ระบบวิทยุติดตามตัวในสำนักงาน

OFFICE PAGER SYSTEM

โดย

นายเดชพล ชนะสมบูรณ์

นายปกรณ์ชัย ระวิพันธ์

นายสุทธิ ขำศรี

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2540

ระบบวิทยุติดตามตัวในสำนักงาน

OFFICE PAGER SYSTEM

โดย

นายเดชพล ชนะสมบูรณ์ 38013014

นายปกรณ์ชัย ระวิพันธ์ 38013019

นายสุทธิ ขำศรี 38013032

อาจารย์ที่ปรึกษา

รศ. สมยศ จุณณะปิยะ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2540

ปริญญานิพนธ์ปีการศึกษา 2540

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบวิทยุติดตามตัวในสำนักงาน

OFFICE PAGER SYSTEM

ผู้จัดทำ

1. นายเดชพล ชนะสมบูรณ์ 38013014
2. นายปกรณ์ชัย ระวิพันธ์ 38013019
3. นายสุทธิ ขำศรี 38013032

..... อาจารย์ที่ปรึกษา

(รศ. สมชาย จุฉะปิยะ)

ระบบวิทยุติดตามตัวในสำนักงาน
OFFICE PAGER SYSTEM

โดย นายเดชพล ชนะสมบูรณ์ 38013014
นายปกรณ์ชัย ระวิพันธ์ 38013019
นายสุทธิ ขำศรี 38013032

อาจารย์ที่ปรึกษา รศ.สมยศ จุณณะปิยะ

บทคัดย่อ

ในโครงงานนี้เป็นการสร้างระบบวิทยุติดตามตัวในสำนักงาน (Office Pager System) สำหรับใช้ใน สถานประกอบการที่มีพนักงานจำนานมากและพนักงานเหล่านี้ไม่มีห้องทำงานเป็นของตัวเอง เช่น พนักงาน บริการในโรงแรม การส่งข้อความจะส่งผ่านโอเปอเรเตอร์(Operator) โดยใช้คอมพิวเตอร์เป็นศูนย์กลางการจัดเก็บข้อมูล ข้อมูลที่จะส่งนั้นจะเป็นตัวเลขและตัวอักษร ใช้การมอดูเลต(Modulate) ในการส่งแบบ FSK (Frequency Shift Keying) ที่ตัวลูกจะส่งเสียงเตือนเมื่อมีการส่งข้อมูลมาและแสดงข้อความออกทางจอ LCD

ABSTRACT

This project is a presentation of office pager system for use in place or company that have a lot of official and they don't have own working room such as servitor in the hotel. Messages are sent via operater by using computer as center of memory and data sending. Sending information are numbers and letters send by using FSK (Frequency Shift Keying) modulation. When information are send to mobile, the mobile will ring and messages are shown on LCD.

บทที่ 1
บทนำ

1.1 ทฤษฎีทั่วไปของระบบวิทยุติดตามตัว

นับเป็นเวลาหลายปีแล้ว ที่ได้มีการเปิดให้บริการวิทยุติดตามตัว โดยบริษัทเทเลซิสเป็นรายแรกในประเทศไทย จากวันนั้นเป็นต้นมาตลาดของอุปกรณ์สื่อสารประเภทนี้ได้มีการขยายตัวอย่างรวดเร็วจำนานผู้ให้บริการมีมากขึ้นซึ่งจากเดิมมีเพียงรายเดียวในด้านความหลากหลายของการให้บริการก็มีมากขึ้นทั้งการให้บริการข่าวสาร การให้บริการเชื่อมโยงข่าวสารกับระบบโทรศัพท์เคลื่อนที่รูปลักษณ์ใหม่ๆ ของเครื่องลูกข่ายมีปรากฏให้เห็นตามแผ่นพับโฆษณาอยู่เสมอจนแทบกล่าวได้ว่าสงครามการห้ำหั่นเพื่อแย่งชิงผู้ใช้บริการยังคงระอุอยู่ในเมืองไทย ด้วยเหตุดังกล่าวการนำเรื่องราวเทคนิคของระบบวิทยุติดตามตัวซึ่งนิยมใช้ในปัจจุบัน

วิทยุติดตามตัวเป็นอุปกรณ์สื่อสารแบบพกพาชนิดหนึ่งเช่นเดียวกับโทรศัพท์เคลื่อนที่ ผิดกันแต่เพียงว่ารูปแบบในการสื่อสารข้อมูลของวิทยุติดตามตัวจะเป็นไปในลักษณะของการฝากข่าวสารไว้ที่ศูนย์รับฝากข้อความโดยศูนย์บริการวิทยุติดตามตัวจะทำหน้าที่ส่งข่าวสารนั้นไปยังเครื่องลูกข่ายที่ถูกระบุหมายเลขไว้ หลายท่านเข้าใจว่าเครื่องลูกข่ายมีหน้าที่เพียงรับ ข่าวสารซึ่งถูกส่งจากศูนย์วิทยุติดตามตัวเท่านั้น ในความเป็นจริงแล้ววิทยุติดตามตัวบางระบบ มีการกำหนดความสามารถของเครื่องลูกข่าย ให้ส่งข่าวสารผ่านกลับมาให้ศูนย์บริการวิทยุ ติดตามตัวได้ สำหรับข่าวสารที่สามารถส่งผ่านเครือข่ายวิทยุติดตามตัวนั้นมีตั้งแต่ตัวเลข ตัวอักษร และเสียงพูด ทั้งนี้ขึ้นอยู่กับการเลือกใช้บริการส่งข่าวสารประเภทใด

1.2 ประเภทของระบบวิทยุติดตามตัว

ในแง่ของการให้บริการ สามารถแบ่งระบบวิทยุติดตามตัวออกเป็น 2 ประเภทใหญ่ๆคือการให้บริการระบบวิทยุติดตามตัวแบบเฉพาะที่(Local area หรือ On - site paging)และการให้บริการระบบวิทยุติดตามตัวแบบพื้นที่ครอบคลุมกว้าง(Wide area paging)

1.2.1 การให้บริการระบบวิทยุติดตามตัวแบบเฉพาะที่(Local Area)

เป็นการให้บริการวิทยุติดตามตัวในขอบเขตพื้นที่จำกัดเช่นภายในอาคารโรงงานหรือโรงพยาบาล ลักษณะของการแจ้งผลต่อผู้ใช้มีทั้งแบบเป็นข้อความ , เสียงพูดเสียงเตือน หรือรูปแบบทั้งสามชนิด สามารถแบ่งประเภทของวิทยุติดตามตัวแบบเฉพาะที่ได้เป็น 4 ระบบ ตามรูปแบบเทคนิคโดยรายละเอียดของการใช้ความถี่ตามตารางที่1.1

แบบวงรอบเหนี่ยวนำ (Induction Loop) ใช้ความถี่ในช่วง 16 ถึง 150 กิโลเฮิรตซ์ (kilohertz) ระบบนี้เหมาะสมสำหรับใช้งานในพื้นที่ซึ่งมีอาณาไม่กว้างมากนัก คือประมาณไม่เกิน 6 ช่วงตึก โดยจะใช้วิธีเดินสายนำสัญญาณรอบๆ บริเวณที่กำหนดให้เป็นพื้นที่ให้บริการ การส่งสัญญาณจะใช้การ เหนี่ยวนำให้เกิดสนามแม่เหล็กไฟฟ้าขึ้น ดังแสดงในรูปที่ 1.1 โดยทั่วไปมักจะใช้สายโคเอกเชียลความรั่วไหลสูง(Leaky Coaxial Cable) เป็นสายนำสัญญาณในระยะแรกของระบบแบบนี้ถูกใช้งานในย่านความถี่ต่ำ เครื่องลูกข่ายแต่ละเครื่องจะถูกจูนความถี่ไปยังความถี่ซึ่งแพร่กระจายออกมาจากสายเคเบิลนั้น พบว่าความสามารถในการให้บริการของระบบ ในช่วงเวลาดังกล่าวถูกจำกัดสัญญาณ ระบบนี้ใช้ความถี่ในช่วง 16 ถึง 150 กิโลเฮิรตซ์สำหรับระบบที่ยอมให้ผู้ใช้ปลายทางส่งข่าวสารย้อนกลับมายังศูนย์บริการวิทยุติดตามตัวได้ จะใช้ความถี่ทางด้านกลับในช่วง 161.10 ถึง

161.50 กิโลเฮิรตซ์ นอกจากจะใช้งานสำหรับ ระบบวิทยุติดตามตัวแล้วยังสามารถนำรูปแบบของวงรอบเหนี่ยวนำไปประยุกต์ใช้ในกิจการ อื่นๆ เช่น การเพิ่มประสิทธิภาพของการรับฟังผู้มีปัญหาทางด้านการรับฟัง โดยเดินสายความรั่วไหลสูงรอบๆ ห้องเพื่อทำหน้าที่เหนี่ยวนำสัญญาณเสียงจากเครื่องรับโทรทัศน์ แล้วให้ทำการขยายสัญญาณดังกล่าวก่อนที่จะถูกรับฟังโดยเครื่องรับฟัง

ตารางที่ 1.1 การกำหนดความถี่ใช้งานของวิทยุติดตามตัวแบบเฉพาะที่

ย่านความถี่	ความถี่จากสถานีส่งไปเครื่องลูกข่าย	ความถี่จากเครื่องลูกข่ายส่งไปสถานี
16 - 150KHz	16 - 150 KHz	161.10 - 161.65 MHz
	ใช้กับระบบวิทยุติดตามตัวแบบวงรอบเหนี่ยวนำ	
ย่าน HF 26 MHz	26.2375 - 26.8655 MHz	ไม่อนุญาตให้มีการส่ง
ย่าน HF 27 MHz	26.978 - 27.262 MHz	ไม่อนุญาตให้มีการส่ง
ย่าน HF 31 MHz	31.725, 31.750,31.775 MHz	161.00 - 161 .10 MHz
	สำหรับให้บริการในเขตโรงพยาบาล	
ย่าน VHF 49 MHz	49.0000 - 49.4875 MHz	ไม่อนุญาตให้มีการส่ง
ย่าน VHF 49 MHz	49.4250 , 49.4375 , 49.4500 MHz 49.4625 , 49.750 MHz	161.00 - 161.10 MHz
	สำหรับให้บริการในเขตโรงพยาบาล	
ย่าน UHF 459 MHz	459.125 - 459.450 MHz	ไม่อนุญาตให้มีการส่ง
ย่าน UHF 459 MHz และย่าน VHF 161 MHz	กรณีใช้เฉพาะความถี่ UHF	
	459.125 - 459.475 MHz	161.0000 - 161.1125 MHz
	กรณีใช้ความถี่ UHF ร่วมกับ VHF	
	459.325 MHzกับ 161.0125 MHz	459.125 MHzกับ 161.000 MHz
	459.475 MHzกับ 161.1125 MHz	459.450 MHzกับ 161.100 MHz
	ใช้สำหรับระบบวิทยุติดตามตัวแบบเฉพาะที่	

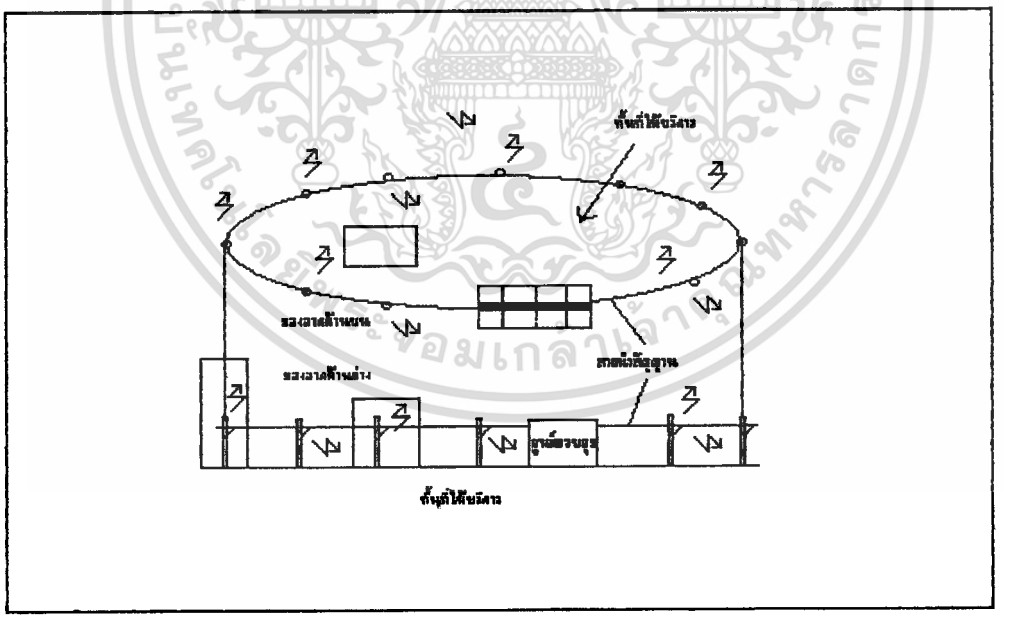
แบบใช้งานย่านความถี่ HF ประมาณ 26 ถึง 31 เมกะเฮิรตซ์ (megahertz) เหมาะสมใช้งานที่ครอบคลุมที่บริเวณกว้าง เช่น ภายในโรงงานและโรงพยาบาล เป็นระบบที่ได้รับความนิยมใช้งานมากที่สุด การส่งสัญญาณเป็นไปโดยการแพร่กระจายของคลื่นวิทยุจากสายอากาศ โดยตรงนับตั้งแต่ปี พ.ศ. 2528 เป็นต้นมาระบบดังกล่าวได้ถูกจำกัดให้ส่งเฉพาะข่าวสาร ประเภทที่ไม่ใช่เสียงพูดเท่านั้น สำหรับการงานใช้ในเขตโรงพยาบาล มีการกำหนดให้ใช้งานได้เฉพาะที่ความถี่ 31 และ 49 เมกะเฮิรตซ์เท่านั้น เนื่องจากทั้ง 2 ความถี่ไม่ก่อให้เกิดการรบกวนต่ออุปกรณ์อิเลคทรอนิกส์ที่ใช้ภายในโรงพยาบาลแต่อย่างใด อีกทั้งยังมีการ อนุญาติให้สามารถให้สามารถ

ส่งข้อความประเภทที่เป็นเสียงพูดทั้งในทิศทางไปและกลับ จากเครื่องลูกข่ายสำหรับย่านความถี่ทั้งสองอีกด้วย โดยความถี่ที่ใช้ในการส่งข่าวสารย้อนกลับ อยู่ในช่วง 161 ถึง 161.100 กิโลเฮิรตซ์

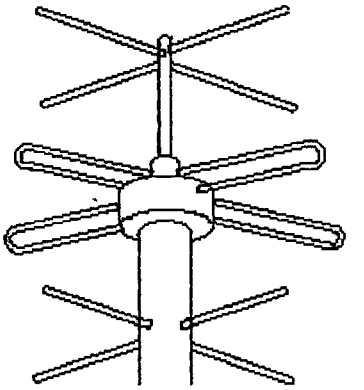
แบบใช้งานย่านความถี่ UHF ประมาณ 459 เมกะเฮิรตซ์ เหมาะสำหรับใช้งานในตึกที่มี กำแพงหนาหรือภายในบริเวณโรงงานอุตสาหกรรมขนาดใหญ่ เนื่องจากความถี่ในย่าน UHF มีความสามารถในการทะลุทะลวงสูง สามารถส่งข่าวสารเฉพาะที่ไม่ใช่เสียงพูดและไม่อนุญาติให้ผู้ใช้ส่งข่าวสารย้อนกลับไปยังศูนย์บริการวิทยุติดตามตัวได้

แบบสื่อสารเฉพาะท้องถิ่นใช้ความถี่ในช่วง 459 เมกะเฮิรตซ์ สำหรับส่งข่าวสารทุกประเภท รวมถึงเสียงพูดไปยังลูกข่าย และรับข่าวสารประเภทเสียงพูดจากเครื่องลูกข่าย ในช่วงความถี่161 เมกะเฮิรตซ์ นับเป็นระบบที่ค่อนข้างใหม่ซึ่งได้รับอนุญาติให้ใช้งานได้ในปี พ.ศ. 2528 ระบบดังกล่าวถูกสร้างขึ้นเพื่อใช้แทนระบบแบบเฉพาะที่ทั้งย่านความถี่แบบ HF และ UHF ซึ่งใช้ในกิจการโรงพยาบาล รวมถึงแบบวงรอบเหนี่ยวนำด้วย นอกจากนี้ยัง สนับสนุนสื่อสารสองทิศทางชนิดสมบูรณ์แบบโดยผู้ใช้งานจะต้องอยู่ภายในรัศมีทำการรอบๆสถานีส่ง

ระบบวิทยุติดตามตัวแบบเฉพาะที่ถูกออกแบบขึ้นเพื่อจุดประสงค์สำหรับใช้งานในพื้นที่เฉพาะโดยระบบมีความสามารถในการรับรองเครื่องลูกข่ายในระดับต่ำตั้งแต่ 10 ถึง 2,000 เครื่อง ความห่างของช่องสัญญาณแต่ละช่องในย่าน VHF และ UHF มีค่าเท่ากับ 25 กิโลเฮิรตซ์ และ 12.5 กิโลเฮิรตซ์ สำหรับกรณีของแบบที่ใช้ความถี่ในช่วง 49 กิโลเฮิรตซ์ เนื่องจากแต่ละช่องสัญญาณมีแบนด์วิดท์ (bandwidth) ค่อนข้างแคบจึงส่งผลให้ความสามารถในการตรวจรับสัญญาณของระบบต่ำ ส่งผลให้เครื่องรับถูกออกแบบอย่างซับซ้อนมากขึ้น ทำให้มีราคาแพง

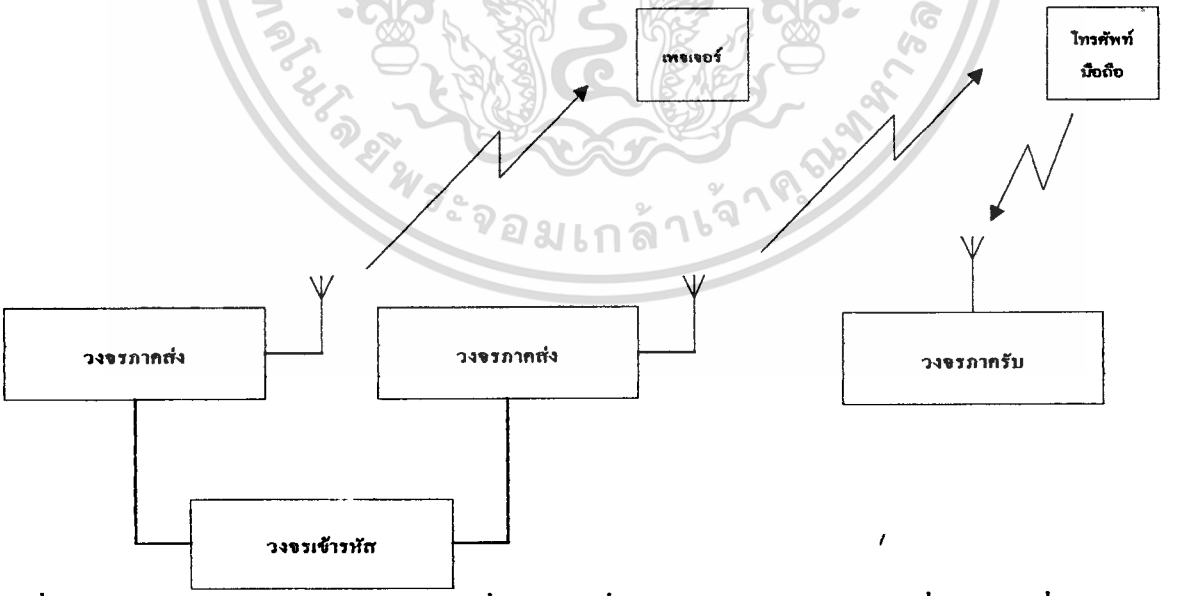


รูปที่ 1.1 การให้บริการวิทยุติดตามตัวแบบที่ใช้วงรอบเหนี่ยวนำ



รูปที่ 1.2 สายอากาศระบบเพจจิ้งท้องถิ่นซึ่งใช้รูปแบบแพร่กระจายของคลื่นเป็นลักษณะรูปโดนลงสู่พื้น

สำหรับการควบคุมขอบเขตการแพร่กระจายของคลื่นวิทยุทำได้โดยการกำหนดกำลังส่ง ของสถานีส่งให้มีค่าต่ำ โดยทั่วไปมักมีค่าเท่ากับ 25 มิลลิวัตต์ (milliwatt) สายอากาศของสถานีส่งจะถูกติดตั้งในลักษณะที่ ทำให้รูปแบบการแพร่กระจายคลื่นเป็นรูปโดนในทิศทางลงสู่พื้น และมีการแพร่กระจายคลื่นในแนวราบรูปแบบของสายอากาศที่ใช้งานทั่วไปเป็นไปตามรูปที่ 1.2 อย่างไรก็ตามเนื่องจากระบบวิทยุติดตามตัวแบบเฉพาะที่โดยส่วนใหญ่มักประกอบไปด้วยระบบและรูปแบบการแพร่กระจายคลื่นมากกว่า 1 อย่าง ทำให้การออกแบบในทางปฏิบัติมี ความซับซ้อนมากกว่าที่ได้กล่าวมา รูปที่ 1.3 แสดงถึงเครื่องลูกข่ายในระบบวิทยุติดตามตัว แบบเฉพาะที่ซึ่งสามารถส่งข่าวสารกลับมายังศูนย์บริการวิทยุติดตามตัวได้ โดยจะต้องมีการ ติดตั้งสายอากาศและเครื่องรับเพิ่มเติมที่สถานีส่ง



รูปที่ 1.3 เปรียบเทียบความแตกต่างระหว่างเครื่องลูกข่ายที่รับข่าวสารได้อย่างเดียวกับเครื่องลูกข่ายที่สามารถส่งข่าวสารได้ด้วย

1.2.2 การให้บริการระบบวิทยุติดตามตัวแบบพื้นที่ครอบคลุมกว้าง(Wide area paging)

เป็นระบบวิทยุติดตามตัวที่อนุญาติให้มีการส่งข่าวสารในทิศทางเดียว จากสถานีส่งไปยังเครื่องลูกข่ายซึ่งอยู่ภายในพื้นที่ให้บริการ โดยส่วนมากพื้นที่ให้บริการจะมีรัศมีประมาณ 5 ถึง 20 กิโลเมตร (kilometer) รอบสถานีส่งทั้งนี้ขึ้นอยู่กับตำแหน่งกำลังส่งและระบบสายอากาศของสถานี สัญญาณที่ถูกส่งออกจากสายอากาศของสถานีส่งจะมีการลดทอน ซึ่งเกิดจากการแพร่กระจาย คลื่นวิทยุความถี่สูงผ่านอาคารสูงหรือร่างกายมนุษย์ ผลจากการทดสอบเพื่อแก้ไขปัญหาด้านการลดทอนของคลื่น ชี้ให้เห็นว่าความถี่ในช่วง 80 ถึง 460 เมกะเฮิรตซ์ มีความเหมาะสมสำหรับใช้กับกิจการวิทยุติดตามตัว โดยเกิดผลจากการลดทอนสัญญาณต่ำที่สุด การลดทอน ของสัญญาณซึ่งเกิดจาการแพร่กระจายคลื่นวิทยุภายในอาคารจะมีค่าอยู่ในช่วง14ถึง18เดซิเบล (decibel) ที่ความถี่ 150 เมกะเฮิรตซ์, 18 เดซิเบลที่ความถี่ 250 เมกะเฮิรตซ์ และ 12 ถึง 18 เดซิเบล ที่ความถี่ 400 เมกะเฮิรตซ์ นอกจากปัญหาการลดทอนของสัญญาณแล้ว ยังมีปัจจัยอื่นๆซึ่งมีผลกระทบต่อประสิทธิภาพในการรับสัญญาณของเครื่องลูกข่ายอีก เช่น ระดับความแรงของสัญญาณรบกวนในพื้นที่ให้บริการ ซึ่งส่วนใหญ่เกิดมากในพื้นที่ที่เป็นเมืองใหญ่ๆ โดยสัญญาณรบกวนมักจะเกิดจากเครื่องยนต์หรือเครื่องจักรกล สัญญาณรบกวนลักษณะนี้จะมีขนาดความแรงของการรบกวนลดลงเมื่อพิจารณาที่ความถี่ใช้งานค่าสูง ทำให้ต้องเลือกใช้ความถี่สูงสำหรับส่งข่าวสารในเขตเมือง ความไวของการรับสัญญาณของเครื่องลูกข่าย ซึ่งมีความเกี่ยวพันโดยตรงกับการออกแบบวงจรของเครื่อง และประสิทธิภาพของสายอากาศในเครื่องลูกข่ายเอง

ระบบวิทยุติตามตัวแบบพื้นที่ครอบคลุมมีการใช้ความถี่สำหรับส่งข่าวสารทั้งในย่าน VHF และUHF โดยการกำหนดความถี่ในช่วง 138 และ 153 เมกะเฮิรตซ์ สำหรับย่าน VHF และ 454 เมกะเฮิรตซ์ สำหรับย่าน UHF ระบบที่ใช้ย่านความถี่ VHF จะสามารถส่งข่าวสาร ที่เป็นทั้งข้อความ (message) หรือเป็นเสียงเตือน (tone) เท่านั้น สำหรับการส่งข่าวสารที่เป็นทั้งข้อความ เสียงเตือน และเสียงพูดจะใช้ได้กับระบบ UHF สำหรับมาตรฐานการส่งข่าวสารของระบบวิทยุติดตามตัวแบบพื้นที่ครอบคลุมกว้างจะเป็นไปตามข้อกำหนดซึ่งเรียกว่ามาตรฐาน POCSAG

ปีพ.ศ. 2519 ณ กรุงลอนดอนมีการรวมตัวกันของกลุ่มวิศวกรนานาชาติเพื่อดำเนินการวางข้อกำหนดของระบบวิทยุติดตามตัวตามแบบพื้นที่ครอบคลุมกว้าง การประชุมจัดขึ้นโดย ใช้สถานที่ของศูนย์ทำการไปรษณีย์กลางของประเทศอังกฤษ ดังนั้นจึงเรียกข้อกำหนดนี้ตามชื่อสถานที่ว่า Post Office Code Standardization Advisory Group หรือ POGSAG สำหรับชื่อ ที่เป็นทางการของมาตรฐานดังกล่าวซึ่งตั้งให้ภายหลังโดยหน่วยงาน CCIR คือ Radio Paging Code No. หรือ RPC No.1 มาตรฐานรหัสการเพจ (page) โดยใช้คลื่นวิทยุหมายเลข 1

1.3 ชนิดของระบบวิทยุติดตามตัว

1.3.1 แบบตัวอักษร(Message หรือ Alpha-Numeric Pager) เป็นแบบที่ได้รับความนิยมอย่างสูงในบ้านเรา เครื่องลูกข่ายประเภทนี้สามารถรับข้อความได้เต็มที่ 200 ตัวอักษรและแสดงผลได้หน้าละ 80 ตัวอักษรต่อหน้าข่าวสาร โดยตัวอักษรในที่นี้จะหมายถึงตัวอักษรภาษาอังกฤษ หากเป็นอักษรไทยจะได้จำนวนน้อยกว่านี้ หน่วยความจำสำรองสามารถเก็บ ข่าวสารได้ 40 ชุด โดยผู้ใช้สามารถสั่งลบหรือป้องกันการลบได้ด้วยตนเอง บริการพิเศษซึ่งได้รับความนิยมมากสำหรับเครื่องลูกข่ายประเภทนี้คือการให้บริการข่าวสารแบบออนไลน์ (online) เช่น อัตราแลกเปลี่ยนการเงิน,ข่าวการเมือง,ข่าวกีฬาหรืออื่นๆ เป็นต้น

1.3.2 แบบตัวเลข(Numeric Pager) รับข่าวสารได้เฉพาะตัวเลข โดยจะเก็บตัวเลขที่ทำการส่งมาได้สูงสุด 20 หลักต่อหนึ่งข่าวสาร ตัวเลขเหล่านี้อาจจะเป็นหมายเลขโทรศัพท์ที่จะให้ติดต่อกลับ, ราคาสินค้า ,รหัสสินค้า หรือรหัสพิเศษเฉพาะกลุ่มแล้วแต่ผู้ใช้บริการ

1.3.3 แบบใช้เสียง(Tone Pager) เป็นเครื่องลูกข่ายที่มีราคาถูกที่สุด โดยการส่งเสียงเตือนเมื่อมีการติดต่อไปยังเครื่องหมายเลขนั้น ทั้งนี้อาจใช้เป็นการเตือนให้เจ้าของเครื่องโทรติดต่อ กลับศูนย์บริการวิทยุติดตามตัว หรือใช้เป็นการเตือนให้โทรกลับไปยังหมายเลขใดหมายเลขหนึ่ง ในกรณีที่มีการตกลงล่วงหน้า

1.3.4 แบบหลายเสียง(Multi Address Pager) คล้ายกับแบบใช้เสียงแต่เครื่องประเภทนี้จะให้เสียงแตกต่างกัน 2 เสียงขึ้นไปเพื่อใช้เป็นการแยกความแตกต่างของหมายเลขที่จะให้โทรติดต่อกลับ

1.3.5 แบบฝากเสียงพูด(Voice Messaging Pager) อีกรูปแบบหนึ่งของเครื่องลูกข่าย โดยที่ผู้ฝากข่าวสารสามารถฝากเสียงพูดของตนผ่านทางคู่สายโทรศัพท์ไปเก็บไว้ยังศูนย์ฝากข้อความเพื่อให้ศูนย์ บริการวิทยุติดตามตัว ส่งข้อความนั้นไปยังเจ้าของเครื่อง

เครื่องลูกข่ายที่มีจำหน่ายในปัจจุบันอาจมีความหลากหลายมากกว่าทั้ง 5 แบบ ที่ได้กล่าวมา ซึ่งอาจมีการผสมความสามารถแต่ละแบบเข้าด้วยกัน สำหรับเวลาที่ใช้ในการส่งข่าวสารนั้นมีความแตกต่างกันไปตามชนิดของเครื่องลูกข่าย กล่าวคือใช้เวลาตั้งแต่ 10 วินาทีลงมา สำหรับเครื่องลูกข่ายแบบฝากเสียงพูด จนถึงมิลลิวินาที สำหรับแบบที่ไม่ใช่เสียงพูด ความพิเศษของระบบวิทยุติดตามตัวก็คือความสามารถในการติดต่อลูกข่ายได้มากกว่า 100,000 รายต่อหนึ่งความถี่ โดยระบบที่ไม่ใช่เสียงพูดสามารถครอบคลุมพื้นที่ให้บริการได้กว้างกว่าแบบใช้เสียงพูด ทั้งนี้เพราะใช้กำลังส่งน้อยกว่าในการทำให้เครื่องทำงาน และตัวเครื่องรับเองมีความไวสูงมากเป็นพิเศษกับสัญญาณในช่วงสั้นๆ ในระดับมิลลิวินาที

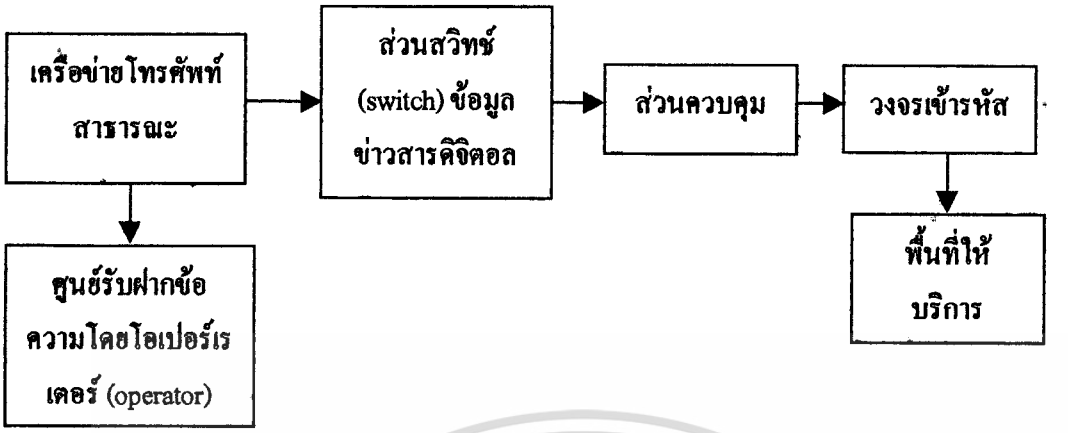
ในระบบวิทยุติดตามตัวหนึ่ง ผู้ใช้บริการสามารถใช้เครื่องลูกข่ายรวมกันได้ทั้ง 5 แบบ ดังที่ได้กล่าวมาผู้ให้บริการจะแยกความถี่ใช้งานสำหรับเครื่องลูกข่ายแต่ละแบบ ทั้งนี้เพื่อให้ ได้คุณภาพสัญญาณที่ดี ข้อดีของวิทยุติดตามตัวเมื่อเปรียบเทียบกับระบบสื่อสารเคลื่อนที่ชนิดอื่นที่เป็น 2 ทิศทาง เช่น โทรศัพท์เคลื่อนที่ ได้แก่

- สามารถพกติดตัวได้สะดวกมากเนื่องจากมีขนาดเล็ก น้ำหนักเบา
- มีจำนวนผู้เช่าได้มากกว่าต่อหนึ่งความถี่
- สามารถใช้เครื่องลูกข่ายต่างยี่ห้อแทนกันได้โดยเป็นเครื่องประเภทเดียวกัน
- เครื่องลูกข่ายส่วนใหญ่สามารถเก็บข่าวสารที่ได้รับแล้วไว้ได้

1.4 โครงสร้างของระบบเครือข่ายวิทยุติดตามตัว

เมื่อพิจารณากระบวนการในการสื่อสารโดยการใช้วิทยุติดตามตัวจะพบว่าเกิดจากการที่ ผู้ใช้โทรศัพท์ติดต่อมายังศูนย์รับฝากข้อความ พนักงานประจำศูนย์วิทยุติดตามตัวจะป้อนข่าวสารนั้นผ่านอุปกรณ์ต้นทาง หรือผู้ใช้โทรศัพท์อาจจะโทรมายังเลขหมายฝากข้อความอัตโนมัติโดยไม่ผ่านพนักงานประจำศูนย์บริการวิทยุติดตามตัว ข้อความทั้งสองแหล่งจะถูกเก็บไว้ โดยผ่านการเข้ารหัสและการส่งออกอากาศไปยังเครื่องลูกข่าย ดังนั้นจึงสามารถวาง โครงสร้างที่เป็นพื้นฐานของระบบวิทยุติดตามตัวได้ดังรูปที่ 1.4 ซึ่งประกอบด้วยส่วนการรับข่าวสาร, คอมพิวเตอร์ส่วนกลางและเครือข่ายอุปกรณ์สื่อสาร โดยในรูปที่ 1.5 เป็นการแสดงถึงรายละเอียดของอุปกรณ์ภายในเครือข่ายวิทยุติดตามตัว ซึ่งกล่าวถึงดังนี้

1.4.1 ส่วนการรับข่าวสาร ข่าวสารที่ถูกส่งดังกล่าวสามารถมาได้จากหลายเส้นทางคือ โอเปอเรเตอร์



รูปที่ 1.4 โครงสร้างพื้นฐานของระบบวิทยุติดตามตัว

ที่ศูนย์รับฝากข้อความส่วนกลางทำการรับข้อความจากผู้ติดต่อเข้ามาทางโทรศัพท์แล้วทำการ ป้อนข้อความเหล่านั้นผ่านคีย์บอร์ด โอเปอเรเตอร์ที่ศูนย์รับฝากข้อความสาขาย่อยทำการส่งข้อความมาศูนย์บริการวิทยุติดตามตัวส่วนกลางโดยผ่านคู่สายโทรศัพท์เช่าหรือคู่สายสาธารณะ ข้อความที่ส่งจากระบบฝากข้อความอัตโนมัติ

1.4.2 คอมพิวเตอร์ส่วนกลาง ทำหน้าที่ควบคุมการปฏิบัติงานของอุปกรณ์ทั้งระบบและยังทำหน้าที่เป็นส่วนควบคุมข่าวสารอีกด้วยข่าวสารที่ถูกส่งมาจากส่วนการรับข่าวสารจะถูกป้อนเข้าสู่คอมพิวเตอร์ส่วนกลางโดยผ่านอุปกรณ์เชื่อมต่อต่างกัน ส่วนควบคุมข่าวสารจะมีฮาร์ดดิสก์ขนาดใหญ่ทำหน้าที่เก็บบันทึกข่าวสารที่จะถูกส่งออกอากาศ แต่ละข่าวสารจะถูกเก็บไว้ในฮาร์ดดิสก์เป็นช่วงเวลาหนึ่งหรืออาจมีการเก็บอย่างถาวรในเท็ปแบ็กอัป นอกจากข่าวสารเหล่านี้แล้วฮาร์ดดิสก์ยังมีการเก็บซอฟแวร์ (software) ซึ่งใช้ควบคุมการทำงานของระบบอีกด้วย จึงนับเป็นความสำคัญอย่างยิ่งที่จะต้องมีส่วนควบคุมข่าวสารสำรองเพื่อทำหน้าที่แทน ส่วนควบคุมข่าวสารหลัก ในกรณีที่เกิดการเสียหายขึ้น

เครือข่ายอุปกรณ์สื่อสาร ประกอบด้วย

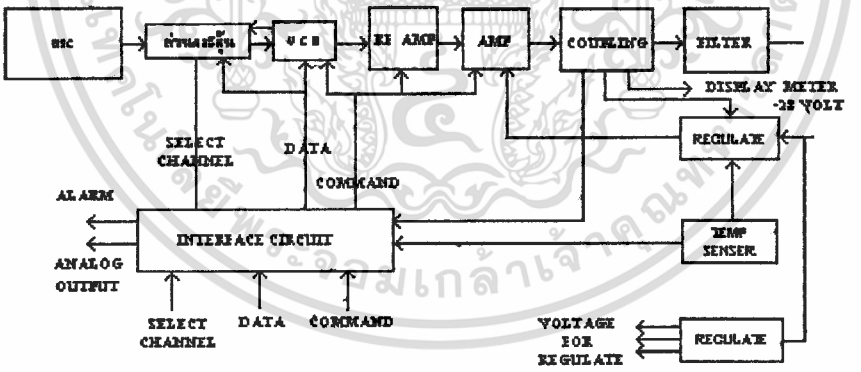
- ส่วนเข้ารหัสข้อมูล
- ส่วนควบคุมอุปกรณ์เครื่องส่งคลื่นวิทยุ
- ส่วนควบคุมการซิงโครนัส (synchronous) ข่าวสารทางคลื่นวิทยุ
- ส่วนบริหารเครือข่าย จะมีหรือไม่มีก็ได้

ส่วนการเข้ารหัสข้อมูลตามมาตรฐาน POCSAG จะรับข่าวสารที่ถูกส่งมาจากส่วนควบคุมข่าวสารในรูปแบบ ASCII ส่วนเข้ารหัสข้อมูลจะแปลงข่าวสารให้เป็นข้อมูลตามมาตรฐาน POCSAG ซึ่งสามารถกำหนดระดับความสำคัญของข่าวสาร (Priority) และรูปแบบการส่งข่าวสารได้ ข่าวสารซึ่งถูกกำหนดความสำคัญสูงสุดจะถูกเขียนลงในบัฟเฟอร์ (buffer) โดยมีการแบ่งแต่ละข้อมูลออกเป็นส่วนๆ ตามรูปแบบโครงสร้างของกลุ่มข้อมูล โดยทั่วไป บัฟเฟอร์หน่วยความจำ จะมีความจุ 64 กิโลไบต์ (kilobyte) ซึ่งสามารถขยายความจุขึ้นได้ถึง 1 เมกะไบต์ (megabyte) การที่ต้องมีวงจรกันชนในส่วนนี้ก็เพื่อป้องกันการสูญหายของข้อมูลในช่วงที่มีปริมาณการส่ง

ข้อมูลหนาแน่นมากๆ สำหรับกลุ่มข้อมูลรหัสว่าง และปรีแอมเบอร์ (preamble) ซึ่งถูกส่งออกอากาศโดยสถานีส่งจะถูกสร้างขึ้นและแทรกร่วมกับข้อมูลข่าวสารโดยส่วนเข้ารหัสข้อมูล

สำหรับข้อมูลที่กำหนดให้ส่งด้วยอัตราความเร็ว 512 และ 1,200 บิตต่อวินาที (bit per second) จะถูกจัดไว้ด้วยโดยส่วนที่เข้ารหัสและข้อมูลเหล่านี้จะถูกส่งในรูปของรหัส POCSAG ไปยังเครื่องส่งทันทีที่ได้รับคำสั่งจากส่วนควบคุมตามพื้นที่ สำหรับส่วนบริหารเครือข่าย (ถ้ามี) จะเป็นส่วนหนึ่งของส่วนควบคุมตามพื้นที่ ซึ่งทำหน้าที่ตรวจสอบการทำงานของอุปกรณ์ควบคุมการส่ง อุปกรณ์ควบคุมการส่งจะถูกติดตั้งไว้ที่สถานีส่งทุกสถานี ซึ่งส่วนควบคุมตามพื้นที่ สามารถติดต่อกับอุปกรณ์ควบคุมการส่งได้สูงสุด 99 ชุด โดยที่การติดต่อจะรวมถึงการควบคุมการทำงานระยะไกลด้วย จึงนับเป็นความสะดวกของผู้บริหารระบบซึ่งสามารถสั่งการควบคุมสถานีส่งทุกสถานีได้จากศูนย์ควบคุมส่วนกลาง

หน้าที่หลักของอุปกรณ์ควบคุมการส่งคือ การตรวจสอบ, การควบคุม และการชดเชย เวลาที่จะหน่วง ให้กับภาคส่ง รวมทั้งรับคำสั่งจากส่วนควบคุมตามพื้นที่ สำหรับภาคส่งจะทำการส่งข่าวสารโดยการมอดูเลต (modulate) แบบ FSK โดยทั่วไปใช้ผลึกคริสตัล (crystal) สำหรับควบคุมภาค ออสซิเลเตอร์ (oscillator) เพื่อความเที่ยงตรงของฐานเวลา และยังเป็นการเพิ่มประสิทธิภาพของการทำงานแบบกึ่งซิงโครนัสด้วย ปกติจะกำหนดให้สัญญาณที่ส่งออกจากสายอากาศนั้นควรมีกำลังส่ง ประมาณ 100 วัตต์ (watt) ระหว่างสายอากาศกับเครื่องส่งจะมีการติดตั้งส่วนป้องกันการรบกวนของคลื่น เพื่อหลีกเลี่ยงปัญหาการเกิดการผิดเพี้ยนของสัญญาณแบบอินเตอร์มอดูเลต (intermodulate) ขึ้นในกรณี สถานีส่งใกล้เคียงทำการส่งความถี่ในย่านความถี่ใกล้กัน รูปที่ 1.5 เป็นตัวอย่างแผนผังของ เครื่องส่งรุ่น PSL 600 ซึ่งนิยมใช้กันมาก



รูปที่ 1.5 แผนผังของเครื่องรุ่น PSL600 ซึ่งมอดูเลชั่น (modulation) แบบเอฟเอสเค (FSK)

เนื่องจากสายอากาศของเครื่องลูกข่ายมีลักษณะเป็นแกนเฟอร์ไรต์ (ferrite) พันด้วยคอยล์ (coil) ขนาดเล็กบรรจุอยู่ภายในเครื่อง ขดลวดเหล่านี้มีค่าประกอบการสูญเสียถึง 16 เดซิเบล เมื่อเทียบกับสายอากาศแบบไดโพล (dipole) ทั่วไป อีกทั้งผลจากการลดทอนสัญญาณที่เกิดจากอาคารสูงและร่างกายมนุษย์ทำให้ประสิทธิภาพในการรับสัญญาณของเครื่องลูกข่ายอยู่ในระดับต่ำ แนวทางในการเพิ่มประสิทธิภาพของการรับสัญญาณทำได้โดยการเพิ่มความแรงของสัญญาณที่ส่งออก จากสถานีส่งออกจากสถานีส่งให้อยู่ในระดับเพียงพอที่เครื่องลูกข่ายจะสามารถทำงานได้อย่างถูกต้อง ในทางปฏิบัติทำโดยการเพิ่มจำนวนสถานีส่งให้ส่งสัญญาณไปยังพื้นที่ให้บริการ

พร้อมๆกัน ดังในรูปที่ 1.6 โดยข่าวสารจากทั้งสองสถานีเป็นข่าวสารเดียวกันและใช้ความถี่เดียวกัน แต่ถ้าระยะทางจากศูนย์ควบคุมของสถานีส่งทั้งสองมีค่าต่างกันจะทำให้ข่าวสารที่ส่งมายังสถานีส่งทั้งสองมาไม่ตรงเวลากัน ดังนั้นสัญญาณที่ถูกส่งออกสถานีส่งทั้งสองย่อมมีเฟสต่างกัน บางครั้งเมื่อสัญญาณมาถึงเครื่องรับอาจเกิดการลดทอนมากกว่ากรณีใช้สถานีส่งสถานีเดียว

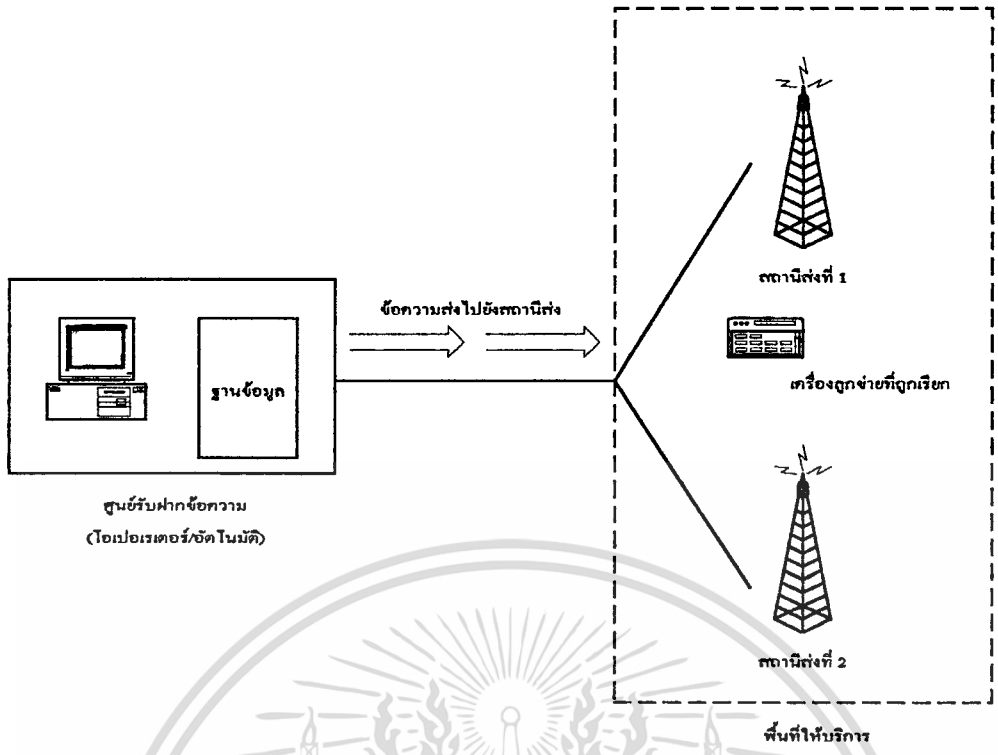
ทางแก้ไขปัญหาดังกล่าวทำได้โดยการใช้เทคนิคการส่งสัญญาณแบบกึ่งซิงโครไนซ์(Quasi syncronouse) โดยหากสถานีใดอยู่ห่างจากศูนย์ควบคุมมากกว่า จะมีการติดตั้งวงจรหน่วงเวลาในวงจรสื่อสารของสถานีส่งที่อยู่ใกล้กว่าเพื่อชดเชยเวลาหน่วงของข่าวสารที่ไปยัง สถานีทั้งสองให้เท่ากัน ทำให้สัญญาณที่ส่งออกจากสถานีทั้งสองมีเฟส (phase) ตรงกันตลอดเวลาโดย ไม่ขึ้นกับตำแหน่งของสถานีส่งทั้งสอง เนื่องจากรูปแบบการเชื่อมต่อระหว่างศูนย์ควบคุมสถานีส่งอาจใช้วงจรหน่วงเชื่อมต่อทั้งแบบคู่สายเช่าหรือวงจรไมโครเวฟ (microwave) ซึ่งทั้งสองรูปแบบ ก็สามารถติดตั้งวงจรหน่วงเวลาได้ทั้งสิ้นดังแสดงในรูปที่ 1.7

เมื่อเครื่องลูกข่ายได้รับสัญญาณจากสถานีส่งทั้งสอง หากเป็นกรณีอุดมคติสัญญาณทั้งสองจะมาถึงเครื่องรับโดยมีเฟสเดียวกัน อย่างไรก็ตามการชดเชยเวลาหน่วงที่ได้กล่าวมาเป็น เพียงการชดเชยในส่วนของวงจรเชื่อมต่อในเครือข่ายเท่านั้น มิได้เป็นการชดเชยเวลาหน่วงที่ เกิดจากการแพร่กระจายของคลื่นวิทยุจากสถานีส่งแต่อย่างใด ในพื้นที่ให้บริการที่มีอาคารสูง อยู่เป็นจำนวนมาก สัญญาณจากสถานีส่งที่เป็นคลื่นความถี่ย่าน VHF และ UHF จะมีการสะท้อนกับอาคารเหล่านั้น สัญญาณที่มาถึงเครื่องลูกข่ายทั้งจากสถานีส่งโดยตรงและที่เกิดจากการสะท้อนอาจก่อให้เกิดสัญญาณรบกวนขึ้นและในกรณีสัญญาณทั้งสองมีเฟสต่างกัน 180 องศาจะทำให้เกิดการหักล้างของสัญญาณเป็นศูนย์ได้ แต่อย่างไรก็ตามมีการทดสอบจะพบว่าเทคนิคแบบกึ่งซิงโครนัสนั้นจะช่วยป้องกันไม่ให้เกิดการหักล้างของสัญญาณจนเป็นศูนย์ที่เครื่องลูกข่าย เนื่องจากข้อมูลที่ถูกส่งไปยังเครื่องลูกข่ายโดยส่วนมากเป็นข้อมูลดิจิตอล (digital) ซึ่งมีความต้านทานต่อรูปแบบการแทรกสอดของสัญญาณจากสถานีส่งโดยตรง และสัญญาณสะท้อนค่อนข้างสูง

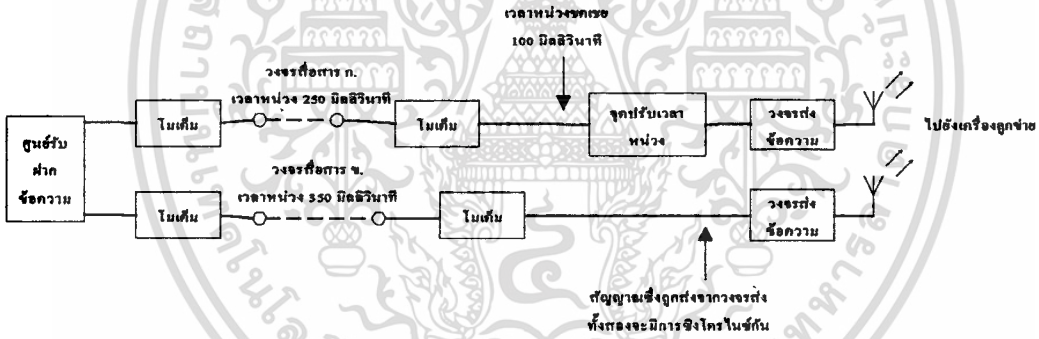
หากสถานีส่งส่งสัญญาณความถี่ 1 กิโลเฮิรตซ์ ซึ่งมีคาบเวลา 1/1000 =1 มิลลิวินาทีสมมติว่าสัญญาณที่มาจากสถานีส่งโดยตรงและสัญญาณที่สะท้อนมายังเครื่องลูกข่ายมีช่วงเวลาห่างกัน 1 มิลลิวินาที จะพบว่าสัญญาณทั้งสองยังคงมีเฟสเดียวกัน แต่หากสัญญาณที่ใช้ส่งมีความถี่ 500 เฮิรตซ์ หรือมีคาบเวลาเป็น 1/500 เท่ากับ 2 มิลลิวินาที และช่วงเวลาห่างของสัญญาณทั้งสองทิศทางยังคงมีเท่าเดิม ในกรณีนี้พบว่าสัญญาณทั้งสองจะกลับเฟสกันพอดี ทำให้สัญญาณที่เครื่องรับได้เป็นศูนย์

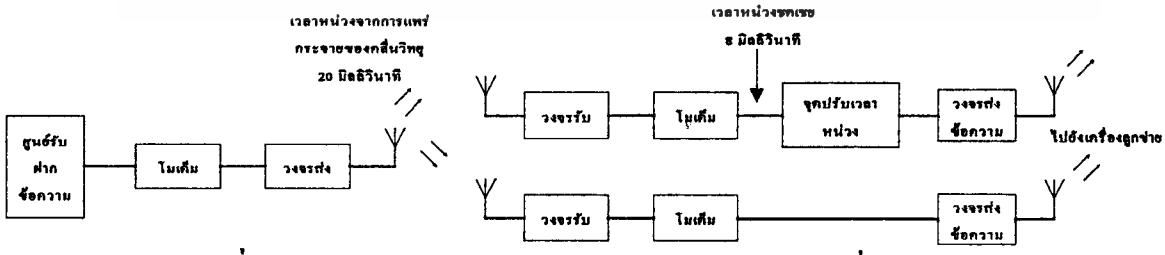
มีรายงานการทดลองยืนยันว่าค่าเวลาหน่วงสูงสุดที่ยอมให้เกิดขึ้นได้กับสัญญาณสียงพูด มีค่าไม่เกิน 10 เปอร์เซ็นต์ (percent) ของคาบเวลาของความถี่สูงสุด ในกรณีย่านความถี่เสียงพูดผ่านโทรศัพท์ซึ่งมีค่าความถี่จำกัดสูงสุด 3,400 เฮิรตซ์ มีคาบเวลาเท่ากับ 1/3,400 หรือ ประมาณ 300 ไมโครวินาที ดังนั้นค่าเวลาหน่วงที่ยอมให้ได้จะมีค่าไม่เกิน 30 ไมโครวินาที

สำหรับข้อมูลดิจิตอลจะเกิดช่วงเวลาที่เรียกว่า ช่วงสับสนซึ่งเป็นช่วงที่มีการเปลี่ยนระดับ ของข้อมูลจาก 0 เป็น 1 ช่วงเวลาหน่วงที่ยอมให้ได้จะมีค่าไม่เกิน 25 เปอร์เซ็นต์ ของคาบเวลาต่อบิต



