ระบบ ไอพี เทเลโฟน บนเครือข่ายลาดกระบัง

IP TELEPHONE ON LADKRABANG CAMPUS NETWORK

โดย

นาย วิฑูรย์ ศิริชัยสุทธิกร

นาย สมพงศ์ คล้ายบัณฑิต

นาย โอฬาริก สงวนพรรค

เลขเรียกหนังสือ ปพ วิ๕๑๔ ร ๒๕๔๑

เลขทะเบียน ๐๐๐๔๑๗

วัน เดือน ปี ๑๗ ก.ค. ๔๓

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2541

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

000417

ระบบ ไอพี เทเลโฟน บนเครือข่ายลาดกระบัง

IP TELEPHONE ON LADKRABANG CAMPUS NETWORK

โดย

นาย วิฑูรย์ ศิริชัยสุทธิกร 39013023

นาย สมพงศ์ คล้ายบัณฑิต 39013031

นาย โอฬาริก สงวนพรรค 39013039

อาจารย์ที่ปรึกษา

รศ.ดร. สุวิพล สิทธิชีวภาค

ผศ. เกรียงไกร วงศ์โรจนภรณ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2541

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ระบบไอพี เทเลโฟน บนเครือข่ายลาดกระบัง

IP TELEPHONE ON LADKRABANG CAMPUS NETWORK

โดย นาย วิฑูรย์ ศิริชัยสุทธิกร 39013023
นาย สมพงศ์ คล้ายบัณฑิต 39013031
นาย โอฬาริก สงวนพรรค 39013039

อาจารย์ที่ปรึกษา รศ.ดร. สุวิพล สิทธิชีวภาค
ผศ. เกรียงไกร วงศ์โรจนภรณ์

บทคัดย่อ

โครงงานนี้เป็นการจำลองระบบไอพีเทเลโฟน(IP TELEPHONE) มาไว้บนเครือข่ายของสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง โดยมี ภาควิชาวิศวกรรมโทรคมนาคม เป็นชุมสายที่ใช้คอมพิวเตอร์ที่มีวินโดว์ 95 เป็นส่วนของ IP gateway เชื่อมต่อระหว่างเครือข่ายขององค์การโทรศัพท์ (PSTN) กับเครือข่ายคอมพิวเตอร์ของลาดกระบัง โดยการนำสัญญาณเสียงมาทำการแซมปลิ้ง แล้วแปลงข้อมูลให้อยู่ในรูปของ แพ็กเกจเสียง แล้วส่งเข้าไปในระบบเน็ตเวิร์คคอมพิวเตอร์ โดยใช้ยูดีพีโพรโตคอล (UDP PROTOCOL)

Abstract

This project is to simulate the IP Telephone System on the campus institute network of King Mongkut's Institute of Technology Ladkrabang at Department of Telecommunication Engineering's IP gateway connected between Public Switch Telephone Network and Computer Network of Ladkrabang by Windows95. The sound signal is used to sampling process and form it to packet then transmits through network system used the UDP Protocol.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ปริญญานิพนธ์ปีการศึกษา 2541

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบ ไอพี เทเลโฟน บนเครือข่ายลาดกระบัง

IP TELEPHONE ON LADKRABANG CAMPUS NETWORK

ผู้จัดทำ

1.	นายวิฑูรย์ ศิริชัยสุทธิกร	39013023
2.	นายสมพงศ์ คล้ายบัณฑิต	39013031
3.	นายโอพาริก สงวนพรรค	39013039

..... อาจารย์ที่ปรึกษา

(รศ.ดร. สุวิพล สิทธิชีวภาค)

..... อาจารย์ที่ปรึกษา

(ผศ. เกรียงไกร วงศ์โรจนภรณ์)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญตาราง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 1
บทนำ

"การสื่อสารไร้พรมแดน" คำกล่าวนี้แสดงถึงมุมมองของโลกในยุคปัจจุบัน หรือที่เราเรียกกันว่า ยุคโลกาภิวัฒน์ (Grobalization) การสื่อสารข้อมูลเป็นหัวใจหลักของธุรกิจในยุคนี้เพื่อการแลกเปลี่ยนข้อมูล ข่าวสาร ที่รวดเร็ว และฉับไว สาเหตุก็เนื่องจากธุรกิจในปัจจุบันมีการแข่งขันกันมาก ผู้ที่มีความชำนาญ และผู้ที่มีข้อมูลอยู่จำนวนมากย่อมได้เปรียบ แต่การมีข้อมูลจำนวนมากซึ่งเปรียบเสมือนข้อมูลดิบไม่ได้เป็นสิ่งช่วยส่งเสริมความก้าวหน้าขององค์กร การปรับตัวขององค์กรเพื่อนำข้อมูลที่มีอยู่มาใช้ให้ได้มากที่สุดจึงเป็นสิ่งสำคัญอย่างยิ่ง และการแลกเปลี่ยนข้อมูลในหน่วยงานต่างๆ ขององค์กรจะช่วยเพิ่มศักยภาพของธุรกิจได้เป็นอย่างดี ดังนั้นเพื่อตอบสนองความต้องการทางธุรกิจ เทคโนโลยีทางด้านการสื่อสารจึงได้มีการพัฒนาในรูปแบบต่างๆ เช่น ระบบการสื่อสารผ่านดาวเทียม ระบบการสื่อสารผ่านใยแก้วนำแสง (Optical Fiber) เป็นต้น และได้มีการผนวกการสื่อสารเข้ากับระบบคอมพิวเตอร์ ที่เรียกว่า " ไอที " (Information Technology) ซึ่งเป็นเทคโนโลยีที่เข้ามามีบทบาทอย่างมากในสังคมปัจจุบัน เพราะในขณะนี้คอมพิวเตอร์ถูกนำมาใช้ในหลายสาขาอาชีพ เช่น งานทางด้านวิศวกรรม ทางด้านสถาปัตยกรรม ทางด้านอุตสาหกรรม เป็นต้น

ปัจจุบันคอมพิวเตอร์ มีการพัฒนาอย่างรวดเร็ว มีประสิทธิภาพสูงในการประมวลผล อีกทั้งระบบการสื่อสารก็ได้ถูกพัฒนาไปอย่างมาก สามารถส่งข้อมูลทั้งภาพ และเสียงได้ หรือที่เรารู้จักกันว่า "มัลติมีเดีย" (Multimedia) ทำให้มีการพัฒนาโปรแกรมประยุกต์ (Software) ขึ้นมารองรับการใช้งานกันอย่างแพร่หลาย จากโปรแกรมแบบปิดสำหรับผู้ใช้คนเดียว (Single User) มาเป็นโปรแกรมแบบเปิดสำหรับผู้ใช้แบบกลุ่มงาน (Working Group) ที่ยอมให้โปรแกรมอื่น หรือผู้ใช้คนอื่นเข้ามาแลกเปลี่ยนข้อมูลได้ การแลกเปลี่ยนข้อมูลภายในระบบคอมพิวเตอร์เครื่องเดียวกันทำได้หลายวิธีเช่น OLE (Object Linking and Embedding) , File Mapping , DDE (Dynamic Data Exchange)เป็นต้น สำหรับการแลกเปลี่ยนข้อมูลระหว่างเครื่องคอมพิวเตอร์สามารถทำได้เช่นเดียวกัน

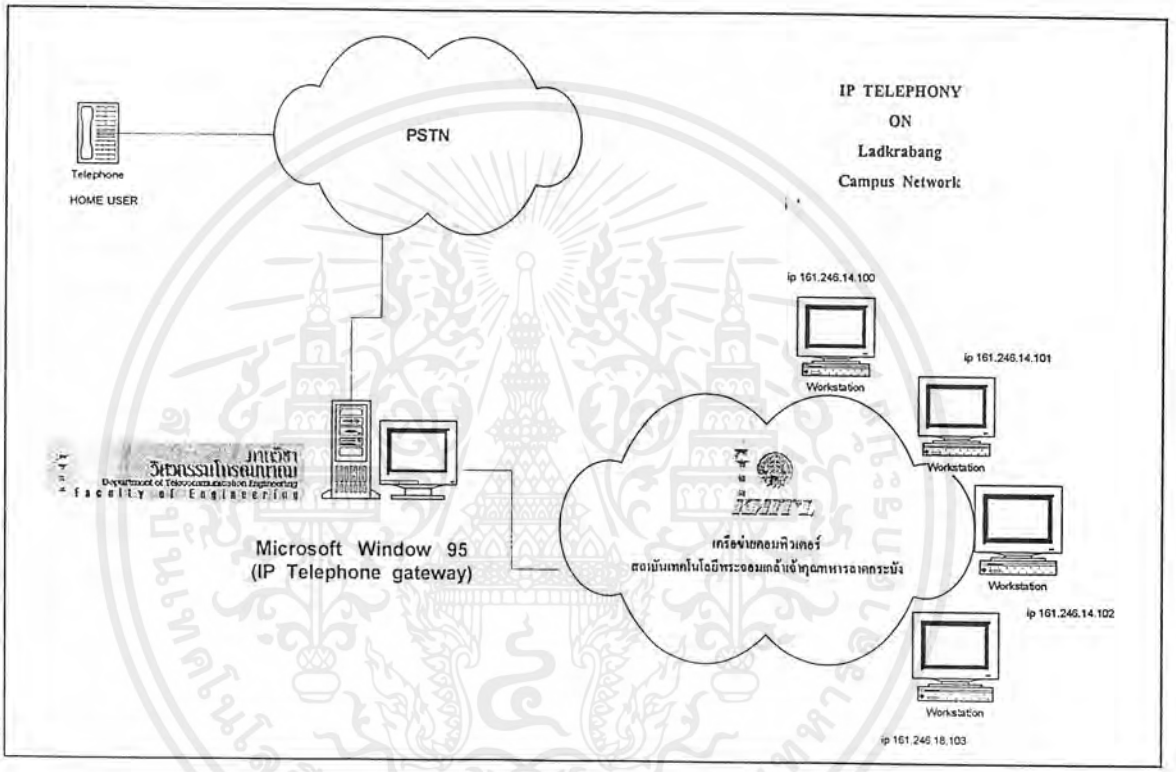
พัฒนาการทางด้านเครือข่ายโทรศัพท์ PSTN ในรอบ 20 ปีที่ผ่านมาไม่มีอะไรก้าวหน้ามากนัก การใช้เทคนิคในเรื่องดิจิตอลและ PCM ได้ใช้กันมานานพอควร เทคนิคของการสวิตชิงก์เป็นเทคนิคที่ได้รับการพัฒนาและใช้กันมานานพอควร

