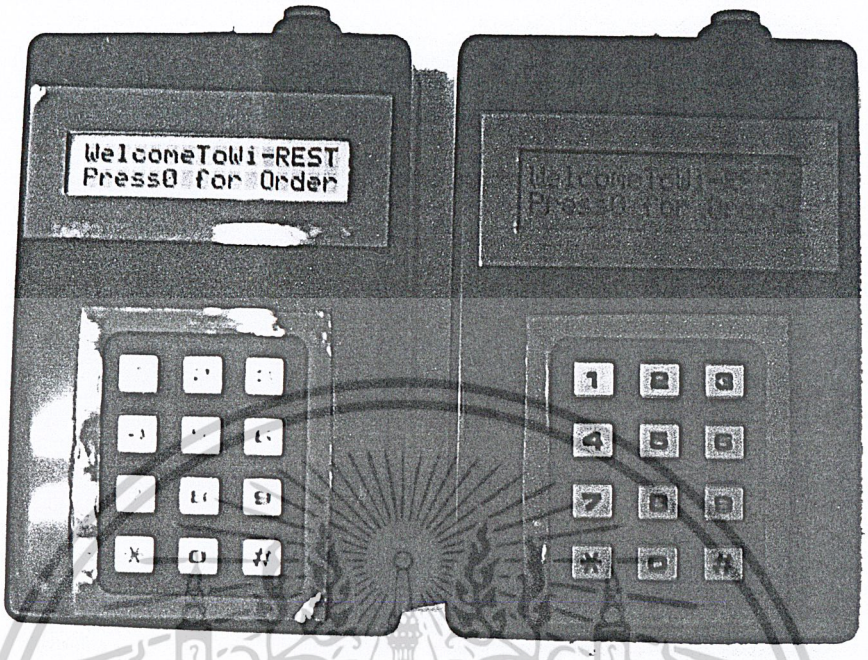


6. ข้อความแจ้งให้ผู้ใช้ทราบผลการสั่งอาหาร

- การสั่งอาหารเสร็จสมบูรณ์

- การสั่งอาหารมีความผิดพลาดเกิดขึ้นในการรับส่งข้อมูล

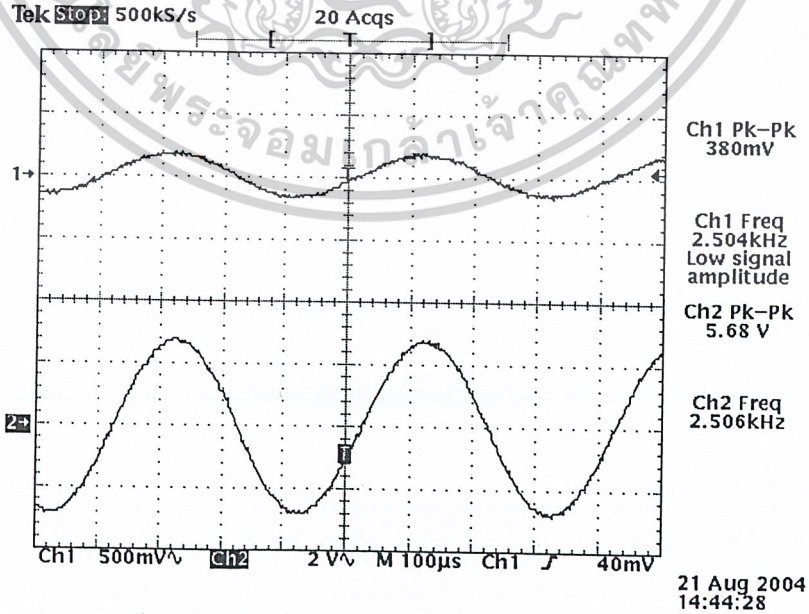
4.2.3 ชิ้นงานของส่วนสั่งอาหาร



รูปที่ 4.10 ชิ้นงานของส่วนสั่งอาหาร

4.3 ส่วนการจัดคิว

4.3.1 การเปรียบเทียบสัญญาณอินพุทกับสัญญาณเอาท์พุทของวงจรเพาเวอร์แอมป์ พบว่ามีเกนที่ความถี่ 2.5 KHz ประมาณ 5.68/0.38 = 15 เท่า



รูปที่ 4.11 กราฟเปรียบเทียบสัญญาณอินพุทกับสัญญาณเอาท์พุทของวงจรเพาเวอร์แอมป์ที่ความถี่ 2.5 KHz

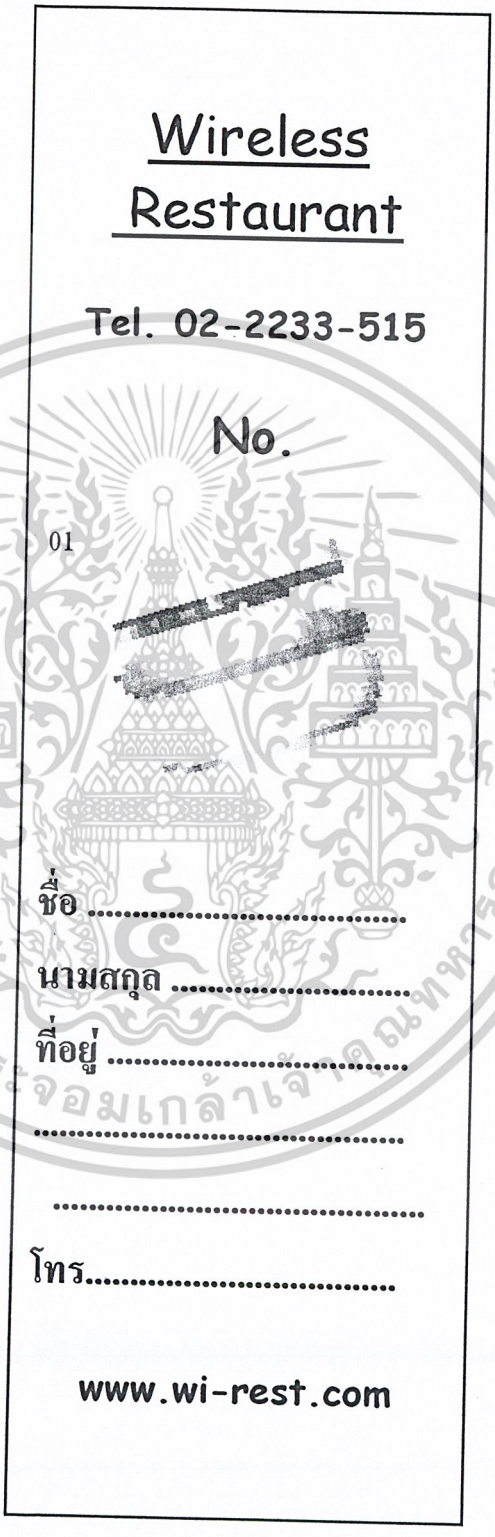
4.3.2 การแสดงผลทางจอแสดงผลแบบแอลอีดีดอทเมทริกซ์

รูปที่ 4.12 ข้อความต้อนรับ

รูปที่ 4.13 การแสดงผลเมื่อกดคิวที่ 99

รูปที่ 4.14 การแสดงผลขณะเรียกคิวที่ 1 เข้าโต๊ะที่ 1

4.3.3 บัตรคิวหมายเลขที่ 01 ซึ่งพิมพ์ออกจากเครื่องพริ้นเตอร์ โดยการสั่งงานของไมโครคอนโทรลเลอร์ MCS-51



รูปที่ 4.15 บัตรคิวหมายเลข 01

4.3.4 ชิ้นงานของส่วนการจัดคิว ประกอบด้วย ปุ่มกดคิว จอแสดงผลแบบแอลอีดีดอทเมทริกซ์ พอร์ทขนานสำหรับเสียบเครื่องพริ้นเตอร์ และช่องเสียบหม้อแปลง 9 โวลต์

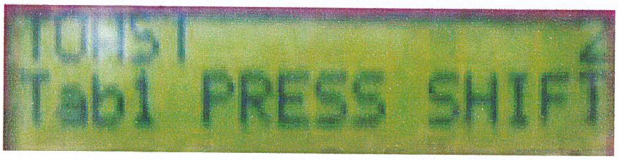
รูปที่ 4.16 ชิ้นงานของส่วนการจัดคิว

4.4 ส่วนครัว

4.4.1 วิธีการใช้งานและการแสดงผลทางจอแอลซีดี

1. ขณะไม่มีข้อมูลการสั่งอาหาร

2. ขณะมีข้อมูลการสั่งอาหาร สามารถกดปุ่มเพื่อดูรายการอาหารถัดไป

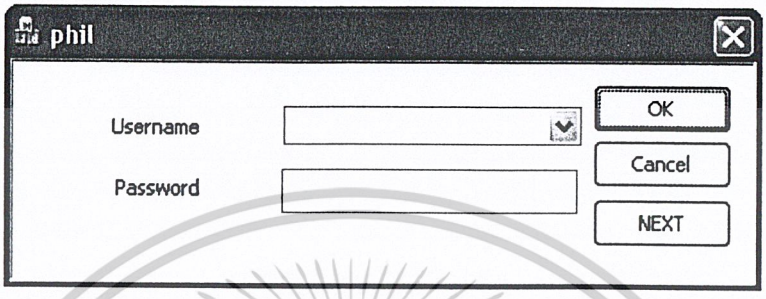


4.4.2 แสดงชิ้นงานของส่วนครัว

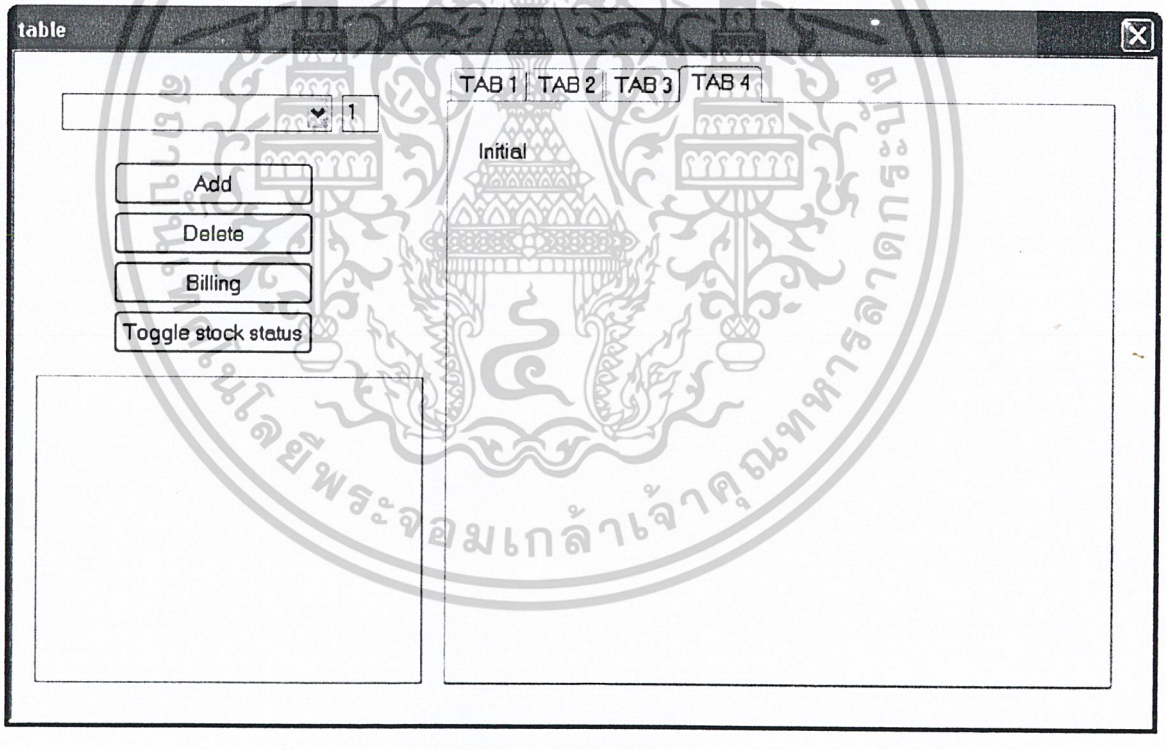
รูปที่ 4.17 ชิ้นงานของส่วนครัว

4.5 ส่วนคิดเงิน

4.5.1 ทำการล็อกอินโดยการใส่ชื่อผู้ใช้และรหัสผ่าน ในหน้าต่างในรูปที่ 4.18 เมื่อข้อมูลถูกต้อง จะปรากฏหน้าต่างหลักในรูปที่ 4.19

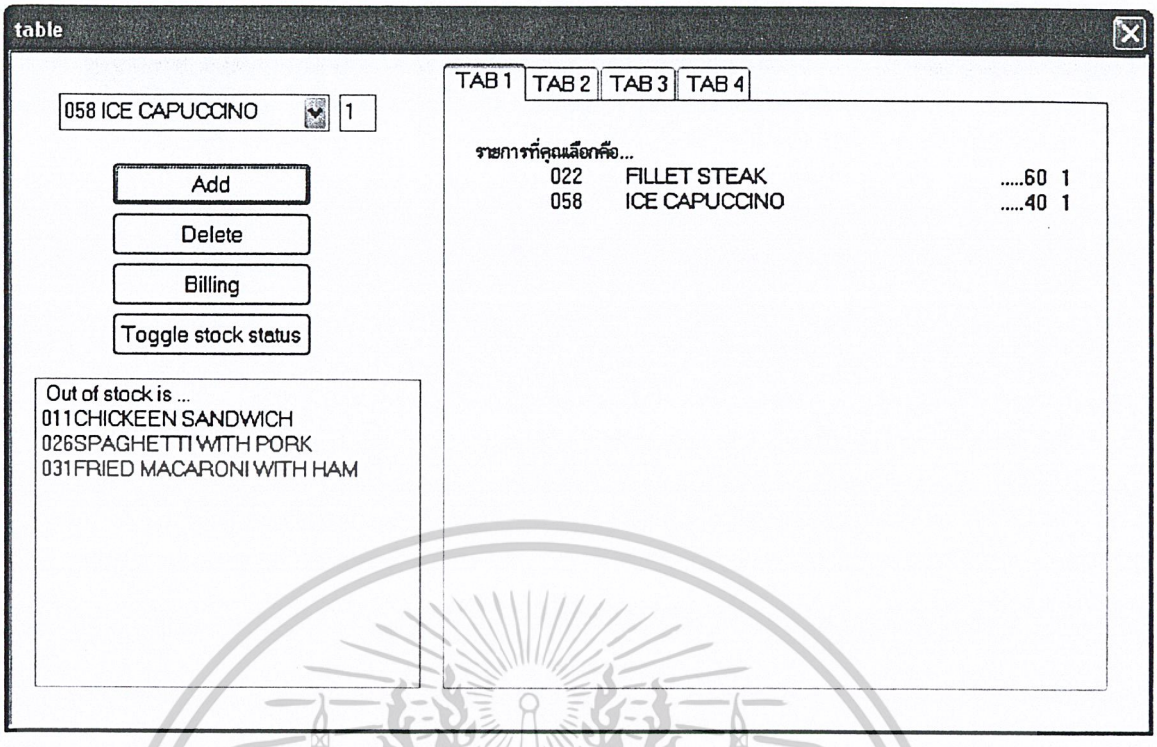


รูปที่ 4.18 หน้าต่างล็อกอิน



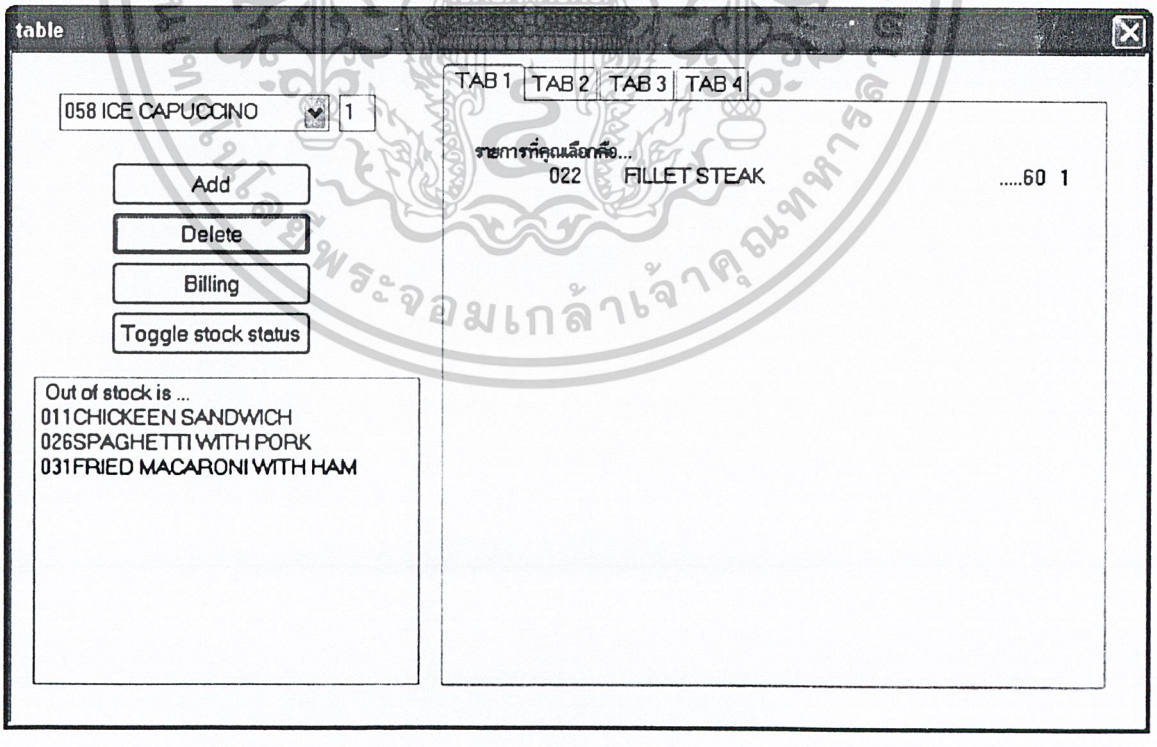
รูปที่ 4.19 หน้าต่างหลัก

4.5.2 ทำการทดลองสั่งอาหารโดยเครื่องสั่งอาหารของโต๊ะที่ 1 รายการอาหารที่สั่ง คือ รายการที่ 22 จำนวน 1 ที่ และได้ทดลองสั่งผ่านทางหน้าจอคอมพิวเตอร์ รายการที่ 58 จำนวน 1 ที่ ปรากฏผลดัง รูปที่4.20



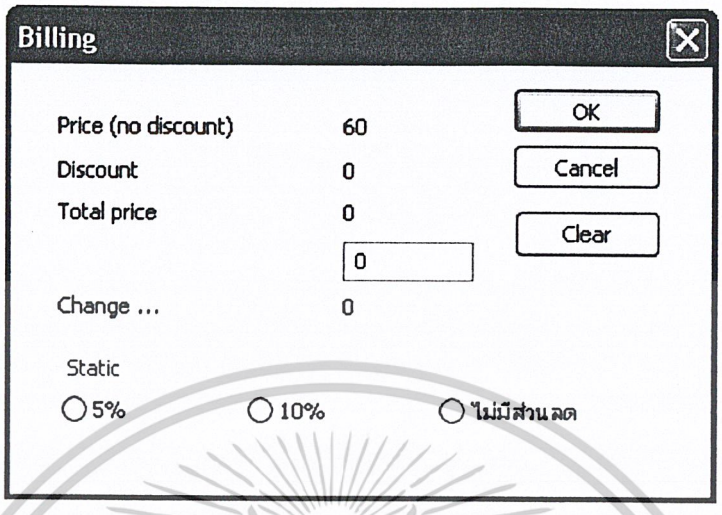
รูปที่ 4.20 การแสดงข้อมูลรายการอาหารในหน้าต่างหลัก

4.5.3 ทำการแก้ไขโดยตัดรายการที่ 58 ออก



รูปที่ 4.21 การแก้ไขข้อมูลรายการอาหาร

4.5.4 คิดเงินโดยกำหนดส่วนลด 0%



รูปที่4.22 หน้าต่างแสดงการคิดเงิน

บทที่ 5
บทวิจารณ์และบทสรุป

จากผลการทดลองที่ได้สามารถสรุปและวิจารณ์ได้ดังนี้

การติดต่อสื่อสารกันภายในระบบซึ่งใช้วิธีการเข้าถึงแบบหยั่งสัญญาณ โดยมีส่วนควบคุมเป็นศูนย์กลางสามารถลดปัญหาการชนกันของข้อมูลที่ส่วนสั่งอาหารส่งเข้ามาได้อย่างสมบูรณ์ และยังสามารถใช้ร่วมกับการเข้ารหัส การจัดเฟรมข้อมูลที่ดัดแปลงมาจากโปรโตคอล KeeLoq ได้เป็นอย่างดี เนื่องจากการเข้ารหัสแบบ KeeLoq สามารถลดปัญหาการสูญเสียการซิงโครนัสได้ และยังมีการส่งพรีแอมเบิลก่อนส่งข้อมูล ทำให้การซิงโครนัสระหว่างด้านส่งและด้านรับดีขึ้น ด้านรับจึงสามารถรับการหยั่งสัญญาณจากด้านส่งได้อย่างถูกต้อง นอกจากนี้ยังมีการเข้ารหัสที่ง่ายต่อการเขียนโปรแกรม เพื่อถอดรหัส

สำหรับในด้านฮาร์ดแวร์นั้นมีวงจรสำคัญคือเครื่องส่งซึ่งใช้ไอซี rfPIC12f675 ที่มีทั้งไมโครคอนโทรลเลอร์และเครื่องส่งในตัว พบว่าเครื่องส่งนี้มีประสิทธิภาพสูง สัญญาณพาห์ที่วัดได้นั้นมีความเสถียร และวัดค่าฮาร์โมนิกดิสทอร์ชั่นได้น้อยมากเพียงไม่ถึง 1 เปอร์เซ็นต์ และมีกำลังส่งที่สูงถึงประมาณ -4 dBm ซึ่งถือว่าสูงมากเมื่อเทียบกับขนาดของไอซี ส่วนเครื่องรับก็รับสัญญาณได้เป็นอย่างดี วงจรเล่น/บันทึกเสียงซึ่งต่อผ่านวงจรขยาย เสียงที่เล่นออกมามีเสียงซ่ารบกวนเล็กน้อยแต่ก็อยู่ในเกณฑ์ที่ฟังได้ชัดเจนดี

ส่วนทางด้านซอร์ฟแวร์นั้นได้ใช้การเขียนโปรแกรมแอสแซมบลี สำหรับไมโครคอนโทรลเลอร์ตระกูล MCS-51 และตระกูล PIC12f พบว่ามีความแตกต่างกันพอสมควร แต่ก็สามารถเข้าใจได้ไม่ยากนัก และได้ใช้ภาษา C++ ในการเขียนโปรแกรมสำหรับรับข้อมูลรายการอาหาร และแสดงผลทางจอคอมพิวเตอร์ พบว่าโปรแกรมที่ได้ สามารถเรียนรู้วิธีการทำงานได้ง่ายและรวดเร็วสำหรับบุคคลทั่วไป

การทำงานของระบบโดยรวม จากการทดลองให้ผู้อื่นได้ลองใช้ระบบ พบว่าการใช้งานในส่วนต่าง ๆ ทั้งทางด้านผู้ให้บริการร้านอาหารและผู้ใช้บริการ สามารถใช้งานได้เป็นอย่างดี มีความรวดเร็วและเข้าใจวิธีการใช้งานได้ง่าย

<u>ปัญหาที่พบ</u>

1. การใช้งานอุปกรณ์บางอย่าง เช่น จอแสดงผลแบบแอลอีดีดอทเมทริกซ์ เครื่องพริ้นเตอร์ หาข้อมูล วิธีการเขียนโปรแกรมควบคุมได้ยาก
2. ไอซีมีขนาดเล็กมาก ไม่สามารถกัดลายปริ้นและบัดกรีได้เอง
3. เครื่องมือที่ใช้วัดสัญญาณความถี่สูงมีน้อย

<u>แนวทางการพัฒนา</u>

1. เพิ่มกำลังการส่งเพื่อให้สามารถใช้งานได้ในระยะไกลขึ้น สำหรับร้านอาหารขนาดใหญ่มาก หรือมีหลายชั้น
2. ลักษณะของฮาร์แวร์ ควรจะพัฒนาให้มีรูปลักษณ์สวยงาม เพื่อให้เหมาะสมกับการใช้งานในร้านอาหารจริง

ภาคผนวก

รายการอาหาร

WIRELESS RESTUARANT

BREAKFAST

001	TOAST	15.-
002	FRIED EGGS WITH HAM	30.-

SOUP

003	CHICKEN CREAM SOUP	25.-
004	MUSHROOM CREAM SOUP	25.-
005	CORN CREAM SOUP	25.-

SALAD

006	MIXED SALAD	30.-
007	TUNA FISH SALAD	35.-
008	CHICKEEN SALAD	35.-
009	BEEF STEAK SALAD	45.-
010	SEAFOOD SALAD	45.-

SANDWICH

011	CHICKEEN SANDWICH	20.-
012	TUNA FISH SANDWICH	20.-

013	HAM SANDWICH	20.-
014	HAM&CHEESE SANDWICH	20.-
015	GARLIC BREAD	20.-
016	FRENCH FRIED	25.-
017	CLUB SANDWICH	30.-

STEAK

018	CHICKEN STEAK	39.-
019	PORK STEAK	39.-
020	FISH STEAK	49.-
021	LONDON FISH STEAK	49.-
022	FILLET STEAK	60.-
023	PEPPER STEAK	80.-
024	FILLET MIGNON STEAK	90.-

SPAGHETTI & MACARONI

025	SPAGHETTI WITH BEEF	35.-
026	SPAGHETTI WITH PORK	35.-
027	SPAGHETTI WITH CHICKEN	35.-
028	SPAGHETTI NAPOLITAN	35.-
029	SPAGHETTI WITH MUSSEL	45.-
030	MACARONI WITH SHRIMP	40.-
031	MACARONI WITH HAM	35.-
032	PORK SAUSAGE	40.-

FRESH COFFEE

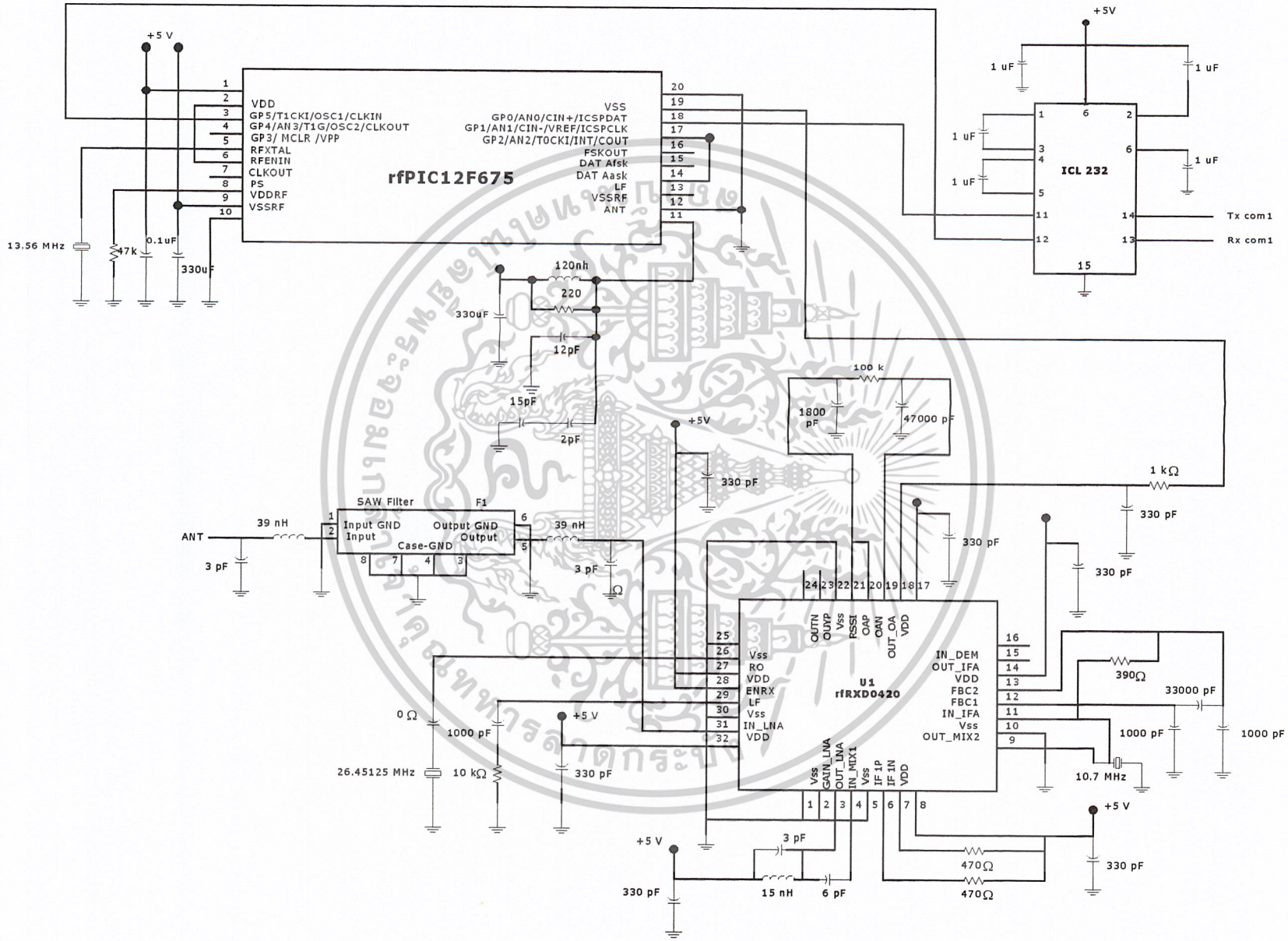
051	ESPRESSO	25.-
052	MOCHA	35.-
053	LATTE	35.-
054	CAPUCCINO	35.-
055	ICE ESPRESSO	30.-
056	ICE MOCHA	40.-
057	ICE LATTE	40.-
058	ICE CAPUCCINO	40.-
059	FRAPPE ESPRESSO	35.-
060	FRAPPE MOCHA	45.-
061	FRAPPE LATTE	45.-
062	FRAPPE CAPUCCINO	45.-

DRINK

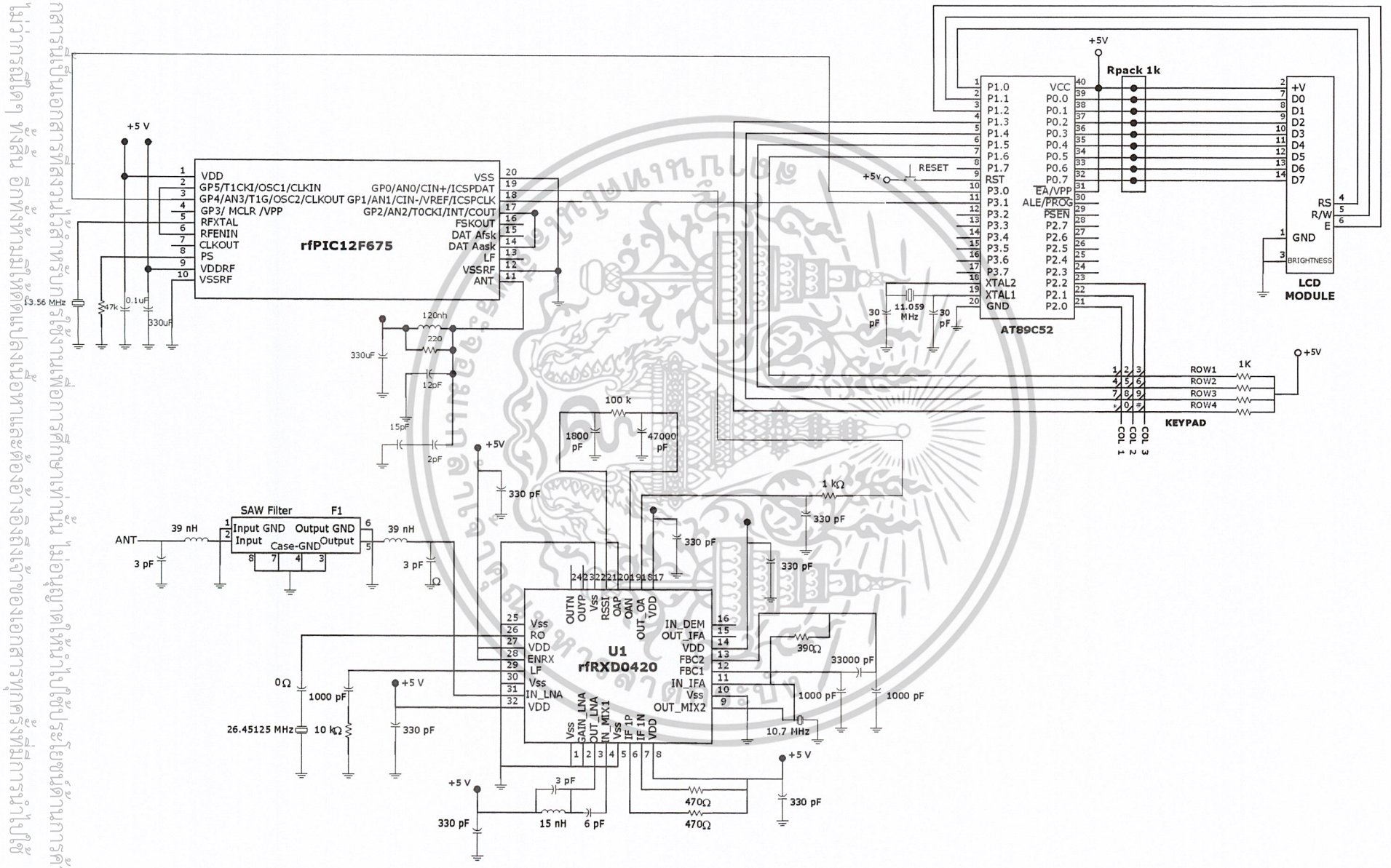
071	HOT TEA	20.-
072	ICE THAI TEA	20.-
073	ICE THAI TEA FRAPPE	25.-
074	ICE LEMON TEA	20.-
075	ICE CHOCOLATE	20.-
076	ICE CHOCOLATE FRAPPE	25.-
077	HOT FRESH MILK	20.-
078	ICE FRESH MILK	25.-
079	ORANGE JUICE	20.-
080	ORANGE JUICE FRAPPE	25.-