รูปที่ 1.6 การเพิ่มประสิทธิภาพในการรับสัญญาณของเครื่องลูกข่ายโดยใช้สถานีส่ง 2 สถานีส่งข้อมูลเดียวกัน





รูปที่ 1.7 การชดเชยเวลาหน่วงของเทคนิคการส่งสัญญาณแบบกึ่งซิงโครนัส

บทที่ 2
ทฤษฎีและหลักการ

2.1 การส่งผ่านข้อมูลอนาลอกและดิจิตอล

ในการส่งผ่านข้อมูลจากต้นทางไปยังปลายทาง มีคุณสมบัติธรรมชาติของสัญญาณข้อมูลอย่างหนึ่งที่น่าสนใจก็คือ การเคลื่อนที่ของข้อมูลและกระบวนการจัดการกับข้อมูลเพื่อให้มั่นใจได้ว่าข้อมูลที่ส่งไปและรับมาสามารถจะเข้าใจกันได้ สำหรับการศึกษาทั้งหมดนี้มีจุดสำคัญที่เราจะเข้าไปเกี่ยวข้องด้วยก็คือ ปริมาณชนิด อนาลอก (analog) และชนิดดิจิตอล

ปริมาณอนาลอกและดิจิตอล เป็นปริมาณที่สอดคล้องกับปริมาณที่มีค่าแบบต่อเนื่อง และไม่ต่อเนื่องตามลำดับ ศัพท์ 2 คำนี้จะถูกนำมาใช้ในเรื่องสามเรื่องดังต่อไปนี้คือ

2.1.1 ข้อมูล (Data)

ข้อมูลแบบอนาลอกนั้นเป็นข้อมูลที่เกิดขึ้นแล้วมีค่าที่ต่อเนื่องในช่วงเวลาที่พิจารณา เป็นต้นว่าเสียง หรือภาพ จะเป็นข้อมูลที่มีการแปรเปลี่ยนรูปแบบของความเข้มอย่างต่อเนื่อง อุณหภูมิ และความดันก็เป็นปริมาณที่ต่อเนื่องที่จัดได้ว่า ให้ข้อมูลแบบอนาลอก แต่สำหรับข้อมูลดิจิตอลเป็นข้อมูลที่มีค่าที่เฉพาะที่กระโดด เป็นค่าที่ไม่ต่อเนื่อง เช่น ค่าเลขจำนวนเต็ม เป็นต้น

ตัวอย่างของข้อมูลดิจิตอลอย่างหนึ่งได้แก่ตัวหนังสือหรือตัวอักษร ซึ่งประสาทสัมผัสของคนสามารถรับรู้และเข้าใจได้ แต่มันไม่สามารถจะสื่อความหมายนั้นโดยตรงให้ระบบสื่อสารต่างๆ รับรู้เข้าใจ เหมือนกับที่มนุษย์เราเข้าใจได้

ระบบในการสื่อสารบางระบบจะถูกออกแบบไว้สำหรับข้อมูลเลขฐานสอง ซึ่งจำนวนของรหัสที่ใช้กันอย่างแพร่หลายที่สุดชนิดหนึ่งคือ รหัส ASCII (American Standard Code for Information Interchanges) ตัวอักขระแต่ละตัวของรหัสนี้แทนได้ด้วยเลขฐานสอง 7 หลัก ซึ่งเลขฐานสอง 7 หลัก แทนอักขระที่ไม่ซ้ำกันได้ทั้งหมดถึง 128 ตัว รหัสเลขฐานสองบางตัวจึงถูกนำมาแทนตัวอักขระสำหรับการควบคุมและตัวอักขระของการควบคุมส่วนหนึ่งก็ใช้ในการควบคุมการพิมพ์ และรหัสเลขฐานสองที่แทนตัวอักขระส่วนที่เหลือบางตัวก็จะใช้ในระบบสื่อสาร โดยทั่วไปแล้วการเข้ารหัสในการเก็บและส่งผ่านข้อมูล จะใช้จำนวนเลขไบนารี (binary) 8 บิท ต่อ 1 ตัวอักษร โดยบิทที่ 8 ก็คือพาริตี้บิท (Parity Bit) ที่ใช้สำหรับการตรวจสอบความผิดพลาด

2.1.2 สัญญาณ (Signal)

ในระบบของการสื่อสารข้อมูลจะเคลื่อนที่จากจุดหนึ่งไปยังอีกจุดหนึ่งในรูปแบบของสัญญาณไฟฟ้า ซึ่งสัญญาณอนาลอกก็คือ คลื่นแม่เหล็กไฟฟ้าที่มีการเปลี่ยนแปลงอย่างต่อเนื่อง และสัญญาณดังกล่าวนี้เองที่จะถูกส่งผ่านเข้าไปในตัวกลางชนิดต่างๆ ทั้งนี้ก็ขึ้นอยู่กับสเปคตรัม (spectrum) ของสัญญาณ ตัวกลางอาจเป็นสาย เช่น สายคู่บิดเกลียว สายโคแอคเชียล (coaxial) สายใยนำแสง หรืออาจจะเป็นตัวกลางแบบไร้สาย เช่น ชั้นบรรยากาศหรือสุญญากาศ ส่วนสัญญาณดิจิตอลนั้นอาจเป็นขบวนของพัลซ์โวลเตจ (pulse voltage) ที่ใช้ระดับของพัลส์ที่ส่งไปเป็นตัวแทนข้อมูล เช่น ใช้ระดับโวลเตจคงที่ค่าบวกจะแทนค่าไบนารี 1 และระดับโวลเตจลบแทนค่าไบนารี 0

ข้อมูลดิจิตอลไบนารีสัญญาณที่ใช้แทนข้อมูลจะใช้ศักย์ไฟฟ้ากระแสตรง 2 ระดับโดยที่ระดับโวลเตจค่าหนึ่งแทนเลขฐานสองค่า 1 และอีกค่าหนึ่งแทนเลขฐานสองค่า 0

2.1.3 ข้อมูลและสัญญาณ (Data and Signal)

จากที่ได้กล่าวมาแล้ว เรากำลังสนใจสัญญาณที่ใช้แทนข้อมูลอนาลอก และข้อมูลดิจิตอลในสถานการณ์โดยทั่วไปแล้วข้อมูลอนาลอกจะเป็นฟังก์ชั่น (function) กับเวลา และมีค่าสเปคตรัมที่จำกัดอยู่ค่าหนึ่ง ซึ่งเราสามารถแทนข้อมูลอนาลอกดังกล่าวได้ด้วยสัญญาณคลื่นแม่เหล็กไฟฟ้าที่มีค่าสเปคตรัมค่าเดียวกันและข้อมูลดิจิตอลนั้นสามารถแทนได้ด้วยสัญญาณดิจิตอลที่มีระดับโวลเตจแตกต่างกันสองระดับเพื่อแทนค่าเลขฐานสอง 0 และ 1

2.1.4 การส่งผ่าน (Transmission)

ความแตกต่างระหว่างสัญญาณดิจิตอลและสัญญาณอนาลอกนั้น ควรพิจารณาให้ชัดเจนขึ้นอีกทั้งสัญญาณอนาลอกและสัญญาณดิจิตอลนั้นจะถูกนำส่งผ่านตัวกลางที่เหมาะสม และขบวนการในการดัดแปลงสัญญาณ เพื่อให้เหมาะสมกับตัวกลางการส่งผ่านก็เป็นหน้าที่ของระบบการส่งผ่าน(Transmission system)การส่งผ่านในรูปแบบของอนาลอกเป็นวิธีการส่งผ่านสัญญาณอนาลอกไปโดยไม่มีการเปลี่ยนแปลงสาระข้อมูล ไม่ว่าจะเป็นกรณีใดๆ สัญญาณอนาลอกจะเกิดการลดทอนหลังจากที่เดินทางผ่านเข้าไปในตัวกลาง ดังนั้นเพื่อให้สัญญาณสามารถเดินทางไปถึงปลายทางได้ระยะทางไกลๆ ระบบของการส่งผ่านแบบอนาลอกก็จะมีตัวเพิ่มความแรงของสัญญาณ เพื่อเพิ่มพลังงานให้แก่สัญญาณ แต่ยังมีปัญหาเพราะว่าตัวเพิ่มความแรงของสัญญาณนอกจากจะเพิ่มพลังงานให้กับสัญญาณแล้ว ยังเพิ่มพลังงานให้กับสัญญาณรบกวนอีกด้วย และยิ่งมีการต่อใช้ตัวขยายหลายๆตัวแบบอนุกรม เพื่อให้ได้ระบบในการส่งสัญญาณได้ไกลๆยิ่งทำให้สัญญาณผิดเพี้ยนไปยิ่งขึ้น

สำหรับข้อมูลแบบอนาลอกเช่น เสียง ความผิดเพี้ยนเพียงส่วนเล็กน้อย สามารถยอมให้เกิดขึ้นได้เพราะว่าข้อมูลยังสามารถเข้าใจได้ แต่สำหรับข้อมูลดิจิตอลการต่อตัวขยายแบบอนุกรมจะทำให้เกิดความผิดพลาด

2.1.5 การส่งผ่านข้อมูลดิจิตอล (Digital Transmission)

การส่งผ่านข้อมูลด้วยวิธีการนี้จะเกี่ยวข้องกับเนื้อหาของสัญญาณ สัญญาณดิจิตอลจะถูกส่งไปได้ในระยะทางที่จำกัด ก่อนที่การลดทอนจะทำอันตรายต่อองค์ประกอบของข้อมูล ดังนั้นเพื่อให้การส่งข้อมูลสามารถทำได้เป็นระยะทางที่ไกลๆ เราจึงใช้ตัวทวนสัญญาณ (Repeater) เพื่อกู้สัญญาณดั้งเดิมกลับคืนมา โดยที่ตัวทวนสัญญาณเมื่อได้รับสัญญาณดิจิตอลมาแล้ว ก็จะทำการกู้รูปแบบของ 1 และ 0 กลับคืนมาอีกครั้งและส่งต่อออกไปใหม่ ซึ่งทำให้สามารถเอาชนะการลดทอนลงไปได้

ด้วยเทคนิคอย่างเดียวกันกับที่ได้กล่าวมาแล้ว อาจจะนำมาใช้ได้กับสัญญาณอนาลอก ที่ใช้เป็นตัวส่งถ่ายข้อมูลดิจิตอล ณ ตำแหน่งในพื้นที่ที่เหมาะสม ระบบของการส่งผ่านก็จะใช้เครื่องทวนสัญญาณแทนที่จะเป็นตัวขยายสัญญาณ ตัวทวนสัญญาณจะกู้สัญญาณดิจิตอลกลับคืนมาจากสัญญาณอนาลอกและสร้างสัญญาณอนาลอกขึ้นมาใหม่ ทำให้ไม่เกิดการสะสมของสัญญาณรบกวนต่อๆไป

ในปัจจุบันมีแนวโน้มที่จะหันมาใช้การสื่อสารระบบดิจิตอลที่ได้แทนระบบอนาลอก แม้ว่าจะไม่ได้มีการลงทุนใช้ระบบอนาลอกมาก่อนอย่างมากก็ตาม เหตุผลที่สำคัญก็คือ

– ดิจิตอลเทคโนโลยี การพัฒนาเทคโนโลยีของวงจรดิจิตอล LSI และ VLSI ทำให้ราคาและขนาดของวงจรลดลง ในขณะที่เครื่องมือทางอนาลอกไม่ได้ลดลง

– คุณภาพของข้อมูล สำหรับขบวนการทางดิจิตอล การใช้ตัวทวนสัญญาณแทนที่จะใช้ตัวขยายสัญญาณ ทำให้อิทธิพลของสัญญาณรบกวนไม่ถูกสะสม ทำให้เราสามารถส่งข้อมูลไปได้ระยะทางไกลๆ แม้ว่าคุณภาพของสายจะไม่ดีก็ตาม

– ความจุของการใช้งานมีมาก มันเป็นเรื่องที่สิ้นเปลืองมากในการที่เราจะต้องสร้างทางเดินการส่งผ่านข้อมูลที่มีแบนด์วิดท์กว้างมากๆ เช่น ช่องสัญญาณดาวเทียม และเส้นใยนำแสง ดังนั้นการนำเอาขบวนการในการมัลติเพลกซ์ (multiplex) ทางด้านเวลาดูจะเป็นวิธีการที่ง่าย และราคาถูกกว่าการมัลติเพลกซ์ทางด้านความถี่

– ความปลอดภัยและความเป็นส่วนตัว เทคนิคการย่อข้อมูลพร้อมที่จะนำเข้ามาใช้กับข้อมูลดิจิตอล และพร้อมที่จะนำมาใช้กับข้อมูลอนาลอกที่ถูกดิจิไตซ์ (digitize) แล้ว

– การรวมกันเข้าเป็นหนึ่งเดียวกัน ด้วยขบวนการทางข้อมูลของอนาลอกและดิจิตอล สัญญาณจะมีรูปแบบที่เหมือนกัน และสามารถดำเนินการได้ในลักษณะเดียวกัน ซึ่งมันก็จะทำให้ประหยัดและสะดวกในการรวมเสียง ภาพและข้อมูลดิจิตอลเข้าไว้ด้วยกัน

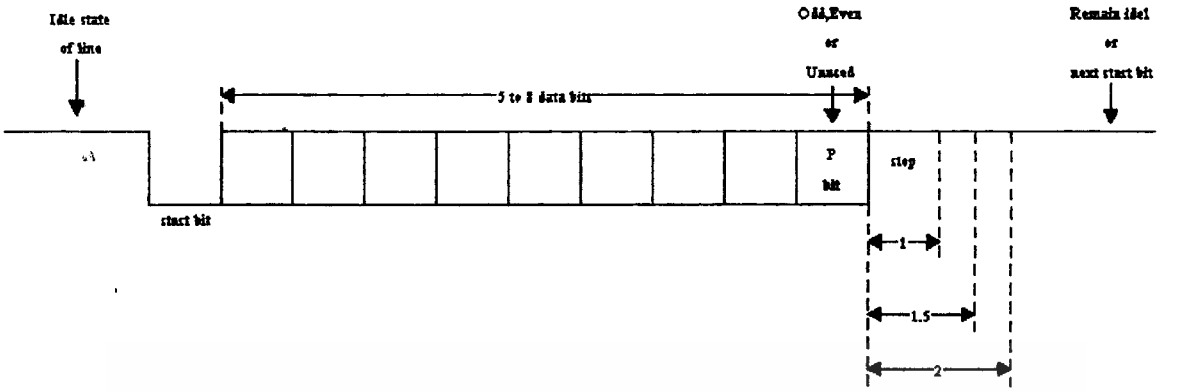
2.2 รูปแบบการสื่อสารข้อมูล

2.2.1 การส่งผ่านแบบอะซิงโครนัส (Asyncronous Transmission)

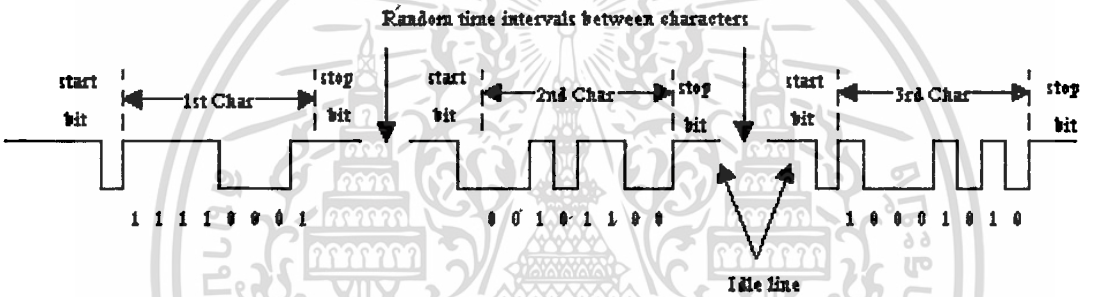
วิธีการแก้ปัญหาการซิงโครไนเซชั่นวิธีหนึ่งก็คือ การส่งบล็อกที่ประกอบด้วยบิทจำนวนน้อยๆ ในแต่ละบล็อก (block) และทำการซิงโครนัสใหม่ทุกครั้ง ที่ตำแหน่งเริ่มต้นของแต่ละบล็อก และเทคนิคเก่าอันหนึ่งที่เป็นที่รู้จักกันก็ได้แก่ start - stop หรือการส่งผ่านแบบอะซิงโครนัส (asynchronous) การส่งผ่านแบบอะซิงโครนัส นับว่าเป็นการส่งแบบที่เรียกว่า character - oriented โดยจำนวนบิทใน 1 อักขระจะถูกกำหนดให้มีได้ตั้งแต่ 5 ถึง 8 บิท

เทคนิคอันนี้สามารถที่จะอธิบายได้ง่ายๆ โดยอ้างอิงถึงรูปที่ 2.1 คือ เมื่อไม่มีการส่งอักขระสายระหว่างผู้รับและผู้ส่งจะอยู่ในสถานะ "idle" ซึ่งตามมาตรฐานจะถูกกำหนดให้มีสถานะเป็นมาร์ค (mark) (1) ดังนั้นสำหรับสัญญาณNRZ - L idle ก็จะถูกแทนด้วยการมีค่าศักย์ไฟฟ้า (หรือกระแส) ในสายที่ตำแหน่งเริ่มต้นของอักขระที่เรียกว่าบิทเริ่มต้นค่าของไบนารีจะมีค่าเป็น 0 ซึ่งต่อไปก็จะตามด้วยกลุ่มบิทของอักขระตั้งแต่ 5-8 บิท ในบางกรณีก็จะมีบิทพาริตี้ด้วย สำหรับบิทพาริตี้จะถูกกำหนดขึ้นโดยผู้ส่ง เพื่อให้ผู้รับใช้สำหรับตรวจจับความผิดพลาด ดังจะได้อธิบายกันต่อไป บิทสุดท้ายของอักขระจะถูกตามด้วยบิทสตอป ซึ่งมีค่าไบนารีเป็น "1" ความยาวต่ำสุดของบิทสิ้นสุดถูกกำหนดให้เป็น 1 บิท โดยทั่วไปแล้วก็จะกำหนดให้เป็น 1, 1.5, 2 เท่าของบิท สำหรับค่านานที่สุดจะไม่ถูกกำหนดเพราะว่าบิทสิ้นสุดต่อเนื่องจนกระทั่งมันพร้อมที่จะส่งตัวอักขระถัดไป ถ้าขบวนของอักขระถูกส่งอย่างต่อเนื่องคงที่สม่ำเสมอ ช่องว่างระหว่างอักขระที่จะมีค่าเท่ากันตลอด ซึ่งจะเท่ากับบิทสิ้นสุด ยกตัวอย่าง ถ้าบิทสิ้นสุดมีความยาว 1 หน่วย และมีการส่งอักขระ ABC โดยอาศัยรหัส ASCII โดยไม่มีบิทพาริตี้ รูปแบบก็จะเป็นดังนี้

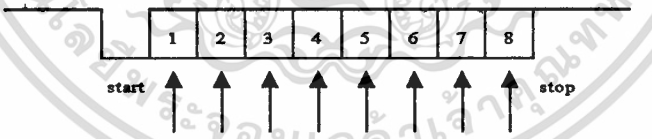
0100000110010000110110000011.....1111 เริ่มต้นด้วยบิทเริ่มต้นคือ (0) และตามด้วยลำดับเวลาของบิทที่เป็นอักขระ ASCII 7 บิท และตามด้วยบิทสิ้นสุดในสถานะไอเดิล (idle) ตัวรับจะมองดูการเปลี่ยนสถานะจาก 1 เป็น 0 เมื่อเริ่มต้นตัวอักขระ แล้วก็ทำการสุ่มสัญญาณอินพุทที่ตำแหน่งระหว่างบิทหนึ่งๆทั้งหมด 7 ครั้ง และก็จับดูการเปลี่ยนแปลงสถานะจาก 1 ไปเป็น 0 เพื่อทำการส่งอักขระตัวต่อไป



(a) Data character form



(b) 8-bit asynchronous bit stream



(c) Effect of timing error

รูปที่ 2.1 รูปแบบการส่งผ่านแบบอะซิงโครนัส

ความจำเป็นในเรื่องความถูกต้องทางเวลาสำหรับโครงสร้างนี้จะถูกกำหนดไว้พอประมาณ ตัวอย่าง เช่น อักขระแบบ ASCII ชนิด 8 bit รวมทั้งพารีตี้บิทด้วยนั้น ถ้าตัวรับช้าหรือเร็วกว่าตัวส่ง 5% บิทข่าวสาร ตัวที่ 8 จะถูกสุ่มผิดตำแหน่งไปจากกลางบิท 45% แต่ยังคงได้ค่าที่ถูกต้อง รูป 2.1c แสดงให้เห็นถึงอิทธิพล เนื่องจากความผิดพลาดทางเวลาที่มีขนาดพอสมควรที่จะทำให้เกิดความผิดพลาดในการรับ

2.2.2 การส่งผ่านแบบซิงโครนัส (Synchronous Transmission)

แม้ว่าชนิดของเฟรมจะเป็นตัวกำหนดความแตกต่างระหว่างการส่งข้อมูลแบบซิงโครนัสและอะซิงโครนัสก็ตาม แต่ความแตกต่างของหลักการพื้นฐานระหว่างวิธีทั้งสองก็คือว่า สัญญาณนาฬิกาที่ควบคุมการส่งผ่าน

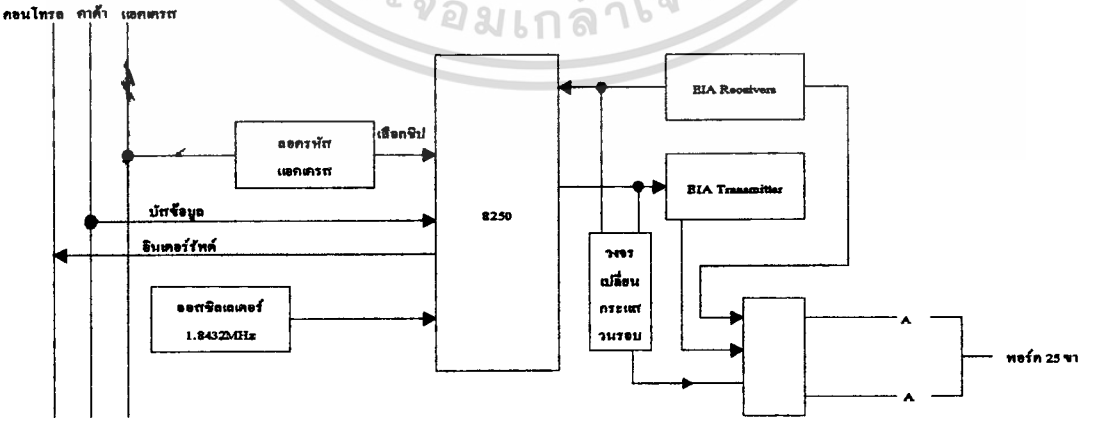
แบบอะซิงโครนัส ของตัวส่งและตัวรับจะไม่ซิงค์กันพอดีก็ได้ ในขณะที่การส่งผ่านของซิงโครนัส สัญญาณนาฬิกาของตัวส่งและรับจะต้องซิงค์กันพอดี ซึ่งอาจจะทำได้โดยการเพิ่มสายต่อระหว่างส่วนของอุปกรณ์ทั้งสองเพื่อนำพาสัญญาณนาฬิกาส่งไปด้วย เพื่อให้อุปกรณ์ทางด้านรับ สามารถที่จะตรวจหาได้อย่างแน่นอนว่าเมื่อใดบิทใหม่แต่ละบิทกำลังถูกส่งมา แต่อย่างไรก็ตามในทางปฏิบัติ วิธีการที่ทำกันปกติคือ ใช้สายข้อมูลเส้นเดียว แต่ฝากข้อมูลของสัญญาณนาฬิกามาร่วมกันกับรูปคลื่นที่ใช้ส่ง โดยวิธีการนี้สัญญาณนาฬิกาในการสุ่มของตัวรับจะต้องถูกแยกออกมาจากขบวนของสัญญาณข้อมูลที่เข้ามาด้วยวงจรแยกสัญญาณที่เหมาะสม ซึ่งวิธีการต่างๆ ของการแยกสัญญาณนาฬิกา เราจะได้กล่าวถึงกันต่อไป

มีวิธีการให้เลือกอยู่ 2 วิธีการในการจัดการกับการเชื่อมต่อข้อมูลแบบซิงโครนัส : ซึ่งได้แก่การจัดแบบ Character oriented (or byte) และการจัดแบบ bit oriented ข้อแตกต่างที่สำคัญของทั้งสองวิธีการได้แก่ วิธีการในการกำหนดการหาจุดเริ่มต้นและสิ้นสุดเฟรม (frame) ในระบบการจัดแบบ bit - oriented ตัวรับจะสามารถตรวจตัดสินการสิ้นสุดเฟรมที่บิทใดบิทหนึ่ง โดยไม่ต้องขึ้นกับการจำกัดวงอยู่กับขอบเขตของ 8 บิท ซึ่งวิธีการนี้หมายความว่าเฟรมอาจจะมีความยาว N บิท โดยที่ N เป็นจำนวนคงที่ใดๆ อย่างไรก็ตามในทางปฏิบัติ โครงสร้างนี้ไม่ค่อยได้ใช้ เพราะว่าในการใช้งานส่วนใหญ่ เราจะใช้เฟรมซึ่งมีความยาวเป็นจำนวนเท่าของ 8 บิท อย่างไรก็ดีระบบการส่งแบบ bit - oriented นั้นมีศักยภาพที่จะเพิ่ม throughput ขึ้นเป็นสองเท่าของระบบการจัดการที่ใช้แบบcharacter oriented ดังนั้น มันจึงเป็นโหมด (mode) ของการปฏิบัติการที่น่าพึงใจมากกว่า อย่างไรก็ตามเนื่องจากระบบ character - oriented ยังคงใช้กันแพร่หลายอยู่

2.3 โครงสร้างของพอร์ทสื่อสาร

พอร์ทสื่อสารของเครื่องไมโครคอนโทรลเลอร์ (Microcontroller) 16 บิตที่จะกล่าวถึงเป็นระบบ มารตรฐานตามแบบเครื่อง ไอบีเอ็ม (IBM) โครงสร้างบล็อกไดอะแกรม (block diagram) แสดงดังรูปที่ 2.2

พิจารณาได้ว่า 8250 เป็นตัวรับข้อมูลจากบัส (bus) ของระบบ ซึ่งก็คือสล็อต (slot) ขนาด 31 * 2 (62) ขาของระบบนั่นเอง ซีพียูติดต่อกับ 8250 ในลักษณะพอร์ท (port) อย่างเจาะจง ระบบไมโครคอมพิวเตอร์ (Microcomputer) 16 บิต คือ คอม 1 (COM_1) และ คอม 2 (COM_2) ทั้ง COM_1 และ COM_2 มีหมายเลขอินพุทเอาต์พุท ดังตารางที่ 2.1



รูปที่ 2.2 โครงสร้างพอร์ทสื่อสารข้อมูลที่ใช้ 8250

อินพุตเอาต์พุตพอร์ต		เลือกรีจิสเตอร์	สถานะ DLAB
พอร์ต COM1	พอร์ต COM2		
3F8	2F8	บัฟเฟอร์ IX	DLAB = 0 (เขียน)
3F8	2F8	บัฟเฟอร์ RX	DLAB = 0 (อ่าน)
3F8	2F8	แลตช์ตัวหาร	DLAB = 1
3F9	2F9	แลตช์ตัวหาร	DLAB = 1
3F9	2F9	อีนาเบิลอินเตอร์รัพต์	
3FA	2FA	กำหนดอินเตอร์รัพต์	
3FB	2FB	ควบคุมสายสื่อสาร	
3FC	2FC	ควบคุมโมเด็ม	
3FD	2FD	แสดงสถานะสายสื่อสาร	
3FE	2FE	แสดงสถานะโมเด็ม	

ตารางที่ 2.1 หมายเลขอินพุตเอาต์พุตพอร์ตของ COM_1 และ COM_2

การเลือกหมายเลขอินพุตเอาต์พุตพอร์ตแยกเป็นสองกลุ่ม กลุ่มหนึ่งคือ COM_1 จะกำหนดหมายเลขพอร์ตจาก 3F8 - 3FE อีกกลุ่มหนึ่งถ้ากำหนดหมายเลขพอร์ตเป็น 2F8 - 2FE ในการเลือกหมายเลขรีจิสเตอร์ (register) ภายในกำหนดด้วยแอดเดรส (address) 3 บิต คือ A_0, A_1 และ A_2 สำหรับการเลือก COM_1 และ COM_2 เราใช้แอดเดรส A_8 เป็นตัวเลือก การเลือกนี้กำหนดเป็นตารางได้ดังตารางที่ 2.2

แอดเดรส 3F8 - 3FF และ 2F8 - 2FE

A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	DLAB	รีจีสเตอร์
1	1/0	1	1	1	1	1	X	X	X		
							0	0	0	0	บัฟเฟอร์สำหรับตัวรับข้อมูล (อ่าน) โฮลดิงสำหรับตัวส่งข้อมูล (เขียน)
							0	0	1	0	อีนาเบิลอินเตอร์รัพต์
							0	1	0	X	กำหนดอินเตอร์รัพต์
							0	1	1	X	ควบคุมสายสื่อสาร
							1	0	0	X	แสดงสถานะสายสื่อสาร
							1	1	0	X	แสดงสถานะโมเด็ม
							1	1	1	X	ไม่ใช้
							0	0	0	1	แลตช์ตัวหาร (LSB)
							0	0	1	1	แลตช์ตัวหาร (MSB)

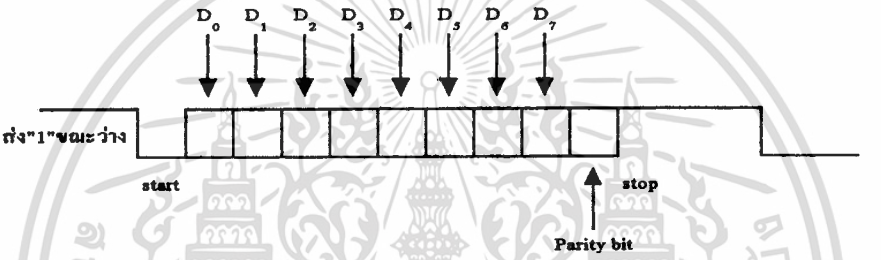
ตารางที่ 2.2 การกำหนดแอดเดรสสำหรับหมายเลขพอร์ตต่างๆ

2.3.1 การอินเตอร์รัพท์ (Interrupt)

โครงสร้างการควบคุมอินเตอร์รัพต์ของชิป 8259 ได้จัดลำดับการอินเตอร์รัพต์ไว้ 8 ระดับ สัญญาณรับอินเตอร์รัพต์ที่เข้าทางชิป 8259 มี 8 เส้น คือ IRQ_0 - IRQ_7 สำหรับกรณีของระบบสื่อสารอนุกรมได้ กำหนดสัญญาณการอินเตอร์รัพต์ไว้แล้วคือให้ IRQ_4 เป็นสัญญาณอินเตอร์รัพต์ของระบบ COM_1 และ IRQ_3 เป็นของ COM_2 ในการที่จะส่งสัญญาณอินเตอร์รัพต์ต้องให้บิต 3 ของรีจิสเตอร์ควบคุมโมเด็มได้รับการเซตค่าเป็น "1" ก่อน จากนั้นข้อมูลอินเตอร์รัพต์ที่อยู่ในรีจิสเตอร์อีนาเบิลอินเตอร์รัพต์จะเป็นตัวสั่งการอินเตอร์รัพต์

2.3.2 รูปแบบข้อมูลที่รับหรือส่ง

การสื่อสารข้อมูลของระบบนี้เป็นการสื่อสารแบบอะซิงโครนัส รูปแบบของข้อมูลจะมีสตาร์ท (start) บิต บิตตรวจสอบพาริตี้ และสต็อป (stop) บิต โครงสร้างของข้อมูลแต่ละเฟรมเป็นดังรูปที่ 2.3



รูปที่ 2.3 รูปแบบข้อมูล 1 เฟรม

ข้อมูลบิตแรกของการส่งเป็นสตาร์ทบิต ระบบจะส่งสตาร์ตบิตก่อนหลังจากนั้นจะตามด้วยข้อมูลและแทรกด้วยบิตตรวจสอบพาริตี้ตามด้วยสต็อปบิตขนาดของข้อมูลมีค่าได้ตั้งแต่ 5-8 บิต แต่ทั่วๆไปใช้ 8 บิตสต็อปบิตมีได้ 1, 1.5 หรือ 2 บิต ค่าเหล่านี้สามารถกำหนดลงไปในรีจิสเตอร์ควบคุมสายสื่อสาร

2.3.3 ชิป (chip) 8250

8250 เป็นไอซีขนาด 40 ขามีการทำงานเพื่อควบคุมสิ่งต่างๆ ที่เกี่ยวกับการสื่อสารแบบอนุกรมได้หมด โครงสร้างทางฮาร์ดแวร์ (hardware) ของอะแดปเตอร์การ์ด (adaptor card) นี้จึงไม่ยุ่งยากมากนัก

ขาสัญญาณอินพุต ชิป CS_0, CS_1, CS_2 (chip select) **คือขา 12 ขา 13 และ ขา 14 ตามลำดับ เป็นสัญญาณเลือกชิป โดยที่เมื่อต้องการเลือกชิปจะให้** CS_0, CS_1 **เป็น "1" และ** CS_2 **เป็น "0" สัญญาณเลือกชิปนี้จะได้รับการเลือกแลตช์ (latch) ไว้ในขณะที่สัญญาณ ADS มีค่าเป็น "0" การเลือกชิปนี้จะมีไว้เพื่อให้ซีพียูติดต่อกับ 8250**

ขาสโตรบข้อมูลอินพุต (data input strobe) **คือ DUSTR (ขา 22)** $\overline{\text{DISTR}}$ **(ขา 21) เมื่อสัญญาณที่ DISTR เป็น "1" และ** $\overline{\text{DISTR}}$ **เป็น "0" ในขณะที่มีการเลือกชิปเป็นขณะที่ซีพียูจะย้ายข้อมูลจากรีจิสเตอร์ภายในที่มีการกำหนดไว้แล้วมายังซีพียู สัญญาณนี้จึงเป็นสัญญาณอ่านข้อมูลหรือ read นั่นเอง**

ขาสโตรบข้อมูลเอาต์พุต คือ DOSTR (ขา 19) $\overline{\text{DOSTR}}$ **(ขา 18) เป็นสัญญาณที่แอกทีฟ** (active) **ขึ้น เพื่อให้ซีพียูเขียนข้อมูลลงมายังรีจิสเตอร์ของ 8250**

ขาสโตรบแอดเดรส (strobe address) คือ $\overline{ADS}$ (ขา 25) เมื่อมีค่าเป็น "0" จะแอกทีฟเพื่อแลตช์ค่า A_0-A_1 เลือกรีจิสเตอร์ภายใน การเลือกรีจิสเตอร์จะทำขณะที่การเลือกชิปแอกทีฟอยู่

ขาเลือกรีจิสเตอร์ คือ A_0 (ขา 26), A_1 (ขา 27), A_2 (ขา 28) ตามลำดับ เป็นตัวกำหนดแอดเดรสของรีจิสเตอร์ภายใน เพื่อให้ซีพียูทำการติดต่อกับ 8250 ตามค่ารีจิสเตอร์ที่กำหนดเพื่อการเขียนหรืออ่าน อย่างไรก็ตาม การทำงาน A_0-A_2 นี้ก็ขึ้นกับสัญญาณเลือกตัวหาร LAB-divisor latch access bit ซึ่งกำหนดค่ารีจิสเตอร์ได้ดังตารางที่ 2.3

ขารีเซต (master reset) คือ MR (ขา 35) เมื่อมีค่าเป็น "1" จะรีเซตการทำงานของชิป 8250 โดยทำให้ค่าต่างๆ ในรีจิสเตอร์ถูกเคลียร์ (clear) หมด (ยกเว้นบัฟเฟอร์ของตัวรับ ตัวส่งและตัวหาร) ขณะทำการรีเซตจะมีผลต่อสัญญาณเอาต์พุตด้วย ผลของการรีเซต แสดงไว้ในตารางที่ 2.3

DLAB	A2	A1	A0	รีจิสเตอร์
0	0	0	0	บัฟฟอร์สำหรับตัวรับข้อมูล (อ่าน) โฮลดิงสำหรับตัวส่งข้อมูล (เขียน)
0	0	0	1	อีนาเบิลอินเตอร์รัพต์
X	0	1	0	กำหนดอินเตอร์รัพต์
X	0	1	1	ควบคุมสายสื่อสาร
X	1	0	0	ควบคุมโมเด็ม
X	1	0	1	แสดงสถานะสายสื่อสาร
X	1	1	0	แสดงสถานะโมเด็ม
X	1	1	1	ไม่ใช้
1	0	0	0	แลตช์ตัวหาร (LSB)
1	0	0	1	แลตช์ตัวหาร (MSB)

ตารางที่ 2.3 การกำหนดค่ารีจิสเตอร์

ขาสัญญาณนาฬิกาตัวรับ (receiver clock) คือ RCLK (ขา 9) เป็นขาที่ตัวรับสัญญาณนาฬิกาเพื่อกำหนดอัตราบอด สัญญาณนาฬิกานี้จะมีค่าเป็น 16 เท่าของที่นำมาใช้

ขาเคลียร์ทูเซนด์ (clear to send) คือ $\overline{CTS}$ (ขา 36) เป็นสัญญาณที่ใช้ในการติดต่อกับโมเด็ม (modem) เงื่อนไขของสัญญาณนี้สามารถเก็บไว้ภายในชิป 8250 ที่จะให้ซีพียูอ่านไปตรวจสอบได้โดยเก็บไว้ที่บิต 4 ของรีจิสเตอร์ แสดงสถานะโมเด็ม ส่วนบิตของรีจิสเตอร์ แสดงสถานะจะเป็นตัวบอกว่า CTS ได้เปลี่ยนสถานะไปหลังจากการอ่านครั้งก่อนแล้วหรือไม่

ขาดาต้าเซตรีดี้ (data set ready) คือ $\overline{DSR}$ (ขา 37) เมื่อเป็น "0" จะแสดงว่าโมเด็มหรือข้อมูลได้รับการเซตเตรียมพร้อมแล้วสำหรับการเชื่อมต่อกับสายสื่อสาร และส่งข้อมูลระหว่าง 8250 กับโมเด็มสัญญาณ DSR เป็นสัญญาณอินพุตของ 8250 ที่ซีพียูสามารถอ่านไปดูได้ทางบิตที่ 5 ของรีจิสเตอร์แสดงสถานะ ส่วนบิต 1 ของรีจิสเตอร์แสดงสถานะเป็นตัวบอกว่า สัญญาณ DSR ได้เปลี่ยนสถานะไปหลังจากที่อ่านครั้งก่อนแล้วหรือไม่

หมายเหตุ ทั้ง $\overline{CTS}$ และ $\overline{DSR}$ เมื่อมีการเปลี่ยนสถานะ และถ้าได้รับการอินาเบลที่ modem status interrupt จะส่งผลในการสร้างสัญญาณอินเตอร์รัพต์

ขาตรวจสอบสายสื่อสาร (received line signal detect) คือ $\overline{RLSD}$ (ขา 38) ถ้าเป็น "0" หมายถึงแอกทีฟ คือ 8250 รับสัญญาณตรวจสอบสัญญาณพาหะจากโมเด็มว่า โมเด็มตรวจสอบได้แล้ว ซีพียูสามารถตรวจสอบสัญญาณนี้ทางบิต 1 ของรีจิสเตอร์แสดงสถานะ ส่วนบิต 3 จะเป็นบิตที่แสดงสถานะว่าสัญญาณนี้ได้รับการเปลี่ยนแปลงหลังจากอ่านไปแล้วหรือยัง

ขาแสดงวงจรเรียก (ring indicator) คือ RI (ขา 39) สัญญาณนี้แอกทีฟด้วยลอจิก (logic) "0" เป็นสัญญาณที่ส่งมาจากโมเด็ม โมเด็มตรวจสอบสัญญาณการเรียก (ringing) สัญญาณนี้ตรวจสอบได้ทางบิต 6 และดูสถานะการเปลี่ยนหลังจากอ่านแล้วจากบิต 2

ขารีเควสต์ทูเซน (request to sent) คือ $\overline{RTS}$ (ขา 32) เริ่มขานี้มีลอจิกเป็น "0" หมายความว่า 8250 พร้อมที่จะส่งข้อมูลแล้ว สัญญาณนี้จะได้รับการเซตให้แอกทีฟด้วย การโปรแกรมค่าลงไปในรีจิสเตอร์ควบคุมที่บิต 1

ขาเอาต์พุต 1 คือ $\overline{OUT1}$ (ขา 34) เป็นขาที่ผู้ใช้สามารถโปรแกรมให้แอกทีฟเป็น "0" ด้วยการโปรแกรมลงไปในบิต 2 ของรีจิสเตอร์ควบคุมโมเด็ม

ขาเอาต์พุต 2 คือ $\overline{OUT2}$ (ขา 31) เป็นขาที่ผู้ใช้สามารถโปรแกรมให้แอกทีฟเป็น "0" ด้วยการโปรแกรมค่าลงในรีจิสเตอร์ควบคุมโมเด็มทางบิต 3

ขาเลือกชิปเอาต์ (chip select out) คือ CSOUT (ขา 24) เมื่อมีค่าเป็น "1" จะบอกว่าชิปนี้ได้รับการเลือกชิปโดยซีพียูทางขา CS_0, CS_1 และ CS_2

ขาไดรด์เวอร์ดีสเอเบิล (driver disable) คือ DDIS (ขา 23) เป็นลอจิก "0" เมื่อซีพียูกำลังอ่านข้อมูลจาก 8250 สัญญาณ DDIS เป็น "1" มีไว้สำหรับการดิสเอเบิล (disable) การรับส่งภายนอกในกรณีที่ใช้ 8250 กับซีพียูผ่านทางบิต D_0 - D_7 เพื่อบอกเวลาว่าซีพียูและ 8250 ติดต่อกันอย่างไร

ขาสัญญาณกำหนดบอด คือ $\overline{BAUDOUT}$ (ขา 15) เป็นสัญญาณนาฬิกาที่มีความถี่เป็น 16 เท่าของสัญญาณนาฬิกา แล้วหารด้วยค่าที่โปรแกรมกำหนดในตัวหาร

ขาอินเตอร์รัพต์ คือ INTRPT (ขา 30)เป็น "1" เป็นการส่งสัญญาณอินเตอร์รัพต์ออกไปจาก 8250

ขาข้อมูลเอาต์พุต คือ SOUT (ขา 11) เป็นขาที่ใช้ส่งข้อมูลอนุกรมออกไปยังสายสื่อสาร

ขาอินพุทข้อมูลอนุกรม (serial input) คือ SIN (ขา 10) เป็นขารับข้อมูลอนุกรมจากสายส่งในการเชื่อมโยงการติดต่อสื่อสาร

ขาสัญญาณอินพุต / เอาต์พุต ข้อมูล D_6 - D_7 เป็นสัญญาณต่อเชื่อมกับบัสข้อมูลของระบบขาสัญญาณ $X'TAL_1$, $X'TAL_2$ คือขา 16, ขา 17 เป็นขาต่อกับคริสตอลเพื่อสร้างสัญญาณนาฬิกา

2.3.4 สถานะของ 8250 เมื่อเริ่มต้น

ก่อนการทำงานหรือโปรแกรมเราควรจะทราบว่าสถานะต่างๆ ของ 8250 เป็นอย่างไร โดยเฉพาะอย่างยิ่งเมื่อเริ่มเปิดเครื่อง จะมีสัญญาณรีเซต (reset) ซึ่งเป็นสัญญาณเดียวกับซีพียูมาทำการรีเซต 8250 ซึ่งขณะนั้นจะมีสถานะเป็นอย่างไร เอาต์พุตของวงจรจะให้สัญญาณอะไร จะทราบได้จากตารางที่ 2.4

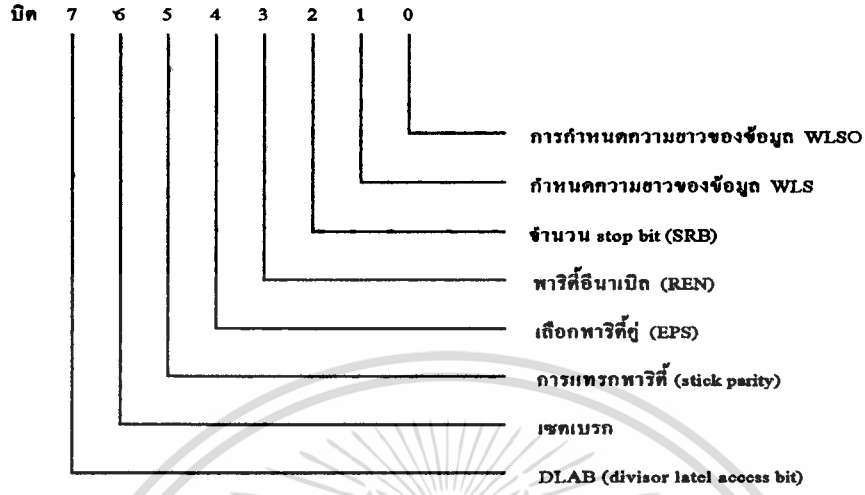
รีจิสเตอร์ / สถานะ	การควบคุม	สถานะเมื่อรีเซต
อินเตอร์รัพต์รีจิสเตอร์ภายใน	มาสเตอร์รีเซต	ทุกบิตเป็น 0 (0 – 3 ถูกกำหนด 4 – 7 จะเป็นถาวร)
รีจิสเตอร์กำหนดอินเตอร์รัพต์	มาสเตอร์รีเซต	บิต 0 เป็น 1 บิต 1 - 2 เป็น 0 บิต 3-7 เป็น 0 ถาวร
รีจิสเตอร์ควบคุมสายสื่อสาร	มาสเตอร์รีเซต	ทุกบิตเป็น 0
รีจิสเตอร์ควบคุมโมเด็ม	มาสเตอร์รีเซต	ทุกบิตเป็น 0
รีจิสเตอร์แสดงสถานะสายสื่อสาร	มาสเตอร์รีเซต	ยกเว้นบิต 5 และ 6 เป็น 1
รีจิสเตอร์แสดงสถานะโมเด็ม	มาสเตอร์รีเซต	บิต 0-3 เป็น 0 บิต 4-7 เป็นสัญญาณอินพุต
SOUT	มาสเตอร์รีเซต	เป็น 1
INTRPT (RCVR ERRORS)	Read LSR/ มาสเตอร์รีเซต	เป็น 0
INTRPT (RCVR Data Ready)	Read RBR/ มาสเตอร์รรีเซต	เป็น 0
INTRPT (RCVR Data Ready)	Read IIR/ มาสเตอร์รีเซต	เป็น 0
INTRPT (เปลี่ยนสถานะโมเด็ม)	Read MSR/ มาสเตอร์รีเซต	เป็น 0
OUT2	มาสเตอร์รีเซต	เป็น 1
RTS	มาสเตอร์รีเซต	เป็น 1
DTR	มาสเตอร์รีเซต	เป็น 1
OUT1	มาสเตอร์รีเซต	เป็น 1

ตารางที่ 2.4 ค่าเริ่มต้นและเอาต์พุตของ 8250

2.3.5 การใช้งานรีจิสเตอร์ต่างๆ บน 8250

รีจิสเตอร์ควบคุมสายสื่อสาร (line control register) **ในการควบคุมรูปแบบของข้อมูลแบบอะซิงโครนัสนั้น ผู้โปรแกรมจะต้องกำหนดค่าลงในรีจิสเตอร์ควบคุมสายสื่อสาร รีจิสเตอร์ตัวนี้มี 8 บิต โดยแต่ละบิตมีความหมายดังนี้**

พอร์ตแอดเดรสหมายเลข 3FB



รูปที่ 2.4 ค่าของบิตในรีจิสเตอร์ควบคุมสายการสื่อสาร

บิต 0 และ 1 เป็นตัวกำหนดความยาวของข้อมูลในการรับส่ง โดยที่

บิต 0	บิต 1	ความหมาย
0	0	หมายถึงข้อมูลขนาด 5 บิต
0	1	หมายถึงข้อมูลขนาด 6 บิต
1	0	หมายถึงข้อมูลขนาด 7 บิต
1	1	หมายถึงข้อมูลขนาด 8 บิต

บิต 2 เป็นบิตที่ใช้ในการกำหนดจำนวนสต๊อปบิต ถ้าเป็น "0" หมายถึงใช้สต๊อปบิต 1 บิต แต่ถ้าบิต 2 เป็น "1" ในกรณีส่งแบบ 5 บิตจะมีความยาวของสต๊อปบิตเป็น 1.5 บิตแต่ถ้าส่งแบบ 6,7 หรือ 8 บิต ความยาวของสต๊อปบิตจะเป็น 2

บิต 3 บิตนี้เป็นบิตแสดงการอีนาเบิลให้มีการตรวจสอบพาริตี้โดยถ้าบิตนี้มีค่าเป็น1 จะมีการเพิ่มพาริตี้บิต

บิต 4 มีค่าเป็น "0" และบิต 3 มีค่าเป็น "1" จะมีการกำหนดเป็นพาริตี้คี่ แต่ถ้าบิตนี้มีค่าเป็น "1" จะเป็นพาริตี้คู่

บิต 5 เมื่อบิต 3 มีค่าเป็น "1" และบิต 4 มีค่าเป็น "1" และบิต 5 มีค่าเป็น "1" จะมีการแทรกหรือตรวจสอบพาริตี้ (stick parity) ด้วยเงื่อนไขกำหนดให้เป็น "0" และถ้าบิต 4 มีค่าเป็น "0" บิต 3 มีค่าเป็น "1" และบิต 5 มีค่าเป็น "1" จะมีการกำหนดบิตพาริตี้เป็น "1"

บิต 6 เป็นบิตที่ควบคุมการเบรก เมื่อบิต 6 มีค่าเป็น "1" ส่วนของ SOUT จะได้รับการกำหนดให้เป็น "0" ตลอด

บิต 7 บิตนี้ทำหน้าที่เป็น DLAB บิตที่จะมีผลต่อการแลตช์ตัวหารดังที่กล่าวตารางที่ 2.1 และตารางที่ 2.2

2.3.6 การโปรแกรมอัตราบอด (boud rate generator) อัตราบอดได้รับการกำหนดเทียบกับสัญญาณนาฬิกา 1.8432 MHz และสามารถโปรแกรมตัวหารได้ตั้งแต่ 1-(2^{16}-1) ค่าความถี่เอาต์พุตของตัวกำหนดอัตราบอดมีค่าเท่ากับ 16* อัตราบอด ดังนั้น

ตัวหาร = ความถี่สัญญาณนาฬิกา / (อัตราบอด *16) การกำหนดอัตราบอดด้วยการกำหนดตัวหารนี้ ตัวหารจึงเป็นค่าที่กำหนดในรีจิสเตอร์ 2 ตัว ตัวหารนี้จะต้องถูกกำหนดค่าก่อนแล้ว โปรแกรมลงมาในรีจิสเตอร์การกำหนดต้องให้ DLAB = 1 แล้วให้ลดลงมาในรีจิสเตอร์ $3F_8$ ซึ่งเรียงกันเป็น LSB ของตัวหารส่วน $3F_9$ เมื่อ DLAB = 1 จะเป็นค่าของตัวหาร MSB ค่าของตัวหารเมื่อเทียบกับสัญญาณ 1.8432 MHz เป็นดังตารางที่ 2.5

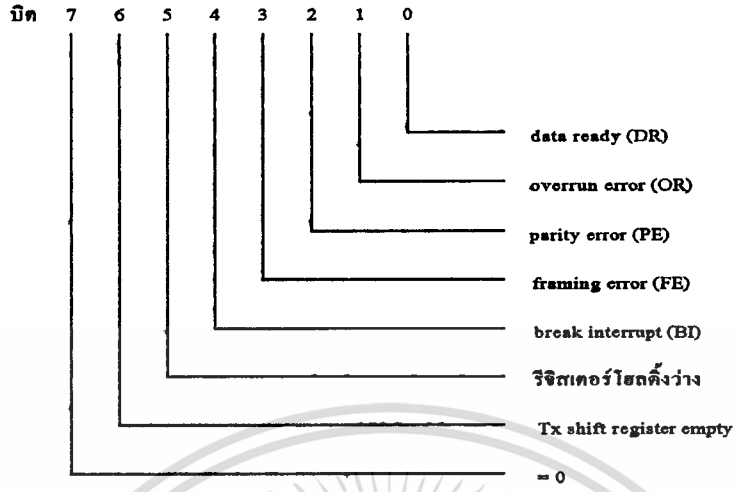
อัตราบอด	ตัวหาร		ค่าผิดพลาด
	ฐานสิบ	ฐานสิบหก	
50	2304	900	-
75	1536	600	-
110	1047	417	0.026
134.5	857	359	0.058
150	768	300	-
300	384	180	-
600	192	0C0	-
1200	96	060	-
1800	64	040	-
2000	58	03A	0.69
2400	48	030	-
3600	32	020	-
4800	24	018	-
7200	16	010	-
9600	12	00C	-

ตารางที่ 2.5 ค่าตัวหารสำหรับการกำหนดอัตราบอด

รีจิสเตอร์แสดงสถานะสายสื่อสาร (line status register) รีจิสเตอร์ตัวนี้เป็นรีจิสเตอร์ที่จะให้ข้อมูลแก่ซีพียูที่เกี่ยวกับการสื่อสารข้อมูลในการสื่อสารค่าของบิตต่างๆ ในรีจิสเตอร์นี้เป็นดังรูปที่ 2.5

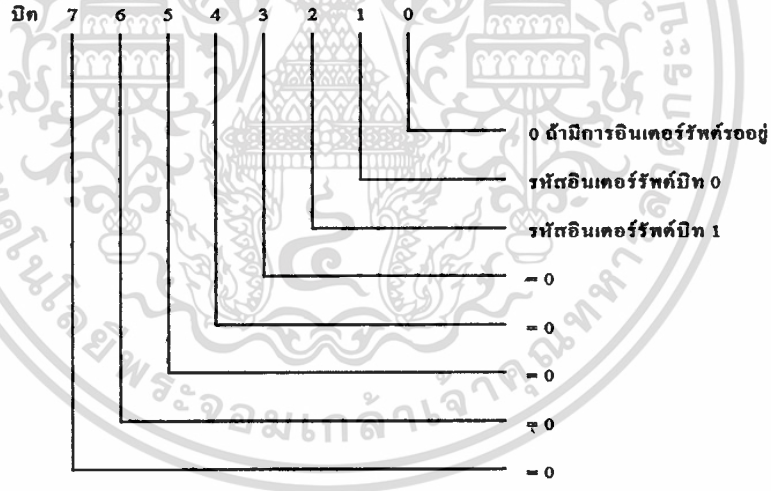
บิต 0 บิตนี้เป็นบิตที่บอกสถานะการรับข้อมูล ถ้าบิตนี้เป็น "1" แสดงว่าการรับข้อมูลเข้ามาในบัฟเฟอร์ได้ครบทุกบิตแล้ว บิตนี้จะได้รับการรีเซตให้เป็น "0" เมื่อซีพียูได้อ่านข้อมูลในบัฟเฟอร์ไปแล้ว หรือจะให้ซีพียูเขียนข้อมูลกลับมาชั่งรีจิสเตอร์นี้ก็ได้