แต่ในช่วงเวลาที่ผ่านมาสิบปี พัฒนาการทางด้านเครือข่ายข้อมูลได้รับการพัฒนาให้ก้าวหน้าไปไกลมาก โดยเฉพาะเทคนิค ACELP ที่ใช้มาตรฐาน G.729 สามารถบีบอัดข้อมูลจนทำให้การส่งข้อมูลไปในสายสัญญาณของเครือข่ายข้อมูลได้มากเป็น 16 เท่าของการรับส่งแบบเดิมที่ใช้ในเทคโนโลยี PSTN ด้วยเหตุนี้เองการพัฒนาการบนพื้นฐานของเครือข่ายข้อมูลที่เป็นแพ็กเกตจึงเป็นที่กล่าวถึงและกำลังอยู่ในระหว่างการแข่งขันกันพัฒนาให้ก้าวหน้ายิ่งขึ้น

สำหรับโครงงานนี้ ระบบไอพี เทเลโฟนนี (IP TELEPHONY) เป็นเทคโนโลยีที่ผนวกเครือข่ายคอมพิวเตอร์เข้ากับเครือข่ายโทรศัพท์พื้นฐาน (PSTN) ทำให้สามารถติดต่อสื่อสารข้อมูลกันในลักษณะของสัญญาณเสียงระหว่างเครื่องคอมพิวเตอร์กับเครื่องคอมพิวเตอร์ และระหว่างเครื่อง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

คอมพิวเตอร์กับเครื่องโทรศัพท์ โดยนำเอาสัญญาณเสียงที่เป็นสัญญาณอนาล็อก (Analog Signal) มาแปลงเป็นสัญญาณดิจิตอล (Digital Signal) โดยใช้วิธี ADPCM (Adaptive Differential Pulse Code Modulation) แล้วจัดรูปแบบของข้อมูลให้อยู่ในรูปแบบของแพ็กเก็ตเสียง โดยใช้โปรโตคอล TCP/IP (Transmission Control Protocal/Internet Protocal) เป็นตัวส่งแพ็กเก็ตเสียงเข้าไปในเครือข่ายคอมพิวเตอร์จากหลักการดังกล่าวข้างต้น เราจะนำมาประยุกต์ใช้กับระบบเครือข่ายคอมพิวเตอร์ของ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง (Ladkrabang Campus Network) โดยมีภาควิชาโทรคมนาคมเป็นที่ตั้งของเซิร์ฟเวอร์ (Telephony Server) เชื่อมต่อกับเครือข่ายขององค์การโทรศัพท์แห่งประเทศไทย



รูป 1.1 แสดงการจำลองระบบไอพีเทเลโฟนบนลาดกระบังเน็ตเวิร์ค

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 2
ทฤษฎีและหลักการ

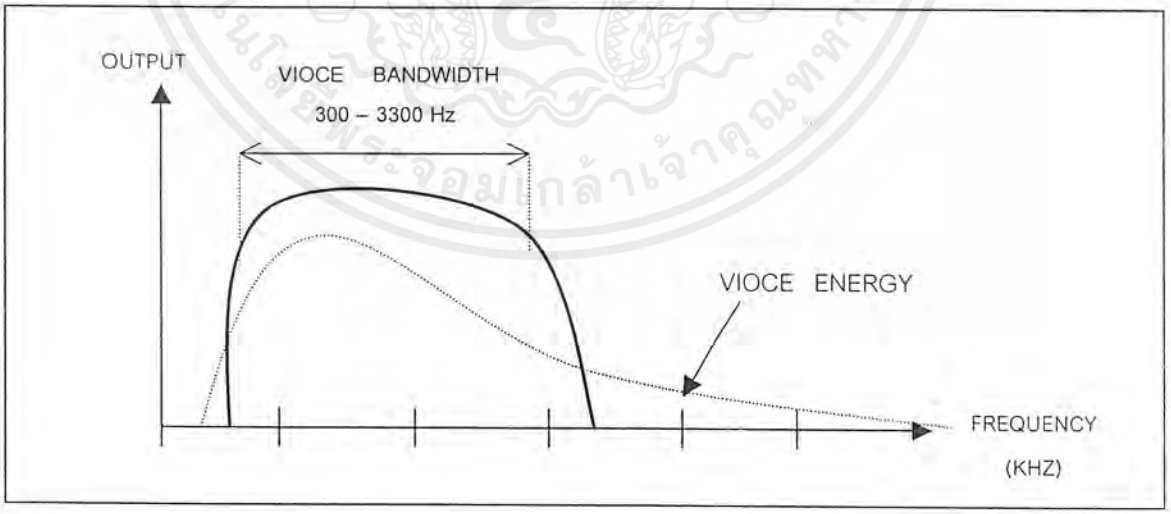
ในระบบ IP TELEPHONY ตัวหลักการในมุมมองทางด้านการสื่อสาร จะใช้ทฤษฎีทางด้านระบบโครงข่ายโทรศัพท์ พื้นฐาน (PSTN) ผนวก เข้ากับตัวระบบเครือข่ายของคอมพิวเตอร์ (COMPUTER NETWORK) ซึ่งจะเชื่อมต่อกันโดยผ่าน IP GATEWAY ที่เราใช้ทำการ์ดอินเตอร์เฟสขึ้นมา ทำงานร่วมกับคอมพิวเตอร์ที่มี WINDOWS 95 เป็น NOS (NETWORK OPERATING SYSTEM) ที่ต่ออยู่กับเครือข่ายในภาควิชาโทรคมนาคมโดยรายละเอียดทางทฤษฎีของส่วนต่าง ๆ ที่เป็นส่วนประกอบหลักในระบบ IP TELEPHONY

2.1 ระบบชุมสายโทรศัพท์

ในการสื่อสารในระบบโทรศัพท์จะเป็นการเปลี่ยนจากสัญญาณเสียงเป็นสัญญาณไฟฟ้า โดยผ่านตัวกลางที่เป็นสายนำสัญญาณในการติดต่อซึ่งในปัจจุบันตัวกลางในการติดต่อสามารถใช้ได้หลายทาง เช่นผ่านระบบไมโครเวฟ ผ่านระบบดาวเทียม หรือผ่านระบบใยแก้วนำแสง เป็นต้น โดย แบนวิทด์ของสัญญาณเสียงที่ใช้ในระบบโทรศัพท์อยู่ช่วงระหว่าง 300-4,000 Hz โดยมีสัญญาณต่าง ๆ ที่คอยควบคุมการทำงาน และแสดงสถานะระหว่างชุมสายกับโทรศัพท์ของผู้ใช้ (Subscriber) ซึ่งมีรายละเอียดดังนี้

2.1.1 สัญญาณที่ส่งในคู่สายโทรศัพท์

สัญญาณเสียงในการสนทนาสัญญาณความถี่เสียงมีความถี่ในช่วง 20-20,000Hz แต่พลังงานส่วนใหญ่จะอยู่ที่ 100-4,000 Hz ซึ่งเป็นความถี่ของช่วงพลังงานของเสียงพูดของมนุษย์ คือ ประมาณ 300-4,000 Hz โดยมีลักษณะการกระจายพลังงานเสียง (Frequency respone) (รูปที่ 2.1)



รูปที่ 2.1 แสดงการกระจายพลังงานของสัญญาณเสียง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.1.2 สัญญาณควบคุม

เป็นสัญญาณลูกข่าย (Subscriber Signal) ที่เครื่องชุมสายโทรศัพท์จะแจ้งสถานะโดยส่งสัญญาณไปที่ลูกข่าย (Subscriber) เพื่อให้ผู้ใช้ทราบสถานะของการติดต่อและขั้นตอนในการทำงานต่อไป ซึ่งจะประกอบด้วย

1. สัญญาณ Dial Tone เป็นสัญญาณเพื่อแสดงให้ลูกข่ายรู้ว่าทางชุมสายพร้อมให้บริการ โดยให้ผู้ใช้สามารถกดเลขหมายที่ต้องการติดต่อ โดยสัญญาณ Dial Tone จะเป็นสัญญาณไซน์ (Sine wave) 400 Hz ที่มอดูเลทกับความถี่ 50 Hz แบบแอมปลิจูดมอดูเลชั่น (AM)

2. สัญญาณไม่ว่าง (Busy Tone) เป็นสัญญาณแสดงสถานะเพื่อบอกว่าปลายทาง ไม่ว่าง หรือทางชุมสายไม่สามารถติดต่อได้ โดยสัญญาณจะมีลักษณะเป็นสัญญาณไซน์ (Sine wave) ความถี่ 400 Hz ส่งเป็นช่วง ๆ โดยที่เวลาในการส่ง 0.5 วินามี และหยุด 0.5 วินาที สลับกันไป