081	LEMON JUICE	20.-
082	LEMON JUICE FRAPPE	25.-
083	GUAVA MIXED DRINK	15.-
084	COKE	10.-
085	ICE CUBE / JUG	10.-
086	ICE CUBE / GLASS	2.-
087	SINGHA BEER	65.-
088	HEINEKEN BEER	75.-

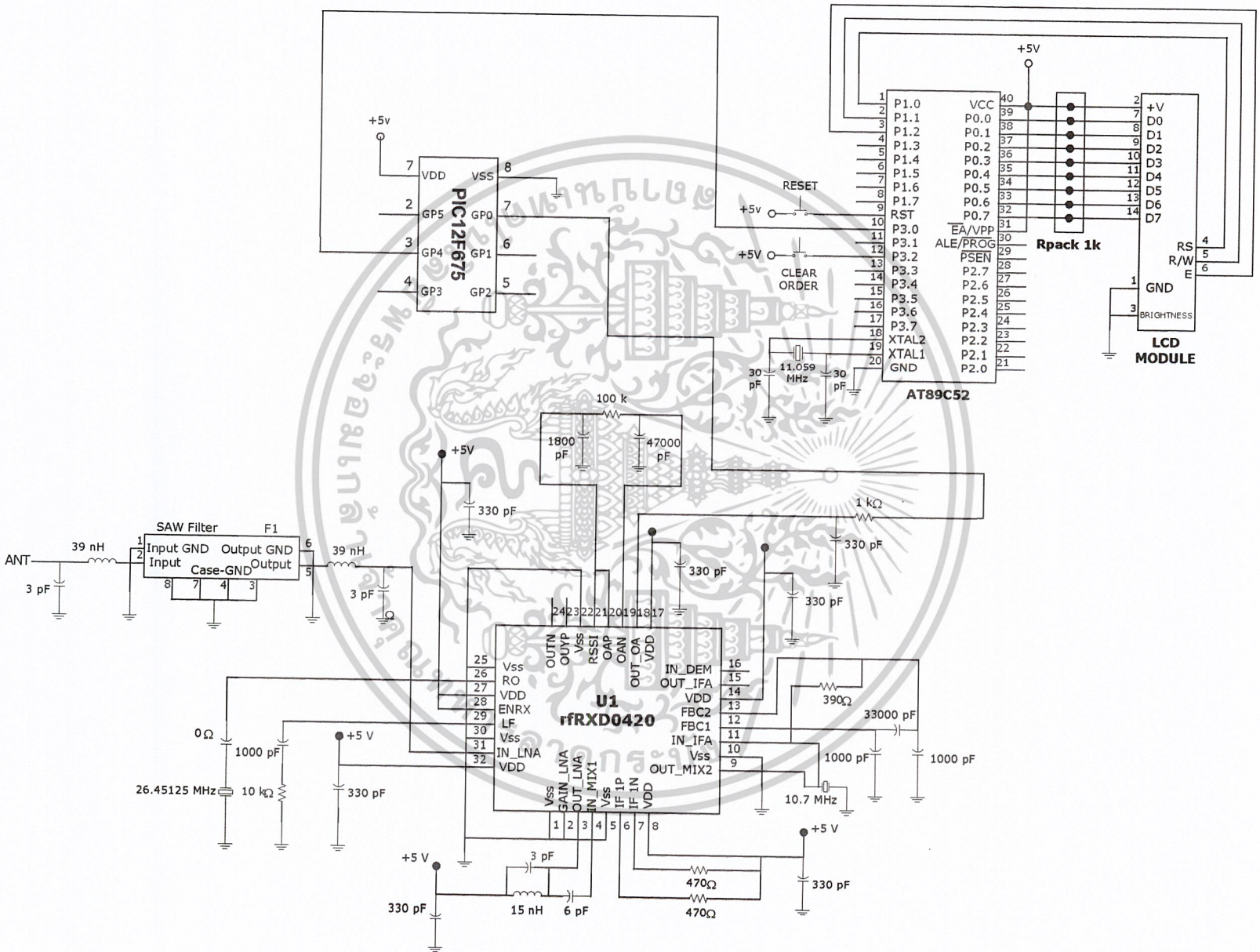
รูปวงจรรวม



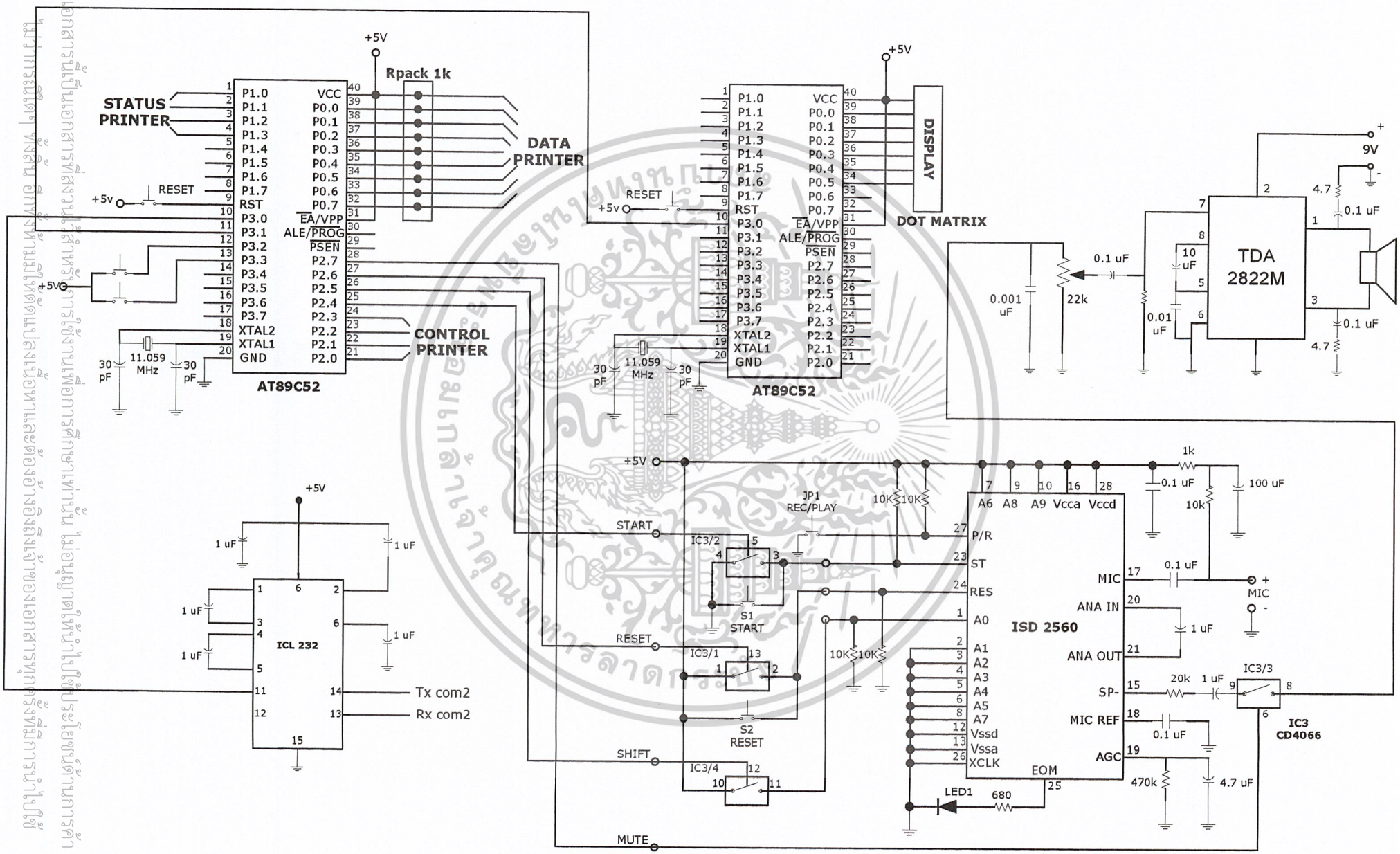
รูปวงจรรวมของส่วนควบคุม



รูปวงจรรวมของส่วนสั่งอาหาร



รูปวงจรรวมของส่วนครัว



รูปวงจรรวมของส่วนจัดเก็บ

เดต้าชีท

MICROCHIP

PIC12F629/675

8-Pin FLASH-Based 8-Bit CMOS Microcontroller

High Performance RISC CPU:

- Only 35 instructions to learn
  - All single cycle instructions except branches
- Operating speed:
  - DC - 20 MHz oscillator/clock input
  - DC - 200 ns instruction cycle
- Interrupt capability
- 8-level deep hardware stack
- Direct, Indirect, and Relative Addressing modes

Special Microcontroller Features:

- Internal and external oscillator options
  - Precision Internal 4 MHz oscillator factory calibrated to ±1%
  - External Oscillator support for crystals and resonators
  - 5 µs wake-up from SLEEP, 3.0V, typical
- Power saving SLEEP mode
- Wide operating voltage range - 2.0V to 5.5V
- Industrial and Extended temperature range
- Low power Power-on Reset (POR)
- Power-up Timer (PWRT) and Oscillator Start-up Timer (OST)
- Brown-out Detect (BOD)
- Watchdog Timer (WDT) with independent oscillator for reliable operation
- Multiplexed $\overline{MCLR}$/Input-pin
- Interrupt-on-pin change
- Individual programmable weak pull-ups
- Programmable code protection
- High Endurance FLASH/EEPROM Cell
  - 100,000 write FLASH endurance
  - 1,000,000 write EEPROM endurance
  - FLASH/Data EEPROM Retention: > 40 years

Low Power Features:

- Standby Current:
  - 1 nA @ 2.0V, typical
- Operating Current:
  - 8.5 µA @ 32 kHz, 2.0V, typical
  - 100 µA @ 1 MHz, 2.0V, typical
- Watchdog Timer Current
  - 300 nA @ 2.0V, typical
- Timer1 oscillator current:
  - 4 µA @ 32 kHz, 2.0V, typical

Peripheral Features:

- 6 I/O pins with individual direction control
- High current sink/source for direct LED drive
- Analog comparator module with:
  - One analog comparator
  - Programmable on-chip comparator voltage reference (CV_{REF}) module
  - Programmable input multiplexing from device inputs
  - Comparator output is externally accessible
- Analog-to-Digital Converter module (PIC12F675):
  - 10-bit resolution
  - Programmable 4-channel input
  - Voltage reference input
- Timer0: 8-bit timer/counter with 8-bit programmable prescaler
- Enhanced Timer1:
  - 16-bit timer/counter with prescaler
  - External Gate Input mode
  - Option to use OSC1 and OSC2 in LP mode as Timer1 oscillator, if INTOSC mode selected
- In-Circuit Serial Programming™ (ICSP™) via two pins

Device	Program Memory	Data Memory		I/O	10-bit A/D (ch)	Comparators	Timers 8/16-bit
	FLASH (words)	SRAM (bytes)	EEPROM (bytes)				
PIC12F629	1024	64	128	6	–	1	1/1
PIC12F675	1024	64	128	6	4	1	1/1

* 8-bit, 8-pin devices protected by Microchip's Low Pin Count Patent: U.S. Patent No. 5,847,450. Additional U.S. and foreign patents and applications may be issued or pending.

2.2.2 SPECIAL FUNCTION REGISTERS

The Special Function registers are registers used by the CPU and peripheral functions for controlling the desired operation of the device (see Table 2-1). These registers are static RAM.

The special registers can be classified into two sets: core and peripheral. The Special Function registers associated with the "core" are described in this section. Those related to the operation of the peripheral features are described in the section of that peripheral feature.

FIGURE 2-2: DATA MEMORY MAP OF THE PIC12F629/675

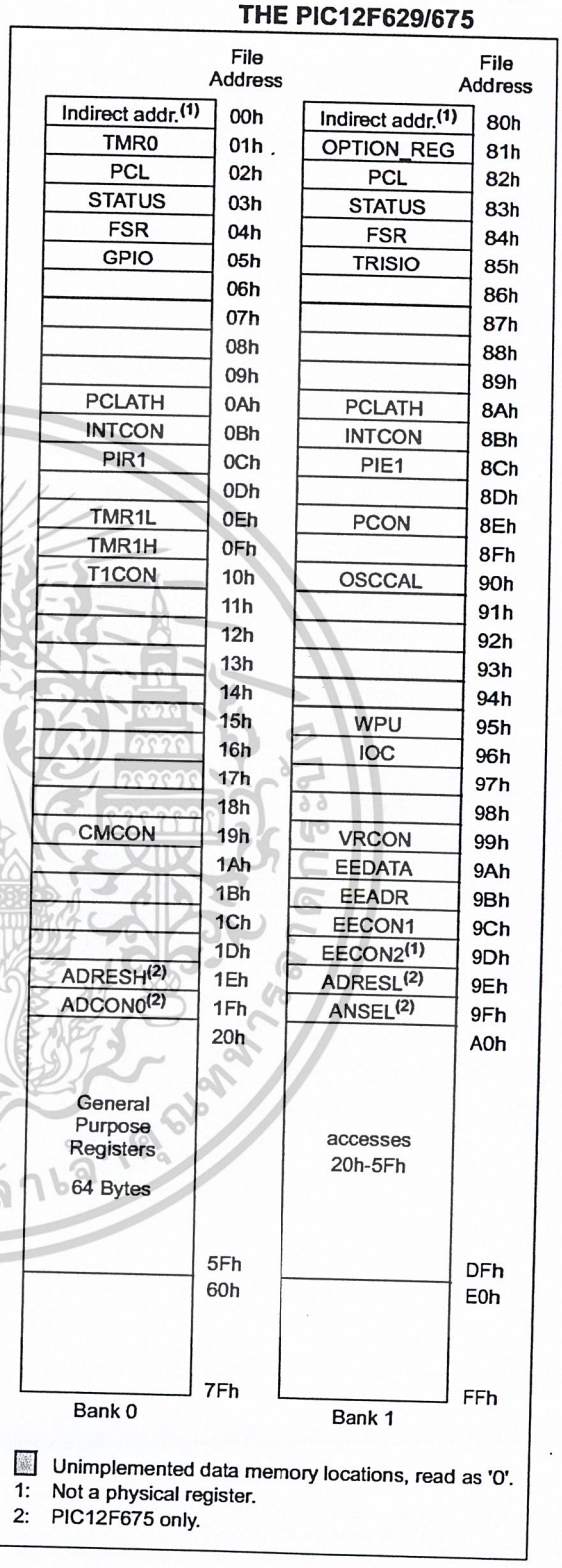


TABLE 2-1: SPECIAL FUNCTION REGISTERS SUMMARY

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOD	Page
Bank 0											
00h	INDF(1)	Addressing this Location uses Contents of FSR to Address Data Memory								0000 0000	18,59
01h	TMR0	Timer0 Module's Register								xxxx xxxx	27
02h	PCL	Program Counter's (PC) Least Significant Byte								0000 0000	17
03h	STATUS	IRP(2)	RP1(2)	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	11
04h	FSR	Indirect Data Memory Address Pointer								xxxx xxxx	18
05h	GPIO	—	—	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0	--xx xxxx	19
06h	—	Unimplemented								—	—
07h	—	Unimplemented								—	—
08h	—	Unimplemented								—	—
09h	—	Unimplemented								—	—
0Ah	PCLATH	—	—	—	Write Buffer for Upper 5 bits of Program Counter					---0 0000	17
0Bh	INTCON	GIE	PEIE	T0IE	INTE	GPIE	T0IF	INTF	GPIF	0000 0000	13
0Ch	PIR1	EEIF	ADIF	—	—	CMIF	—	—	TMR1IF	00-- 0--0	15
0Dh	—	Unimplemented								—	—
0Eh	TMR1L	Holding Register for the Least Significant Byte of the 16-bit Timer1								xxxx xxxx	30
0Fh	TMR1H	Holding Register for the Most Significant Byte of the 16-bit Timer1								xxxx xxxx	30
10h	T1CON	—	TMR1GE	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{T1SYNC}$	TMR1CS	TMR1ON	-000 0000	32
11h	—	Unimplemented								—	—
12h	—	Unimplemented								—	—
13h	—	Unimplemented								—	—
14h	—	Unimplemented								—	—
15h	—	Unimplemented								—	—
16h	—	Unimplemented								—	—
17h	—	Unimplemented								—	—
18h	—	Unimplemented								—	—
19h	CMCON	—	COUT	—	CINV	CIS	CM2	CM1	CM0	-0-0 0000	35
1Ah	—	Unimplemented								—	—
1Bh	—	Unimplemented								—	—
1Ch	—	Unimplemented								—	—
1Dh	—	Unimplemented								—	—
1Eh	ADRESH(3)	Most Significant 8 bits of the Left Shifted A/D Result or 2 bits of the Right Shifted Result								xxxx xxxx	42
1Fh	ADCON0(3)	ADFM	VCFG	—	—	CHS1	CHS0	GO/$\overline{DONE}$	ADON	00-- 0000	43,59

Legend: — = unimplemented locations read as '0', u = unchanged, x = unknown, q = value depends on condition, shaded = unimplemented

Note 1: This is not a physical register.
2: These bits are reserved and should always be maintained as '0'.
3: PIC12F675 only.

TABLE 2-1: SPECIAL FUNCTION REGISTERS SUMMARY (CONTINUED)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOD	Page
Bank 1											
80h	INDF(1)	Addressing this Location uses Contents of FSR to Address Data Memory								0000 0000	18,59
81h	OPTION_REG	$\overline{GPPU}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	12,28
82h	PCL	Program Counter's (PC) Least Significant Byte								0000 0000	17
83h	STATUS	IRP(2)	RP1(2)	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	11
84h	FSR	Indirect Data Memory Address Pointer								xxxx xxxx	18
85h	TRISIO	—	—	TRISIO5	TRISIO4	TRISIO3	TRISIO2	TRISIO1	TRISIO0	--11 1111	19
86h	—	Unimplemented								—	—
87h	—	Unimplemented								—	—
88h	—	Unimplemented								—	—
89h	—	Unimplemented								—	—
8Ah	PCLATH	—	—	—	Write Buffer for Upper 5 bits of Program Counter					---0 0000	17
8Bh	INTCON	GIE	PEIE	T0IE	INTE	GPIE	T0IF	INTF	GPIF	0000 0000	13
8Ch	PIE1	EEIE	ADIE	—	—	CMIE	—	—	TMR1IE	00-- 0--0	14
8Dh	—	Unimplemented								—	—
8Eh	PCON	—	—	—	—	—	—	$\overline{POR}$	$\overline{BOD}$	---- --0x	16
8Fh	—	Unimplemented								—	—
90h	OSCCAL	CAL5	CAL4	CAL3	CAL2	CAL1	CAL0	—	—	1000 00--	16
91h	—	Unimplemented								—	—
92h	—	Unimplemented								—	—
93h	—	Unimplemented								—	—
94h	—	Unimplemented								—	—
95h	WPU	—	—	WPU5	WPU4	—	WPU2	WPU1	WPU0	--11 -111	20
96h	IOC	—	—	IOC5	IOC4	IOC3	IOC2	IOC1	IOC0	--00 0000	21
97h	—	Unimplemented								—	—
98h	—	Unimplemented								—	—
99h	VRCON	VREN	—	VRR	—	VR3	VR2	VR1	VR0	0-0- 0000	40
9Ah	EEDATA	Data EEPROM Data Register								0000 0000	47
9Bh	EEADR	—	Data EEPROM Address Register							-000 0000	47
9Ch	EECON1	—	—	—	—	WRERR	WREN	WR	RD	---- x000	48
9Dh	EECON2(1)	EEPROM Control Register 2								---- ----	48
9Eh	ADRESL(3)	Least Significant 2 bits of the Left Shifted A/D Result of 8 bits or the Right Shifted Result								xxxx xxxx	42
9Fh	ANSEL(3)	—	ADCS2	ADCS1	ADCS0	ANS3	ANS2	ANS1	ANS0	-000 1111	44,59

Legend: — = unimplemented locations read as '0', u = unchanged, x = unknown, q = value depends on condition, shaded = unimplemented

Note 1: This is not a physical register.
2: These bits are reserved and should always be maintained as '0'.
3: PIC12F675 only.

10.0 INSTRUCTION SET SUMMARY

The PIC12F629/675 instruction set is highly orthogonal and is comprised of three basic categories:

- **Byte-oriented** operations
- **Bit-oriented** operations
- **Literal and control** operations

Each PIC12F629/675 instruction is a 14-bit word divided into an **opcode,** which specifies the instruction type, and one or more **operands,** which further specify the operation of the instruction. The formats for each of the categories is presented in Figure 10-1, while the various opcode fields are summarized in Table 10-1.

Table 10-2 lists the instructions recognized by the MPASM™ assembler. A complete description of each instruction is also available in the PICmicro™ Mid-Range Reference Manual (DS33023).

For **byte-oriented** instructions, 'f' represents a file register designator and 'd' represents a destination designator. The file register designator specifies which file register is to be used by the instruction.

The destination designator specifies where the result of the operation is to be placed. If 'd' is zero, the result is placed in the W register. If 'd' is one, the result is placed in the file register specified in the instruction.

For **bit-oriented** instructions, 'b' represents a bit field designator, which selects the bit affected by the operation, while 'f' represents the address of the file in which the bit is located.

For **literal and control** operations, 'k' represents an 8-bit or 11-bit constant, or literal value.

One instruction cycle consists of four oscillator periods; for an oscillator frequency of 4 MHz, this gives a normal instruction execution time of 1 µs. All instructions are executed within a single instruction cycle, unless a conditional test is true, or the program counter is changed as a result of an instruction. When this occurs, the execution takes two instruction cycles, with the second cycle executed as a NOP.

Note:	To maintain upward compatibility with future products, do not use the OPTION and TRISIO instructions.

All instruction examples use the format '0xhh' to represent a hexadecimal number, where 'h' signifies a hexadecimal digit.

10.1 READ-MODIFY-WRITE OPERATIONS

Any instruction that specifies a file register as part of the instruction performs a Read-Modify-Write (R-M-W) operation. The register is read, the data is modified, and the result is stored according to either the instruction, or the destination designator 'd'. A read operation is performed on a register even if the instruction writes to that register.

For example, a CLRF GPIO instruction will read GPIO, clear all the data bits, then write the result back to GPIO. This example would have the unintended result that the condition that sets the GPIF flag would be cleared.

TABLE 10-1: OPCODE FIELD DESCRIPTIONS

Field	Description
f	Register file address (0x00 to 0x7F)
W	Working register (accumulator)
b	Bit address within an 8-bit file register
k	Literal field, constant data or label
x	Don't care location (= 0 or 1). The assembler will generate code with x = 0. It is the recommended form of use for compatibility with all Microchip software tools.
d	Destination select; d = 0: store result in W, d = 1: store result in file register f. Default is d = 1.
PC	Program Counter
TO	Time-out bit
PD	Power-down bit

FIGURE 10-1: GENERAL FORMAT FOR INSTRUCTIONS

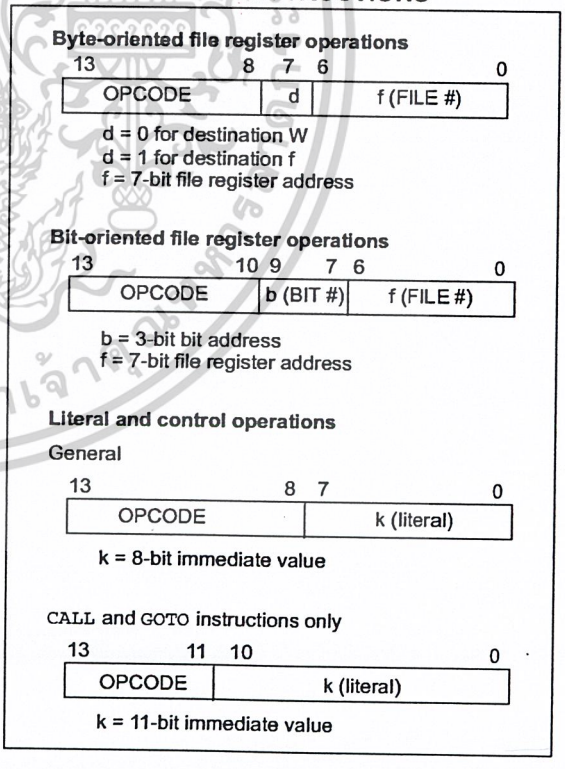


TABLE 10-2: PIC12F629/675 INSTRUCTION SET

Mnemonic, Operands		Description	Cycles	14-Bit Opcode MSb	LSb	Status Affected	Notes
BYTE-ORIENTED FILE REGISTER OPERATIONS							
ADDWF	f, d	Add W and f	1	00 0111 dfff ffff		C,DC,Z	1,2
ANDWF	f, d	AND W with f	1	00 0101 dfff ffff		Z	1,2
CLRF	f	Clear f	1	00 0001 lfff ffff		Z	2
CLRW	-	Clear W	1	00 0001 0xxx xxxx		Z	
COMF	f, d	Complement f	1	00 1001 dfff ffff		Z	1,2
DECF	f, d	Decrement f	1	00 0011 dfff ffff		Z	1,2
DECFSZ	f, d	Decrement f, Skip if 0	1(2)	00 1011 dfff ffff			1,2,3
INCF	f, d	Increment f	1	00 1010 dfff ffff		Z	1,2
INCFSZ	f, d	Increment f, Skip if 0	1(2)	00 1111 dfff ffff			1,2,3
IORWF	f, d	Inclusive OR W with f	1	00 0100 dfff ffff		Z	1,2
MOVF	f, d	Move f	1	00 1000 dfff ffff		Z	1,2
MOVWF	f	Move W to f	1	00 0000 lfff ffff			
NOP	-	No Operation	1	00 0000 0xx0 0000			
RLF	f, d	Rotate Left f through Carry	1	00 1101 dfff ffff		C	1,2
RRF	f, d	Rotate Right f through Carry	1	00 1100 dfff ffff		C	1,2
SUBWF	f, d	Subtract W from f	1	00 0010 dfff ffff		C,DC,Z	1,2
SWAPF	f, d	Swap nibbles in f	1	00 1110 dfff ffff			1,2
XORWF	f, d	Exclusive OR W with f	1	00 0110 dfff ffff		Z	1,2
BIT-ORIENTED FILE REGISTER OPERATIONS							
BCF	f, b	Bit Clear f	1	01 00bb bfff ffff			1,2
BSF	f, b	Bit Set f	1	01 01bb bfff ffff			1,2
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01 10bb bfff ffff			3
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01 11bb bfff ffff			3
LITERAL AND CONTROL OPERATIONS							
ADDLW	k	Add literal and W	1	11 111x kkkk kkkk		C,DC,Z	
ANDLW	k	AND literal with W	1	11 1001 kkkk kkkk		Z	
CALL	k	Call subroutine	2	10 0kkk kkkk kkkk			
CLRWDT	-	Clear Watchdog Timer	1	00 0000 0110 0100		$\overline{TO}$,$\overline{PD}$	
GOTO	k	Go to address	2	10 1kkk kkkk kkkk			
IORLW	k	Inclusive OR literal with W	1	11 1000 kkkk kkkk		Z	
MOVLW	k	Move literal to W	1	11 00xx kkkk kkkk			
RETFIE	-	Return from interrupt	2	00 0000 0000 1001			
RETLW	k	Return with literal in W	2	11 01xx kkkk kkkk			
RETURN	-	Return from Subroutine	2	00 0000 0000 1000			
SLEEP	-	Go into Standby mode	1	00 0000 0110 0011		$\overline{TO}$,$\overline{PD}$	
SUBLW	k	Subtract W from literal	1	11 110x kkkk kkkk		C,DC,Z	
XORLW	k	Exclusive OR literal with W	1	11 1010 kkkk kkkk		Z	

Note 1: When an I/O register is modified as a function of itself (e.g., `MOVF GPIO, 1`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

2: If this instruction is executed on the TMR0 register (and, where applicable, d = 1), the prescaler will be cleared if assigned to the Timer0 module.

3: If Program Counter (PC) is modified, or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a `NOP`.

Note: Additional information on the mid-range instruction set is available in the PICmicro™ Mid-Range MCU Family Reference Manual (DS33023).

10.2 Instruction Descriptions

ADDLW	Add Literal and W
Syntax:	[*label*] ADDLW k
Operands:	0 ≤ k ≤ 255
Operation:	(W) + k → (W)
Status Affected:	C, DC, Z
Description:	The contents of the W register are added to the eight-bit literal 'k' and the result is placed in the W register.

ADDWF	Add W and f
Syntax:	[*label*] ADDWF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(W) + (f) → (destination)
Status Affected:	C, DC, Z
Description:	Add the contents of the W register with register 'f'. If 'd' is 0, the result is stored in the W register. If 'd' is 1, the result is stored back in register 'f'.

ANDLW	AND Literal with W
Syntax:	[*label*] ANDLW k
Operands:	0 ≤ k ≤ 255
Operation:	(W) .AND. (k) → (W)
Status Affected:	Z
Description:	The contents of W register are AND'ed with the eight-bit literal 'k'. The result is placed in the W register.

ANDWF	AND W with f
Syntax:	[*label*] ANDWF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(W) .AND. (f) → (destination)
Status Affected:	Z
Description:	AND the W register with register 'f'. If 'd' is 0, the result is stored in the W register. If 'd' is 1, the result is stored back in register 'f'.

BCF	Bit Clear f
Syntax:	[*label*] BCF f,b
Operands:	0 ≤ f ≤ 127 0 ≤ b ≤ 7
Operation:	0 → (f<b>)
Status Affected:	None
Description:	Bit 'b' in register 'f' is cleared.

BSF	Bit Set f
Syntax:	[*label*] BSF f,b
Operands:	0 ≤ f ≤ 127 0 ≤ b ≤ 7
Operation:	1 → (f<b>)
Status Affected:	None
Description:	Bit 'b' in register 'f' is set.

BTFSS	Bit Test f, Skip if Set
Syntax:	[*label*] BTFSS f,b
Operands:	0 ≤ f ≤ 127 0 ≤ b < 7
Operation:	skip if (f<b>) = 1
Status Affected:	None
Description:	If bit 'b' in register 'f' is '0', the next instruction is executed. If bit 'b' is '1', then the next instruction is discarded and a NOP is executed instead, making this a 2TCY instruction.

BTFSC	Bit Test, Skip if Clear
Syntax:	[*label*] BTFSC f,b
Operands:	0 ≤ f ≤ 127 0 ≤ b ≤ 7
Operation:	skip if (f<b>) = 0
Status Affected:	None
Description:	If bit 'b' in register 'f' is '1', the next instruction is executed. If bit 'b', in register 'f', is '0', the next instruction is discarded, and a NOP is executed instead, making this a 2TCY instruction.

CALL	Call Subroutine
Syntax:	[*label*] CALL k
Operands:	0 ≤ k ≤ 2047
Operation:	(PC)+ 1→ TOS, k → PC<10:0>, (PCLATH<4:3>) → PC<12:11>
Status Affected:	None
Description:	Call Subroutine. First, return address (PC+1) is pushed onto the stack. The eleven-bit immediate address is loaded into PC bits <10:0>. The upper bits of the PC are loaded from PCLATH. CALL is a two-cycle instruction.

CLRWDT	Clear Watchdog Timer
Syntax:	[*label*] CLRWDT
Operands:	None
Operation:	00h → WDT 0 → WDT prescaler, 1 → $\overline{TO}$ 1 → $\overline{PD}$
Status Affected:	$\overline{TO}$, $\overline{PD}$
Description:	CLRWDT instruction resets the Watchdog Timer. It also resets the prescaler of the WDT. STATUS bits $\overline{TO}$ and $\overline{PD}$ are set.

CLRF	Clear f
Syntax:	[*label*] CLRF f
Operands:	0 ≤ f ≤ 127
Operation:	00h → (f) 1 → Z
Status Affected:	Z
Description:	The contents of register 'f' are cleared and the Z bit is set.

COMF	Complement f
Syntax:	[*label*] COMF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	($\bar{f}$) → (destination)
Status Affected:	Z
Description:	The contents of register 'f' are complemented. If 'd' is 0, the result is stored in W. If 'd' is 1, the result is stored back in register 'f'.

CLRW	Clear W
Syntax:	[*label*] CLRW
Operands:	None
Operation:	00h → (W) 1 → Z
Status Affected:	Z
Description:	W register is cleared. Zero bit (Z) is set.

DECF	Decrement f
Syntax:	[*label*] DECF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f) - 1 → (destination)
Status Affected:	Z
Description:	Decrement register 'f'. If 'd' is 0, the result is stored in the W register. If 'd' is 1, the result is stored back in register 'f'.

DECFSZ	Decrement f, Skip if 0
Syntax:	[*label*] DECFSZ f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f) - 1 → (destination); skip if result = 0
Status Affected:	None
Description:	The contents of register 'f' are decremented. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is placed back in register 'f'. If the result is 1, the next instruction is executed. If the result is 0, then a NOP is executed instead, making it a 2TCY instruction.

GOTO	Unconditional Branch
Syntax:	[*label*] GOTO k
Operands:	0 ≤ k ≤ 2047
Operation:	k → PC<10:0> PCLATH<4:3> → PC<12:11>
Status Affected:	None
Description:	GOTO is an unconditional branch. The eleven-bit immediate value is loaded into PC bits <10:0>. The upper bits of PC are loaded from PCLATH<4:3>. GOTO is a two-cycle instruction.

INCF	Increment f
Syntax:	[*label*] INCF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f) + 1 → (destination)
Status Affected:	Z
Description:	The contents of register 'f' are incremented. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is placed back in register 'f'.