แอดเดรส 3FD



รูปที่ 2.5 ค่าของบิตในรีจิสเตอร์แสดงสถานะสายสื่อสาร

แอดเดรส 3FA



รูปที่ 2.6 ค่าของบิตในรีจิสเตอร์กำหนดอินเตอร์รัพต์

บิต 1 บิตนี้ถ้ามีค่าเป็น "1" แสดงว่าเกิด overrun error (OR) **กล่าวคือ ขณะที่มีข้อมูลที่บัฟเฟอร์แต่ซีพียูยังไม่ได้อ่านไป ปรากฏว่ามีข้อมูลชุดใหม่มาเขียนทับ**

บิต 2 บิตนี้ถ้ามีค่าเป็น "1" แสดงว่าเกิด parity error (PE) **กล่าวคือ ถ้ามีการตรวจสอบบิตพาริตี้แล้วไม่เป็นไปตามที่กำหนดไว้ บิตนี้จะได้รับการรีเซตโดยซีพียู เมื่อซีพียูอ่านค่าจากรีจิสเตอร์นี้ไปแล้ว**

บิต 3 บิตนี้ถ้ามีเฟรมของข้อมูลไม่เป็นไปตามที่กำหนด เช่น ตรวจสอบจำนวนบิตโดยดูที่พาริตี้และสต๊อปบิตไม่เป็นไปตามที่กำหนด

บิต 4 บิตนี้เรียกว่า break interrupt (BI) บิตนี้จะได้รับการเซตให้มีค่าเป็น "1" ถ้าหากว่ารับข้อมูลอินพุตเป็น "0" เป็นเวลายาวนานกว่าเวิร์ด (word) ของการสื่อสาร

บิต 5 บิตนี้เป็นบิตที่บอกว่า 8250 พร้อมที่จะรับข้อมูลจากสายสื่อสาร บิตนี้จะได้รับการเซตให้มีค่าเป็น "1" บิตนี้ยังคงสร้างสัญญาณอินเตอร์รัพต์เพื่อส่งไปบอกซีพียูด้วย นอกจากนี้จะมีสถานะเซตเมื่อมีการส่งถ่ายข้อมูลจากโฮลดิงรีจิสเตอร์ไปยังชิปรีจิสเตอร์ เพื่อพร้อมที่จะส่ง

บิต 6 เป็นบิตที่จะบอกว่าชิปรีจิสเตอร์ว่างเปล่า บิตนี้จะได้รับการเซตให้มีค่าเป็น "1" เพื่อบอกว่าพร้อมส่งแล้ว

บิต 7 จะเป็น "0" ตลอด

รีจิสเตอร์กำหนดอินเตอร์รัพต์ (IIR - interrupt identification register) ไอซี 8250 มีขีดความสามารถในการส่งอินเตอร์รัพต์ภายในชิป เพื่อว่าให้การทำงานระหว่าง 8250 กับซีพียูเป็นไปอย่างมีประสิทธิภาพสูง และเพื่อให้ผู้เขียนซอฟต์แวร์สามารถเขียนซอฟแวร์ได้ง่ายและสั้นลงได้มาก 8250 กำหนดความสำคัญของอินเตอร์รัพต์ไว้ 4 ระดับคือ ระดับแรก สถานะการรับข้อมูลจากสายสื่อสาร ระดับที่สอง การพร้อมรับข้อมูล ระดับที่สาม ขณะรีจิสเตอร์โฮลดิง (holding) สำหรับส่งข่าว ระดับที่สี่ สัญญาณสถานะโมเด็ม

ในขณะที่มีความต้องการอินเตอร์รัพต์หลายระดับพร้อมกัน 8250 จะให้ระดับที่มีความสำคัญน้อยกว่ารอไว้ก่อน โดยเก็บสถานะการอินเตอร์รัพต์นี้ไว้ในรีจิสเตอร์กำหนดอินเตอร์รัพต์ความหมายของรีจิสเตอร์นี้มีดังนี้

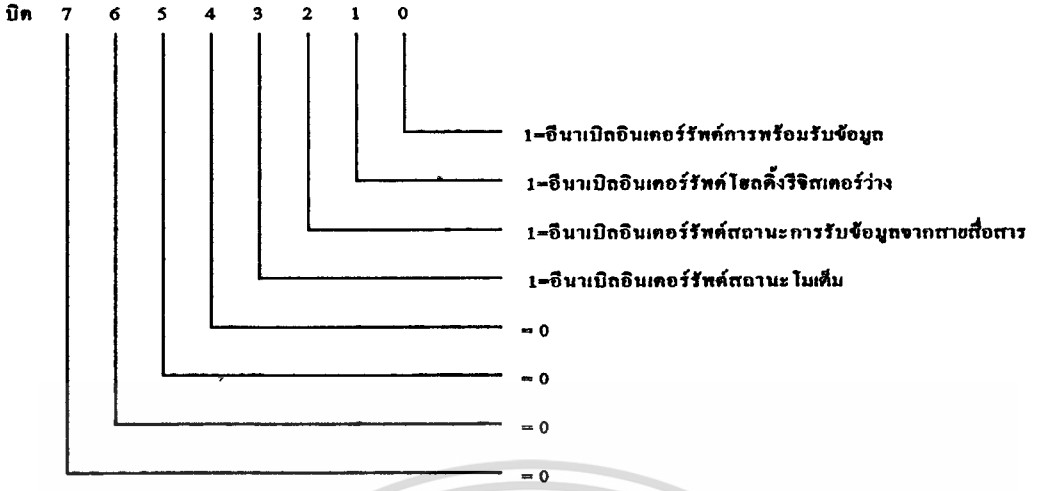
บิต0 เป็นบิตที่ใช้แสดงว่ามีอินเตอร์รัพต์เกิดขึ้นหรือไม่ ซึ่งสามารถให้ซีพียูตรวจสอบดูด้วยวิธีการ polling ได้ถ้าบิตนี้เป็น "1" หมายถึง ไม่มีอินเตอร์รัพต์เกิดขึ้น

บิต1-2 เป็นบิตที่แสดงความหมายบอกว่าการอินเตอร์รัพต์ที่เกิดขึ้นนั้นมาจากการอินเตอร์รัพต์ตามฟังก์ชันใด

บิต3-7 มีค่าเป็น "0"

รีจิสเตอร์อีนาเบิลอินเตอร์รัพต์ (INTRPT - interrupt enable register) ใน COM_1 เมื่อให้ DLAB = 0 พอร์ต $3F_9$ จะเป็นรีจิสเตอร์อีนาเบิลอินเตอร์รัพต์ ผู้ใช้สามารถกำหนดให้เกิดอินเตอร์รัพต์หรือไม่ก็ได้ โดยการกำหนดค่าลงในรีจิสเตอร์นี้ จากที่กล่าวแล้วว่าการอินเตอร์รัพต์ของ 8250 นี้มี 4 แบบ ดังนั้นจึงต้องกำหนดการอีนาเบิลได้ทั้ง 4 แบบ โดยการใช้ข้อมูลแต่ละบิตของรีจิสเตอร์นี้ เพื่อการกำหนดการอีนาเบิล (enable) ข้อมูลที่อยู่ในรีจิสเตอร์นี้มีความหมายดังนี้

พอร์ต 3F9 DLAB = 0



รูปที่ 2.7 ค่าของบิตในรีจิสเตอร์อีนาเบิลอินเตอร์รัพต์

บิต 2 1 0	ระดับความสำคัญ	ชนิดของอินเตอร์รัพต์	แหล่งเกิดอินเตอร์รัพต์	การรีเซตควบคุมอินเตอร์รัพต์
0 0 1	สูงสุด	ไม่เกิด	ไม่เกิด	อ่านข้อมูลจากรีจิสเตอร์
1 1 0		สถานะการรับข้อมูลจากสายสื่อสาร	overrun error parity error framming error break interrupt	สถานะสายสื่อสาร
1 0 0	ที่สอง	การพร้อมรับข้อมูล	มีข้อมูลที่ตัวรับ	การอ่านข้อมูลจากบัฟเฟอร์
0 1 0	ที่สาม	โฮลดิงรีจิสเตอร์สำหรับส่งข่าว	โฮลดิงรีจิสเตอร์สำหรับส่งข่าว	อ่านรีจิสเตอร์กำหนดอินเตอร์รัพต์ IIR หรือ เขียนลงไปยังโฮลดิงรีจิสเตอร์สำหรับส่ง
0 0 0	ที่สี่	สถานะโมเด็ม	CIS DSR RI ตรวจสอบสายส่งโดยตรง	อ่านรีจิสเตอร์แสดงสถานะของโมเด็ม

ตารางที่ 2.6 ฟังก์ชันการอินเตอร์รัพต์รหัสอินเตอร์รัพต์

บิต 0 บิตนี้ได้รับการเซตเป็น "1" เมื่อต้องการอีนาเบิลอินเตอร์รัพต์การพร้อมรับข้อมูล

บิต 1 บิตนี้ได้รับการเซตเป็น "1" เมื่อต้องการอีนาเบิลอินเตอร์รัพต์โฮลดิงรีจิสเตอร์ว่าง

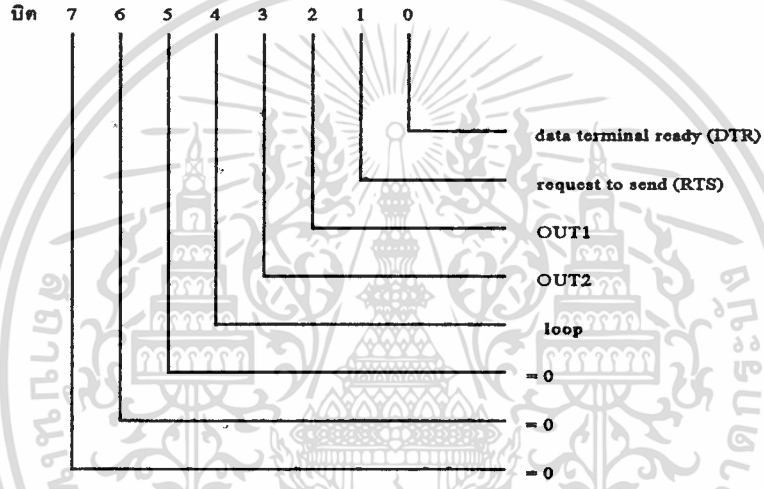
บิต 2 บิตนี้ได้รับการเซตเป็น "1" เมื่อต้องการอีนาเบิลอินเตอร์รัพต์จากสถานะการรับข้อมูลจากสายสื่อสาร

บิต 3 บิตนี้ได้รับการเซตเป็น "1" เมื่อต้องการอีนาเบิลอินเตอร์รัพต์จากสถานะโมเด็ม

บิต 4-7 ได้รับการกำหนดให้เป็น 0 เสมอ

รีจิสเตอร์ควบคุมโมเด็ม (modem control register) **รีจิสเตอร์ตัวนี้มีไว้ให้ซีพียูส่งผ่านข้อมูลมาเก็บเพื่อเป็นรหัสสำหรับควบคุมการทำงานของโมเด็ม การกำหนดพอร์ตของรีจิสเตอร์ตัวนี้คือ 3FC ข้อมูลต่างๆ ที่มีในรีจิสเตอร์ตัวนี้มีความหมายดังนี้**

พอร์ตหมายเลข 3FC



รูปที่ 2.8 ค่าของบิตในรีจิสเตอร์ควบคุมโมเด็ม

บิต 0 บิตนี้มีความหมายถึงการควบคุมสัญญาณ $\overline{\text{DTR}}$ เมื่อบิตนี้มีค่าเป็น "1" เอาต์พุตที่ DTR จะได้รับการกำหนดให้เป็น "0" และถ้าบิตนี้มีค่าเป็น "0" เอาต์พุตที่ $\overline{\text{DTR}}$ จะได้รับการกำหนดให้เป็น "1"

บิต 1 บิตนี้มีความหมายถึงสัญญาณ RTS ซึ่งจะมีผลเหมือนกับบิต 0 ในกรณีของ DTR

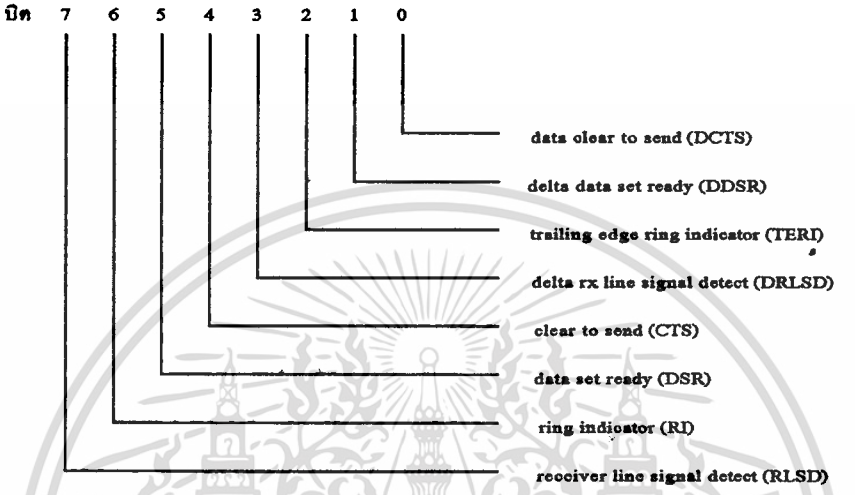
บิต 2 บิตนี้ใช้ควบคุมขาเอาต์พุต 1 (OUT1) ซึ่งจะมีผลเหมือนบิต 0

บิต 3 บิตนี้ใช้ควบคุมขาเอาต์พุต 2 (OUT2) ซึ่งจะมีผลเหมือนบิต 0

บิต 4 บิตนี้จะใช้สำหรับการกำหนดวงรอบสำหรับการตรวจสอบ 8250 เมื่อบิต 4 นี้ได้รับการเซตเป็น "1" สิ่งที่จะเกิดขึ้นเป็นดังนี้ ข้อมูลที่ SOUT จะได้รับการเซตให้เป็นลอจิก "1" ขาข้อมูลอินพุต SIN จะได้รับการแยกตัวออก ข้อมูลของเอาต์พุตชิปรีจิสเตอร์ จะได้รับการป้อนกลับมายังรีจิสเตอร์ข้อมูลอินพุต ส่วนสัญญาณ CTS, DSR, RLSD และ RI จะได้รับการแยกออกจากระบบแต่สัญญาณควบคุมโมเด็มคือ DTR, RTS, OUT1 และ OUT2 จะต่อเข้ากับสัญญาณทั้งสี่ที่เป็นอินพุต ดังนั้นจึงตรวจสอบระบบการทำงานได้

รีจิสเตอร์แสดงสถานะโมเด็ม รีจิสเตอร์ตัวนี้จะเป็นตัวที่รับสถานะจากโมเด็มมาเก็บไว้เพื่อให้ซีพียูสามารถอ่านตรวจสอบดูได้ สถานะของข้อมูลจะแอกทีฟเมื่อมีข้อมูลเป็น "1" และจะได้รับการรีเซตเมื่อซีพียูอ่านข้อมูลในรีจิสเตอร์นี้ไป พอร์ตที่ใช้กำหนดเป็นพอร์ตหมายเลข 3FE ข้อมูลภายในรีจิสเตอร์เป็นดังนี้

พอร์ตแอดเดรส



รูปที่ 2.9 ค่าของบิตในรีจิสเตอร์แสดงสถานะโมเด็ม

บิต 0 บิตนี้ใช้สำหรับแสดงการเปลี่ยนแปลงของสัญญาณ CTS กล่าวคือ เมื่อขา CTS ของ 8250 ได้เปลี่ยนสถานะหลังจากที่ซีพียูได้อ่านสถานะนี้ไปแล้ว บิตนี้ก็จะบอกด้วยเซตและเมื่อซีพียูอ่านก็จะได้รับการรีเซต "0" และจะได้รับการเซต "1" เมื่อมีการเปลี่ยนสถานะที่ขา CTS

บิต 1 เหมือนบิต 0 แต่เป็นบิตที่แสดงสถานะการเปลี่ยนแปลงของขา DSR

บิต 2 บิตนี้เป็นบิตแสดงว่าสัญญาณ RI ซึ่งเป็นอินพุตของ 8250 ได้รับการเปลี่ยนจากออน (on) "1" มาเป็นออฟ (off) "0"

บิต 3 บิตนี้เหมือนบิต 0 แต่เป็นบิตแสดงสถานะการเปลี่ยนแปลงของline signal detectorซึ่งเป็นขาอินพุต RLSD

บิต 4 บิตนี้เก็บสัญญาณคอมพลีเมนต์ (complement) กับสัญญาณที่ขา CTS

บิต 5 บิตนี้เก็บสัญญาณคอมพลีเมนต์กับสัญญาณที่ขา DSR

บิต 6 บิตนี้เก็บสัญญาณคอมพลีเมนต์กับสัญญาณที่ขา RI

บิต 7 บิตนี้เก็บสัญญาณคอมพลีเมนต์กับสัญญาณที่ขา RLSD

อนึ่งถ้าบิต 4 ของ MCR ได้รับการเซตหรือให้ทำลูป (loop) ตรวจสอบข้อมูลในบิต 4 จะเหมือนกับ RTS ใน MCR ข้อมูลในบิต 5 จะเหมือนกับ DTR ใน MCR ข้อมูลในบิต 6 จะเหมือนกับ OUT1 ใน MCR ข้อมูลในบิต 7 จะเหมือนกับ OUT2 ใน MCR รีจิสเตอร์บัฟเฟอร์สำหรับตัวรับข้อมูล (receiver buffer register) เป็นรีจิส

เตอร์สำหรับการรับข้อมูลที่มาจากสายสื่อสารสัญญาณ พอร์ตที่กำหนดคือ หมายเลขแอดเดรส 3F8 ขณะที่ DLAB = 0 หากซีพียูอ่านข้อมูลที่รีจิสเตอร์นี้ก็หมายถึงได้อ่านข้อมูลที่มาจากสายสัญญาณสื่อสารนั่นเอง

รีจิสเตอร์โฮลดิงสำหรับตัวส่งข้อมูล (transmitter holding register) เป็นรีจิสเตอร์บัฟเฟอร์สำหรับส่งข้อมูล รีจิสเตอร์นี้จะรับข้อมูลจากซีพียู โดยที่กำหนดพอร์ตเป็นหมายเลข 3F8 เมื่อ DLAB = 0 ข้อมูลของซีพียูที่เอาต์พุตมาที่พอร์ตนี้ก็เพื่อจะส่งออกไปยังสายส่งข้อมูล

2.4 การมอดูเลทสัญญาณดิจิตอล

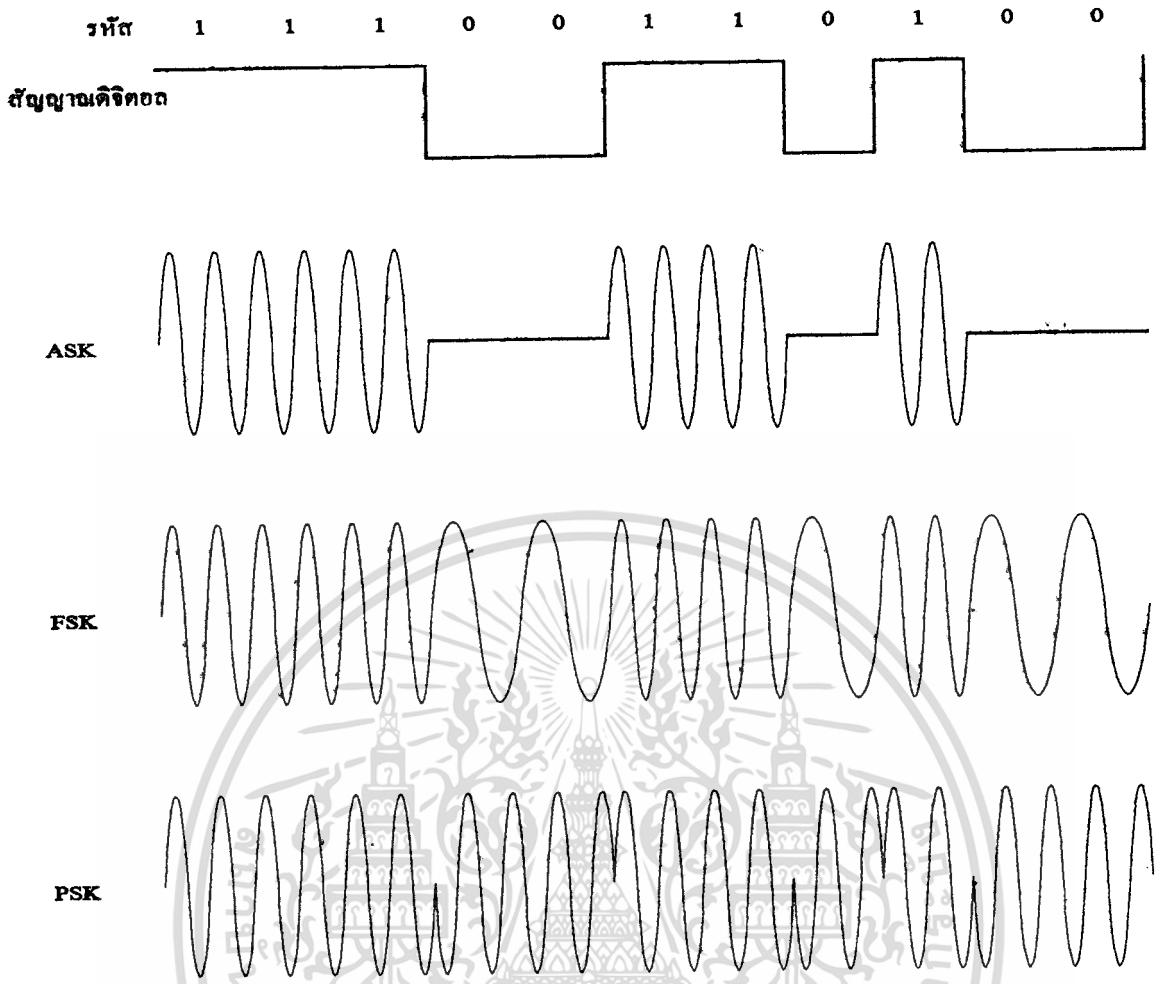
มีวิธีการมอดูเลทอยู่ 3 วิธี

– **การมอดูเลทแบบ เอ เอส เค (ASK)** เป็นการเปลี่ยนขนาดของสัญญาณตามสัญญาณดิจิตอล รูปคลื่นสัญญาณที่ได้จากการมอดูเลทแบบสัญญาณดิจิตอลเปลี่ยนแปลงขนาดสัญญาณที่ได้ตามระดับของสัญญาณดิจิตอลที่เปลี่ยนแปลงไป จากรูปที่ 2.10 ที่ระดับดิจิตอลมีสถานะลอจิก 0 สัญญาณที่ได้จะมีขนาดเป็น 0 และจะมีขนาดเปลี่ยนแปลงไปตามคลื่นพาห์ (Carrier) เมื่อระดับลอจิกมีสถานะเป็น 1 วิธีการนี้มีข้อดีคือ ทั้งภาคมอดูเลเตอร์และดีมอดูเลเตอร์มีส่วนประกอบวงจรที่ง่าย ราคาถูก แต่มีข้อเสียคือข้อมูลที่รับเข้ามาภาคปลายทางผิดพลาดได้ง่าย อันเนื่องมาจากสัญญาณรบกวนที่จะมีผลต่อขนาดของสัญญาณ และทางภาครับมีวงจรขยายชดเชยการลดทอนสัญญาณในสายอัตโนมัติ อีกทั้งยังมีอัตราการส่งข้อมูลได้ไม่สูงมากนัก

– **การมอดูเลทแบบ เอฟ เอส เค (FSK)** เป็นการเปลี่ยนความถี่ตามสัญญาณดิจิตอล รูปคลื่นสัญญาณที่ได้จากการมอดูเลทแบบนี้ สัญญาณดิจิตอลควบคุมความถี่ของสัญญาณที่จะส่งออกจากวงจรมอดูเลชัน โดยให้รูปคลื่นที่ได้มีความถี่สูง เมื่อระดับของสัญญาณดิจิตอลเป็น 1 และมีความถี่ต่ำเมื่อเป็น 0 ซึ่งมีอัตราการส่งข้อมูลต่ำพอๆกับวิธีการ ASK สำหรับกรณีที่ใช้ส่งทางสายที่มีแบนด์วิดท์ไม่เกิน 3.4 กิโลเฮิรตซ์อัตราการส่งข้อมูลสูงสุดจะไม่เกิน 1200 บิตต่อวินาที และวิธีการนี้มีข้อดีเหมือนกับวิธีการมอดูเลทแบบ เอ เอส เค คือมีส่วนประกอบของวงจรที่ง่ายไม่สลับซับซ้อน ราคาถูก และมีเสถียรภาพต่อสัญญาณรบกวนได้สูงกว่า

– **การมอดูเลทแบบ พี เอส เค (PSK)** เป็นการเปลี่ยนเฟสตามสัญญาณดิจิตอล รูปคลื่นสัญญาณที่ได้จากการมอดูเลทแบบสัญญาณดิจิตอลควบคุมการเปลี่ยนเฟสของสัญญาณ จะเห็นได้ว่า เมื่อใช้ความถี่ในการส่งสัญญาณตามระดับของสัญญาณดิจิตอลด้วยความถี่เดียวกันตลอด แต่เมื่อมีการเปลี่ยนแปลงระดับของสัญญาณดิจิตอล ก็จะมีการเปลี่ยนเฟสของคลื่นพาห์เป็นตรงข้าม (180 องศา) วงจรของภาครับและส่งข้อมูลมีความซับซ้อนยากมากราคาสูง แต่สามารถส่งข้อมูลได้สูงกว่า 1200 บิต/วินาที

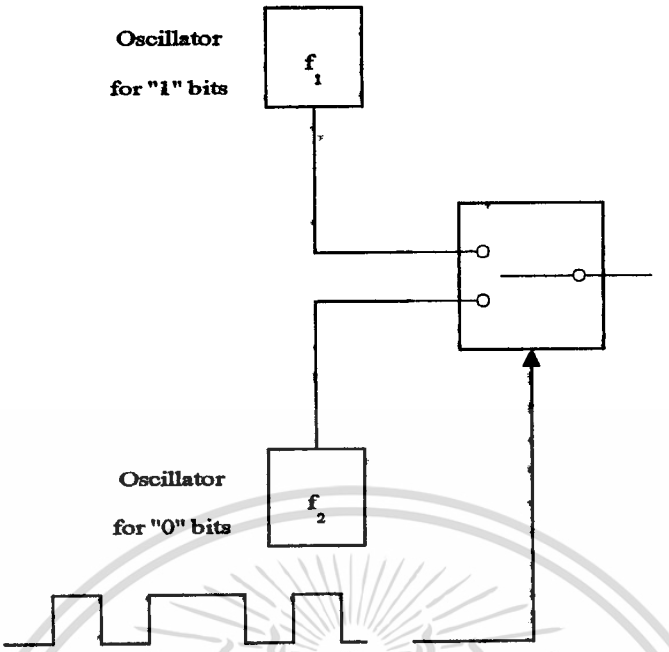
ดังนั้น เมื่อพิจารณาข้อดีข้อเสียของรูปแบบการมอดูเลทสัญญาณแบบดิจิตอลมอดูเลชันด้วยวิธีการต่างๆที่กล่าวมาแล้วจะเห็นว่า เอฟ เอส เค จะมีเสถียรภาพต่อสัญญาณรบกวนได้ดีกว่าระบบ เอ เอส เค และระบบ พี เอส เค จึงเลือกระบบ เอฟ เอส เค มาใช้ในโครงงานนี้



รูปที่ 2.10 รูปคลื่นของการมอดูเลทสัญญาณแบบดิจิตอลมอดูเลชัน

2.4.1 ตัวกำเนิดสัญญาณ เอฟ เอส เค (FSK Modulator)

เนื่องจากการส่งข้อมูลด้วยวิธีการมอดูเลทแบบ เอฟ เอส เค นี้ เป็นการเปลี่ยนแปลงความถี่ในการรับ-ส่งข้อมูลตามระดับของสัญญาณดิจิตอล เมื่อข้อมูลดิจิตอลแบบไบนารีได้มีการเปลี่ยนแปลงระดับที่ใช้ตามสถานะลอจิก 1 และ 0 สัญญาณ เอฟ เอส เค ที่ได้จะมีการเปลี่ยนแปลงระหว่าง 2 ความถี่ด้วยกัน คือให้ความถี่ที่สถานะลอจิก 1 เป็นความถี่มาร์ค (Mark Frequency) และความถี่ที่สถานะลอจิก 0 เป็นความถี่สเปซ (Space Frequency) ดังนั้น ที่เอาท์พุตจะมีการเปลี่ยนแปลงความถี่เมื่อข้อมูลเข้ามาที่อินพุตเปลี่ยนสถานะไป ในระบบดิจิตอลมอดูเลชันนั้น อัตราการเปลี่ยนแปลงของสัญญาณอินพุตเรียกว่าอัตราบิต โดยมีหน่วยเป็นบิตต่อวินาที ส่วนอัตราการเปลี่ยนแปลงของสัญญาณด้านเอาท์พุตเรียกว่า อัตราบอด (Baud Rate) ดังนั้น การส่งสัญญาณข้อมูลแบบระบบ เอฟ เอส เค จะเห็นว่าค่าอัตราบิตและอัตราบอดจะเท่ากันเสมอ แผนภาพการทำงานของภาคมอดูเลชันของการส่งข้อมูลดิจิตอลด้วยวิธีการ เอฟ เอส เค มอดูเลชันในปัจจุบันมีการทำงานอย่างง่ายๆ ดังรูปที่ 2.11 โดยมีแหล่งกำเนิดสัญญาณความถี่หรือออสซิลเลเตอร์ความถี่ f_1 และ f_2 โดยจะใช้ข้อมูลดิจิตอลแบบไบนารีมาควบคุมการทำงานของสวิทช์ให้เลือกตัวกำเนิดสัญญาณ เอฟ เอส เค ซึ่งทำหน้าที่คล้ายสวิทช์ 2 ทาง โดยเลื่อนไปเลือกอินพุตตามสัญญาณไบนารีที่ควบคุม



รูปที่ 2.11 หลักการทำงานอย่างง่ายๆ ของ เอฟ เอส เค มอดูเลเตอร์

ถ้าข้อมูลเข้ามามีลอจิกเป็น 0 จะต้องทำการเลื่อนสวิทช์ไปทางวงจร f_1 เพื่อให้สัญญาณ f_1 ออกที่เอาท์พุต และเมื่อลอจิกเป็น 1 เลื่อนสวิทช์ไปทางวงจร f_2 เพื่อให้ความถี่ f_2 ออกที่เอาท์พุต

สัญญาณ เอฟ เอส เค ที่ได้จะมีสมการดังนี้

$$v(t) = A\cos(\omega_c \pm \Delta\omega)t$$

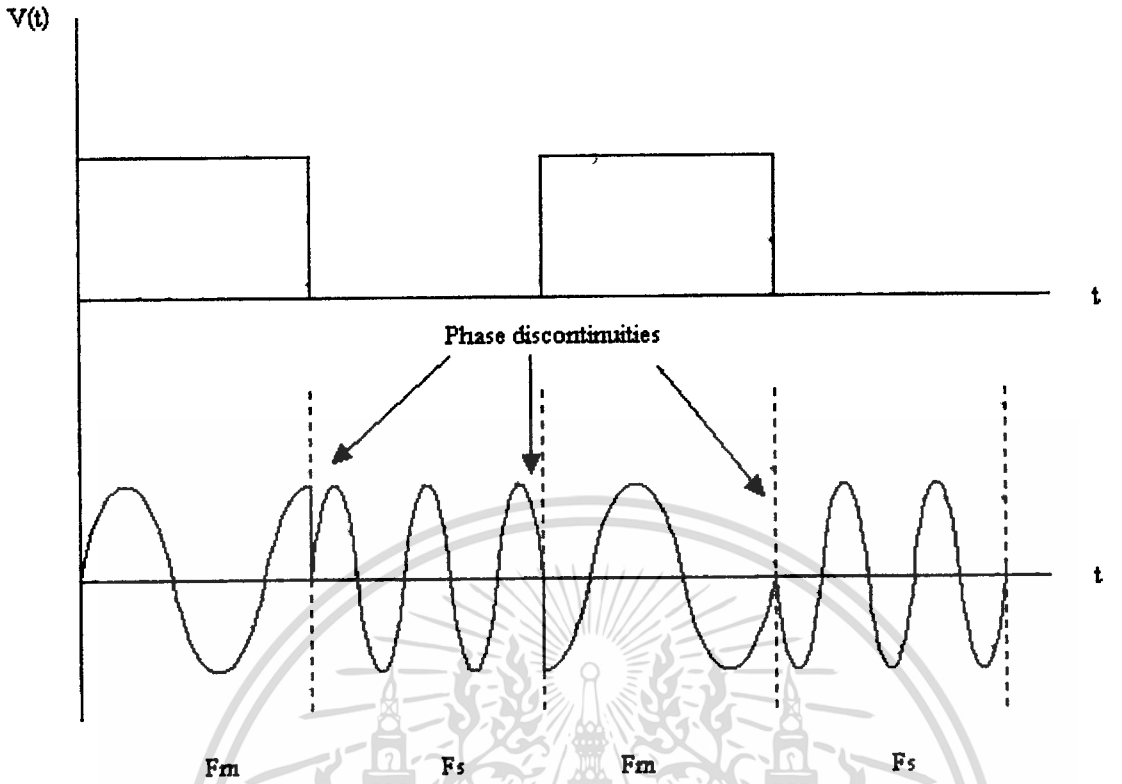
A = ขนาดของสัญญาณ FSK

$\omega_c = 2\pi f_c$

$\Delta\omega$ = ความถี่เบี่ยงเบน

t = เวลา

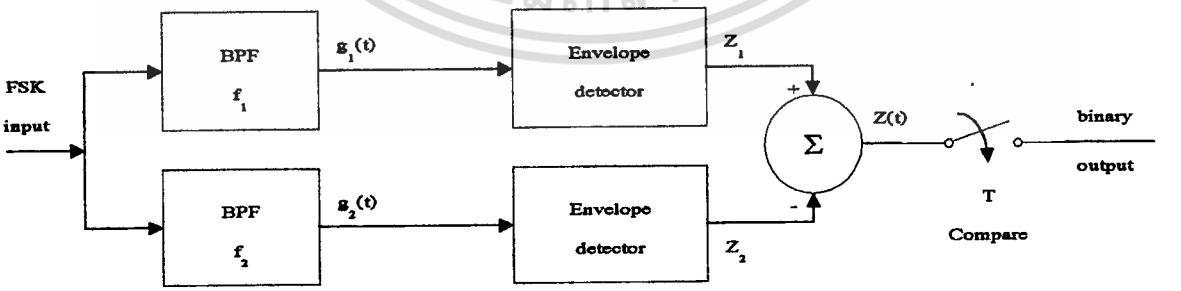
จากวงจรรูปที่ 2.11 จะเห็นว่าการทำงานของวงจรใช้แหล่งกำเนิดสัญญาณความถี่แยกกัน 2 ชุด จึงเป็นสาเหตุทำให้สัญญาณ เอฟ เอส เค ที่ได้ มีเฟสไม่ต่อเนื่อง (Discontinuous Phase) ตรงที่มีการเปลี่ยนแปลงของระดับลอจิกในสัญญาณดิจิตอล ดังรูปที่ 2.12 และจะเห็นว่าสัญญาณที่ได้จากวิธีการแบบ เอฟ เอส เค ที่ได้จะมีรูปคลื่นมากกว่า 1 คาบเวลาต่อสัญญาณดิจิตอล 1 บิต ซึ่งเป็นสาเหตุที่ทำให้อัตราบิตในการส่งข้อมูลได้ไม่สูง



รูปที่ 2.12 สัญญาณ เอฟ เอส เค ที่มีเฟสไม่ต่อเนื่อง

2.4.2 เอฟ เอส เค ดีมอดูเลเตอร์ (FSK Demodulator)

แผนภาพของวงจร เอฟ เอส เค ดีมอดูเลเตอร์ แสดงได้ดังรูปที่ 2.13 ซึ่งวงจรแบ่งออกเป็น 2 ส่วน คือ ส่วนที่ใช้สำหรับแยกความถี่ f_1 และ f_2 โดยใช้วงจรกรองแถบความถี่ผ่าน (Band Pass Filter,BPF) ทำงานร่วมกับวงจรเอนเวลโลปดีเทคเตอร์ (Envelope Detector) ซึ่งมีการทำงานเหมือนกับวิธีการดีมอดูเลทของเครื่องรับวิทยุระบบ เอ เอ็ม เมื่อให้สัญญาณที่ได้ผ่านเข้าวงจรรวมสัญญาณแบบลบ ก็จะได้สัญญาณข้อมูลดิจิตอลตามแบบสัญญาณอินพุตของภาคมอดูเลเตอร์

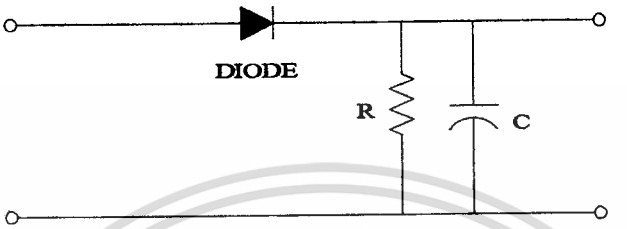


รูปที่ 2.13 การทำงานของภาค เอฟ เอส เค ดีมอดูเลเตอร์

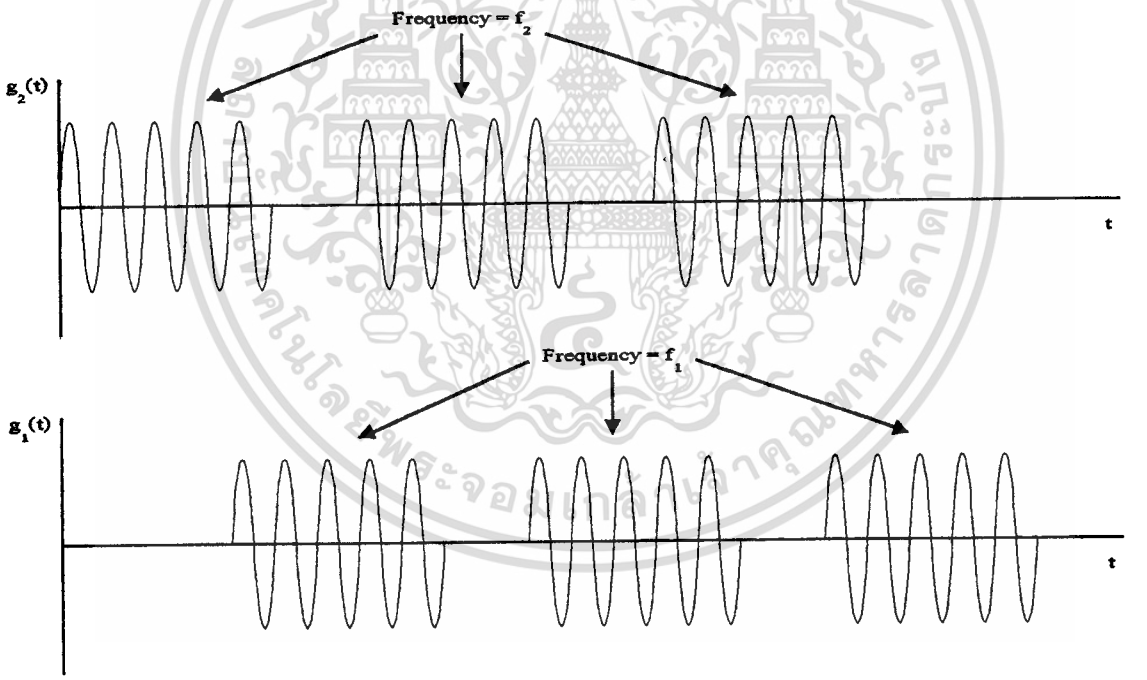
จากการพิจารณาวงจรกรองแถบความถี่ผ่าน ซึ่งเสมือนเป็นตัวกำหนดให้อัตราบิตในการส่งไม่สูง เพราะว่า ถ้าให้ความถี่ f_1 และ f_2 มีค่าใกล้เคียงกันมาก ค่า Q ของวงจรกรองแถบความถี่ต้องสูงจึงสามารถที่จะแยก

สัญญาณได้เด็ดขาด ดังนั้น ความถี่ f_1 และ f_2 จะต้องมีความถี่ห่างกันพอสมควรเพื่อป้องกันการผิดพลาดที่จะเกิดขึ้นในการแยกสัญญาณ

วงจรเอนเวลโลปดีเทคเตอร์ที่แสดงได้ดังรูปที่ 2.14 มีการทำงานเหมือนกับการดีมอดูเลทในเครื่องรับวิทยุ เอ เอ็ม หรือที่รู้จักกันในชื่อของวงจรไดโอดดีเทคเตอร์ (Diode Detector) สัญญาณเอาท์พุตที่ออกจากวงจรกรองแถบความถี่ผ่านทั้งสองมีลักษณะดังรูปที่ 2.15



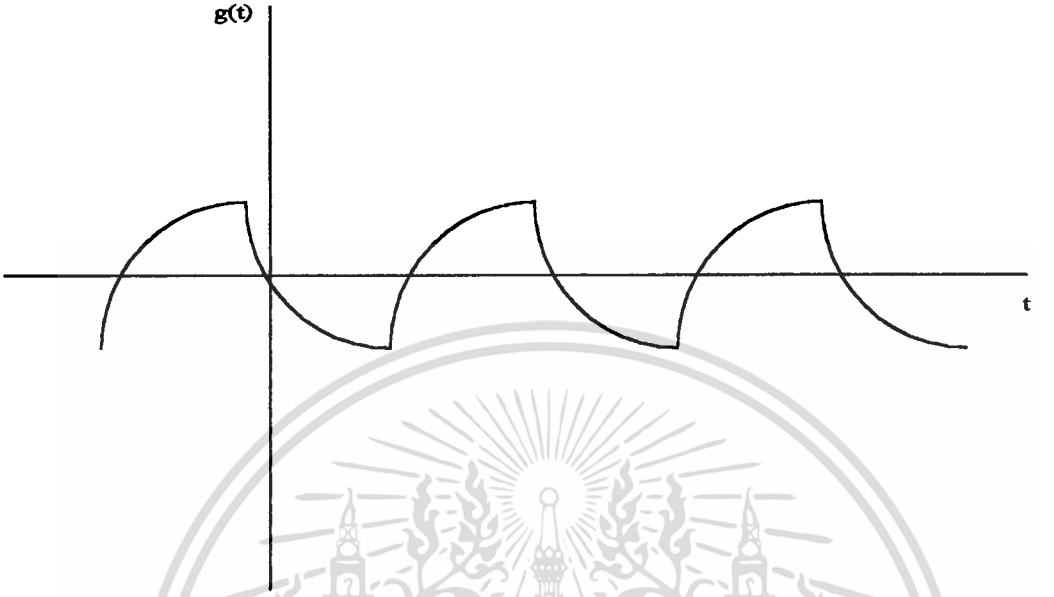
รูปที่ 2.14 วงจรเอนเวลโลปดีเทคเตอร์



รูปที่ 2.15 สัญญาณอินพุตของวงจรเอนเวลโลปดีเทคเตอร์

หลักการทำงานของวงจรเอนเวลโลปดีเทคเตอร์จะเลือกเอาสัญญาณซีกบวกของสัญญาณ $g_1(t)$ และ $g_2(t)$ เข้ามาประจุ (Charge) แรงดันให้แก่ตัวเก็บประจุ และคายประจุ (Discharge) จนกว่าแรงดันที่คร่อมตัวเก็บประจุมีค่าต่ำกว่าขนาดแรงดันที่เข้ามาใหม่ก็จะทำการเก็บประจุใหม่ จากการเลือกค่าของตัวต้านทานและตัวเก็บประจุที่ถูกต้อง จะได้แรงดันที่เอาท์พุตมีขนาดเปลี่ยนแปลงตามเอนเวลโลปของสัญญาณ

สัญญาณเอาท์พุตที่วงจรเอนเวลโลปดีเทคเตอร์ มีลักษณะเป็นแบบเอ็กโพเนนเชียล (Exponential) แสดงได้ดังรูปที่ 2.16



รูปที่ 2.16 สัญญาณที่เอาท์พุตของวงจรเอนเวลโลปดีเทคเตอร์

ในการใช้วงจรกรองแถบความถี่ผ่านที่มีค่า Q ของวงจรสูงมากๆ จะทำให้การทำงานของวงจรเอนเวลโลปดีเทคเตอร์ทำงานผิดพลาดได้ง่าย ดังนั้นสัญญาณที่ออกจากวงจรเอนเวลโลปดีเทคเตอร์ Z_1 และ Z_2 จะถูกนำมาลบกันแล้วส่งให้วงจรเปรียบเทียบแรงดัน เพื่อให้ได้ข้อมูลกลับออกมาเป็นข้อมูลดิจิตอลต่อไป

จากหลักการมอดูเลทและดีมอดูเลทด้วยวิธี เอฟ เอส เค นี้ได้นำมาใช้ในระบบโมเด็มในการสื่อสารข้อมูลผ่านทางสายโทรศัพท์ จะเป็นโมเด็มที่มีอัตราความเร็วในการส่งต่ำ เช่น ถ้าให้ความถี่สูงและความถี่ต่ำ มีค่า 2 กิโลเฮิรตซ์ และ 1 กิโลเฮิรตซ์ตามลำดับ จะเห็นว่าในการมอดูเลทและดีมอดูเลทต้องใช้ 2 ไซเคิลต่อการส่งข้อมูล 1 บิต ที่การส่งความถี่สูงทำให้ส่งข้อมูลได้สูงสุดไม่เกิน 1200 บิตต่อวินาที หรือถ้าใช้ความถี่สูงและความถี่ต่ำต่างกันมากๆ เพื่อให้วงจรกรองแถบความถี่ผ่านไม่ต้องมีค่า Q สูงๆ ก็จะเป็นตัวทำให้อัตราการส่ง ถูกจำกัดด้วยความถี่ต่ำสุดที่เลือกใช้ เพราะว่าความถี่ต่ำนี้จะเป็นตัวกำหนดอัตราส่งต่ำสุดที่ส่งได้ ดังนั้นจะเห็นว่าอัตราการส่ง-รับข้อมูล จะถูกกำหนดด้วยวงจรกรองแถบความถี่ผ่านของภาคดีมอดูเลเตอร์ด้วย

อีกสิ่งหนึ่งที่เป็นตัวจำกัดอัตราการส่ง-รับ ข้อมูลก็คือเฟสของสัญญาณไม่ต่อเนื่องตรงจุดที่การเปลี่ยนแปลงของสัญญาณข้อมูลจาก 0 เป็น 1 หรือจาก 1 เป็น 0 ดังรูปที่ 2.12 เพราะจะเกิดสเปคตรัมของสัญญาณความถี่สูง และจะทำให้การทำงานของภาคดีมอดูเลเตอร์ทำงานผิดพลาด โมเด็มที่ใช้วิธี เอฟ เอส เค ที่จัดอยู่ในกลุ่มความเร็วต่ำมีค่าอัตราบิตไม่เกิน 1200 บิตต่อวินาที เช่น Bell 103, Bell 202, CCITT V21 และ CCITT V23 เป็นต้น

2.5 เครื่องส่งสัญญาณคลื่นวิทยุ

จุดเริ่มต้นของการส่งคลื่นวิทยุนั้นประกอบด้วยการสร้างสัญญาณที่เป็นสื่อหรือพาหะที่เรียกว่าแคร์เรียร์เจเนอเรชั่น (Carrier generation) นำสัญญาณมามอดูเลทกับสัญญาณข้อมูลหรือเสียง แล้วจึงทำการขยายสัญญาณเพื่อส่งออกไปยังสายอากาศหรือตัวนำคลื่นเป็นสัญญาณความถี่วิทยุ (RF) ขอยกตัวอย่างเช่นเครื่องส่งสัญญาณรหัสมอร์สหรือเครื่องส่งสัญญาณแบบคลื่นต่อเนื่อง (CW, Continuous Wave) ที่ให้สัญญาณแบบสั้นและยาวที่เรียกว่าดอตส์แอนด์แดชส์ (Dots and Dashs) โดยมีวงจรสร้างสัญญาณพาหะก็คือออสซิลเลเตอร์ ซึ่งต่อเชื่อมเข้ากับวงจรขยายเพื่อเพิ่มกำลังส่งออกไปกับสายอากาศ ส่วนที่สร้างสัญญาณข้อมูลเพื่อมอดูเลตสัญญาณที่ได้จากออสซิลเลเตอร์เข้ากับสวิทช์แบบกดติดปล่อยดับ ที่ต่อสัญญาณลงกราวด์ (ground) จากตัวอย่างดังกล่าวทำให้พอที่จะมองภาพของส่วนประกอบของเครื่องส่งสัญญาณได้ดังนี้

2.5.1 วงจรสร้างสัญญาณพาหะ (Carrier Generator) โดยส่วนมากจะเป็นวงจรคริสตอลออสซิลเลเตอร์ (Crystal Oscillator) ซึ่งจะให้สัญญาณความถี่ที่ต้องการได้เที่ยงตรงและมีเสถียรภาพดี โดยส่วนมากมักมีการต่อวงจรขยายแบบบัฟเฟอร์เข้าไปเพื่อแยกวงจรออสซิลเลเตอร์ออกจากโหลด เป็นการป้องกันการเปลี่ยนความถี่เนื่องจากค่าโหลดของวงจรออสซิลเลเตอร์มีการเปลี่ยนแปลงไป

2.5.2 วงจรมอดูเลทสัญญาณ (Modulator) ทำการแปลงคุณสมบัติของสัญญาณพาหะให้มีลักษณะตามการเปลี่ยนแปลงของข้อมูลหรือเสียงพูดที่ต้องการส่ง จากตัวอย่างข้างต้นก็เสมือนกับสวิทช์ที่ต่อลงกราวด์หรือจะเป็นวงจรมอดูเลทในวิธีการต่างๆ เช่นแอมปลิจูดมอดูเลชั่น (AM) หรือฟรีเควนซี่มอดูเลชั่น (FM)

2.5.3 วงจรขยาย (Amplifier) เป็นการขยายสัญญาณในรูปแบบต่างๆ ในขั้นตอนที่ต่างกัน ตัวอย่างเช่นวงจรขยายภาคสุดท้ายก่อนที่จะออกไปที่สายอากาศ สำหรับวงจรขยายมีการจัดออกเป็นหลายแบบมีการเรียกเป็นคลาส (Class) ตามวิธีการไบแอสวงจรขยาย ดังต่อไปนี้

2.5.3.1 วงจรขยายคลาสเอ (Class A) เป็นวงจรขยายที่ต่อทรานซิสเตอร์แบบที่มีการไบแอสให้วงจรทรานซิสเตอร์มีกระแสไหลผ่านขาคอลเล็กเตอร์ (Collector) หรือที่เรียกว่ากระแสเดรน (Drain Current) ตลอดเวลา เป็นวงจรขยายแบบเชิงเส้น (Linear Amplifier) เนื่องจากสัญญาณที่ได้ในขาออกเป็นสัดส่วนโดยตรงกับสัญญาณขาเข้า แต่วงจรคลาสเอก็เป็นวงจรขยายที่ไม่ค่อยมีประสิทธิภาพ เนื่องจากวงจรมีการขยายสัญญาณโดยทำงานตลอดลูกคลื่นสัญญาณซายน์ขาเข้า หรือที่เรียกว่าครบ 360 องศา ดังนั้นวงจรขยายคลาสเอจึงไม่เหมาะที่จะเป็นวงจรขยายกำลัง (Power Amplifier) เพราะโดยปกติมักใช้ในวงจรขยายขั้นต้นที่มีสัญญาณความต่างศักย์ต่ำ (Low-power Amplifier) ตัวอย่างเช่นวงจรขยายแบบบัฟเฟอร์ (Buffer Amplifier)

2.5.3.2 วงจรขยายคลาสบี (Class B) เป็นวงจรขยายที่ต่อทรานซิสเตอร์แบบที่มีการไบแอสให้ทรานซิสเตอร์อยู่ในช่วงคัตออฟ (Cutoff) เป็นภาวะที่ไม่มีกระแสไหล โดยปกติเมื่อไม่มีสัญญาณขาเข้าก็จะไม่มีกระแสไหลที่ขาคอลเล็กเตอร์ ตัวทรานซิสเตอร์จะทำงานนำกระแสเพียงช่วงครึ่งลูกของสัญญาณซายน์คือจะทำงานเพียง 180 องศาของสัญญาณขาเข้า มีเพียงสัญญาณครึ่งลูกเท่านั้นที่ถูกขยายดังนั้นในเวลาที่ต้องการขยายสัญญาณเต็มลูกคลื่นจึงมีการต่อวงจรแบบที่เรียกว่าพุชแอนด์พูล (Push and Pull) โดยใช้วงจรขยายคลาสบีสองชุดทำงานทั้งในช่วงสัญญาณบวกและสัญญาณลบสลับต่อเนื่องกันไป วงจรขยายคลาสบีมีประสิทธิภาพดีกว่าวงจรขยายคลาสเอ เนื่องจากการไหลของกระแสไฟจะเกิดเพียงช่วงหนึ่งของสัญญาณเท่านั้น ซึ่งเหมาะสำหรับ

วงจรขยายกำลัง แต่สัญญาณที่ได้ก็ยังมีความเพี้ยนผิดรูปแบบ (Distortion) ไป ดังนั้นจึงมีการต่อวงจรขยายแบบพุชแอนด์พูลเพื่อลดความเพี้ยนของสัญญาณด้วย

2.5.3.3 วงจรขยายคลาสเอบี (Class AB) เป็นวงจรขยายที่มีการไบแอสทรานซิสเตอร์ให้อยู่ในช่วงที่เกือบจะคัตออฟจึงมีกระแสไหลที่ขาคอลเล็กเตอร์เพียงเล็กน้อย ทำให้เมื่อมีสัญญาณขาเข้ามาถึงก็จะทำงานในช่วงของสัญญาณที่มากกว่า 180 องศา แต่ไม่ถึง 360 องศาของลูกคลื่นซายน์และก็มักมีการต่อใช้ในรูปแบบของวงจรขยายแบบพุชแอนด์พูลเช่นเดียวกับคลาสบี ซึ่งทำให้มีความเพี้ยนของสัญญาณน้อยกว่าในแบบคลาสบี