3. สัญญาณเรียกกลับ (Ring BackTone) เป็นสัญญาณที่บอกผู้เรียกว่า การติดต่อกับเครื่องปลายทางประสบความสำเร็จ และกำลังรอปลายทางรับสาย สัญญาณที่สงเป็นสัญญาณไซน์ (Sine wave) 400 Hz ส่งเป็นช่วง ๆ เวลาการส่งประมาณ 1 วินาที และหยุดส่งประมาณ 4 วินาที สลับกันไป จนกว่าปลายทางจะรับสาย

4. สัญญาณกระดิ่งเรียก (Ringing Tone) ใช้ส่งให้ปลายทางเพื่อให้เครื่องโทรศัพท์ใช้สร้างกระดิ่งเรียกผู้รับสาย เมื่อการติดต่อสำเร็จสัญญาณที่ส่งเป็นสัญญาณไซน์ (Sine wave) ความถี่เท่ากับ 25 Hz ระดับแรงดันประมาณ 75-100 โวลท์ ช่วงเวลาในการส่งเหมือนกับสัญญาณเรียกกลับ

2.1.3 หลักการเบื้องต้นของระบบโทรศัพท์

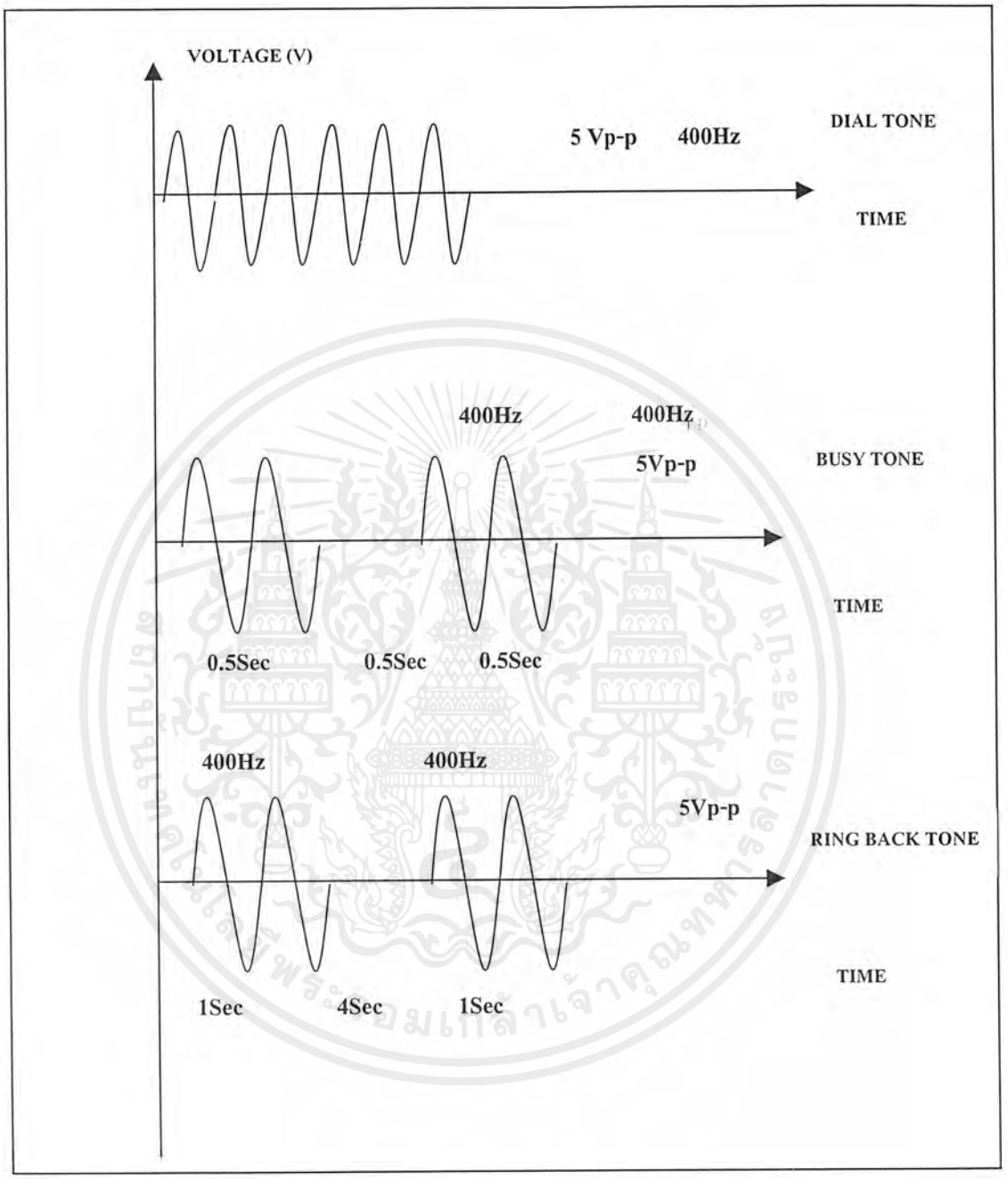
เมื่อเริ่มต้นโทรศัพท์วางหูอยู่เรียกว่าอยู่ในสภาวะวางหู (On Hook) วงจรระหว่างชุมสายหลักและโทรศัพท์จะเป็นวงจรเปิด ยกเว้นวงจรสัญญาณเรียก (Ring Circuit) ซึ่งจะต่ออยู่กับชุมสายหลักตลอดเวลา โดยมีคาปาซิเตอร์ (Capacitor) ทำหน้าที่กันไฟตรงไม่ให้ผ่าน และให้ผ่านได้เฉพาะสัญญาณเรียกซึ่งเป็นไฟฟ้ากระแสสลับ

เมื่อผู้เรียกยกหูโทรศัพท์ขึ้น เรียกว่า อยู่ในสภาวะยกหู (Off Hook) จะทำให้ระดับไฟตรง (DC Voltage) ที่คู่สายโทรศัพท์เปลี่ยนจาก 48 โวลต์ เป็น 10 โวลต์ ชุมสายโทรศัพท์จะรู้ว่าเริ่มต้นการเรียก ก็จะทำการติดต่อระหว่างผู้เรียกกับชุมสาย โดยชุมสายจะส่งสัญญาณหมุน (Dial Tone) ในกรณีที่ชุมสายพร้อมที่จะรับเลขหมาย หรือส่งสัญญาณไม่ว่าง (Busy Tone) ในกรณีที่ชุมสายมีปัญหาหรือเกิดข้อผิดพลาดบางอย่างเพื่อให้ผู้เรียกวางหูแล้วค่อยยกใหม่

เมื่อผู้เรียกได้ยินสัญญาณให้หมุน ก็จะกดหมายเลขโทรศัพท์ของปลายทาง เมื่อชุมสายได้รับสัญญาณการกดเลขหมายตัวแรกก็จะหยุดการส่งสัญญาณให้หมุน และทำการถอดรหัส เมื่อได้หมายเลขครบตามต้องการชุมสายจะทำการตรวจสอบว่าปลายทางว่างหรือไม่ ถ้าว่างก็จะส่งสัญญาณเรียก (Ring Tone) ไปยังปลายทางและจะส่งสัญญาณเรียกกลับ (Ring back Tone) มายังผู้เรียก แต่ถ้าปลายทางไม่ว่างก็จะส่งสัญญาณไม่ว่าง (Busy Tone) กลับมายังผู้เรียก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ถ้าการติดต่อสำเร็จ คือ ปลายทางว่าง และมีผู้รับสาย ชุมสายจะหยุดส่งสัญญาณเรียก และเรียกกลับ พร้อมทั้งทำการสร้างเส้นทางในการติดต่อระหว่างผู้เรียกและผู้รับ (รูปที่ 2.2)



รูปที่ 2.2 แสดงสัญญาณพื้นฐานของระบบโทรศัพท์

2.2 ระบบเน็ตเวิร์คคอมพิวเตอร์

การใช้งานเครื่องไมโครคอมพิวเตอร์ในปัจจุบันนี้ มีอยู่ 2 แบบด้วยกัน คือ Stand Alone และระบบ LAN (Local Area Network) ซึ่งในงานในแบบ Stand Alone เพียงเครื่องเดียวเหมาะสำหรับนักเรียนหรือนักธุรกิจเล็กๆไม่ใหญ่มาก แต่ถ้าเป็นบริษัทที่มีธุรกิจในระดับกลางขึ้นไปดูจะไม่เหมาะสม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ซึ่งการต้องการใช้ทรัพยากรร่วมกันปรับปรุงข้อมูลให้ใหม่และถูกต้องเสมอตลอดจนระบบรักษาความปลอดภัยของข้อมูล เน็ตเวิร์คคงจะเป็นทางเลือกที่ดีที่สุด

ระบบเน็ตเวิร์ค จะหมายถึงการนำเครื่องไมโครคอมพิวเตอร์ตั้งแต่ 2 เครื่องขึ้นไปมาเชื่อมต่อกัน เพื่อทำการแชร์ข้อมูลและใช้ทรัพยากรร่วมกันเช่น ไฟล์ข้อมูล และเครื่องพิมพ์ ระบบเน็ตเวิร์คสามารถแบ่งออกเป็น 3 ประเภทด้วยกันคือ