INCFSZ	Increment f, Skip if 0
Syntax:	[*label*] INCFSZ f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f) + 1 → (destination), skip if result = 0
Status Affected:	None
Description:	The contents of register 'f' are incremented. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is placed back in register 'f'. If the result is 1, the next instruction is executed. If the result is 0, a NOP is executed instead, making it a 2TCY instruction.

IORLW	Inclusive OR Literal with W
Syntax:	[*label*] IORLW k
Operands:	0 ≤ k ≤ 255
Operation:	(W) .OR. k → (W)
Status Affected:	Z
Description:	The contents of the W register are OR'ed with the eight-bit literal 'k'. The result is placed in the W register.

IORWF	Inclusive OR W with f
Syntax:	[*label*] IORWF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(W) .OR. (f) → (destination)
Status Affected:	Z
Description:	Inclusive OR the W register with register 'f'. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is placed back in register 'f'.

MOVF	Move f
Syntax:	[*label*] MOVF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f) → (destination)
Status Affected:	Z
Description:	The contents of register f are moved to a destination dependant upon the status of d. If d = 0, destination is W register. If d = 1, the destination is file register f itself. d = 1 is useful to test a file register, since status flag Z is affected.

NOP	No Operation
Syntax:	[*label*] NOP
Operands:	None
Operation:	No operation
Status Affected:	None
Description:	No operation.

MOVLW	Move Literal to W
Syntax:	[*label*] MOVLW k
Operands:	0 ≤ k ≤ 255
Operation:	k → (W)
Status Affected:	None
Description:	The eight-bit literal 'k' is loaded into W register. The don't cares will assemble as 0's.

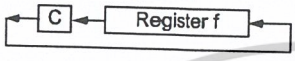
MOVWF	Move W to f
Syntax:	[*label*] MOVWF f
Operands:	0 ≤ f ≤ 127
Operation:	(W) → (f)
Status Affected:	None
Description:	Move data from W register to register 'f'.

RETFIE	Return from Interrupt
Syntax:	[*label*] RETFIE
Operands:	None
Operation:	TOS → PC, 1 → GIE
Status Affected:	None

RETLW	Return with Literal in W
Syntax:	[*label*] RETLW k
Operands:	0 ≤ k ≤ 255
Operation:	k → (W); TOS → PC
Status Affected:	None
Description:	The W register is loaded with the eight-bit literal 'k'. The program counter is loaded from the top of the stack (the return address). This is a two-cycle instruction.

RLF — **Rotate Left f through Carry**

Syntax:	[*label*] RLF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	See description below
Status Affected:	C
Description:	The contents of register 'f' are rotated one bit to the left through the Carry Flag. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is stored back in register 'f'.



SLEEP

Syntax:	[*label*] SLEEP
Operands:	None
Operation:	00h → WDT, 0 → WDT prescaler, 1 → $\overline{TO}$, 0 → $\overline{PD}$
Status Affected:	$\overline{TO}$, $\overline{PD}$
Description:	The power-down STATUS bit, $\overline{PD}$ is cleared. Time-out STATUS bit, $\overline{TO}$ is set. Watchdog Timer and its prescaler are cleared. The processor is put into SLEEP mode with the oscillator stopped.

RETURN — **Return from Subroutine**

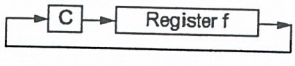
Syntax:	[*label*] RETURN
Operands:	None
Operation:	TOS → PC
Status Affected:	None
Description:	Return from subroutine. The stack is POPed and the top of the stack (TOS) is loaded into the program counter. This is a two-cycle instruction.

SUBLW — **Subtract W from Literal**

Syntax:	[*label*] SUBLW k
Operands:	$0 \le k \le 255$
Operation:	k - (W) → (W)
Status Affected:	C, DC, Z
Description:	The W register is subtracted (2's complement method) from the eight-bit literal 'k'. The result is placed in the W register.

RRF — **Rotate Right f through Carry**

Syntax:	[*label*] RRF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	See description below
Status Affected:	C
Description:	The contents of register 'f' are rotated one bit to the right through the Carry Flag. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is placed back in register 'f'.



SUBWF — **Subtract W from f**

Syntax:	[*label*] SUBWF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(f) - (W) → (destination)
Status Affected:	C, DC, Z
Description:	Subtract (2's complement method) W register from register 'f'. If 'd' is 0, the result is stored in the W register. If 'd' is 1, the result is stored back in register 'f'.

SWAPF	Swap Nibbles in f
Syntax:	[*label*] SWAPF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f<3:0>) → (destination<7:4>), (f<7:4>) → (destination<3:0>)
Status Affected:	None
Description:	The upper and lower nibbles of register 'f' are exchanged. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is placed in register 'f'.

XORWF	Exclusive OR W with f
Syntax:	[*label*] XORWF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(W) .XOR. (f) → (destination)
Status Affected:	Z
Description:	Exclusive OR the contents of the W register with register 'f'. If 'd' is 0, the result is stored in the W register. If 'd' is 1, the result is stored back in register 'f'.

XORLW	Exclusive OR Literal with W
Syntax:	[*label*] XORLW k
Operands:	0 ≤ k ≤ 255
Operation:	(W) .XOR. k → (W)
Status Affected:	Z
Description:	The contents of the W register are XOR'ed with the eight-bit literal 'k'. The result is placed in the W register.

rfRXD0420/0920

UHF ASK/FSK/FM Receiver

Features:

- Low cost single conversion superheterodyne receiver architecture
- Compatible with rfPIC™ and rfHCS series of RF transmitters
- Easy interface to PICmicro® microcontroller (MCU) and KEELOQ® decoders
- VCO phase locked to quartz crystal reference:
  - Narrow receiver bandwidth
  - Maximizes range and interference immunity
- Selectable LNA gain control for improved dynamic range
- Selectable IF bandwidth via external ceramic IF filter
- Received Signal Strength Indicator (RSSI) for signal strength indication (FSK, FM) and ASK demodulation
- FSK/FM quadrature (phase coincidence) detector demodulator
- 32-Lead LQFP package

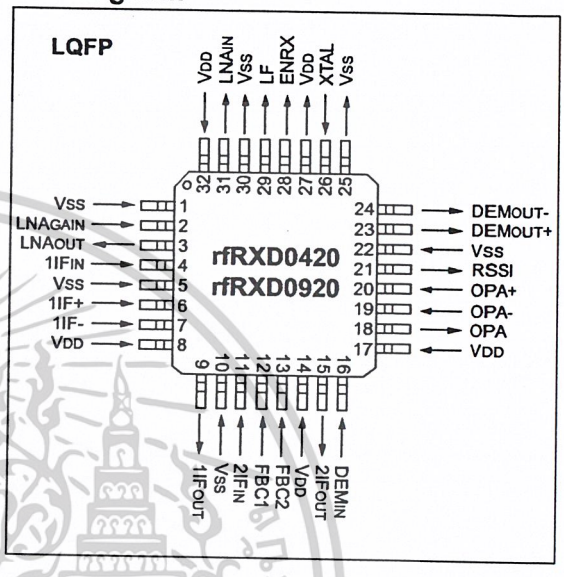
UHF ASK/FSK Receiver:

- Single frequency receiver set by crystal frequency
- Receive frequency range:

Device	Frequency Range
rfRXD0420	300 MHz to 450 MHz
rfRXD0920	800 MHz to 930 MHz

- Maximum data rate:
  - ASK: 80 Kbps NRZ
  - FSK: 40 Kbps NRZ
- IF frequency range: 455 kHz to 21.4 MHz
- RSSI range: 70 dB
- Frequency deviation range: ±5 kHz to ±120 kHz
- Maximum FM modulation frequency: 15 kHz

Pin Diagram:



Applications:

- Wireless remote command and control
- Wireless security systems
- Remote Keyless Entry (RKE)
- Low power telemetry
- Low power FM receiver
- Home automation
- Remote sensing

Bi-CMOS Technology:

- Wide operating voltage range
- Low current consumption in Active and Standby modes
  - rfRXD0420
    - 8.2 mA (typical, LNA High Gain mode)
    - <100 nA standby
  - rfRXD0920
    - 9.2 mA (typical, LNA High Gain mode)
    - <100 nA standby
- Wide temperature range:
  - Industrial: -40°C to +85°C

MICROCHIP

rfPIC12F675

FLASH-Based Microcontroller with ASK/FSK Transmitter

High Performance RISC CPU:

- Only 35 instructions to learn
- All single cycle instructions (DC-200 ns), except for program branches which are two-cycle
- Operating speed:
  - 4 MHz precision internal oscillator
  - DC - 20 MHz HS crystal oscillator/clock input
  - DC - 4 MHz external RC oscillator
  - DC - 4 MHz XT crystal oscillator
  - DC - 200 kHz LP crystal oscillator
- Memory
  - 1024 x 14 words of FLASH program memory
  - 64 x 8 bytes of data memory (SRAM)
  - 128 x 8 bytes of EEPROM data memory
- 16 special function hardware registers
- Interrupt capability
- 8-level deep hardware stack
- Direct, Indirect and Relative Addressing modes
- Industrial and Extended temperature range
- Wide operating voltage range from 2.0 – 5.5V
- Low power consumption: (typical values @ 3.0V)
  - 14 mA transmitting full power RF (+ 6 dBm)
  - 4 mA transmitting low power RF (-15 dBm)
  - 500 µA, 4.0 MHz INTOSC, RF off
  - 15 µA, 32 kHz LP, RF off
  - 0.1 µA standby current, RF off

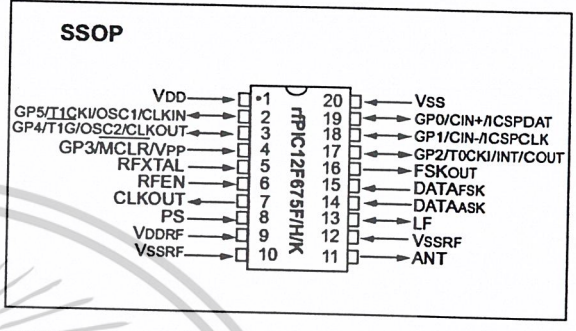
Peripheral Features:

- 6 I/O pins with individual direction control
- High current sink/source for direct LED drive
- Analog comparator: 16 internal reference levels
- Analog-to-Digital Converter: 10 bits, 4 channels
- Timer0: 8-bit timer/counter with 8-bit prescaler
- Timer1: 16-bit timer/counter with 3-bit prescaler
- 32kHz Timer1 oscillator while in INTOSC mode

Device	Frequency	Modulation
rfPIC12F675K	290-350 MHz	ASK/FSK
rfPIC12F675F	380-450 MHz	ASK/FSK
rfPIC12F675H	850-930 MHz	ASK/FSK

Pin Diagram:

SSOP



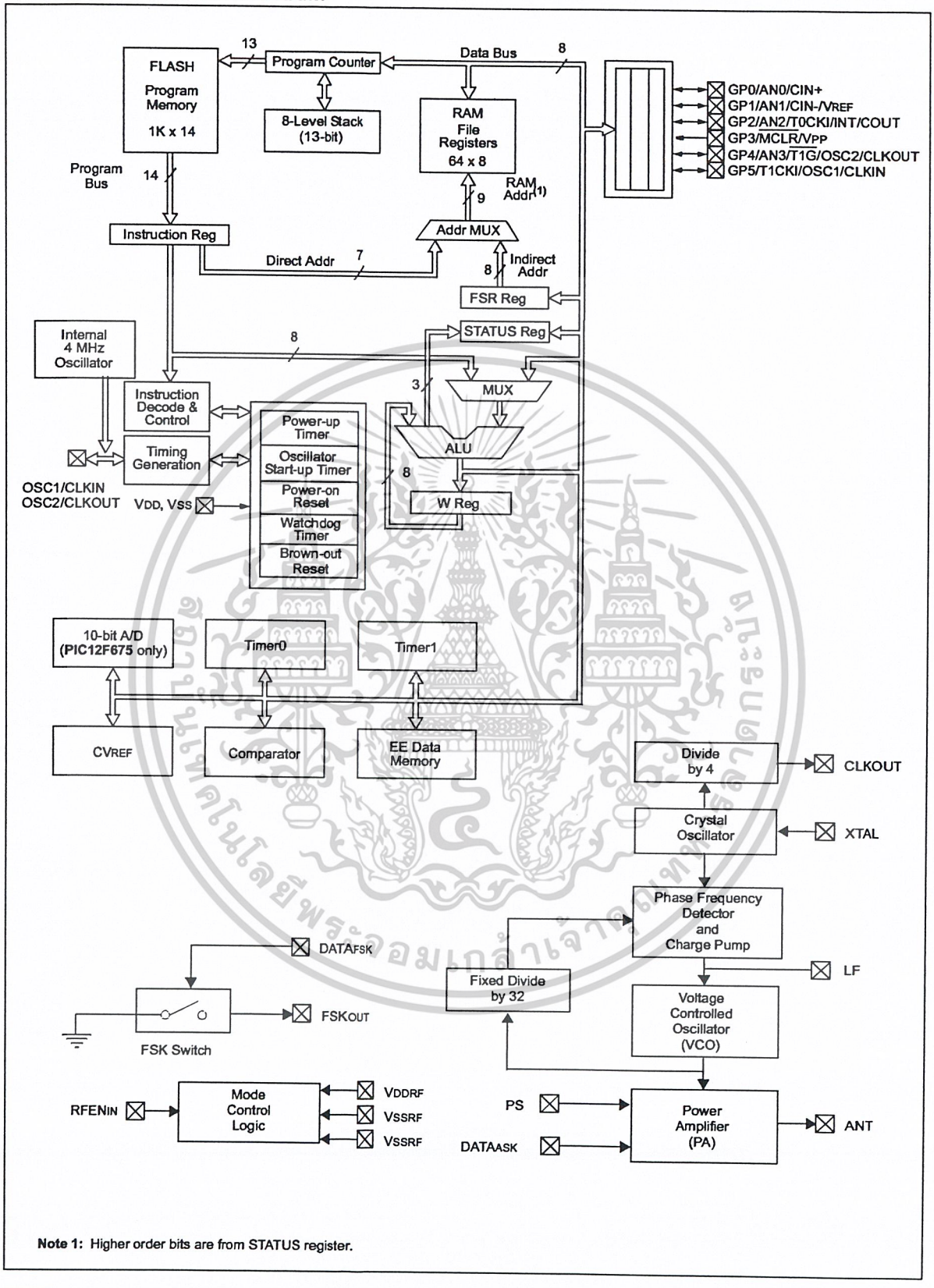
UHF ASK/FSK Transmitter:

- Integrated crystal oscillator, VCO, loop filter and power amp for minimum external components
- ASK data rate: 0 – 40 Kbps
- FSK data rate: 0 – 20 Kbps by crystal pulling
- Output power: +6 dBm to -15 dBm in 4 steps
- Adjustable transmitter power consumption
- RF Frequency set by crystal multiplied by 32
- VCO phase locked to quartz crystal reference; allows narrow band receivers to be used to maximize range and interference immunity
- Crystal frequency divide by 4 available (CLKOUT)
- Used in applications conforming to US FCC Part 15.231 and European EN 300 220 regulations

Applications:

- Automotive Remote Keyless Entry (RKE) systems
- Automotive alarm systems
- Community gate and garage door openers
- Burglar alarm systems
- Building access
- Low power telemetry
- Meter Reading
- Tire pressure senors
- Wireless sensors

FIGURE 1: BLOCK DIAGRAM:



Note 1: Higher order bits are from STATUS register.

Advance Information

SGS-THOMSON MICROELECTRONICS

TDA2822M

DUAL LOW-VOLTAGE POWER AMPLIFIER

- SUPPLY VOLTAGE DOWN TO 1.8V
- LOW CROSSOVER DISTORSION
- LOW QUIESCENT CURRENT
- BRIDGE OR STEREO CONFIGURATION

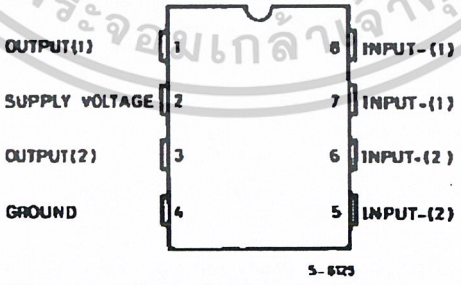
MINIDIP

ORDERING NUMBER : TDA2822M

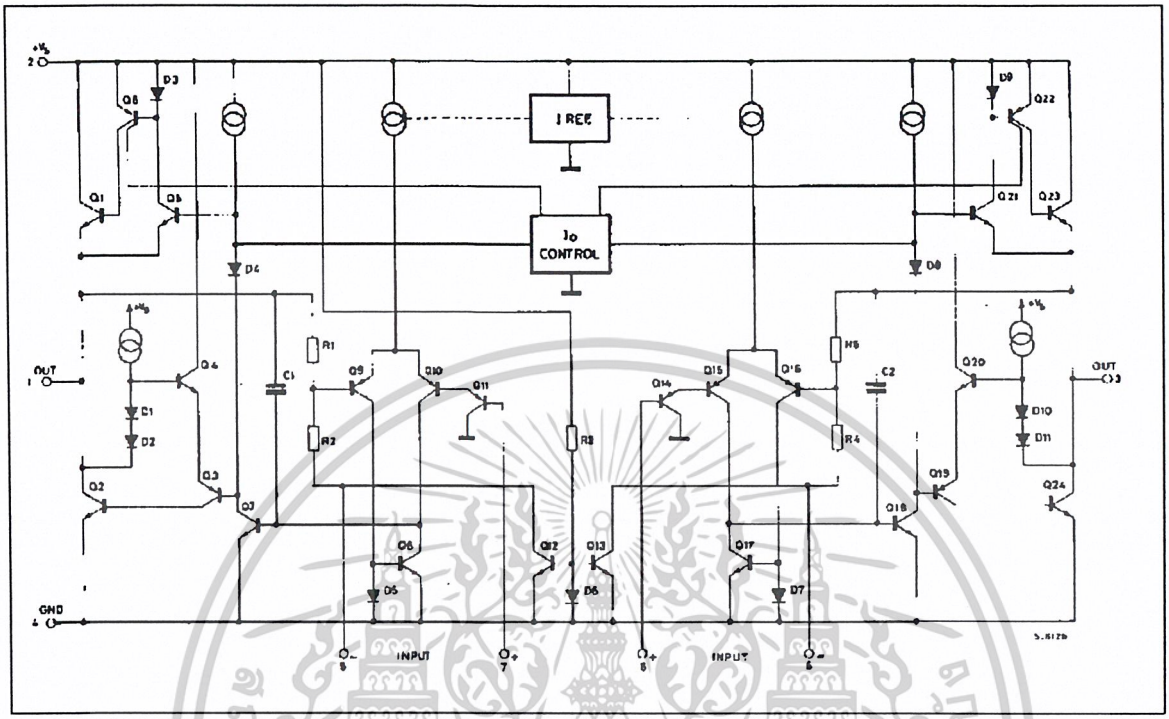
DESCRIPTION

The TDA2822M is a monolithic integrated circuit in 8 lead Minidip package. It is intended for use as dual audio power amplifier in portable cassette players and radios.

PIN CONNECTION (Top view)



SCHEMATIC DIAGRAM



ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_s	Supply Voltage	15	V
I_o	Peak Output Current	1	A
P_{tot}	Total Power Dissipation at T_{amb} = 50 °C at T_{case} = 50 °C	1 1.4	W W
T_{stg}, T_j	Storage and Junction Temperature	– 40, + 150	°C

THERMAL DATA

Symbol	Parameter		Value	Unit
$R_{th\ j\text{-}amb}$	Thermal Resistance Junction-ambient	Max.	100	°C/W
$R_{th\ j\text{-}case}$	Thermal Resistance Junction-pin (4)	Max.	70	°C/W

ELECTRICAL CHARACTERISTICS (V_S = 6V, T_{amb} = 25°C, unless otherwise specified)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
STEREO (test circuit of Figure 1)						
V_s	Supply Voltage		1.8		15	V
V_o	Quiescent Output Voltage	V_s = 3V		2.7 1.2		V V
I_d	Quiescent Drain Current			6	9	mA
I_b	Input Bias Current			100		nA
P_o	Output Power (each channel) (f = 1kHz, d = 10%)	R_L = 32Ω V_S = 9V V_S = 6V V_S = 4.5V V_S = 3V V_S = 2V R_L = 16Ω V_S = 6V R_L = 8Ω V_S = 9V V_S = 6V R_L = 4Ω V_S = 6V V_S = 4.5V V_S = 3V	 90 15 170 300 450	300 120 60 20 5 220 1000 380 650 320 110		mW
d	Distortion (f = 1kHz)	R_L = 32Ω P_o = 40mW R_L = 16Ω P_o = 75mW R_L = 8Ω P_o = 150mW		0.2 0.2 0.2		% % %
G_v	Closed Loop Voltage Gain	f = 1kHz	36	39	41	dB
ΔG_v	Channel Balance				± 1	dB
R_i	Input Resistance	f = 1kHz	100			kΩ
e_N	Total Input Noise	R_s = 10kΩ B = Curve A B = 22Hz to 22kHz		2 2.5		μV μV
SVR	Supply Voltage Rejection	f = 100Hz, C1 = C2 = 100μF	24	30		dB
C_s	Channel Separation	f = 1kHz		50		dB
BRIDGE (test circuit of Figure 2)						
V_s	Supply Voltage		1.8		15	V
I_d	Quiescent Drain Current	R_L = ∞		6	9	mA
V_{os}	Output Offset Voltage (between the outputs)	R_L = 8Ω			± 50	mV
I_b	Input Bias Current			100		nA
P_o	Output Power (f = 1kHz, d = 10%)	R_L = 32Ω V_S = 9V V_S = 6V V_S = 4.5V V_S = 3V V_S = 2V R_L = 16Ω V_S = 9V V_S = 6V V_S = 3V R_L = 8Ω V_S = 6V V_S = 4.5V V_S = 3V R_L = 4Ω V_S = 4.5V V_S = 3V V_S = 2V	 320 50 900 200 	1000 400 200 65 8 2000 800 120 1350 700 220 1000 350 80		mW
d	Distortion	P_o = 0.5W, R_L = 8Ω, f = 1kHz		0.2		%
G_v	Closed Loop Voltage Gain	f = 1kHz		39		dB
R_i	Input Resistance	f = 1kHz	100			kΩ
e_N	Total Input Noise	R_s = 10kΩ B = Curve A B = 22Hz to 22kHz		2.5 3		μV μV
SVR	Supply Voltage Rejection	f = 100Hz		40		dB
B	Power Bandwidth (–3dB)	R_L = 8Ω, P_o = 1W		120		kHz

Figure 1 : Test Circuit (Stereo)

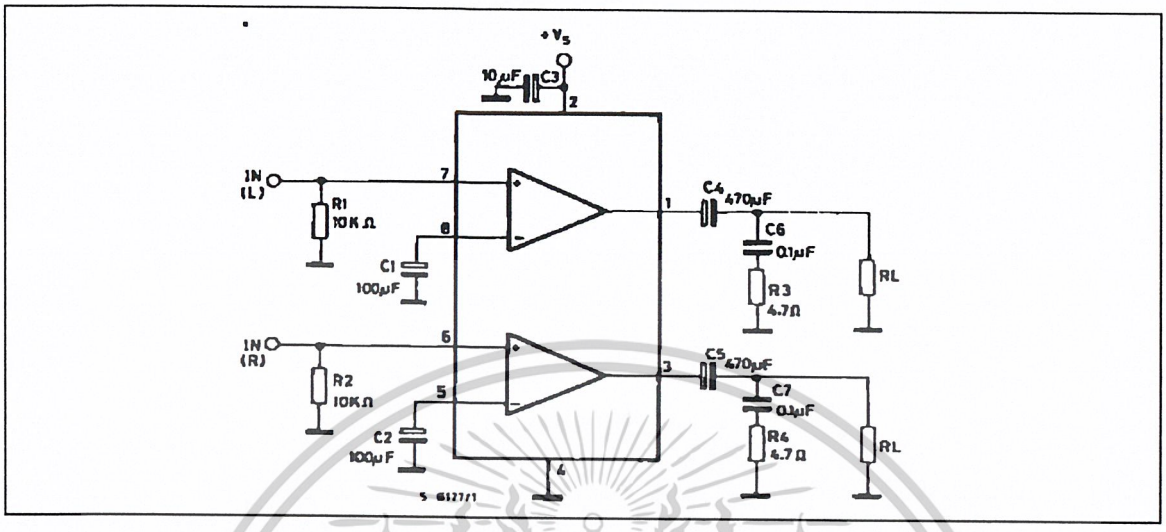


Figure 2 : Test Circuit (Bridge)

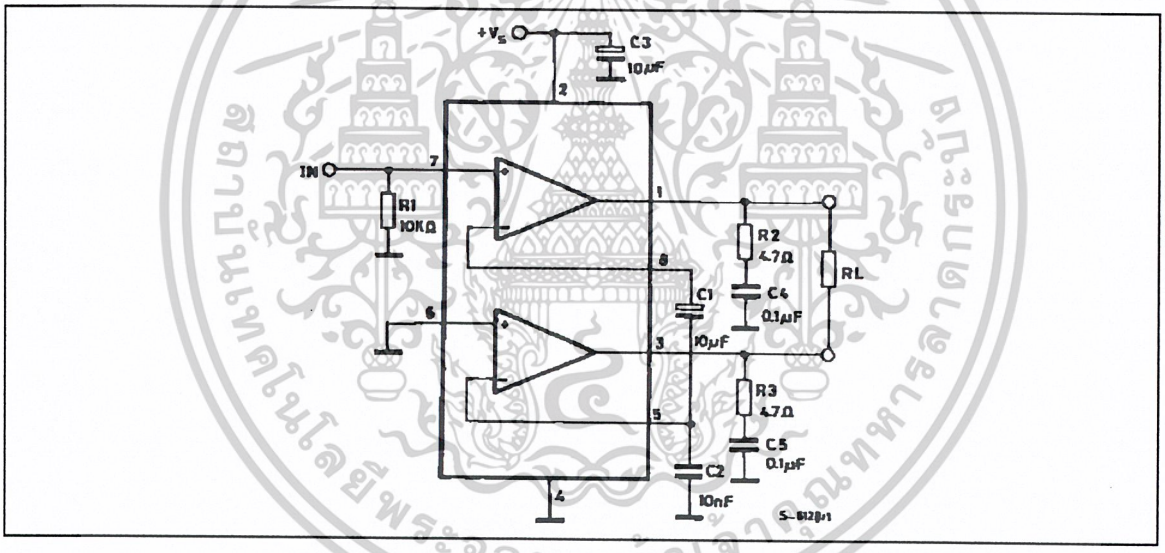


Figure 3 : P.C. Board and Components Layout of the Circuit of Figure 1

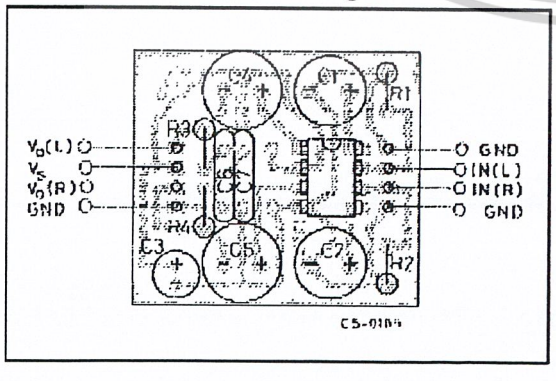
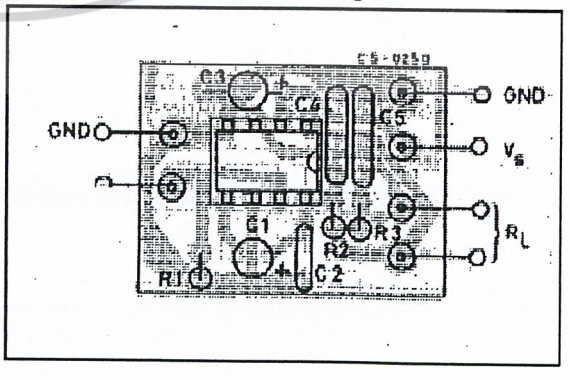


Figure 4 : P.C. Board and Components Layout of the Circuit of Figure 2



Winbond Electronics Corp.

1. GENERAL DESCRIPTION

Winbond's ISD2500 ChipCorder® Series provide high-quality, single-chip, Record/Playback solutions for 60- to 120-second messaging applications. The CMOS devices include an on-chip oscillator, microphone preamplifier, automatic gain control, antialiasing filter, smoothing filter, speaker amplifier, and high density multi-level storage array. In addition, the ISD2500 is microcontroller compatible, allowing complex messaging and addressing to be achieved. Recordings are stored into on-chip nonvolatile memory cells, providing zero-power message storage. This unique, single-chip solution is made possible through Winbond's patented multilevel storage technology. Voice and audio signals are stored directly into memory in their natural form, providing high-quality, solid-state voice reproduction.

2. FEATURES

- Easy-to-use single-chip, voice record/playback solution
- High-quality, natural voice/audio reproduction
- Single-chip with duration of 60, 75, 90, or 120 seconds.
- Manual switch or microcontroller compatible
- Playback can be edge- or level-activated
- Directly cascadable for longer durations
- Automatic power-down (push-button mode)
  - Standby current 1 µA (typical)
- Zero-power message storage
  - Eliminates battery backup circuits
- Fully addressable to handle multiple messages
- 100-year message retention (typical)
- 100,000 record cycles (typical)
- On-chip clock source
- Programmer support for play-only applications
- Single +5 volt power supply
- Available in die form, PDIP, SOIC and TSOP packaging
- Temperature = die (0°C to +50°C) and package (0°C to +70°C)

intersil

ICL232

June 2001

+5V Powered, Dual RS-232 Transmitter/Receiver

Features

- Meets All RS-232C and V.28 Specifications
- Requires Only Single +5V Power Supply
- Onboard Voltage Doubler/Inverter
- Low Power Consumption
- 2 Drivers
  - ±9V Output Swing for +5V Input
  - 300Ω Power-off Source Impedance
  - Output Current Limiting
  - TTL/CMOS Compatible
  - 30V/μs Maximum Slew Rate
- 2 Receivers
  - ±30V Input Voltage Range
  - 3kΩ to 7kΩ Input Impedance
  - 0.5V Hysteresis to Improve Noise Rejection
- All Critical Parameters are Guaranteed Over the Entire Commercial, Industrial and Military Temperature Ranges

Applications

- Any System Requiring RS-232 Communications Port
  - Computer - Portable and Mainframe
  - Peripheral - Printers and Terminals
  - Portable Instrumentation
  - Modems
- Dataloggers

Description

The ICL232 is a dual RS-232 transmitter/receiver interface circuit that meets all EIA RS-232C and V.28 specifications. It requires a single +5V power supply, and features two onboard charge pump voltage converters which generate +10V and -10V supplies from the 5V supply.

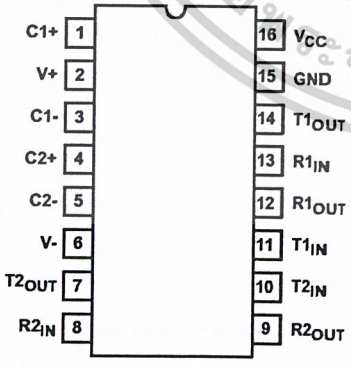
The drivers feature true TTL/CMOS input compatibility, slew-rate-limited output, and 300Ω power-off source impedance. The receivers can handle up to +30V, and have a 3kΩ to 7kΩ input impedance. The receivers also have hysteresis to improve noise rejection.

Ordering Information

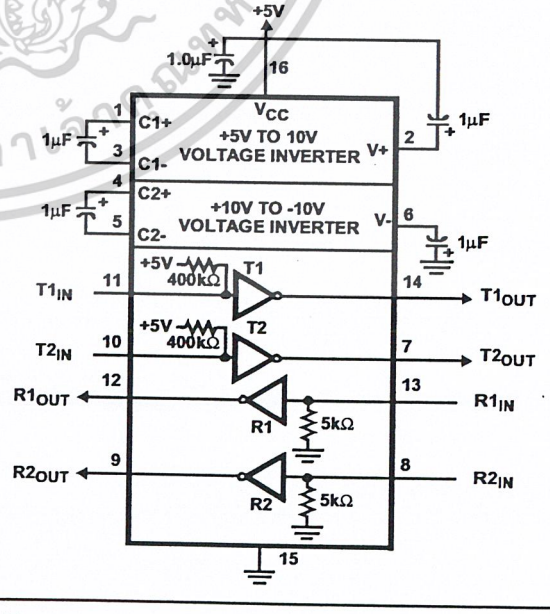
PART NUMBER	TEMP. RANGE (°C)	PACKAGE	PKG. NO.
ICL232CPE	0 to 70	16 Ld PDIP	E16.3
ICL232CBE	0 to 70	16 Ld SOIC	M16.3
ICL232IPE	-40 to 85	16 Ld PDIP	E16.3
ICL232IBE	-40 to 85	16 Ld SOIC	M16.3
ICL232MJE	-55 to 125	16 Ld CERDIP	F16.3

Pinout

ICL232 (PDIP, CERDIP, SOIC)
TOP VIEW



Functional Diagram



CAUTION: These devices are sensitive to electrostatic discharge; follow proper IC Handling Procedures.
1-888-INTERSIL or 321-724-7143 | Intersil (and design) is a registered trademark of Intersil Americas Inc.