วงจรขยายในแบบคลาสเอ, คลาสบี และคลาสเอบี เป็นวงจรขยายเชิงเส้นที่มักใช้ในการขยายสัญญาณคลื่นวิทยุที่มีการเปลี่ยนแปลงแอมปลิจูด (amplitude) เช่นวงจรขยาย AM แบบกำลังต่ำหรือแบบซิงเกิลไซด์แบนด์ (SSB, Single SideBand) วงจรขยายแบบไม่เชิงเส้นเช่นวงจรขยายคลาสซี (Class C) ที่เป็นวงจรซึ่งใช้มากในเครื่องส่งแบบ AM และ FM สำหรับการขยายกำลังในรูปของวงจรขับ (Driver), วงจรสูณความถี่ (Frequency Multiplier) และวงจรขยายภาคสุดท้าย (Final Amplifier)

2.5.3.4 วงจรขยายคลาสซี (Class C) เป็นวงจรทรานซิสเตอร์ที่ถูกไบแอสที่ทำให้มีการนำสัญญาณเพียงส่วนที่น้อยกว่า 180 องศาของลูกคลื่นซายน์ขาเข้า วิธีการไบแอสทรานซิสเตอร์สำหรับคลาสซีมีอยู่ด้วยกัน 3 วิธี คือไบแอสด้วยสัญญาณ (Signal Bias) ไบแอสด้วยวงจรภายนอก (External Bias), ไบแอสด้วยตนเอง (Self Bias) และปกติมีมุมช่วงทำงานของวงจรอยู่ในช่วง 90 องศาถึง 150 องศา นั่นหมายถึงมีเพียงสัญญาณเพียงพัลซ์เล็กๆเท่านั้นออกมาที่ขาออก ดังนั้นการทำให้มีสัญญาณขยายเต็มลูกคลื่นจะต้องมีการนำวงจรจูนเรโซแนนซ์ (Resonant Tuned Circuit) มาต่อเข้าที่ขาออกเพื่อที่จะได้สัญญาณลูกคลื่นซายน์ (sine) เต็มลูกคลื่น ตัวอย่างเช่นการทำงานของวงจรจูนแบบคู่ขนาน (Parallel Tuned Circuit) ที่จะให้กำเนิดสัญญาณความถี่ที่ความถี่เรโซแนนซ์ เมื่อใดก็ตามที่ได้รับสัญญาณพัลซ์จากวงจรขยาย วงจรจูนแบบคู่ขนานประกอบไปด้วยคาปาซิเตอร์ (Capacitor) และลวดตัวนำ (Inductor) เมื่อได้รับสัญญาณพัลซ์ก็จะเกิดการถ่ายทอดพลังงานระหว่างคาปาซิเตอร์และตัวนำ ซึ่งเรียกว่าปรากฏการณ์ ฟลายวีล (Fly Wheel Effect) ซึ่งจะสร้างสัญญาณลูกคลื่นซายน์ที่ความถี่เรโซแนนซ์ ขณะเดียวกันวงจรจูนดังกล่าวก็ทำหน้าที่กรองสัญญาณความถี่ฮาร์มอนิก (harmonic) ที่ไม่ต้องการออกไปด้วย วงจรขยายคลาสซีสามารถใช้เป็นวงจรคูณความถี่ก็ได้ โดยการต่อเข้ากับวงจรเรโซแนนซ์ที่สร้างความถี่เป็นจำนวนเต็มเท่าของสัญญาณความถี่ขาเข้า และที่เหนือกว่าคลาสอื่นๆ ก็คือคลาสซีมีการขยายสัญญาณขาเข้าเพียงช่วงสั้นๆเท่านั้น ดังนั้นจึงมีประสิทธิภาพที่ดีที่สุดในบรรดาวงจรขยายทั้งหมด

2.5.4 วงจรอิมพีแดนซ์แมตชิ่ง (Impedance Matching Circuit) เป็นวงจรที่ใช้สำหรับเชื่อมต่อระหว่างวงจรขยายความถี่วิทยุ (RF amplifier) ในแต่ละภาคเพื่อให้ได้กำลังส่งที่มากที่สุด การที่จะทำให้มีการถ่ายพลังงานมากที่สุดจากวงจรขยายชุดหนึ่งไปยังอีกชุดหนึ่งจะต้องมีค่าของอิมพีแดนซ์ (impedance) ของวงจรแรกเท่ากับค่าอิมพีแดนซ์ขาเข้าของวงจรถัดไป วงจรอิมพีแดนซ์แมตชิ่งโดยทั่วไปเป็นวงจรของตัวนำและตัวเก็บประจุ LC (Inductors and Capacitors) ที่มีรูปแบบการต่อต่างๆกัน เช่นโครงข่ายรูป L และโครงข่ายรูป T หรืออาจจะเป็นหม้อแปลงรูปโดนัทที่เป็นแกนผงเหล็กเรียกว่าทอรอยด์ (Toroid)

2.5.5 วงจรกระบวนการเสียง (Speech Processing Circuit) เป็นวงจรอีกส่วนหนึ่งซึ่งทำหน้าที่เกี่ยวกับเสียงในระบบของเครื่องส่ง เช่นในเครื่องส่งอาจมีวงจรที่ใช้สำหรับป้องกันการมอดูเลชั่นมากเกินไป (Over Modulation) หรือตัวอย่างของวงจรกระบวนการเสียง เช่นวงจรจำกัดขนาดของเสียง (Voice Clipper) ซึ่งใช้ไดโอดในการลดแอมปลิจูดของสัญญาณในการมอดูเลทสัญญาณเสียง

2.6 เครื่องรับสัญญาณ (Communications Receivers)

หน้าที่ของเครื่องรับสัญญาณคือทำการเลือกช่องสัญญาณที่ต้องการออกมาจากสัญญาณอื่นๆ ที่ถูกส่งออกมาในอากาศ และขยายสัญญาณกลับไปเป็นสัญญาณข้อมูลที่ส่งมาได้ โดยปกติเครื่องรับจะมีปัจจัย 2 ประการที่ต้องคำนึงถึงดังนี้

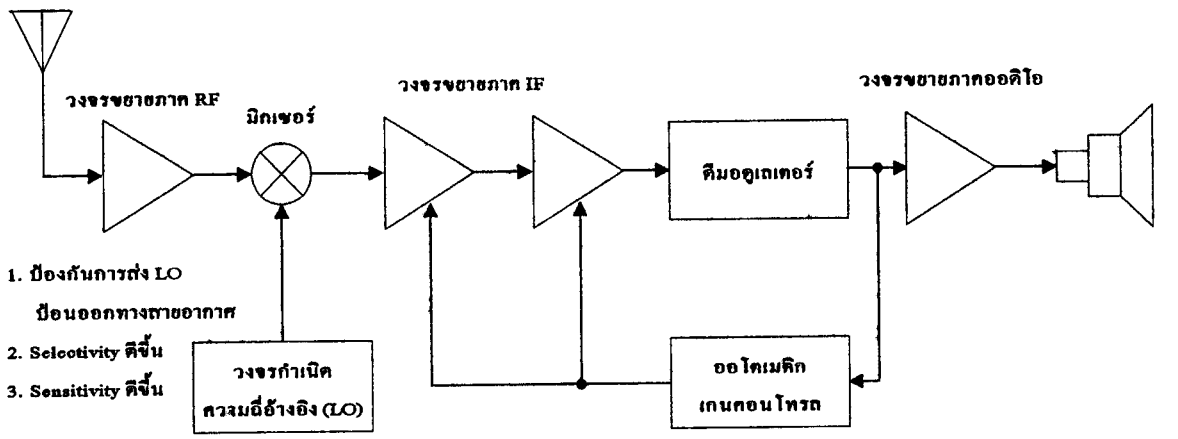
2.6.1 ค่าซีเล็กติวิตี้ (Selectivity) หมายถึงความสามารถในการรับสัญญาณโดยเลือกเอาเฉพาะช่องสัญญาณที่ต้องการเข้ามาเท่านั้น หากถ้าเครื่องรับสัญญาณมีค่าซีเล็กติวิตี้ (Selectivity) ที่ดีก็จะสามารถรับสัญญาณช่องที่ต้องการและกำจัดช่องสัญญาณข้างเคียงออกไปได้

2.6.2 ค่าเซนซิติวิตี้ (Sensitivity) หมายถึงความสามารถในการรับสัญญาณที่ต้องการที่มีขนาดเล็กหรือสัญญาณอ่อนแล้วนำมาขยายให้ได้สัญญาณที่มีความแรงมากขึ้น โดยปกติค่าเซนซิติวิตี้จะแสดงถึงการขยายสัญญาณด้วย นั่นคือยิ่งมีกำลังขยายมากค่าเซนซิติวิตี้ก็ยิ่งดี และจะแสดงอยู่ในรูปของค่าแรงดันของสัญญาณขาเข้าที่มีขนาดเล็กที่สุดซึ่งจะสามารถขยายสัญญาณได้มากกว่า 10 เท่าของสัญญาณรบกวน

สำหรับเครื่องรับที่มีรูปแบบง่ายได้แก่เครื่องรับจูนความถี่วิทยุ TRF (Tuned radio frequency receiver) ซึ่งมีการทำงานดังนี้ สัญญาณที่รับเข้ามาทางสายอากาศจะถูกต่อเข้ากับวงจรจูนซึ่งมีการต่อกับวงจรขยายที่เป็นวงจรสำหรับภาคความถี่ย่านคลื่นวิทยุ วงจรจูนอาจมีการต่อขนานกันหลายชั้น ซึ่งจะช่วยเพิ่มค่าซีเล็กติวิตี้ให้กับเครื่องรับ ส่วนวงจรขยายสัญญาณภาคความถี่ย่านคลื่นวิทยุ (RF Radio Frequency Amplifier) ก็ได้ช่วยให้เครื่องรับมีค่าเซนซิติวิตี้ดีขึ้น เป็นการขยายสัญญาณที่รับเข้ามาก่อนที่จะนำไปเข้าวงจรตรวจจับสัญญาณ (Detector) ผลที่ได้ก็จะเป็นสัญญาณข้อมูลหรือสัญญาณเสียงที่สามารถนำมาขยายต่อในวงจรขยายภาคสัญญาณความถี่เสียง (AF Audio Frequency Amplifier) ให้ได้สัญญาณออกมาที่ลำโพง วงจรเครื่องรับในแบบ TRF นั้นยังมีความยุ่งยากในการปรับความถี่อยู่มากเนื่องจากการปรับวงจรจูนที่มีหลายชุดต่อๆกัน จะต้องทำการปรับหลายครั้ง ต่อมาในภายหลังจึงมีการต่อวงจรจูนหลายๆชุดเข้าด้วยกัน ทำให้การปรับเครื่องรับสัญญาณง่ายขึ้น ปัญหาที่สำคัญอีกอย่างของเครื่องรับสัญญาณแบบ TRF ก็คือค่าซีเล็กติวิตี้จะเปลี่ยนแปลงไปตามค่าของความถี่ของสัญญาณที่สูงขึ้น ค่าซีเล็กติวิตี้จะมีค่าดีที่ความถี่ต่ำ

เครื่องรับที่มีการแก้ไขปัญหาข้างต้นได้อย่างดีก็คือเครื่องรับในแบบที่เรียกว่าซูเปอร์เฮตเทอโรไดน์ (Superheterodyne) หลักการของวงจรซูเปอร์เฮตเทอโรไดน์ก็คือการแปลงความถี่ของสัญญาณที่เข้ามาให้เป็นความถี่กลางค่าหนึ่งซึ่งเรียกว่าความถี่ไอเอฟ (IF Intermediate Frequency)

วงจรซูเปอร์เฮตเทอโรไดน์สามารถใช้วงจรขยายเพียงชุดเดียวก็สามารถให้ค่าซีเล็กติวิตี้และค่าเซนซิติวิตี้ที่ดีได้ วงจรหลักในเครื่องรับซูเปอร์เฮตเทอโรไดน์ก็คือวงจรมิกเซอร์ (mixer) ซึ่งทำการแปลงความถี่ของสัญญาณที่เข้ามา รูปไดอะแกรมของวงจรซูเปอร์เฮตเทอโรไดน์แสดงดังรูปที่ 2.17



รูปที่ 2.17 วงจรซูเปอร์เฮดเทอโรไดน์

วงจรขยายสัญญาณความถี่วิทยุให้ค่ากำลังขยายและค่าซีเล็กติวิตี้ในช่วงแรกที่เรียกกันว่า ปรีซีเล็กเตอร์ (Preselector) ถัดมาในภาคที่ 2 เป็นวงจรจูน (Tuned Circuit) สำหรับช่วยในการเลือกสัญญาณที่ต้องการหรือช่วงสัญญาณที่ต้องการ วงจรจูนอาจสร้างให้มีค่า Q สูงๆ ทำให้มีค่าซีเล็กติวิตี้ดีขึ้น แต่โดยปกติแล้ววงจรจูนในภาคนี้มักต้องทำงานในช่วงความถี่กว้าง เพื่อให้สามารถรับสัญญาณได้หลายช่อง ในเครื่องรับบางเครื่องอาจไม่ใช้วงจรขยายสัญญาณความถี่วิทยุในชุดแรกเนื่องจากไม่มีความจำเป็น เพราะความแรงของสัญญาณที่ได้รับอาจมีมากอยู่แล้ว เช่นในสัญญาณความถี่ต่ำแต่จะไปขยายสัญญาณอีกครั้งในภาคความถี่ตัวกลาง (IF amplifier) แต่โดยทั่วไปจะเป็นการดีกว่าที่จะมีวงจรขยายความถี่วิทยุอยู่เพื่อเพิ่มค่าเซนซิติวิตี้ เนื่องจากว่าจะได้กำลังขยายมากขึ้นและเพิ่มค่าซีเล็กติวิตี้เพราะเป็นวงจรจูนอยู่ด้วยส่วนหนึ่ง และทำให้อัตราส่วนของสัญญาณที่ต้องการต่อสัญญาณรบกวนมากขึ้นด้วย (Signal/Noise Ratio) อีกเหตุผลหนึ่งที่ควรจะมีวงจรขยายความถี่วิทยุเพราะจะช่วยแยกสัญญาณรบกวนที่อาจจะเกิดขึ้นได้กับเครื่องรับข้างเคียงที่เป็นผลมาจากการแพร่กระจายของสัญญาณจากวงจรโลคอลออสซิลเลเตอร์ (Local Oscillator) ที่อาจผ่านไปทางสายอากาศได้ สัญญาณจาก LO มีความแรงมากอาจจะรั่วไหลและไปเข้าที่ขาเข้าของวงจรมิกเซอร์ได้ ในการสร้างวงจรขยายและวงจรมิกเซอร์หากใช้วงจรทรานซิสเตอร์ชนิดมอสเฟต (MOSFET) ก็จะช่วยลดสัญญาณรบกวนได้ดีกว่าอุปกรณ์ทรานซิสเตอร์แบบไบโพล่า (Bipolar Transistor)

สัญญาณที่ได้ออกจากมิกเซอร์จะเป็นผลรวมและผลต่างของความถี่ของสัญญาณขาเข้าและสัญญาณความถี่คงที่จาก LO และจะมีวงจรจูนซึ่งเป็นวงจรกรองเพื่อเลือกเอาสัญญาณผลต่างของความถี่ที่ต้องการออกมา นั่นคือค่าความถี่กลาง (Intermediate Frequency) วงจรของมิกเซอร์อาจสร้างจากไดโอดหรือบาลานซ์มอดูเลเตอร์ (Balanced Modulator) สำหรับเครื่องรับที่สามารถรับสัญญาณได้ในช่วงความถี่หนึ่งๆ วงจร LO จะต้องสามารถจูนได้ ความถี่ของวงจรต้องสามารถเปลี่ยนได้ในช่วงความถี่ที่ค่อนข้างกว้างเพื่อที่จะทำให้วงจรมิกเซอร์สามารถแปลงความถี่ที่เข้ามาให้เป็นความถี่กลาง IF ได้ ในวงจรทั่วไปมิกเซอร์และ LO จะเป็นวงจรแยกกัน แต่สำหรับวงจรความถี่ต่ำมิกเซอร์อาจจะรวมกับ LO ได้ ซึ่งเรียกว่าเป็นวงจรแปลง (Converter)

สัญญาณขาออกของมิกเซอร์เป็นสัญญาณที่ความถี่กลางซึ่งมีคุณสมบัติของสัญญาณที่ถูกมอดูเลทจากทางด้านเครื่องส่งเช่นเดียวกับสัญญาณที่ถูกส่งมาจะถูกขยายโดยวงจรขยายความถี่กลางอีกหลายชุด และใน

เครื่องรับส่วนใหญ่จะมีวงจรขยายอยู่ในภาคความถี่กลางนี้ เมื่อ IF มักอยู่ในช่วงความถี่ต่ำกว่าสัญญาณขาเข้า วงจรขยายความถี่กลางก็จะถูกออกแบบได้ง่ายกว่าและมีค่าซีเล็กติวิตี้ดีกว่า พร้อมกันนี้ก็จะมีการสร้างวงจรจูนในภาคนี้ด้วย ซึ่งก็จะให้ค่าซีเล็กติวิตี้ที่ดีขึ้นอีกระดับหนึ่ง วงจรจูนในที่นี้ก็คือวงจรกรองแบบคริสตอล, เมคานิคอล (Mechanical), และแบบเซรามิก (Ceramic)

สัญญาณ IF จะถูกส่งต่อไปเข้าวงจรตรวจจับหรือดีมอดูเลเตอร์ (Demodulator) ซึ่งทำหน้าที่แปลงสัญญาณที่เข้ามาให้กลับคืนเป็นสัญญาณข้อมูลเดิมหรือคือเสียงพูดจากต้นทาง ผลลัพธ์เป็นสัญญาณที่ได้มักจะถูกต่อเข้ากับวงจรขยายสัญญาณความถี่เสียง (Audio amplifier) เพื่อให้ได้ค่าความแรงของสัญญาณที่เพียงพอจะออกไปที่ลำโพง

วงจรที่สำคัญอีกชุดหนึ่งในเครื่องรับแบบซูเปอร์เฮตเทอโรไดน์ก็คือ วงจรควบคุมกำลังขยายอัตโนมัติ AGC (Autometic Gain Control) ขนาดของสัญญาณที่ออกมาจากวงจรดีมอดูเลเตอร์จะเป็นสัดส่วนโดยตรงกับขนาดของสัญญาณขาเข้าที่รับเข้ามา สัญญาณที่ได้ออกมาเป็นสัญญาณในแบบไฟสลับที่ถูกปรับและกรองให้เป็นสัญญาณไฟตรง ซึ่งไฟตรงนี้จะถูกป้อนกลับ (Feedback) ไปยังวงจรขยายความถี่กลาง หรือในบางครั้งอาจเป็นวงจรขยายความถี่วิทยุ เพื่อควบคุมกำลังขยายของเครื่องรับ วัตถุประสงค์ของ AGC ก็เพื่อช่วยควบคุมค่าผลลัพธ์ของสัญญาณขาออกให้คงที่ตลอดช่วงระดับของช่องสัญญาณคลื่นวิทยุที่เข้ามา

ค่าแอมปลิจูดของสัญญาณคลื่นวิทยุที่สายอากาศของเครื่องรับสามารถมีค่าตั้งแต่ระดับไมโครโวลต์ไปจนถึงระดับหลายโวลต์ ซึ่งแสดงถึงช่วงกว้างของสัญญาณที่เรียกว่าช่วงไดนามิก (Dynamic Range) โดยปกติเครื่องรับมักมีกำลังขยาย (Gain) เพื่อที่จะรับสัญญาณที่มีระดับอ่อนได้ดี แต่ถ้าหากสัญญาณขาเข้ามีแอมปลิจูดสูงมากก็จะทำให้วงจรขยายมากเกินไปเกิดโอเวอร์โหลด (Overload) เกิดความผิดเพี้ยนของสัญญาณและทำให้ไม่สามารถเข้าใจข้อความที่ถูกส่งมาได้ โดยการใช้วงจรควบคุมกำลังขยายอัตโนมัติ กำลังขยายโดยรวมของเครื่องรับจะสามารถปรับโดยอัตโนมัติขึ้นอยู่กับสัญญาณขาเข้า หากสัญญาณที่ออกมาหลังวงจรตรวจจับสูงมาก วงจร AGC จะให้กำเนิดสัญญาณไฟกระแสตรงที่มีค่าความต่างศักย์ค่าสูงค่าหนึ่งซึ่งจะถูกป้อนย้อนกลับไปลดกำลังขยายของวงจรขยายความถี่กลาง

ปัญหาที่พบและสำคัญมากในวงจรซูเปอร์เฮตเทอโรไดน์เมื่อความถี่กลางมีค่าต่ำก็คือเรื่องของ อิมเมจฟรีเควนซี่ (Image Frequency) ซึ่งมีลักษณะเป็นความถี่ที่อยู่ใกล้เคียงกับความถี่ที่ต้องการแต่อยู่สูงขึ้นไปสองเท่าของความถี่กลาง IF และอยู่ต่ำลงมากกว่าความถี่ที่ต้องการสองเท่า เมื่อความถี่อิมเมจฟรีเควนซี่เข้ามาในวงจรมิกเซอร์และได้ผลลัพธ์ของสัญญาณความถี่กลางที่มีค่าความถี่เช่นเดียวกันกับสัญญาณจริง ทำให้สัญญาณที่ถูกเลือกมาผิดค่าเกิดเป็นสัญญาณรบกวนกับสัญญาณที่ต้องการ สัญญาณอิมเมจอาจเกิดได้ในกรณีที่แถบความถี่มีการใช้งานอย่างหนาแน่น สัญญาณอีกช่องหนึ่งอาจเข้ามากวนสัญญาณช่องที่ต้องการก็ได้

วิธีการแก้ปัญหาเบื้องต้นก็อาจใช้วงจรจูนเพื่อเลือกเอาเฉพาะสัญญาณความถี่ที่ต้องการเข้ามาในเครื่องรับเท่านั้น และกำจัดสัญญาณอิมเมจออกไป แต่การแก้ไขดังกล่าวก็ไม่สามารถทำได้ในวงจรเครื่องรับที่ต้องการใช้กับความถี่ในช่วงกว้าง วิธีการที่สองที่ใช้ในการแก้ปัญหาก็คือเพิ่มค่าความถี่กลาง IF ให้มากขึ้นจนอิมเมจฟรีเควนซี่อยู่ห่างมากจนเลยออกนอกวงจรจูนไป แต่เมื่อค่าความถี่กลางสูงขึ้นก็จะทำให้ออกแบบวงจรยากขึ้น ดังนั้นการออกแบบวงจรซูเปอร์เฮตเทอโรไดน์ต้องออกแบบให้ความถี่กลางมีค่ามากที่สุดเพื่อลดผลของอิมเมจฟรีเควนซี่ และในขณะเดียวกันต้องทำให้มีค่าน้อยที่สุดเพื่อที่จะให้ออกแบบวงจรได้ง่ายขึ้นที่ความถี่ต่ำ วิธีแก้

ปัญหาสัญญาณอิมเมจสุดท้ายที่นิยมก็คือใช้วงจรแปลงความถี่สองครั้งที่เรียกว่า ดูออลคอนเวอร์ชั่นซูเปอร์เฮตเทอโรไดน์ (Dual Conversion Superheterrodyne Receiver) ซึ่งมีการแปลงความถี่กลาง 2 ชุดด้วยวงจรมิกเซอร์ 2 ชุด ชุดแรกมี LO ที่สามารถปรับค่าได้ ส่วน LO ชุดที่สองคงที่เพื่อปรับค่าได้เล็กน้อย มิกเซอร์ชุดแรกจะแปลงให้สัญญาณมาอยู่ในความถี่กลางค่าสูง โดยจะช่วยในการลดปรากฏการณ์อิมเมจฟรีเควนซี่ ส่วนมิกเซอร์ชุดที่สองจะแปลงสัญญาณ IF ชุดแรกให้ต่ำลงเป็นสัญญาณ IF ความถี่ที่สองซึ่งให้ค่าซีเล็กติวิตี้ที่ดีกว่าวงจรดูออลคอนเวอร์ชั่น (Dual conversion) มักใช้ในวงจรเครื่องรับความถี่คลื่นสั้น (Short Wave Receiver), เครื่องรับคลื่น VHF, UHF และไมโครเวฟ

เครื่องรับแบบ AM จะมี IF ที่	455 kHz, 30 MHz, 3385 kHz, 9MHz
เครื่องรับแบบ FM จะมี IF ที่	10.7 MHz
เครื่องรับโทรทัศน์จะมี IF ที่	40-50 MHz
เครื่องรับเรดาห์จะมี IF ที่	60 MHz
เครื่องรับดาวเทียมจะมี IF ที่	70 MHz, 140 MHz

บทที่ 3

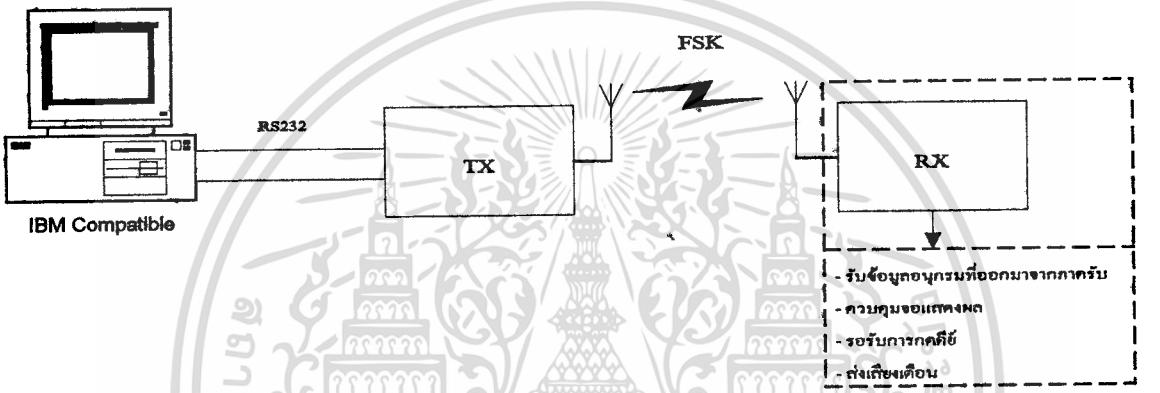
การออกแบบและการสร้าง

หลักการของระบบวิทยุติดตามตัว

ในโครงงานนี้ระบบวิทยุติดตามตัว ประกอบด้วย 3 ส่วนหลักคือ

- ศูนย์รับข้อมูลและจัดเก็บข้อมูลโดยใช้เครื่องคอมพิวเตอร์
- เครื่องส่งข้อมูล
- เครื่องรับข้อมูลและแสดงผลข้อมูล

ซึ่งมีบล็อกไดอะแกรมแสดงโครงสร้างของระบบวิทยุติดตามตัวดังรูปที่ 3.1

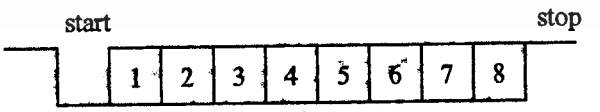


รูปที่ 3.1 แสดงโครงสร้างของระบบวิทยุติดตามตัว

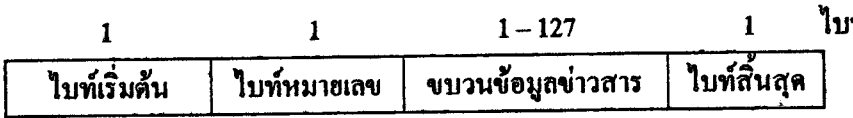
รายละเอียดหน้าที่และการทำงานของส่วนต่าง ๆ ของโครงสร้างระบบวิทยุติดตามตัวมีดังนี้

3.1 ศูนย์รับข้อมูลและจัดเก็บข้อมูล

เครื่องคอมพิวเตอร์ที่รับข้อมูลและจัดเก็บข้อมูล จะทำหน้าที่ในการจัดการเกี่ยวกับการรอรับข้อมูลจากผู้เรียก การเก็บข้อมูล และการจัดส่งข้อมูล สำหรับข้อมูลที่จะส่งออกไปจากเครื่องคอมพิวเตอร์นั้น จะส่งออกทางพอร์ทอนุกรมตามมาตรฐาน RS - 232 ซึ่งรูปแบบการจัดเรียงข้อมูลอนุกรมที่ส่งออกจากเครื่องคอมพิวเตอร์ได้แสดงไว้ดังรูปที่ 3.2



a) รูปแบบของข้อมูลแต่ละไบท์(Byte) ต่อ 1 อักขระ



b) รูปแบบของข้อมูลและจำนวนไบท์ (Byte) ต่อ 1 ชุดข้อมูล

รูปที่ 3.2 รูปแบบข้อมูลที่ส่งออกมาจากเครื่องคอมพิวเตอร์

ภายใน 1 ชุดข้อมูล จะประกอบด้วยจำนวนไบท์ ตั้งแต่ 4 - 130 ไบท์ โดยจะมีจำนวนอักขระ (Character) ที่จะส่งได้ไม่เกิน 127 อักขระสามารถเปลี่ยนแปลงจำนวนไบท์ข้อมูลได้ตามต้องการ เนื่องจากภายในตัวไมโครคอนโทรลเลอร์ AT89C51 จะมีหน่วยความจำสำหรับเก็บโปรแกรมได้ 4 กิโลไบท์ โดยแต่ละอักขระจะเป็นรหัสแอสกี้(ASCII) ที่จะส่งออกมาจากพอร์ทอนุกรม RS - 232 ชุดข้อมูลจะถูกแบ่งออกเป็นส่วนต่างๆ เรียงตามลำดับต่อไปนี้

– ไบท์เริ่มต้น (Start Byte) จะเป็นไบท์แรกที่ส่งออกมาเพื่อให้เครื่องรับรู้ว่าเครื่องส่งได้เริ่มส่งข้อมูลออกมาแล้ว ซึ่งจะต้องมีการกำหนดรูปแบบของไบท์ที่ตายตัวและจะต้องตรงกันระหว่างเครื่องส่งและเครื่องรับ

– แอดเดรสไบท์ (Address Byte) เป็นตัวบอกแอดเดรสหรือหมายเลขของเครื่องรับวิทยุติดตามตัว ซึ่งเครื่องรับแต่ละเครื่องจะมีหมายเลขที่ต่างกัน โดยจะมีการตั้งค่ากำหนดหมายเลขเครื่องของแต่ละเครื่องไว้ภายในส่วนโปรแกรมควบคุมของตัวไมโครคอนโทรลเลอร์เบอร์ AT89C51 ที่เครื่องรับ เมื่อมีการส่งข้อมูลมาเครื่องรับจะทำการรับข้อมูลเฉพาะที่มีแอดเดรสไบท์ตรงกับเครื่องของตัวเองเท่านั้น

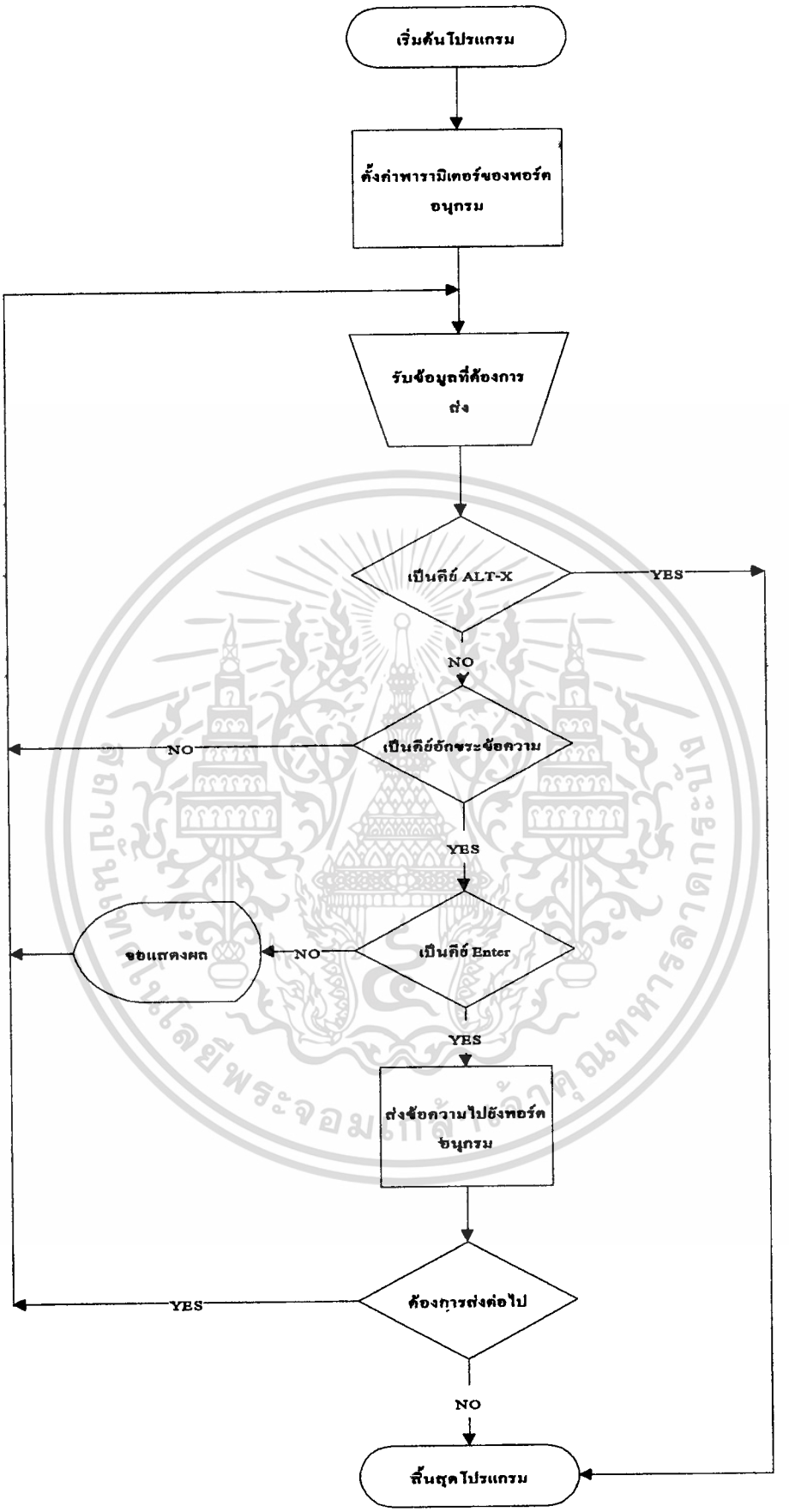
– ข้อมูลข่าวสาร (Information Byte) จะเป็นชุดอักขระรหัสแอสกี้ของข้อความที่ต้องการจะส่งจากผู้เรียกไปยังเครื่องรับวิทยุติดตามตัวของผู้รับ ซึ่งจะมีความยาวได้ตั้งแต่ 1 - 127 อักขระ

– ไบท์สิ้นสุด (Stop Byte) เป็นไบท์สุดท้ายของชุดข้อมูลที่ส่งมา เพื่อบอกให้เครื่องรับว่าข้อมูลที่จัดส่งมานั้นหมดแล้วเป็นการสิ้นสุดขบวนการรับส่งข้อมูล

3.2 ส่วนของภาคส่ง

จะใช้คอมพิวเตอร์เป็นศูนย์กลางเพื่อควบคุมการส่งข้อมูลโดยจะรับการสั่งงานจากผู้ใช้และส่งข้อความป้อนโดยการป้อนข้อมูลจากแป้นพิมพ์ (keyboard) การเชื่อมต่อระหว่างเครื่องส่งและคอมพิวเตอร์จะใช้พอร์ท RS -232 ซึ่งจะต้องใช้ IC เบอร์ MAX 232 ให้เป็นระดับเดียวกันกับที่ใช้ในระบบไมโครคอนโทรลเลอร์ในภาครับสัญญาณ

โปรแกรมที่ใช้ควบคุมการส่งข้อความจะถูกสร้างขึ้นมาจากโปรแกรมภาษา C โดยจะรับข้อมูลหมายเลขเครื่องและรับส่งข้อความที่จะส่งโดยส่งได้ทีละ 127 ตัวอักษร โดยสามารถมีเครื่องถูกข่ายได้ถึง 255 เครื่องหรือมากกว่านั้น ส่วนขั้นตอนการใช้งานโปรแกรมควบคุมการส่งข้อความสามารถดูได้จากโฟลวชาร์ต (Flow Chart) ข้างล่างนี้



รูปที่ 3.3 โฟลวชาร์ตแสดงโปรแกรมควบคุมการส่งข้อความ

ในส่วนของเครื่องส่งที่ใช้จะมีวงจรตามรูปวงจรแผ่นที่ 4 ในภาคผนวก เมื่อต่อวงจรตามรูปเสร็จเรียบร้อยแล้ว จะต้องปรับแต่งให้เครื่องรับสามารถรับสัญญาณจากเครื่องส่งนี้ได้ วิธีปรับแต่งความถี่ทำได้โดย จูนเครื่องรับวิทยุ FM ไปที่ความถี่ประมาณ 108 MHz จะเป็นตำแหน่งที่ว่างจากสถานีส่งอื่นๆ แล้วปรับ TC_1 ซึ่งต่ออยู่กับ L_1 เป็นวงจรเรโซแนนซ์ให้มีความถี่เรโซแนนซ์เท่ากับ 108 MHz โดยสังเกตจากเสียงรบกวนที่เครื่องรับเงียบหายไป ถ้าปรับ TC_1 แล้วยังไม่ได้ให้ทดลองดึง L_1 ให้ยืดออกหรือบีบ L_1 ให้หดลงไปจนได้ความถี่เรโซแนนซ์ที่ต้องการ ทำการทดลองป้อนสัญญาณเสียงจากแหล่งกำเนิดใดๆเข้าไปถ้าเครื่องรับสามารถรับได้ก็เป็นอันว่าสามารถปรับความถี่ที่ต้องการส่งออกอากาศตรงกับเครื่องรับแล้ว

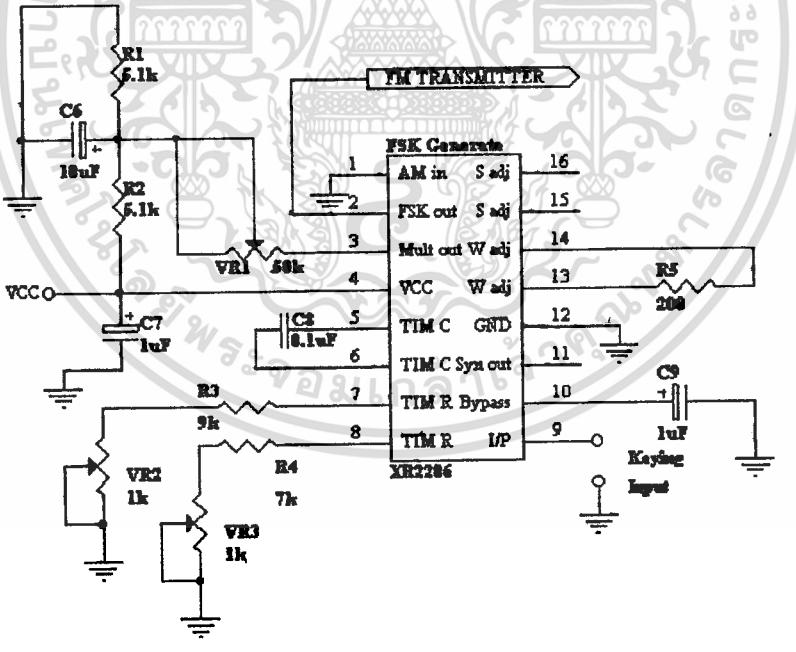
3.2.1 วงจรมอดูเลทแบบ FSK

วงจรที่ใช้สำหรับการมอดูเลทแบบเอฟเอสเคจะใช้ไอซีเบอร์เดียวกับวงจรมอดูเลทแบบเอฟเอ็ม คือ XR-2206 ซึ่งสามารถใช้เป็นวงจรมอดูเลทแบบเอฟเอสเคด้วย การออกแบบวงจรมอดูเลทแบบเอฟเอสเค สำหรับการทดลองนี้ จะเลือกค่าอัตราบอด, ความถี่มาร์ค และความถี่สเปซตามมาตรฐาน V 21 ของ CCITT คือ อัตราบอดเท่ากับ 300 บิตต่อวินาที ความถี่มาร์ค (f_1) เท่ากับ 1,070 เฮิรตซ์ ความถี่สเปซ (f_2) เท่ากับ 1,270 เฮิรตซ์

รูปที่ 3.4 แสดงวงจรที่ใช้สำหรับการมอดูเลทแบบเอฟเอสเค ซึ่งมีอุปกรณ์ที่ไม่ได้กำหนดค่าไว้ คือตัวต้านทาน R_1, R_2 และตัวเก็บประจุ C ซึ่งเราสามารถคำนวณหาค่า R_1, R_2 และ C ได้จาก

$$f_1 = 1/CR_1 \qquad (3.1)$$

$$f_2 = 1/CR_2 \qquad (3.2)$$



รูปที่ 3.4 วงจรมอดูเลทแบบ เอฟ เอส เค

ดังนั้นจากค่า f_1 และ f_2 ที่เลือกไว้เราสามารถคำนวณหาค่า R_1 และ R_2 ได้โดยเลือกใช้ค่า

$$C = 0.1\mu F$$

$$R_1 = 1/Cf_1 = \frac{1}{(0.1\mu F)(1,070)} = 9.345k\Omega$$

$$R_2 = 1/Cf_2 = \frac{1}{(0.1\mu F)(1,270)} = 7.874k\Omega$$

ส่วนวงจรที่ใช้งานจริงซึ่งจะต่อร่วมกับ MAX 232 เพื่อแปลงสัญญาณตามมาตรฐาน RS-232 ให้เป็นสัญญาณระดับต่ำก่อนที่จะผ่านเข้าเครื่องส่งจะแสดงในรูปวงจรแผ่นที่ 2 ในภาคผนวก

3.2.2 การสร้างวงจรมอดูเลทแบบ FSK

จากค่า R_1 และ R_2 ที่คำนวณได้ จะเลือกใช้ตัวต้านทานแบบค่าคงที่ต่ออนุกรมกับตัวต้านทานแบบปรับค่าได้ โดยควรเลือกใช้ชนิดที่ปรับค่าแบบละเอียดเพื่อให้มีค่าความต้านทานใกล้เคียงกับค่าที่คำนวณมากที่สุด ส่วนตัวเก็บประจุที่ใช้ในวงจรชนิดไม่มีขั้วควรจะเลือกใช้ตัวเก็บประจุชนิดไมล่าร์

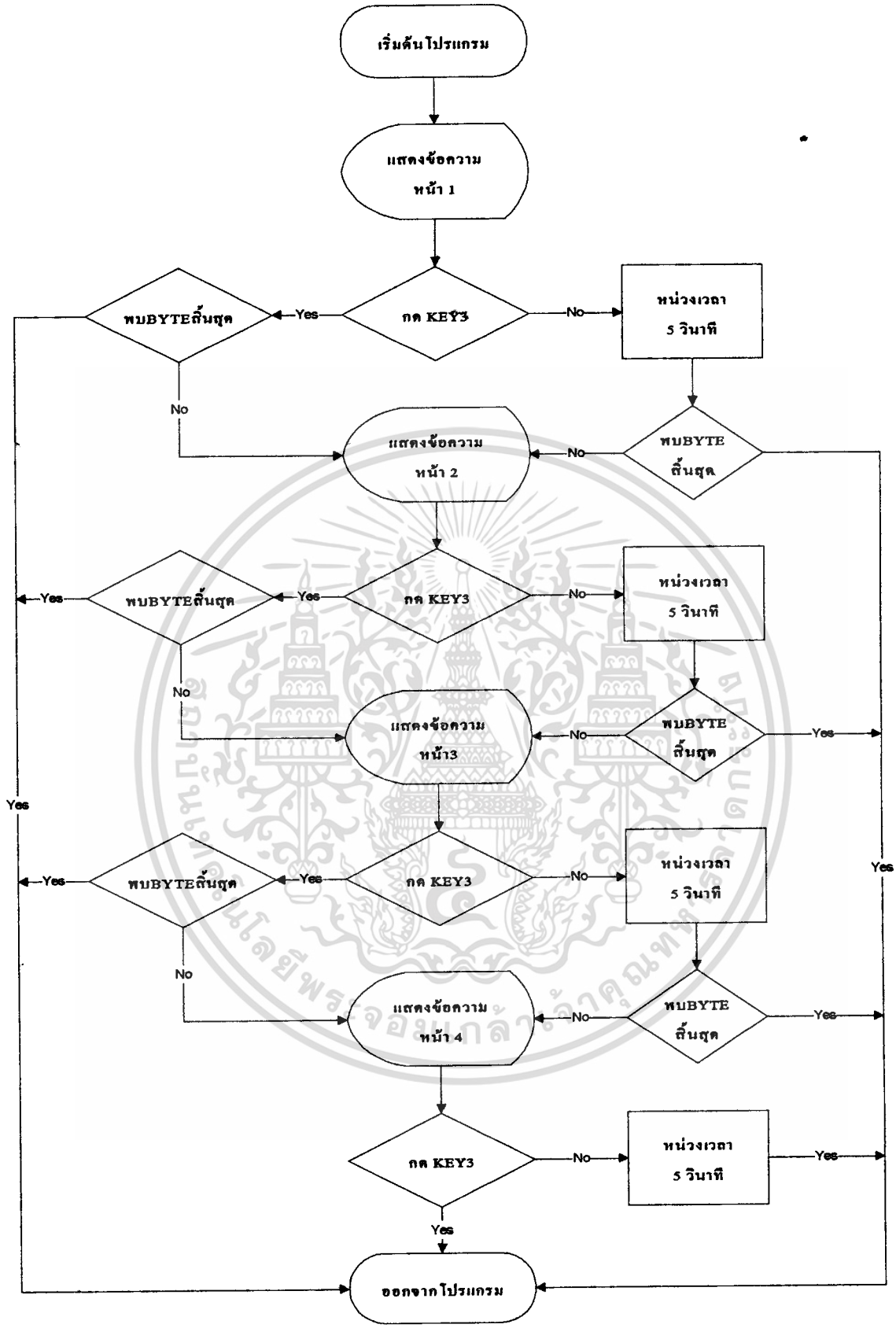
ในการสร้างวงจรจะต้องทำการปรับแต่งวงจรให้ได้ความถี่มาร์คและความถี่สเปซตรงกับค่าที่ได้กำหนดไว้ในการออกแบบ ซึ่งเราสามารถทำการปรับแต่งได้โดยกรณีปรับความถี่มาร์คให้ปลด R_2 ออกจากวงจรและป้อนสัญญาณดิจิตอลเข้าที่อินพุต จากนั้นวัดสัญญาณที่เอาท์พุตซึ่งจะต้องเป็นสัญญาณคลื่นไซน์ที่มีความถี่มาร์ค โดยสามารถที่จะปรับความถี่ให้เท่ากับความถี่มาร์คได้ที่ R_{1b} กรณีปรับความถี่สเปซปลด R_1 ออกและทำเช่นเดียวกับการปรับความถี่มาร์ค ซึ่งความถี่สเปซนี้จะปรับได้ที่ตัวต้านทาน R_{2b}

3.3 อุปกรณ์รับและแสดงผล

โครงงานนี้จะใช้ไมโครคอนโทรลเลอร์ของบริษัท ATMEL เบอร์ AT89C51 เป็นตัวควบคุมการทำงานทั้งหมด ซึ่งโครงสร้างเหมือนกับไมโครคอนโทรลเลอร์ 8051 แต่จะมีหน่วยความจำที่โปรแกรมได้ (Flash Memory)ขนาด 4 กิโลไบท์ สามารถโปรแกรมและลบได้ 1,000 ครั้ง และสามารถป้องกันการอ่านข้อมูลโปรแกรมได้

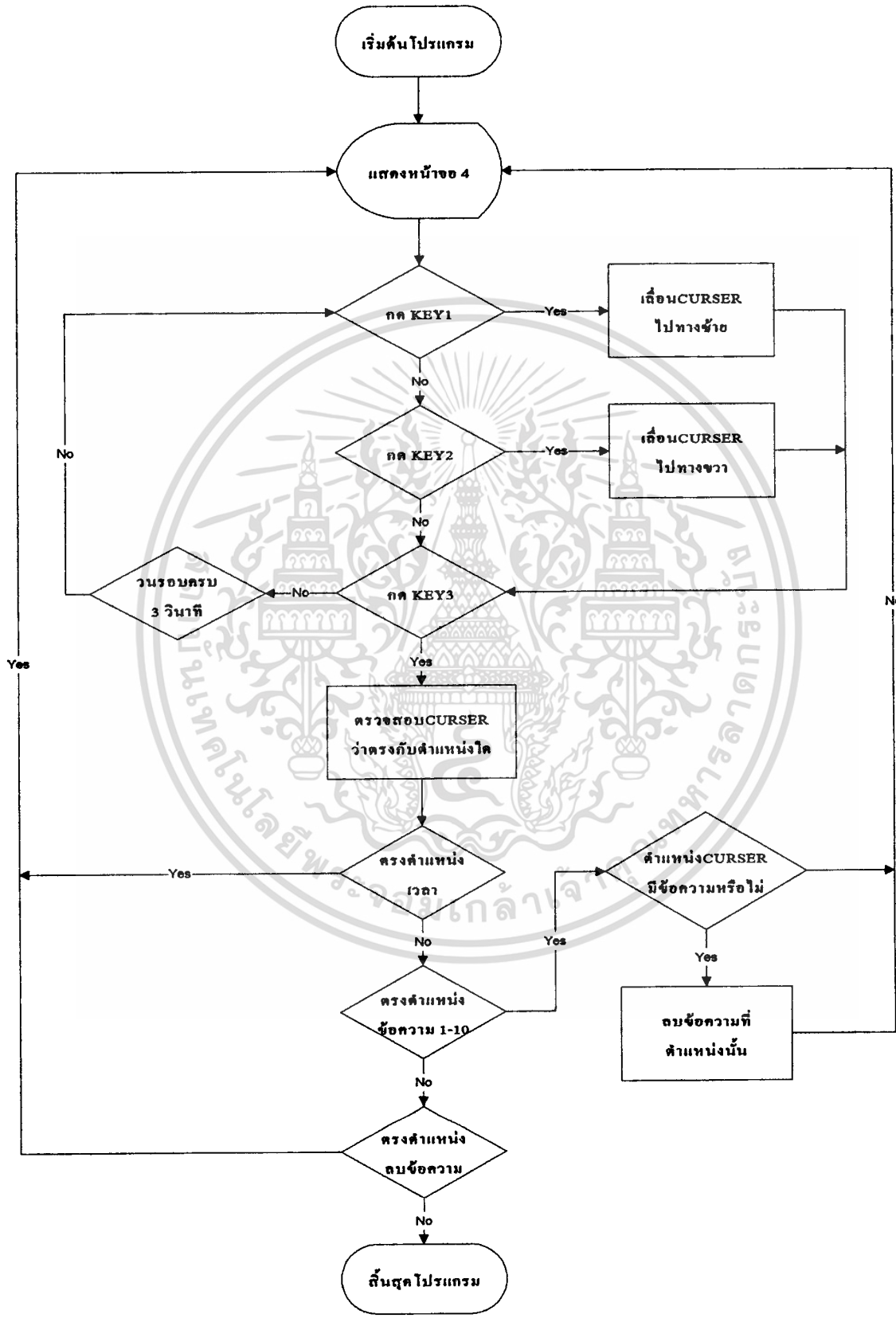
ส่วนแสดงผลจะใช้ LCD Module (Liquid Crystal Display Module) ที่เป็นแบบ dot matrix ขนาด 16 ตัวอักษร 2 บรรทัด โดยเชื่อมต่อกับไมโครคอนโทรลเลอร์แบบ 4 บิท ซึ่งทำให้ประหยัดพอร์ทที่ใช้งาน สามารถเขียนข้อมูลที่เป็นตัวอักษรหรือรูปที่เราต้องการเข้าไปใหม่ได้ 8 ตัวอักษรซึ่งและตัวอักษรจะแสดงด้วย dot matrix ขนาด 5 * 7 dot ในส่วนแสดงผลของเครื่องรับนั้นจะต้องใช้โปรแกรมในการควบคุมให้แสดงผลออกมาตามต้องการ โดยจะแบ่งส่วนของโปรแกรมออกดังนี้

– โปรแกรมการแสดงข้อความ ใช้เพื่อควบคุมเมื่อมีการกดคีย์ต่างๆเพื่อดูข้อความหรือต้องการใช้ฟังก์ชั่นต่างๆของตัวเครื่องรับ ซึ่งโฟลวชาร์ตของโปรแกรมส่วนนี้แสดงได้ดังนี้



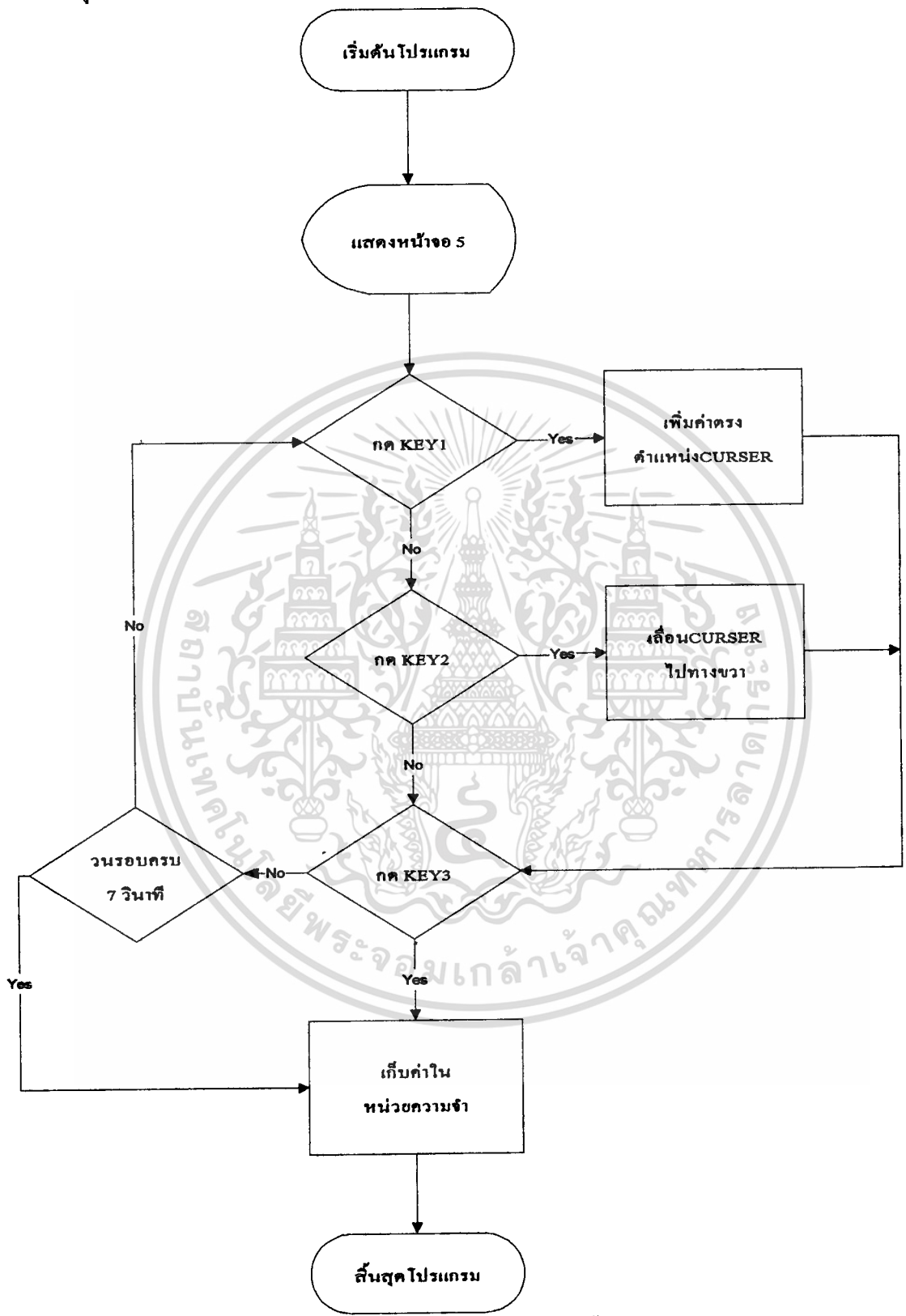
รูปที่ 3.5 โฟลวชาร์ตของโปรแกรมควบคุมการแสดงผล