1. LAN (Local Area Network)

ระบบเครือข่ายระดับท้องถิ่น เป็นเน็ตเวิร์คในระยะไม่เกิน 10 กิโลเมตร ไม่ต้องใช้โครงข่ายการสื่อสารขององค์การโทรศัพท์ คือ จะเป็นระบบเน็ตเวิร์คที่อยู่ภายในอาคารเดียวกัน หรือต่างอาคารกันในระยะใกล้ๆ

2. MAN (Metropolitan Area Network)

ระบบเครือข่ายระดับเมือง เป็นเน็ตเวิร์กที่จะต้องใช้การสื่อสารขององค์การโทรศัพท์หรือการสื่อสารแห่งประเทศไทย เพราะเป็นการติดต่อกันในเมือง เช่นมีเครื่องเวิร์กสเตชันอยู่ที่บางนา มีการติดต่อสื่อสารกับเครื่องเวิร์กสเตชันที่ดอนเมือง

3. WAN (Wide Area network)

ระบบเครือข่ายระดับกว้างไกลหรือเรียกได้ว่าเป็น World Wide ของระบบเน็ตเวิร์กโดยจะเป็นการติดต่อสื่อสารกันในระดับประเทศ ข้ามทวีป จะต้องใช้มีเดีย (Media) ในการสื่อสารขององค์กรโทรศัพท์หรือการสื่อสารแห่งประเทศไทย(คู่สายโทรศัพท์ dial-up line / คู่สายเช่า leased line / ISDN (Integrated Service Digital network) สามารถส่งได้ทั้งข้อมูล เสียง และภาพในเวลาเดียวกัน

ประโยชน์ของเน็ตเวิร์ก

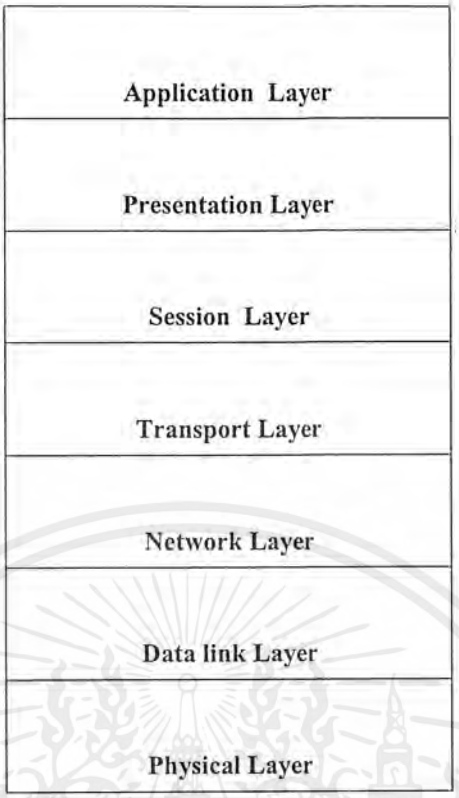
ระบบเน็ตเวิร์คเป็นระบบที่มีการใช้ทรัพยากรและใช้ข้อมูลร่วมกัน สามารถแบ่งประโยชน์ออกเป็น 5 ข้อ คือ

1. การใช้โปรแกรม-ข้อมูลร่วมกัน (Shared Applications)
2. การใช้ฮาร์ดแวร์ร่วมกัน (Shared Hardware)
3. การกระจายการประมวลผล (Distributed processing)
4. การติดต่อสื่อสารแบบรวดเร็ว (Rapid Communication)
5. การติดต่อสื่อสารระหว่างเวิร์กสเตชัน (Inter connected Station)

2.2.1 OSI โมเดล

องค์กรมาตรฐานสากล ISO (International Organization for Standardization) ได้กำหนดมาตรฐานของเครือข่าย โดยจัดแบ่งกิจกรรมของเครือข่าย ออกเป็นงานย่อย ๆ และกำหนดโมเดลแบ่งเป็นขั้น ๆ ตามลำดับเรียกว่ามาตรฐาน OSI (Open System Interconnection) โดยที่จะแบ่งกิจกรรมที่ซับซ้อนในเครือข่าย ออกเป็นงานย่อย ๆ ก็จะช่วยในการออกแบบ และการใช้งานเครือข่ายรวมถึงการเชื่อมโยงกันเป็นไปได้ด้วยความสะดวก และมีวิธีการทำงานอยู่ในกรอบเดียวกัน (รูปที่ 2.3)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.3 แสดงการแบ่งการทำงานของเครือข่ายออกเป็น OSI model

ในแต่ละขั้นของ OSI model จะมีการติดต่อสื่อสารกันเป็นขั้น ๆ ตามลำดับลงมาเช่น Application layer ก็จะติดต่อสื่อสารกับ Presentation Layer ตามลำดับไปจนถึงขั้นแรกสุดคือ Physical Layer

Application Layer เป็นชั้นบนสุดของ โมเดลเป็นส่วนที่จะทำให้การติดต่อระหว่างเครือข่ายกับผู้ใช้เป็นไปได้ตามต้องการ ตัวอย่างแอปพลิเคชั่นของเครือข่าย เช่น ระบบ E-Mail การโอนถ่ายข้อมูล (File Transfer) การขอเข้าใช้ระบบคอมพิวเตอร์ในเครือข่ายเป็นต้น

Presentation Layer เป็นชั้นที่จัดการในเรื่อง "การติดต่อแต่ละครั้ง" หรือ session ให้ระบบคอมพิวเตอร์ทั้งสองฝั่ง โดยทำหน้าที่ตั้งแต่เริ่มการติดต่อ ดูแลในการส่งผ่านข้อมูล ในการติดต่อครั้งนั้น ๆ เป็นไปได้โดยไม่มีปัญหาจนถึงเลิกการติดต่อเมื่อเสร็จงาน

Session Layer เป็นชั้นที่จัดการในเรื่อง "การติดต่อแต่ละครั้ง" หรือ session ให้ระบบคอมพิวเตอร์ทั้งสองฝั่ง โดยทำหน้าที่ตั้งแต่เริ่มการติดต่อ ดูแลในการส่งผ่านข้อมูล ในการติดต่อครั้งนั้น ๆ เป็นไปได้โดยไม่มีปัญหาจนถึงเลิกการติดต่อเมื่อเสร็จงาน

Transport Layer ทำหน้าที่ควบคุมปริมาณ และรายละเอียดวิธีการรับส่งข้อมูล ให้เป็นไปตามกำหนดที่ตั้งไว้ และจัดการให้การเชื่อมโยงเครือข่ายเป็นไปด้วยความราบรื่น Transport Layer จะเป็นชั้นสุดท้าย ที่จัดการเรื่องเส้นทางในการส่งข้อมูล และจัดการตรวจสอบความผิดพลาดของข้อมูล ซึ่งส่วนของ TCP (Transmission Control Protocol) ในโพรโตคอล TCP/IP ทำงานที่ระดับนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Network Layer ทำหน้าที่ควบคุมวิธีการส่งผ่านข้อมูลระหว่างเครือข่ายให้ถูกต้อง และเป็นไปตามเส้นทางที่กำหนด โดยจะจัดการส่งผ่าน packet ข้อมูล ผ่านอุปกรณ์ต่าง ๆ ไปยังเครือข่ายย่อยได้อย่างถูกต้องตามที่ต้องการ นอกจากนี้ยังจัดการดูแลเส้นทางในการส่งข้อมูล (Routing table) และกั้นหรือกรอง packet ข้อมูลที่ส่งไปยังเครือข่ายเดียวกันไม่ให้ข้ามไปยังเครือข่ายอื่น ซึ่งจะช่วยลดปริมาณข้อมูลที่จะวิ่งบนเครือข่ายได้ส่วนหนึ่ง โพรโตคอล IP, TCP/IP และ IPX เป็นโพรโตคอลที่ทำงานอยู่ใน layer นี้

Data link Layer ทำหน้าที่เรียกใช้หรือกำหนดช่องทาง ในการส่งข้อมูลที่ถูกต้อง เช่น Ethernet, Token ring หรือ FDDI เป็นต้น รวมถึงการลำดับและอัตราการรับส่งข้อมูลหรือ flow control และสถานที่ ที่จะส่งข้อมูลไป (address) ทั้งนี้ Data link layer จะเป็นชั้นแรกที่จัดการแปลงข้อมูลจาก bit ให้เป็น packet โดยจะมีการเพิ่มข้อมูลเพื่อตรวจสอบความถูกต้อง ในกรณีที่ส่งข้อมูลออกไป หรือในกรณีที่อ่านข้อมูลที่เข้ามาก็จะตรวจสอบผ่าน checksum เพื่อดูว่าข้อมูลที่ได้รับมาถูกต้องครบถ้วน และถ้าได้รับ packet ข้อมูลที่ไม่ถูกต้องก็จะไม่เอาข้อมูลนั้นไปใช้งาน และจะบอกให้ต้นทางส่งข้อมูลเดิมมาใหม่

Physical Layer รับผิดชอบดูแลในรายละเอียดในการส่งข้อมูลในด้าน hardware เช่น การควบคุม Network Interface Card การส่งสัญญาณผ่านสายสัญญาณแบบต่าง ๆ การเชื่อมต่อเข้ากับเครือข่ายแบบต่าง ๆ โดยใช้ Physical layer จะจัดสร้างสัญญาณทางไฟฟ้า, สัญญาณเสียง หรือสัญญาณที่จำเป็นในการสื่อสารโดยตรงเนื่องจาก Network Layer เป็นชั้นที่โพรโตคอล IP, TCP/IP ทำงานอยู่จะกล่าวโดยละเอียดต่อไป