File Number **3020.6**

Ordering number: EN 2780B

SANYO

No.2780B

CMOS LSI

LC7932,7932M

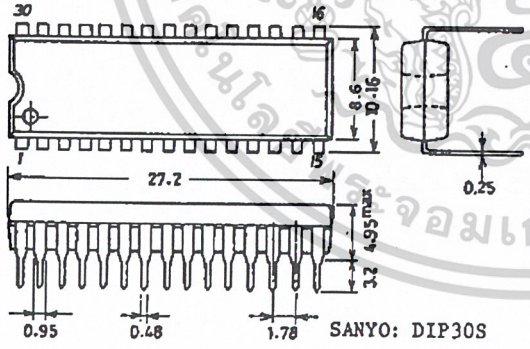
16-Bit LED Driver

The LC7932,7932M are LSIs that contain a 16-bit bidirectional shift register and are capable of direct driving a multiple lighting LED (dot matrix or dot array). The LC7932,7932M are especially suited for use in LED display panel,PPC photosensitive drum LED erase head applications.

Features

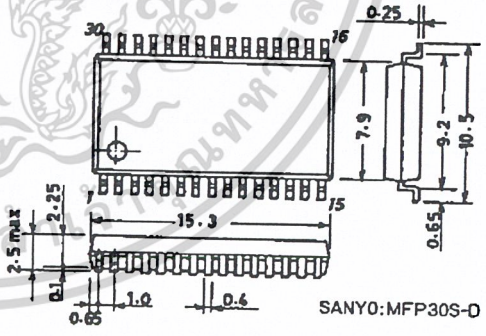
- Silicon gate C-MOS device capable of high-speed, high-current drive
- High-speed shiftable 16-bit bidirectional shift register/16-bit latch/output control circuit/16-bit N-channel transistor·open drain output transistor on chip
- Serial shift data is shifted on the positive transition of the clock (CLOCK) pulse.
- The data latch circuit outputs input data when the latch control ($\overline{\text{LATCH}}$) pin is at "L" level and holds output data when the latch control ($\overline{\text{LATCH}}$) pin is at "H" level.
- Maximum ratings of driver output: $V_O = +15V$, $I_{OL} = 30mA$(STATIC) /120mA(DYNAMIC).
- Operating voltage of logic unit: $V_{DD} = 4.5V$ to $5.5V$
- Operating clock frequency: $f_{CLK} = DC$ to 5MHz(max)
- Package : LC7932 : DIP30S
  LC7932M : MFP30S
- The bidirectional shift register is so designed as to cause a shift to occur in the SI to SO direction when L/R="L" level and in the SO to SI direction when L/R="H" level.
- When a high level is applied to the LSET pin ("latch set"), the latch data is set to the high level. The latch data does not change when the LSET pin is low or open.

Package Dimensions 3061 [LC7932]
(unit:mm)



SANYO: DIP30S

Package Dimensions 3073A [LC7932M]
(unit:mm)



SANYO:MFP30S-D

Absolute Maximum Ratings at Ta = 25°C

				unit
Maximum Supply Voltage	V_{DD} max		−0.3 to +7.0	V
Input Voltage	V_I		−0.3 to V_{DD}+0.3	V
Output Voltage	V_O(1)	SOUT(SIN) output	−0.3 to V_{DD}+0.3	V
	V_O(2)	D1 to D16 output,output Tr OFF	15	V
Output Current	I_O	D1 to D16 output,per output pin	30	mA
Operating Temperature	Topr		−25 to +85	°C
Storage Temperature	Tstg	(Note)	−35 to +125	°C
Allowable Power Dissipation	Pd max	LC7932 Ta = 85°C	400	mW
		LC7932M Ta = 85°C	270	mW

(Note) When mounting the MFP package version, do not dip it in solder.

Allowable Operating Conditions at Ta = −25°C to +85°C

			min	typ	max	unit
Supply Voltage	V_{DD}	V_{DD}	4.5		5.5	V
Input "H"-Level Voltage	V_{IH}	SIN(SOUT),CLOCK, $\overline{\text{LATCH}}$,BEO, $\overline{\text{STROBE}}$, LSET,L/R	0.8V_{DD}		V_{DD}	V
Input "L"-Level Voltage	V_{IL}	SIN(SOUT),CLOCK, $\overline{\text{LATCH}}$,BEO, $\overline{\text{STROBE}}$, LSET,L/R	V_{SS}(L)		0.2V_{DD}	V
Clock Frequency	fCLK	CLOCK Duty:50%			5.0	MHz
Clock Pulse Width	$t_{W\phi}$	CLOCK	75			ns
Clock Rise/Fall Time	t_r,t_f	CLOCK			200	ns
Data Setup Time	t_{DS}	SIN(SOUT) CLOCK	100			ns
Data Hold Time	t_{DH}	SIN(SOUT) CLOCK	50			ns
Latch Pulse Widh	t_{WL}	$\overline{\text{LATCH}}$	100			ns

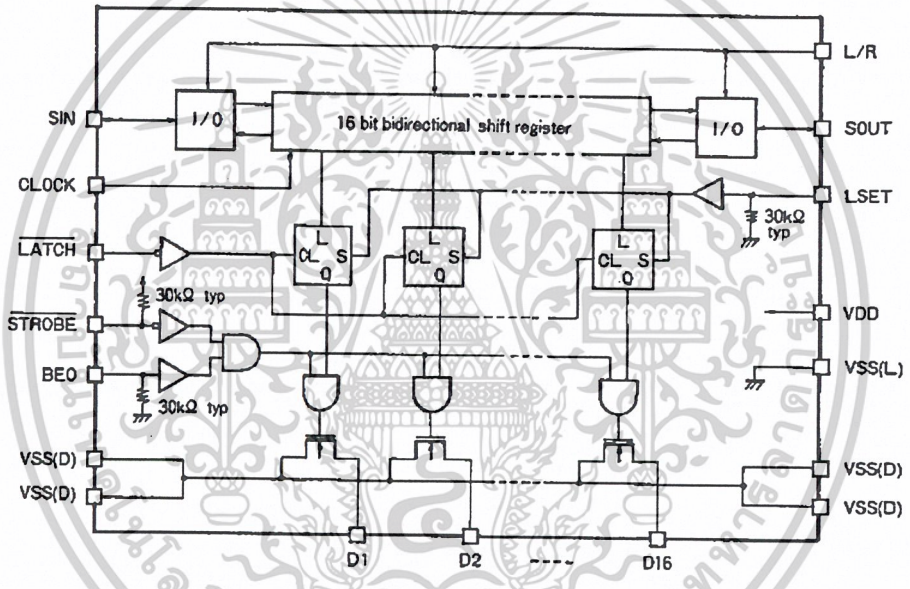
Electrical Characteristics at Ta = 25°C

				min	typ	max	unit
Input "H"-Level Current	I_{IH}(1)	SIN(SOUT),CLOCK, $\overline{\text{LATCH}}$,L/R				10	μA
	I_{IH}(2)	BEO,LSET			170		μA
Input "L"-Level Current	I_{IL}(1)	SIN(SOUT),CLOCK, $\overline{\text{LATCH}}$,L/R		−10			μA
	I_{IL}(2)	$\overline{\text{STROBE}}$			170		μA
Output "H"-Level Voltage	V_{OH}	SOUT(SIN)	I_{OH} = −0.5mA, V_{DD} = 5V	V_{DD}−0.5			V
Output "L"-Level Voltage	V_{OL}(1)	SOUT(SIN)	I_{OL} = 0.5mA, V_{DD} = 5V			0.5	V
	V_{OL}(2)	D1 to D16	I_{OL} = 30mA, V_{DD} = 5V			0.5	V
Output OFF-State Leakage Current	I_{OFF}	D1 to D16	V_O = 15V			20	μA
Input Capacitance	C_{IN}	CLOCK			5.0		pF
Operating Current	I_{DD}	V_{DD}	fCLK = 5MHz, V_{DD} = 5V, All outputs with no load			5	mA

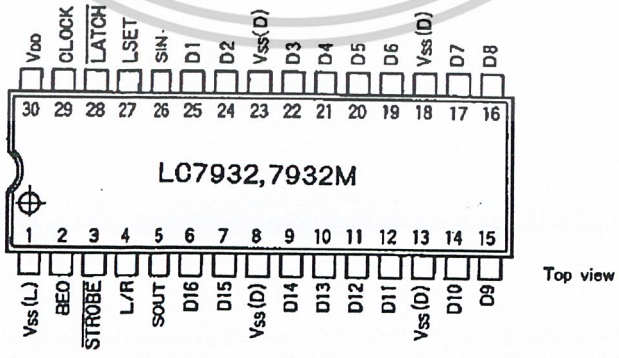
Switching Characteristics at Ta = 25°C

					min	typ	max	unit
Clock Latch Delay Width	t_{CL}	CLOCK,$\overline{\text{LATCH}}$		V_{DD}=5V	100			ns
Latch Clock Delay Width	t_{LC}	CLOCK,$\overline{\text{LATCH}}$		V_{DD}=5V	0			ns
Output"H"-Level Propagation Delay Time	t_{PLH}(1)	$\overline{\text{LATCH}}$ D1 to D16	Dn; RL=1.0kΩ, CL=15pF	V_{DD}=5V			400	ns
	t_{PLH}(2)	BEO,$\overline{\text{STROBE}}$ D1 to 16	Dn; RL=1.0kΩ, CL=15pF	V_{DD}=5V			300	ns
	t_{PLH}(3)	CLOCK,SOUT(SIN)	SOUT;CL=15pF	V_{DD}=5V			200	ns
Output"L"-Level Propagation Delay Time	t_{PHL}(1)	$\overline{\text{LATCH}}$,LSET D1 to D16	Dn; RL=1.0kΩ, CL=15pF	V_{DD}=5V			200	ns
	t_{PHL}(2)	BEO,$\overline{\text{STROBE}}$ D1 to D16	Dn; RL=1.0kΩ, CL=15pF	V_{DD}=5V			100	ns
	t_{PHL}(3)	CLOCK,SOUT(SIN)	SOUT;CL=15pF	V_{DD}=5V			200	ns

Equivalent Circuit



Pin Assignment

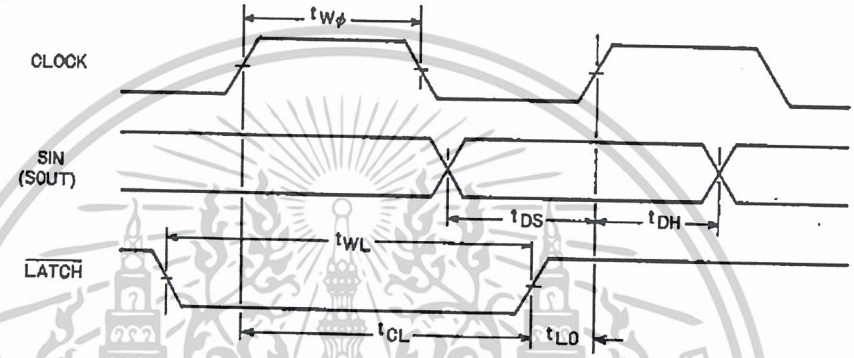


The package comes in two types - DIP30S and MFP30S.

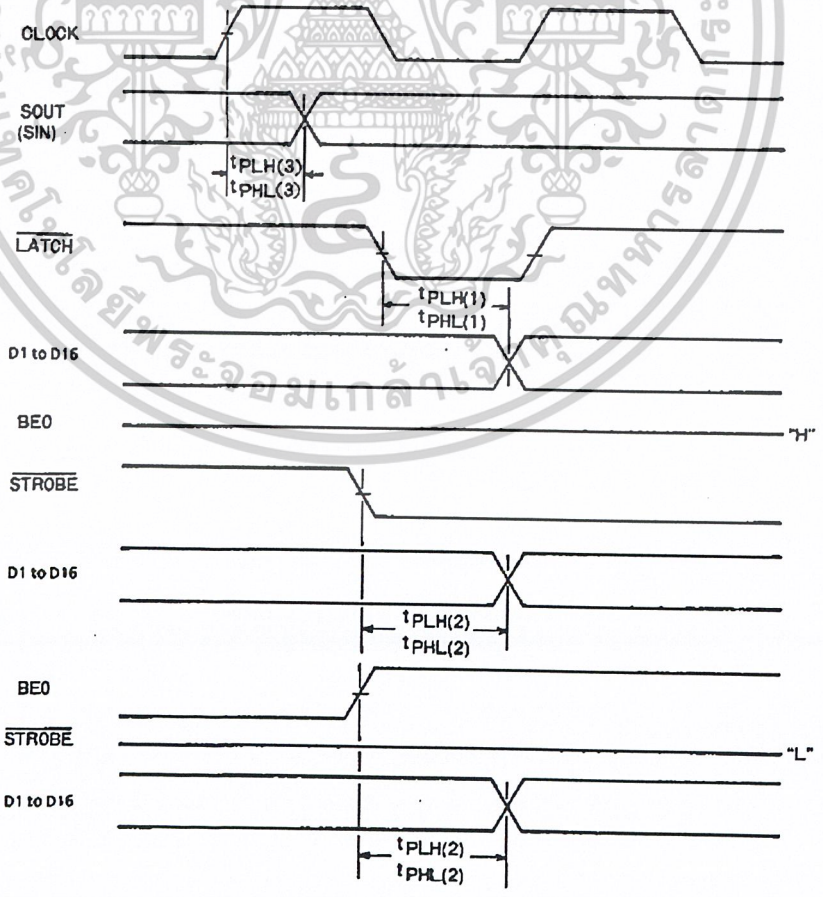
LED Driver ON/OFF Truth Table

Latch Data (Q)	BEO	$\overline{\text{STROBE}}$	LED Driver
0	0	0	OFF
1	0	0	OFF
0	1	0	OFF
1	1	0	ON Driver ON
0	0	1	OFF
1	0	1	OFF
0	1	1	OFF
1	1	1	OFF

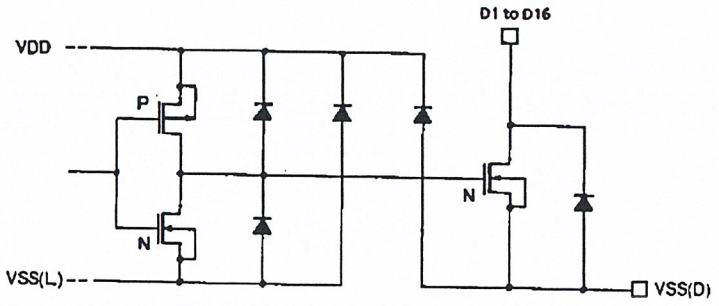
Input Data Timing Chart



Output Data Timing Chart



Equivalent Circuit for Output Driver Section



(Note) L/R="H" level : ( )

โปรแกรม

- C++

```
// philDlg.h : header file
//

#pragma once
#include "afxwin.h"


// CphilDlg dialog
class CphilDlg : public CDialog
{
// Construction
public:
	CphilDlg(CWnd* pParent = NULL);	// standard constructor
// Dialog Data
	enum { IDD = IDD_PHIL_DIALOG };

	protected:
	virtual void DoDataExchange(CDataExchange* pDX);	// DDX/DDV
support
// Implementation
protected:
	HICON m_hIcon;

	// Generated message map functions
	virtual BOOL OnInitDialog();
	afx_msg void OnSysCommand(UINT nID, LPARAM lParam);
	afx_msg void OnPaint();
	afx_msg HCURSOR OnQueryDragIcon();
	DECLARE_MESSAGE_MAP()
public:

	CString Pass;
	CString User;
	afx_msg void OnBnClickedOk();
	afx_msg void OnBnClickedNextp1();

	CEdit pass_1;
	afx_msg void OnEnSetfocusEdit1();
	CComboBox Usern;
	afx_msg void OnCbnSelendokCombo1();
};
```

```
// philDlg.cpp : implementation file
#include "stdafx.h"
#include "phil.h"
#include "philDlg.h"
#include "page2.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#endif
// CAboutDlg dialog used for App About
class CAboutDlg : public CDialog
{
public:
	CAboutDlg();
// Dialog Data
	enum { IDD = IDD_ABOUTBOX };

	protected:
	virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV
support

// Implementation
protected:
	DECLARE_MESSAGE_MAP()
};
CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
}
void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
}
BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
END_MESSAGE_MAP()

// CphilDlg dialog

CphilDlg::CphilDlg(CWnd* pParent /*=NULL*/)
	: CDialog(CphilDlg::IDD, pParent)
	, Pass(_T(""))
, User(_T(""))
{
	m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CphilDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	DDX_Text(pDX, IDC_EDIT1, Pass);
	DDX_CBString(pDX, IDC_COMBO1, User);
	DDX_Control(pDX, IDC_EDIT1, pass_1);
	DDX_Control(pDX, IDC_COMBO1, Usern);
}

BEGIN_MESSAGE_MAP(CphilDlg, CDialog)
	ON_WM_SYSCOMMAND()
	ON_WM_PAINT()
	ON_WM_QUERYDRAGICON()
```

```
	ON_BN_CLICKED(IDOK, OnBnClickedOk)
	ON_BN_CLICKED(IDC_NEXTp1, OnBnClickedNextp1)

	ON_EN_SETFOCUS(IDC_EDIT1, OnEnSetfocusEdit1)

ON_CBN_SELENDOK(IDC_COMBO1, OnCbnSelendokCombo1)
END_MESSAGE_MAP()
// CphilDlg message handlers
BOOL CphilDlg::OnInitDialog()
{
	CDialog::OnInitDialog();

	// Add "About..." menu item to system menu.
	// IDM_ABOUTBOX must be in the system command range.
	ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
	ASSERT(IDM_ABOUTBOX < 0xF000);
	CMenu* pSysMenu = GetSystemMenu(FALSE);
	if (pSysMenu != NULL)
	{
		CString strAboutMenu;
		strAboutMenu.LoadString(IDS_ABOUTBOX);
		if (!strAboutMenu.IsEmpty())
		{
			pSysMenu->AppendMenu(MF_SEPARATOR);
			pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX,
strAboutMenu);
		}
	}
	// Set the icon for this dialog.  The framework does this
automatically
	//  when the application's main window is not a dialog
	SetIcon(m_hIcon, TRUE);			// Set big icon
	SetIcon(m_hIcon, FALSE);		// Set small icon
	return TRUE;  // return TRUE  unless you set the focus to a
control
}
void CphilDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
	if ((nID & 0xFFF0) == IDM_ABOUTBOX)
	{
		CAboutDlg dlgAbout;
		dlgAbout.DoModal();
	}
	else
	{
		CDialog::OnSysCommand(nID, lParam);
	}
}
// If you add a minimize button to your dialog, you will need the code
below
//  to draw the icon.  For MFC applications using the document/view
model,
//  this is automatically done for you by the framework.

void CphilDlg::OnPaint()
```

```
{     if (IsIconic())
      {
            CPaintDC dc(this); // device context for painting
            SendMessage(WM_ICONERASEBKGND,
reinterpret_cast<WPARAM>(dc.GetSafeHdc()), 0);

            // Center icon in client rectangle
            int cxIcon = GetSystemMetrics(SM_CXICON);
            int cyIcon = GetSystemMetrics(SM_CYICON);
            CRect rect;
            GetClientRect(&rect);
            int x = (rect.Width() - cxIcon + 1) / 2;
            int y = (rect.Height() - cyIcon + 1) / 2;
            dc.DrawIcon(x, y, m_hIcon);
      }
      else
      {
            CDialog::OnPaint();
      }
}
// The system calls this function to obtain the cursor to display while
the user drags
//  the minimized window.
HCURSOR CphilDlg::OnQueryDragIcon()
{
      return static_cast<HCURSOR>(m_hIcon);
}
void CphilDlg::OnBnClickedOk()
{
      OnOK();
}

void CphilDlg::OnBnClickedNextp1()
{     UpdateData(TRUE);
      Cpage2 pag2;
      if( (Pass=="6126" && User=="Suphanee")||(Pass=="6126" &&
User=="Napa"))
      {     OnCancel();
            pag2.DoModal();
      }
      else
      {     MessageBox("Invalid username and password","Sorry",MB_OK);
            UpdateData(FALSE);
      }
            }

void CphilDlg::OnEnSetfocusEdit1()
{     UpdateData(TRUE);
      pass_1.SetPasswordChar('*');
      UpdateData(FALSE);
}
void CphilDlg::OnCbnSelendokCombo1()
{     CString item;
      Usern.GetLBText(0,item);
      Usern.FindString(0,item);
      UpdateData(FALSE);
      }
```

```
// Cpage2.h : header file
#pragma once
#include "afxcmn.h"
#include "afxwin.h"
#include "mscomm1.h"
#include "mscomm2.h"
// Cpage2 dialog
class Cpage2 : public CDialog
{     DECLARE_DYNAMIC(Cpage2)
public:
      Cpage2(CWnd* pParent = NULL);    // standard constructor
      virtual ~Cpage2();
// Dialog Data
      enum { IDD = IDD_DIALOG1 };
protected:
      virtual void DoDataExchange(CDataExchange* pDX);     // DDX/DDV
support
      DECLARE_MESSAGE_MAP()
public:
      int Notab;
      afx_msg void OnBnClickedOk();
      CTabCtrl table;
      CString     tab_str[5];
      //price in either menu from menu 1 to menu 30
      int Price_I[100];
      int statusmenu[100];
      // number of menu in list for a table from tab 1 to tab 4
      int   tabNummenu[5];
      int price;
      // all of menu for a table;
      CString Array_menu[100];
      int Qtd[5][100];
      afx_msg void OnTcnSelchangeTab1(NMHDR *pNMHDR, LRESULT *pResult);
      virtual BOOL OnInitDialog();
      CString menu_str,menu_now;
      void initial(void);
      CComboBox menu_combo;
      CString menu_comb;
      afx_msg void OnBnClickedAdd();
      afx_msg void OnBnClickedDel();
      int tab;
      int Index;
      afx_msg void OnCbnKillfocusmenucombo();
      int Qtd_m;
      // for show menu item
      void show_menu(void);
      afx_msg void OnBnClickedCancel();
      afx_msg void OnBnClickedBilling();
      CMscomm1 comport1;
      DECLARE_EVENTSINK_MAP()
      void OnCommMscomm1();
      afx_msg void OnBnClickedsend();
      int n;
      afx_msg void OnBnClickedstatus();
      CMscomm2 comport2;
      CListBox emptytext;
};
```

```
// page2.cpp : implementation file
#include "stdafx.h"
#include "phil.h"
#include "page2.h"
#include "stdlib.h"
#include "page3.h"
#include "string.h"
// Cpage2 dialog
IMPLEMENT_DYNAMIC(Cpage2, CDialog)
Cpage2::Cpage2(CWnd* pParent /*=NULL*/)
	: CDialog(Cpage2::IDD, pParent)
	, menu_str(_T("No Item"))
	, menu_comb(_T(""))
	, tab(4)
	, Index(0)
	, Qtd_m(1)
{
}
Cpage2::~Cpage2()
{
}

void Cpage2::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	DDX_Control(pDX, IDC_TAB1, table);
	DDX_Text(pDX, IDC_menutext, menu_str);
	DDX_Control(pDX, IDC_menucombo, menu_combo);
	DDX_CBString(pDX, IDC_menucombo, menu_comb);
	DDX_Text(pDX, IDC_EDIT1, Qtd_m);
	DDV_MinMaxInt(pDX, Qtd_m, 0, 100);
	DDX_Control(pDX, IDC_MSCOMM1, comport1);
	DDX_Control(pDX, IDC_MSCOMM2, comport2);
	DDX_Control(pDX, IDC_empty, emptytext);
}
BEGIN_MESSAGE_MAP(Cpage2, CDialog)
	ON_BN_CLICKED(IDOK, OnBnClickedOk)
	ON_NOTIFY(TCN_SELCHANGE, IDC_TAB1, OnTcnSelchangeTab1)

	ON_BN_CLICKED(IDC_Add, OnBnClickedAdd)
	ON_BN_CLICKED(IDC_Del, OnBnClickedDel)

ON_CBN_KILLFOCUS(IDC_menucombo, OnCbnKillfocusmenucombo)
ON_BN_CLICKED(IDCANCEL, OnBnClickedCancel)
ON_BN_CLICKED(IDC_Billing, OnBnClickedBilling)
ON_BN_CLICKED(IDC_status, OnBnClickedstatus)
END_MESSAGE_MAP()
// Cpage2 message handlers
void Cpage2::OnBnClickedOk()
{
	exit(2);
}
void Cpage2::OnTcnSelchangeTab1(NMHDR *pNMHDR, LRESULT *pResult)
{	int i;
	tab = table.GetCurFocus()+1;
	for (i=1;i<=5;i++)
	{	if (tab == i)
```

```
            {
            menu_str=tab_str[i];
            }
        }
        UpdateData(FALSE);
        *pResult = 0;
}
BOOL Cpage2::OnInitDialog()
{
        CDialog::OnInitDialog();
            // COM Port Initialisation
        initial();
        comport1.put__CommPort(1);
        comport1.put_Settings("9600,N,8,2");
        comport1.put_InputLen(1);
        comport1.put_InputMode(0);//text mode
        comport1.put_RTSEnable(TRUE);
        comport1.put_RThreshold(1);
        comport1.put_PortOpen(true);
        comport2.put__CommPort(2);
        comport2.put_Settings("9600,N,8,2");
        comport2.put_InputLen(1);
        comport2.put_InputMode(0);//text mode
        comport2.put_RTSEnable(TRUE);
        comport2.put_RThreshold(1);
        comport2.put_PortOpen(true);
        table.InsertItem(0,"TAB 4");
        table.InsertItem(0,"TAB 3");
        table.InsertItem(0,"TAB 2");
        table.InsertItem(0,"TAB 1");
        menu_combo.AddString("001 TOAST                ");
        menu_combo.AddString("002 FRIED EGGS WITH HAM");
        menu_combo.AddString("003 CHICKEN CREAM SOUP ");
        menu_combo.AddString("004 MUSHROOM CREAM SOUP");
        menu_combo.AddString("005 CORN CREAM SOUP     ");
        menu_combo.AddString("006 MIXED SALAD         ");
        menu_combo.AddString("007 TUNA FISH SALAD     ");
        menu_combo.AddString("008 CHICKEEN SALAD      ");
        menu_combo.AddString("009 BEEF STEAK SALAD    ");
        menu_combo.AddString("010 SEAFOOD SALAD       ");
        menu_combo.AddString("011 CHICKEEN SANDWICH   ");
        menu_combo.AddString("012 TUNA FISH SANDWICH ");
        menu_combo.AddString("013 HAM SANDWICH        ");
        menu_combo.AddString("014 GRILLED HAM&CHEESE SANDWICH");
        menu_combo.AddString("015 GARLIC BREAD                ");
        menu_combo.AddString("016 FRENCH FRIED                ");
        menu_combo.AddString("017 CLUB SANDWICH               ");
        menu_combo.AddString("018 CHICKEN STEAK               ");
        menu_combo.AddString("019 PORK STEAK                  ");
        menu_combo.AddString("020 FISH STEAK                  ");
        menu_combo.AddString("021 LONDON FISH STEAK           ");
        menu_combo.AddString("022 FILLET STEAK                ");
        menu_combo.AddString("023 PEPPER STEAK                ");
        menu_combo.AddString("024 FILLET MIGNON STEAK         ");
        menu_combo.AddString("025 SPAGHETTI WITH BEEF         ");
        menu_combo.AddString("026 SPAGHETTI WITH PORK         ");
        menu_combo.AddString("027 SPAGHETTI WITH CHICKEN      ");
```

```
menu_combo.AddString("028 SPAGHETTI NAPOLITAN          ");
menu_combo.AddString("029 SPAGHETTI WITH MUSSEL        ");
menu_combo.AddString("030 FRIED MACARONI WITH SHRIMP   ");
menu_combo.AddString("031 FRIED MACARONI WITH HAM      ");
menu_combo.AddString("032 FRIED PORK SAUSAGE           ");
menu_combo.AddString("033                              ");
menu_combo.AddString("034                              ");
menu_combo.AddString("035                              ");
menu_combo.AddString("036                              ");
menu_combo.AddString("037                              ");
menu_combo.AddString("038                              ");
menu_combo.AddString("039                              ");
menu_combo.AddString("040                              ");
menu_combo.AddString("041                              ");
menu_combo.AddString("042                              ");
menu_combo.AddString("043                              ");
menu_combo.AddString("044                              ");
menu_combo.AddString("045                              ");
menu_combo.AddString("046                              ");
menu_combo.AddString("047                              ");
menu_combo.AddString("048                              ");
menu_combo.AddString("049                              ");
menu_combo.AddString("050                              ");
menu_combo.AddString("051 ESPRESSO          ");
menu_combo.AddString("052 MOCHA             ");
menu_combo.AddString("053 LATTE             ");
menu_combo.AddString("054 CAPUCCINO         ");
menu_combo.AddString("055 ICE ESPRESSO      ");
menu_combo.AddString("056 ICE MOCHA         ");
menu_combo.AddString("057 ICE LATTE         ");
menu_combo.AddString("058 ICE CAPUCCINO     ");
menu_combo.AddString("059 FRAPPE ESPRESSO   ");
menu_combo.AddString("060 FRAPPE MOCHA      ");
menu_combo.AddString("061 FRAPPE LATTE      ");
menu_combo.AddString("062 FRAPPE CAPUCCINO");
menu_combo.AddString("063                                ");
menu_combo.AddString("064                                ");
menu_combo.AddString("065                                ");
menu_combo.AddString("066                                ");
menu_combo.AddString("067                                ");
menu_combo.AddString("068                                ");
menu_combo.AddString("069                                ");
menu_combo.AddString("070                                ");
menu_combo.AddString("071 HOT TEA    ");
menu_combo.AddString("072 ICE THAI TEA     ");
menu_combo.AddString("073 ICE THAI TEA FRAPPE    ");
menu_combo.AddString("074 ICE LEMON TEA    ");
menu_combo.AddString("075 ICE CHOCOLATE    ");
menu_combo.AddString("076 ICE CHOCOLATE FRAPPE   ");
menu_combo.AddString("077 HOT FRESH MILK   ");
menu_combo.AddString("078 ICE FRESH MILK   ");
menu_combo.AddString("079 ORANGE JUICE     ");
menu_combo.AddString("080 ORANGE JUICE FRAPPE    ");
menu_combo.AddString("081 LEMON JUICE      ");
menu_combo.AddString("082 LEMON JUICE FRAPPE     ");
menu_combo.AddString("083 GUAVA MIXED DRINK      ");
menu_combo.AddString("084 COKE       ");
```

```
      menu_combo.AddString("085 ICE CUBE / JUG  ");
      menu_combo.AddString("086 ICE CUBE / GLASS      ");
      menu_combo.AddString("087 SINGHA BEER     ");
      menu_combo.AddString("088 HEINEKEN BEER   ");
Array_menu[1]     ="    001   TOAST                          ";
Array_menu[2]     ="    002   FRIED EGGS WITH HAM            ";
Array_menu[3]     ="    003   CHICKEN CREAM SOUP             ";
Array_menu[4]     ="    004   MUSHROOM CREAM SOUP            ";
Array_menu[5]     ="    005   CORN CREAM SOUP                ";
Array_menu[6]     ="    006   MIXED SALAD                    ";
Array_menu[7]     ="    007   TUNA FISH SALAD                ";
Array_menu[8]     ="    008   CHICKEEN SALAD                 ";
Array_menu[9]     ="    009   BEEF STEAK SALAD               ";
Array_menu[10]    ="    010   SEAFOOD SALAD                  ";
Array_menu[11]    ="    011   CHICKEEN SANDWICH              ";
Array_menu[12]    ="    012   TUNA FISH SANDWICH             ";
Array_menu[13]    ="    013   HAM SANDWICH                   ";
Array_menu[14]    ="    014   GRILLED HAM&CHEESE SANDWICH    ";
Array_menu[15]    ="    015   GARLIC BREAD                   ";
Array_menu[16]    ="    016   FRENCH FRIED                   ";
Array_menu[17]    ="    017   CLUB SANDWICH                  ";
Array_menu[18]    ="    018   CHICKEN STEAK                  ";
Array_menu[19]    ="    019   PORK STEAK                     ";
Array_menu[20]    ="    020   FISH STEAK                     ";
Array_menu[21]    ="    021   LONDON FISH STEAK              ";
Array_menu[22]    ="    022   FILLET STEAK                        ";
Array_menu[23]    ="    023   PEPPER STEAK                   ";
Array_menu[24]    ="    024   FILLET MIGNON STEAK           ";
Array_menu[25]    ="    025   SPAGHETTI WITH BEEF           ";
Array_menu[26]    ="    026   SPAGHETTI WITH PORK           ";
Array_menu[27]    ="    027   SPAGHETTI WITH CHICKEN        ";
Array_menu[28]    ="    028   SPAGHETTI NAPOLITAN           ";
Array_menu[29]    ="    029   SPAGHETTI WITH MUSSE          ";
Array_menu[30]    ="    030   FRIED MACARONI WITH SHRIMP    ";
Array_menu[31]    ="    031   FRIED MACARONI WITH HAM       ";
Array_menu[32]    ="    032   FRIED PORK SAUSAGE            ";
Array_menu[51]    ="    051   ESPRESSO                       ";
Array_menu[52]    ="    052   MOCHA                          ";
Array_menu[53]    ="    053   LATTE                          ";
Array_menu[54]    ="    054   CAPUCCINO                      ";
Array_menu[55]    ="    055   ICE ESPRESSO                   ";
Array_menu[56]    ="    056   ICE MOCHA                      ";
Array_menu[57]    ="    057   ICE LATTE                      ";
Array_menu[58]    ="    058   ICE CAPUCCINO                  ";
Array_menu[59]    ="    059   FRAPPE ESPRESSO               ";
Array_menu[60]    ="    060   FRAPPE MOCHA                  ";
Array_menu[61]    ="    061   FRAPPE LATTE                  ";
Array_menu[62]    ="    062   FRAPPE CAPUCCINO              ";
Array_menu[71]    ="    071   HOT TEA                              ";
Array_menu[72]    ="    072   ICE THAI TEA                          ";
Array_menu[73]    ="    073   ICE THAI TEA FRAPPE                   ";
Array_menu[74]    ="    074   ICE LEMON TEA                  ";
Array_menu[75]    ="    075   ICE CHOCOLATE                 ";
Array_menu[76]    ="    076   ICE CHOCOLATE FRAPPE          ";
Array_menu[77]    ="    077   HOT FRESH MILK                ";
Array_menu[78]    ="    078   ICE FRESH MILK                ";
Array_menu[79]    ="    079   ORANGE JUICE                  ";
```

```
Array_menu[80]    ="    080   ORANGE JUICE FRAPPE          ";
Array_menu[81]    ="    081   LEMON JUICE                  ";
Array_menu[82]    ="    082   LEMON JUICE FRAPPE          ";
Array_menu[83]    ="    083   GUAVA MIXED DRINK           ";
Array_menu[84]    ="    084   COKE                         ";
Array_menu[85]    ="    085   ICE CUBE / JUG              ";
Array_menu[86]    ="    086   ICE CUBE / GLASS            ";
Array_menu[87]    ="    087   SINGHA BEER                 ";
Array_menu[88]    ="    088   HEINEKEN BEER               ";
      Price_I[1]=15;
      Price_I[2]=30;
      Price_I[3]=25;
      Price_I[4]=25;
      Price_I[5]=25;
      Price_I[6]=30;
      Price_I[7]=35;
      Price_I[8]=35;
      Price_I[9]=45;
      Price_I[10]=45;
      Price_I[11]=20;
      Price_I[12]=20;
      Price_I[13]=20;
      Price_I[14]=20;
      Price_I[15]=20;
      Price_I[16]=25;
      Price_I[17]=30;
      Price_I[18]=39;
      Price_I[19]=39;
      Price_I[20]=49;
      Price_I[21]=49;
      Price_I[22]=60;
      Price_I[23]=80;
      Price_I[24]=90;
      Price_I[25]=35;
      Price_I[26]=35;
      Price_I[27]=35;
      Price_I[28]=35;
      Price_I[29]=45;
      Price_I[30]=40;
      Price_I[31]=35;
      Price_I[32]=40;
      Price_I[51]=25;
      Price_I[52]=35;
      Price_I[53]=35;
      Price_I[54]=35;
      Price_I[55]=30;
      Price_I[56]=40;
      Price_I[57]=40;
      Price_I[58]=40;
      Price_I[59]=35;
      Price_I[60]=45;
      Price_I[61]=45;
      Price_I[62]=45;
      Price_I[71]=20;
      Price_I[72]=20;
      Price_I[73]=25;
      Price_I[74]=20;
```

```
	Price_I[75]=20;
	Price_I[76]=25;
	Price_I[77]=20;
	Price_I[78]=25;
	Price_I[79]=20;
	Price_I[80]=25;
	Price_I[81]=20;
	Price_I[82]=25;
	Price_I[83]=15;
	Price_I[84]=10;
	Price_I[85]=10;
	Price_I[86]=2;
	Price_I[87]=65;
	Price_I[88]=75;
	return TRUE;  // return TRUE unless you set the focus to a
control
	// EXCEPTION: OCX Property Pages should return FALSE
}
void Cpage2::initial(void)
{
	for (int i=1 ;i<100;i++)
	{	menu_str="Initial";
		Price_I[i]=0;
		statusmenu[i]=1;
		for (int j=1 ;j<5;j++)
			Qtd[j][i]=0;
	}
	n = 3;
	UpdateData(FALSE);
}
void Cpage2::OnBnClickedAdd()
{	UpdateData(TRUE);
	int m=1;
while (m==1)
{	Index=menu_combo.GetCurSel();
	if(Index==LB_ERR)
	{MessageBox("Please,Select menu from list",
			"Restuarant",MB_ICONINFORMATION);
	break;
	}
	UpdateData(TRUE);
	Index++;
	if (Price_I[Index]==0)
		MessageBox("No item","Shop",MB_OK);
//		inack = NAK;
	else
		{if (statusmenu[Index]==0)
			MessageBox("This item is empty","Shop",MB_OK);
		else
			{	Qtd[tab][Index]+=Qtd_m;
			}
		}
	show_menu();
	UpdateData(FALSE);
	break;
}
}
```

```
void Cpage2::OnBnClickedDel()
{     UpdateData(TRUE);
      int m=1;
while (m==1)
{     Index=menu_combo.GetCurSel();
      if(Index==LB_ERR)
      {MessageBox("Please,Select menu from list",
                  "Restuarant",MB_ICONINFORMATION);
      break;
      }
      UpdateData(TRUE);
      Index++;
      if (Qtd[tab][Index]<Qtd_m)
      {MessageBox("Sorry, No item to delete",
                  "Restuarant",MB_ICONINFORMATION);
      break;
      }
      Qtd[tab][Index]-=Qtd_m;
      show_menu();
      UpdateData(FALSE);
      break;
}
      // TODO: Add your control notification handler code here
}
void Cpage2::OnCbnKillfocusmenucombo()
{     int status_combo;
      CString item;
      UpdateData(TRUE);
      menu_combo.GetLBText(0,item);
      Index=menu_combo.FindString(0,item);
      status_combo=menu_combo.GetCurSel();
      UpdateData(FALSE);
      // TODO: Add your control notification handler code here
}
// for show menu item
void Cpage2::show_menu(void)
{     char str[10];
      char Qtd_str[10];
      tab_str[tab]="รายการที่คุณเลือกคือ...\n";
      for (int i=1; i < 100;i++)
      {     if (Qtd[tab][i]!=0)

      {     _itoa(Price_I[i],str,10);
            _itoa(Qtd[tab][i],Qtd_str,10);

            tab_str[tab]+=Array_menu[i]+"....."+str+"    "+Qtd_str+"\n";
            }
      }
            menu_str=tab_str[tab];

      UpdateData(FALSE);
}
void Cpage2::OnBnClickedCancel()
{
      exit(2);
      OnCancel();
}
```

```
void Cpage2::OnBnClickedBilling()
{     char Price_str[10];
      CString text;
      Cpage3 pag3;
      int Price_int=0;

      for (int i=1; i < 100;i++)
      {     if (Qtd[tab][i]!=0)
            {
            Price_int+=Price_I[i]*Qtd[tab][i];
            }
      }
      price = Price_int;
      _itoa(Price_int,Price_str,10);
      pag3.bypassprice_(price);
      pag3.DoModal();
      if (pag3.onclear()==TRUE)
      {     char table;
            CString table_str;
            table = tab+47;//2fh
            table_str= table;
            comport2.put_Output(COleVariant(table_str));
            menu_str="";
            tab_str[tab]=menu_str;
                  for (int i=1 ;i<100;i++)
      {     Qtd[tab][i]=0;
      }
      }

      UpdateData(FALSE);
}
BEGIN_EVENTSINK_MAP(Cpage2, CDialog)
      ON_EVENT(Cpage2, IDC_MSCOMM1, 1, OnCommMscomm1, VTS_NONE)
END_EVENTSINK_MAP()
void Cpage2::OnCommMscomm1()
{
      if (comport1.get_CommEvent()==2 )
            if (comport1.get_InBufferCount()!=0 )
            {     VARIANT in_dat;
                  int in_dat2;
                  char in,inack;
                  char ACK=64;//40h
                  char EMPTY=71;//47h
                  char NAK=79;//4fh
                  CString send;
                  in_dat2=comport1.get_InBufferCount();

                  in_dat =  comport1.get_Input();
                  CString strInput(in_dat.bstrVal);
                  in = strInput[0];
                  switch (n)
                  {           case 1:
                                    n=4;
                                    if (Price_I[Index]==0)
                                          inack = NAK;
                                    else
```

```
                                        {if (statusmenu[Index]==0)
                                              inack = EMPTY;
                                         else
                                         {    inack = ACK;
                                              Qtd_m = in;
                                              Qtd[tab][Index]+=Qtd_m;
                                         }
                                        }
                                  send = inack;
                              comport1.put_Output(COleVariant(send));
                                  show_menu();
                                  break;
                              case 2:
                                  Index = in;

                                  break;
                              case 3:
                                  tab = in-64;
                                  table.SetCurFocus(tab-1);
                        }
                        n--;
                        UpdateData(FALSE);
                  }
      // TODO: Add your message handler code here
}
void Cpage2::OnBnClickedstatus()
{
      int m=1;
while (m==1)
{     Index=menu_combo.GetCurSel();
      if(Index==LB_ERR)
      {MessageBox("Please,Select menu from list",
            "Restuarant",MB_ICONINFORMATION);
      break;
      }
      UpdateData(TRUE);
      Index++;
// toggle menu status
      if (statusmenu[Index] == 0)
      {     statusmenu[Index]=1;
      }
      else
      {     statusmenu[Index]=0;
      }
            emptytext.ResetContent();
            emptytext.AddString(" Out of stock is ...");
      for (int i=1; i < 100;i++)
      {     if (statusmenu[i]==0)
      {
            emptytext.AddString(Array_menu[i]);
      }
      }
      UpdateData(FALSE);
      break;
}     // TODO: Add your control notification handler code here
}
```

```
// Cpage3.h : header file
#pragma once
#include "afxwin.h"

// Cpage3 dialog

class Cpage3 : public CDialog
{
	DECLARE_DYNAMIC(Cpage3)

public:
	Cpage3(CWnd* pParent = NULL);   // standard constructor
	virtual ~Cpage3();

// Dialog Data
	enum { IDD = IDD_DIALOG2 };

protected:
	virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV
support

	DECLARE_MESSAGE_MAP()
public:
	afx_msg void OnBnClicked5p();
	afx_msg void OnBnClicked10p();
	afx_msg void OnBnClickedNop();
	int getbank;
	double percent;
	int price_r;
	int price_n;

	afx_msg void OnBnClickedOk();
	double change;
	int price_real;
	virtual BOOL OnInitDialog();
	void bypassprice_(int price_);
	CString staticprice;
	CString staticchange;
	afx_msg void OnBnClickedCancel();
	bool onclear(void);
	bool onclear_bool;
	afx_msg void OnBnClickedclear();
	afx_msg void OnStnClickedstaticchange();
	CStatic disc;
	int disc_;
	int total;
};
```

```
// page3.cpp : implementation file
//
#include "stdafx.h"
#include "phil.h"
#include "page3.h"
#include "page2.h"
// Cpage3 dialog

IMPLEMENT_DYNAMIC(Cpage3, CDialog)
Cpage3::Cpage3(CWnd* pParent /*=NULL*/)
      : CDialog(Cpage3::IDD, pParent)
      , getbank(0)
      , change(0)
, staticprice(_T("Price (no discount) "))
, staticchange(_T("Change ... "))
, disc_(0)
, total(0)
{
}

Cpage3::~Cpage3()
{
}
void Cpage3::DoDataExchange(CDataExchange* pDX)
{
      CDialog::DoDataExchange(pDX);
      DDX_Text(pDX, IDC_getbank, getbank);
      DDV_MinMaxInt(pDX, getbank, 0, 100000);
      DDX_Text(pDX, IDC_change, change);
      DDX_Text(pDX, IDC_price, price_real);
      DDX_Text(pDX, IDC_staticpri, staticprice);
      DDX_Text(pDX, IDC_staticchange, staticchange);

      DDX_Text(pDX, IDC_staticpri3, disc_);
      DDX_Text(pDX, IDC_staticpri5, total);
}
BEGIN_MESSAGE_MAP(Cpage3, CDialog)
      ON_BN_CLICKED(IDC_5P, OnBnClicked5p)
      ON_BN_CLICKED(IDC_10P, OnBnClicked10p)
      ON_BN_CLICKED(IDC_NoP, OnBnClickedNop)
      ON_BN_CLICKED(IDOK, OnBnClickedOk)
      ON_BN_CLICKED(IDCANCEL, OnBnClickedCancel)
      ON_BN_CLICKED(IDC_clear, OnBnClickedclear)
      ON_STN_CLICKED(IDC_staticchange, OnStnClickedstaticchange)
END_MESSAGE_MAP()
// Cpage3 message handlers

void Cpage3::OnBnClicked5p()
{     UpdateData(TRUE);
      percent=0.95;
      price_n = percent*price_real;
      total =price_n;
      disc_ =price_real-price_n;
      UpdateData(FALSE);
}
void Cpage3::OnBnClicked10p()
{     UpdateData(TRUE);
```

```
	percent=0.90;
	price_n = percent*price_real;
	total =price_n;
	disc_ =price_real-price_n;
	UpdateData(FALSE);
}
void Cpage3::OnBnClickedNop()
{	UpdateData(TRUE);
	percent=1.00;
	price_n = percent*price_real;
	total =price_n;
	disc_ =price_real-price_n;
	UpdateData(FALSE);
}
void Cpage3::OnBnClickedOk()
{
	UpdateData(TRUE);
	if  (getbank<price_n)
		MessageBox("Not enough money","Sorry",MB_OK);
	else
	{
		change=getbank-price_n;
	}
	UpdateData(FALSE);
}
BOOL Cpage3::OnInitDialog()
{
	CDialog::OnInitDialog();
	price_n=price_real;
	return TRUE;  // return TRUE unless you set the focus to a
control
	// EXCEPTION: OCX Property Pages should return FALSE
}
void Cpage3::bypassprice_(int price_)
{
	price_real=price_;
}

void Cpage3::OnBnClickedCancel()
{	onclear_bool=FALSE;
	OnCancel();
}
bool Cpage3::onclear(void)
{
	return onclear_bool;
}

void Cpage3::OnBnClickedclear()
{
	onclear_bool=TRUE;
	OnCancel();
}

void Cpage3::OnStnClickedstaticchange()
{
	// TODO: Add your control notification handler code here
}
```