– โปรแกรมลบข้อความ เป็นโปรแกรมควบคุมการลบข้อความโดยใช้การตรวจสอบคีย์ทั้ง 3 คีย์ เมื่อมีการกดคีย์เอนเทอร์ก็จะลบข้อความที่ตรงกับตำแหน่งของเคอร์เซอร์ โปรแกรมส่วนนี้สามารถแสดงกระบวนการทำงานได้ดังโฟลวชาร์ตข้างล่างนี้



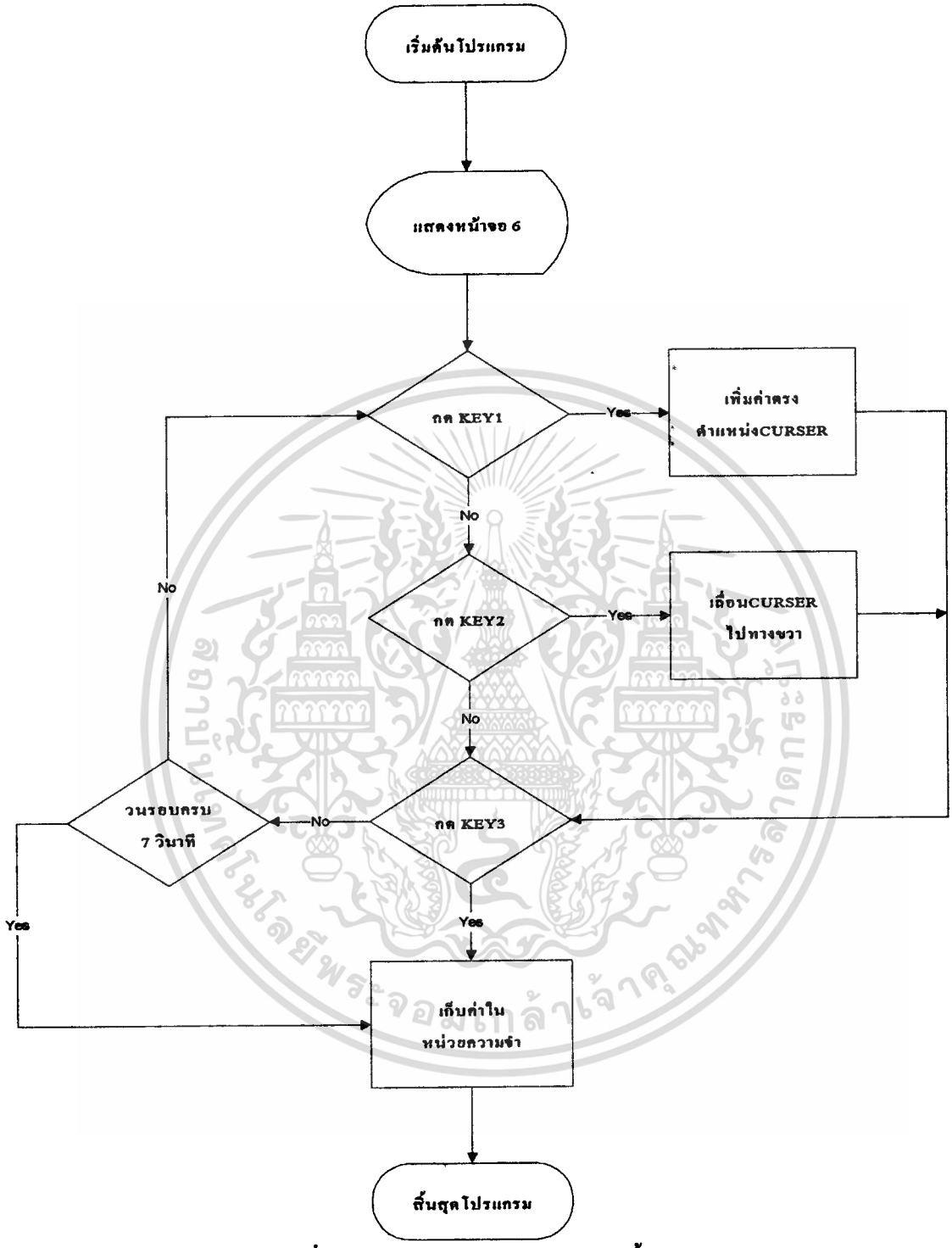
รูปที่ 3.6 โฟลวชาร์ตของโปรแกรมการลบข้อความ

– โปรแกรมตั้งวัน-เวลา เป็นโปรแกรมที่ใช้ตั้งวัน-เวลาที่แสดงบนหน้าจอของเพจเจอร์ โดยใช้ RTC เป็นตัวควบคุมการแสดงผลให้วัน-เวลาตรงกับฐานเวลาจริง โฟลวชาร์ตของโปรแกรมนี้แสดงได้ดังนี้



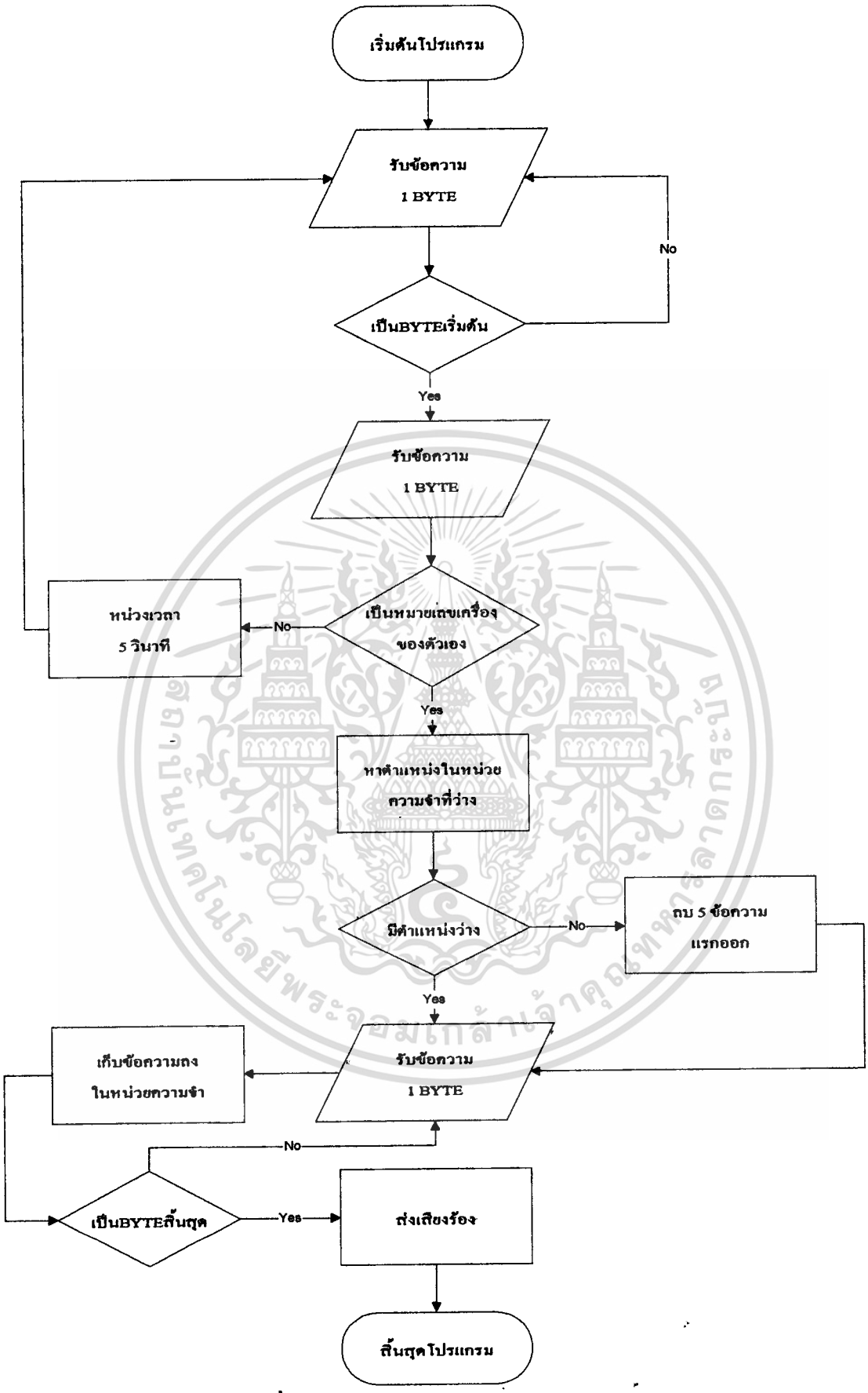
รูปที่ 3.7 โฟลวชาร์ตของโปรแกรมการตั้งวัน-เวลา

– โปรแกรมตั้งปลุก ใช้ตั้งเวลาเพื่อให้เป็นนาฬิกาปลุกได้ โฟลวชาร์ตของโปรแกรมการตั้งปลุกแสดงได้ดังนี้



รูปที่ 3.8 โฟลวชาร์ตของโปรแกรมการตั้งปลุก

– โปรแกรมรับข้อความ เมื่อเครื่องรับรับข้อความเข้ามาแล้วจะทำการตรวจสอบดูว่าเป็นไบท์เริ่มต้นหรือไม่ถ้าไม่ใช่ก็จะเริ่มต้นรับข้อความโดยตรวจสอบหมายเลขเครื่องของตัวเองก่อน ถ้าไม่ใช่ก็จะไม่รับ และโปรแกรมนี้สามารถลบข้อความได้โดยอัตโนมัติเมื่อมีข้อความเข้ามาใหม่ในขณะที่ข้อความเต็มอยู่ โฟลวชาร์ตของโปรแกรมนี้แสดงได้ดังนี้



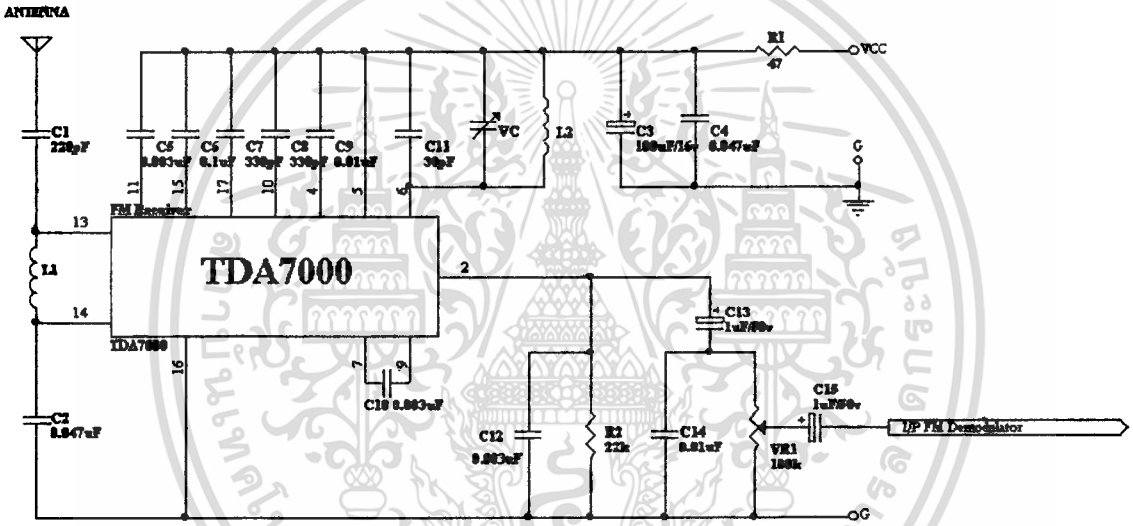
รูปที่ 3.9 โฟลวชาร์ตของโปรแกรมรับข้อความ

ส่วนที่เป็นเวลานาฬิกา (Real Time Clock)จะใช้ IC ของบริษัท Dallus Semiconductor เบอร์ DS 1202 ซึ่งเป็น IC ขนาดเล็กทำให้ประหยัดพื้นที่ สามารถบอกเวลา วันที่ เดือน ปี และยังมี แรม(RAM)ขนาด 24 ไบท์ ใช้งานได้อิสระ การอ่านหรือเขียนข้อมูลจะอ่านหรือเขียนแบบอนุกรม ทำให้ประหยัดพอร์ทและป้องกันข้อมูลผิดพลาด

ส่วนที่ใช้เก็บข้อมูลจะใช้สเตติกแรม(Static RAM) เบอร์ HM 6116 ซึ่งเป็นแรมขนาด 2 กิโลไบท์ โดยแบ่งหน่วยความจำให้สามารถเก็บข้อความได้ 10 ข้อความ ข้อความละ 128 ตัวอักษรและสามารถเพิ่มขนาดได้โดยไม่เกิน 2 กิโลไบท์ สามารถเก็บข้อมูลไว้ได้แม้ไฟดับโดยใช้ IC ของบริษัท MAXIM เบอร์ 691

3.3.1 หลักการทำงานของวงจรในระบบรับข้อมูล

ในภาครับสัญญาณจะใช้วงจรรับเอฟเอ็มธรรมดาเป็นตัวรับสัญญาณและผ่านวงจรดีมอดูเลทแบบ FSK เพื่อดีเทคสัญญาณคลื่นพาห์ออกแล้วนำเอาสัญญาณข้อมูลที่ได้มาแสดงผลโดยผ่านไมโครคอนโทรลเลอร์ MCS-51 เป็นตัวควบคุมการแสดงผล



รูปที่ 3.10 วงจรจูนเนอร์เอฟเอ็ม

การทำงานของวงจร

วงจรเครื่องรับวิทยุซึ่งมีหน้าที่รับสัญญาณจากสถานีวิทยุซึ่งส่งคลื่นอากาศ ในวงจรนี้ได้ใช้ไอซีเบอร์ TDA 7000 ทำหน้าที่เป็นภาครับของวิทยุเอฟเอ็ม ภายในของไอซีเบอร์นี้ประกอบด้วยภาคโลคอลออสซิลเลเตอร์ มิกเซอร์ ภาคขยายไอเอฟควอดดราเจอร์ ดีเทคเตอร์ ภาคมิวติ้ง จากรูป สัญญาณความถี่วิทยุจากเสาอากาศจะผ่าน C_1 ไปเข้าวงจรแบนด์พาสฟิลเตอร์ L_1 เข้าขา 13 และขา 14 และ C_2 จะทำหน้าที่คัปปลิ้ง C_5 ถึง C_{10} จะทำหน้าที่ฟิลเตอร์ให้กับวงจรขยายและวงจรดีมอดูเลเตอร์ขา 6 ของ IC จะเป็นขาสำหรับจูนหาสถานีซึ่งมี C_{11}, L_2 และ VC จะทำหน้าที่ปรับแต่งและจูนหาสถานีตามต้องการ ขา 16 จะต่อรับไฟลบ ขา 5 จะต่อรับไฟบวก โดยผ่านทาง R_1 ส่วน C_3 จะทำหน้าที่ฟิลเตอร์ C_4 จะทำหน้าที่บายพาสลงกราวด์ ขา 2 จะเป็นขาเอาท์พุต R_2 และ C_{12} จะทำหน้าที่ดีเอ็มฟาซิส (deemphasis) ของสัญญาณเสียงผ่าน C_{13} ผ่าน VR_1 ซึ่งจะทำหน้าที่เร่งและหรี่เสียงผ่าน C_{15} มาที่จุดเอาท์พุตซึ่งที่จุดเอาท์พุตนี้จะต่อไปเข้าวงจรขยายเสียง

3.3.2 การสร้างวงจรดีมอดูเลเตอร์สัญญาณ FSK

อุปกรณ์ที่ไม่ทราบค่าสามารถคำนวณค่าได้จากสูตรต่อไปนี้

$$C_0 = 1/Rf_0 \quad (3.2)$$

โดยที่ f_0 คือความถี่กลางของ VCO ซึ่งหาได้จาก

$$f_0 = (f_1+f_2)/2 \quad (3.4)$$

$$R_1 = R_0[\, f_0/(f_2-f_1)] \quad (3.5)$$

$$C_1 = C_0/4 \quad (3.6)$$

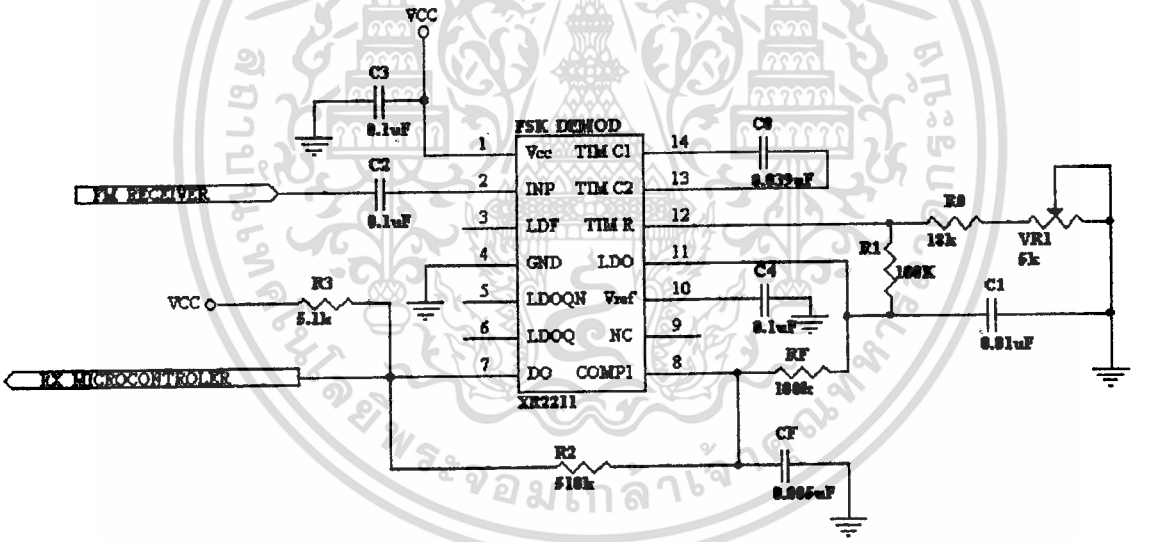
สมการ (3.4) กรณีลูปแคมปิ้งมีค่าเท่ากับ 0.5

$$C_F \approx 3/(\text{Baudrate})\ \mu F \quad (3.7)$$

สมการที่ (3.6) สำหรับค่า $R_F = 100$ kΩ และค่า $R_B = 510$ kΩ

ในดาต้าชีต (data sheet) ของ XR-2211 ได้กำหนดค่าอุปกรณ์เหล่านี้ สำหรับอัตราบอด 300 bps, f_1 เท่ากับ 1,070 เฮิรตซ์ และ $f_2 = 1,270$ เฮิรตซ์ ไว้แล้ว ซึ่งได้ค่าต่างๆดังนี้

$R_0 = 18$kΩ, $R_1 = 100$kΩ, $C_0 = 0.039\mu$F, $C_1 = 0.01\mu$F, $C_F = 0.005\mu$F ส่วนค่า R_F ใช้ค่า 100kΩ เพื่อให้สอดคล้องกับสมการที่ 3.6



รูปที่ 3.11 วงจรดีมอดูเลทสัญญาณ เอฟ เอส เค

จากการออกแบบจะได้วงจรสำหรับการดีมอดูเลทสัญญาณ เอฟ เอส เค ดังรูปที่ 3.11 โดยค่า R_0 จะเลือกใช้ค่าความต้านทานแบบลงที่อนุกรมกับตัวต้านทานแบบปรับค่าได้เพื่อให้เราสามารถที่จะปรับค่าความถี่กลางของ VCO ให้ตรงกับค่าที่ต้องการ

บทที่ 4

การทดลองและผลการทดลอง

ในโปรเจกต์ 1 (Project I) ที่ผ่านมาได้ทำการทดลองรับส่งข้อมูลผ่านทางสาย RS-232 โดยยังไม่ส่งออกอากาศ ซึ่งได้ผลดี เครื่องรับสามารถรับข้อมูลได้ถูกต้องแต่อาจจะมีความผิดพลาดบ้าง เพราะในการต่อฮาร์ดแวร์นั้นยังอยู่ในขั้นทดลอง สายบัสซึ่งใช้ต่อ LCD อาจจะมีความยาวเกินไป ทำให้ข้อมูลเดินทางจากบอร์ดไปยัง LCD ผิดพลาด ซึ่งได้ทำการทดลองแก้ไขจุดนี้ดูแล้วได้ผลเป็นที่น่าพอใจคือข้อมูลที่ได้รับไม่ผิดพลาด ส่วนโปรแกรมที่ใช้สำหรับส่งข้อมูลนั้นเขียนจาก BORLAND C++ ซึ่งมีฟังก์ชันในการเซตค่าต่างๆ ดังนี้

- กด F2 ใช้เซตพอร์ตอนุกรมของคอมพิวเตอร์ว่าเป็น COM1 หรือ COM2
- กด F3 ใช้เซตอัตราบอดในการส่งข้อมูล
- กด F4 ใช้กำหนดแอดเดรสของเครื่องลูกข่าย

ซึ่งโปรแกรมนี้จะใช้ส่งข้อความได้ 128 ตัวอักษร

ในส่วนของภาครับนั้นในขั้นการทดลองจะใช้ IC DS5000T เป็นไมโครคอนโทรลเลอร์ซึ่งจะมีโครงสร้างคล้ายกับ 8051 แต่มีแรมภายใน 32 กิโลไบต์ ซึ่งจะเป็นหน่วยความจำโปรแกรม (Program memory) สามารถโปรแกรมได้ไม่จำกัดจำนวนครั้งและจะเก็บข้อมูลไว้ได้แม้ไม่มีไฟเลี้ยงเป็นเวลา 10 ปี โดยจะมีโปรแกรมสำหรับโหลดข้อมูลอยู่ภายในตัวเอง สามารถโหลดโปรแกรมได้โดยตรงจากพอร์ตอนุกรม RS-232 โดยจะต้องปรับระดับลอจิกให้เท่ากันด้วย โดยใช้ IC MAX232 ก่อน เพื่อปรับระดับแรงดันลอจิก "1" และ "0" ให้อยู่ในระดับแรงดันมาตรฐานคือลอจิก "0" จะมีแรงดัน +3V ถึง +25V ระดับลอจิก "1" จะมีแรงดัน –3V ถึง –25V

4.1 ขั้นตอนการทดลองและผลการทดลองของชุดแสดงผล

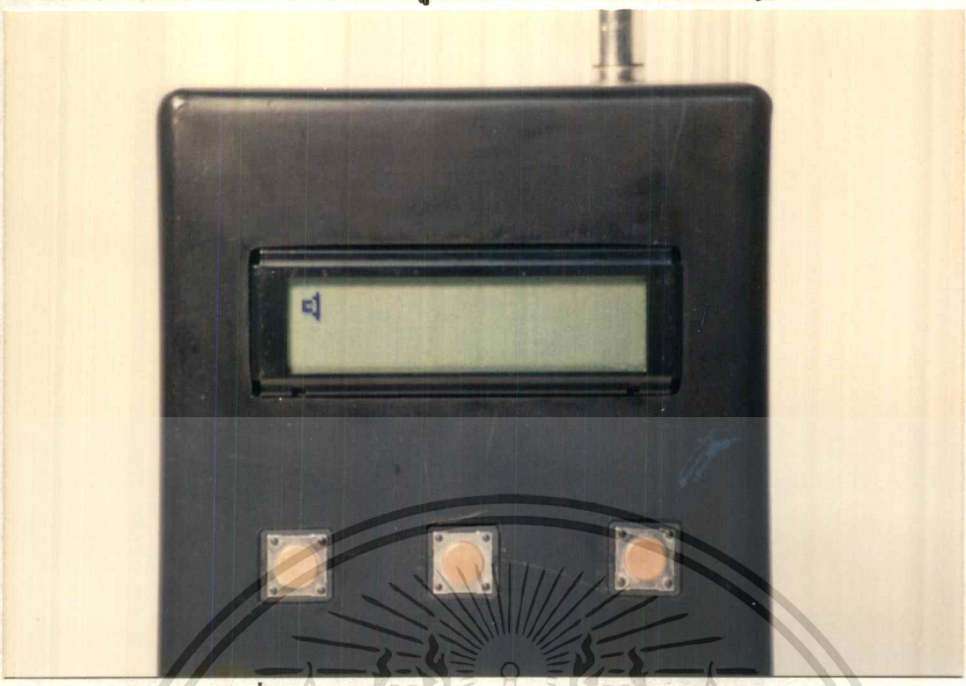
4.1.1 ขั้นตอนการทดลอง

- ต่อวงจรตามรูปวงจรแผ่นที่ 1 ในภาคผนวก
- เขียนโปรแกรมเพื่อเซตสภาวะเริ่มต้นให้กับ LCD module
- เขียนโปรแกรมเพื่อกำหนดตัวอักษรหรือสัญญลักษณ์ที่ต้องการที่นอกเหนือจากตัวอักษรที่มากับ LCD
- เขียนโปรแกรมนาฬิกาโดยใช้ RTC เพื่อให้เวลาเดินเท่ากับฐานเวลาจริง
- เขียนโปรแกรมการตั้งปลุก
- เขียนโปรแกรมเพื่อควบคุมการทำงานทั้งหมดของเครื่องรับ

4.1.2 ผลการทดลอง

จากการทดลองจะเห็นว่าเป็นการเขียนโปรแกรมให้กับตัวเครื่องรับ ฉะนั้นผลการทดลองจะเป็นการแสดงสภาวะต่างๆของ LCD

- เมื่อไม่มีข้อความเข้ามาเครื่องรับจะอยู่ในสภาวะสแตนด์บาย (stand by) ตลอดเวลา



รูปที่ 4.1 แสดงหน้าจอในขณะสแตนด์บายรอข้อความ

- เมื่อกดคีย์เอ็นเทอร์ (enter) จะแสดงเวลา วันที่ และจำนวนข้อความที่มีอยู่

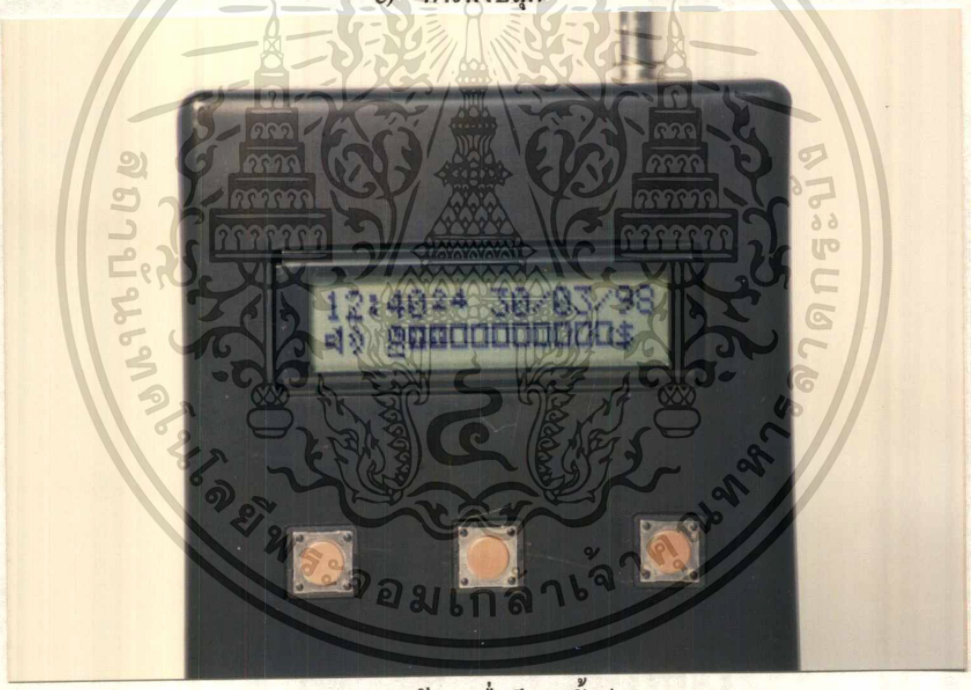
รูปที่ 4.2 แสดงหน้าจอของเมนูหลัก

– สามารถตั้งวันที่ เดือน ปี และเวลาได้จนถึงปี 2020 หรือเพิ่มได้ถึงปี 2100

a) เลือกการตั้งเวลาหรือตั้งปลุก

b) การตั้งเวลา

c) การตั้งปลุก



d) แสดงหน้าจอเมื่อมีการตั้งปลุก

รูปที่ 4.3 แสดงหน้าจอการเลือกการตั้งเวลาและตั้งปลุก

- **สามารถลบข้อความได้โดยตัวผู้ใช้เองหรือลบข้อความอัตโนมัติเมื่อมีการส่งข้อความมาแต่ไม่มีที่ว่างที่จะรับข้อความ**

รูปที่ 4.4 แสดงหน้าจอฟังก์ชันการลบข้อความ

– เมื่อมีข้อความเข้ามาจะมีเสียงดังจากบัสเซอร์และที่หน้าจอจะมีการกระพริบของคลื่นเสียง

รูปที่ 4.5 แสดงหน้าจอเมื่อมีข้อความเข้ามา

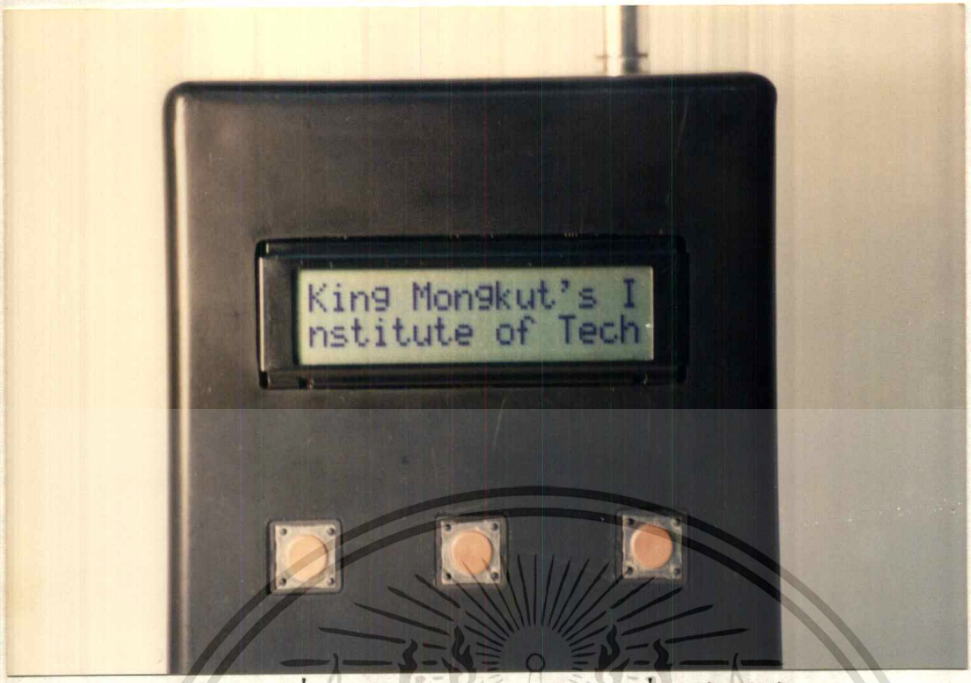
– ที่ตำแหน่งข้อความใดๆที่ไม่มีข้อความเมื่อกดเอ็นเทอร์เข้าไปดูจะมีการแจ้งว่าไม่มีข้อความ

รูปที่ 4.6 แสดงหน้าจอเมื่อ ไม่มีข้อความให้อ่าน

– เมื่ออยู่นอกพื้นที่บริการจะแจ้งว่าขณะนี้อยู่นอกพื้นที่บริการ

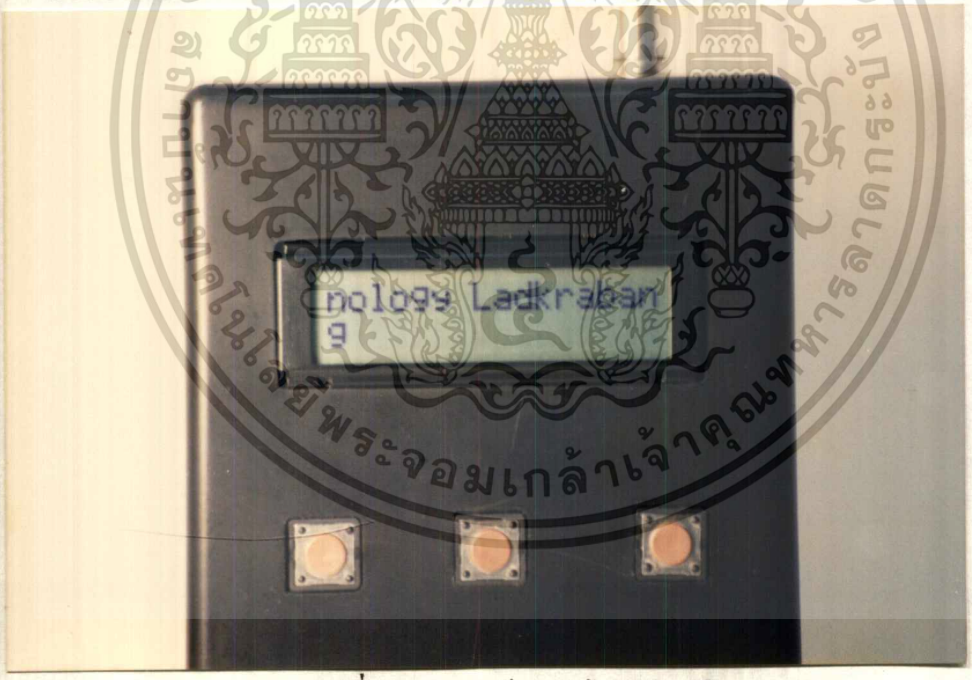
รูปที่ 4.7 แสดงหน้าจอเมื่อเครื่องรับอยู่นอกพื้นที่ให้บริการ

- สามารถรับข้อความได้ 10 ข้อความ ข้อความละ 127 ตัวอักษร



รูปที่ 4.8 แสดงตัวอย่างของข้อความที่รับเข้ามาได้

- สามารถอ่านข้อความหน้าต่อไปได้โดยอัตโนมัติ



รูปที่ 4.9 แสดงหน้าจอถัดไปอัตโนมัติ

- ใช้ปุ่มเพียง 3 ปุ่มในการควบคุมการทำงาน

4.2 ขั้นตอนการทดลองและผลการทดลองของชุดมอดูเลเตอร์

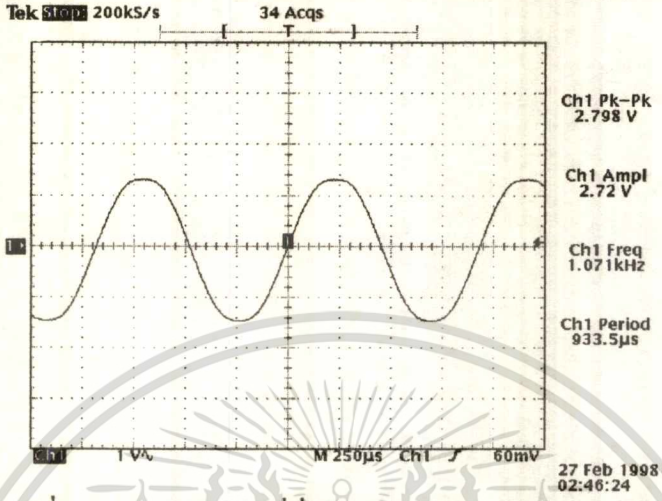
4.2.1 ขั้นตอนการทดลอง

- ต่อวงจร เอฟ เอส เค มอดูเลเตอร์ที่แสดงในรูปวงจรแผ่นที่ 2 ในภาคผนวก
- ป้อนสัญญาณลอจิก "1" ซึ่งมีระดับแรงดัน 5 โวลต์เข้าที่อินพุตของ เอฟ เอส เค มอดูเลเตอร์ จากนั้นทำการวัดสัญญาณที่เอาท์พุต

– ป้อนสัญญาณลอจิก "0" ซึ่งมีระดับแรงดัน 0 โวลต์เข้าที่อินพุตของ เอฟ เอส เค มอดูเลเตอร์ จากนั้นทำการวัดสัญญาณที่เอาท์พุต

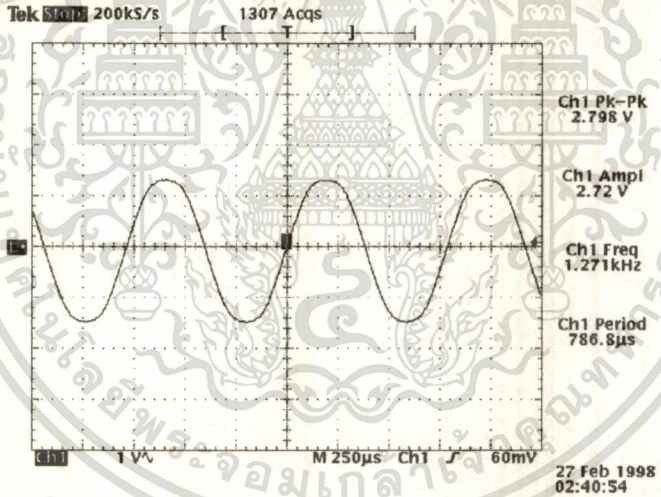
4.2.2 ผลการทดลอง

– เมื่อป้อนสัญญาณที่เป็น "1" จะได้ความถี่มาร์ค = 1071 เฮิรตซ์ ซึ่งมีสัญญาณดังรูปต่อไปนี้



รูปที่ 4.10 สัญญาณความถี่ที่ได้จากการมอดูเลทสัญญาณมาร์ค

– เมื่อป้อนสัญญาณที่เป็น "0" จะได้ความถี่สเปซ = 1271 เฮิรตซ์ ซึ่งมีสัญญาณดังรูปต่อไปนี้



รูปที่ 4.11 สัญญาณความถี่ที่ได้จากการมอดูเลทสัญญาณสเปซ

4.3 ขั้นตอนการทดลองและผลการทดลองของชุดดีมอดูเลเตอร์

4.3.1 ขั้นตอนการทดลอง

– ต่อวงจร เอฟ เอส เค ดีมอดูเลเตอร์ ที่แสดงในรูปวงจรแผ่นที่ 3 ในภาคผนวก

– นำเอาท์พุตของวงจรมอดูเลเตอร์ในการทดลองที่ผ่านมา มาต่อเข้ากับอินพุตของวงจรดีมอดูเลเตอร์ จากนั้นป้อนสัญญาณมาร์คเข้าที่อินพุตของวงจรมอดูเลเตอร์ทำการวัดสัญญาณที่เอาท์พุตของดีมอดูเลเตอร์

– ทำการป้อนสัญญาณสเปซเข้าที่อินพุตของวงจรมอดูเลเตอร์ แล้ววัดสัญญาณที่เอาท์พุตของดีมอดูเลเตอร์

4.3.2 ผลการทดลอง

- เมื่อป้อนสัญญาณมาร์คที่อินพุตของมอดูเลเตอร์จะได้สัญญาณมาร์คที่เอาท์พุตของดีมอดูเลเตอร์
- เมื่อป้อนสัญญาณสเปซที่อินพุตของมอดูเลเตอร์จะได้สัญญาณสเปซที่เอาท์พุตของดีมอดูเลเตอร์

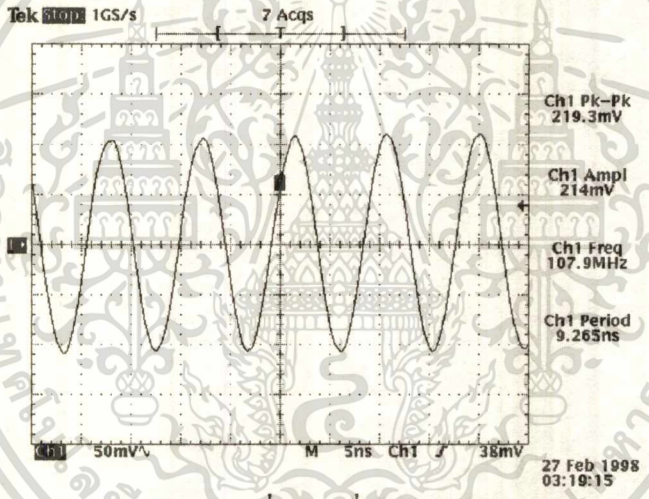
4.4 ขั้นตอนการทดลองและผลการทดลองของชุดเครื่องส่ง

4.4.1 ขั้นตอนการทดลอง

- ต่อวงจรตามรูปที่แสดงในรูปวงจรแผ่นที่ 4 ในภาคผนวก
- จูนเครื่องรับวิทยุ FM ไปที่คลื่น 108 MHz จากนั้นปรับ TC_1 ให้เครื่องรับสามารถรับสัญญาณได้ ถ้าปรับจนสุดแล้วยังรับไม่ได้ก็ให้ลองดึง L_1 เพื่อเปลี่ยนความยาวคู
- นำโปรแกรมที่เป็นตัวกำเนิดเฟรมข้อมูลมาทำการส่ง แล้ววัดสัญญาณที่ได้จากพอร์ตอนุกรมของเครื่องคอมพิวเตอร์ ซึ่งระดับสัญญาณที่ได้ยังคงเป็นระดับสัญญาณตามมาตรฐาน RS-232 อยู่

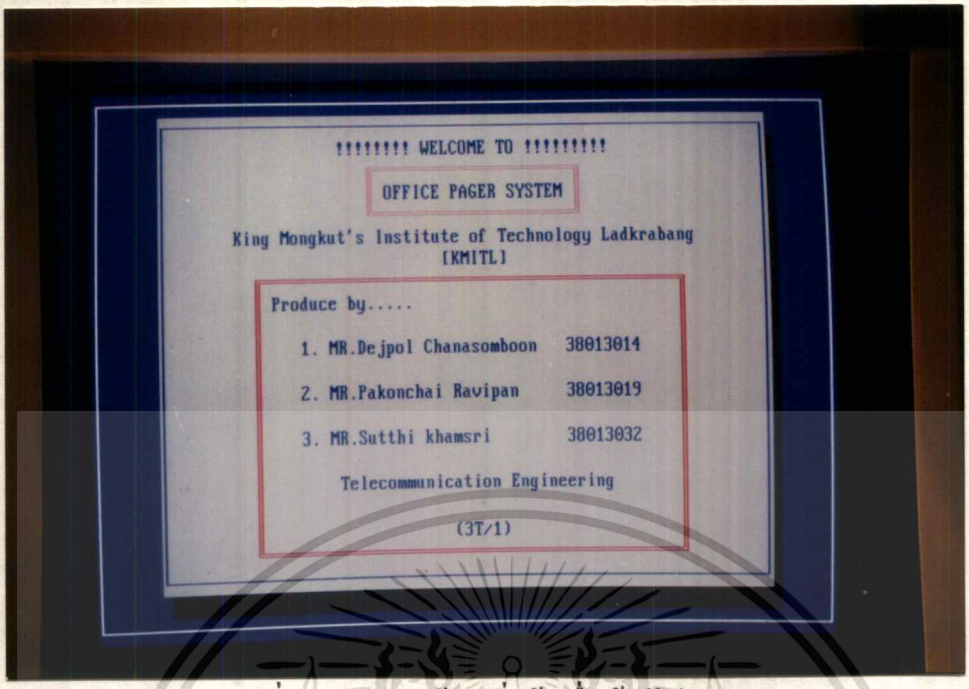
4.4.2 ผลการทดลอง

เครื่องรับวิทยุ FM สามารถรับคลื่นจากเครื่องส่งได้ดีพอสมควร และได้ทดลองส่งเฉพาะความถี่พาหะ (carrier) เพียงอย่างเดียว สามารถวัดสัญญาณที่จุดต่อเสาอากาศได้สัญญาณดังรูป



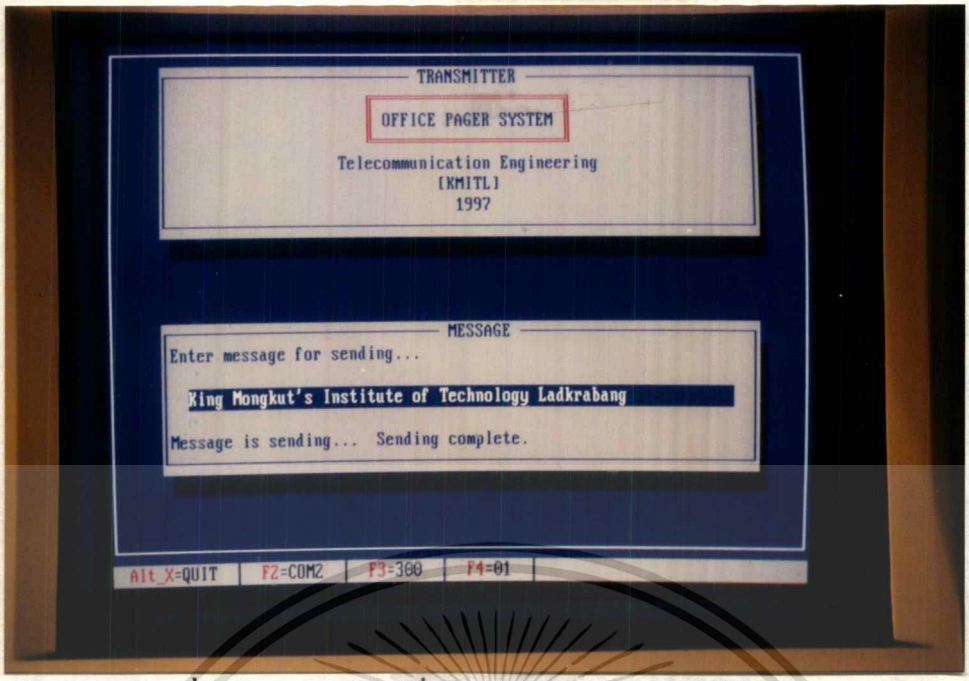
รูปที่ 4.12 แสดงสัญญาณความถี่พาหะเมื่อไม่มีสัญญาณข่าวสารมามอดูเลท

ผลการรันโปรแกรมเครื่องส่งในส่วนของฟังก์ชั่นการใช้งานต่างๆแสดงได้ดังรูป



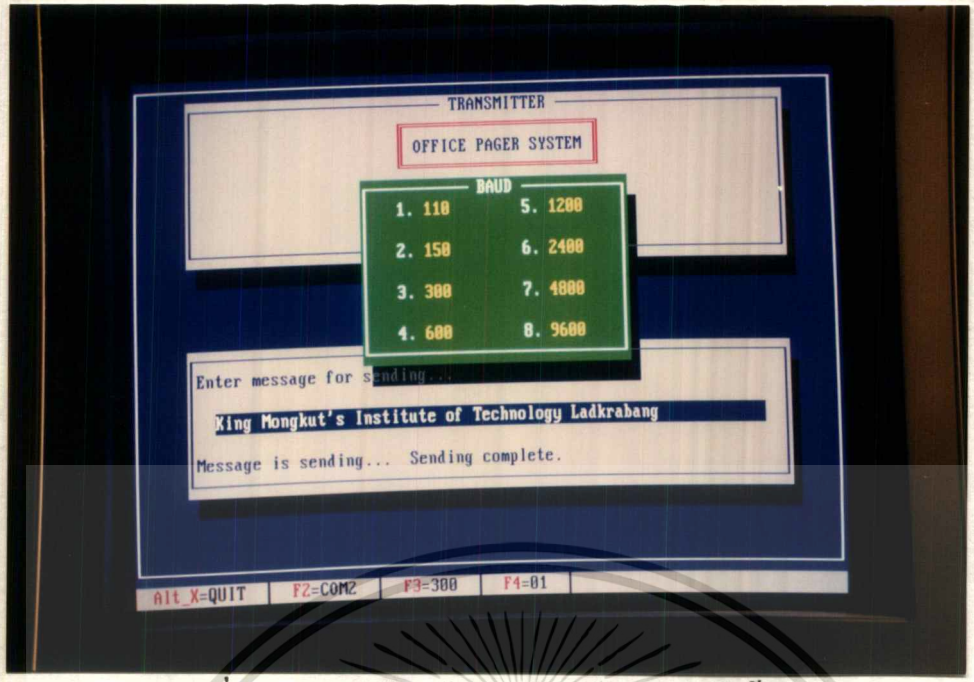
รูปที่ 4.13 แสดงหน้าจอเริ่มต้นเมื่อเข้าสู่โปรแกรม

รูปที่ 4.14 แสดงหน้าจอการเซตแอดเดรสของเครื่องรับ



รูปที่ 4.15 แสดงหน้าจอเมื่อการส่งข้อความไม่เกิดการผิดพลาด

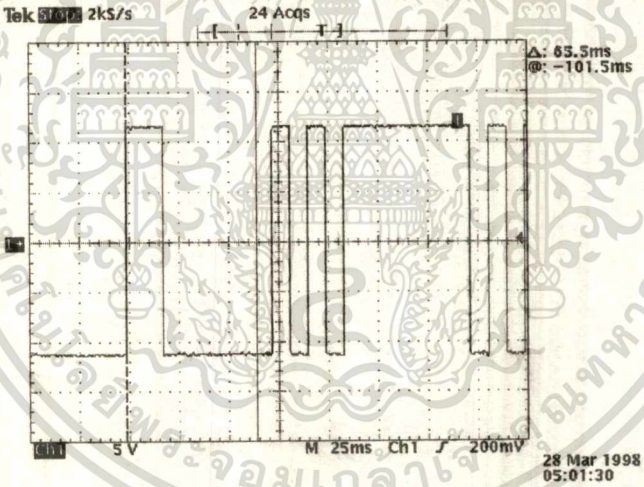
รูปที่ 4.16 แสดงหน้าจอการเซตพอร์ตของคอมพิวเตอร์ที่จะส่งข้อความ



รูปที่ 4.17 แสดงหน้าจอการเซตอัตราบอดของการส่งข้อความ

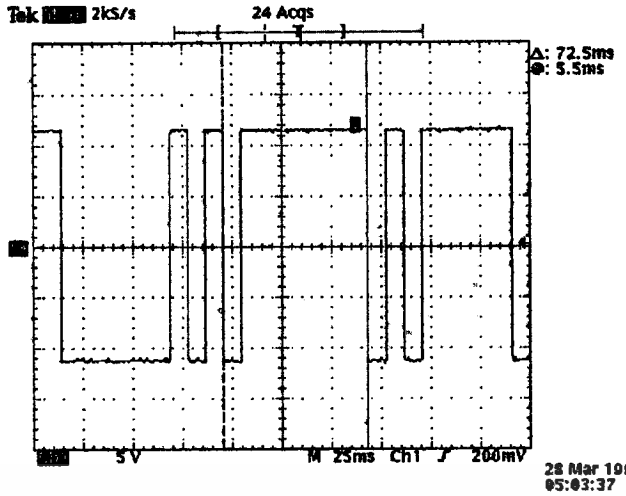
ส่วนของเฟรมข้อมูลต่างๆที่วัดได้จะแสดงดังรูปต่อไปนี้

– **สตาร์ทบิท เป็นบิทที่บอกทางเครื่องรับว่าขณะนี้ได้มีการส่งข้อมูลมาแล้ว**



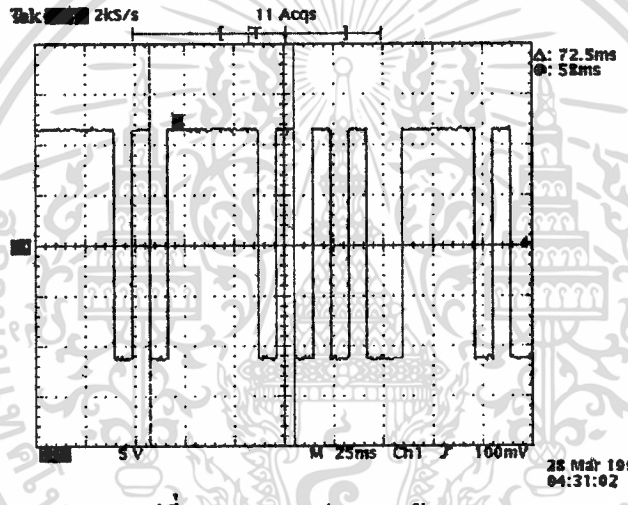
รูปที่ 4.18 แสดงสตาร์ทบิทของเฟรมข้อมูล

– **แอดเดรสของเพจเจอร์ เป็นข้อมูลสำหรับกำหนดหมายเลขเครื่องของเพจเจอร์ ถ้าหมายเลขเครื่องที่ส่งมาไม่ตรงกับเพจเจอร์ตัวใดเพจเจอร์ตัวนั้นจะไม่รับข้อความ ที่แสดงในรูปที่ 4.11 เป็นหมายเลข 01**



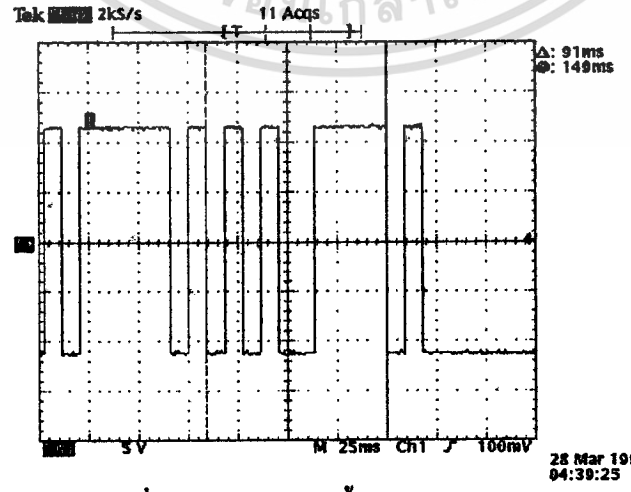
รูปที่ 4.19 แสดงเฟรมข้อมูลที่เป็นหมายเลขของเครื่องเพจเจอร์