การใช้โปรมแกรมข้อมูลร่วมกัน สามารถจะทำให้ User หลายๆ คนใช้โปรแกรม-ข้อมูลร่วมกันได้ และเป็นการประหยัดเวลาในการติดตั้งโปรแกรม โดยทำการติดตั้งโปรมแกรมใช้งานไว้ที่ ไฟล์เซิร์ฟเวอร์และเก็บข้อมูลไว้ที่ไฟล์เซิร์ฟเวอร์ เพียงตัวเดียว เครื่องเวิร์กสเตชันไม่จำเป็นต้องมีดิสค์ไดร์ฟ หรือฮาร์ดดิสก์ ก็สามารถใช้งานโปรแกรมและข้อมูลจากเครื่องไฟล์เซิร์ฟเวอร์ได้

การใช้ฮาร์ดแวร์ร่วมกัน ระบบเวิร์กเป็นระบบที่สามารถจะใช้อุปกรณ์จากฮาร์ดแวร์ร่วมกันได้ดี เช่น ระบบเน็ตเวิร์คของท่านมีเครื่องเวิร์กสเตชันอยู่ 12 เครื่อง แต่ละเครื่องมีงานที่ต้องพิมพ์บนเครื่องพิมพ์เลเซอร์ ถ้าไม่มีระบบเน็ตเวิร์กจะต้องใช้งบประมาณในการซื้อเครื่องพิมพ์อย่างน้อย 4-6 เครื่อง แต่สำหรับเน็ตเวิร์คแล้ว สามารถใช้เครื่องพิมพ์เพียง 1-2 เครื่องโดยการทำเป็น Print Server คอยรับงานพิมพ์จากเครื่องทั้ง 12 เครื่อง นอกจากนี้ยังใช้งานโมเด็มและฮาร์ดดิสก์ร่วมกันได้ด้วย

การกระจายการประมวลผล เป็นที่ทราบแล้วว่าโปรมแกรมใช้งานต่างๆ เช่น Excel, word, Data Base ฯลฯได้ถูกติดตั้งในฮาร์ดดิสก์ของเครื่องไฟล์เซิร์ฟเวอร์ ส่วนเครื่องเวิร์กสเตชัน จะมีเพียงจอภาพ ซีพียู และคีย์บอร์ด สำหรับทำงานเมื่อ User ของเวิร์กสเตชันเครื่องหนึ่งต้องการ ใช้งานโปรแกรม Word เครื่องไฟล์เซิร์ฟเวอร์จะทำงาน Copy โปรมแกรม Word จากฮาร์ดดิสก์ของไฟล์เซิร์ฟเวอร์มาไว้ยังหน่วยความจำ Ram ของเวิร์กสเตชันเครื่องนั้นต่อจากนี้เป็นหน้าที่ของ ซีพียู ภายใน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เครื่องเวิร์กสเตชันทำการ Ramโปรมแกรม Word ต่อไปเป็นการกระจายการประมวลผลอย่างมีประสิทธิภาพ

การติดต่อสื่อสารแบบรวดเร็ว ด้วยมาตรฐาน IEEE 802.3 สามารถส่งข้อมูลด้วยอัตราความเร็ว 10 เมกะบิตต่อวินาที (โดยใช้หลักการส่งข้อมูลแบบ Base-band) ความเร็วในระดับนี้จะทำให้การปรับปรุง(Update) ข้อมูลเป็นไปอย่างรวดเร็วโดยเฉพาะการนำอิเล็กทรอนิกส์เมล์ (E-Mail) มาใช้บนระบบเน็ตเวิร์ค ปัจจุบันระบบ Ethernet สามารถจะส่งข้อมูลได้ด้วยความเร็ว 100 เมกะบิตต่อนาที

2.2.2 การติดต่อสื่อสารระหว่างเวิร์กสเตชัน

ในขณะที่เครื่องเวิร์กสเตชันแต่ละเครื่องกำลังทำงานอยู่นั้น บังเอิญต้องมีการปิด (Shut Down) เครื่องไฟล์เซิร์ฟเวอร์เพื่อเปลี่ยนฮาร์ดดิสก์หรือ อุปกรณ์อื่นๆ สามารถส่งข้อความไปยังเครื่องเวิร์กสเตชั่นทุกเครื่องเพื่อเตือนให้เซฟข้อมูลภายใน 10 นาที เช่น Shut Down File Server Within 10 minute. Please save data now. ข้อความเหล่านี้จะปรากฏที่หน้าจอเครื่องเวิร์กสเตชั่นทุกเครื่องทันที ถ้าเวลาการเซฟน้อยไปจะขอเพิ่มเป็น 20 นาที เครื่องเวิร์กสเตชันสามารถจะส่งข้อความไปบอกไฟล์เซิร์ฟเวอร์ได้เช่น Please give me 20 minute for save data...OK

Peer To Peer เป็นระบบที่เครื่องเวิร์กสเตชั่นทุกเครื่องบนระบบเน็ตเวิร์กมีฐานะเทียมเท่ากัน คือทุกเครื่องสามารถจะใช้ไฟล์ในเครื่องอื่นได้ และสามารถใช้เครื่องอื่นมาใช้ไฟล์ของตนเองได้เช่นกัน ระบบ Peer To Peer มีลักษณะการทำงานแบบ Distributed system โดยกระจายทรัพยากรต่างๆ ไปสู่เวิร์กสเตชันอื่นๆ แต่จะมีปัญหาเรื่องการรักษาความปลอดภัย เพราะข้อมูลที่เป็นความลับจะถูกส่งออกไปสู่เวิร์กสเตชันอื่นเช่นกัน

โปรมแกรมที่มีความสามารถทาง Peer To Peer และเป็นที่รู้จักกันดีคือ Windows for Workgroup และ Personal NetWare

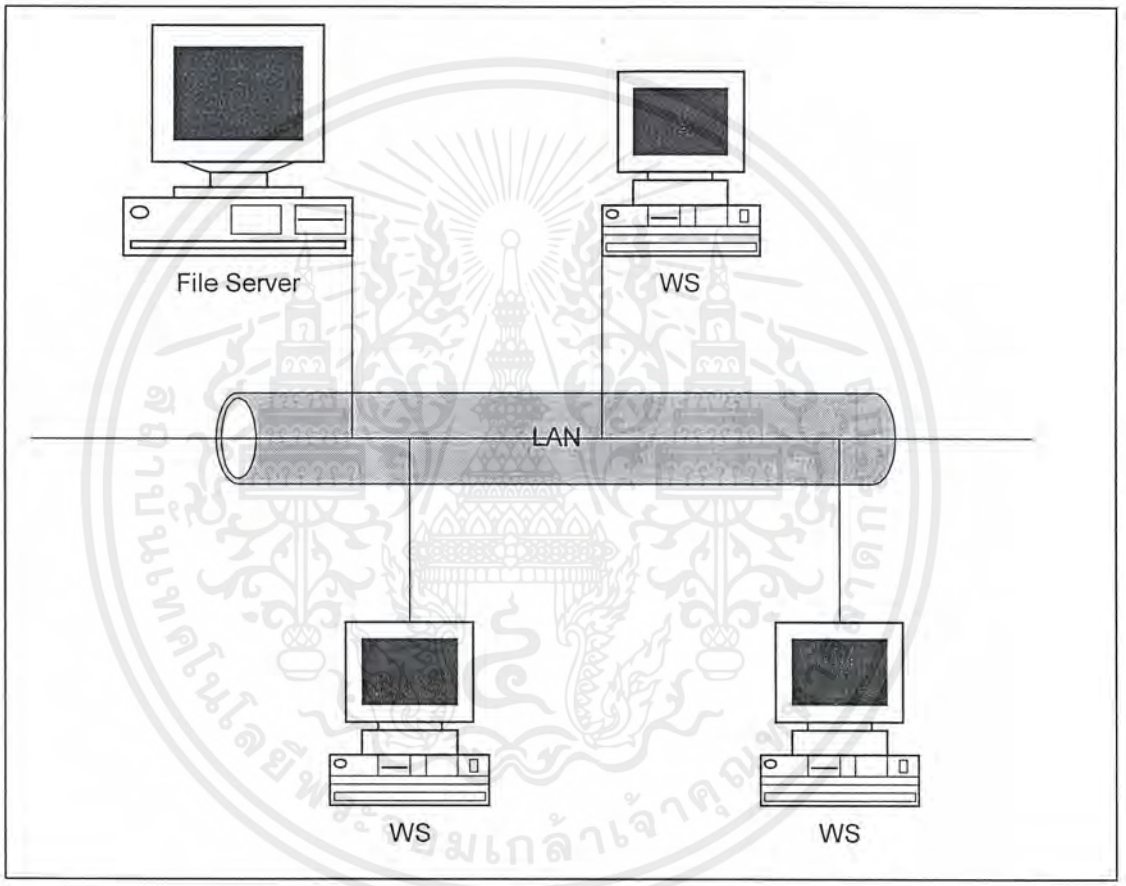
Client/Server เป็นการทำงานแบบ Distributed Processing หรือการประมวลผลแบบกระจาย โดยจะแบ่งกันประมวลผลระหว่างเครื่อง เซิร์ฟเวอร์กับเครื่องเวิร์กสเตชั่นดด้วยและเมื่อใดที่เครื่องเวิร์กสเตชั่นต้องการผลลัพธ์ของข้อมูลบางส่วน จะมีการเรียกใช้ไปยังเครื่องเซิร์ฟเวอร์นำเฉพาะข้อมูลบางส่วนเท่านั้น ส่งกลับมาให้เครื่องเวิร์กสเตชันเพื่อทำการคำนวณข้อมูลนั้นต่อไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.2.3 รูปแบบการเชื่อมต่อของระบบ LAN (Topology)