- PIC12f675

โปรแกรมภาษาแอสเซมบลีของ PIC-12F675 ในส่วนเครื่องควบคุมการสั่งอาหาร

```
    list        p=12f675        ; list directive to define processor
    #include    <p12f675.inc>   ; processor specific variable
definitions

    errorlevel  -302            ; suppress message 302 from list file

    __CONFIG    _CPD_OFF & _CP_OFF & _BODEN_OFF & _MCLRE_OFF &
_PWRTE_OFF & _WDT_OFF & _INTRC_OSC_NOCLKOUT

    ; Register locations 0x20 to 0x5F (64 bytes) are General Purpose
    ; registers, implemented as static RAM and are mapped across both
    ; banks.

    cblock  0x20

        w_temp                      ; used for context saving
        status_temp                 ; used for context saving

        TEMP                        ; General Purpose Temporary
register

        CSR0                        ; TX buffer shift register
            Countpre
            Counttx1
            Countrx
        Count
        Count2
            CountRe
            CountRe2
        BitCount
        TimeHi
        TimeLo
        FuncBits                    ; Function Bits
        Counttab
        Numtab
        N_temp
            RX_BUFFER
            TX_BUFFER
            DATADDR
            DATMENU
            DATNUM
    endc

#define RXDCOM    GPIO, 1             ; (Input) Receive data from 51
#define TXDCOM    GPIO, 4             ; (Output ) Transmit Data to 51
#define TXD     GPIO, 2             ; (Output) Transmit Data
#define PB3     GPIO, 3             ; (Input Only) Push button switch
GP3
#define RXD  GPIO, 0          ; (Input) Receive data from wireless
#define RFENA   GPIO, 5             ; (Output) RF Enable

    ; Setting a TRISIO bit (= 1) will make the corresponding GPIO pin
and input.

    ;   GPIO Pins = xx543210
#define GPTRIS B'00001011'
#define TGUARD  D'8'                ; 46 X TE
#define PREAMB  D'16'               ; Preamble length = 16 pulses
```

```
#define Bytepoll  H'80'                    ;Poll
#define ByteACK  H'88'                     ;ACK
#define GROUPADDR  H'FF'                         ;group address
#define ACKCOM H'40'
#define NAKCOM H'4F'
#define EMPCOM H'47'

;-----------------------------------------------------------------------
--
; Program Memory
;-----------------------------------------------------------------------
--

    ; Program Memory Organization (Section 2.1)

        ORG     0x000               ; RESET Vector

        nop                         ; for ICD use
        goto    INITALIZE           ; goto INITALIZE


        ORG     0x004               ; Interrupt Vector

        movwf   w_temp              ; save W register
        swapf   STATUS, W           ; swap status to be saved into W
        bcf     STATUS, RP0         ; ---- Select Bank 0 -----
        movwf   status_temp         ; save STATUS register

;---------------------------------------
; Interrupt Service Routine (ISR)
;
; Description:
;
;---------------------------------------

; Interrupt-on-change (Section 3.2.2 and 9.4.3)
;
; An input change on GPIO change sets the GPIF bit. The interrupt can
; be enabled/disabled be setting/clearing the GPIE bit. Individual
pins
; are configured through the IOC register (see INITIALIZATION below).
;
; Clear the IOC interrupt by:
;  a) Any read or write of GPIO. This will end the mismatch condition.
;  b) Clear the flag bit GPIF

        movfw   GPIO                ; read GPIO
        bcf     INTCON, GPIF

;---------------------------------------

        swapf   status_temp, W      ; swap status_temp into W, sets
bank to original state
        movwf   STATUS              ; restore STATUS register
        swapf   w_temp, F
        swapf   w_temp, W           ; restore W register

        retfie
```

```
;-------------------------------------------------------------------------
--
; Subroutine DATA_EEPROM_READ
DATA_EEPROM_READ

        bsf     STATUS, RP0         ; ---- Select Bank 1 -----

        movwf   EEADR               ; move EEPROM address in W to
EEADR
        bsf     EECON1, RD          ; initiate EEPROM read
        movf    EEDATA, W           ; move data to W

        bcf     STATUS, RP0         ; ---- Select Bank 0 -----

        return


;-------------------------------------------------------------------------
--
; Subroutine DATA_EEPROM_WRITE

DATA_EEPROM_WRITE

        bsf     STATUS, RP0         ; ---- Select Bank 1 -----

        bsf     EECON1, WREN        ; EEPROM Write Enable: allow
write cycles
        bcf     INTCON, GIE         ; disable global interrupts
                                    ; *** required sequence, do not
alter ***
        movlw   0x55
        movwf   EECON2
        movlw   0xAA
        movwf   EECON2
        bsf     EECON1, WR          ; initiate EEPROM write
                                    ; *** end required sequence ***
        bsf     INTCON, GIE         ; enable global interrupts
        bcf     EECON1, WREN        ; EEPROM Write Enable: inhibit
write cycles

        bcf     STATUS, RP0         ; ---- Select Bank 0 -----

        return


;-------------------------------------------------------------------------
--
; Subroutine READ_ANALOG_AN0; READ_ANALOG_AN1
;

;-------------------------------------------------------------------------
--
; Subroutine: WaitxTE

WaitxTE
        movwf   Count2      ; [1]

waitxlp
        movlw   D'80'       ; [1]
        movwf   Count       ; [1]
```

```
wait400lp
        nop                     ; [1]
        nop                     ; [1]
        decfsz  Count,F         ; [1]
         goto    wait400lp      ; [2]
;                               --------
;                             20 x 5 = 100us

        decfsz  Count2,F        ; [1]
         goto    waitxlp        ; [2]

        retlw   0               ; [2]

;                           total 2 (call) + W x (100 + 5) + 2 (return)
;                           w = 1 -> 111us @4MHz
;                           w = 2 -> 216us @4MHz


;-----------------------------------------------------------------------
--
; Initialize PICmicro (PIC12F675)
;-----------------------------------------------------------------------
--

INITALIZE

; Disable global interrupts during initialization

        bcf     INTCON, GIE             ; disable global interrupts


;-------------------------------------
; Calibrating the Internal Oscillator (Section 9.2.5.1)

        bsf     STATUS, RP0             ; ---- Select Bank 1 -----

        call    0x3FF                   ; retrieve factory calibration
value
        movwf   OSCCAL                  ; update register with factory
cal value

        bcf     STATUS, RP0             ; ---- Select Bank 0 -----


;-------------------------------------
; GPIO Port (Section 3.0)
;
; Store GPTRIS value defined above into the TRISIO direction register

        bsf     STATUS, RP0             ; ---- Select Bank 1 -----

        movlw   GPTRIS
        movwf   TRISIO                  ; Write to TRISIO register

        bcf     STATUS, RP0             ; ---- Select Bank 0 -----


;-------------------------------------
; Comparator Module (Section 6.0)

        ; CM2:CM0 = 111 - Comparator Off (lowest power)
```

```
        bsf     CMCON, CM2          ; Comparator Mode bit 2
        bsf     CMCON, CM1          ; Comparator Mode bit 1
        bsf     CMCON, CM0          ; Comparator Mode bit 0

        ; VRCON (Register 6-2)
        bsf     STATUS, RP0         ; ---- Select Bank 1 -----

        bcf     VRCON, VREN         ; CVref circuit: powered down, no
Idd drain


        bcf     STATUS, RP0         ; ---- Select Bank 0 -----


;---------------------------------------
; Analog-to-Digital Converter (A/D) Module (Section 7.0) (PIC12F675
Only)
        bcf     ADCON0, ADFM        ; A/D Result Formed: left
justified
        bcf     ADCON0, VCFG        ; Voltage Reference: Vdd
        bcf     ADCON0, ADON        ; ADC is shut-off and consumes no
operating current

        bsf     STATUS, RP0         ; ---- Select Bank 1 -----

        ; select A/D Conversion Clock Source: Fosc/8
        bcf     ANSEL, ADCS2        ; A/D Conversion Clock Select bit
2
        bcf     ANSEL, ADCS1        ; A/D Conversion Clock Select bit
1
        bsf     ANSEL, ADCS0        ; A/D Conversion Clock Select bit
0

        ; select GPIO pins that will be analog inputs: GP0/AN0,
GP1/AN1
        bcf     ANSEL, ANS3         ; Analog Select GP4/AN3: digital
I/O
        bcf     ANSEL, ANS2         ; Analog Select GP2/AN2: digital
I/O
        bcf     ANSEL, ANS1         ; Analog Select GP1/AN1: digital
I/O
        bcf     ANSEL, ANS0         ; Analog Select GP0/AN0: digital
I/O

;       bsf     ANSEL, ANS1         ; Analog Select GP1/AN1: analog
input
;       bsf     ANSEL, ANS0         ; Analog Select GP0/AN0: analog
input

        bcf     STATUS, RP0         ; ---- Select Bank 0 -----
;---------------------------------------
; TIMER1 Module with Gate Control (Section 5.0)
;
; The TIMER1 Control Register (T1CON) is used to enable/disable
TIMER1
; and select various features of the TIMER1 module.

        bcf     T1CON, TMR1ON       ; TIMER1: stopped

        bcf     T1CON, TMR1CS       ; TIMER1 Clock Source Select:
Internal Clock (Fosc/4)
```

```
        bcf     T1CON, NOT_T1SYNC   ; TIMER1 External Clock Input
Sync Control: Syncronize external clock input

        ; T1OSCEN only if INTOSC without CLKOUT oscillator is active,
else ignored
        bcf     T1CON, T1OSCEN      ; LP Oscillator Enable Control:
LP oscillator off

        ; TIMER1 Input Prescale Select: 1:1
        bcf     T1CON, T1CKPS1      ; TIMER1 Input Clock Prescale
Select bit 1
        bcf     T1CON, T1CKPS0      ; TIMER1 Input Clock Prescale
Select bit 0

        ; TMR1GE only if TMR1ON = 1, else ignored
        bcf     T1CON, TMR1GE       ; TIMER1 Gate Enable: on
;---------------------------------------
; Weak Pull-up Register (WPU) (Section 3.2.1)

        bsf     STATUS, RP0         ; ---- Select Bank 1 -----

;    GPIO Pins = xx54x210
        movlw   B'00000000'         ; GP4 pull-up enabled
        movwf   WPU

        bcf     STATUS, RP0         ; ---- Select Bank 0 -----


;---------------------------------------
; OPTION Register (OPTION_REG) (Section 2.2.2.2)

        bsf     STATUS, RP0         ; ---- Select Bank 1 -----

        bcf     OPTION_REG, NOT_GPPU  ; GPIO pull-ups: enabled

        bsf     OPTION_REG, INTEDG  ; Interrupt Edge: on rising edge
of GP2/INT pin

        bcf     OPTION_REG, T0CS    ; TMR0 Clock Source: internal
instruction cycle (CLKOUT)
        bcf     OPTION_REG, T0SE    ; TMR0 Source Edge: increment
low-to-high transition on GP2/T0CKI pin

        bcf     OPTION_REG, PSA     ; Prescaler Assignment: assigned
to TIMER0

        ; TMR0 Prescaler Rate: 1:2
        bcf     OPTION_REG, PS2     ; Prescaler Rate Select bit 2
        bcf     OPTION_REG, PS1     ; Prescaler Rate Select bit 1
        bcf     OPTION_REG, PS0     ; Prescaler Rate Select bit 0

        bcf     STATUS, RP0         ; ---- Select Bank 0 -----


;---------------------------------------
; Interrupt-on-Change Register (IOCB) (Section 3.2.2)

        bsf     STATUS, RP0         ; ---- Select Bank 1 -----

    ; GPIO Pins = xx543210
```

```
;       movlw   b'00011000'
;       movwf   IOCB                    ; Interrupt-on-change enabled:
GP3, GP4
        clrf        IOCB
        bcf     STATUS, RP0             ; ---- Select Bank 0 -----


;--------------------------------------
; Peripheral Interrupt Enable Register (PIE1) (Section 2.2.2.4)
;
; The PIE1 register contains peripheral interrupt enable bits.
;
; Note: The PEIE bit (INTCON<6>) must be set to enable any
;       peripheral interrupt.

        bsf     STATUS, RP0             ; ---- Select Bank 1 -----

        bcf     PIE1, EEIE              ; EE Write Complete Interrupt:
disabled
        bcf     PIE1, ADIE              ; A/D Converter Interrupt
(PIC12F675 Only): disabled
        bcf     PIE1, CMIE              ; Comparator Interrupt: disabled
        bcf     PIE1, TMR1IE            ; TMR1 Overflow Interrupt:
disabled

        bcf     STATUS, RP0             ; ---- Select Bank 0 -----


;--------------------------------------
; Interrupt Control Register (INTCON) (Section 2.2.2.3)
;
; The INTCON register contains enable and disable flag bits for TMR0
; register overflow, GPIO port change and external GP2/INT pin
; interrupts.

        bcf     INTCON, PEIE            ; disable Peripheral Interrupt
Enable bit
        bcf     INTCON, T0IE            ; disable TMR0 Overflow Interrupt
Enable bit
        bcf     INTCON, INTE            ; disable GP2/INT External
Interrupt Enable bit
        bcf     INTCON, GPIE            ; disable Port Change Interrupt
Enable bit

;       bcf     INTCON, GIE             ; disable global interrupts
;-----------------------------------------------------------------------
--
;-----------------------------------------------------------------------
--
; Main Program
;-----------------------------------------------------------------------
--
;-----------------------------------------------------------------------
--

MAIN
            bsf   TXDCOM
        bcf     RFENA               ; Disable Transmitter
        bcf     INTCON, GIE         ; disable global interrupts

Sendinit
```

```
        movlw   0x0A            ;10 table
        movwf   Counttab
        movlw   0x41            ;No. of table 1
        movwf   Numtab
Send
            call        XMIT
            movlw Bytepoll
            movwf       N_temp
            call  TXNextByte
            bcf         RFENA
            call        Receive
            incf       Numtab
            decfsz   Counttab
            goto      Send
            goto      Sendinit

;----------------------------------
; fill in transmission buffer
;----------------------------------

XMIT

        bsf     RFENA           ; Enable Transmitter
        movlw   D'6'
        call    WaitxTE

         movf   Numtab,w
        movwf  N_temp

;----------------------------------
; Transmission Loop
;----------------------------------

TXLoop

        ; send preamble (50% duty cycle)

Preamble
        movlw   PREAMB
        movwf   BitCount        ; init number of preamble bits

PreL
        bsf     TXD             ; ON
        movlw   1
        call    WaitxTE         ; delay
        bcf     TXD             ; OFF
        movlw   1
        call    WaitxTE         ; delay

        decfsz  BitCount,F      ; loop
        goto    PreL

; sync pause

TXloop
        movlw   D'10'           ; Theader = 10 x Te
        call    WaitxTE

TXNextByte
        movlw   D'8'
        movwf   BitCount
```

```
TXNextBit
        rrf     N_temp,1
        btfsc   STATUS,C
        goto    ONE


ZERO
        movlw   2                       ;
        movwf   TimeHi                  ;    +---+---+    +--
        movlw   1                       ;    |       |    |
        movwf   TimeLo                  ;----+       +----+
        goto    Trasm_BIT               ;    |  2Te    Te |

ONE
        movlw   1                       ;
        movwf   TimeHi                  ;    +---+        +--
        movlw   2                       ;    |   |        |
        movwf   TimeLo                  ;----+   +---+---+
                                        ;    | Te   2Te  |
Trasm_BIT
        bsf     TXD                     ; ON
        movf    TimeHi,W
        call    WaitxTE

        bcf     TXD                     ; OFF
        movf    TimeLo,W
        call    WaitxTE

        decfsz  BitCount,F
        goto    TXNextBit               ; loop on bits
; guard time
        movlw   TGUARD
        call    WaitxTE
         return
;-----------------------------------------------------------
Receive
     movlw   H'40'
           movwf   CountRe2         ; [1]

waitxlp_
       movlw   D'20'            ; [1]
       movwf   CountRe          ; [1]

wait400lp_
           nop                      ; [1]
           btfsc RXD
           goto  RXkeeloq
Nopre
           decfsz  CountRe,F        ; [1]
       goto    wait400lp_  ; [2]
;                          --------
;                        79 x 5 = 395us

       decfsz  CountRe2,F     ; [1]
       goto    waitxlp_     ; [2]
     return
RXkeeloq
;preamble
           btfss RXD
```

```
            goto  Nopre

            movlw D'16'
            movwf Countpre
            movlw D'1'
            call  WaitxTEhalf
chpre
            btfss RXD

            goto  Nopre
            movlw D'1'
            call  WaitxTE
            btfsc RXD

            goto  Nopre
            movlw D'1'
            call  WaitxTE

            decfsz      Countpre,f
            goto  chpre
            movlw D'8'
            call  WaitxTE

            call  detect
            movf  RX_BUFFER,W
            movwf DATADDR

            call  detect
            movf  RX_BUFFER,W
            movwf DATMENU

            call  detect
            movf  RX_BUFFER,W
            movwf DATNUM

            movf  Numtab,W                    ;move ACK0 Code to
W(have)
            subwf DATADDR,0

            BZ          YES
            goto  RETFRX
YES
            movf  DATADDR,W
            call  TX
            movf  DATMENU,W
            call  TX
            movf  DATNUM,W
            call  TX
            btfsc RXDCOM
            goto  $-1
            call  RX9600B
            movlw   D'12'
                  call    WaitxTE

            movlw ACKCOM                      ;move ACK0 Code to W(have)
            subwf RX_BUFFER,0
            BZ          ACK

            movlw EMPCOM                            ;move ACK0 Code to
W(have)
            subwf RX_BUFFER,0
```

```
            BZ          addrG
            goto  RETFRX
ACK
            call  XMIT

            movlw ByteACK
            movwf       N_temp
            call  TXNextByte

            call  XMIT
            movlw ByteACK
            movwf       N_temp
            call  TXNextByte
;*********************************

            goto  RETFRX
addrG
            bsf     RFENA           ; Enable Transmitter

            movlw   D'6'
                  call    WaitxTE


            movlw   GROUPADDR
                  movwf  N_temp
            call  Preamble
            movf  DATMENU,W
            movwf       N_temp
            call  TXNextByte
;*************************************
          movlw   GROUPADDR
        movwf  N_temp
            call  Preamble
            movf  DATMENU,W
            movwf       N_temp
            call  TXNextByte
;*************************************
            goto  RETFRX

;return from rxkeeloq
RETFRX
            bcf   RFENA
            return

;-------------------------------------------------------------------------
---

;waitheader
detect
            btfss RXD
            goto  $-1

            movlw .8
            movwf       Countrx
Keeloop
            movlw D'1'
            call  WaitxTEhalf
            movlw D'1'
            call  WaitxTE
            bsf     STATUS,C
            btfsc RXD
```

```
            bcf         STATUS,C
            rrf         RX_BUFFER,1
            decfsz      Countrx,1
            goto  nonturn
            goto  turn
nonturn
            movlw D'1'
            call  WaitxTE
            btfss RXD
            goto  $-1
            goto  Keeloop
turn
            movlw D'1'
            call  WaitxTEhalf
            movlw D'1'
            call  WaitxTE
            btfsc RXD
            goto  $-1
            return
;-------------------------------------------------------------------
WaitxTEhalf
        movwf   Count2      ; [1]

waitxh
        movlw   D'40'       ; [1]
        movwf   Count       ; [1]

wait400h
        nop                 ; [1]
        nop                 ; [1]
        decfsz  Count,F     ; [1]
         goto    wait400h   ; [2]
;                           --------
;                          79 x 5 = 395us
        decfsz  Count2,F    ; [1]
         goto    waitxh     ; [2]
        retlw   0           ; [2]
;-------------------------------------------------------------------
TX
      movwf TX_BUFFER
      movlw .8
      movwf Counttx1
      call  DELAY_TX
      bcf         TXDCOM
      call  DELAY_TX
LOOP_SEND
      rrf         TX_BUFFER,1
      btfsc STATUS,C
      goto  L2
LOGIC_0
      bcf         TXDCOM
LOGIC_1
      call  DELAY_TX
      decfsz      Counttx1,1
      goto  LOOP_SEND
      bsf         TXDCOM
      call  DELAY_TX
      return
L2
      bsf         TXDCOM
      goto  LOGIC_1
```

```
DELAY_TX
      movlw .30
      movwf Count
      decfsz      Count,1
      goto  $-1
      return
;-------------------------------------------------------------------
--
RX9600B
      call  DELAY_HALF
      movlw .9
      movwf       Countrx
RXLOOP2
      call  DELAY_BIT
      bcf     STATUS,C
      btfsc RXDCOM
      bsf         STATUS,C
      rrf         RX_BUFFER,1
      decfsz      Countrx,1
      goto  RXLOOP2
      rlf         RX_BUFFER,1
      BTFSS RXDCOM
      goto  $-1
      return
DELAY_BIT
      movlw .30
      movwf Count
      decfsz      Count,1
      goto  $-1
      return

DELAY_HALF
      movlw D'15'
      movwf Count
      decfsz      Count,1
      goto  $-1
      return
;-------------------------------------------------------------------
--
; Data EEPROM Memory (Section 8.0)
;
; PIC12F629/675 devices have 128 bytes of data EEPROM with address
; range 0x00 to 0x7F.
       ; Initialize Data EEPROM Memory locations
;       ORG 0x2100
;       DE  0x00, 0x01, 0x02, 0x03

;-------------------------------------------------------------------
; Calibrating the Internal Oscillator (Section 9.2.5.1)
; Oscillator Calibration Register (OSCCAL) (Section 2.2.2.7)
;
; The below statements are placed here so that the program can be
; simulated with MPLAB SIM.  The programmer (PICkit or PROMATE II)
; will save the actual OSCCAL value in the device and restore it.
; The value below WILL NOT be programmed into the device.
       org     0x3ff
       retlw    0x80
;-------------------------------------------------------------------
       end                          ; end of program directive
;-------------------------------------------------------------------
```