– ตัวอย่างเฟรมข้อมูลตัวอักษร "A" เป็นตัวอย่างของข้อมูลที่ถูกส่งออกไป ถ้าข้อมูลเป็นข้อความอักขระแต่ละตัวก็จะมีลักษณะเช่นเดียวกัน



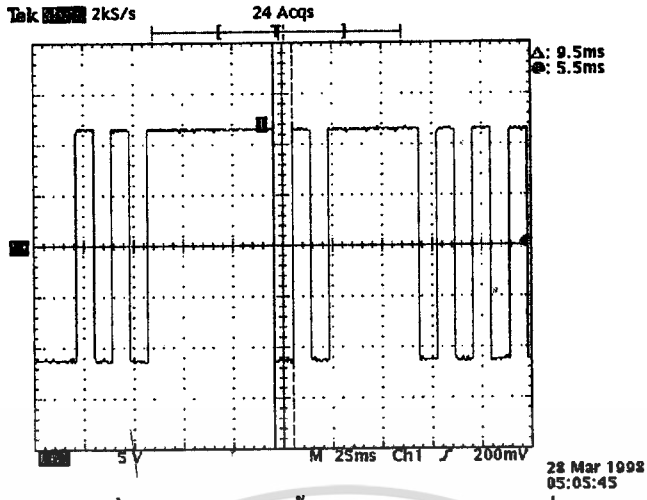
รูปที่ 4.20 แสดงเฟรมของอักขระ "A"

– เฟรมสิ้นสุดการส่งข้อมูล เป็นเฟรมที่บอกให้เครื่องรับรู้ว่าข้อความที่ส่งมาได้สิ้นสุดแล้ว ในที่นี้จะใช้รหัส ASCII ของปุ่มเอ็นเทอร์เป็นเฟรมที่บอกการสิ้นสุด ซึ่งมีรหัส 0Dh



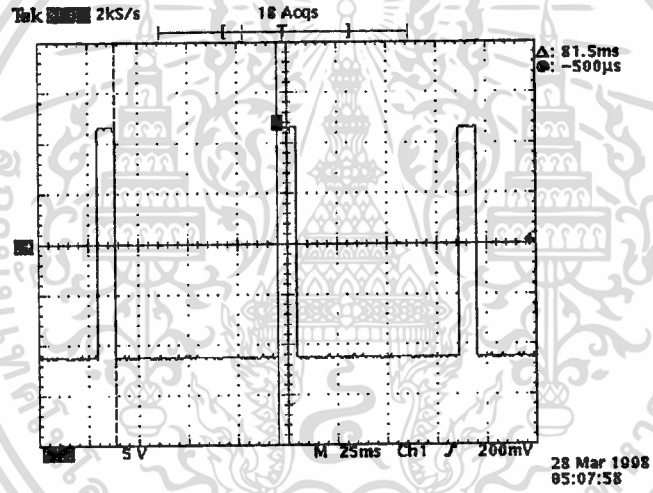
รูปที่ 4.21 แสดงเฟรมสิ้นสุดการส่งข้อความ

– สต๊อปบิท เป็นบิทที่ให้เครื่องรับรู้ว่าจบการส่งเฟรมข้อมูลหนึ่งๆแล้วตามมาตรฐาน RS-232



รูปที่ 4.22 แสดงบิทสิ้นสุดของเฟรมข้อมูลหนึ่งๆ

– สภาวะไอเดิล เป็นสภาวะที่ไม่มีการส่งข้อความมา ที่สภาวะนี้จะมีสถานะเป็น "1" ตลอดเวลาเมื่อเป็น "0" เมื่อไรแสดงว่าเป็นการเริ่มสตาร์ทบิทของเฟรมข้อมูลแล้ว



รูปที่ 4.23 แสดงสถานะเมื่อไม่มีการส่งข้อความ

4.5 ขั้นตอนการทดลองและผลการทดลองของเครื่องรับ

4.5.1 ขั้นตอนการทดลอง

– ต่อวงจรตามรูปวงจรแผ่นที่ 5 ในภาคผนวก

– ต่อเอาท์พุตของวงจรเครื่องรับเข้ากับวงจรเครื่องขยายเสียงเพื่อออกลำโพง

– ปรับ VC เพื่อเปลี่ยนสถานีการรับไปเรื่อยๆ ถ้าสถานีใดรับไม่ชัดให้ปรับทริมเมอร์ (trimmer) ซึ่งอยู่ที่จุด G ที่ตัวของ VC

4.5.2 ผลการทดลอง

สามารถรับคลื่นวิทยุที่คลื่นความถี่ 108 MHz ได้ เป็นผลให้สามารถรับความถี่ของเครื่องส่ง FM ที่ทดลองก่อนหน้านี้ได้

4.6 การนำแต่ละภาคที่ทดลองมาต่อกัน

วงจรตามรูปวงจรแผ่นที่ 6 ในภาคผนวกเป็นวงจรรวมในส่วนของเครื่องรับคือมีภาคเครื่องรับFM, เอฟเอสเคดีมอดูเลเตอร์ และชุดคอนโทรลควบคุมการแสดงผล ทั้งหมดนี้จะใช้แหล่งจ่ายไฟ +6 โวลต์ จากแบตเตอรี่ขนาด AA จำนวน 4 ก้อน

ในส่วนของเครื่องส่งประกอบไปด้วยเครื่องคอมพิวเตอร์, เอฟเอสเคมอดูเลเตอร์ และเครื่องส่งFM ในส่วนนี้จะใช้แหล่งจ่ายไฟ +12 โวลต์ ให้กับเครื่องส่งและ +5 โวลต์ ให้กับ เอฟเอสเคมอดูเลเตอร์ อัตราการส่งข้อมูลที่ใช้ในการส่งระบบนี้คือ 300 bps

ในขั้นตอนนี้จะทดลองส่งข้อมูลออกอากาศแทนการส่งข้อมูลตามสาย ผลปรากฏว่าระยะทางการส่งสัญญาณสามารถส่งได้ประมาณ 20 เมตร และสามารถรับข้อความได้ถูกต้องแต่ในการทดลอง การแสดงผลอาจจะมีความผิดพลาดบ้างเนื่องจากสายบัสที่ใช้ต่อ LCD มีความยาวเกินไป ทำให้ข้อมูลเดินทางผิดพลาดได้ แต่จุดบกพร่องนี้ได้ทำการแก้ไขเมื่อนำมาประกอบในแผ่นปริ้นท์ที่ใช้งานจริง ส่วนตัวเครื่องรับและเครื่องส่งที่ใช้งานจริงจะแสดงได้ดังรูปข้างล่าง

รูปที่ 4.24 แสดงอุปกรณ์ภายในของเครื่องส่ง

รูปที่ 4.25 แสดงอุปกรณ์ภายในของเครื่องรับ

บทที่ 5
บทวิจารณ์และบทสรุปผลการทดลอง

โครงงานนี้เป็นการสร้างระบบวิทยุติดตามตัว ซึ่งทำการสร้างโดยใช้ทฤษฎีที่เรียน นำมาประยุกต์ใช้ ในระบบวิทยุติดตามตัวนี้จะแบ่งเป็น 3 ส่วน คือ ส่วนโปรแกรมควบคุมการส่ง ส่วนของการสื่อสารผ่านคลื่นวิทยุ และส่วนของเครื่องรับ

ปัญหาที่เกิดขึ้นคือเรื่องของแหล่งจ่ายไฟที่จะใช้ในเครื่องรับ เพราะว่าจะใช้ไมโครคอนโทรเลอร์ซึ่งใช้แรงดันไฟกระแสตรง 5 โวลต์ ทำให้ตัวเครื่องรับอาจจะมีขนาดใหญ่ ในส่วนของฟังชั่นในตัวเครื่องรับนั้นต้องปรับปรุงเพื่อให้มีการใช้งานได้ดีขึ้นและสะดวกขึ้น

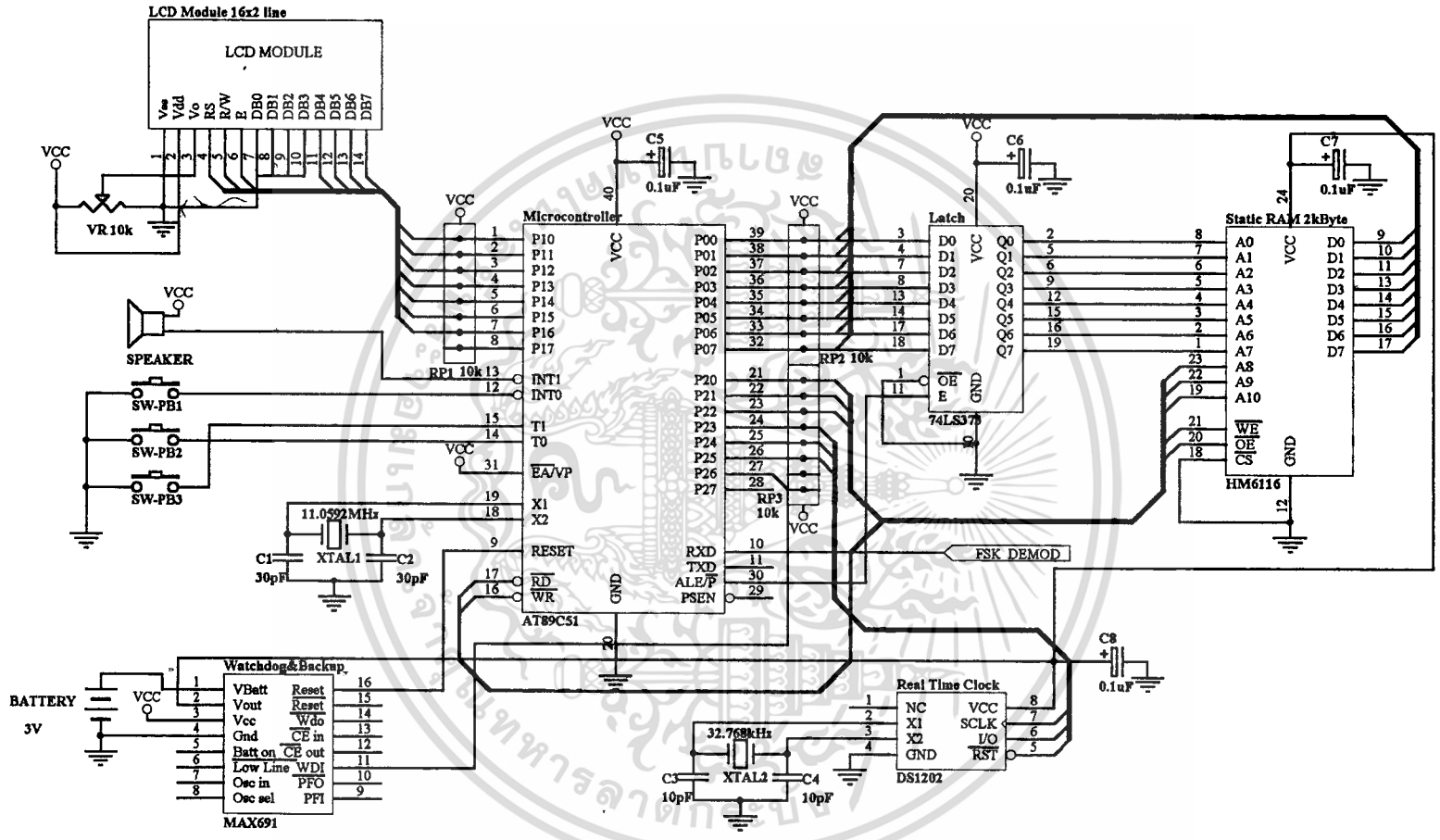
ในส่วนของระบบการส่งข้อความนั้นจะไม่ค่อยมีปัญหานักเพราะว่าจะใช้การ์ดสื่อสารอนุกรมที่มีอยู่แล้วในเครื่องคอมพิวเตอร์ทั่วไป ซึ่งการเขียนโปรแกรมเพื่อใช้ในการส่งจะมีอุปสรรคเล็กน้อยตรงที่จะต้องเขียนโปรแกรมให้ใช้งานได้สะดวกที่สุด ในการติดต่อกับพอร์ตสื่อสารอนุกรมก็จะใช้อินเตอร์รัพต์หมายเลข14 ของไบออส (BIOS) ซึ่งเป็นการบริการอินเตอร์รัพต์ของคอมพิวเตอร์

ในส่วนของเครื่องส่งนั้นสามารถที่จะใช้แหล่งจ่ายไฟ 12 โวลต์ได้ เพราะระบบของการส่งนั้นจะติดตั้งอยู่กับที่จึงสามารถใช้ระบบแหล่งจ่ายไฟที่ใหญ่ได้ การทำโครงงานนี้จะมีปัญหาต่างๆเกิดขึ้นดังนี้คือ

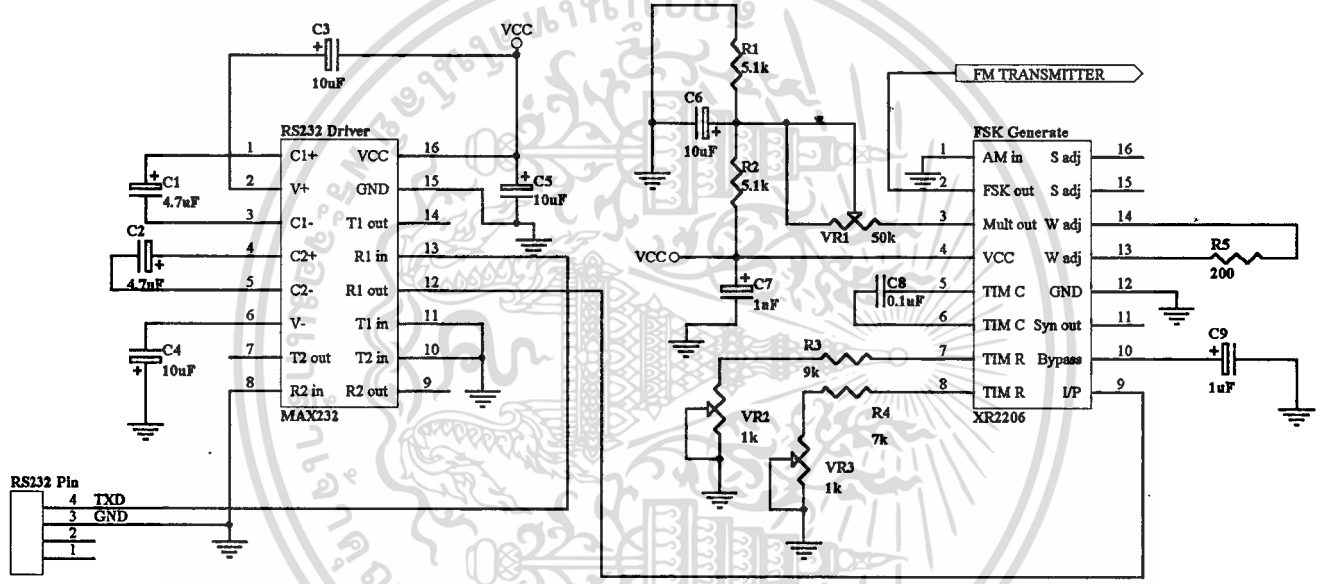
- การหาอุปกรณ์ อุปกรณ์ที่จะนำมาประกอบเป็นเครื่องรับ-ส่งนั้นจะหาได้ยากโดยเฉพาะขดลวดตัวนำซึ่งใช้เป็นวงจรเรโซแนนซ์หรือในภาคอื่นๆ เพราะไม่มีขายตามท้องตลาด
- ความไม่แน่นอนจากการต่ออุปกรณ์ เช่นการวางตำแหน่งอุปกรณ์ไม่ถูกต้องอาจทำให้เกิดการรบกวนกันได้ หรือถ้าการเดินสายของลายบนแผ่นปริ้นท์ (print) ไม่ดีก็อาจจะทำให้เกิดการรบกวนกันเกิดขึ้นไม่สามารถรับ-ส่งข้อมูลกันได้
- อุปกรณ์ที่ใช้ในเครื่องรับ-ส่ง จะต้องมีคุณภาพดีพอสมควรไม่เช่นนั้นอาจจะทำให้เกิดค่าผิดพลาดต่างๆขึ้นไม่สามารถรับ-ส่งข้อมูลกันได้

ในการทำโครงงานนี้ไม่ได้มีจุดประสงค์ที่จะทำให้ระบบเพจเจอร์มีความทันสมัยหรือดีกว่าที่มีขายตามท้องตลาดแต่มีจุดมุ่งหมายเพื่อที่จะศึกษาการทำงานของระบบและนำทฤษฎีความรู้ที่เรียนมานำมาประยุกต์ใช้งานให้เกิดผลขึ้นจริงในทางปฏิบัติ

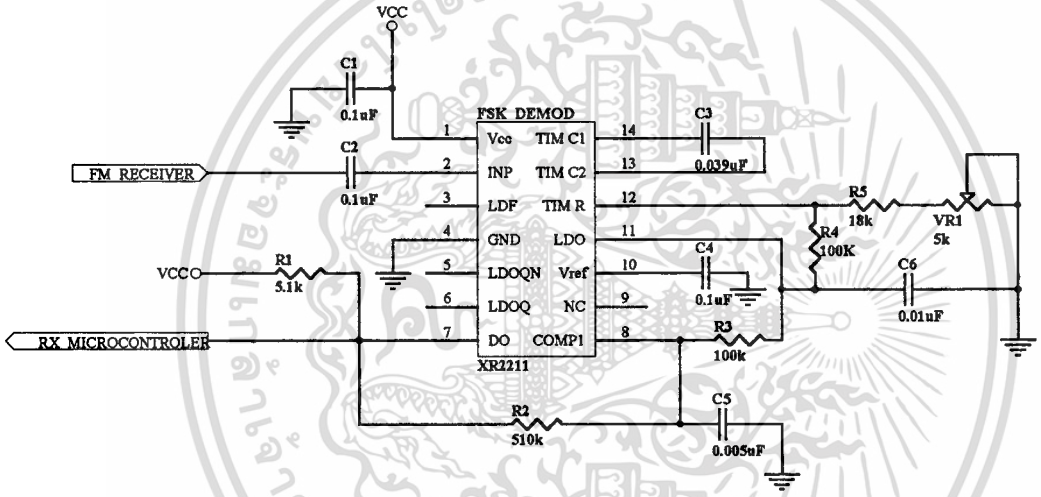
ภาคผนวก



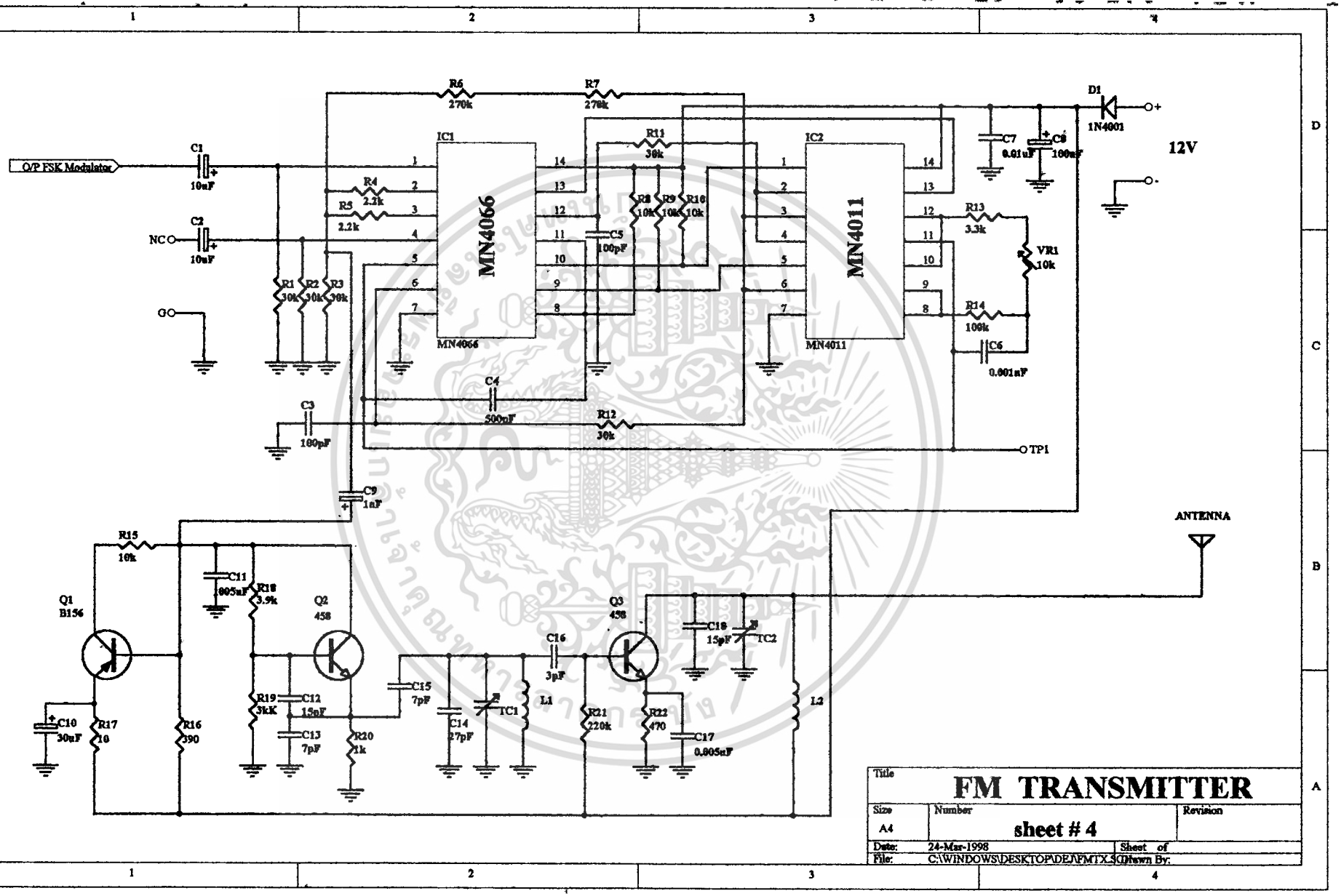
Title	**Office Pager System**	
Size: A4	Number: **sheet # 1**	Revision:
Date: 24-Mar-1998	Sheet of	
File: C:\WINDOWS\DESKTOP\DEJ\PAGER.SCH	Drawn By:	

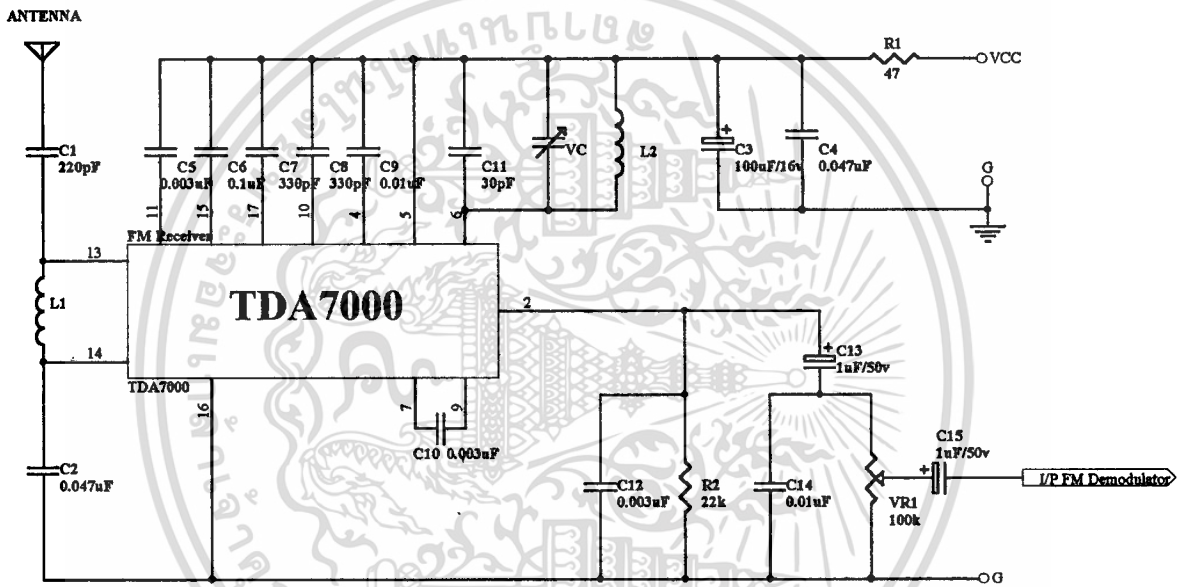


Title	**FSK MODULATOR**	
Size: A4	Number: **sheet # 2**	Revision:
Date: 24-Mar-1998	Sheet of	
File: C:\WINDOWS\DESKTOP\DEJ\FSKM.SCH	Drawn By:	

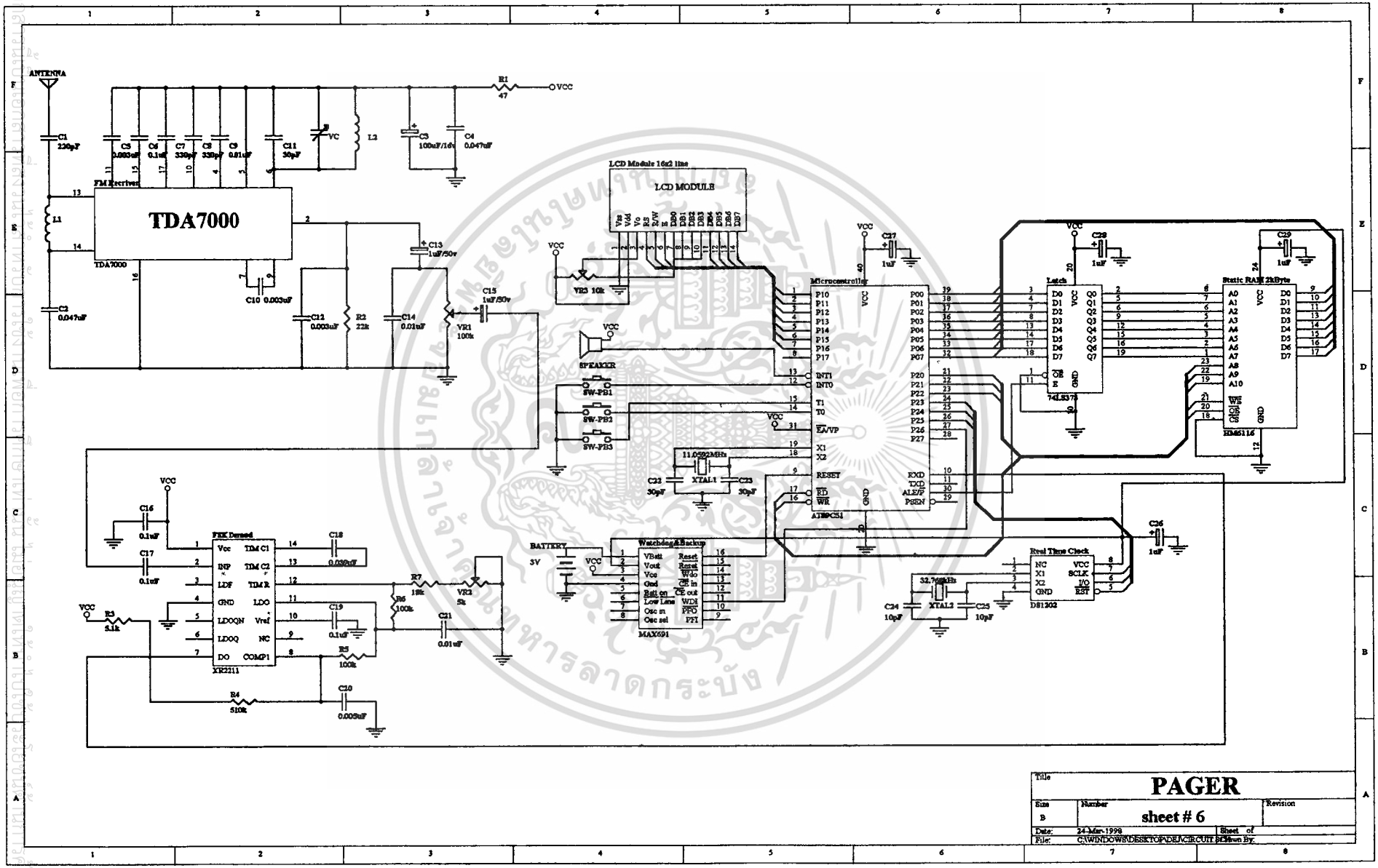


Title	**FSK DEMODULATOR**	
Size: A4	Number: **sheet # 3**	Revision:
Date: 24-Mar-1998	Sheet of	
File: C:\WINDOWS\DESKTOP\DEJ\FSKD.SC	Drawn By:	


O/P FSK Modulator
NC
GO
C1
10uF
C2
10uF
R1
30k
R2
30k
R3
30k
R4
2.2k
R5
2.2k
R6
270k
R7
270k
IC1
MN4066
MN4066
R8
10k
R9
10k
R10
10k
R11
30k
C5
100pF
C4
500nF
C3
180pF
R12
30k
IC2
MN4011
MN4011
R13
3.3k
VR1
10k
R14
100k
C6
0.001uF
C7
0.01uF
C8
100uF
D1
1N4001
12V
TP1
C9
1uF
ANTENNA
R15
10k
C11
0.005uF
R18
3.9k
Q1
B156
Q2
458
Q3
458
C10
30uF
R17
10
R16
390
R19
3kK
C12
15nF
C13
7pF
R20
1k
C15
7pF
C14
27pF
TC1
L1
C16
3pF
R21
220k
R22
470
C17
0.005uF
C18
15pF
TC2
L2
Title
FM TRANSMITTER
Size
A4
Number
sheet # 4
Revision
Date:
24-Mar-1998
Sheet of
File:
C:\WINDOWS\DESKTOP\DEJAFMTX.SCH
Drawn By:



Title	**FM RECEIVER**	
Size: A4	Number: **sheet # 5**	Revision:
Date: 24-Mar-1998	Sheet of	
File: C:\WINDOWS\DESKTOP\DE\FMRX.SCH	Drawn By:	


ANTENNA
TDA7000
FM Receiver
LCD Module 16x2 line
LCD MODULE
Microcontroller
AT89C51
Latch
74LS373
Static RAM 2kByte
XM6116
SPEAKER
SW-PB1
SW-PB2
SW-PB3
BATTERY
3V
Watchdog&Backup
MAX691
Real Time Clock
DS1202
FSK Demod
XR2211
PAGER
sheet # 6
Size
B
Number
Revision
Date:
24-Mar-1999
Sheet of
File:
C:\WINDOWS\DESKTOP\DEJACR.CUT
Drawn By:

โปรแกรมการส่งข้อมูล

```
#include <stdio.h>
#include <conio.h>
#include <dos.h>
#include <bios.h>
#include <stdlib.h>

void TextWindow(int x1,int y1,int x2,int y2,int Ver,int Hor,
     int LUCorner,int RUCorner,int LBCorner,int RBCorner,int fcolor,
     int bcolor);
unsigned int ReadFullKey(void);
void TextTitle(void);
void TextTitle_for_Send(void);
void Send_message(void);
void init_port(int port,unsigned code);
void Setport(void);
void RecoveryBigWindow(void);
void RecoverySmallWindow(void);
void QuitWindow(void);
void SetBaud(void);
void DefineAddress(void);
int Check_Special_Key(void);
int send_port(int port,unsigned char string[]);
int normal_message(void);
int shift_message(void);
int Check_Backspace(void);
int type_message(void);

unsigned char message[200],a;
int first_char, x3, pt1=0, chtest, baud1=0x43, d, AddressNet,
    check_first_loop=1;
char c;

void main()
{
    clrscr();
    TextWindow(1,1,79,24,0xb3,0xc4,0xda,0xbf,0xc0,0xd9,WHITE,BLUE);
    TextWindow(9,3,73,23,0xb3,0xc4,0xda,0xbf,0xc0,0xd9,BLACK,BLACK);
    TextWindow(8,2,72,22,0xb3,0xc4,0xda,0xbf,0xc0,0xd9,BLUE,LIGHTGRAY);
    TextWindow(30,4,52,6,0xba,0xcd,0xc9,0xbb,0xc8,0xbc,BLINK|RED,LIGHTGRA
               Y);
    TextWindow(18,9,63,21,0xba,0xcd,0xc9,0xbb,0xc8,0xbc,RED,LIGHTGRAY);
    TextTitle();

    while(1)
      {
        while(!kbhit());
        chtest=ReadFullKey();
        if((chtest>>8)==0x2d)
          {
            if((chtest&0xff)==0)
              {
                _AH=0;
                _AL=0x3;
                geninterrupt(0x10);
                break;
              }
          }
        if(!((chtest&0xff)==0))
          {
            TextWindow(1,1,79,24,0xb3,0xc4,0xda,0xbf,0xc0,0xd9,WHITE,BLU
                       E);
```

```
                TextWindow(8,3,74,10,0xb3,0xc4,0xda,0xbf,0xc0,0xd9,BLACK,BLA
                        CK);
                TextWindow(7,2,73,9,0xb3,0xc4,0xda,0xbf,0xc0,0xd9,BLUE,LIGHT
                        GRAY);
                TextWindow(8,15,74,21,0xb3,0xc4,0xda,0xbf,0xc0,0xd9,BLACK,BL
                        ACK);
                TextWindow(7,14,73,20,0xb3,0xc4,0xda,0xbf,0xc0,0xd9,BLUE,LIG
                        HTGRAY);
                TextWindow(30,3,52,5,0xba,0xcd,0xc9,0xbb,0xc8,0xbc,RED,LIGHT
                        GRAY);
                TextTitle_for_Send();
                while(1)
                   {
                     DefineAddress();
                     if(((c=='n')||(c=='N')||(c==0x1b))||(check_first_loop==1))
                       continue;
                     first_char=1;
                     for(x3=0;x3<128;x3++) message[x3]=0xff;
                     textcolor(WHITE);
                     textbackground(BLUE);
                     for(x3=0;x3<=60;x3++)
                       {
                         gotoxy(x3+10,17);
                         cprintf(" ");
                       }
                     textcolor(LIGHTGRAY);
                     textbackground(LIGHTGRAY);
                     for(x3=0;x3<=60;x3++)
                       {
                         gotoxy(x3+8,19);
                         cprintf(" ");
                       }
                     Send_message();
                   }
              }
       }
}

void TextWindow(int x1,int y1,int x2,int y2,int Ver,int Hor,
     int LUCorner,int RUCorner,int LBCorner,int RBCorner,int fcolor,
     int bcolor)
{
   int x,y;

   textcolor(fcolor);
   textbackground(bcolor);

   for(y=y1+1;y<y2;y++)
     {
       for(x=x1+1;x<x2;x++)
         {
           gotoxy(x,y);
           putch(0x20);
         }
       gotoxy(x1,y);putch(Ver);
       gotoxy(x2,y);putch(Ver);
     }
   gotoxy(x1+1,y1);
   for(x=x1+1;x<x2;x++) putch(Hor);
   gotoxy(x1+1,y2);
   for(x=x1+1;x<x2;x++) putch(Hor);
```

```
   gotoxy(x1,y1);putch(LUCorner);
   gotoxy(x2,y1);putch(RUCorner);
   gotoxy(x1,y2);putch(LBCorner);
   gotoxy(x2,y2);putch(RBCorner);
}

unsigned int ReadFullKey()
{
   int keycheck;

   keycheck=bioskey(1);
   if(keycheck)
     return bioskey(0);
   else
     return 0;
}

void TextTitle()
{
   TextWindow(19,10,62,20,0x20,0x20,0x20,0x20,0x20,0x20,LIGHTGRAY,LIGHTGR
              AY);
   textcolor(BLUE);
   textbackground(LIGHTGRAY);
   gotoxy(27,3);cprintf("!!!!!!!! WELCOME TO !!!!!!!!!");
   gotoxy(32,5);cprintf("OFFICE PAGER SYSTEM");
   gotoxy(16,7);cprintf("King Mongkut's Institute of Technology
                        Ladkrabang");
   gotoxy(38,8);cprintf("[KMITL]");
   gotoxy(20,10);cprintf("Produce by.....");
   gotoxy(23,12);cprintf("1. MR.Dejpol Chanasomboon   38013014");
   gotoxy(23,14);cprintf("2. MR.Pakonchai Ravipan     38013019");
   gotoxy(23,16);cprintf("3. MR.Sutthi khamsri        38013032");
   gotoxy(27,18);cprintf("Telecommunication Engineering");
   gotoxy(39,20);cprintf("(3T/1)");
}

void TextTitle_for_Send()
{
   textcolor(BLUE);
   textbackground(LIGHTGRAY);
   gotoxy(35,2);cprintf(" TRANSMITTER ");
   gotoxy(32,4);cprintf("OFFICE PAGER SYSTEM");
   gotoxy(27,6);cprintf("Telecommunication Engineering");
   gotoxy(38,7);cprintf("[KMITL]");
   gotoxy(40,8);cprintf("1997");
   gotoxy(38,14);cprintf(" MESSAGE ");
   gotoxy(8,15);cprintf("Enter message for sending...");
   TextWindow(1,25,79,25,0x20,0x20,0x20,0x20,0x20,0x20,BLACK,LIGHTGRAY);
   gotoxy(8,25);cprintf("=QUIT  %c    =COM1  %c    =300  %c    =
                        %c",0xb3,0xb3,0xb3,0xb3);
   textcolor(RED);
   textbackground(LIGHTGRAY);
   gotoxy(3,25);cprintf("Alt_X");
   gotoxy(18,25);cprintf("F2");
   gotoxy(30,25);cprintf("F3");
   gotoxy(41,25);cprintf("F4");
   TextWindow(10,17,70,17,0x20,0x20,0x20,0x20,0x20,0x20,WHITE,BLUE);
   gotoxy(10,17);
}
```

```
void Send_message()
{
   type_message();
   init_port(pt1,baud1);
   textcolor(LIGHTGRAY);
   textbackground(LIGHTGRAY);
   gotoxy(8,19);
   cprintf("                                                    ");
   textcolor(BLUE);
   textbackground(LIGHTGRAY);
   gotoxy(8,19);
   cprintf("Message is sending...");
   send_port(pt1,message);
}

void init_port(int port,unsigned code)
{
   _DX=port;
   _AH=0;
   _AL=code;
   geninterrupt(0x14);
}

int send_port(int port,unsigned char string[])
{
   int x;

   _DX=port;
   _AH=1;
   _AL=0x7e;
   geninterrupt(0x14);

   _DX=port;
   _AH=1;
   _AL=AddressNet;
   geninterrupt(0x14);
   delay(10);

   x=0;
   do
      {
         a=string[x];

         _DX=port;
         _AH=1;
         _AL=a;
         geninterrupt(0x14);x++;
         if(_AH & 0x80)
            {
               textcolor(RED);
               textbackground(LIGHTGRAY);
               cprintf("  Message sending error.\a");gotoxy(60,19);
               Check_Special_Key();
               return 0;
            }
      }while(x<128);cprintf("  Sending complete.");
   Check_Special_Key();
   return 0;
}
```

```
int type_message()
{
   normal_message();
   if(message[x3-1]==0x0d)
      {
        return 0;
      }
   shift_message();
   return 0;
}

int normal_message()
{
   gotoxy(10,17);
   for(x3=0;x3<60;x3++)
      {
        message[x3]=Check_Special_Key();
        if(message[x3]==0x08)
           {
             message[x3-1]=0x20;
             putch(0x08);putch(0x20);putch(0x08);
             x3-=2;
           }
        else
           {
             if(message[x3]==0x0d)
                {
                  x3++;
                  return 0;
                }
             gotoxy(x3+10,17);putch(message[x3]);
           }
      }
   return 0;
}

int shift_message()
{
   while(1)
      {
        message[x3]=Check_Special_Key();
        Check_Backspace();
        if(message[x3-1]==0x0d)
           {
           return 0;
           }
      }
}

int Check_Backspace()
{
   int y3;

   if(message[x3]==0x08)
      {
        if(x3<=60)
           {
             putch(0x08);putch(0x20);putch(0x08);
             x3-=1;
           }
        else
```

```
          {
            x3-=61;
            y3=x3;
            gotoxy(10,17);
            while(x3<y3+60)
               {
                  cprintf("%c",message[x3]);
                  x3++;
               }
            first_char--;
          }
        return 0;
      }
   else
      {
        if(message[x3]==0x0d)
           {
              x3++;
              return 0;
           }
        else
           {
              if(x3<60)
                 {
                  gotoxy(x3+10,17);putch(message[x3]);
                  x3++;
                 }
              else
                 {
                  gotoxy(10,17);
                  for(y3=first_char;y3<first_char+60;y3++)
                    cprintf("%c",message[y3]);
                  first_char++;x3++;
                 }
           }
        return 0;
      }
}

int Check_Special_Key()
{
   while(1)
      {
        textcolor(WHITE);
        textbackground(BLUE);
        while(!kbhit());
        chtest=ReadFullKey();
        if(bioskey(2)&0x04) continue;
        if((x3==0)&((chtest&0xff)==0x08)) continue;
        if((chtest>>8)==0x3c)
           {
              Setport();
              gotoxy(x3+10,17);
              if(message[x3-1]==0x0d) gotoxy(60,19);
              continue;
           }
        if((chtest>>8)==0x3d)
           {
              SetBaud();
              gotoxy(x3+10,17);
              if(message[x3-1]==0x0d) gotoxy(60,19);
```

```
            continue;
          }
        if(((chtest>>8)==0x3e)&(message[x3-1]!=0x0d))
          {
            while(1)
              {
                DefineAddress();
                if((c=='n')|(c=='N')|(c==0x1b)) continue;
                break;
              }
            gotoxy(x3+10,17);
            if(message[x3-1]==0x0d) gotoxy(60,19);
            continue;
          }
        if(((chtest&0xff)==0x1b)&(message[x3-1]!=0x0d)) continue;
        if(((chtest>>8)==0x2d)&((chtest&0xff)==0))
          {
            QuitWindow();
            gotoxy(x3+10,17);
            if(message[x3-1]==0x0d) gotoxy(60,19);
            continue;
          }
        if((chtest&0xff)==0) continue;
        if(x3==127)
          {
            if(((chtest&0xff)==0x0d)|((chtest&0xff)==0x08))
              {
                textcolor(LIGHTGRAY);
                textbackground(LIGHTGRAY);
                gotoxy(8,19);
                cprintf("
                        ");
                textcolor(WHITE);
                textbackground(BLUE);
                return chtest;
              }
            textcolor(RED);
            textbackground(LIGHTGRAY);
            gotoxy(8,19);
            cprintf("End of message!!! Press ENTER for
                    send,please...\a");
            gotoxy(70,17);
            continue;
          }
        return chtest;
      }
}

void Setport()
{
  TextWindow(31,10,52,14,0x20,0x20,0x20,0x20,0x20,0x20,DARKGRAY,BLACK);
  for(d=31;d<=52;d++)
    {
      gotoxy(d,14);
      cprintf("%c",0xc4);
    }
  gotoxy(38,14);cprintf(" MESSAGE ");
  TextWindow(30,9,51,13,0xb3,0xc4,0xda,0xbf,0xc0,0xd9,WHITE,GREEN);
  gotoxy(38,9);cprintf(" PORT ");
  gotoxy(38,10);cprintf("1.");
  gotoxy(38,12);cprintf("2.");
```

```
  TextWindow(40,10,50,12,0x20,0x20,0x20,0x20,0x20,0x20,GREEN,GREEN);
  textcolor(YELLOW);
  textbackground(GREEN);
  gotoxy(41,10);cprintf("COM1");
  gotoxy(41,12);cprintf("COM2");
  while(1)
    {
      while(!kbhit());
      chtest=ReadFullKey();
      textcolor(BLACK);
      textbackground(LIGHTGRAY);
      if((chtest&0xff)==0x1b) break;
      if((chtest&0xff)==0x31)
        {
          pt1=0;
          gotoxy(21,25);cprintf("COM1");
          break;
        }
      if((chtest&0xff)==0x32)
        {
          pt1=1;
          gotoxy(21,25);cprintf("COM2");
          break;
        }
    }
  RecoverySmallWindow();
}

void RecoveryBigWindow()
{
  TextWindow(26,6,56,14,0x20,0x20,0x20,0x20,0x20,0x20,BLUE,BLUE);
  TextWindow(26,6,56,10,0x20,0x20,0x20,0x20,0x20,0x20,BLACK,BLACK);
  TextWindow(26,6,56,9,0x20,0x20,0x20,0x20,0x20,0x20,LIGHTGRAY,LIGHTGRAY
            );
  TextWindow(26,9,56,9,0xc4,0xc4,0xc4,0xc4,0xc4,0xc4,BLUE,LIGHTGRAY);
  TextWindow(26,14,56,15,0x20,0x20,0x20,0x20,0x20,0x20,LIGHTGRAY,LIGHTGR
           AY);
  TextWindow(26,14,56,14,0xc4,0xc4,0xc4,0xc4,0xc4,0xc4,BLUE,LIGHTGRAY);
  gotoxy(40,8);cprintf("1997");
  gotoxy(38,14);cprintf(" MESSAGE ");
  gotoxy(27,6);cprintf("Telecommunication Engineering");
  gotoxy(38,7);cprintf("[KMITL]");
  gotoxy(8,15);cprintf("Enter message for sending...");
}

void QuitWindow()
{
  TextWindow(31,9,52,14,0x20,0xc4,0xc4,0xc4,0xc4,0xc4,DARKGRAY,BLACK);
  gotoxy(38,14);cprintf(" MESSAGE ");
  TextWindow(30,8,51,13,0xb3,0xc4,0xda,0xbf,0xc0,0xd9,WHITE,GREEN);
  TextWindow(31,9,50,12,0x20,0x20,0x20,0x20,0x20,0x20,GREEN,GREEN);
  textcolor(RED);
  textbackground(GREEN);
  gotoxy(34,9);cprintf("QUIT to MS-DOS");
  gotoxy(34,11);cprintf("Are you sure ?");
  gotoxy(39,12);cprintf("(Y/N)");
  do
    {
      c=getch();
      if((c=='y')||(c=='Y'))
        {
```

```
            _AH=0;
            _AL=0x3;
            geninterrupt(0x10);
            exit(0);
          }
     }while(!((c=='n')|(c=='N')|(c==0x1b)));
   RecoverySmallWindow();
}

void SetBaud()
{
   TextWindow(27,7,56,15,0x20,0x20,0x20,0x20,0x20,0x20,DARKGRAY,BLACK);
   gotoxy(56,9);cprintf("%c",0xc4);
   gotoxy(56,14);cprintf("%c",0xc4);
   gotoxy(8,15);printf("Enter message for sending...");
   TextWindow(26,6,55,14,0xb3,0xc4,0xda,0xbf,0xc0,0xd9,WHITE,GREEN);
   gotoxy(38,6);cprintf(" BAUD ");
   gotoxy(30,7);cprintf("1.");
   gotoxy(30,9);cprintf("2.");
   gotoxy(30,11);cprintf("3.");
   gotoxy(30,13);cprintf("4.");
   gotoxy(44,7);cprintf("5.");
   gotoxy(44,9);cprintf("6.");
   gotoxy(44,11);cprintf("7.");
   gotoxy(44,13);cprintf("8.");
   TextWindow(46,7,54,13,0x20,0x20,0x20,0x20,0x20,0x20,GREEN,GREEN);
   textcolor(YELLOW);
   textbackground(GREEN);
   gotoxy(33,7);cprintf("110");
   gotoxy(33,9);cprintf("150");
   gotoxy(33,11);cprintf("300");
   gotoxy(33,13);cprintf("600");
   gotoxy(47,7);cprintf("1200");
   gotoxy(47,9);cprintf("2400");
   gotoxy(47,11);cprintf("4800");
   gotoxy(47,13);cprintf("9600");
   while(1)
      {
        while(!kbhit());
        chtest=ReadFullKey();
        textcolor(BLACK);
        textbackground(LIGHTGRAY);
        if((chtest&0xff)==0x1b) break;
        switch(chtest&0xff)
          {
            case 0x31:baud1=0x03;
                      gotoxy(33,25);cprintf("110 ");
                      break;
            case 0x32:baud1=0x23;
                      gotoxy(33,25);cprintf("150 ");
                      break;
            case 0x33:baud1=0x43;
                      gotoxy(33,25);cprintf("300 ");
                      break;
            case 0x34:baud1=0x63;
                      gotoxy(33,25);cprintf("600 ");
                      break;
            case 0x35:baud1=0x83;
                      gotoxy(33,25);cprintf("1200");
                      break;
            case 0x36:baud1=0xa3;
```

```
                    gotoxy(33,25);cprintf("2400");
                    break;
            case 0x37:baud1=0xc3;
                    gotoxy(33,25);cprintf("4800");
                    break;
            case 0x38:baud1=0xe3;
                    gotoxy(33,25);cprintf("9600");
                    break;
            default  :continue;
          }
        break;
      }
   RecoveryBigWindow();
}

void RecoverySmallWindow()
{
   TextWindow(30,8,52,14,0x20,0x20,0x20,0x20,0x20,0x20,BLUE,BLUE);
   TextWindow(30,8,52,10,0x20,0x20,0x20,0x20,0x20,0x20,BLACK,BLACK);
   TextWindow(30,8,52,9,0x20,0x20,0x20,0x20,0x20,0x20,LIGHTGRAY,LIGHTGRAY
            );
   TextWindow(30,9,52,9,0xc4,0xc4,0xc4,0xc4,0xc4,0xc4,BLUE,LIGHTGRAY);
   TextWindow(30,14,52,14,0xc4,0xc4,0xc4,0xc4,0xc4,0xc4,BLUE,LIGHTGRAY);
   gotoxy(40,8);cprintf("1997");
   gotoxy(38,14);cprintf(" MESSAGE ");
}

void DefineAddress()
{
   int x4,x5,SubAdd1,SubAdd2;
   char Address[3];

   TextWindow(31,10,52,14,0x20,0x20,0x20,0x20,0x20,0x20,DARKGRAY,BLACK);
   for(d=31;d<=52;d++)
     {
       gotoxy(d,14);
       cprintf("%c",0xc4);
     }
   gotoxy(38,14);cprintf(" MESSAGE ");
   TextWindow(30,9,51,13,0xb3,0xc4,0xda,0xbf,0xc0,0xd9,WHITE,GREEN);
   gotoxy(37,9);cprintf(" ADDRESS ");
   gotoxy(31,10);cprintf("Address of receiver.");
   TextWindow(31,11,50,12,0x20,0x20,0x20,0x20,0x20,0x20,YELLOW,GREEN);
   gotoxy(40,12);
   for(x4=0;x4<3;x4++)
     {
       while(!kbhit());
       x5=ReadFullKey();
       Address[x4]=x5;
       if(((x5>>8)==0x2d)&((x5&0xff)==0))
         {
           QuitWindow();
           return;
         }
       if((x4>0)&(((int)Address[x4]&0xff)==0x08))
         {
           putch(0x08);putch(0x20);putch(0x08);
           x4-=2;
           continue;
         }
       if((x4==2)&(((int)Address[x4]&0xff))==0x0d) break;
```

```
        if((((int)Address[x4]&0xff)==0x1b)&(check_first_loop==0)) break;
        if((!(((int)Address[x4]<=57)&((int)Address[x4]>=48)))|(x4==2))
          {
            x4--;
            continue;
          }
        gotoxy(40+x4,12);putch(Address[x4]);
      }
    RecoverySmallWindow();
    if(!(((int)Address[x4]&0xff)==0x1b))
      {
        SubAdd1=(int)Address[0]-48;
        SubAdd2=(int)Address[1]-48;
        AddressNet=(SubAdd1*10)+SubAdd2;
        textcolor(BLACK);
        textbackground(LIGHTGRAY);
        gotoxy(44,25);cprintf("%d",SubAdd1);
        gotoxy(45,25);cprintf("%d",SubAdd2);
      }
    check_first_loop=0;c=0;
  }
```