1) Bus

BUS การเชื่อมต่อแบบบัลจะมีสายหลักอยู่ 1 เส้นเครื่องคอมพิวเตอร์ทั้งเซิร์ฟเวอร์และเวิร์กสเตชั่นทุกเครื่องจะเชื่อมต่อกับสายหลักเส้นนี้เครื่องคอมพิวเตอร์จะถูกมองเป็น Node เมื่อเครื่องเวิร์กสเตชันเครื่องหนึ่ง(Node A) ต้องการจะส่งข้อมูลให้กับเครื่องอื่น (Node C) จะต้องส่งข้อมูลและแอดเดรสของ Node C ลงไปบนบัสสายหลัก เมื่อเครื่อง Node C ได้รับข้อมูลแล้วก็จะนำข้อมูลไปทำงานทันที (รูปที่ 2.4)

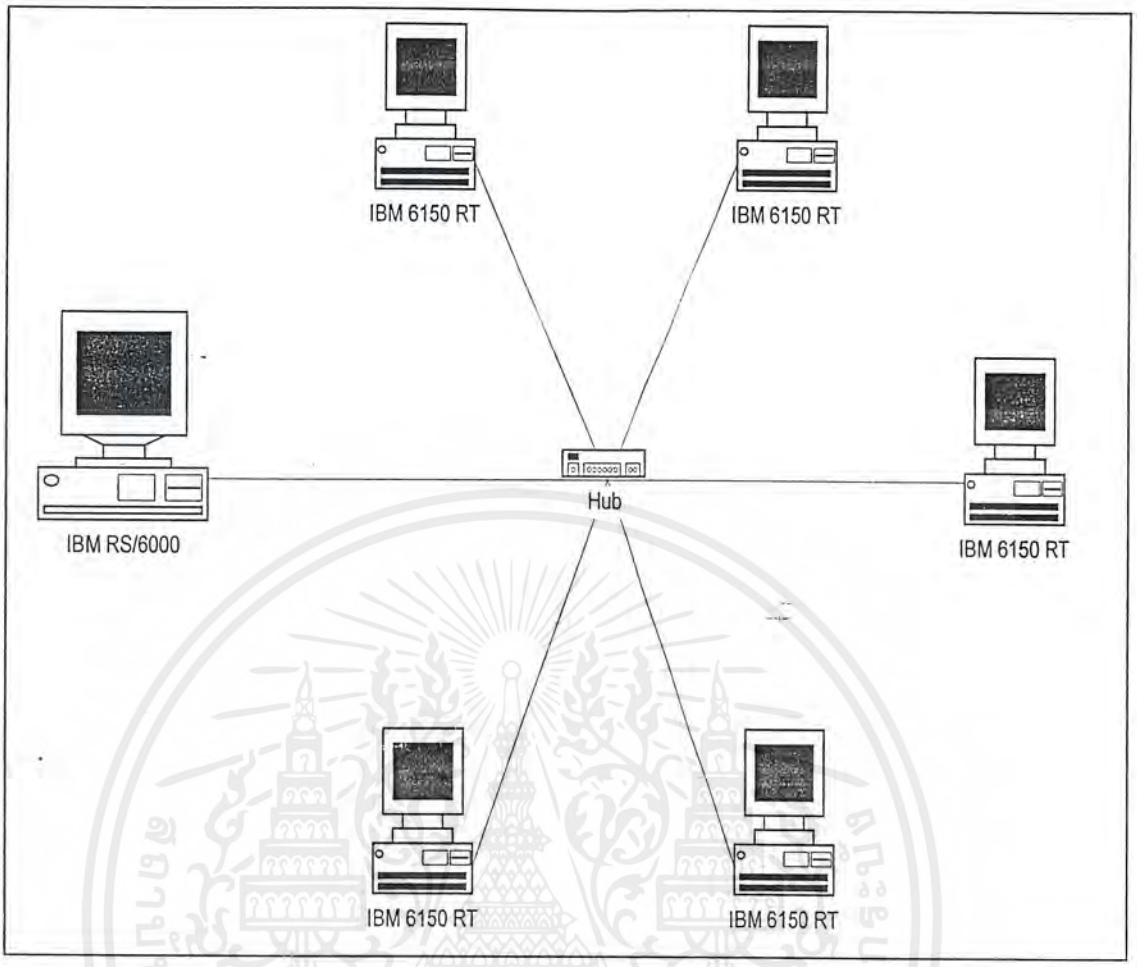


รูป2.4 แสดงการเชื่อมต่อแบบบัส

2) Star

Star การเชื่อมต่อแบบสตาร์นี้จะใช้ Hub เป็นศูนย์กลางในการเชื่อมต่อ คือเครื่องทุกเครื่องจะต่อผ่าน Hub สายเคเบิลที่ใช้ส่วนมากจะเป็น UTP และ Fiber Optic ในการส่งข้อมูล Hub จะเป็นเสมือนตัวทวนสัญญาณ (Repeater) และ Hub บางรุ่นยังสามารถตรวจจับข้อมูล (Data Detection) ต่างๆ เช่น Receive-Sent Date, Jadders, Collision Data Short Frames (รูปที่ 2.5)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

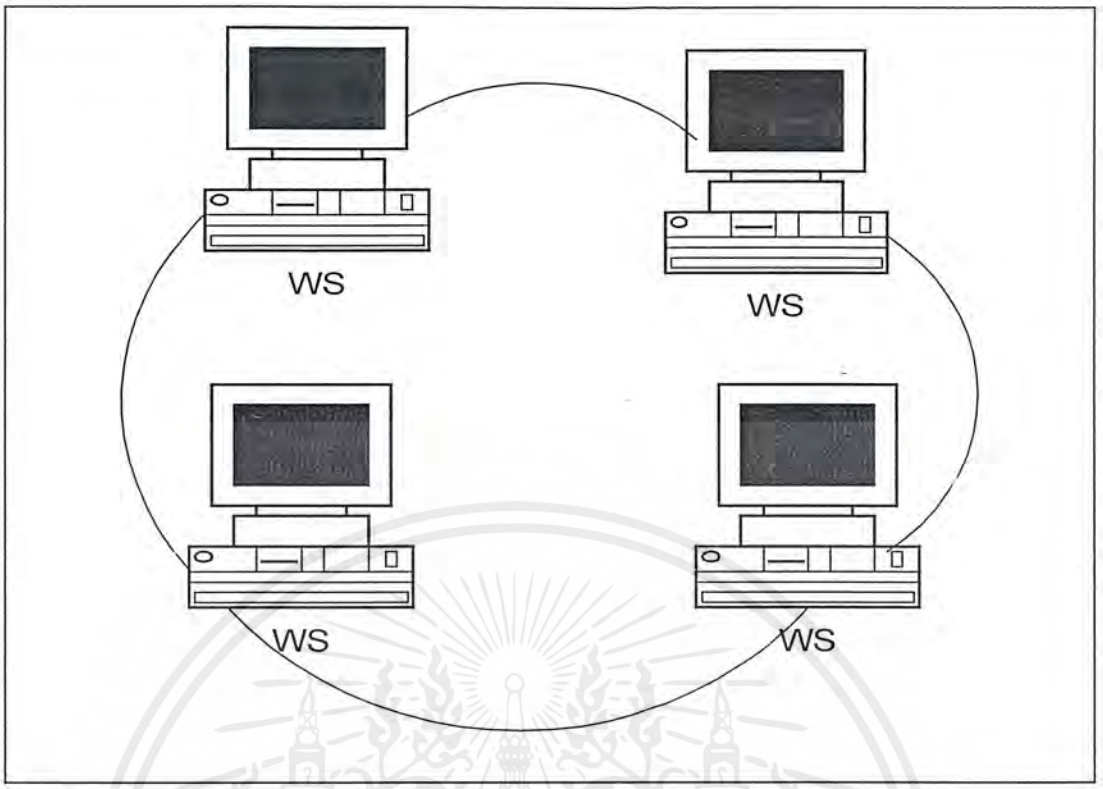


รูป2.5 แสดงการเชื่อมต่อแบบสตาร์ (star)

3). Ring

Ring การเชื่อมต่อแบบวงแหวน เป็นการเชื่อมต่อจากเครื่องหนึ่งไปยังอีกเครื่องหนึ่งจนครบวงจร ในการส่งข้อมูลจะส่งออกมาที่สายสัญญาณวงแหวน โดยจะเป็นการส่งผ่านเครื่องหนึ่งไปสู่อีกเครื่องหนึ่ง จนกว่าจะถึงปลายทาง ซึ่งจะคล้ายขบวนรถไฟที่ข้อมูลอยู่ในโบกี้รถไฟเมื่อข้อมูลวิ่งจะผ่านไปทุกเครื่องที่อยู่ในระบบที่เป็นดังรูป ถ้าเครื่องไหนเป็นจุดหมายปลายทางก็จะดึงข้อมูลออกไปแล้วส่งเฟรมต่อไป เมื่อถึงเครื่องต้นทางก็ดึงข้อมูลออกไปเป็นการเช็คว่าข้อมูลได้ส่งถึงปลายทางหรือไม่ ซึ่งระบบนี้เป็นระบบที่บริษัทไอบีเอ็ม (IBM) เป็นผู้คิดค้นและพัฒนา (รูปที่ 2.6)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูป2.6 แสดงการเชื่อมต่อแบบริง (Ring)