โปรแกรมแอสเซมบลีของ PIC-12F675 ในส่วนเครื่องควบคุมการสั่งอาหาร

โปรแกรมภาษาแอสเซมบลีของ PIC-12F675 ในส่วนเครื่องสั่งอาหาร

```
    list        p=12f675        ; list directive to define processor
    #include    <p12f675.inc>   ; processor specific variable
definitions

    errorlevel  -302            ; suppress message 302 from list file


    ;__CONFIG    _CPD_OFF & _CP_OFF & _BODEN_OFF & _MCLRE_OFF &
_PWRTE_ON & _WDT_OFF & _INTRC_OSC_NOCLKOUT
      __CONFIG    0x3184

    cblock  0x20

        w_temp                      ; used for context saving
        status_temp                 ; used for context saving

        TEMP                        ; General Purpose Temporary
register

            RX_BUFFER
            Countrx
        Count
        Count2
        BitCount
        TimeHi
        TimeLo

        FuncBits                    ; Function Bits
        SaveFuncBits                ; Function Bits save register
            Countpre

        PBCntr
        PBCntr1
            TX_BUFFER
            Counttx1
            DATmenu
            DATnum
            N_temp
            DAT_1
            DAT_2
            empty_1
            empty_2
            empty_3
            empty_4
            check
    endc

#define RXDCOM    GPIO, 1             ; (Input) Receive data from 51
#define TXDCOM    GPIO, 4             ; (Output ) Transmit Data to 51
#define TXD     GPIO, 2             ; (Output) Transmit Data
#define PB3     GPIO, 3             ; (Input Only) Push button switch
GP3
#define RX_PIN  GPIO, 0          ; (Input) Receive data from wireless
#define RFENA   GPIO, 5             ; (Output) RF Enable
    ; Setting a TRISIO bit (= 1) will make the corresponding GPIO pin
an input.
    ; Clearing a TRISIO bit (= 0)will make the corresponding GPIO pin
an output. The exception is
```

```
    ; GP3, which is input only and its TRIS bit will always read as a
'1'.

    ; GPIO Pins = xx543210

#define NAK_051  B'01000001'          ;empty
#define NAK_151  B'01000011'          ;error
#define ACK_51   B'01000101'          ;success

#define   GPTRIS B'00001011'
#define TAB     B'01000010'                   ;Number of Table
#define Poll      B'10000000'                 ;code of Poll
#define ACK    B'10001000'                    ;YES have
#define Group    B'11111111'                  ;Number of group address
#define TGUARD  D'8'                        ; 46 X TE
#define PREAMB  D'16'                        ; Preamble length = 16 pulses
    ; Program Memory Organization (Section 2.1)

        ORG     0x000                       ; RESET Vector

        nop                                 ; for ICD use
        goto    INITALIZE                   ; goto INITALIZE


        ORG     0x004                       ; Interrupt Vector
            movwf   w_temp                    ; save W register
        swapf   STATUS, W                   ; swap status to be saved into W
        bcf     STATUS, RP0                 ; ---- Select Bank 0 -----
        movwf   status_temp                 ; save STATUS register

;--------------------------------------
; Interrupt Service Routine (ISR)
;
; Description:
;
;--------------------------------------

; Interrupt-on-change (Section 3.2.2 and 9.4.3)
;
; An input change on GPIO change sets the GPIF bit. The interrupt can
; be enabled/disabled be setting/clearing the GPIE bit. Individual
pins
; are configured through the IOC register (see INITIALIZATION below).
;
; Clear the IOC interrupt by:
;  a) Any read or write of GPIO. This will end the mismatch condition.
;  b) Clear the flag bit GPIF


;       movfw   GPIO                    ; read GPIO
           btfsc RXDCOM
           goto  $-1
           call  RX9600B
           movf  RX_BUFFER,W
           movwf DATmenu

           btfsc RXDCOM
           goto  $-1
           call  RX9600B
           movf  RX_BUFFER,W
           movwf DATnum
```

```
                movf   DATmenu,W
                subwf  empty_1,0
                bz             NO1

                movf   DATmenu,W
                subwf  empty_2,0
                bz             NO1

                movf   DATmenu,W
                subwf  empty_3,0
                bz             NO1

                movf   DATmenu,W
                subwf  empty_4,0
                bz             NO1

Waitpoll
                call   chPoll

                movf   RX_BUFFER,W
                movwf  DAT_1

                call   detect
                movf   RX_BUFFER,W
                movwf  DAT_2

                movlw  TAB
                subwf  DAT_1,0
                BZ             ChDat_2
                goto   Waitpoll
ChDat_2
                movlw  Poll
                subwf  DAT_2,0
                BZ             sendmas
                goto   Waitpoll

sendmas

                movlw  TAB
                call   XMIT

                movf   DATmenu,W

                movwf          N_temp
                call   TXNextByte

                movf   DATnum,W

                movwf          N_temp
                call   TXNextByte


                bcf            RFENA

ACK_
                call   chPoll
                movf   RX_BUFFER,W
                movwf  DAT_1
```

```
                call  detect
                movf  RX_BUFFER,W
                movwf DAT_2

                movlw TAB
                subwf DAT_1,0
                BZ          chACK51

ChACK
                movlw Group
                subwf DAT_1,0
                BZ          NAK051
                goto  NAK151
chACK51
                movlw ACK
                subwf DAT_2,0
                BZ          ACK51
                goto  NAK151

ACK51
                movlw ACK_51
                call  TX
                goto  RETINT
NAK151
                movlw NAK_151                          ;error
                call  TX
                goto  RETINT
NAK051
;*********************************
                movlw H'ff'
                subwf empty_1,0
                bz          one_1
                goto  two1
one_1
                movf  DAT_2,W
                movwf empty_1
                goto  NO1
two1
                movlw H'ff'
                subwf empty_2,0
                bz          two_1
                goto  three1
two_1
                movf  DAT_2,W
                movwf empty_2
                goto  NO1
three1
                movlw H'ff'
                subwf empty_3,0
                bz          three_1
                goto  four1
three_1
                movf  DAT_2,W
                movwf empty_3
                goto  NO1
four1
                movlw H'ff'
                subwf empty_2,0
                bz          four_1
                goto  NO1
```

```
four_1
            movf  DAT_2,W
            movwf empty_4
;**********************************


NO1

            movlw NAK_051                                   ;empty
            call  TX

RETINT
            clrf  check
        bcf     INTCON, GPIF


;----------------------------------------

        swapf   status_temp, W      ; swap status_temp into W, sets
bank to original state
        movwf   STATUS              ; restore STATUS register
        swapf   w_temp, F
        swapf   w_temp, W           ; restore W register
;           goto  main
            retfie
INITALIZE

; Disable global interrupts during initialization

        bcf     INTCON, GIE         ; disable global interrupts


;----------------------------------------
; Calibrating the Internal Oscillator (Section 9.2.5.1)
; Oscillator Calibration Register (Section 2.2.2.7)
;
; A calibration instruction is programmed into the last location of
; program memory. This instruction is a RETLW XX, where the literal
is
; the calibration value. The literal is placed in the OSCCAL register
; to set the calibration of the internal oscillator.

        bsf     STATUS, RP0         ; ---- Select Bank 1 -----

        call    0x3FF               ; retrieve factory calibration
value
        movwf   OSCCAL              ; update register with factory
cal value

        bcf     STATUS, RP0         ; ---- Select Bank 0 -----


;----------------------------------------
; GPIO Port (Section 3.0)
;
; Store GPTRIS value defined above into the TRISIO direction register

        bsf     STATUS, RP0         ; ---- Select Bank 1 -----

        movlw   GPTRIS
        movwf   TRISIO              ; Write to TRISIO register
```

```
        bcf     STATUS, RP0         ; ---- Select Bank 0 -----

;----------------------------------------
; Comparator Module (Section 6.0)
;
; The PIC12F629/675 devices have one analog comparator. The inputs to
; the comparator are multiplexed with the GP0 and GP1 pins. There is
; an on-chip Comparator Voltage Reference that can also be applied to
; an input of the comparator. In addition, GP2 can be configured as
; the comparator output. The Comparator Control Register (CMCON)
; contains bits to control the comparator. The Voltage Reference
; Control Register (VRCON) controls the voltage reference module.


        ; Comparator Configuration (Figure 6-2)
;       bcf     CMCON, CINV         ; Comparator Output Inversion:
not inverted
;       bcf     CMCON, COUT         ; Comparator Output bit: Vin+ <
Vin-
;       bcf     CMCON, CIS          ; Comparator Input Switch: Vin-
connectos to Cin-

        ; CM2:CM0 = 111 - Comparator Off (lowest power)
        bsf     CMCON, CM2          ; Comparator Mode bit 2
        bsf     CMCON, CM1          ; Comparator Mode bit 1
        bsf     CMCON, CM0          ; Comparator Mode bit 0

        ; VRCON (Register 6-2)
        bsf     STATUS, RP0         ; ---- Select Bank 1 -----

        bcf     VRCON, VREN         ; CVref circuit: powered down, no
Idd drain

;       bcf     VRCON, VRR          ; CVref Range Selection: High
Range

;       bcf     VRCON, VR3          ; CVref value selection bit 3
;       bcf     VRCON, VR2          ; CVref value selection bit 2
;       bcf     VRCON, VR1          ; CVref value selection bit 1
;       bcf     VRCON, VR0          ; CVref value selection bit 0

        bcf     STATUS, RP0         ; ---- Select Bank 0 -----
;----------------------------------------
;Analog-to-Digital Converter (A/D) Module (Section 7.0) (PIC12F675
Only)
;
;The analog-to-digital converter (A/D) allows conversion of an analog
;input signal to a 10-bit binary representation of that signal. The
;PIC12F675 has four analog inputs multiplexed into one sample and
hold
;circuit. There are two registers to control the functions of the A/D
;module:
; A/D Control Regester (ADCON0)
; Analog Select Register (ANSEL)
;
;Note:When using GPIO pins as analog inputs,ensure the TRISIO
register
;     bits are set (=1) for input.
```

```
            bcf         ADCON0,ADFM             ;A/D Result Formed:
left justified
            bcf         ADCON0,VCFG             ;Voltage Reference:
Vdd
            bcf         ADCON0,ADON             ;ADC is shut-off and
consumes no operating current

            bsf         STATUS,RP0              ;----Select Bank 1---
-

;select A/D Conversion Clock Source: Fosc/8
            bcf         ANSEL,ADCS2             ;A/D Conversion Clock
Select bit 2
            bcf         ANSEL,ADCS1             ;A/D Conversion Clock
Select bit 1
            bsf         ANSEL,ADCS0             ;A/D Conversion Clock
Select bit 0
;select GPIO pins that will be analog inputs: GP0/AN0,GP1/AN1
            bcf         ANSEL,ANS3              ;Analog Select
GP4/AN3: digital I/O
            bcf   ANSEL,ANS2                ;Analog Select GP2/AN2:
digital I/O
            bcf         ANSEL,ANS1              ;Analog Select
GP1/AN1: digital I/O
            bcf         ANSEL,ANS0              ;Analog Select
GP0/AN0: digital I/O

            bcf   STATUS,RP0                ;----Select Bank 0----

;--------------------------------------
; TIMER1 Module with Gate Control (Section 5.0)
;
; The TIMER1 Control Register (T1CON) is used to enable/disable
TIMER1
; and select various features of the TIMER1 module.

        bcf     T1CON, TMR1ON       ; TIMER1: stopped

        bcf     T1CON, TMR1CS       ; TIMER1 Clock Source Select:
Internal Clock (Fosc/4)

        bcf     T1CON, NOT_T1SYNC   ; TIMER1 External Clock Input
Sync Control: Syncronize external clock input

        ; T1OSCEN only if INTOSC without CLKOUT oscillator is active,
else ignored
        bcf     T1CON, T1OSCEN      ; LP Oscillator Enable Control:
LP oscillator off

        ; TIMER1 Input Prescale Select: 1:1
        bcf     T1CON, T1CKPS1      ; TIMER1 Input Clock Prescale
Select bit 1
        bcf     T1CON, T1CKPS0      ; TIMER1 Input Clock Prescale
Select bit 0

        ; TMR1GE only if TMR1ON = 1, else ignored
        bcf     T1CON, TMR1GE       ; TIMER1 Gate Enable: on


;--------------------------------------
; Weak Pull-up Register (WPU) (Section 3.2.1)
```

```
;
; Each of the GPIO pins, except GP3, has an individually configurable
; weak internal pull-up. Control bits WPUx enable or disable each
; pull-up. Refer to Register 3-1. Each weak pull-up is automatically
; turned off when the port pin is configured as an output. The pull-
ups
; are disabled on a Power-on Reset by the NOT_GPPU bit (see
OPTION_REG below).

        bsf     STATUS, RP0             ; ---- Select Bank 1 -----

;     GPIO Pins = xx54x210
        movlw   B'00000000'             ; GP4 pull-up enabled
        movwf   WPU

        bcf     STATUS, RP0             ; ---- Select Bank 0 -----

;--------------------------------------
; OPTION Register (OPTION_REG) (Section 2.2.2.2)
; TIMER0 Module (Section 4.0)
;
; The OPTION_REG contains control bits to configure:
;   Weak pull-ups on GPIO (see also WPU Register above)
;   External GP2/INT interrupt
;   TMR0
;   TMR0/WDT prescaler

        bsf     STATUS, RP0             ; ---- Select Bank 1 -----

        bcf     OPTION_REG, NOT_GPPU    ; GPIO pull-ups: enabled

        bsf     OPTION_REG, INTEDG      ; Interrupt Edge: on rising edge
of GP2/INT pin

        bcf     OPTION_REG, T0CS        ; TMR0 Clock Source: internal
instruction cycle (CLKOUT)
        bcf     OPTION_REG, T0SE        ; TMR0 Source Edge: increment
low-to-high transition on GP2/T0CKI pin

        bcf     OPTION_REG, PSA         ; Prescaler Assignment: assigned
to TIMER0

        ; TMR0 Prescaler Rate: 1:2
        bcf     OPTION_REG, PS2         ; Prescaler Rate Select bit 2
        bcf     OPTION_REG, PS1         ; Prescaler Rate Select bit 1
        bcf     OPTION_REG, PS0         ; Prescaler Rate Select bit 0

        bcf     STATUS, RP0             ; ---- Select Bank 0 -----

;--------------------------------------
; Interrupt-on-Change Register (IOCB) (Section 3.2.2)
;
; Each of the GPIO pins is individually configurable as an interrupt-
; on-change pin. Control bits IOCBx enable or disable the interrupt
; function for each pin. Refer to Register 3-2. The interrupt-on-
change
; is disabled on a Power-on Reset.
;
; Note: Global interrupt enables (GIE and GPIE) must be enabled for
```

```
;       individual interrupts to be recognized.

        bsf     STATUS, RP0             ; ---- Select Bank 1 -----

    ; GPIO Pins = xx543210
          movlw   b'00000010'

;       movlw   b'00000001'
        movwf   IOCB                    ; Interrupt-on-change enabled:
GP3, GP4

        bcf     STATUS, RP0             ; ---- Select Bank 0 -----


;--------------------------------------
; Peripheral Interrupt Enable Register (PIE1) (Section 2.2.2.4)
;
; The PIE1 register contains peripheral interrupt enable bits.
;
; Note: The PEIE bit (INTCON<6>) must be set to enable any
;       peripheral interrupt.

        bsf     STATUS, RP0             ; ---- Select Bank 1 -----

        bcf     PIE1, EEIE              ; EE Write Complete Interrupt:
disabled
        bcf     PIE1, ADIE              ; A/D Converter Interrupt
(PIC12F675 Only): disabled
        bcf     PIE1, CMIE              ; Comparator Interrupt: disabled
        bcf     PIE1, TMR1IE            ; TMR1 Overflow Interrupt:
disabled

        bcf     STATUS, RP0             ; ---- Select Bank 0 -----


;--------------------------------------
; Interrupt Control Register (INTCON) (Section 2.2.2.3)
;
; The INTCON register contains enable and disable flag bits for TMR0
; register overflow, GPIO port change and external GP2/INT pin
; interrupts.

        bcf     INTCON, PEIE            ; disable Peripheral Interrupt
Enable bit
        bcf     INTCON, T0IE            ; disable TMR0 Overflow Interrupt
Enable bit
        bcf     INTCON, INTE            ; disable GP2/INT External
Interrupt Enable bit
        bsf     INTCON, GPIE            ; enable Port Change Interrupt
Enable bit

;       bcf     INTCON, GIE             ; disable global interrupts


;-----------------------------------------------------------------------
--
;-----------------------------------------------------------------------
--
; Main Program
;-----------------------------------------------------------------------
--
```

```
;-----------------------------------------------------------------
--

        bcf         INTCON,GIE          ;Disable global interrupt
        bsf   TXDCOM            ; Disable Transmitter
        bcf         TXD
        bcf         RFENA
        movlw H'ff'
        movwf empty_1
        movwf empty_2
        movwf empty_3
        movwf empty_4
        movwf check

        clrf  Count
        clrf  Count2
        decfsz      Count,1
        goto  $-1
        decfsz      Count2,1
        goto  $-3

        clrf  Count
        clrf  Count2
        decfsz      Count,1
        goto  $-1
        decfsz      Count2,1
        goto  $-3

MAIN
        bsf         INTCON,GIE          ;Enable global interrupt
        bcf         TXD
        bsf         TXDCOM
        call  DELAY_TX


main1

             call  chPoll
             movf  RX_BUFFER,W
             movwf DAT_1

             call  detect
             movf  RX_BUFFER,W
             movwf DAT_2

             movlw H'ff'
             subwf check,0
             bz          ch_
             goto  NO
ch_
             movlw Group
             subwf DAT_1,0
             BZ          YES
             goto  NO
YES
             movf  empty_1,W
             subwf DAT_2,0
             bz          NO
             movf  empty_2,W
             subwf DAT_2,0
```

```
                bz          NO
                movf  empty_3,W
                subwf DAT_2,0
                bz          NO
                movf  empty_4,W
                subwf DAT_2,0
                bz          NO


                movlw H'ff'
                subwf empty_1,0
                bz          one_
                goto  two
one_
                movf  DAT_2,W
                movwf empty_1
                goto  NO

two
                movlw H'ff'
                subwf empty_2,0
                bz          two_
                goto  three
two_
                movf  DAT_2,W
                movwf empty_2
                goto  NO
three
                movlw H'ff'
                subwf empty_3,0
                bz          three_
                goto  four
three_
                movf  DAT_2,W
                movwf empty_3
                goto  NO
four
                movlw H'ff'
                subwf empty_2,0
                bz          four_
                goto  NO
four_
                movf  DAT_2,W
                movwf empty_4
                goto  NO

NO
      MOVLW H'ff'
      movwf check
      GOTO  main1

TX
      movwf TX_BUFFER
      movlw .8
      movwf Counttx1
      call  DELAY_TX
      bcf         TXDCOM
      call  DELAY_TX
LOOP_SEND
      rrf         TX_BUFFER,1
      btfsc STATUS,C
```

```
        goto  L2
LOGIC_0
        bcf         TXDCOM
LOGIC_1
        call  DELAY_TX
        decfsz      Counttx1,1
        goto  LOOP_SEND
        bsf         TXDCOM
        call  DELAY_TX

        return

L2
        bsf         TXDCOM
        goto  LOGIC_1

;************************************************
LOOP
    bsf     RFENA           ; Enable Transmitter

        clrf  Count
        clrf  Count2
        decfsz      Count,1
        goto  $-1
        decfsz      Count2,1
        goto  $-3

        clrf  Count
        clrf  Count2
        decfsz      Count,1
        goto  $-1
        decfsz      Count2,1
        goto  $-3
        goto  MAIN

DELAY_TX
        movlw .30
        movwf Count
        decfsz      Count,1
        goto  $-1
        return
;-------------------------------------------------------------------------
--
chPoll:     btfsc RX_PIN
            goto  chPoll
            goto  preamble


preamble

            btfss RX_PIN

            goto  $-1

            movlw D'16'
            movwf Countpre
            movlw D'1'
            call  WaitxTEhalf

chpre
```

```
;           bcf         TXD
            btfss RX_PIN

            goto  chPoll
            movlw D'1'
            call  WaitxTE
;           bsf         TXD
            btfsc RX_PIN

            goto  chPoll
            movlw D'1'
            call  WaitxTE
            decfsz      Countpre,f
            goto  chpre

            movlw D'8'
            call  WaitxTE

detect
            btfss RX_PIN
            goto  $-1
            movlw .8
            movwf       Countrx
Keeloop
            movlw D'1'
            call  WaitxTEhalf
            movlw D'1'
            call  WaitxTE
            bsf     STATUS,C
            btfsc RX_PIN
            bcf         STATUS,C
            rrf         RX_BUFFER,1
            decfsz      Countrx,1
            goto  nonturn
            goto  turn
nonturn
            movlw D'1'
            call  WaitxTE
            btfss RX_PIN
            goto  $-1
            goto  Keeloop
turn
            movlw D'1'
            call  WaitxTEhalf
            movlw D'1'
            call  WaitxTE
            btfsc RX_PIN
            goto  $-1
            return
WaitxTE
        movwf   Count2      ; [1]

waitx
        movlw   D'80'       ; [1]
        movwf   Count       ; [1]

wait400
        nop                 ; [1]
        nop                 ; [1]
        decfsz  Count,F     ; [1]
        goto    wait400  ; [2]
```

```
;                                --------
;                               79 x 5 = 395us

        decfsz  Count2,F     ; [1]
         goto    waitx      ; [2]

        retlw   0            ; [2]

;                           total 2 (call) + W x (395 + 5) + 2 (return)
;                           w = 1 -> 406us @4MHz
;                           w = 2 -> 806us @4MHz
WaitxTEhalf
        movwf   Count2       ; [1]

waitxh
        movlw   D'40'        ; [1]
        movwf   Count        ; [1]

wait400h
        nop                  ; [1]
        nop                  ; [1]
        decfsz  Count,F      ; [1]
         goto    wait400h    ; [2]
;                            --------
;                           79 x 5 = 395us

        decfsz  Count2,F     ; [1]
         goto    waitxh      ; [2]

        retlw   0            ; [2]

;                           total 2 (call) + W x (395 + 5) + 2 (return)
;                           w = 1 -> 406us @4MHz
;                           w = 2 -> 806us @4MHz
;-------------------------------------------------------------------------
--
XMIT
        movwf  N_temp
;increase
        movlw   D'12'
        call    WaitxTE
;increase
        bsf     RFENA              ; Enable Transmitter
            movlw   D'6'
        call    WaitxTE



;----------------------------------
; Transmission Loop
;----------------------------------

TXLoop

        ; send preamble (50% duty cycle)

Preamble
        movlw   PREAMB
        movwf   BitCount           ; init number of preamble bits

PreL
```

```
        bsf     TXD             ; ON
        movlw   1
        call    WaitxTE         ; delay
        bcf     TXD             ; OFF
        movlw   1
        call    WaitxTE         ; delay

        decfsz  BitCount,F      ; loop
        goto    PreL

; sync pause

TXloop
        movlw   D'10'           ; Theader = 10 x Te
        call    WaitxTE

TXNextByte
        movlw   D'8'
        movwf   BitCount

TXNextBit
        rrf     N_temp,1
        btfsc   STATUS,C
        goto    ONE


ZERO
        movlw   2               ;
        movwf   TimeHi          ;   +---+---+     +--
        movlw   1               ;   |       |     |
        movwf   TimeLo          ;---+       +-----+
        goto    Trasm_BIT       ;   |  2Te    Te  |

ONE
        movlw   1               ;
        movwf   TimeHi          ;   +---+         +--
        movlw   2               ;   |   |         |
        movwf   TimeLo          ;---+   +---+-----+
                                ;   |  Te   2Te   |
Trasm_BIT
        bsf     TXD             ; ON
        movf    TimeHi,W
        call    WaitxTE

        bcf     TXD             ; OFF
        movf    TimeLo,W
        call    WaitxTE

        decfsz  BitCount,F
        goto    TXNextBit       ; loop on bits

        movlw   TGUARD
        call    WaitxTE
          return
;-------------------------------------------------------------------------
-------------
RX9600B
      call  DELAY_HALF
      movlw .9
      movwf       Countrx
RXLOOP2
```

```
        call   DELAY_BIT
        bcf      STATUS,C
;       movf   GPIO,W
        btfsc RXDCOM
        bsf          STATUS,C
        rrf          RX_BUFFER,1
        decfsz       Countrx,1
        goto   RXLOOP2
        rlf          RX_BUFFER,1
        BTFSS RXDCOM
        goto   $-1
        return
DELAY_BIT
        movlw .30
        movwf Count
        decfsz       Count,1
        goto   $-1
        return

DELAY_HALF
        movlw D'15'
        movwf Count
        decfsz       Count,1
        goto   $-1
        return
;-----------------------------------------------------------------
; Data EEPROM Memory (Section 8.0)
;
; PIC12F629/675 devices have 128 bytes of data EEPROM with address
; range 0x00 to 0x7F.

        ; Initialize Data EEPROM Memory locations

;       ORG 0x2100
;       DE  0x00, 0x01, 0x02, 0x03


;-----------------------------------------------------------------
--
; Calibrating the Internal Oscillator (Section 9.2.5.1)
; Oscillator Calibration Register (OSCCAL) (Section 2.2.2.7)
;
; The below statements are placed here so that the program can be
; simulated with MPLAB SIM.  The programmer (PICkit or PROMATE II)
; will save the actual OSCCAL value in the device and restore it.
; The value below WILL NOT be programmed into the device.

        org     0x3ff
        retlw    0x80

;-----------------------------------------------------------------
--
        end                             ; end of program directive
;-----------------------------------------------------------------
--
```