โปรแกรมการรับข้อมูล

```
;*************************************************************************
;******************      Office   Pager   System      ********************
;*************************************************************************

DB4             EQU         P1.0
DB5             EQU         P1.1
DB6             EQU         P1.2
DB7             EQU         P1.3
RS              EQU         P1.4
RW              EQU         P1.5
E               EQU         P1.6
KEY1            EQU         P3.4
KEY2            EQU         P3.5
KEY3            EQU         P3.2
SPEAKER         EQU         P3.3
RST             EQU         P2.3
IO              EQU         P2.4
SCLK            EQU         P2.5
WDI             EQU         P1.7

HOUR            EQU         30H
MIN             EQU         31H
SEC             EQU         32H
DATE            EQU         33H
MONTH           EQU         34H
YEAR            EQU         35H
A_HOUR          EQU         36H
A_MIN           EQU         37H
A_DATE          EQU         38H
A_MONTH         EQU         39H
A_YEAR          EQU         3AH
A_FLAG          EQU         3BH

M1_FLAG         EQU         3CH
M2_FLAG         EQU         3DH
M3_FLAG         EQU         3EH
M4_FLAG         EQU         3FH
M5_FLAG         EQU         40H
M6_FLAG         EQU         41H
M7_FLAG         EQU         42H
M8_FLAG         EQU         43H
M9_FLAG         EQU         44H
M10_FLAG        EQU         45H

EM_POINTER_H    EQU         46H
EM_POINTER_L    EQU         47H
MS_POINTER_H    EQU         48H
MS_POINTER_L    EQU         49H

INT_FLAG        EQU         4AH
END_FLAG        EQU         4BH
NUM             EQU         4CH
AREA            EQU         4DH
CHK_RECV        EQU         4EH
IN_AREA         EQU         4FH

M1_LOCATE       EQU         0080H
M2_LOCATE       EQU         0180H
M3_LOCATE       EQU         0280H
M4_LOCATE       EQU         0380H
M5_LOCATE       EQU         0480H
```

```
M6_LOCATE       EQU             0580H
M7_LOCATE       EQU             0680H
M8_LOCATE       EQU             0780H
M9_LOCATE       EQU             0880H
M10_LOCATE      EQU             0980H

R_HOUR          EQU             1000H
R_MIN           EQU             1001H
R_DATE          EQU             1002H
R_MONTH         EQU             1003H
R_YEAR          EQU             1004H

R_A_HOUR        EQU             1005H
R_A_MIN         EQU             1006H
R_A_DATE        EQU             1007H
R_A_MONTH       EQU             1008H
R_A_YEAR        EQU             1009H
R_A_FLAG        EQU             100AH

R_M1_FLAG       EQU             100BH
R_M2_FLAG       EQU             100CH
R_M3_FLAG       EQU             100DH
R_M4_FLAG       EQU             100EH
R_M5_FLAG       EQU             100FH
R_M6_FLAG       EQU             1010H
R_M7_FLAG       EQU             1011H
R_M8_FLAG       EQU             1012H
R_M9_FLAG       EQU             1013H
R_M10_FLAG      EQU             1014H

;*************************************************************************
;*********************  S    T    A    R    T  ********************
;*************************************************************************

ORG    0000H

START:          LJMP      MAIN

ORG    0003H

INT0_VEC:       LJMP      INT0_KEY

;ORG    000BH

;INT_T0_VEC:    LJMP      WDOG

ORG    0023H

INT_RI_VEC:     LJMP      RECEIVE


;*************************************************************************
;*************************  M    A    I    N  ************************
;*************************************************************************

; *** Main program ***

MAIN:           MOV       A,#100
                LCALL     DELAY_ms
                LCALL     OUT_ALL
                LCALL     CHECK
```

```
                MOV     IN_AREA,#0FFH
                MOV     AREA,#00H
                CLR     P3.1
                MOV     IE,#00H                 ;Distable Interrupt
                MOV     IP,#90H                 ;First Rx,next INT0
                LCALL   WAIT                    ;Wait for initial LCD
                LCALL   INIT_LCD1               ;Initial LCD,curser off
                LCALL   PLOT                    ;Plot special character
                LCALL   SERIAL_RX               ;Ready for receive massage

LOOP1:          LCALL   KEEP_ALL
                MOV     INT_FLAG,#00H
                LCALL   INIT_LCD1               ;Initial LCD,curser off
                LCALL   SHOW1                   ;Show map1
                MOV     IE,#93H                 ;Enable interrupt
  LP0:          MOV     A,A_FLAG
                CJNE    A,#00H,LP1              ;Disable alarm clock
                MOV     PCON,#01H               ;Idle mode
                JMP     $

  LP1:          MOV     A,A_FLAG
                CJNE    A,#0FFH,LP0             ;Enable alarm clock
                LJMP    READY_TIME              ;Ready alarm clock

LOOP2:          LCALL   KEEP_ALL
                MOV     INT_FLAG,#0FFH
                LCALL   INIT_LCD2               ;Initial LCD,curser on
                LCALL   SHOW2                   ;Show map2
                MOV     A,#200
                LCALL   DELAY_MS
                MOV     A,#200
                LCALL   DELAY_MS
                LJMP    MAP2_SCANK              ;Scankey on map2

RETURN1:        MOV     SP,#09
                MOV     DPTR,#LOOP1             ;Start SP at LOOP1
                MOV     09,DPH
                MOV     08,DPL
                RETI

RETURN2:        MOV     SP,#09H
                MOV     DPTR,#RECV4             ;Start SP at RECV4
                MOV     09,DPH
                MOV     08,DPL
                RETI

RETURN3:        MOV     SP,#09H
                MOV     DPTR,#RECV3             ;Start SP at RECV3
                MOV     09,DPH
                MOV     08,DPL
                RETI

RETURN4:        MOV     SP,#09H
                MOV     DPTR,#RECV10             ;Start SP at RECV10
                MOV     09,DPH
                MOV     08,DPL
                RETI

RETURN5:        MOV     SP,#09H
                MOV     DPTR,#RECV11             ;Start SP at RECV11
                MOV     09,DPH
```

```
                MOV     08,DPL
                RETI

RETURN6:        MOV     SP,#09H
                MOV     DPTR,#RECV6             ;Start SP at RECV6
                MOV     09,DPH
                MOV     08,DPL
                RETI

 WDOG:          PUSH    ACC
                MOV     A,IN_AREA
                CJNE    A,#00H,WD1
                SETB    P2.6
                MOV     A,#1
                LCALL   DELAY_SEC
                CLR     P2.6
                POP     ACC
    WD1:        RETI

;*************************************************************************
;***********  R   E   C   E   I   V   E       M   E   S   S   A   G   E ******
;*************************************************************************

; *** Ready for receive message ***

SERIAL_RX:      MOV     PCON,#00H               ;SMOD=0
                MOV     SCON,#50H               ;Serial mode1
                MOV     TMOD,#22H               ;Timer1,2 mode2
                MOV     TH1,#0A0H               ;300 baud
                MOV     TH0,#00H
                SETB    TR1                     ;Start timer1
                SETB    TR0                     ;Start timer0
                RET

; *** Receive massage and get in massage location ***

RECEIVE:        MOV     IE,#92H                 ;Disable interrupt
                MOV     R7,AREA
  RECV0:        JNB     RI,$
                CLR     RI
                MOV     A,SBUF
                CJNE    A,#7EH,RECV8
                JNB     RI,$
                CLR     RI
                MOV     A,SBUF
                MOV     NUM,A
                CJNE    A,#00H,RECV12
                LCALL   EM_LOCATE
                LJMP    RECV1
  RECV12:       CJNE    A,#01H,RX_DELAY
                LCALL   EM_LOCATE

  RECV1:        MOV     R0,#128                 ;Receive 128 byte
  RECV2:        MOV     IN_AREA,#00H
                JNB     RI,$
                CLR     RI
                MOV     A,SBUF

  RECV5:        MOV     DPH,EM_POINTER_H        ;Move empty locate pointer
in DPTR
                MOV     DPL,EM_POINTER_L
```

```
                MOVX    @DPTR,A
                INC     DPTR
                LCALL   GET_POINTER1            ;Get new empty locate
pointer
                DJNZ    R0,RECV2
                MOV     IN_AREA,#0FFH

                LCALL   INIT_LCD1               ;Initial LCD,curser off
                MOV     A,#00H                  ;Write 'speaker'
                LCALL   WRITE_CHR
                MOV     A,#01H                  ;Write 'wave'
                LCALL   WRITE_CHR
    RECV6:
                MOV     IE,#93H
                MOV     A,INT_FLAG
                CJNE    A,#0FFH,RECV3           ;Check old map
                LJMP    RETURN3
    RECV3:      LJMP    RETURN2
    RECV4:      MOV     A,NUM
                CJNE    A,#00H,RECV13
                LCALL   BEEP1
                MOV     IE,#93H
                LJMP    LOOP1
    RECV13:     CJNE    A,#01H,RECV7
                LCALL   BEEP1                   ;Beep.. Beep...
    RECV7:      MOV     IE,#93H
                LJMP    LOOP1

    RECV8:      INC     R7
                CJNE    R7,#20,RECV9
                MOV     AREA,#00H
                LCALL   OUT_AREA
                MOV     A,#1
                LCALL   DELAY_SEC
                LJMP    RECV6
                MOV     A,INT_FLAG
                CJNE    A,#0FFH,RECV10          ;Check old map
                LJMP    RETURN4
   RECV10:      LJMP    RETURN1

    RECV9:      MOV     AREA,R7
                MOV     A,INT_FLAG
                CJNE    A,#0FFH,RECV11          ;Check old map
                LJMP    RETURN5
   RECV11:      LJMP    RETURN1

RX_DELAY:
                MOV     R0,#128
                MOV     IN_AREA,#00H
      RX1:      JNB     RI,$
                CLR     RI
                MOV     A,SBUF
                DJNZ    R0,RX1
                MOV     IN_AREA,#0FFH
                LJMP    RECV6

; *** Check for empty locate ***

EM_LOCATE:      MOV      R1,#10
                MOV      R0,#M1_FLAG
      EM1:      CJNE     @R0,#00H,EM2
```

```
            LJMP    SET_LOCATE
      EM2:  INC     R0
            DJNZ    R1,EM1
            LCALL   AUTO_DEL_MESSAGE      ;Auto delete massage
            MOV     M1_FLAG,#0FFH
            MOV     DPTR,#M1_LOCATE
            LCALL   GET_POINTER1          ;Get new locate pointer
            RET

; *** Set pointer at empty message location ***

SET_LOCATE:
       SL0: CJNE    R0,#M1_FLAG,SL1
            MOV     @R0,#0FFH
            MOV     DPTR,#M1_LOCATE
            LCALL   GET_POINTER1
            RET
       SL1: CJNE    R0,#M2_FLAG,SL2
            MOV     @R0,#0FFH
            MOV     DPTR,#M2_LOCATE
            LCALL   GET_POINTER1
            RET
       SL2: CJNE    R0,#M3_FLAG,SL3
            MOV     @R0,#0FFH
            MOV     DPTR,#M3_LOCATE
            LCALL   GET_POINTER1
            RET
       SL3: CJNE    R0,#M4_FLAG,SL4
            MOV     @R0,#0FFH
            MOV     DPTR,#M4_LOCATE
            LCALL   GET_POINTER1
            RET
       SL4: CJNE    R0,#M5_FLAG,SL5
            MOV     @R0,#0FFH
            MOV     DPTR,#M5_LOCATE
            LCALL   GET_POINTER1
            RET
       SL5: CJNE    R0,#M6_FLAG,SL6
            MOV     @R0,#0FFH
            MOV     DPTR,#M6_LOCATE
            LCALL   GET_POINTER1
            RET
       SL6: CJNE    R0,#M7_FLAG,SL7
            MOV     @R0,#0FFH
            MOV     DPTR,#M7_LOCATE
            LCALL   GET_POINTER1
            RET
       SL7: CJNE    R0,#M8_FLAG,SL8
            MOV     @R0,#0FFH
            MOV     DPTR,#M8_LOCATE
            LCALL   GET_POINTER1
            RET
       SL8: CJNE    R0,#M9_FLAG,SL9
            MOV     @R0,#0FFH
            MOV     DPTR,#M9_LOCATE
            LCALL   GET_POINTER1
            RET
       SL9: CJNE    R0,#M10_FLAG,SL10
            MOV     @R0,#0FFH
            MOV     DPTR,#M10_LOCATE
            LCALL   GET_POINTER1
```

```
                RET
        SL10:   LJMP    SET_LOCATE

; *** Get pointer ***

GET_POINTER1:
                MOV     EM_POINTER_H,DPH        ;Get new empty locate
pointer
                MOV     EM_POINTER_L,DPL
                RET

; *** Delete message ***

AUTO_DEL_MESSAGE:
                MOV     EM_POINTER_H,#00H       ;Auto delete message
                MOV     EM_POINTER_L,#00H
                MOV     MS_POINTER_H,#00H
                MOV     MS_POINTER_L,#00H
                MOV     END_FLAG,#00H
                MOV     M1_FLAG,#00H
                MOV     M2_FLAG,#00H
                MOV     M3_FLAG,#00H
                MOV     M4_FLAG,#00H
                MOV     M5_FLAG,#00H
                MOV     M6_FLAG,#00H
                MOV     M7_FLAG,#00H
                MOV     M8_FLAG,#00H
                MOV     M9_FLAG,#00H
                MOV     M10_FLAG,#00H
                RET

KEEP_ALL:       MOV     R1,#26
                MOV     R0,#A_HOUR
                MOV     DPTR,#R_A_HOUR
     K_A1:      MOV     A,@R0
                MOVX    @DPTR,A
                INC     DPTR
                INC     R0
                DJNZ    R1,K_A1
                RET

OUT_ALL:        MOV     R1,#26
                MOV     R0,#A_HOUR
                MOV     DPTR,#R_A_HOUR
    O_A1:       MOVX    A,@DPTR
                MOV     @R0,A
                INC     DPTR
                INC     R0
                DJNZ    R1,O_A1
                RET

CHECK:          MOV     R1,#11
                MOV     R0,#A_FLAG
 CHK1:          CJNE    @R0,#00H,CHK2
                INC     R0
                DJNZ    R1,CHK1
                RET
 CHK2:          CJNE    @R0,#0FFH,CHK3
                INC     R0
                DJNZ    R1,CHK1
                RET
```

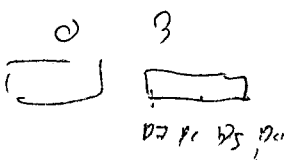
```
CHK3:       MOV     @R0,#00H
            INC     R0
            DJNZ    R1,CHK1
            RET

OUT_AREA:   LCALL   INIT_LCD1
            MOV     R0,#16
            MOV     DPTR,#OUA_DATA1
   OUA1:    CLR     A
            MOVC    A,@A+DPTR
            LCALL   WRITE_CHR
            INC     DPTR
            DJNZ    R0,OUA1

            MOV     A,#40H
            LCALL   GOTODD

            MOV     R0,#16
            MOV     DPTR,#OUA_DATA2
   OUA2:    CLR     A
            MOVC    A,@A+DPTR
            LCALL   WRITE_CHR
            INC     DPTR
            DJNZ    R0,OUA2
            RET

OUA_DATA1:  DB '    OUT OF      '
OUA_DATA2:  DB '  SERVICE AREA!  '

;*************************************************************************
;*************** R  E  A  D  Y     A  L  A  R  M  ************
;*************************************************************************

; *** Check alarm clock ***

READY_TIME: MOV     R1,#HOUR
            MOV     R2,#85H
            LCALL   STATUS
            MOV     R1,#MIN
            MOV     R2,#83H
            LCALL   STATUS
            MOV     R1,#DATE
            MOV     R2,#87H
            LCALL   STATUS
            MOV     R1,#MONTH
            MOV     R2,#89H
            LCALL   STATUS
            MOV     R1,#YEAR
            MOV     R2,#8DH
            LCALL   STATUS
            LCALL   READY_ALARM
            MOV     A,#20
            LCALL   DELAY_SEC
            LJMP    READY_TIME

READY_ALARM: MOV    A,YEAR                  ;Check alarm and time
            CJNE    A,A_YEAR,PASS
            MOV     A,MONTH
            CJNE    A,A_MONTH,PASS
            MOV     A,DATE
            CJNE    A,A_DATE,PASS
```

```
                MOV     A,HOUR
                CJNE    A,A_HOUR,PASS
                MOV     A,MIN
                CJNE    A,A_MIN,PASS
                LCALL   BEEP2                   ;Beep Beep Beep  Beep..
PASS:           RET

STATUS:         MOV     A,R2
                LCALL   COM_TIME
                LCALL   RD_TIME
                RET

;*************************************************************************
;*****************   I   N   T   E   R   R   U   P   T   *****************
;*************************************************************************

INT0_KEY:       JNB     IE0,$                   ;Wait for interrupt INT0
                MOV     A,#255
                LCALL   DELAY_MS
                LJMP    LOOP2                   ;Goto map2

;*************************************************************************
;***************  S    H    O    W     P    L    O    T  *****************
;*************************************************************************

; *** Plot special character pattern on LCD ***

PLOT:           PUSH    ACC
                MOV     R0,#64
                MOV     A,#00H                  ;Set CG RAM address 00
                LCALL   GOTOCG
                MOV     DPTR,#DATA_PLOT
PL1:            CLR     A
                MOVC    A,@A+DPTR
                LCALL   WRITE_CHR
                INC     DPTR
                DJNZ    R0,PL1
                POP     ACC
                RET

DATA_PLOT:      DB 01H,03H,1DH,11H,1DH,03H,01H,00H      ;Speaker
                DB 04H,02H,09H,05H,09H,02H,04H,00H      ;Wave
                DB 0EH,15H,17H,11H,0EH,00H,00H,00H      ;Clock
                DB 1FH,11H,11H,11H,11H,1FH,00H,00H      ;No massage
                DB 06H,09H,02H,04H,0FH,00H,00H,00H      ;2
                DB 02H,06H,0AH,1FH,02H,00H,00H,00H      ;4
                DB 1FH,1BH,11H,1BH,1BH,1FH,00H,00H      ;Massage
                DB 0EH,15H,17H,11H,0EH,11H,00H,00H      ;Alarm clock

SHOW1:          PUSH    ACC
                MOV     A,#00H                  ;Set DD RAM address 00
                LCALL   GOTODD
                MOV     A,#00H                  ;Write 'speaker'
                LCALL   WRITE_CHR
                MOV     A,#20H                  ;Write ' '
                LCALL   WRITE_CHR
                POP     ACC
                RET

SHOW2:          MOV     A,#00H                  ;Set DD RAM address 00
                LCALL   GOTODD
```

```
                LCALL   TIME                    ;Show time
SUB_SHOW2:      MOV     A,#40H
                LCALL   GOTODD
                MOV     A,#00H                  ;Write 'speaker'
                LCALL   WRITE_CHR
                MOV     A,#01H                  ;Write 'wave'
                LCALL   WRITE_CHR
                MOV     A,#' '                  ;Write ' '
                LCALL   WRITE_CHR

SW0:            MOV     A,A_FLAG                ;If alarm off
                CJNE    A,#00H,SW1
                MOV     A,#02H                  ;Write 'clock'
                LCALL   WRITE_CHR
                LJMP    SW2
SW1:
                MOV     A,A_FLAG                ;If alarm on
                CJNE    A,#0FFH,SW0
                MOV     A,#07H                  ;Write 'alarm clock'
                LCALL   WRITE_CHR
SW2:
                MOV     R1,#M1_FLAG
SW3:            CJNE    @R1,#00H,SW4            ;If massage flag=00H
                MOV     A,#03H                  ;Write 'no massage'
                LCALL   WRITE_CHR
                LJMP    SW5
SW4:            CJNE    @R1,#0FFH,SW2           ;If massage flag=0FFH
                MOV     A,#06H                  ;Write 'no massage'
                LCALL   WRITE_CHR
                LJMP    SW5

SW5:            INC     R1
                CJNE    R1,#46H,SW3

                MOV     A,#24H                  ;Write '$'
                LCALL   WRITE_CHR

                MOV     A,#43H                  ;Set DD RAM address 43
                LCALL   GOTODD
                RET

;*************************************************************************
;********************   S     O     U     N     D   **********************
;*************************************************************************

;*** Sound  Beep.....  Beep..... ***

BEEP1:          PUSH    00
                PUSH    01
                PUSH    02
                PUSH    03
                PUSH    ACC
                MOV     R3,#10
BEP:            MOV     R2,#30
                LCALL   BEP0
                MOV     A,#255
                LCALL   DELAY_MS
                DJNZ    R3,BEP
                POP     ACC
                POP     03
                POP     02
```

```
                POP     01
                POP     00
                RET


BEP0:           MOV     R1,#100
BEP1:           CPL     SPEAKER
                MOV     R0,#60
                DJNZ    R0,$
                CPL     SPEAKER
                MOV     R0,#60
                DJNZ    R0,$
                DJNZ    R1,BEP1
                DJNZ    R2,BEP0
BEP2:           RET

; *** Sound Beep Beep Beep  Beep......! ***

BEEP2:          MOV     A_FLAG,#00H
                PUSH    00
                PUSH    01
                PUSH    02
                PUSH    03
                PUSH    ACC
                MOV     R3,#20
BEPP:           MOV     R2,#5
                LCALL   BEP0
                MOV     A,#100
                LCALL   DELAY_MS
                MOV     R2,#5
                LCALL   BEP0
                MOV     A,#100
                LCALL   DELAY_MS
                MOV     R2,#5
                LCALL   BEP0
                MOV     A,#100
                LCALL   DELAY_MS
                MOV     R2,#15
                LCALL   BEP0
                MOV     A,#150
                LCALL   DELAY_MS
                DJNZ    R3,BEPP
                POP     ACC
                MOV     A,#30
                LCALL   DELAY_SEC
                POP     03
                POP     02
                POP     01
                POP     00
                RET


;*************************************************************************
;******************   L            C            D   ********************
;*************************************************************************

; ***  Initial LCD  ***

WAIT:           MOV     A,#15                   ;Wait 15mS
                LCALL   DELAY_MS
                MOV     A,#03H
```

```
                MOV     P1,A
                LCALL   STROBE
                MOV     A,#5                    ;Wait 5mS
                LCALL   DELAY_MS
                MOV     A,#03H
                MOV     P1,A
                LCALL   STROBE
                MOV     A,#1                    ;Wait 1mS
                LCALL   DELAY_MS
                MOV     A,#03H
                MOV     P1,A
                LCALL   STROBE
                LCALL   BUSY
                MOV     A,#02H
                MOV     P1,A
                LCALL   STROBE
                LCALL   BUSY
                MOV     A,#28H
                LCALL   COM_LCD
                MOV     A,#06H
                LCALL   COM_LCD
                RET

; *** Initial LCD for curser off ***

INIT_LCD1:      LCALL   WAIT
                MOV     A,#0CH                  ;Display-on,curser-off
                LCALL   COM_LCD
                MOV     A,#01H                  ;Clear display
                LCALL   COM_LCD
                MOV     A,#5                    ;Delay 5mS
                LCALL   DELAY_MS
                RET

; *** Initial LCD for curser on not blink ***

INIT_LCD2:      LCALL   WAIT
                MOV     A,#0EH                  ;Display-on,curser-on,not
blink
                LCALL   COM_LCD
                MOV     A,#01H                  ;Clear display
                LCALL   COM_LCD
                MOV     A,#5                    ;Delay 5mS
                LCALL   DELAY_MS
                RET

; *** Send command to LCD ***

COM_LCD:        PUSH    ACC
                SWAP    A
                MOV     P1,A
                CLR     RS
                CLR     RW
                LCALL   STROBE
                POP     ACC
                MOV     P1,A
                CLR     RS
                CLR     RW
                LCALL   STROBE
                RET
```



```
; *** Strobe for enable LCD ***

STROBE:         SETB    E
                LCALL   DELAY
                CLR     E
                RET

; *** Write Data To DD or CG RAM  ***

WRITE_CHR:      PUSH    ACC
                SWAP    A
                MOV     P1,A
                SETB    RS
                CLR     RW
                LCALL   STROBE
                POP     ACC
                MOV     P1,A
                SETB    RS
                CLR     RW
                LCALL   STROBE
                RET

; *** Goto DD RAM address of LCD ***

GOTODD:         PUSH    ACC
                SWAP    A
                MOV     P1,A
                CLR     RS
                CLR     RW
                SETB    DB7
                LCALL   STROBE
                POP     ACC
                MOV     P1,A
                CLR     RS
                CLR     RW
                LCALL   STROBE
                RET

; *** Goto CG RAM address of LCD ***

GOTOCG:         PUSH    ACC
                SWAP    A
                MOV     P1,A
                CLR     RS
                CLR     RW
                SETB    DB6
                LCALL   STROBE
                POP     ACC
                MOV     P1,A
                CLR     RS
                CLR     RW
                LCALL   STROBE
                RET

; *** Wait for ready ,check busy flag ***

BUSY:           CLR     RS
                SETB    RW
B1:             LCALL   STROBE
                JB      DB7,B1
                CLR     RW
```

```
                RET

;*******************************************************************************
;*********************  D    E    L    A    Y  ************************
;*******************************************************************************

; *** Delay ***

DELAY:          MOV     R3,#0FH
  DEL:          NOP
                DJNZ    R3,DEL
                RET

; *** Delay ***

DELAY_MS:       PUSH    ACC
                PUSH    B
                MOV     B,#0
DD:             DJNZ    B,$         ;500 uS AT 12 MHZ
                DJNZ    B,$         ;500 uS AT 12 MHZ
                DJNZ    ACC,DD
                POP     B
                POP     ACC
                RET

DELAY_SEC:      PUSH    ACC
                PUSH    B
                MOV     B,A
DDD:            MOV     A,#250
                LCALL   DELAY_MS    ;250  mS
                LCALL   DELAY_MS    ;500  mS
                LCALL   DELAY_MS    ;750  mS
                LCALL   DELAY_MS    ;1000 mS
                DJNZ    B,DDD
                POP     B
                POP     ACC
                RET

;*******************************************************************************
;*********************  R       T       C  ************************
;*******************************************************************************

; *** Initial RTC ***

INIT_RTC:       MOV     HOUR,#00H
                MOV     MIN,#00H
                MOV     SEC,#50H
                MOV     DATE,#01H
                MOV     MONTH,#12H
                MOV     YEAR,#97H

                MOV     A_HOUR,#00H
                MOV     A_MIN,#01H
                MOV     A_DATE,#01H
                MOV     A_MONTH,#12H
                MOV     A_YEAR,#97H
                RET

; *** Set initial to RTC ***

SET_INIT_RTC: CLR       RST
```

```
                LCALL   DELAY
                SETB    SCLK
                LCALL   DELAY
                MOV     R1,#8EH                 ;Open write protect
                MOV     R2,#00H
                LCALL   S_W_TIME
                MOV     R1,#84H                 ;Set hour
                MOV     R2,HOUR
                LCALL   S_W_TIME
                MOV     R1,#82H                 ;Set minute
                MOV     R2,MIN
                LCALL   S_W_TIME
                MOV     R1,#80H                 ;Set sec
                MOV     R2,SEC
                LCALL   S_W_TIME
                MOV     R1,#86H                 ;Set date
                MOV     R2,DATE
                LCALL   S_W_TIME
                MOV     R1,#88H                 ;Set month
                MOV     R2,MONTH
                LCALL   S_W_TIME
                MOV     R1,#8CH                 ;Set year
                MOV     R2,YEAR
                LCALL   S_W_TIME
                MOV     R1,#8EH                 ;Write protect
                MOV     R2,#80H
                LCALL   S_W_TIME
                RET

; *** Show time ***

TIME:           LCALL   READ_TIME               ;Read time
                MOV     A,#04H
                LCALL   WRITE_CHR
                MOV     A,#05H
                LCALL   WRITE_CHR
                MOV     A,#' '
                LCALL   WRITE_CHR
                LCALL   READ_DAY                ;Read date,month,year
                RET

READ_TIME:      PUSH    ACC
                MOV     R1,#HOUR
                MOV     R2,#85H                 ;Read hour
                LCALL   S_R_TIME
                MOV     A,#3AH                  ;":"
                LCALL   WRITE_CHR
                MOV     R1,#MIN
                MOV     R2,#83H                 ;Read minute
                LCALL   S_R_TIME
                MOV     R1,#SEC
                MOV     R2,#81H
                LCALL   COM_TIME
                LCALL   RD_TIME
                POP     ACC
                RET

READ_DAY:       MOV     R1,#DATE
                MOV     R2,#87H                 ;Read date
                LCALL   S_R_TIME
                MOV     A,#2FH                  ;"/"
```

```
                LCALL   WRITE_CHR
                MOV     R1,#MONTH
                MOV     R2,#89H                 ;Read month
                LCALL   S_R_TIME
                MOV     A,#2FH                  ;"/"
                LCALL   WRITE_CHR
                MOV     R1,#YEAR
                MOV     R2,#8DH                 ;Read year
                LCALL   S_R_TIME
                RET

; *** Send command and read time from RTC ***

S_R_TIME:       MOV     A,R2
                LCALL   COM_TIME
                LCALL   RD_TIME
                LCALL   CHANGE
                RET

COM_TIME:       CLR     SCLK
                SETB    RST
                MOV     R0,#08H
      CT:       RRC     A
                MOV     IO,C
                LCALL   PLUSEUP
                DJNZ    R0,CT
                RET

RD_TIME:        MOV     R0,#08H
      RT:       LCALL   PLUSEDW
                MOV     C,IO
                RRC     A
                DJNZ    R0,RT
                MOV     @R1,A
                CLR     RST
                RET

  CHANGE:       PUSH    ACC
                PUSH    ACC                     ;Change data from BCD
                SWAP    A                       ;to character pattern
                SETB    ACC.4
                SETB    ACC.5
                CLR     ACC.6
                CLR     ACC.7
                LCALL   WRITE_CHR
                POP     ACC
                SETB    ACC.4
                SETB    ACC.5
                CLR     ACC.6
                CLR     ACC.7
                LCALL   WRITE_CHR
                POP     ACC
                RET

; *** Send command and write time for RTC ***

S_W_TIME:       PUSH    ACC
                CLR     SCLK
                LCALL   DELAY
                SETB    RST
                LCALL   DELAY
```

```
                MOV     A,R1
                LCALL   WRITE
                MOV     A,R2
                LCALL   WRITE
                CLR     RST
                LCALL   DELAY
                POP     ACC
                RET

    WRITE:      PUSH    ACC
                MOV     R0,#08H
       L2:      RRC     A
                MOV     IO,C
                LCALL   PLUSEDW
                DJNZ    R0,L2
                POP     ACC
                RET

; ***  Clock for RTC ***

 PLUSEUP:       CLR     SCLK
                NOP
                SETB    SCLK
                NOP
                RET

 PLUSEDW:       SETB    SCLK
                NOP
                CLR     SCLK
                NOP
                RET

;*************************************************************************
;************  S   C   A   N       K   E   Y   B   O   R   D  ************
;*************************************************************************

;*** Scan keybord on map 2 ***

MAP2_SCANK:     MOV     A,#43H
                LCALL   GOTODD
    DLY0:       MOV     R4,#10H
    DLY1:       MOV     R5,#00H
    DLY2:       MOV     R6,#00H
    SK0:        JB      KEY2,SK1
                LCALL   BOUNCE
                JB      KEY2,SK0
                LCALL   BOUNCE
                LJMP    SHIFTR
    SK1:        JB      KEY3,SK2
                LCALL   BOUNCE
                JB      KEY3,SK0
                LCALL   BOUNCE
                LJMP    ENTER
    SK2:        JB      KEY1,SK3
                LCALL   BOUNCE
                JB      KEY1,SK0
                LCALL   BOUNCE
                LJMP    SHIFTL
    SK3:        DJNZ    R6,SK0
                DJNZ    R5,DLY2
                DJNZ    R4,DLY1
```

```
                LJMP    RETURN1

  SHIFTR:       PUSH    ACC
                MOV     A,#14H
                LCALL   COM_LCD
                POP     ACC
                INC     A
                CJNE    A,#4FH,SFR0
                MOV     A,#43H
                LCALL   GOTODD
    SFR0:       LJMP    DLY0

  SHIFTL:       PUSH    ACC
                MOV     A,#10H
                LCALL   COM_LCD
                POP     ACC
                DEC     A
                CJNE    A,#42H,SFL0
                MOV     A,#4EH
                LCALL   GOTODD
    SFL0:       LJMP    DLY0

   ENTER:       CJNE    A,#43H,EN1
                LJMP    SET_MODE
     EN1:       CJNE    A,#44H,EN2
                LJMP    MESSAGE1
     EN2:       CJNE    A,#45H,EN3
                LJMP    MESSAGE2
     EN3:       CJNE    A,#46H,EN4
                LJMP    MESSAGE3
     EN4:       CJNE    A,#47H,EN5
                LJMP    MESSAGE4
     EN5:       CJNE    A,#48H,EN6
                LJMP    MESSAGE5
     EN6:       CJNE    A,#49H,EN7
                LJMP    MESSAGE6
     EN7:       CJNE    A,#4AH,EN8
                LJMP    MESSAGE7
     EN8:       CJNE    A,#4BH,EN9
                LJMP    MESSAGE8
     EN9:       CJNE    A,#4CH,EN10
                LJMP    MESSAGE9
     EN10:      CJNE    A,#4DH,EN11
                LJMP    MESSAGE10
     EN11:      CJNE    A,#4EH,EN12
                LJMP    DEL_MESSAGE
     EN12:      LJMP    DLY0


  BOUNCE:       PUSH    ACC
                MOV     A,#80
                LCALL   DELAY_MS
                POP     ACC
                RET

;*************************************************************************
;******************   M   E   S   S   A   G   E   ************************
;*************************************************************************

; *** Message ***
```

```
MESSAGE1:       LCALL   INIT_LCD1
                MOV     A,M1_FLAG
                CJNE    A,#0FFH,SUB_NO_MS
                MOV     DPTR,#M1_LOCATE
                LCALL   SHOW_MS
                LJMP    LOOP2

MESSAGE2:       LCALL   INIT_LCD1
                MOV     A,M2_FLAG
                CJNE    A,#0FFH,SUB_NO_MS
                MOV     DPTR,#M2_LOCATE
                LCALL   SHOW_MS
                LJMP    LOOP2

MESSAGE3:       LCALL   INIT_LCD1
                MOV     A,M3_FLAG
                CJNE    A,#0FFH,SUB_NO_MS
                MOV     DPTR,#M3_LOCATE
                LCALL   SHOW_MS
                LJMP    LOOP2

; *** Sub ***

SUB_NO_MS:      LJMP    NO_MS

; **********

MESSAGE4:       LCALL   INIT_LCD1
                MOV     A,M4_FLAG
                CJNE    A,#0FFH,SUB_NO_MS
                MOV     DPTR,#M4_LOCATE
                LCALL   SHOW_MS
                LJMP    LOOP2

MESSAGE5:       LCALL   INIT_LCD1
                MOV     A,M5_FLAG
                CJNE    A,#0FFH,SUB_NO_MS
                MOV     DPTR,#M5_LOCATE
                LCALL   SHOW_MS
                LJMP    LOOP2

MESSAGE6:       LCALL   INIT_LCD1
                MOV     A,M6_FLAG
                CJNE    A,#0FFH,SUB_NO_MS
                MOV     DPTR,#M6_LOCATE
                LCALL   SHOW_MS
                LJMP    LOOP2

MESSAGE7:       LCALL   INIT_LCD1
                MOV     A,M7_FLAG
                CJNE    A,#0FFH,SUB_NO_MS
                MOV     DPTR,#M7_LOCATE
                LCALL   SHOW_MS
                LJMP    LOOP2

MESSAGE8:       LCALL   INIT_LCD1
                MOV     A,M8_FLAG
                CJNE    A,#0FFH,SUB_NO_MS
                MOV     DPTR,#M8_LOCATE
                LCALL   SHOW_MS
                LJMP    LOOP2
```

```
MESSAGE9:       LCALL   INIT_LCD1
                MOV     A,M9_FLAG
                CJNE    A,#0FFH,SUB_NO_MS
                MOV     DPTR,#M9_LOCATE
                LCALL   SHOW_MS
                LJMP    LOOP2

MESSAGE10:      LCALL   INIT_LCD1
                MOV     A,M10_FLAG
                CJNE    A,#0FFH,SUB_NO_MS
                MOV     DPTR,#M10_LOCATE
                LCALL   SHOW_MS
                LJMP    LOOP2

GET_POINTER2:   MOV     MS_POINTER_L,DPL
                MOV     MS_POINTER_H,DPH
                RET

SHOW_MS:        LCALL   GET_POINTER2
                LCALL   READ_MS
                MOV     A,END_FLAG
                CJNE    A,#0FFH,CMP1
                MOV     END_FLAG,#00H
                LCALL   NEXT_MS
                LJMP    LOOP2
    CMP1:       LCALL   GET_POINTER2
                LCALL   NEXT_MS
                LCALL   READ_MS
                MOV     A,END_FLAG
                CJNE    A,#0FFH,CMP2
                MOV     END_FLAG,#00H
                LCALL   NEXT_MS
                LJMP    LOOP2
    CMP2:       LCALL   GET_POINTER2
                LCALL   NEXT_MS
                LCALL   READ_MS
                MOV     A,END_FLAG
                CJNE    A,#0FFH,CMP3
                MOV     END_FLAG,#00H
                LCALL   NEXT_MS
                LJMP    LOOP2
    CMP3:       LCALL   GET_POINTER2
                LCALL   NEXT_MS
                LCALL   READ_MS
                MOV     A,END_FLAG
                CJNE    A,#0FFH,CMP4
                MOV     END_FLAG,#00H
                LCALL   NEXT_MS
                LJMP    LOOP2
    CMP4:       LCALL   GET_POINTER2
                LCALL   NEXT_MS
                RET

; *** Read massage from RAM ***

READ_MS:        LCALL   INIT_LCD1
                MOV     R0,#16
    MSG0:       MOV     DPL,MS_POINTER_L
                MOV     DPH,MS_POINTER_H
```

```
                MOVX    A,@DPTR
                CJNE    A,#0DH,MSG00
                MOV     END_FLAG,#0FFH
                RET
  MSG00:        LCALL   WRITE_CHR
                INC     DPTR
                LCALL   GET_POINTER2
                DJNZ    R0,MSG0

                MOV     A,#40H
                LCALL   GOTODD

                MOV     R0,#16
   MSG1:        MOV     DPL,MS_POINTER_L
                MOV     DPH,MS_POINTER_H
                MOVX    A,@DPTR
                CJNE    A,#0DH,MSG11
                MOV     END_FLAG,#0FFH
                RET
 MSG11:         LCALL   WRITE_CHR
                INC     DPTR
                LCALL   GET_POINTER2
                DJNZ    R0,MSG1
MSG22:          RET

; *** Scan key3 for next page ***

NEXT_MS:
   NM0:         MOV     R4,#20H
   NM1:         MOV     R5,#00H
   NM2:         MOV     R6,#00H
   NM3:         JB      KEY3,NM4
                LCALL   BOUNCE
                JB      KEY3,NM3
                LCALL   BOUNCE
                RET
   NM4:         DJNZ    R6,NM3
                DJNZ    R5,NM2
                DJNZ    R4,NM1
                RET

; *** No massage ***

NO_MS:          LCALL   INIT_LCD1
                MOV     A,#03H
                LCALL   GOTODD
                MOV     R0,#10
                MOV     DPTR,#NO
 NMS1:          CLR     A
                MOVC    A,@A+DPTR
                LCALL   WRITE_CHR
                INC     DPTR
                DJNZ    R0,NMS1
                MOV     A,#1
                LCALL   DELAY_SEC
                LJMP    LOOP2

NO:             DB 'NO MESSAGE'
```

```
;*************************************************************************
;********** D   E   L   E   T   E       M   E   S   S   A   G   E *******
;*************************************************************************

; *** Delete message ***

DEL_MESSAGE:  LCALL   INIT_LCD2
              MOV     R0,#16
              MOV     DPTR,#DEL_DATA
     DMS1:    CLR     A
              MOVC    A,@A+DPTR
              LCALL   WRITE_CHR
              INC     DPTR
              DJNZ    R0,DMS1
              LCALL   SUB_SHOW2
              LJMP    DEL_SCANK

DEL_DATA:     DB ' Delete Message '

;*** Scan keybord on delete massage map ***

DEL_SCANK:    MOV     A,#43H
              LCALL   GOTODD
   DLY0D:     MOV     R4,#07H
   DLY1D:     MOV     R5,#00H
   DLY2D:     MOV     R6,#00H
    SK0D:     JB      KEY2,SK1D
              LCALL   BOUNCE
              JB      KEY2,SK0D
              LCALL   BOUNCE
              LJMP    SHIFTRD
    SK1D:     JB      KEY3,SK2D
              LCALL   BOUNCE
              JB      KEY3,SK0D
              LCALL   BOUNCE
              LJMP    ENTER_DEL
    SK2D:     JB      KEY1,SK3D
              LCALL   BOUNCE
              JB      KEY1,SK0D
              LCALL   BOUNCE
              LJMP    SHIFTLD
    SK3D:     DJNZ    R6,SK0D
              DJNZ    R5,DLY2D
              DJNZ    R4,DLY1D
              LJMP    LOOP2

  SHIFTRD:    PUSH    ACC
              MOV     A,#14H
              LCALL   COM_LCD
              POP     ACC
              INC     A
              CJNE    A,#4FH,SFR0D
              MOV     A,#43H
              LCALL   GOTODD
   SFR0D:     LJMP    DLY0D

  SHIFTLD:    PUSH    ACC
              MOV     A,#10H
              LCALL   COM_LCD
              POP     ACC
              DEC     A
```

```
                CJNE    A,#42H,SFL0D
                MOV     A,#4EH
                LCALL   GOTODD
    SFL0D:      LJMP    DLY0D

    ENTER_DEL:  CJNE    A,#43H,EN1D
                LJMP    DEL_SCANK
     EN1D:      CJNE    A,#44H,EN2D
                MOV     M1_FLAG,#00H
                LJMP    DEL_MESSAGE
     EN2D:      CJNE    A,#45H,EN3D
                MOV     M2_FLAG,#00H
                LJMP    DEL_MESSAGE
     EN3D:      CJNE    A,#46H,EN4D
                MOV     M3_FLAG,#00H
                LJMP    DEL_MESSAGE
     EN4D:      CJNE    A,#47H,EN5D
                MOV     M4_FLAG,#00H
                LJMP    DEL_MESSAGE
     EN5D:      CJNE    A,#48H,EN6D
                MOV     M5_FLAG,#00H
                LJMP    DEL_MESSAGE
     EN6D:      CJNE    A,#49H,EN7D
                MOV     M6_FLAG,#00H
                LJMP    DEL_MESSAGE
     EN7D:      CJNE    A,#4AH,EN8D
                MOV     M7_FLAG,#00H
                LJMP    DEL_MESSAGE
     EN8D:      CJNE    A,#4BH,EN9D
                MOV     M8_FLAG,#00H
                LJMP    DEL_MESSAGE
     EN9D:      CJNE    A,#4CH,EN10D
                MOV     M9_FLAG,#00H
                LJMP    DEL_MESSAGE
     EN10D:     CJNE    A,#4DH,EN11D
                MOV     M10_FLAG,#00H
                LJMP    DEL_MESSAGE
     EN11D:     CJNE    A,#4EH,EN12D
                LJMP    DEL_SCANK
     EN12D:     LJMP    DLY0D

;*************************************************************************
;**********   S   E   T     D   A   T   E   &   T   I   M   E  *******
;*************************************************************************

; *** Select set time mode or set date mode ***

SET_MODE:       LCALL   INIT_LCD1
                MOV     A,#02H
                LCALL   GOTODD
                MOV     R0,#13
                MOV     DPTR,#S_DATA1
      STM0:     CLR     A
                MOVC    A,@A+DPTR
                LCALL   WRITE_CHR
                INC     DPTR
                DJNZ    R0,STM0
                MOV     A,#42H
                LCALL   GOTODD
                MOV     R0,#13
                MOV     DPTR,#S_DATA2
```

```
STM1:   CLR     A
        MOVC    A,@A+DPTR
        LCALL   WRITE_CHR
        INC     DPTR
        DJNZ    R0,STM1

        MOV     A,#01H
        LCALL   GOTODD
        MOV     A,#7EH
        LCALL   WRITE_CHR

DLY3:   MOV     R4,#12H
DLY4:   MOV     R5,#00H
DLY5:   MOV     R6,#00H
MK0:    JB      KEY3,MT0
        LCALL   BOUNCE
        JB      KEY3,MK0
        LCALL   BOUNCE
        LJMP    SET_T_D

MT0:    JB      KEY1,MT2
        LCALL   BOUNCE
        JB      KEY1,MT0
        LCALL   BOUNCE
        MOV     A,#01H
        LCALL   GOTODD
        MOV     A,#20H
        LCALL   WRITE_CHR
        MOV     A,#41H
        LCALL   GOTODD
        MOV     A,#7EH
        LCALL   WRITE_CHR

DLY6:   MOV     R4,#12H
DLY7:   MOV     R5,#00H
DLY8:   MOV     R6,#00H

MK1:    JB      KEY3,MT1
        LCALL   BOUNCE
        JB      KEY3,MK1
        LCALL   BOUNCE
        LJMP    ALARM_MODE

MT1:    JB      KEY1,MT3
        LCALL   BOUNCE
        JB      KEY1,MT1
        LCALL   BOUNCE
        MOV     A,#01H
        LCALL   GOTODD
        MOV     A,#7EH
        LCALL   WRITE_CHR
        MOV     A,#41H
        LCALL   GOTODD
        MOV     A,#20H
        LCALL   WRITE_CHR

MT2:    DJNZ    R6,MK0
        DJNZ    R5,DLY5
        DJNZ    R4,DLY4
        LJMP    LOOP2
```

```
    MT3:        DJNZ    R6,MK1
                DJNZ    R5,DLY8
                DJNZ    R4,DLY7
                LJMP    LOOP2

S_DATA1:        DB 'Set Time/Date'
S_DATA2:        DB 'Set Alarm    '

; *** Show set time map ***

SET_T_D:        LCALL   SHOW_T_D
                MOV     A,#01H
                LCALL   GOTODD
                LJMP    T_D_SCANK

SHOW_T_D:       LCALL   INIT_LCD2
                MOV     A,#00H                  ;Set DD RAM address 00
                LCALL   GOTODD
                LCALL   READ_TIME               ;Read time
                MOV     A,#04H                  ;Write '2'
                LCALL   WRITE_CHR
                MOV     A,#05H                  ;Write '4'
                LCALL   WRITE_CHR
                MOV     A,#20H                  ;Wrtie ' '
                LCALL   WRITE_CHR
                LCALL   READ_DAY                ;Read date
                LCALL   PRESS
                RET

; *** Show '+' and '^' ***

PRESS:          MOV     A,#40H                  ;Set DD RAM address 40
                LCALL   GOTODD
                MOV     A,#2BH                  ;Write '+'
                LCALL   WRITE_CHR
                MOV     A,#47H                  ;Set DD RAM address 47
                LCALL   GOTODD
                MOV     A,#7EH                  ;Write '>'
                LCALL   WRITE_CHR
                MOV     A,#4FH                  ;Set DD RAM address 4F
                LCALL   GOTODD
                MOV     A,#03H                  ;Write 'Save & Exit'
                LCALL   WRITE_CHR
                MOV     A,#01H                  ;Set DD RAM address 01
                LCALL   GOTODD
                RET

; *** Scan keybord on set time map ***

T_D_SCANK:
    DLY9:       MOV     R4,#12
   DLY10:       MOV     R5,#00
   DLY11:       MOV     R6,#00
    TSK0:       JB      KEY1,TSK1
                LCALL   BOUNCE
                JB      KEY1,TSK0
                LCALL   BOUNCE
                LJMP    UP_HOUR
    TSK1:       JB      KEY3,TSK2
                LCALL   BOUNCE
```

```
                JB      KEY3,TSK0
                LCALL   BOUNCE
                LJMP    SAVE_EXIT
    TSK2:       JB      KEY2,TSK3
                LCALL   BOUNCE
                JB      KEY2,TSK0
                LCALL   BOUNCE
                LJMP    LEFT0
    TSK3:       DJNZ    R6,TSK0
                DJNZ    R5,DLY11
                DJNZ    R4,DLY10
                LJMP    LOOP2

; *** When press '>' ***

   LEFT0:       INC     A
                INC     A
                INC     A
                CJNE    A,#07,LF00
                LJMP    LF02
    LF00:       CJNE    A,#12H,LF01
                MOV     A,#01H
                LCALL   GOTODD
                LJMP    T_D_SCANK
    LF01:       LCALL   UP3
                LJMP    T_D_SCANK
    LF02:       INC     A
                INC     A
                LCALL   UP2
                LJMP    LF01

; *** When press '+' ***

   UP_HOUR:     CJNE    A,#01H,UP_MIN
                MOV     A,HOUR
                CJNE    A,#23H,UPH
                MOV     HOUR,#00H
                LJMP    SET_TIME_H
        UPH:    INC     A
                LCALL   SETH
                CJNE    A,#0FAH,GETHL
                MOV     A,HOUR
                SWAP    A
                INC     A
                LCALL   CLRH
                CJNE    A,#03H,GETHH
                MOV     HOUR,#20H
                LJMP    SET_TIME_H
    GETHH:      SWAP    A
                MOV     HOUR,A
                LJMP    SET_TIME_H
    GETHL:      PUSH    ACC
                MOV     A,HOUR
                CALL    SETL
                MOV     HOUR,A
                POP     ACC
                ANL     HOUR,A
                LJMP    SET_TIME_H

   UP_MIN:      CJNE    A,#04H,UP_DATE
                MOV     A,MIN
```

```
                CJNE    A,#59H,UPMI
                MOV     MIN,#00H
                LJMP    SET_TIME_MI
      UPMI:     INC     A
                LCALL   SETH
                CJNE    A,#0FAH,GETMIL
                MOV     A,MIN
                SWAP    A
                INC     A
                LCALL   CLRH
                CJNE    A,#06H,GETMIH
                MOV     MIN,#50H
                LJMP    SET_TIME_MI
    GETMIH:     SWAP    A
                MOV     MIN,A
                LJMP    SET_TIME_MI
    GETMIL:     PUSH    ACC
                MOV     A,MIN
                CALL    SETL
                MOV     MIN,A
                POP     ACC
                ANL     MIN,A
                LJMP    SET_TIME_MI

    UP_DATE:    CJNE    A,#09H,UP_MONTH
                MOV     A,DATE
                CJNE    A,#31H,UPD
                MOV     DATE,#01H
                LJMP    SET_DATE_D
        UPD:    INC     A
                LCALL   SETH
                CJNE    A,#0FAH,GETDL
                MOV     A,DATE
                SWAP    A
                INC     A
                LCALL   CLRH
                CJNE    A,#04H,GETDH
                MOV     DATE,#30H
                LJMP    SET_DATE_D
     GETDH:     SWAP    A
                MOV     DATE,A
                LJMP    SET_DATE_D
     GETDL:     PUSH    ACC
                MOV     A,DATE
                CALL    SETL
                MOV     DATE,A
                POP     ACC
                ANL     DATE,A
                LJMP    SET_DATE_D

    UP_MONTH:   CJNE    A,#0CH,UP_YEAR
                MOV     A,MONTH
                CJNE    A,#12H,UPM
                MOV     MONTH,#01H
                LJMP    SET_DATE_M
      UPM:      INC     A
                LCALL   SETH
                CJNE    A,#0FAH,GETML
                MOV     A,MONTH
                SWAP    A
                INC     A
```

```
                LCALL   CLRH
                CJNE    A,#02H,GETMH
                MOV     MONTH,#10H
                LJMP    SET_DATE_M
    GETMH:      SWAP    A
                MOV     MONTH,A
                LJMP    SET_DATE_M
    GETML:      PUSH    ACC
                MOV     A,MONTH
                CALL    SETL
                MOV     MONTH,A
                POP     ACC
                ANL     MONTH,A
                LJMP    SET_DATE_M

  UP_YEAR:      CJNE    A,#0FH,SUB_TD_SCANK
                MOV     A,YEAR
                CJNE    A,#20H,UPY0
                MOV     YEAR,#97H
                LJMP    SET_DATE_Y
     UPY0:      CJNE    A,#99H,UPY
                MOV     YEAR,#00H
                LJMP    SET_DATE_Y
      UPY:      INC     A
                LCALL   SETH
                CJNE    A,#0FAH,GETYL
                MOV     A,YEAR
                SWAP    A
                INC     A
                LCALL   CLRH
                CJNE    A,#03H,GETYH
                MOV     YEAR,#20H
                LJMP    SET_DATE_Y
     GETYH:     SWAP    A
                MOV     YEAR,A
                LJMP    SET_DATE_Y
     GETYL:     PUSH    ACC
                MOV     A,YEAR
                CALL    SETL
                MOV     YEAR,A
                POP     ACC
                ANL     YEAR,A
                LJMP    SET_DATE_Y

  ; *** Sub ***

  SUB_TD_SCANK:  LJMP     T_D_SCANK

  ; *** Set time  ***

  SET_TIME_H:    LCALL    SET_INIT_RTC
                 LCALL    SHOW_T_D
                 MOV      A,#01H
                 LCALL    GOTODD
                 LJMP     T_D_SCANK

  SET_TIME_MI:   LCALL    SET_INIT_RTC
                 LCALL    SHOW_T_D
                 MOV      A,#04H
                 LCALL    GOTODD
                 LJMP     T_D_SCANK
```

```
SET_DATE_D:   LCALL   SET_INIT_RTC
              LCALL   SHOW_T_D
              MOV     A,#09H                  ;Set DD RAM address 05
              LCALL   GOTODD
              LJMP    T_D_SCANK

SET_DATE_M:   LCALL   SET_INIT_RTC
              LCALL   SHOW_T_D
              MOV     A,#0CH                  ;Set DD RAM address 08
              LCALL   GOTODD
              LJMP    T_D_SCANK

SET_DATE_Y:   LCALL   SET_INIT_RTC
              LCALL   SHOW_T_D
              MOV     A,#0FH                  ;Set DD RAM address 0B
              LCALL   GOTODD
              LJMP    T_D_SCANK

; *** Save time/date and exit ***

 SAVE_EXIT:   LCALL   SET_INIT_RTC
              LJMP    LOOP2

;*************************************************************************
;***************   S   E   T     A   L   A   R   M    ********************
;*************************************************************************

; *** Select alarm on/off mode ***

ALARM_MODE:   LCALL   INIT_LCD1
              MOV     A,#04H
              LCALL   GOTODD
              MOV     R0,#9
              MOV     DPTR,#A_DATA1
    ALM0:     CLR     A
              MOVC    A,@A+DPTR
              LCALL   WRITE_CHR
              INC     DPTR
              DJNZ    R0,ALM0
              MOV     A,#44H
              LCALL   GOTODD
              MOV     R0,#9
              MOV     DPTR,#A_DATA2
     ALM1:    CLR     A
              MOVC    A,@A+DPTR
              LCALL   WRITE_CHR
              INC     DPTR
              DJNZ    R0,ALM1

              MOV     A,#03H
              LCALL   GOTODD
              MOV     A,#7EH
              LCALL   WRITE_CHR

 DLY12:       MOV     R4,#12
 DLY13:       MOV     R5,#00
 DLY14:       MOV     R6,#00
   AK0:       JB      KEY3,AT0
              LCALL   BOUNCE
              JB      KEY3,AK0
```

```
                LCALL   BOUNCE
                LJMP    SET_ALARM

     AT0:       JB      KEY1,AT2
                LCALL   BOUNCE
                JB      KEY1,AT0
                LCALL   BOUNCE
                MOV     A,#03H
                LCALL   GOTODD
                MOV     A,#20H
                LCALL   WRITE_CHR
                MOV     A,#43H
                LCALL   GOTODD
                MOV     A,#7EH
                LCALL   WRITE_CHR

   DLY15:       MOV     R4,#12H
   DLY16:       MOV     R5,#00H
   DLY17:       MOV     R6,#00H

     AK1:       JB      KEY3,AT1
                LCALL   BOUNCE
                JB      KEY3,AK1
                LCALL   BOUNCE
                MOV     A_FLAG,#00H
                LJMP    LOOP2

     AT1:       JB      KEY1,AT3
                LCALL   BOUNCE
                JB      KEY1,AT1
                LCALL   BOUNCE
                MOV     A,#03H
                LCALL   GOTODD
                MOV     A,#7EH
                LCALL   WRITE_CHR
                MOV     A,#43H
                LCALL   GOTODD
                MOV     A,#20H
                LCALL   WRITE_CHR

     AT2:       DJNZ    R6,AK0
                DJNZ    R5,DLY14
                DJNZ    R4,DLY13
                LJMP    LOOP2

     AT3:       DJNZ    R6,AK1
                DJNZ    R5,DLY17
                DJNZ    R4,DLY16
                LJMP    LOOP2

   A_DATA1:     DB 'Alarm ON '
   A_DATA2:     DB 'Alarm OFF'

   ;*************************************************************************
   ;***************   A   L   A   R   M         M   A   P   ******************
   ;*************************************************************************

   ; *** Show set alarm map ***

   SET_ALARM:   LCALL   SHOW_ALARM
                MOV     A,#01H
```

```
                LCALL   GOTODD
                LJMP    AL_SCANK

SHOW_ALARM:     LCALL   INIT_LCD2
                MOV     A,#00H                  ;Set DD RAM address 04
                LCALL   GOTODD
                LCALL   READ_ALARM              ;Read date,month,year
                LCALL   PRESS
                RET

; *** Initial alarm ***

; *** Read alarm buffer and show set alarm ***

READ_ALARM:     MOV     A,A_HOUR
                LCALL   CHANGE
                MOV     A,#':'
                LCALL   WRITE_CHR
                MOV     A,A_MIN
                LCALL   CHANGE
                MOV     A,#04H
                LCALL   WRITE_CHR
                MOV     A,#05H
                LCALL   WRITE_CHR
                MOV     A,#' '
                LCALL   WRITE_CHR
                MOV     A,A_DATE
                LCALL   CHANGE
                MOV     A,#'/'
                LCALL   WRITE_CHR
                MOV     A,A_MONTH
                LCALL   CHANGE
                MOV     A,#'/'
                LCALL   WRITE_CHR
                MOV     A,A_YEAR
                LCALL   CHANGE
                RET

; *** Scan keybord on alarm map ***

AL_SCANK:
  DLY18:        MOV     R4,#12H
  DLY19:        MOV     R5,#00H
  DLY20:        MOV     R6,#00H

    DSK0:       JB      KEY1,DSK1
                LCALL   BOUNCE
                JB      KEY1,DSK0
                LCALL   BOUNCE
                LJMP    UP_A_HOUR
    DSK1:       JB      KEY3,DSK2
                LCALL   BOUNCE
                JB      KEY3,DSK1
                LCALL   BOUNCE
                LJMP    SAVE_ALARM
    DSK2:       JB      KEY2,DSK3
                LCALL   BOUNCE
                JB      KEY2,DSK2
                LCALL   BOUNCE
                LJMP    LEFT1
    DSK3:
```

```
                DJNZ    R6,DSK0
                DJNZ    R5,DLY20
                DJNZ    R4,DLY19
                LJMP    LOOP2

; *** When press key '>' ***

  LEFT1:        INC     A
                INC     A
                INC     A
                CJNE    A,#07,LF10
                LJMP    LF12
   LF10:        CJNE    A,#12H,LF11
                MOV     A,#01H
                LCALL   GOTODD
                LJMP    AL_SCANK
   LF11:        LCALL   UP3
                LJMP    AL_SCANK
   LF12:        INC     A
                INC     A
                LCALL   UP2
                LJMP    LF11

UP_A_HOUR:      CJNE    A,#01H,UP_A_MIN
                MOV     A,A_HOUR
                CJNE    A,#23H,UPAH
                MOV     A_HOUR,#00H
                LJMP    SET_ALARM_H
     UPAH:      INC     A
                LCALL   SETH
                CJNE    A,#0FAH,GETAHL
                MOV     A,A_HOUR
                SWAP    A
                INC     A
                LCALL   CLRH
                CJNE    A,#03H,GETAHH
                MOV     A_HOUR,#20H
                LJMP    SET_TIME_H
  GETAHH:       SWAP    A
                MOV     A_HOUR,A
                LJMP    SET_ALARM_H
  GETAHL:       PUSH    ACC
                MOV     A,A_HOUR
                CALL    SETL
                MOV     A_HOUR,A
                POP     ACC
                ANL     A_HOUR,A
                LJMP    SET_ALARM_H

UP_A_MIN:       CJNE    A,#04H,UP_A_DATE
                MOV     A,A_MIN
                CJNE    A,#59H,UPAMI
                MOV     A_MIN,#00H
                LJMP    SET_ALARM_MI
     UPAMI:     INC     A
                LCALL   SETH
                CJNE    A,#0FAH,GETAMIL
                MOV     A,A_MIN
                SWAP    A
                INC     A
                LCALL   CLRH
```

```
                CJNE    A,#06H,GETAMIH
                MOV     A_MIN,#50H
                LJMP    SET_ALARM_MI
  GETAMIH:      SWAP    A
                MOV     A_MIN,A
                LJMP    SET_ALARM_MI
  GETAMIL:      PUSH    ACC
                MOV     A,A_MIN
                CALL    SETL
                MOV     A_MIN,A
                POP     ACC
                ANL     A_MIN,A
                LJMP    SET_ALARM_MI

UP_A_DATE:      CJNE    A,#09H,UP_A_MONTH
                MOV     A,A_DATE
                CJNE    A,#31H,UPAD
                MOV     A_DATE,#01H
                LJMP    SET_ALARM_D
      UPAD:     INC     A
                LCALL   SETH
                CJNE    A,#0FAH,GETADL
                MOV     A,A_DATE
                SWAP    A
                INC     A
                LCALL   CLRH
                CJNE    A,#04H,GETADH
                MOV     A_DATE,#30H
                LJMP    SET_ALARM_D
   GETADH:      SWAP    A
                MOV     A_DATE,A
                LJMP    SET_ALARM_D
   GETADL:      PUSH    ACC
                MOV     A,A_DATE
                CALL    SETL
                MOV     A_DATE,A
                POP     ACC
                ANL     A_DATE,A
                LJMP    SET_ALARM_D

UP_A_MONTH:     CJNE    A,#0CH,UP_A_YEAR
                MOV     A,A_MONTH
                CJNE    A,#12H,UPAM
                MOV     A_MONTH,#01H
                LJMP    SET_ALARM_M
     UPAM:      INC     A
                LCALL   SETH
                CJNE    A,#0FAH,GETAML
                MOV     A,A_MONTH
                SWAP    A
                INC     A
                LCALL   CLRH
                CJNE    A,#02H,GETAMH
                MOV     A_MONTH,#10H
                LJMP    SET_ALARM_M
   GETAMH:      SWAP    A
                MOV     A_MONTH,A
                LJMP    SET_ALARM_M
   GETAML:      PUSH    ACC
                MOV     A,A_MONTH
                CALL    SETL
```

```
                MOV     A_MONTH,A
                POP     ACC
                ANL     A_MONTH,A
                LJMP    SET_ALARM_M

UP_A_YEAR:      CJNE    A,#0FH,SUB_AL_SCANK
                MOV     A,A_YEAR
                CJNE    A,#20H,UPAY0
                MOV     A_YEAR,#97H
                LJMP    SET_ALARM_Y
  UPAY0:        CJNE    A,#99H,UPAY
                MOV     A_YEAR,#00H
                LJMP    SET_ALARM_Y
   UPAY:        INC     A
                LCALL   SETH
                CJNE    A,#0FAH,GETAYL
                MOV     A,A_YEAR
                SWAP    A
                INC     A
                LCALL   CLRH
                CJNE    A,#03H,GETAYH
                MOV     A_YEAR,#20H
                LJMP    SET_ALARM_Y
  GETAYH:       SWAP    A
                MOV     A_YEAR,A
                LJMP    SET_ALARM_Y
  GETAYL:       PUSH    ACC
                MOV     A,A_YEAR
                CALL    SETL
                MOV     A_YEAR,A
                POP     ACC
                ANL     A_YEAR,A
                LJMP    SET_ALARM_Y

; *** Set alarm  ***

SET_ALARM_H:    LCALL   SHOW_ALARM
                MOV     A,#01H                  ;Set DD RAM address 01
                LCALL   GOTODD
                LJMP    AL_SCANK

SET_ALARM_MI:   LCALL   SHOW_ALARM
                MOV     A,#04H                  ;Set DD RAM address 04
                LCALL   GOTODD
                LJMP    AL_SCANK

SET_ALARM_D:    LCALL   SHOW_ALARM
                MOV     A,#09H                  ;Set DD RAM address 09
                LCALL   GOTODD
                LJMP    AL_SCANK

SET_ALARM_M:    LCALL   SHOW_ALARM
                MOV     A,#0CH                  ;Set DD RAM address 0C
                LCALL   GOTODD
                LJMP    AL_SCANK

SET_ALARM_Y:    LCALL   SHOW_ALARM
                MOV     A,#0FH                  ;Set DD RAM address 0F
                LCALL   GOTODD
                LJMP    AL_SCANK
```

```
; *** Sub ***

SUB_AL_SCANK: LJMP    AL_SCANK

; *** Save and exit ***

SAVE_ALARM:   MOV     A_FLAG,#0FFH
              LJMP    LOOP2

; *** Inc curser right 3 block ***

    UP3:      PUSH    ACC
              MOV     A,#14H
              LCALL   COM_LCD
              MOV     A,#14H
              LCALL   COM_LCD
              MOV     A,#14H
              LCALL   COM_LCD
              POP     ACC
              RET

    UP2:      PUSH    ACC
              MOV     A,#14H
              LCALL   COM_LCD
              MOV     A,#14H
              LCALL   COM_LCD
              POP     ACC
              RET

; *** Set and clear accumulator ***

    SETH:     SETB    ACC.7
              SETB    ACC.6
              SETB    ACC.5
              SETB    ACC.4
              RET

    SETL:     SETB    ACC.0
              SETB    ACC.1
              SETB    ACC.2
              SETB    ACC.3
              RET

    CLRH:     CLR     ACC.7
              CLR     ACC.6
              CLR     ACC.5
              CLR     ACC.4
              RET

              END
```