2.2.4 โพรโตคอลมาตรฐานของระบบเครือข่าย

หน่วยงานสากล ที่มีหน้าที่ในการกำหนดมาตรฐานของการออกแบบผลิตภัณฑ์อิเล็กทรอนิกส์ กำหนดรูปแบบการส่งสัญญาณ ฯลฯ คือ IEEE (Institute of Electrical and Electronics Engineers), EIA (Electronics-Industries Association) โดยจะมีโพรโตคอลอยู่ 3 แบบด้วยกันคือ

1. ARCnet
2. Etherrnet
3. token Ring

ARCnet เป็นโพรโตคอลที่ออกแบบโดยบริษัท Data Point ประมาณปี 1977 (Attached ARCnet Resource Computing network) ใช้หลักการออกแบบ "Transmission Permission ในการส่งข้อมูลจะมีการกำหนดตำแหน่งแอดเดรสของเครื่องเวิร์กสเตชั่นลงไปด้วย สามารถจะเชื่อมต่อได้ทั้ง แบบ Bus และ Star มีความเร็วในการส่งผ่านข้อมูลน้อยมากเพียง 2.5 Mbps (2.5 เมกะบิตต่อวินาที) ทำให้ไม่เป็นที่นิยมใช้งาน

Ethernet เป็นโพรโตคอลที่ออกแบบโดยบริษัท Xerox ประมาณปี 1970 ใช้หลักการทำงานแบบ CSMA/CD (Carrier Sense Multiple Access With Collision Detection) ในการส่งเมสเสจไปบนสายสัญญาณของระบบเครือข่าย ถ้าหากมีการส่งออกมาพร้อมกัน ย่อมจะเกิดการชนกัน(Collision) ของสัญญาณทำให้การส่งผ่านข้อมูลต้องหยุดลงทันทีCSMA/CD จะใช้วิธีของ Listen-

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

before-transmitting คือ ก่อนจะส่งสัญญาณรออกไปจะต้องตรวจสอบว่าในขณะนั้นมีเครื่องเวิร์กสเตชั่นเครื่องใดทำการรับ-ส่งเมสเสจบนสายเคเบิลอยู่หรือไม่ ถ้ามีต้องรอจนกว่าสายเคเบิลจะว่าง แล้วจึงส่งข้อมูลออกไปบนสายเคเบิล

ระบบโพรโตคอล Ethernet นั้นเป็นมาตรฐานของ IEEE 802.3 สามารถจะเชื่อมต่อได้ทั้งแบบ Bus และ Star โดยใช้สอย Coaxial หรือ สายทองแดงคู่ตีเกลียว (UTP = Unshielded Twisted Pair) มีความเร็วในการรับ – ส่งข้อมูล 10 Mbps (10 เมกะบิตต่อวินาที) ในปัจจุบันพัฒนาความเร็วเกิน 100 Mbps มีความยาวสูงสุดระหว่างเครื่องเวิร์กสเตชัน 2.8 กิโลเมตร ในการส่งผ่านสัญญาณอิเล็กทรอนิกส์ไปบนสายเคเบิลจะใช้แบบ Manchester – Encoded Digital Baseband และกล่าวถึงสัญญาณดิจิตอล 0-1 ในการส่งผ่านไปบนสายเคเบิล Ethernet มีรูปแบบการต่อสายเคเบิล 3 แบบ ด้วยกันคือ

1. 10 Base T
2. 10 Base 2
3. 10 Base 5

10 Base T เป็นรูปแบบในการต่อสายที่นิยมมาก "10" หมายถึง ความเร็วในการรับส่งข้อมูล (10 เมกะบิตต่อวินาที) " Base" หมายถึง ลักษณะการส่งข้อมูลแบบ Base band ซึ่งเป็นดิจิตอล และ T หมายถึง Twisted Pair (สายทองแดงคู่ตีเกลียว) สรุปแล้ว 10 Base T ก็คือ การใช้สาย Twisted Pair ในการรับ – ส่งข้อมูลมีความเร็ว 10 Mbps ด้วยสัญญาณแบบ Base band ปัจจุบันจะใช้สาย UTP (Unshield Twisted Pair) ซึ่งจะมีสายเส้นเล็ก ๆ ภายใน 8 เส้นตีเกลียวกัน 4 คู่

10 Base 2 เป็นรูปแบบในการต่อสายโดยใช้สาย Coaxial เรียกว่า Thin Coaxial สายมีความยาวไม่เกิน 180 เมตร

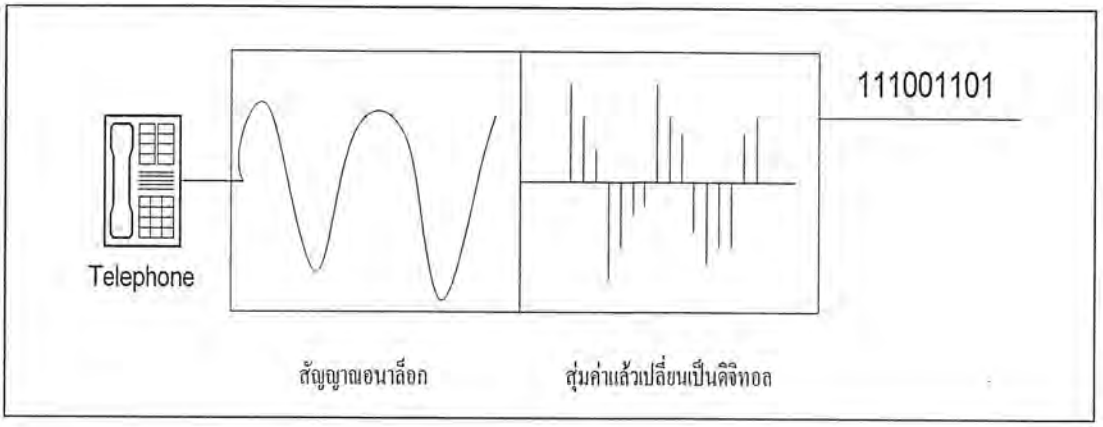
10 Base 5 เป็นรูปแบบในการต่อสายโดยใช้สาย Coaxial ขนาดใหญ่ มีเส้นผ่านศูนย์กลาง เรียกว่าสาย Thick Coaxial การเชื่อมต่อในแต่ละจุดจะมี Transceiver เป็นตัวเชื่อมต่อ และใช้สายเคเบิล AUI เชื่อมระหว่างเครื่องเวิร์กสเตชัน สายมีความยาวไม่เกิน 500 เมตร

2.3 เครือข่ายระบบสื่อสารแพ็กเกจเสียง

ทำความเข้าใจกับการเข้ารหัสเสียงเริ่มจากการใช้งานในส่วนบุคคลพอการใช้งานเริ่มมากขึ้น ทำให้เกิดเป็นระบบเครือข่ายขึ้น เกิดกระบวนการรับส่งข่าวสารขึ้น พัฒนาการบีบอัดข้อมูล โดยการประมวลเป็นดิจิตอลเพื่อประมวลและสื่อสารง่าย ซึ่งปัจจุบันมี DSP ใช้ประมวลสัญญาณความเร็วสูง เช่น โมเด็ม

การส่งสัญญาณแบบอนาล็อกมีข้อเสียหลายอย่าง โดยเฉพาะยากต่อการประมวลผลและจัดการทั้งยังมีสัญญาณอื่นมารบกวนได้มากและขจัดได้ยาก การขยายสัญญาณอนาล็อกก็ทำให้เกิดการขยายสัญญาณรบกวนให้มากขึ้นด้วย การรับส่งสัญญาณแบบอนาล็อกจึงมักได้คุณภาพที่ไม่ดีนัก เมื่อมีการพัฒนาเทคโนโลยี ก็มีการแปลงสัญญาณอนาล็อกนี้ให้เป็นสัญญาณดิจิตอล ที่ประกอบด้วยบิตที่มีค่าเป็น "0" หรือ "1" มีการเข้ารหัส โดยเฉพาะสัญญาณที่ใช้มีการเข้ารหัสแบบพัลซ์โค้ดมอดูเลชัน หรือ PCM (รูปที่ 2.7)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.7 แสดงการเปลี่ยนสัญญาณเสียง

ลักษณะของการแปลงสัญญาณ PCM ที่ใช้คือ มีการสุ่มค่าจำนวน 8,000 ครั้งต่อวินาที โดยแต่ละครั้งห่างกัน 125 ไมโครวินาที เมื่อสุ่มค่าได้แล้วก็แปลงเป็นรหัสดิจิตอลขนาด 8 บิต การสุ่มขนาด 125 ไมโครวินาทีนี้พอเพียงที่จะสุ่มสัญญาณอะนาล็อกที่มีความถี่ได้ไม่เกิน 4,000 เฮิรตซ์ ซึ่งพอเพียงสำหรับความถี่เสียงที่เราใช้งาน

การสุ่มค่า 8,000 ครั้งต่อวินาที และแต่ละครั้งแปลงสัญญาณเป็นดิจิตอลขนาด 8 บิต ทำให้ PCM ที่ใช้กับสัญญาณเสียงหนึ่ง ๆ มีขนาดความเร็ว 64,000 บิตต่อวินาที

การแปลงสัญญาณอนาล็อกเป็นดิจิตอลด้วยอัตราสุ่ม 8,000 ครั้งต่อวินาที เป็นไปตามมาตรฐานข้อกำหนดที่ชื่อว่า G.711 ขนาดของความกว้างของสัญญาณเสียงที่ใช้จึงมีค่าเท่ากับ 64 กิโลบิตต่อวินาที