ไปรแกรมภาษาแอสเซมบลีของ PIC-12F675 ในส่วนครัว

```
          list        p=12f675        ; list directive to define
processor
          #include    <p12f675.inc>   ; processor specific variable
definitions

          errorlevel  -302            ; suppress message 302 from
list file
          ;__CONFIG     _CPD_OFF & _CP_OFF & _BODEN_OFF & _MCLRE_OFF &
_PWRTE_ON & _WDT_OFF & _INTRC_OSC_NOCLKOUT
            __CONFIG    0x3184

          cblock  0x20

              w_temp                      ; used for context saving
              status_temp                 ; used for context saving

              TEMP                        ; General Purpose Temporary
register

                  RX_BUFFER
                  Countrx
              Count
              Count2
              BitCount
              TimeHi
              TimeLo

              FuncBits                    ; Function Bits
              SaveFuncBits                ; Function Bits save
register
                  Countpre

              PBCntr
              PBCntr1
                  TX_BUFFER
                  Counttx1
                  DAT_1
                  DAT_2
                  DAT_3
                  DAT_addr
          endc

      #define RXDCOM    GPIO, 1             ; (Input) Receive data
from com
      #define TXDCOM    GPIO, 4             ; (Output ) Transmit Data
to com
      #define TXD     GPIO, 2             ; (Output) Transmit Data
      #define PB3     GPIO, 3             ; (Input Only) Push button
switch GP3
      #define RX_PIN  GPIO, 0           ; (Input) Receive data from
wireless
      #define RFENA   GPIO, 5             ; (Output) RF Enable
          ; Setting a TRISIO bit (= 1) will make the corresponding
GPIO pin an input.
          ; Clearing a TRISIO bit (= 0)will make the corresponding
GPIO pin an output. The exception is
          ; GP3, which is input only and its TRIS bit will always
read as a '1'.
```

```
    ; GPIO Pins = xx543210
#define GPTRIS  B'00001011'
#define TAB     B'01000001'
#define NAK       B'00011001'
#define Poll    B'10000000'                ;YES have
#define ACK     B'10001000'                ;NO
#define     Group B'11111111'              ;Group address
    ; Program Memory Organization (Section 2.1)

        ORG     0x000                      ; RESET Vector

        nop                                ; for ICD use
        goto    INITALIZE                  ; goto INITALIZE


        ORG     0x004                      ; Interrupt Vector
            retfie
INITALIZE

; Disable global interrupts during initialization
        bcf     INTCON, GIE                ; disable global interrupts
;-----------------------------------------
; Calibrating the Internal Oscillator (Section 9.2.5.1)
; Oscillator Calibration Register (Section 2.2.2.7)
;
; A calibration instruction is programmed into the last
location of
; program memory. This instruction is a RETLW XX, where the
literal is
; the calibration value. The literal is placed in the OSCCAL
register
; to set the calibration of the internal oscillator.

        bsf     STATUS, RP0                ; ---- Select Bank 1 -----

        call    0x3FF                      ; retrieve factory
calibration value
        movwf   OSCCAL                     ; update register with
factory cal value

        bcf     STATUS, RP0                ; ---- Select Bank 0 -----


;-----------------------------------------
; GPIO Port (Section 3.0)
;
; Store GPTRIS value defined above into the TRISIO direction
register

        bsf     STATUS, RP0                ; ---- Select Bank 1 -----

        movlw   GPTRIS
        movwf   TRISIO                     ; Write to TRISIO register

        bcf     STATUS, RP0                ; ---- Select Bank 0 -----


;-----------------------------------------
; Comparator Module (Section 6.0)
;
```

```
      ; The PIC12F629/675 devices have one analog comparator. The
inputs to
      ; the comparator are multiplexed with the GP0 and GP1 pins.
There is
      ; an on-chip Comparator Voltage Reference that can also be
applied to
      ; an input of the comparator. In addition, GP2 can be
configured as
      ; the comparator output. The Comparator Control Register (CMCON)
      ; contains bits to control the comparator. The Voltage
Reference
      ; Control Register (VRCON) controls the voltage reference
module.


            ; Comparator Configuration (Figure 6-2)
      ;     bcf     CMCON, CINV         ; Comparator Output
Inversion: not inverted
      ;     bcf     CMCON, COUT         ; Comparator Output bit:
Vin+ < Vin-
      ;     bcf     CMCON, CIS          ; Comparator Input Switch:
Vin- connectos to Cin-

            ; CM2:CM0 = 111 - Comparator Off (lowest power)
            bsf     CMCON, CM2          ; Comparator Mode bit 2
            bsf     CMCON, CM1          ; Comparator Mode bit 1
            bsf     CMCON, CM0          ; Comparator Mode bit 0

            ; VRCON (Register 6-2)
            bsf     STATUS, RP0         ; ---- Select Bank 1 -----

            bcf     VRCON, VREN         ; CVref circuit: powered
down, no Idd drain

      ;     bcf     VRCON, VRR          ; CVref Range Selection:
High Range

      ;     bcf     VRCON, VR3          ; CVref value selection
bit 3
      ;     bcf     VRCON, VR2          ; CVref value selection
bit 2
      ;     bcf     VRCON, VR1          ; CVref value selection
bit 1
      ;     bcf     VRCON, VR0          ; CVref value selection
bit 0

            bcf     STATUS, RP0         ; ---- Select Bank 0 -----
      ;---------------------------------------
      ;Analog-to-Digital Converter (A/D) Module (Section 7.0)
(PIC12F675 Only)
      ;
      ;The analog-to-digital converter (A/D) allows conversion of an
analog
      ;input signal to a 10-bit binary representation of that signal.
The
      ;PIC12F675 has four analog inputs multiplexed into one sample
and hold
      ;circuit. There are two registers to control the functions of
the A/D
      ;module:
      ; A/D Control Regester (ADCON0)
```

```
      ; Analog Select Register (ANSEL)
      ;
      ;Note:When using GPIO pins as analog inputs,ensure the TRISIO
register
      ;     bits are set (=1) for input.

                  bcf         ADCON0,ADFM             ;A/D Result
Formed: left justified
                  bcf         ADCON0,VCFG             ;Voltage
Reference: Vdd
                  bcf         ADCON0,ADON             ;ADC is shut-
off and consumes no operating current

                  bsf         STATUS,RP0              ;----Select
Bank 1----

      ;select A/D Conversion Clock Source: Fosc/8
                  bcf         ANSEL,ADCS2             ;A/D Conversion
Clock Select bit 2
                  bcf         ANSEL,ADCS1             ;A/D Conversion
Clock Select bit 1
                  bsf         ANSEL,ADCS0             ;A/D Conversion
Clock Select bit 0
      ;select GPIO pins that will be analog inputs: GP0/AN0,GP1/AN1
                  bcf         ANSEL,ANS3              ;Analog Select
GP4/AN3: digital I/O
                  bcf   ANSEL,ANS2                ;Analog Select
GP2/AN2: digital I/O
                  bcf         ANSEL,ANS1              ;Analog Select
GP1/AN1: digital I/O
                  bcf         ANSEL,ANS0              ;Analog Select
GP0/AN0: digital I/O

                  bcf   STATUS,RP0                ;----Select Bank 0---
-
      ;--------------------------------------
      ; TIMER1 Module with Gate Control (Section 5.0)
      ;
      ; The TIMER1 Control Register (T1CON) is used to enable/disable
TIMER1
      ; and select various features of the TIMER1 module.

            bcf     T1CON, TMR1ON       ; TIMER1: stopped

            bcf     T1CON, TMR1CS       ; TIMER1 Clock Source
Select: Internal Clock (Fosc/4)

            bcf     T1CON, NOT_T1SYNC   ; TIMER1 External Clock
Input Sync Control: Syncronize external clock input

            ; T1OSCEN only if INTOSC without CLKOUT oscillator is
active, else ignored
            bcf     T1CON, T1OSCEN      ; LP Oscillator Enable
Control: LP oscillator off

            ; TIMER1 Input Prescale Select: 1:1
            bcf     T1CON, T1CKPS1      ; TIMER1 Input Clock
Prescale Select bit 1
            bcf     T1CON, T1CKPS0      ; TIMER1 Input Clock
Prescale Select bit 0
```

```
                ; TMR1GE only if TMR1ON = 1, else ignored
                bcf     T1CON, TMR1GE           ; TIMER1 Gate Enable: on


        ;---------------------------------------
        ; Weak Pull-up Register (WPU) (Section 3.2.1)
        ;
        ; Each of the GPIO pins, except GP3, has an individually
configurable
        ; weak internal pull-up. Control bits WPUx enable or disable
each
        ; pull-up. Refer to Register 3-1. Each weak pull-up is
automatically
        ; turned off when the port pin is configured as an output. The
pull-ups
        ; are disabled on a Power-on Reset by the NOT_GPPU bit (see
OPTION_REG below).

                bsf     STATUS, RP0             ; ---- Select Bank 1 -----

        ;       GPIO Pins = xx54x210
                movlw   B'00000000'             ; GP4 pull-up enabled
                movwf   WPU

                bcf     STATUS, RP0             ; ---- Select Bank 0 -----


        ;---------------------------------------
        ; OPTION Register (OPTION_REG) (Section 2.2.2.2)
        ; TIMER0 Module (Section 4.0)
        ;
        ; The OPTION_REG contains control bits to configure:
        ;   Weak pull-ups on GPIO (see also WPU Register above)
        ;   External GP2/INT interrupt
        ;   TMR0
        ;   TMR0/WDT prescaler

                bsf     STATUS, RP0             ; ---- Select Bank 1 -----

                bcf     OPTION_REG, NOT_GPPU    ; GPIO pull-ups: enabled

                bsf     OPTION_REG, INTEDG      ; Interrupt Edge: on rising
edge of GP2/INT pin

                bcf     OPTION_REG, T0CS        ; TMR0 Clock Source:
internal instruction cycle (CLKOUT)
                bcf     OPTION_REG, T0SE        ; TMR0 Source Edge:
increment low-to-high transition on GP2/T0CKI pin

                bcf     OPTION_REG, PSA         ; Prescaler Assignment:
assigned to TIMER0

                ; TMR0 Prescaler Rate: 1:2
                bcf     OPTION_REG, PS2         ; Prescaler Rate Select bit
2
                bcf     OPTION_REG, PS1         ; Prescaler Rate Select bit
1
                bcf     OPTION_REG, PS0         ; Prescaler Rate Select bit
0

                bcf     STATUS, RP0             ; ---- Select Bank 0 -----
```

```
      ;----------------------------------------
      ; Interrupt-on-Change Register (IOCB) (Section 3.2.2)
      ;
      ; Each of the GPIO pins is individually configurable as an
interrupt-
      ; on-change pin. Control bits IOCBx enable or disable the
interrupt
      ; function for each pin. Refer to Register 3-2. The interrupt-
on-change
      ; is disabled on a Power-on Reset.
      ;
      ; Note: Global interrupt enables (GIE and GPIE) must be enabled
for
      ;       individual interrupts to be recognized.

            bsf     STATUS, RP0         ; ---- Select Bank 1 -----

        ; GPIO Pins = xx543210
            movlw   b'00011000'
            movwf   IOCB                ; Interrupt-on-change
enabled: GP3, GP4

            bcf     STATUS, RP0         ; ---- Select Bank 0 -----


      ;----------------------------------------
      ; Peripheral Interrupt Enable Register (PIE1) (Section 2.2.2.4)
      ;
      ; The PIE1 register contains peripheral interrupt enable bits.
      ;
      ; Note: The PEIE bit (INTCON<6>) must be set to enable any
      ;       peripheral interrupt.

            bsf     STATUS, RP0         ; ---- Select Bank 1 -----

            bcf     PIE1, EEIE          ; EE Write Complete
Interrupt: disabled
            bcf     PIE1, ADIE          ; A/D Converter Interrupt
(PIC12F675 Only): disabled
            bcf     PIE1, CMIE          ; Comparator Interrupt:
disabled
            bcf     PIE1, TMR1IE        ; TMR1 Overflow Interrupt:
disabled

            bcf     STATUS, RP0         ; ---- Select Bank 0 -----


      ;----------------------------------------
      ; Interrupt Control Register (INTCON) (Section 2.2.2.3)
      ;
      ; The INTCON register contains enable and disable flag bits for
TMR0
      ; register overflow, GPIO port change and external GP2/INT pin
      ; interrupts.

            bcf     INTCON, PEIE        ; disable Peripheral
Interrupt Enable bit
            bcf     INTCON, T0IE        ; disable TMR0 Overflow
Interrupt Enable bit
```

```
                bcf     INTCON, INTE        ; disable GP2/INT External
Interrupt Enable bit
                bsf     INTCON, GPIE        ; enable Port Change
Interrupt Enable bit

      ;         bcf     INTCON, GIE         ; disable global
interrupts


      ;-----------------------------------------------------------------
--------
      ;-----------------------------------------------------------------
--------
      ; Main Program
      ;-----------------------------------------------------------------
--------
      ;-----------------------------------------------------------------
--------

            bsf   TXDCOM            ; Disable Transmitter

            bcf         INTCON,GIE      ;Disable global interrupt
            clrf  Count
            clrf  Count2
            decfsz      Count,1
            goto  $-1
            decfsz      Count2,1
            goto  $-3

            clrf  Count
            clrf  Count2
            decfsz      Count,1
            goto  $-1
            decfsz      Count2,1
            goto  $-3

      MAIN


      main1

                  call  chPoll                  ;detect poll
                  movf  RX_BUFFER,W
                  movwf DAT_1

                  call  detect
                  movf  RX_BUFFER,W
                  movwf DAT_2

                  movlw Group                   ;move ACK0 Code to
W(have)
                  subwf DAT_1,0

                  BZ          GRP                     ;group
address
                  goto  PRI                     ;poll
      GRP
                  goto  main1

      PRI
```

```
                movlw Poll                          ;detect data from
slave
                subwf DAT_2,0
                BZ          Poll_
                goto  main1
      Poll_
                call  chPoll
                movf  RX_BUFFER,W
                movwf DAT_addr

                call  detect
                movf  RX_BUFFER,W
                movwf DAT_2

                movf  DAT_addr,W
                subwf DAT_1,0
                BZ          RXdat                   ;if the same
address then detect order and qtd.
                movf  DAT_addr,W
                movwf DAT_1
                goto  Poll_
      RXdat
                call  detect
                movf  RX_BUFFER,W
                movwf DAT_3

                call  chPoll                        ;     detect
ack
                movf  DAT_1,W
                subwf RX_BUFFER,0
                BZ          CHack
                movf  DAT_addr,W
                movwf DAT_1
                goto  Poll_
      CHack
                movf  DAT_1,W
                call  TX
                call  DELAY_TX
                movf  DAT_2,W
                call  TX
                call  DELAY_TX
                movf  DAT_3,W
                call  TX
                call  DELAY_TX
                goto  main1

      TX
            movwf TX_BUFFER
            movlw .8
            movwf Counttx1
            call  DELAY_TX
            bcf         TXDCOM
            call  DELAY_TX
      LOOP_SEND
            rrf         TX_BUFFER,1
            btfsc STATUS,C
            goto  L2
      LOGIC_0
            bcf         TXDCOM
      LOGIC_1
            call  DELAY_TX
```

```
        decfsz      Counttx1,1
        goto  LOOP_SEND
        bsf         TXDCOM
        call  DELAY_TX

        return

L2
        bsf         TXDCOM
        goto  LOGIC_1

;*************************************************
LOOP
    bsf     RFENA           ; Enable Transmitter

        clrf  Count
        clrf  Count2
        decfsz      Count,1
        goto  $-1
        decfsz      Count2,1
        goto  $-3

        clrf  Count
        clrf  Count2
        decfsz      Count,1
        goto  $-1
        decfsz      Count2,1
        goto  $-3
        goto  MAIN

DELAY_TX
        movlw .30
        movwf Count
        decfsz      Count,1
        goto  $-1
        return
;-------------------------------------------------------------------
--------
chPoll:     btfsc RX_PIN
            goto  chPoll
            goto  preamble

preamble
            btfss RX_PIN
            goto  $-1
            movlw D'16'
            movwf Countpre
            movlw D'1'
            call  WaitxTEhalf

chpre
            btfss RX_PIN

            goto  chPoll
            movlw D'1'
            call  WaitxTE
            btfsc RX_PIN

            goto  chPoll
            movlw D'1'
            call  WaitxTE
```

```
                decfsz      Countpre,f
                goto  chpre
        detect
                btfss RX_PIN
                goto  $-1
                movlw .8
                movwf       Countrx
        Keeloop
                movlw D'1'
                call  WaitxTEhalf
                movlw D'1'
                call  WaitxTE
                bsf     STATUS,C
                btfsc RX_PIN
                bcf         STATUS,C
                rrf         RX_BUFFER,1
                decfsz      Countrx,1
                goto  nonturn
                goto  turn
        nonturn
                movlw D'1'
                call  WaitxTE
                btfss RX_PIN
                goto  $-1
                goto  Keeloop
        turn
                movlw D'1'
                call  WaitxTEhalf
                movlw D'1'
                call  WaitxTE
                btfsc RX_PIN
                goto  $-1
                return
        WaitxTE
              movwf   Count2        ; [1]

        waitx
              movlw   D'80'         ; [1]
              movwf   Count         ; [1]

        wait400
              nop                   ; [1]
              nop                   ; [1]
              decfsz  Count,F       ; [1]
               goto    wait400      ; [2]
        ;                           --------
        ;                         79 x 5 = 395us

              decfsz  Count2,F      ; [1]
               goto    waitx        ; [2]

              retlw   0             ; [2]

      ;                         total 2 (call) + W x (395 + 5) + 2
(return)
      ;                         w = 1 -> 406us @4MHz
      ;                         w = 2 -> 806us @4MHz
      WaitxTEhalf
              movwf   Count2        ; [1]

      waitxh
```

```
                movlw   D'40'           ; [1]
                movwf   Count           ; [1]

        wait400h
                nop                     ; [1]
                nop                     ; [1]
                decfsz  Count,F         ; [1]
                 goto    wait400h       ; [2]
        ;                               --------
        ;                              79 x 5 = 395us

                decfsz  Count2,F        ; [1]
                 goto    waitxh         ; [2]

                retlw   0               ; [2]

        ;                              total 2 (call) + W x (395 + 5) + 2
(return)
        ;                              w = 1 -> 406us @4MHz
        ;                              w = 2 -> 806us @4MHz
        ; Data EEPROM Memory (Section 8.0)
        ;
        ; PIC12F629/675 devices have 128 bytes of data EEPROM with
address
        ; range 0x00 to 0x7F.

                ; Initialize Data EEPROM Memory locations

        ;        ORG 0x2100
        ;        DE  0x00, 0x01, 0x02, 0x03


        ;-----------------------------------------------------------------
        ; Calibrating the Internal Oscillator (Section 9.2.5.1)
        ; Oscillator Calibration Register (OSCCAL) (Section 2.2.2.7)
        ;
        ; The below statements are placed here so that the program can
be
        ; simulated with MPLAB SIM.  The programmer (PICkit or PROMATE
II)
        ; will save the actual OSCCAL value in the device and restore
it.
        ; The value below WILL NOT be programmed into the device.

                org     0x3ff
                retlw    0x80

        ;-----------------------------------------------------------------
                end                             ; end of program directive
        ;-----------------------------------------------------------------
```

- MCS-51

โปรแกรมภาษาแอสเซมบลีของ MCS-51 ในส่วนเครื่องสั่งอาหาร

```
      ORG   0000H
      AJMP MAINNORMAL
;***********INTERRUPT************
      ORG   0003H
      LJMP INT_0

      ORG   0030H
MAINNORMAL:
      ACKPIC      EQU 45H
      NAKPIC      EQU 43H
      EMPPIC      EQU 41h
      MOV IE,#10000001B
;     SETB EA           ENABLE INTERUPT
;     SETB EX0    ENABLE EXTERNAL INTERRUPT 0
      MOV TCON,#00000001B
;     SETB ITO    SET FALLING EDGE TYPE FOR INT0
      MOV PCON,#00H             ;SMOD=0
      MOV SCON,#50H             ;MODE 1,REN=1
      MOV TMOD,#20H             ;TIMER1 MODE2,TIMER0 MODE0
      MOV TH1,#0FDH             ;9600BAUD
      SETB TR1          ;START TIMER1
;**************REGISTER***********
;29h,32H-34H:a3a2a1,b1  order,No.
;35:NUMBER OF ORDER

      LCALL INITIAL             ;SHOW 'WELCOME TO CLUBR'

loop_1:     LCALL MAINSCAN
      MOV A,30H
      CJNE A,#0BH,loop_1        ;PRESS'0'?

F_ORDER:
      LCALL CLRDISP
      MOV DPTR,#TEXT_LINE03   ;SHOW 'ORDER:    No.  '
      MOV A,#80H
      LCALL LCDLDS

      LCALL MAINSCAN
a3:   LCALL ORDER         ;RECEIVE a3
      MOV 29H,A
      MOV A,#86H
      LCALL LCDWI
      MOV A,29H           ;SHOW NUMBER a3
      LCALL LCDWD
      MOV DPTR,#TEXT_LINE04   ;SHOW '         *-CANCEL',LINE 2
      MOV A,#0C0H
      LCALL LCDLDS

a2:   LCALL MAINSCAN            ;RECEIVE a2
      LCALL ORDER
      MOV 32H,A
      MOV A,#87H
      LCALL LCDWI
      MOV A,32H           ;SHOW NUMBER a2
      LCALL LCDWD

a1:   LCALL MAINSCAN            ;RECEIVE a1
      LCALL ORDER
```

```
        MOV 33H,A
        MOV A,#88H
        LCALL LCDWI
        MOV A,33H           ;SHOW NUMBER a1
        LCALL LCDWD

b1:     LCALL MAINSCAN          ;RECEIVE b1
        LCALL ORDER
        CJNE A,#30H,b11
        LJMP F_ORDER
b11:    MOV 34H,A
        MOV A,#8FH
        LCALL LCDWI
        MOV A,34H
        LCALL LCDWD

CAL:    MOV 35H,#00H            ;INITIAL 35H AS#00H
        MOV A,33H

CAL1:   SUBB A,#30H
        ADD A,35H
        MOV 35H,A           ;ADD 35H WITH 33H

        MOV A,32H
CAL10:      SUBB A,#30H
        MOV B,#0AH
        MUL AB
        ADD A,35H
        MOV 35H,A           ;ADD 35H WITH 32H*10

        MOV A,29H
CAL100:     SUBB A,#30H
        MOV B,#64H
        MUL AB
        ADD A,35H
        MOV 35H,A           ;ADD 35H WITH 29H*100
        CLR A
        ADDC A,B    ;+C     ;B>0? OVER FLOW?
        CJNE A,#00H,MISS
        LJMP TRANSL

TRANSL:     MOV A,35H
      CJNE A,#58H,CHECKC        ;0-88 MENU A<DATA :C=1
CARRY:      SETB C
CHECKC:
      JB PSW.7,TOMENU

MISS: MOV DPTR,#TEXT_LINE07   ;MISSING ORDER
      MOV A,#80H
      LCALL LCDLDS
      LJMP PRESSDOWN
TOMENU:     MOV DPTR,#MENU000
      MOV A,35H
      MOV B,#10H
      MUL AB
      ADD A,DPL
      MOV DPL,A
      MOV A,B
      ADDC A,DPH
      MOV DPH,A
;*************************
```

```
        CLR A
        MOVC A,@A+DPTR
        CJNE  A,#' ',NEXT
        SJMP MISS
;*************************
NEXT: MOV A,#80H
        LCALL LCDLDS
        MOV A,#8FH
        LCALL LCDWI
        MOV A,34H
        LCALL LCDWD
        MOV DPTR,#TEXT_LINE05          ;'SHOW 1-CONF *-CANCEL'
        MOV A,#0C0H
        LCALL LCDLDS

CHCON:      LCALL MAINSCAN
      MOV A,30H
      CJNE A,#01H,CHCAN
      SJMP SEND
CHCAN:      CJNE A,#0AH,CHCON
      LJMP F_ORDER

;*************SENDING***************
SEND: LCALL CLRDISP
      MOV DPTR,#TEXT_LINE08    ;SHOW '      SENDING... '
      MOV A,#0C0H
      LCALL LCDLDS

      MOV A,35H
      MOV SBUF,A
WAITTI:     JNB TI,WAITTI
      CLR TI
      MOV A,34H
      CLR CY

      SUBB A,#30H
      MOV SBUF,A

WAITTI_1:
      JNB TI,WAITTI_1
      CLR TI


      LCALL DELAYRI

      JNB RI,$
      CLR RI
      MOV A,SBUF

;*********************************
;*********************************

      CJNE A,#ACKPIC,COMNAK
      SJMP COMP
COMNAK:     CJNE A,#EMPPIC,ERRINTX
;     SJMP COMP

      MOV DPTR,#TEXT_LINE06    ;THIS ORDER IS 0000
      MOV A,#80H
      LCALL LCDLDS
      MOV DPTR,#TEXT_LINE23    ;PRESS ANY KEY
```

```
        MOV A,#0C0H
        LCALL LCDLDS
        LCALL MAINSCAN
        LJMP F_ORDER
COMP:   MOV DPTR,#TEXT_LINE09    ;COMPLETE
        MOV A,#80H
        LCALL LCDLDS
        MOV DPTR,#TEXT_LINE23    ;PRESS ANY KEY
        MOV A,#0C0H
        LCALL LCDLDS
        LCALL MAINSCAN
        LJMP F_ORDER
ERRINTX:
        MOV DPTR,#TEXT_LINE10    ;SENDIND ERROR
        MOV A,#80H
        LCALL LCDLDS
        MOV DPTR,#TEXT_LINE23    ;PRESS ANY KEY
        MOV A,#0C0H
        LCALL LCDLDS
        LCALL MAINSCAN
        LJMP F_ORDER

PRESSDOWN:
        MOV DPTR,#TEXT_LINE23    ;PRESS ANY KEY
        MOV A,#0C0H
        LCALL LCDLDS
        LCALL MAINSCAN
        LJMP F_ORDER
        SJMP $
;***************INT_0***************
INT_0:      PUSH ACC
        PUSH PSW
        NOP
        POP PSW
        POP ACC

        RETI

;************************************

ORDER:
        MOV P1,#00H
        LCALL DELAY
CHECK_:                  ;*
        MOV A,30H
        CJNE A,#0AH,CHECK__
        POP ACC
        POP ACC
        LJMP F_ORDER      ;CANCEL
CHECK__:          ;#
        CJNE A,#0CH,CHECK
        LCALL MAINSCAN
        SJMP ORDER  ;*
CHECK:      CJNE A,#0BH,NOZERO
        MOV A,#00H
NOZERO:
        ADD A,#30H

RETORDER:
        RET
;*********LCD INIT*****************
```

```
INITIAL:
      MOV R2,#00H
LOOP:
      LCALL DELAY
      DJNZ R2,LOOP
      LCALL DELAY          ;wait for >15ms
      MOV A,#00111100B
      LCALL LCDWI
      MOV A,#00111100B
      LCALL LCDWI          ;INCREASE >100uS
      MOV A,#00111100B     ;8 bit 2 line
      LCALL LCDWI          ;INCREASE
      MOV A,#00001100B     ;DISP ON CURSOR ON BLINK OFF
      LCALL LCDWI
      MOV A,#00000001B     ;DISP CLEAR
      LCALL LCDWI
      MOV A,#00000110B     ;INCREASE ADDR
      LCALL LCDWI
      MOV R2,#05H
      LCALL DELAY
MAIN:
      MOV DPTR,#TEXT_LINE01
      MOV A,#80H
      LCALL LCDLDS
      MOV DPTR,#TEXT_LINE02
      MOV A,#0C0H
      LCALL LCDLDS
      RET
LCDLDS:
      MOV P1,#00H
      LCALL DELAY
      LCALL LCDWI
      MOV R2,#10H
LCDLDS1: CLR A
      MOVC A,@A+DPTR
      ACALL LCDWD
      INC DPTR
      DJNZ R2,LCDLDS1
      RET
;*******************LCD WRITE INSTRUCTION****************
LCDWI:      PUSH DPH
      PUSH DPL
      MOV P0,A
      CLR ACC.0
      CLR ACC.1
      CLR ACC.2
      MOV P1,A
      SETB ACC.2
      MOV P1,A
      CLR ACC.2
      MOV P1,A
      MOV A,#00H
LCDWI1:     DEC A
      JNZ LCDWI1
      POP DPL
      POP DPH
      RET
;*************LCD WRITE DATA************************
;**************A=INPUT DATA*****************************
LCDWD:  PUSH DPH
      PUSH DPL
```

```
        MOV P0,A
;***************WRITE INSTRUCTION*****************
        SETB ACC.0
        CLR ACC.1
        CLR ACC.2
        MOV P1,A
        SETB ACC.2
        MOV P1,A
        CLR ACC.2
        MOV P1,A
;***************DELAY************************
        MOV A,#00H
LCDWD1:     DEC A
        JNZ LCDWD1
        POP DPL
        POP DPH
        RET
MENU000:DB "MENU000          "
MENU001:DB "TOAST            "
MENU002:DB "FRIED EGGS WITH  "
MENU003:DB "CHICKEN C SOUP   "
MENU004:DB "MUSHROOM C SOUP  "
MENU005:DB "CORN CREAM SOUP  "
MENU006:DB "MIXED SALAD      "
MENU007:DB "TUNA FISH SALAD  "
MENU008:DB "CHICKEEN SALAD   "
MENU009:DB "BEEF STEAK SALAD"
MENU010:DB "SEAFOOD SALAD    "
MENU011:DB "CHICKEN SANDWICH"
MENU012:DB "TUNA SANDWICH    "
MENU013:DB "HAM SANDWICH     "
MENU014:DB "HAM&CHEESE SANDW"
MENU015:DB "GARLIC BREAD     "
MENU016:DB "FRENCH FRIED     "
MENU017:DB "CLUB SANDWICH    "
MENU018:DB "CHICKEN STEAK    "
MENU019:DB "PORK STEAK       "
MENU020:DB "FISH STEAK       "
MENU021:DB "LONDONFISH STEAK"
MENU022:DB "FILLET STEAK     "
MENU023:DB "PEPPER STEAK     "
MENU024:DB "FILLET MIGNON    "
MENU025:DB "SPAGHETTI BEEF   "
MENU026:DB "SPAGHETTI PORK   "
MENU027:DB "SPAGHETTI CHICK  "
MENU028:DB "SPAGHETTI NAPOLI"
MENU029:DB "SPAGHETTI MUSSEL"
MENU030:DB "MACARONI SHRIMP  "
MENU031:DB "MACARONI HAM     "
MENU032:DB "PORK SAUSAGE     "
MENU033:DB "                 "
MENU034:DB "                 "
MENU035:DB "                 "
MENU036:DB "                 "
MENU037:DB "                 "
MENU038:DB "                 "
MENU039:DB "                 "
MENU040:DB "                 "
MENU041:DB "                 "
MENU042:DB "                 "
MENU043:DB "                 "
```

```
MENU044:DB "                "
MENU045:DB "                "
MENU046:DB "                "
MENU047:DB "                "
MENU048:DB "                "
MENU049:DB "                "
MENU050:DB "                "

MENU051:DB "ESPRESSO        "
MENU052:DB "MOCHA           "
MENU053:DB "LATTE           "
MENU054:DB "CAPUCCINO       "
MENU055:DB "ICE ESPRESSO    "
MENU056:DB "ICE MOCHA       "
MENU057:DB "ICE LATTE       "
MENU058:DB "ICE CAPUCCINO   "
MENU059:DB "FRAPPE ESPRESSO "
MENU060:DB "FRAPPE MOCHA    "
MENU061:DB "FRAPPE LATTE    "
MENU062:DB "FRAPPE CAPUCCINO"
MENU063:DB "                "
MENU064:DB "                "
MENU065:DB "                "
MENU066:DB "                "
MENU067:DB "                "
MENU068:DB "                "
MENU069:DB "                "
MENU070:DB "                "
MENU071:DB "HOT TEA         "
MENU072:DB "ICE THAI TEA    "
MENU073:DB "ICE TEA FRAPPE  "
MENU074:DB "ICE LEMON TEA   "
MENU075:DB "ICE CHOCOLATE   "
MENU076:DB "ICE CHOCO FRAPPE"
MENU077:DB "HOT FRESH MILK  "
MENU078:DB "ICE FRESH MILK  "
MENU079:DB "ORANGE JUICE    "
MENU080:DB "ORANGE FRAPPE   "
MENU081:DB "LEMON JUICE     "
MENU082:DB "LEMON FRAPPE    "
MENU083:DB "GUAVA MIXED     "
MENU084:DB "COKE            "
MENU085:DB "ICE CUBE / JUG  "
MENU086:DB "ICE CUBE / GLASS"
MENU087:DB "SINGHA BEER     "
MENU088:DB "HEINEKEN BEER   "

TEXT_LINE01:DB "WelcomeToWi-REST"
TEXT_LINE02:DB "Press0 for Order"
TEXT_LINE03:DB "Order:___  Qty._"
TEXT_LINE04:DB "        *-Cancel"
TEXT_LINE05:DB "1-Ok    *-Cancel"
TEXT_LINE06:DB "    Empty...    "
TEXT_LINE07:DB " Missing Order  "
TEXT_LINE08:DB "   Sending...   "
;*****************************
TEXT_LINE09:DB "Sending Complete"
TEXT_LINE10:DB " Sending Error  "
TEXT_LINE23:DB  "  Press any key "
```

```
MAINSCAN:
        MOV R3,#11110111B
;       MOV R3,#10111111B old
        MOV R4,#04H
        MOV P2,#0FFH                ;INITIAL INPUT PORT
SCAN:
        MOV A,R3
        MOV 31H,R3
        clr ACC.0
        clr ACC.1
        clr ACC.2
        MOV P1,A
INKEY:

        MOV A,P2
        ANL A,#07H
        CJNE A,#07H,KEYDOWN
        MOV A,R3
        RL A
        MOV R3,A
        DJNZ R4,SCAN
        SJMP MAINSCAN
KEYDOWN:
        MOV 60H,A
        MOV R1,#28H
        LCALL DELAY
        MOV A,P2
        ANL A,#07H
        CJNE A,60H,MAINSCAN
BACK:
        MOV A,P2
        ANL A,#07H
        CJNE A,#07H,BACK
RE:
        MOV A,P2
        ANL A,#07H
        MOV 61H,A
        MOV R1,#28H
        LCALL DELAY
        MOV A,P2
        ANL A,#07H
        CJNE A,61H,MAINSCAN
RELESE:     LJMP CHKVAL

;**********check value**********

CHKVAL:
            MOV A,R3
ROW1: ANL A,#01111000B
      CJNE A,#00111000B,ROW2
      MOV 30H,#01H
      AJMP COL
ROW2: CJNE A,#01011000B,ROW3
      MOV 30H,#04H
      AJMP COL
ROW3: CJNE A,#01101000B,ROW4
      MOV 30H,#07H
      AJMP COL
ROW4: CJNE A,#01110000B,RTURN
      MOV 30H,#0AH
      AJMP COL
```

```
RTURN:      RET
;***COL*******
COL:  MOV A,60H
      ANL A,#00000111B
COL1: CJNE A,#00000110B,COL2
      MOV A,30H
      RET
COL2:       CJNE A,#00000101B,COL3
      MOV A,30H
      INC A
      MOV 30H,A
      RET
COL3:       CJNE A,#00000011B,CTURN
      MOV A,30H
      INC A
      INC A
      MOV 30H,A
CTURN:      RET

;**********CLEAR DISPLAY************
CLRDISP:
      MOV P1,#00H
      LCALL DELAY
      MOV A,#01H
      LCALL LCDWI
      RET

;**************DELAY 2mS***************
DELAY:
      MOV R1,#04H
DELAY1: MOV R4,#00H
      DJNZ R4,$
      DJNZ R1,DELAY1
      RET
;**************delay 1-10s************
delayisd:
      mov r2,#32h
dd:   lcall delay
      djnz r2,dd
      ret
;**************20MS******************
DELAY15S:
            MOV R2,#71H
DELAY2:     MOV R1,#00H
DELAY3: MOV R4,#00H
      DJNZ R4,$
      DJNZ R1,DELAY3
      DJNZ R2,DELAY2
      RET
DELAYRI:
      MOV R2,#10H
DEL1S1:
      MOV R1,#0FFH
DEL1S2:
      MOV R0,#0FFH
DEL1S3:
      DJNZ R0,DEL1S3
      DJNZ R1,DEL1S2
      DJNZ R2,DEL1S1
      RET
      END
```