19-0094; Rev 6; 7/95

MAXIM

Microprocessor Supervisory Circuits

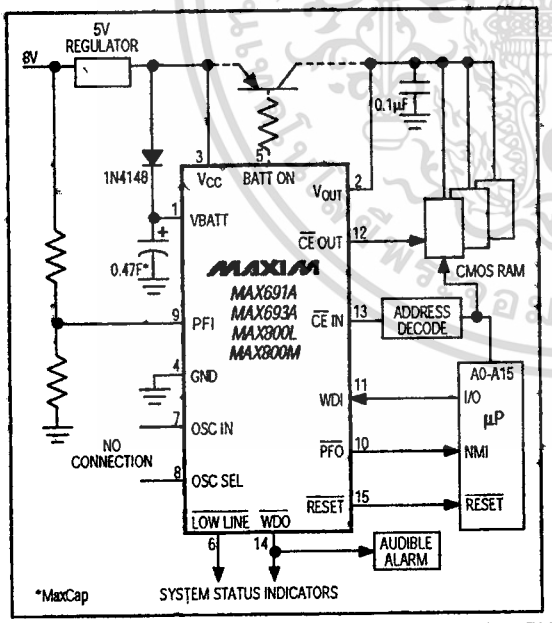
General Description

The MAX691A/MAX693A/MAX800L/MAX800M microprocessor (µP) supervisory circuits are pin-compatible upgrades to the MAX691, MAX693, and MAX695. They improve performance with 30µA supply current, 200ms typ reset active delay on power-up, and 6ns chip-enable propagation delay. Features include write protection of CMOS RAM or EEPROM, separate watchdog outputs, backup-battery switchover, and a $\overline{RESET}$ output that is valid with V_{CC} down to 1V. The MAX691A/MAX800L have a 4.65V typical reset-threshold voltage, and the MAX693A/MAX800M's reset threshold is 4.4V typical. The MAX800L/MAX800M guarantee power-fail accuracies to ±2%.

Applications

Computers
Controllers
Intelligent Instruments
Automotive Systems
Critical µP Power Monitoring

Typical Operating Circuit



Features

- ♦ 200ms Power-OK/Reset Timeout Period
- ♦ 1µA Standby Current, 30µA Operating Current
- ♦ On-Board Gating of Chip-Enable Signals, 10ns Max Delay
- ♦ MaxCap™ or SuperCap™ Compatible
- ♦ Guaranteed $\overline{RESET}$ Assertion to V_{CC} = 1V
- ♦ Voltage Monitor for Power-Fail or Low-Battery Warning
- ♦ Power-Fail Accuracy Guaranteed to ±2% (MAX800L/M)
- ♦ Available in 16-Pin Narrow SO and Plastic DIP Packages

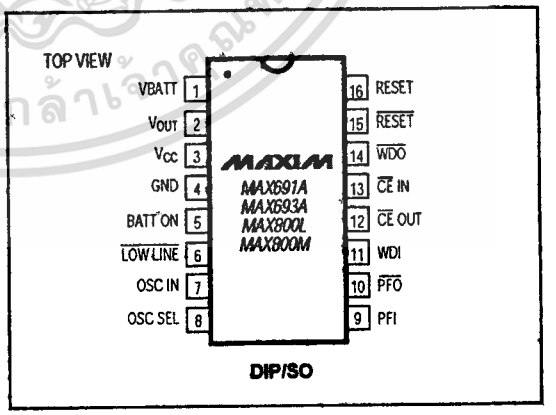
Ordering Information

PART	TEMP. RANGE	PIN-PACKAGE
MAX691ACPE	0 °C to +70 °C	16 Plastic DIP
MAX691ACSE	0 °C to +70 °C	16 Narrow SO
MAX691ACWE	0 °C to +70 °C	16 Wide SO
MAX691AC/D	0 °C to +70 °C	Dice*
MAX691AEPE	-40 °C to +85 °C	16 Plastic SO
MAX691AESE	-40 °C to +85 °C	16 Narrow SO
MAX691AEWE	-40 °C to +85 °C	16 Wide SO
MAX691AEJE	-40 °C to +85 °C	16 CERDIP
MAX691AMJE	-55 °C to +125 °C	16 CERDIP

Ordering Information continued on last page.

** Dice are specified at T_A = +25 °C.*

Pin Configuration



™ SuperCap is a registered trademark of Baknor Industries. ™ MaxCap is a registered trademark of The Carborundum Corp.

Call toll free 1-800-998-8800 for free samples or literature.

Pin Description

PIN	NAME	FUNCTION
1	VBATT	Battery-Backup Input. Connect to external battery or capacitor and charging circuit. If backup battery is not used, connect to GND.
2	V_{OUT}	Output Supply Voltage. When V_{CC} is greater than VBATT and above the reset threshold, V_{OUT} connects to V_{CC}. When V_{CC} falls below VBATT and is below the reset threshold, V_{OUT} connects to VBATT. Connect a 0.1µF capacitor from V_{OUT} to GND. Connect V_{OUT} to V_{CC} if no backup battery is used.
3	V_{CC}	Input Supply Voltage, 5V input.
4	GND	Ground. 0V reference for all signals.
5	BATT ON	Battery On Output. When V_{OUT} switches to VBATT, BATT ON goes high. When V_{OUT} switches to V_{CC}, BATT ON goes low. Connect the base of a PNP through a current-limiting resistor to BATT ON for V_{OUT} current requirements greater than 250mA.
6	$\overline{\text{LOW LINE}}$	$\overline{\text{LOW LINE}}$ output goes low when V_{CC} falls below the reset threshold. It returns high as soon as V_{CC} rises above the reset threshold.
7	OSC IN	External Oscillator Input. When OSC SEL is unconnected or driven high, a 10µA pull-up connects from V_{OUT} to OSC IN, the internal oscillator sets the reset and watchdog timeout periods, and OSC IN selects between fast and slow watchdog timeout periods. When OSC SEL is driven low, the reset and watchdog timeout periods may be set either by a capacitor from OSC IN to ground or by an external clock at OSC IN (see Figure 3.)
8	OSC SEL	Oscillator Select. When OSC SEL is unconnected or driven high, the internal oscillator sets the reset delay and watchdog timeout period. When OSC SEL is low, the external oscillator input (OSC IN) is enabled (see Table 1). OSC SEL has a 10µA internal pull-up.
9	PFI	Power-Fail Input. This is the noninverting input to the power-fail comparator. When PFI is less than 1.25V, $\overline{\text{PFO}}$ goes low. When PFI is not used, connect PFI to GND or V_{OUT}.
10	$\overline{\text{PFO}}$	Power-Fail Output. This is the output of the power-fail comparator. $\overline{\text{PFO}}$ goes low when PFI is less than 1.25V. This is an uncommitted comparator, and has no effect on any other internal circuitry.
11	WDI	Watchdog Input. WDI is a three-level input. If WDI remains either high or low for longer than the watchdog timeout period, $\overline{\text{WDO}}$ goes low and reset is asserted for the reset timeout period. $\overline{\text{WDO}}$ remains low until the next transition at WDI. Leaving WDI unconnected disables the watchdog function. WDI connects to an internal voltage divider between V_{OUT} and GND, which sets it to mid-supply when left unconnected.
12	$\overline{\text{CE}}$ OUT	Chip-Enable Output. $\overline{\text{CE}}$ OUT goes low only when $\overline{\text{CE}}$ IN is low and V_{CC} is above the reset threshold. If CE IN is low when reset is asserted, $\overline{\text{CE}}$ OUT will stay low for 15µs or until $\overline{\text{CE}}$ IN goes high, whichever occurs first.
13	$\overline{\text{CE}}$ IN	Chip-Enable Input. The input to chip-enable gating circuit. If $\overline{\text{CE}}$ IN is not used, connect $\overline{\text{CE}}$ IN to GND or V_{OUT}.
14	$\overline{\text{WDO}}$	Watchdog Output. If WDI remains high or low longer than the watchdog timeout period, $\overline{\text{WDO}}$ goes low and reset is asserted for the reset timeout period. $\overline{\text{WDO}}$ returns high on the next transition at WDI. $\overline{\text{WDO}}$ remains high if WDI is unconnected.
15	$\overline{\text{RESET}}$	$\overline{\text{RESET}}$ Output goes low whenever V_{CC} falls below the reset threshold. $\overline{\text{RESET}}$ will remain low typically for 200ms after V_{CC} crosses the reset threshold on power-up.
16	RESET	RESET is an active-high output. It is open drain, and the inverse of $\overline{\text{RESET}}$.

Detailed Description

RESET and RESET Outputs

The MAX691A/MAX693A/MAX800L/MAX800M's $\overline{\text{RESET}}$ and RESET outputs ensure that the µP (with reset inputs asserted either high or low) powers up in a known state, and prevents code-execution errors during power-down or brownout conditions.

The $\overline{\text{RESET}}$ output is active low, and typically sinks 3.2mA at 0.1V saturation voltage in its active state. When deasserted, $\overline{\text{RESET}}$ sources 1.6mA at typically V_{OUT} - 0.5V. RESET output is open drain, active high, and typically sinks 3.2mA with a saturation voltage of 0.1V. When no backup battery is used, $\overline{\text{RESET}}$ output is guaranteed to be valid down to V_{CC} = 1V, and an external 10kΩ pull-down resistor on $\overline{\text{RESET}}$ insures that it will be valid with V_{CC} down to GND (Figure 1). As V_{CC} goes below 1V, the gate drive to the $\overline{\text{RESET}}$ output switch reduces accordingly, increasing the $r_{DS(ON)}$ and the saturation voltage. The 10kΩ pull-down resistor insures the parallel combination of switch plus resistor is around 10kΩ and the output saturation voltage is below 0.4V while sinking 40µA. When using a 10kΩ external pull-down resistor, the high state for $\overline{\text{RESET}}$ output with V_{CC} = 4.75V will be 4.5V typical. For battery voltages ≥ 2V connected to VBATT, RESET and $\overline{\text{RESET}}$ remain valid for V_{CC} from 0V to 5.5V.

Microprocessor Supervisory Circuits

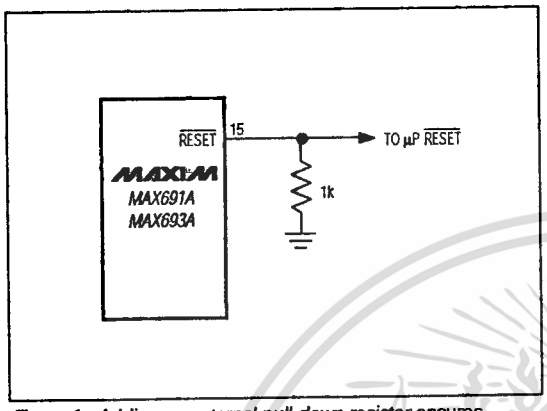


Figure 1. Adding an external pull-down resistor ensures RESET is valid with V_{CC} down to GND.

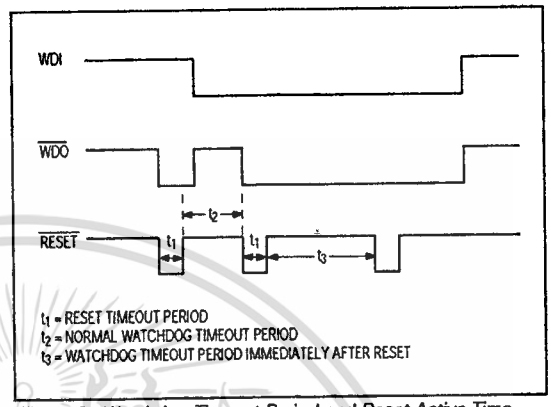


Figure 2. Watchdog Timeout Period and Reset Active Time

RESET and $\overline{RESET}$ are asserted when V_{CC} falls below the reset threshold (4.65V for the MAX691A/MAX800L, 4.4V for the MAX693A/MAX800M) and remain asserted for 200ms typ after V_{CC} rises above the reset threshold on power-up (Figure 5). The devices' battery-switchover comparator does not affect reset assertion. However, both reset outputs are asserted in battery-backup mode since V_{CC} must be below the reset threshold to enter this mode.

Watchdog Function

The watchdog monitors µP activity via the Watchdog Input (WDI). If the µP becomes inactive, $\overline{RESET}$ and RESET are asserted. To use the watchdog function, connect WDI to a bus line or µP I/O line. If WDI remains high or low for longer than the watchdog timeout period (1.6sec nominal), $\overline{WDO}$, RESET, and $\overline{RESET}$ are asserted (see *RESET and $\overline{RESET}$ Outputs* section, and the *Watchdog Output* discussion on this page).

Watchdog Input

A change of state (high to low, low to high, or a minimum 100ns pulse) at the Watchdog Input (WDI) during the watchdog period resets the watchdog timer. The watchdog default timeout is 1.6sec.

To disable the watchdog function, leave WDI floating. An internal resistor network (100kΩ equivalent impedance at WDI) biases WDI to approximately 1.6V. Internal comparators detect this level and disable the watchdog timer. When V_{CC} is below the reset threshold, the watchdog function is disabled and WDI is disconnected from its internal resistor network, thus becoming high impedance.

Watchdog Output

The Watchdog Output ($\overline{WDO}$) remains high if there is a transition or pulse at WDI during the watchdog timeout period. The watchdog function is disabled and $\overline{WDO}$ is a logic high when V_{CC} is below the reset threshold, battery-backup mode is enabled, or WDI is an open circuit. In watchdog mode, if no transition occurs at WDI during the watchdog timeout period, RESET and $\overline{RESET}$ are asserted for the reset timeout period (200ms typical). $\overline{WDO}$ goes low and remains low until the next transition at WDI (Figure 2). If WDI is held high or low indefinitely, RESET and $\overline{RESET}$ will generate 200ms pulses every 1.6sec. $\overline{WDO}$ has a 2 x TTL output characteristic.

Selecting an Alternative Watchdog and Reset Timeout Period

The OSC SEL and OSC IN inputs control the watchdog and reset timeout periods. Floating OSC SEL and OSC IN or tying them both to V_{OUT} selects the nominal 1.6sec watchdog timeout period and 200ms reset timeout period. Connecting OSC IN to GND and floating or connecting OSC SEL to V_{OUT} selects the 100ms normal watchdog timeout delay and 1.6sec delay immediately after reset. The reset timeout delay remains 200ms (Figure 2). Select alternative timeout periods by connecting OSC SEL to GND and connecting a capacitor between OSC IN and GND, or by externally driving OSC IN (Table 1 and Figure 3). OSC IN is internally connected to a ±100nA (typ) current source that charges and discharges the timing capacitor to create the oscillator frequency, which sets the reset and watchdog timeout periods (see *Connecting a Timing Capacitor at OSC IN* in the *Applications Information* section).

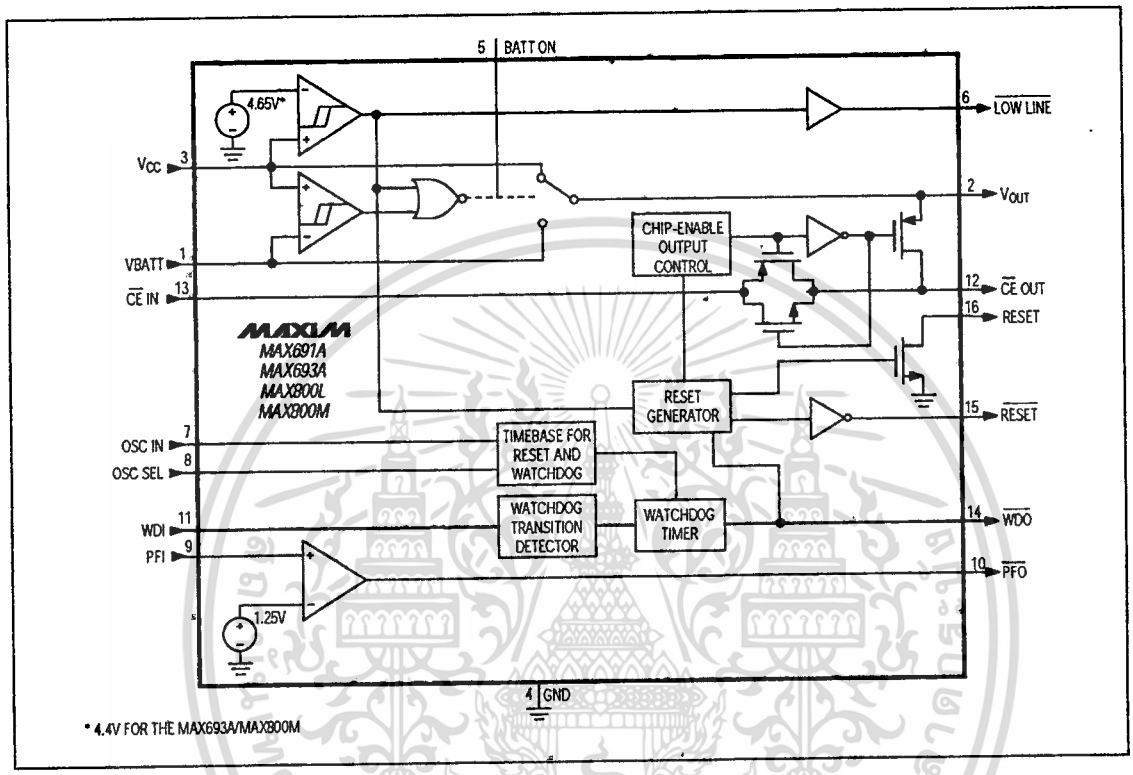


Figure 4. MAX691A/MAX693A/MAX800L/MAX800M Block Diagram

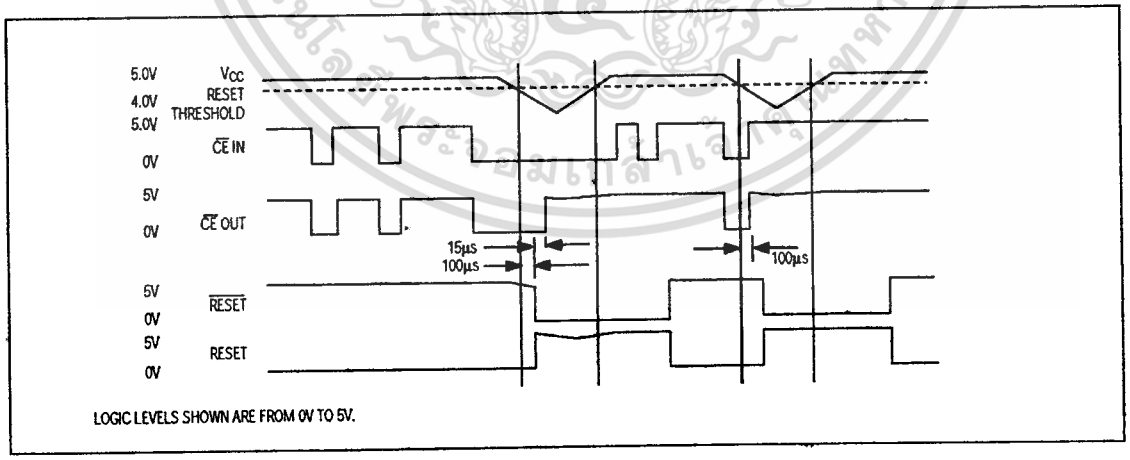


Figure 5. Reset and Chip-Enable Timing

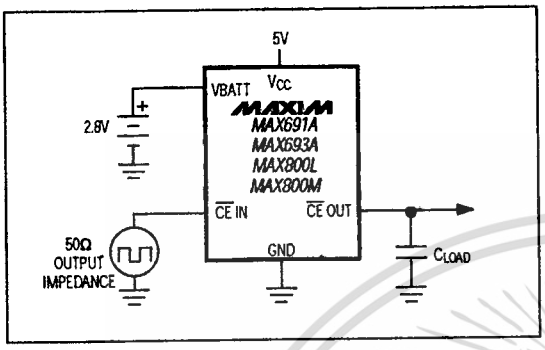


Figure 6. CE Propagation Delay Test Circuit

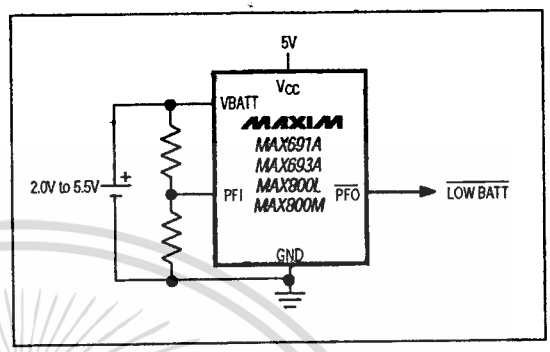


Figure 7. Low-Battery Indicator

Table 2. Input and Output Status in Battery-Backup Mode

PIN	NAME	STATUS
1	VBATT	Supply Current is 1µA max.
2	VOUT	VOUT is connected to VBATT through an internal PMOS switch.
3	VCC	Battery switchover comparator monitors VCC for active switchover.
4	GND	GND 0V, 0V reference for all signals.
5	BATT ON	Logic high. The open-circuit output is equal to VOUT.
6	$\overline{\text{LOWLINE}}$	Logic Low*
7	OSC IN	OSC IN is ignored.
8	OSC SEL	OSC SEL is ignored.
9	PFI	The power-fail comparator remains active in the battery-backup mode for VCC ≥ VBATT - 1.2V typ.
10	$\overline{\text{PFO}}$	The power-fail comparator remains active in the battery-backup mode for VCC ≥ VBATT - 1.2V typ. Below this voltage, PFO is forced low.
11	WDI	Watchdog is ignored.
12	$\overline{\text{CE}}$ OUT	Logic high. The open-circuit voltage is equal to VOUT.
13	$\overline{\text{CE}}$ IN	High Impedance.
14	$\overline{\text{WDO}}$	Logic high. The open-circuit voltage is equal to VOUT.
15	$\overline{\text{RESET}}$	Logic low*
16	RESET	High Impedance*

** V_{CC} must be below the reset threshold to enter battery-backup mode.*

Power-Fail Output

The Power-Fail Output ($\overline{\text{PFO}}$) goes low when PFI goes below 1.25V. It typically sinks 3.2mA with a saturation voltage of 0.1V. With PFI above 1.25V, $\overline{\text{PFO}}$ is actively pulled to V_{OUT}.

Battery-Backup Mode

Two conditions are required to switch to battery-backup mode: 1) V_{CC} must be below the reset threshold, and 2) V_{CC} must be below VBATT. Table 2 lists the status of the inputs and outputs in battery-backup mode.

Battery On Output

The Battery On (BATT ON) output indicates the status of the internal V_{CC}/battery-switchover comparator, which controls the internal V_{CC} and VBATT switches. For V_{CC} greater that VBATT (ignoring the small hysteresis effect), BATT ON typically sinks 3.2mA at 0.1V saturation voltage. In battery-backup mode, this terminal sources approximately 10µA from V_{OUT}. Use BATT ON to indicate battery-switchover status or to supply base drive to an external pass transistor for higher-current applications (see *Typical Operating Circuit*).

Input Supply Voltage

The Input Supply Voltage (V_{CC}) should be a regulated 5V. V_{CC} connects to V_{OUT} via a parallel diode and a large PMOS switch. The switch carries the entire current load for currents less than 250mA. The parallel diode carries any current in excess of 250mA. Both the switch and the diode have impedances less than 1Ω each. The maximum continuous current is 250mA, but power-on transients may reach a maximum of 1A.

DALLAS SEMICONDUCTOR

DS1202, DS1202S
Serial Timekeeping Chip

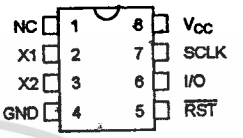
FEATURES

- Real time clock counts seconds, minutes, hours, date of the month, month, day of the week, and year with leap year compensation valid up to 2100
- 24 x 8 RAM for scratchpad data storage
- Serial I/O for minimum pin count
- 2.0–5.5 volt full operation
- Uses less than 300 nA at 2 volts
- Single–byte or multiple–byte (burst mode) data transfer for read or write of clock or RAM data
- 8–pin DIP or optional 16–pin SOIC for surface mount
- Simple 3–wire interface
- TTL–compatible (V_{CC} = 5V)
- Optional industrial temperature range –40°C to +85°C (IND)

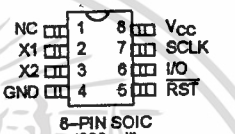
ORDERING INFORMATION

DS1202	8–pin DIP
DS1202S	16–pin SOIC
DS1202S–8	8–pin SOIC
DS1202N	8–pin DIP (IND)
DS1202SN	16–pin SOIC (IND)
DS1202SN–8	8–pin SOIC (IND)

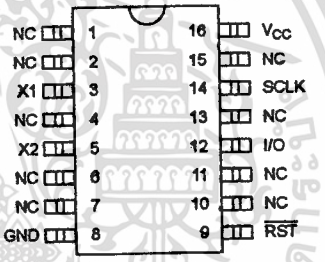
PIN ASSIGNMENT



8–PIN DIP



8–PIN SOIC (208 mil)



16–PIN SOIC

PIN DESCRIPTION

NC	– No Connection
X1, X2	– 32.768 KHz Crystal Input
GND	– Ground
$\overline{RST}$	– Reset
I/O	– Data Input/Output
SCLK	– Serial Clock
V_{CC}	– Power Supply Pin

DESCRIPTION

The DS1202 Serial Timekeeping Chip contains a real time clock/calendar and 24 bytes of static RAM. It communicates with a microprocessor via a simple serial interface. The real time clock/calendar provides seconds, minutes, hours, day, date, month, and year information. The end of the month date is automatically adjusted for months with less than 31 days, including corrections for leap year. The clock operates in either the 24–hour or 12–hour format with an AM/PM indicator. Interfacing the DS1202 with a microprocessor is simplified by using synchronous serial communication. Only three wires are required to communicate with the clock/RAM: (1) $\overline{RST}$ (Reset), (2) I/O (Data line), and (3) SCLK (Serial clock). Data can be transferred to and from the clock/RAM one byte at a time or in a burst of up to 24 bytes. The DS1202 is designed to operate on very low power and retain data and clock information on less than 1 microwatt.

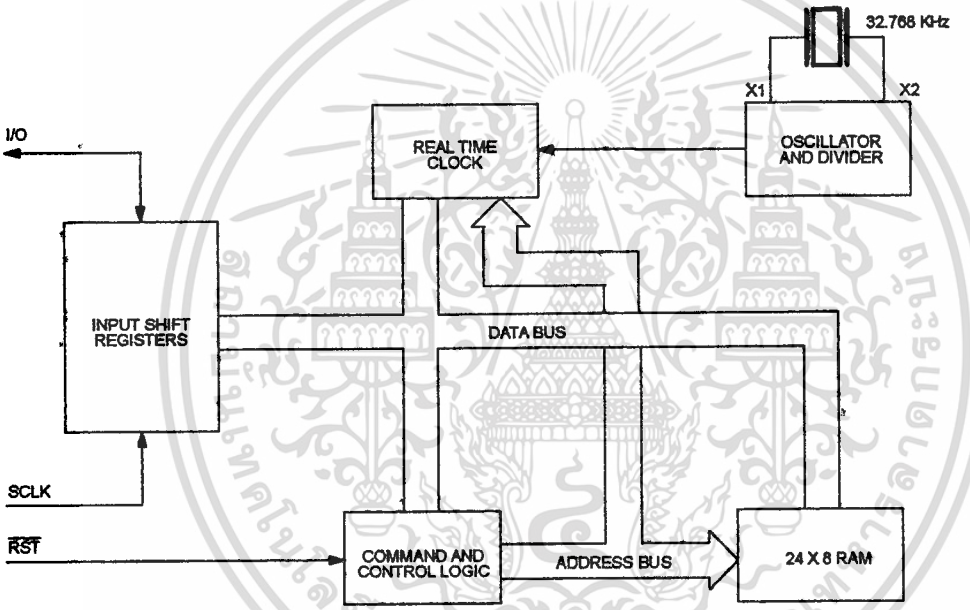
OPERATION

The main elements of the Serial Timekeeper are shown in Figure 1: shift register, control logic, oscillator, real time clock, and RAM. To initiate any transfer of data, $\overline{RST}$ is taken high and eight bits are loaded into the shift register providing both address and command information. Data is serially input on the rising edge of the SCLK. The first eight bits specify which of 32 bytes will be accessed, whether a read or write cycle will take place, and whether a byte or burst mode transfer is to occur. After the first eight clock cycles have occurred which load the command word into the shift register, additional clocks will output data for a read or input data for a write. The number of clock pulses equals eight plus eight for byte mode or eight plus up to 192 for burst mode.

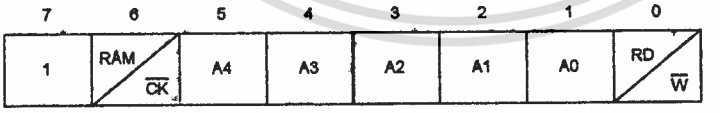
COMMAND BYTE

The command byte is shown in Figure 2. Each data transfer is initiated by a command byte. The MSB (Bit 7) must be a logic 1. If it is zero, further action will be terminated. Bit 6 specifies clock/calendar data if logic 0 or RAM data if logic 1. Bits one through five specify the designated registers to be input or output, and the LSB (Bit 0) specifies a write operation (input) if logic 0 or read operation (output) if logic 1. The command byte is always input starting with the LSB (bit 0).

DS1202 BLOCK DIAGRAM Figure 1



ADDRESS/COMMAND BYTE Figure 2



RESET AND CLOCK CONTROL

All data transfers are initiated by driving the $\overline{RST}$ input high. The $\overline{RST}$ input serves two functions. First, $\overline{RST}$ turns on the control logic which allows access to the shift register for the address/command sequence. Second, the $\overline{RST}$ signal provides a method of terminating either single byte or multiple byte data transfer. A clock cycle is a sequence of a falling edge followed by a rising edge. For data inputs, data must be valid during the rising edge of the clock and data bits are output on t he falling edge of clock. All data transfer terminates if the $\overline{RST}$ input is low and the I/O pin goes to a high impedance state. Data transfer is illustrated in Figure 3.

DATA INPUT

Following the eight SCLK cycles that input a write command byte, a data byte is input on the rising edge of the next eight SCLK cycles. Additional SCLK cycles are ignored should they inadvertently occur. Data is input starting with bit 0. Due to the inherent nature of the logic state machine, writing times containing an absolute value of "59" seconds should be avoided.

DATA OUTPUT

Following the eight SCLK cycles that input a read command byte, a data byte is output on the falling edge of the next eight SCLK cycles. Note that the first data bit to be transmitted occurs on the first falling edge after the last bit of the command byte is written. Additional SCLK cycles retransmit the data bytes should they inadvertently occur so long as $\overline{RST}$ remains high. This operation permits continuous burst mode read capability. Data is output starting with bit 0.

BURST MODE

Burst mode may be specified for either the clock/calendar or the RAM registers by addressing location 31 decimal (address/command bits one through five = logical one). As before, bit six specified clock or RAM and bit 0 specifies read or write. There is no data storage capacity at locations 8 through 31 in the Clock/Calendar Registers or locations 24 through 31 in the RAM registers. When writing to the clock registers in the burst mode, the first eight registers must be written in order for the data to be transferred.

However, when writing to RAM in burst mode it is not necessary to write all 24 bytes for the data to transfer. Each byte that is written to will be transferred to RAM regardless of whether all 24 bytes are written or not.

CLOCK/CALENDAR

The clock/calendar is contained in eight write/read registers as shown in Figure 4. Data contained in the clock/calendar registers is in binary coded decimal format (BCD).

CLOCK HALT FLAG

Bit 7 of the seconds register is defined as the clock halt flag. When this bit is set to logic 1, the clock oscillator is stopped and the DS1202 is placed into a low–power standby mode with a current drain of not more than 100 nanoamps. When this bit is written to logic 0, the clock will start.

AM–PM/12–24 MODE

Bit 7 of the hours register is defined as the 12– or 24–hour mode select bit. When high, the 12–hour mode is selected. In the 12–hour mode, bit 5 is the AM/PM bit with logic high being PM. In the 24–hour mode, bit 5 is the second 10 hour bit (20–23 hours).

WRITE PROTECT BIT

Bit 7 of the control register is the write protect bit. The first seven bits (bits 0–6) are forced to zero and will always read a zero when read. Before any write operation to the clock or RAM, bit 7 must be zero. When high, the write protect bit prevents a write operation to any other register.

CLOCK/CALENDAR BURST MODE

The clock/calendar command byte specifies burst mode operation. In this mode the eight clock/calendar registers can be consecutively read or written (see Figure 4) starting with bit 0 of address 0.

RAM

The static RAM is 24 x 8 bytes addressed consecutively in the RAM address space.

RAM BURST MODE

The RAM command byte specifies burst mode operation. In this mode, the 24 RAM registers can be consecutively read or written (see Figure 4) starting with bit 0 of address 0.

REGISTER SUMMARY

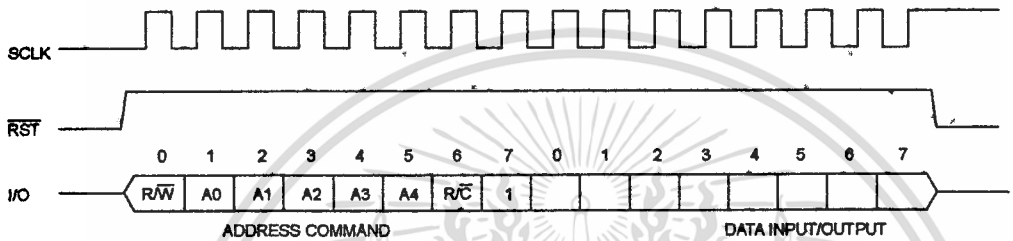
A register data format summary is shown in Figure 4.

CRYSTAL SELECTION

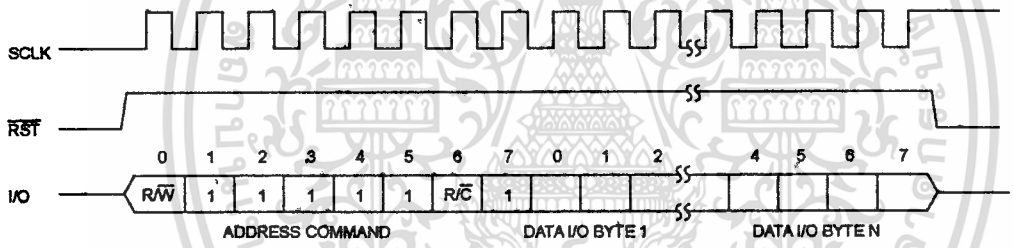
A 32.768 KHz crystal, can be directly connected to the DS1202 via pins 2 and 3 (X1, X2). The crystal selected for use should have a specified load capacitance (CL) of 6 pF. The crystal is connected directly to the X1 and X2 pins. There is no need for external capacitors or resistors. Note: X1 and X2 are very high impedance nodes. It is recommended that they and the crystal be guard-ringed with ground and that high frequency signals be kept away from the crystal area. For more information on crystal selection and crystal layout considerations, please consult Application Note 58, "Crystal Considerations with Dallas Real Time Clocks".

DATA TRANSFER SUMMARY Figure 3

SINGLE BYTE TRANSFER



BURST MODE TRANSFER



FUNCTION	BYTE N	SCLK n
CLOCK	8	72
RAM	24	200

REGISTER ADDRESS/DEFINITION Figure 4

REGISTER ADDRESS
A. CLOCK

	7	6	5	4	3	2	1	0
SEC	1	0	0	0	0	0	0	RD/$\overline{W}$
MIN	1	0	0	0	0	0	1	RD/$\overline{W}$
HR	1	0	0	0	0	1	0	RD/$\overline{W}$
DATE	1	0	0	0	0	1	1	RD/$\overline{W}$
MONTH	1	0	0	0	1	0	0	RD/$\overline{W}$
DAY	1	0	0	0	1	0	1	RD/$\overline{W}$
YEAR	1	0	0	0	1	1	0	RD/$\overline{W}$
CONTROL	1	0	0	0	1	1	1	RD/$\overline{W}$
CLOCK BURST	1	0	1	1	1	1	1	RD/$\overline{W}$

REGISTER DEFINITION

00–59	CH	10 SEC		SEC		
00–59	0	10 MIN		MIN		
01–12 00–23	12/24	0	10 A/P	HR	HR	
01–28/29 01–30 01–31	0	0	10 DATE		DATE	
01–12	0	0	0	10 M	MONTH	
01–07	0	0	0	0	0	DAY
0–99	10 YEAR				YEAR	
	WP	FORCED TO ZERO				

B. RAM

RAM 0	1	1	0	0	0	0	0	RD/$\overline{W}$
RAM 23	1	1	1	0	1	1	1	RD/$\overline{W}$
RAM BURST	1	1	1	1	1	1	1	RD/$\overline{W}$

RAM DATA 0

RAM DATA 23

ABSOLUTE MAXIMUM RATINGS*

Voltage on Any Pin Relative to Ground	–0.3V to +7.0V
Operating Temperature	0°C to 70°C
Storage Temperature	–55°C to +125°C
Soldering Temperature	260°C for 10 seconds

* This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operation sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods of time may affect reliability.

RECOMMENDED DC OPERATING CONDITIONS

(0°C to 70°C)

PARAMETER	SYMBOL		MIN	TYP	MAX	UNITS	NOTES
Supply Voltage	V_{CC}		2.0		5.5	V	1
Logic 1 Input	V_{IH}		2.0		V_{CC}+0.3	V	1
Logic 0 Input	V_{IL}	V_{CC}=2.0V	–0.3		+0.3	V	1
		V_{CC}=5V	–0.3		+0.8		

DC ELECTRICAL CHARACTERISTICS

(0°C to 70°C; V_{CC} = 2.0 to 5.5V*)

PARAMETER	SYMBOL		MIN	TYP	MAX	UNITS	NOTES
Input Leakage	I_{LI}				±500	µA	6
I/O Leakage	I_{LO}				±500	µA	6
Logic 1 Output	V_{OH}	V_{CC}=2V	1.6			V	2
		V_{CC}=5V	2.4				
Logic 0 Output	V_{OL}	V_{CC}=2V			0.4	V	3
		V_{CC}=5V			0.4		
Active Supply Current	I_{CC}	V_{CC}=2V			0.4	mA	5
		V_{CC}=5V			1.2		
Timekeeping Current	I_{CC1}	V_{CC}=2V			0.3	µA	4
		V_{CC}=5V			1		
Leakage Current	I_{CC2}	V_{CC}=2V			100	nA	10
		V_{CC}=5V			100		

*Unless otherwise noted.

CAPACITANCE

(t_A = 25°C)

PARAMETER	SYMBOL	CONDITION	TYP	MAX	UNITS	NOTES
Input Capacitance	C_I		5		pF	
I/O Capacitance	$C_{I/O}$		10		pF	
Crystal Capacitance	C_X		6		pF	

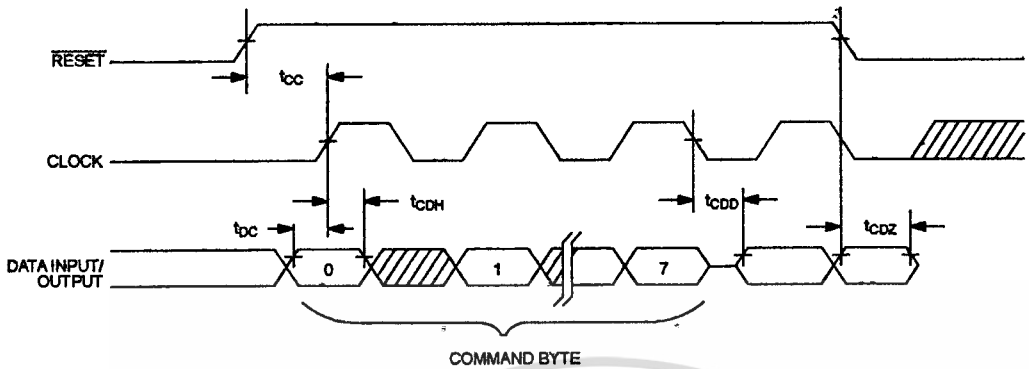
AC ELECTRICAL CHARACTERISTICS

(0°C to 70°C; V_{CC} = 2.0 to 5.5V*)

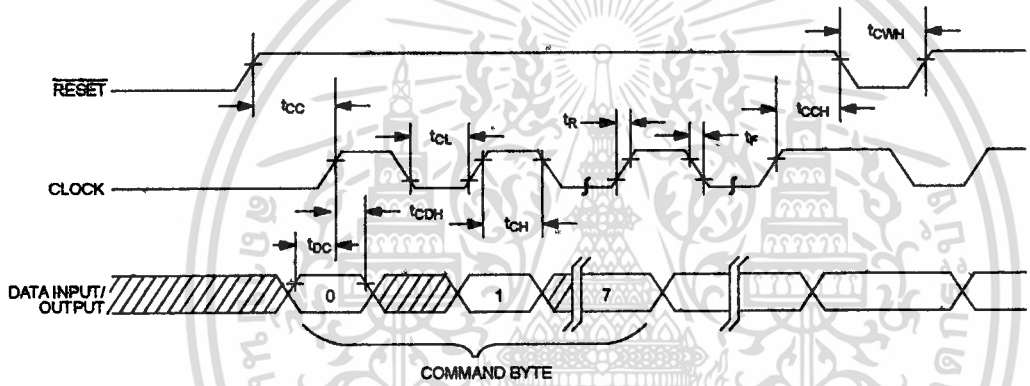
PARAMETER	SYMBOL		MIN	TYP	MAX	UNITS	NOTES
Data to CLK Setup	t_{DC}	V_{CC}=2V	200			ns	7
		V_{CC}=5V	50				
CLK to Data Hold	t_{CDH}	V_{CC}=2V	280			ns	7
		V_{CC}=5V	70				
CLK to Data Delay	t_{CDD}	V_{CC}=2V			800	ns	7, 8, 9
		V_{CC}=5V			200		
CLK Low Time	t_{CL}	V_{CC}=2V	1000			ns	7
		V_{CC}=5V	250				
CLK High Time	t_{CH}	V_{CC}=2V	1000			ns	7, 12
		V_{CC}=5V	250				
CLK Frequency	f_{CLK}	V_{CC}=2V			0.5	MHz	7, 12
		V_{CC}=5V	DC		2.0		
CLK Rise and Fall	t_R, t_F	V_{CC}=2V			2000	ns	
		V_{CC}=5V			500		
$\overline{RST}$ to CLK Setup	t_{CC}	V_{CC}=2V	4			µs	7
		V_{CC}=5V	1				
CLK to $\overline{RST}$ Hold	t_{CCH}	V_{CC}=2V	1000			ns	7
		V_{CC}=5V	250				
$\overline{RST}$ Inactive Time	t_{CWH}	V_{CC}=2V	4			µs	7
		V_{CC}=5V	1				
$\overline{RST}$ to I/O High Z	t_{CDZ}	V_{CC}=2V			280	ns	7
		V_{CC}=5V			70		

*Unless otherwise noted.

TIMING DIAGRAM: READ DATA TRANSFER Figure 5



TIMING DIAGRAM: WRITE DATA TRANSFER Figure 6



NOTES:

1. All voltages are referenced to ground.
2. Logic one voltages are specified at a source current of 1 mA at V_{CC}=5V and 0.4 mA at V_{CC}=2V, V_{OH}=V_{CC} for capacitive loads.
3. Logic zero voltages are specified at a sink current of 4 mA at V_{CC}=5V and 1.5 mA at V_{CC}=2V.
4. I_{CC1} is specified with I/O open, $\overline{RST}$ set to a logic 0, and clock halt flag=0 (oscillator enabled).
5. I_{CC} is specified with the I/O pin open, $\overline{RST}$ high, SCLK=2 MHz at V_{CC}=5V; SCLK=500 KHz, V_{CC}=2V and clock halt flag=0 (oscillator enabled).
6. $\overline{RST}$, SCLK, and I/O all have 40KΩ pull–down resistors to ground.
7. Measured at V_{IH}=2.0V or V_{IL}=0.8V and 10 ms maximum rise and fall time.
8. Measured at V_{OH}=2.4V or V_{OL}=0.4V.
9. Load capacitance = 50 pF.

กิตติกรรมประกาศ

ปริญญานิพนธ์ชิ้นนี้สามารถที่จะสำเร็จลงได้ด้วยความช่วยเหลือของผู้อุปการะคุณหลายฝ่าย โดยเฉพาะอย่างยิ่ง พระคุณของบิดา มารดาผู้ซึ่งให้โอกาส และทุกสิ่งทุกอย่างแก่พวกเราคณะผู้จัดทำ ตลอดจนพระคุณของท่านอาจารย์ที่ปรึกษา ท่านอาจารย์ สมชศ จุฉเฉะปิยะ ที่คอยให้คำปรึกษา คำแนะนำ ต่าง ๆ รวมทั้งเครื่องใช้ที่ใช้ในการดำเนินงาน และสุดท้ายก็คงจะเป็นกลุ่มของเพื่อน ๆ ที่คอยให้กำลังใจ และแนวความคิดบางสิ่งบางอย่างแก่เรา ทางคณะผู้จัดทำจึงขอขอบพระคุณเป็นอย่างสูงไว้ ณ โอกาสนี้ด้วย

คณะผู้จัดทำ

บรรณานุกรม

1. รศ. สมชศ จุณณะปิยะ "การประยุกต์ใช้งานไมโครคอนโทรลเลอร์ MCS-51" กรุงเทพมหานคร:สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
2. ดร.ปราโมทย์ วาดเขียน, ดร.วิวัฒน์ กิรานนท์ "พื้นฐานการสื่อสารข้อมูล" กรุงเทพมหานคร:สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
3. รันวา ศรีประโมง "การเขียนโปรแกรมภาษาซีสำหรับวิศวกรรม" กรุงเทพมหานคร:มหาวิทยาลัยเทคโนโลยีมหานคร, 2539
4. ไพโรจน์ ไววานิชกิจ, กมล เขมะรังษี "เปิดโลกการสื่อสารไร้สาย" กรุงเทพมหานคร:ซีเอ็ดยูเคชั่น, 2539
5. พรศักดิ์ ทับเที่ยง "การส่งข้อมูลดิจิตอลด้วยวิธี เอฟ เอส เค ที่มีอัตราการส่งสูง" วิทยานิพนธ์ตามหลักสูตรวิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมไฟฟ้า สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, 2537