โปรแกรมภาษาแอสเซมบลีของ MCS-51 ในส่วนคิว

```
        ORG 0000H
        SJMP MAIN
        ORG 0003H
        LJMP INT_0
        ORG 0023H           ;serial interrupt
        LJMP INT_1
        ORG 0030H
MAIN:
        DATA  EQU   P0
        SELIN EQU   P2.0
        INIT  EQU   P2.1
        STB   EQU   P2.3
        AOTUF EQU   P2.2
        BUSY  EQU   P1.3

        MUTE  EQU P2.7
        REST  EQU P2.6
        SHIFT EQU P2.5
        ST    EQU P2.4
        TAB1  EQU 30H
        TAB2  EQU 31H
        TAB3  EQU 32H
        MOV IE,#10010101B
;       SETB EA             ENABLE INTERUPT
;       SETB ES             ENABLE SERIAL PORT INTERRUPT
;       SETB EX0    ENABLE EXTERNAL INTERRUPT 0
;       SETB EX1    ENABLE EXTERNAL INTERRUPT 1
        MOV TCON,#00000101B
;       SETB IT0    SET FALLING EDGE TYPE FOR INT0
;       SETB IT1    SET FALLING EDGE TYPE FOR INT1
        MOV PCON,#00H           ;SMOD=0
        MOV SCON,#50H           ;MODE 1,REN=1
        MOV TMOD,#20H           ;TIMER1 MODE2,TIMER0 MODE0
        MOV TH1,#0FDH           ;9600BAUD
        SETB TR1            ;START TIMER1
                            ;___   ___
;       SETB ST             ;     1_1     start pulse
        clr st
        CLR SHIFT           ;      _
        CLR REST            ;___1 1____ reset pulse(rising edge)

        clr MUTE

        SETB STB
        CLR SELIN
        CLR INIT
        CLR AOTUF
        MOV P1,#0FFH
        LCALL CHBUSY
        MOV DATA,#11H
        LCALL SENDSTB

        LCALL CHBUSY
        MOV DATA,#0EH
        LCALL SENDSTB

        LCALL CHBUSY
        MOV DATA,#27                ;esc
        LCALL SENDSTB
```

```
LCALL CHBUSY
MOV DATA,#2AH                   ;*
LCALL SENDSTB

LCALL CHBUSY
MOV DATA,#'c'                   ;c
LCALL SENDSTB

LCALL CHBUSY
MOV DATA,#'1'                   ;8
LCALL SENDSTB

LCALL CHBUSY
MOV DATA,#'2'                   ;3
LCALL SENDSTB

LCALL CHBUSY
MOV DATA,#'0'                   ;3
LCALL SENDSTB

LCALL CHBUSY
MOV DATA,#'D';                  ;D

LCALL CHBUSY
MOV DATA,#27                    ;esc
LCALL SENDSTB

LCALL CHBUSY
MOV DATA,#40                    ;(
LCALL SENDSTB

LCALL CHBUSY
MOV DATA,#115                   ;s
LCALL SENDSTB

LCALL CHBUSY
MOV DATA,#'2'                   ;2
LCALL SENDSTB

LCALL CHBUSY
MOV DATA,#'0'                   ;0
LCALL SENDSTB

LCALL CHBUSY
MOV DATA,#56H;                  ;V

LCALL SENDSTB

LCALL CHBUSY
MOV DATA,#27                    ;esc
LCALL SENDSTB

LCALL CHBUSY
MOV DATA,#')'                   ;)
LCALL SENDSTB

LCALL CHBUSY
MOV DATA,#115                   ;s
LCALL SENDSTB
```

```
        LCALL CHBUSY
        MOV DATA,#'2'                   ;1
        LCALL SENDSTB
        LCALL CHBUSY
        MOV DATA,#'0'                   ;2
        LCALL SENDSTB
        LCALL CHBUSY
        MOV DATA,#56H;                  ;V
        lcall SENDSTB
        MOV 30H,#01H
        MOV 31H,#00H
        MOV 20H,#00H
WAIT:
        LCALL CHEC
;CHECK NQ<=QR (Next Queue ที่จะได้รับบริการ,Queue Receive หมายเลขล่าสุดที่อยู่ในคิว)
        LJMP WAIT
;30H NQ
;31H QR
;20H Buffer Table Blank ( ---- -321 )
;33H Buffer NQ
;32H Buffer TAB
;50-51h buffer tab,que
CHEC: MOV A,30H
        CJNE A,31H,CHEC2   ;A<31H:C=1
        LJMP ISD
CHEC2:      JB CY,ISD
        RET

ISD:
        MOV 32H,#00H
        JNB 20H.0,TAB1_
        JNB 20H.1,TAB2_
        JNB 20H.2,TAB3_
        RET
TAB3_:      INC 32H
        INC 32H
        INC 32H
        setb 20H.2
        sjmp CONT
TAB2_:      INC 32H
        INC 32H
        setb 20H.1
        sjmp CONT
TAB1_:      INC 32H
        setb 20H.0

CONT:
        CLR EX1
        MOV 33H,30H
        LCALL DISPLAY
        MOV 33H,30H
        INC 30H
        mov R1,#0ah             ;เชิญหมายเลข
        lcall READ
        mov r2,#20h
        lcall delay_

        mov a,33h
        mov b,#64h
        div ab
        mov R1,a
```

```
        lcall READ
        mov r2,#15h
        lcall delay_
        mov a,b
        mov b,#0ah
        div ab
        mov R1,a
        lcall READ
        mov r2,#15h
        lcall delay_

        mov r1,b
        lcall READ
        mov r2,#15h
        lcall delay_
BACK2:      MOV R1,#0bH
        LCALL READ
        mov r2,#15h
        lcall delay_
        MOV R1,32H
        LCALL READ
        mov r2,#15h
        lcall delay_

        MOV R1,#0cH
        LCALL READ
        mov r2,#15h
        lcall delay_
        SETB EX1
        RET
READ: mov 80h,r1
        SETB REST
        LCALL DELAY_ISD
        CLR REST
        LCALL DELAY_ISD
        mov a,80h
        cjne a,#00,normal
        sjmp start
normal:
        SETB SHIFT

SHIFT_:     LCALL DELAY_ISD
      setb ST
      mov r2,#09h
      lcall dd
      clr ST
      DJNZ 80h,SHIFT_
      LCALL DELAY_ISD
start:
      CLR SHIFT
      LCALL DELAY_ISD
      setb ST
      LCALL DELAY_ISD
      clr ST
      setb MUTE
      RET
;**************PRESS*******************
;CHECK NQ<=QR (Next Queue ที่จะได้รับบริการ, Queue Receive หมายเลขล่าสุดที่อยู่ในคิว)
      LJMP WAIT
;30H NQ
;31H QR
```

```
;20H Buffer Table Blank ( ---- -321 )
;33H Buffer NQ
;32H Buffer TAB
INT_0:      PUSH ACC
      CLR EA
      INC 31H
      MOV A,30H
      CJNE A,31H,QUE1
      LJMP CHSIT
CHSIT:      JB 20H.0,CHT2
      MOV 32H,#01H
      setb 20h.0
      LJMP SIT
CHT2: JB 20H.1,CHT3
      MOV 32H,#02H
      setb 20h.1
      LJMP SIT
CHT3: JB 20H.2,QUE1
      MOV 32H,#03H
      setb 20h.2
      LJMP SIT
QUE1: LJMP QUE
SIT:  mov 33h,31h
      lcall display
      LCALL PRINTER
      mov 33h,30h
      inc 30h
      mov R1,#0ah
      lcall READ
      mov r2,#20h
      lcall delay_
      mov a,33h
      mov b,#64h
      div ab
      mov R1,a
      lcall READ
      mov r2,#15h
      lcall delay_
      mov a,b
      mov b,#0ah
      div ab
      mov R1,a
      lcall READ
      mov r2,#15h
      lcall delay_
      mov r1,b
      lcall READ
      mov r2,#15h
      lcall delay_

BACK: MOV R1,#0bH
      LCALL READ
      mov r2,#15h
      lcall delay_
      MOV R1,32H
      LCALL READ
      mov r2,#15h
      lcall delay_
      MOV R1,#0cH
      LCALL READ
      mov r2,#15h
```

```
        lcall delay_
RETINT0:SETB EA
        POP ACC
        RETI
QUE:    mov 32h,#00h
        mov 33h,31h
        lcall display
        LCALL PRINTER
        mov r2,#0AH
        lcall delay_
        mov 32h,50h
        mov 33h,51h
        lcall display
        sjmp RETINT0
;*************************************
PRINTER:
        MOV A,33H
        MOV B,#0AH
        DIV AB
        ADD a,#30H
        mov 35H,A
        mov A,b
        ADD A,#30h
        mov 34h,A
        MOV R2,#07H
NULL:
        LCALL CHBUSY
        MOV DATA,#' '                   ;NULL
        LCALL SENDSTB
        DJNZ R2,NULL

        mov R2,#0FH
LF:     LCALL CHBUSY
        MOV DATA,#0AH                   ;Line Feed
        LCALL SENDSTB
        DJNZ  R2,LF
NEXT:
        LCALL CHBUSY
        MOV DATA,35h                    ;
        LCALL SENDSTB

        LCALL CHBUSY
        MOV DATA,34h                    ;
        LCALL SENDSTB

;*****************************************************************
**
        LCALL CHBUSY
        MOV DATA,#0DH                   ;Carriage return
        LCALL SENDSTB
;*****************************************************************
*******
FF:     LCALL CHBUSY
        MOV DATA,#0CH                   ;Form Feed
        LCALL SENDSTB
        RET
DISPLAY:
        mov a,32h           ;หมายเลขโต๊ะ
        mov sbuf,a
        jnb ti,$
        clr ti
```

```
        nop
        nop
        mov a,33h           ;หมายเลขคิว
        mov sbuf,a
        jnb ti,$
        clr ti
        mov a,32h
        cjne a,#00h,SAVE
        sjmp return
SAVE:   mov 50h,32h
        mov 51h,33h
return:     RET
;*********************************
CHECK:      CJNE A,#TAB1,CHECK2
        CLR 20H.0
        RET
CHECK2:     CJNE A,#TAB2,CHECK3
        CLR 20H.1
        RET
CHECK3: CJNE A,#TAB3,WAIT_
        CLR 20H.2
WAIT_:      RET
;************************************
INT_1:      PUSH ACC
        jnb ri,RETSE
        clr ri
        MOV A,SBUF
        LCALL CHECK ;CLEAR TAB
RETSE:      POP ACC
        RETI
;************************************
delay_:
        mov r1,#88h
dd_1:   mov r4,#00h
        djnz r4,$
        djnz r1,dd_1
        djnz r2,delay_
        clr MUTE
        ret
DELAY_ISD:
        mov r2,#25h         ;32h
dd:     lcall delay
        djnz r2,dd
        ret
DELAY:
        MOV R1,#04H
DELAY1: MOV R4,#00H
        DJNZ R4,$
        DJNZ R1,DELAY1
        RET
CHBUSY:
        JB BUSY,$
        RET

SENDSTB:
        CLR STB
        MOV R0,#10H
        DJNZ R0,$
        SETB STB
        RET
        END
```

โปรแกรมภาษาแอสเซมบลีของ MCS-51 ในส่วนการแสดงผลบนจอ Dot-matrix

```
        ORG 0000H
        AJMP MAIN_
        ORG 0023H
        LJMP INT_S

        ORG 0030H
MAIN_:
        ROW    EQU    P0.0

COL_1   EQU    P0.1
        COL_2  EQU    P0.2

        CLK    EQU    P0.3
        LATCH  EQU    P0.4
        STR    EQU    P0.5
;20H    BUFFER ROW
;21H    BUFFER COL_11
;22H    BUFFER COL_12
;23H    BUFFER COL_21
;24H    BUFFER COL_22
;30H    COUNTER1 COUNT     2(*8 BIT)
;31H    COUNTER2 COUNT   8 BIT
;32H    COUNTER3 COUNT   16 ROUND
;33H    COUNTER4 COUNT   Num OF BIT IN 1 ROUND
;34H    COUNTER5 COUNT   Num OF ROUND
;        ----------
;       |1-1 | 1-2|
;       |    |    |
;       |-----------
;       |2-1 | 2-2|
;       |    |    |
;        ----------
;40H    DPH COL_1 1-2
;41H    DPL
;42H    DPH COL_1 1-1
;43H    DPL
;44H    DPH COL_1 2-2
;45H    DPL
;46H    DPH COL_1 2-1
;47H    DPL
;50H    DPH COL_2 1-2
;51H    DPL
;52H    DPH COL_2 1-1
;53H    DPL
;54H    DPH COL_2 2-2
;55H    DPL
;56H    DPH COL_2 2-1
;57H    DPL
        MOV IE,#10010000B
;       SETB EA             ENABLE INTERUPT
;       SETB ES             ENABLE SERIAL PORT INTERRUPT
        MOV TCON,#00000001B
;       SETB ITO     SET FALLING EDGE TYPE FOR INT0
        MOV PCON,#00H               ;SMOD=0
        MOV SCON,#50H               ;MODE 1,REN=1
        MOV TMOD,#20H               ;TIMER1 MODE2,TIMER0 MODE0
        MOV TH1,#0FDH               ;9600BAUD
        SETB TR1            ;START TIMER1
```

```
        mov dptr,#wel_2
        MOV 40H,DPH
        MOV 41H,DPL
        mov dptr,#wel_3
        MOV 52H,DPH
        MOV 53H,DPL
        mov dptr,#wel_5
        MOV 46H,DPH
        MOV 47H,DPL
        mov dptr,#wel_4
        MOV 50H,DPH
        MOV 51H,DPL
        mov dptr,#wel_1
        MOV 42H,DPH
        MOV 43H,DPL
        mov dptr,#wel_6
        mov 44h,dph
        mov 45h,dpl
        mov dptr,#wel_8
        mov 54h,dph
        mov 55h,dpl
        mov dptr,#wel_7
        MOV 56H,DPH
        MOV 57H,DPL
        CLR ROW
        CLR CLK
        SETB LATCH
        SETB STR
MAIN:
        MOV R3,#08H
        MOV R1,#40H
        MOV R0,#60H
INIT:
        MOV A,@R1
        MOV @R0,A
        INC R1
        INC R0
        DJNZ R3,INIT
        MOV R3,#08H
        MOV R1,#50H
        MOV R0,#70H
INIT_1:
        MOV A,@R1
        MOV @R0,A
        INC R1
        INC R0
        DJNZ R3,INIT_1
        mov dph,60h
        mov dpl,61h
        clr a
        movc  a,@a+dptr
        mov 21h,a
        mov dph,62h
        mov dpl,63h
        clr a
        movc  a,@a+dptr
        mov 22h,a
        mov dph,70h
        mov dpl,71h
        clr a
        movc  a,@a+dptr
```

```
        mov 23h,a
        mov dph,72h
        mov dpl,73h
        clr a
        movc  a,@a+dptr
        mov 24h,a
        CLR ROW
        CLR CLK
        MOV 34H,#01H
        MOV 32H,#08H
LOOP_8R:
        LCALL LOOP_
        setb STR
        clr LATCH
        setb LATCH
        clr STR
        mov a,#01h
        add a,61H
        mov 61h,a
        CLR A
        ADDC A,60H
        MOV 60H,A
        MOV DPH,60H
        MOV DPL,61H
        CLR A
        MOVC A,@A+DPTR
        MOV 21H,A
        mov a,#01h
        add a,63H
        mov 63h,a
        CLR A
        ADDC A,62H
        MOV 62H,A
        MOV DPH,62H
        MOV DPL,63H
        CLR A
        MOVC A,@A+DPTR
        MOV 22H,A
        mov a,#01h
        add a,71H
        mov 71h,a
        CLR A
        ADDC A,70H
        MOV 70H,A
        MOV DPH,70H
        MOV DPL,71H
        CLR A
        MOVC A,@A+DPTR
        MOV 23H,A
        mov a,#01h
        add a,73H
        mov 73h,a
        CLR A
        ADDC A,72H
        MOV 72H,A
        MOV DPH,72H
        MOV DPL,73H
        CLR A
        MOVC A,@A+DPTR
        MOV 24H,A
```

```
        INC 34H
        DJNZ 32H,LOOP_8R
        mov dph,64h
        mov dpl,65h
        clr a
        movc  a,@a+dptr
        mov 21h,a

        mov dph,66h
        mov dpl,67h
        clr a
        movc  a,@a+dptr
        mov 22h,a

        mov dph,74h
        mov dpl,75h
        clr a
        movc  a,@a+dptr
        mov 23h,a

        mov dph,76h
        mov dpl,77h
        clr a
        movc  a,@a+dptr
        mov 24h,a

        CLR ROW
        CLR CLK

        MOV 32H,#08H
LOOP_16R:
        LCALL LOOP_

        setb STR
        clr LATCH
        setb LATCH
        clr STR
        mov a,#01h
        add a,65H
        mov 65h,a
        CLR A
        ADDC A,64H
        MOV 64H,A
        MOV DPH,64H
        MOV DPL,65H
        CLR A
        MOVC A,@A+DPTR
        MOV 21H,A

        mov a,#01h
        add a,67H
        mov 67h,a
        CLR A
        ADDC A,66H
        MOV 66H,A
        MOV DPH,66H
        MOV DPL,67H
        CLR A
        MOVC A,@A+DPTR
        MOV 22H,A
```

```
        mov a,#01h
        add a,75H
        mov 75h,a
        CLR A
        ADDC A,74H
        MOV 74H,A
        MOV DPH,74H
        MOV DPL,75H
        CLR A
        MOVC A,@A+DPTR
        MOV 23H,A

        mov a,#01h
        add a,77H
        mov 77h,a
        CLR A
        ADDC A,76H
        MOV 76H,A
        MOV DPH,76H
        MOV DPL,77H
        CLR A
        MOVC A,@A+DPTR
        MOV 24H,A

        INC 34H
        DJNZ 32H,LOOP_16R

        LJMP MAIN

LOOP_:

        MOV 33H,#01H

        MOV 31H,#08H
LOOP_1:

        mov C,21H.0
        MOV COL_1,C

        mov C,23H.0
        MOV COL_2,C
        MOV A,33H

        CLR ROW
        CJNE A,34H,L_1
        SETB ROW
L_1:
        SETB CLK
        CLR CLK


        INC 33H

        MOV A,21H
        RR A
        MOV 21H,A

        MOV A,23H
        RR A
```

```
        MOV 23H,A

        DJNZ 31H,LOOP_1

        MOV 31H,#08H
LOOP_2:
        mov C,22H.0
        MOV COL_1,C
        mov C,24H.0
        MOV COL_2,C
        MOV A,33H

        CLR ROW
        CJNE A,34H,L_2

        SETB ROW

L_2:
        SETB CLK
        CLR CLK


        INC 33H
        MOV A,22H
        RR A
        MOV 22H,A

        MOV A,24H
        RR A
        MOV 24H,A

        DJNZ 31H,LOOP_2


        RET
;*****************************************
INT_S:
        PUSH ACC
        push DPH
        PUSH DPL
        JNB RI,$
        MOV A,SBUF
        CLR RI
        MOV 4EH,A                ;table

        JNB RI,$
        MOV A,SBUF
        CLR RI
        MOV 4FH,A

        MOV A,4EH
        CJNE A,#00H,TAB          ;SHOW No. OF Q
        ljmp NOTAB
TAB:
        MOV DPTR,#Sign_1
        MOV 46H,DPH
        MOV 47H,DPL
        MOV DPTR,#Sign_2
        MOV 44H,DPH
        MOV 45H,DPL
```

```
        MOV A,4EH
        MOV DPTR,#Zero
        mov b,#08h
        mul ab
        ADD A,DPL
        MOV 57H,A
        CLR A
        ADDC A,DPH
        MOV 56H,A

        MOV DPTR,#BLANK
        MOV 54H,DPH
        MOV 55H,DPL
        MOV 42H,DPH
        MOV 43H,DPL
        MOV 50H,DPH
        MOV 51H,DPL

        MOV A,4FH
        MOV B,#0AH
        DIV AB
        mov 49h,b

        MOV DPTR,#Zero
        mov b,#08h
        mul ab
        ADD A,DPL
        MOV 41H,A
        CLR A
        ADDC A,DPH
        MOV 40H,A

        mov b,49h
        MOV A,B
        MOV DPTR,#Zero
        mov b,#08h
        mul ab
        ADD A,DPL
        MOV 53H,A
        CLR A
        ADDC A,DPH
        MOV 52H,A

        SJMP RET_
NOTAB:
        MOV DPTR,#BLANK
        MOV 40H,DPH
        MOV 41H,DPL
        MOV 42H,DPH
        MOV 43H,DPL
        MOV 46H,DPH
        MOV 47H,DPL

        MOV 50H,DPH
        MOV 51H,DPL
        MOV 52H,DPH
        MOV 53H,DPL
        MOV 54H,DPH
        MOV 55H,DPL

        MOV A,4FH
```

```
        MOV B,#0AH
        DIV AB
        mov 49h,b

        MOV DPTR,#Zero
        mov b,#08h
        mul ab

        ADD A,DPL
        MOV 45H,A
        CLR A
        ADDC A,DPH
        MOV 44H,A

        mov b,49h
        MOV A,B
        MOV DPTR,#Zero
        mov b,#08h
        mul ab

        ADD A,DPL
        MOV 57H,A
        CLR A
        ADDC A,DPH
        MOV 56H,A


        SJMP RET_
RET_:
        POP DPL
        POP DPH
        POP ACC
        RETI
;*****************************************
Zero:
DB      00000000B
DB      00111000B
DB      01000100B
DB      01000100B
DB      01000100B
DB      01000100B
DB      01000100B
DB      00111000B
One:
DB      00000000B
DB      00010000B
DB      00110000B
DB      00010000B
DB      00010000B
DB      00010000B
DB      00010000B
DB      00111000B
Two:
DB      00000000B
DB      00111000B
DB      01000100B
DB      00000100B
DB      00001000B
DB      00010000B
DB      00100000B
DB      01111100B
```

```
Three:
DB     00000000B
DB     00111000B
DB     01000100B
DB     00000100B
DB     00011000B
DB     00000100B
DB     01000100B
DB     00111000B
Four:
DB     00000000B
DB     00001000B
DB     00011000B
DB     00101000B
DB     01001000B
DB     01111100B
DB     00001000B
DB     00001000B
Five:
DB     00000000B
DB     01111100B
DB     01000000B
DB     01111000B
DB     00000100B
DB     00000100B
DB     01000100B
DB     00111000B
Six:
DB     00000000B
DB     00111000B
DB     01000100B
DB     01000000B
DB     01111000B
DB     01000100B
DB     01000100B
DB     00111000B
Seven:
DB     00000000B
DB     01111100B
DB     00000100B
DB     00001000B
DB     00010000B
DB     00100000B
DB     00100000B
DB     00100000B
Eight:
DB     00000000B
DB     00111000B
DB     01000100B
DB     01000100B
DB     00111000B
DB     01000100B
DB     01000100B
DB     00111000B
Nine:
DB     00000000B
DB     00111000B
DB     01000100B
DB     01000100B
DB     00111100B
DB     00000100B
```

```
DB     01000100B
DB     00111000B


Sign_1:
DB     00000000B
DB     00000000B
DB     00000000B
DB     00111111B
DB     00000000B
DB     00000000B
DB     00000000B
DB     00000000B

Sign_2:
DB     00000000B          ;line1
DB     01000000B          ;line2
DB     01100000B          ;line3
DB     11110000B          ;line4
DB     01100000B          ;line5
DB     01000000B          ;line6
DB     00000000B          ;line7
DB     00000000B          ;line8

BLANK:
DB     00000000B
DB     00000000B
DB     00000000B
DB     00000000B
DB     00000000B
DB     00000000B
DB     00000000B
DB     00000000B

wel_1:
DB     00000000B
DB     00000001B
DB     00000000B
DB     00000001B
DB     00000001B
DB     00000001B
DB     00000001B
DB     00000001B
wel_2:
DB     00000000B
DB     11100000B
DB     00000000B
DB     10100110B
DB     00100010B
DB     10100010B
DB     00100011B
DB     11100010B

wel_3:
DB     00000001B
DB     00011111B
DB     00000000B
DB     10001110B
DB     10010001B
DB     10010101B
DB     11010101B
```

```
DB     10011001B

wel_4:
DB     00000000B
DB     00000000B
DB     00000000B
DB     00000000B
DB     00000000B
DB     00000000B
DB     00000000B
DB     00000000B

wel_5:
DB     00110000B
DB     00011100B
DB     00000000B
DB     00111001B
DB     01010100B
DB     01000101B
DB     01010101B
DB     00100101B

wel_6:
DB     00000000B
DB     00000000B
DB     00000000B
DB     11100110B
DB     00100010B
DB     10100010B
DB     00100011B
DB     11100010B

wel_7:
DB     00000010B
DB     00000011B
DB     00000000B
DB     10000110B
DB     10001000B
DB     10001110B
DB     11000010B
DB     10000110B

wel_8:
DB     00000000B
DB     11000000B
DB     00000000B
DB     01100100B
DB     00100100B
DB     00100100B
DB     00100100B
DB     00111100B

       end
```

โปรแกรมภาษาแอสเซมบลีของ MCS-51 ในส่วนครัว

```
      ORG   0000H
      AJMP MAINNORMAL
;***********INTERRUPT***********
      ORG   0003H
      LJMP INT_0

      org   0023h
      ljmp  serial
      ORG   0030H
MAINNORMAL:
      MOV IE,#10010001B
;     SETB EA           ENABLE INTERUPT
;     SETB EX0    ENABLE EXTERNAL INTERRUPT 0
;     setb ES           enable Serial Port interrupt
      MOV TCON,#00000001B
;     SETB ITO    SET FALLING EDGE TYPE FOR INT0
      setb PS           ;define piority of serial port interrupt
      clr PX0
      clr IP.7          ;can assign priority

      MOV PCON,#00H           ;SMOD=0
      MOV SCON,#50H           ;MODE 1,REN=1
      MOV TMOD,#20H           ;TIMER1 MODE2,TIMER0 MODE0
      MOV TH1,#0FDH           ;9600BAUD
      SETB TR1          ;START TIMER1
;**************REGISTER***********
;20-71:REGISTER 20:ORDER,21:NO.
;72H:NO.OF ORDER
;70H:Table
;73H:BUFFER ORDER 74H:BUFFER NO.
;75H:POINTER
;76H:BUFFER NO.OF ORDER
;R1-R0:POINTER SHIFT
      LCALL INITIAL

      MOV 72H,#00H
MAIN0:
      MOV 75H,#20H
MAIN1:
      MOV A,75H
      CJNE A,#72H,MAIN1
      LJMP MAIN0
;*************INTERRUPT**********
INT_0:      PUSH PSW
      MOV R5,#0FFH
DEL:  LCALL DELAY
      LCALL DELAY
      LCALL DELAY
      DJNZ R5,DEL

      MOV A,72H
      CJNE a,#00H,CH_1
      LJMP RETINT
CH_1: CJNE a,#01H,ORDER
NOORDER:
      DEC 72H
      DEC 75H
      DEC 75H
      MOV DPTR,#TEXT_LINE01   ;
```

```
        MOV A,#80H
        LCALL LCDLDS
        MOV DPTR,#TEXT_LINE02    ;NO ORDER
        MOV A,#0C0H
        LCALL LCDLDS
        LJMP RETINT
ORDER:      MOV 76H,72H
        DEC 76H
        MOV A,76H
        MOV B,#02H
        MUL AB
        MOV 76H,A
        MOV R1,#20H
        MOV R0,#22H
        mov a,76h
WAIT: MOV A,@R0
        MOV @R1,A
        INC R0
        INC R1
        DJNZ 76H,WAIT
wait1:      DEC 72H
        DEC 75H
        DEC 75H
        MOV 73H,20H
        MOV 74H,21H
TOMENU:     MOV DPTR,#MENU000
        MOV A,73H
        MOV B,#10H
        MUL AB
        ADD A,DPL
        MOV DPL,A
        MOV A,B
        ADDC A,DPH
        MOV DPH,A
        MOV A,#80H
        LCALL LCDLDS
        MOV A,#8FH
        LCALL LCDWI
        MOV A,74H
        anl a,#0fh
        add a,#30h
        LCALL LCDWD
        MOV DPTR,#TEXT_LINE03
        MOV A,#0C0H
        LCALL LCDLDS
        MOV A,#0c3H
        LCALL LCDWI
        MOV A,74H
        anl a,#0f0h
        swap a
        add a,#30h
        LCALL LCDWD
;**********************
RETINT:     MOV A,75H
        POP PSW
        RETI
;***************RECEIVEDATA********
serial:
        PUSH PSW
        nop
wait_:      jnb RI,wait_                ;table
```

```
        mov a,sbuf
        CLR RI
        clr CY
        subb a,#40h
        swap a
        mov 74h,a
        jnb ri,$                    ;menu
        mov a,sbuf
        CLR RI
        mov 73h,a
        jnb ri,$                    ;Qdt
        mov a,sbuf
        CLR RI
        orl a,74h
        mov 74h,a
        MOV R1,75H
        MOV @R1,73H
        INC R1
        MOV @R1,74H
        INC R1
        INC 72H
        MOV 75H,R1
        MOV A,75H
        MOV A,72H
        CJNE A,#01H,RETS
        LJMP MENU
RETS:   LJMP RETINT1
MENU:
        MOV DPTR,#MENU000
        MOV A,73H
        MOV B,#10H
        MUL AB
        ADD A,DPL
        MOV DPL,A
        MOV A,B
        ADDC A,DPH
        MOV DPH,A
        MOV A,#80H
        LCALL LCDLDS
        MOV A,#8FH
        LCALL LCDWI
        MOV A,74H
        anl a,#0fh
        add a,#30h
        LCALL LCDWD
        MOV DPTR,#TEXT_LINE03
        MOV A,#0C0H
        LCALL LCDLDS
        MOV A,#0c3H
        LCALL LCDWI
        MOV A,21H
        anl a,#0f0h
        swap a
        add a,#30h
        LCALL LCDWD
;*************************************
RETINT1:
      MOV A,75H
      POP PSW
      RETI
```

```
INITIAL:
      MOV R2,#00H
LOOP:
      LCALL DELAY
      DJNZ R2,LOOP
      LCALL DELAY          ;wait for >15ms
      MOV A,#00111100B
      LCALL LCDWI
      MOV A,#00111100B
      LCALL LCDWI          ;INCREASE >100uS
      MOV A,#00111100B     ;8 bit 2 line
      LCALL LCDWI          ;INCREASE
      MOV A,#00001100B     ;DISP ON CURSOR ON BLINK OFF
      LCALL LCDWI
      MOV A,#00000001B     ;DISP CLEAR
      LCALL LCDWI
      MOV A,#00000110B     ;INCREASE ADDR
      LCALL LCDWI
      MOV R2,#05H
      LCALL DELAY
MAIN:
      MOV DPTR,#TEXT_LINE01
      MOV A,#80H
      LCALL LCDLDS
      MOV DPTR,#TEXT_LINE02
      MOV A,#0C0H
      LCALL LCDLDS
      RET
LCDLDS:
      MOV P1,#00H
      LCALL DELAY
      LCALL LCDWI
      MOV R2,#10H
LCDLDS1: CLR A
      MOVC A,@A+DPTR
      ACALL LCDWD
      INC DPTR
      DJNZ R2,LCDLDS1
      RET
;******************LCD WRITE INSTRUCTION***************
LCDWI:      PUSH DPH
      PUSH DPL
      MOV P0,A
      CLR ACC.0
      CLR ACC.1
      CLR ACC.2
      MOV P1,A
      SETB ACC.2
      MOV P1,A
      CLR ACC.2
      MOV P1,A
      MOV A,#00H
LCDWI1:     DEC A
      JNZ LCDWI1
      POP DPL
      POP DPH
      RET
;*************LCD WRITE DATA************************
;**************A=INPUT DATA*****************************
LCDWD:  PUSH DPH
      PUSH DPL
```

```
        MOV P0,A
;***************WRITE INSTRUCTION*****************
        SETB ACC.0
        CLR ACC.1
        CLR ACC.2
        MOV P1,A
        SETB ACC.2
        MOV P1,A
        CLR ACC.2
        MOV P1,A
;****************DELAY************************
        MOV A,#00H
LCDWD1:     DEC A
        JNZ LCDWD1
        POP DPL
        POP DPH
        RET
MENU000:DB "MENU000          "
MENU001:DB "TOAST            "
MENU002:DB "FRIED EGGS WITH  "
MENU003:DB "CHICKEN C SOUP   "
MENU004:DB "MUSHROOM C SOUP  "
MENU005:DB "CORN CREAM SOUP  "
MENU006:DB "MIXED SALAD      "
MENU007:DB "TUNA FISH SALAD  "
MENU008:DB "CHICKEEN SALAD   "
MENU009:DB "BEEF STEAK SALAD"
MENU010:DB "SEAFOOD SALAD    "
MENU011:DB "CHICKEN SANDWICH"
MENU012:DB "TUNA SANDWICH    "
MENU013:DB "HAM SANDWICH     "
MENU014:DB "HAM&CHEESE SANDW"
MENU015:DB "GARLIC BREAD     "
MENU016:DB "FRENCH FRIED     "
MENU017:DB "CLUB SANDWICH    "
MENU018:DB "CHICKEN STEAK    "
MENU019:DB "PORK STEAK       "
MENU020:DB "FISH STEAK       "
MENU021:DB "LONDONFISH STEAK"
MENU022:DB "FILLET STEAK     "
MENU023:DB "PEPPER STEAK     "
MENU024:DB "FILLET MIGNON    "
MENU025:DB "SPAGHETTI BEEF   "
MENU026:DB "SPAGHETTI PORK   "
MENU027:DB "SPAGHETTI CHICK  "
MENU028:DB "SPAGHETTI NAPOLI"
MENU029:DB "SPAGHETTI MUSSEL"
MENU030:DB "MACARONI SHRIMP  "
MENU031:DB "MACARONI HAM     "
MENU032:DB "PORK SAUSAGE     "
MENU033:DB "                 "
MENU034:DB "                 "
MENU035:DB "                 "
MENU036:DB "                 "
MENU037:DB "                 "
MENU038:DB "                 "
MENU039:DB "                 "
MENU040:DB "                 "
MENU041:DB "                 "
MENU042:DB "                 "
MENU043:DB "                 "
```

```
MENU044:DB "                "
MENU045:DB "                "
MENU046:DB "                "
MENU047:DB "                "
MENU048:DB "                "
MENU049:DB "                "
MENU050:DB "                "
MENU051:DB "ESPRESSO        "
MENU052:DB "MOCHA           "
MENU053:DB "LATTE           "
MENU054:DB "CAPUCCINO       "
MENU055:DB "ICE ESPRESSO    "
MENU056:DB "ICE MOCHA       "
MENU057:DB "ICE LATTE       "
MENU058:DB "ICE CAPUCCINO   "
MENU059:DB "FRAPPE ESPRESSO "
MENU060:DB "FRAPPE MOCHA    "
MENU061:DB "FRAPPE LATTE    "
MENU062:DB "FRAPPE CAPUCCINO"
MENU063:DB "                "
MENU064:DB "                "
MENU065:DB "                "
MENU066:DB "                "
MENU067:DB "                "
MENU068:DB "                "
MENU069:DB "                "
MENU070:DB "                "
MENU071:DB "HOT TEA         "
MENU072:DB "ICE THAI TEA    "
MENU073:DB "ICE TEA FRAPPE  "
MENU074:DB "ICE LEMON TEA   "
MENU075:DB "ICE CHOCOLATE   "
MENU076:DB "ICE CHOCO FRAPPE"
MENU077:DB "HOT FRESH MILK  "
MENU078:DB "ICE FRESH MILK  "
MENU079:DB "ORANGE JUICE    "
MENU080:DB "ORANGE FRAPPE   "
MENU081:DB "LEMON JUICE     "
MENU082:DB "LEMON FRAPPE    "
MENU083:DB "GUAVA MIXED     "
MENU084:DB "COKE            "
MENU085:DB "ICE CUBE / JUG  "
MENU086:DB "ICE CUBE / GLASS"
MENU087:DB "SINGHA BEER     "
MENU088:DB "HEINEKEN BEER   "
TEXT_LINE01:DB "   Cooking      "
TEXT_LINE02:DB "    NO ORDER    "
TEXT_LINE03:DB "Tab. PRESS SHIFT"
TEXT_LINE04:DB "        *-Cancel"
TEXT_LINE05:DB "1-Conf  *-Cancel"
TEXT_LINE06:DB "THIS ORDER 0000 "

;***************DELAY 2mS***************
DELAY:
      MOV R1,#04H
DELAY1: MOV R4,#00H
      DJNZ R4,$
      DJNZ R1,DELAY1
      RET
;***************delay 1-10s************
      END
```

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงได้ดีด้วยคำปรึกษา คำแนะนำ และความช่วยเหลือของอาจารย์ที่ปรึกษาทั้ง 2 ท่าน คือ ผศ. เกรียงไกร วงศ์โรจนภรณ์ และ รศ.ดร. สุวิพล สิทธิชีวภาค และอาจารย์ภาควิชาโทรคมนาคมทุกท่านที่คอยอบรมสั่งสอนทั้งด้านวิชาการและศีลธรรมตลอด 4 ปีที่ผ่านมา รวมถึงเพื่อนๆ ห้อง T312 ทุกคนที่คอยให้ความช่วยเหลือ ความบันเทิง และกำลังใจในการทำงาน สุดท้ายนี้ขอขอบคุณสถาบันแห่งนี้ที่ทำให้พวกเรามีสถานที่ศึกษาเล่าเรียนจนสำเร็จการศึกษา

นางสาวนภา แซ่เบ๊
นายพชร บุญญลักษม์
นางสาวสุภาณี โอศิริวัฒนกุล

หนังสืออ้างอิง

1. กฤดากร กล่อมการ, "การสื่อสารข้อมูล", คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, 2545
2. วิวัฒน์ กิรานนท์, "วิศวกรรมการสื่อสาร", คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, 2546
3. ยุทธนา ลีลาศวัฒนกุล, "คู่มือการเขียนโปรแกรมและใช้งาน Visual C++.NET", บริษัทด่านสุทธาการพิมพ์ จำกัด, 2546
4. รศ. สมยศ จุณณะปิยะ, "การประยุกต์ใช้งานไมโครคอนโทรลเลอร์ตระกูล MCS-51", คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, 2541
5. ณัฏฐพล วงศ์สุนทรชัย และคณะ, "ปฏิบัติการไมโครคอนโทรลเลอร์ PIC 12F675", อินโนเวตีฟเอ็กเพอริเมนต์, 2546
6. Kim G. House และ คณะ, "Printer connections bible", Indiana : Howard W. Sams, 1985
7. www.microchip.com
8. www.google.com