อย่างไรก็ดี เพื่อลดขนาดความกว้างในการรับส่งให้ลดลง เพื่อจะได้ส่งได้ด้วยจำนวนช่องสัญญาณเสียงได้มากขึ้น ได้มีการกำหนดมาตรฐานใหม่ที่ชื่อ ADPCM หรือ Adaptive Differential PCM โดยมีลักษณะการเข้ารหัสเพียง 4 บิต ให้แต่ละการสุ่มเปรียบเทียบความแตกต่างจากครั้งก่อนหรือเก็บผลต่าง จึงทำให้การสุ่มเท่าเดิม แต่การเข้ารหัสเพียง 4 บิต จึงลดแถบกว้างให้เหลือเพียง 32 กิโลบิตต่อวินาทีได้ การลดขนาดลงครั้งหนึ่งนี้เป็นหนทางที่จะทำให้สัญญาณเสียงมีจำนวนช่องในการสื่อสารระบบสายสัญญาณดิจิตอลได้มากขึ้น

การลดขนาดแถบกว้างในการส่งยังทำได้อีก เช่น ถ้าใช้หลักการ ADPCM แต่เก็บเพียง 2 บิตที่เป็นผลต่างของข้อมูลการสุ่มครั้งก่อน ก็จะทำให้การรับส่งใช้แถบกว้างเพียง 16 กิโลบิตต่อวินาที แต่คุณภาพของเสียงจะลดลง

2.3.1 เทคนิคการบีบอัดข้อมูลได้รับการพัฒนา

ด้วยเทคนิคของการประมวลผลสัญญาณเชิงดิจิตอล ทำให้มีการพัฒนาเทคนิคของการบีบอัดข้อมูลที่มีประสิทธิภาพดียิ่งขึ้น อัลกอริทึมในการบีบอัดข้อมูลเสียงที่ได้พัฒนาและใช้กันแพร่หลายแล้ว คือ LPC-Linear Prediction Code วิธีการบีบอัดนี้มีรูปแบบเพื่อความเหมาะสมกับงานที่ใช้แทนเสียงพูด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การใช้งานทางโทรศัพท์ส่วนใหญ่ใช้แบบ PCM ที่เป็นแบบ 64 กิโลบิตต่อวินาที ทั้งนี้เพราะการประมวลผลสัญญาณแบบนี้ใช้กันมากในงานโทรศัพท์ที่แพร่หลายและมีอุปกรณ์รองรับอยู่มาก แต่เมื่อใช้แบบ LPC ซึ่งสามารถให้เหลือเพียงประมาณ 8 กิโลบิตต่อวินาทีได้

มาตรฐานการสร้างรหัสของข้อมูลเสียงจึงมีหลายมาตรฐาน โดย ITU เป็นผู้กำหนดโดยจัดอยู่ในข้อแนะนำที่เรียกว่า G-series ซึ่งประกอบด้วยมาตรฐานที่สำคัญคือ

- G.711 เป็นมาตรฐานที่ใช้สัญญาณเข้ารหัสแบบ PCM มีขนาด 64,000 บิตต่อวินาที ใช้ในกิจการโทรศัพท์ รวมถึงการสวิตช์ใน PBX ทั่วไป
- G.726 เป็นมาตรฐานการบีบอัดเสียงด้วยวิธีการ ADPCM ทำให้ได้ขนาดสัญญาณลดลงเป็น 40, 32, 24 และ 16 กิโลบิตต่อวินาที เป็นที่นิยมใช้ในการสร้างแพ็กเกจเสียงที่ใช้ในสวิชชิงแบบ PBX
- G.728 เป็นมาตรฐานที่ใช้หลักการของ LPC ที่ใช้ชื่อว่า GELP Code Excited Linear Prediction สามารถบีบอัดสัญญาณให้เหลือเพียง 16 กิโลบิตต่อวินาที G.728 เป็นมาตรฐานที่ใช้กันในการแปลงสัญญาณเสียงเพื่อส่งกันระหว่าง PBX
- G.729 ใช้ CELP เช่นกัน แต่บีบอัดข้อมูลเสียงให้เหลือเพียงแถบกว้าง 8 กิโลบิตต่อวินาที มีรูปแบบมาตรฐานอยู่สองรูปแบบ คุณภาพของเสียงยังคงได้เท่ากับแบบ ADPCM
- G.723.1 เป็นมาตรฐานที่สร้างสายสัญญาณเสียงที่ใช้ในเรื่องมัลติมีเดีย ซึ่งใช้กันในระบบการรับส่งข้อมูลที่ความเร็วไม่สูงมาก สามารถบีบอัดข้อมูลให้เหลือเพียง 5.3 และ 6.3 กิโลบิตต่อวินาที โดยคุณภาพของเสียงยังดี

2.3.2 คุณภาพกับการบีบอัด

การบีบอัดย่อมทำให้คุณภาพตกลง แต่คุณภาพยังอยู่ที่ระดับความพอใจ ดังนั้นการบีบอัดจึงเป็นเทคนิคที่จะทำให้ได้ระดับความพอใจ และยิ่งบีบอัดได้มากเท่าไร โดยที่ยังอยู่ในระดับความพอใจได้ก็นับว่าเป็นเรื่องที่ดี ในการสื่อสารสัญญาณเสียงในย่านความถี่นี้ โดยจะใช้การวัดคุณภาพความพอใจใช้มาตรฐานที่ชื่อว่า MUS-Mean Opinion Score หรือให้เป็นคะแนน โดยให้คะแนนจาก 0 ถึง 5 ถ้า 0 หมายถึงคุณภาพแย่ และ 5 คือคุณภาพเยี่ยม

เมื่อนำเทคนิคต่าง ๆ มาหาค่า MUS ผลปรากฏว่า PCM ได้คะแนน MOS=4.4 ขณะที่G.726 ADPCM ได้คะแนน 4.2 และ G.728 ได้คะแนน 4.2 เท่ากัน G.729 ก็ยังได้คะแนน 4.2 แสดงให้เห็นว่าเทคนิคการบีบอัดที่ดีก็ยังทำให้ระดับคุณภาพอยู่ในเกณฑ์ดีได้แต่ต้องเสียเวลาในการประมวลผล

ถึงแม้ว่าจะได้คุณภาพที่มีความพอใจสูง แต่ก็ต้องเสียเวลาหน่วงหรือที่เรียกว่า delay time ทั้งนี้เพราะการบีบอัดข้อมูลต้องใช้หลักการคำนวณทางคณิตศาสตร์ช่วย ทำให้ต้องเสียเวลาการเข้ารหัสและการถอดรหัสทำให้มีเวลาหน่วงสูงกว่า

เวลาหน่วงนี้เป็นผลให้สัญญาณเสียงที่รับส่งจะล่าช้าไป และอาจสร้างปัญหาในเรื่องการสะท้อนเสียงกลับ ซึ่งต้องมีวิธีในการแก้ปัญหานี้ (ตารางที่ 2.1)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เทคนิคการบีบอัดเสียง	คะแนน MOS	การหน่วง(ms)
PCM (G.711)	4.4	0.75
32K ADPCM (G.726)	4.2	1
16K LD-CELP (G.728)	4.2	3 –5
8K CS-ACELP (G.729)	4.2	10
8K CS-ACELP (G.729a)	4.2	10
6.3K MPMLQ (G.723.1)	3.98	30
5.3K ACLEP (G.723.1)	3.5	30

ตารางที่ 2.1 เป็นตารางเปรียบเทียบเทคนิควิธีการต่าง ๆ กับคุณภาพและเวลาหน่วง

ซึ่งจะเห็นได้ชัดว่าเวลาหน่วงจะมีมากขึ้นเมื่อต้องการบีบอัดสัญญาณให้ได้มาก การหน่วงเวลาจะเป็นปัญหาใหญ่ปัญหาหนึ่งของระบบสื่อสารโทรคมนาคมในปัจจุบัน อุปกรณ์ในการสวิตช์แพ็กเกจเสียงต้องเสียเวลาเพิ่มมากขึ้น และเมื่อรวม ๆ กันแล้วจะมีมากขึ้น ซึ่งอาจถึงระดับที่เป็นปัญหาได้แต่หากดูจากตารางการเข้ารหัสแล้วพบว่าเสียเวลาเพิ่มอีกเพียงเล็กน้อย ดังนั้นสภาพการเสียเวลาหน่วงในการรับส่งและประมวลผลจึงน่าจะอยู่ในระดับการยอมรับได้ อีกทั้งการประมวลผลดิจิตอลมีแนวโน้มที่พัฒนาให้ดียิ่งขึ้นต่อไปอีก ปัญหาในเรื่องเวลาหน่วงจึงไม่ใช่ปัญหาใหญ่ทีเดียว

เมื่อสามารถเข้ารหัสสัญญาณเสียงที่มีประสิทธิภาพได้ สัญญาณเสียงจึงมีรูปลักษณะเป็นข้อมูลดิจิตอลที่มีขนาดไม่มากนัก เพียงประมาณ 8 กิโลบิตต่อวินาที ซึ่งนั่นหมายถึงสามารถส่งไปยังเครือข่ายคอมพิวเตอร์หรือเครือข่ายสื่อสารอื่น ๆ ได้ รูปแบบการผสมผสานสัญญาณเสียงไปในเครือข่ายคอมพิวเตอร์และเครือข่ายอื่นที่ดีและเหมาะสมจึงต้องใช้การรับส่งเป็นแพ็กเกต

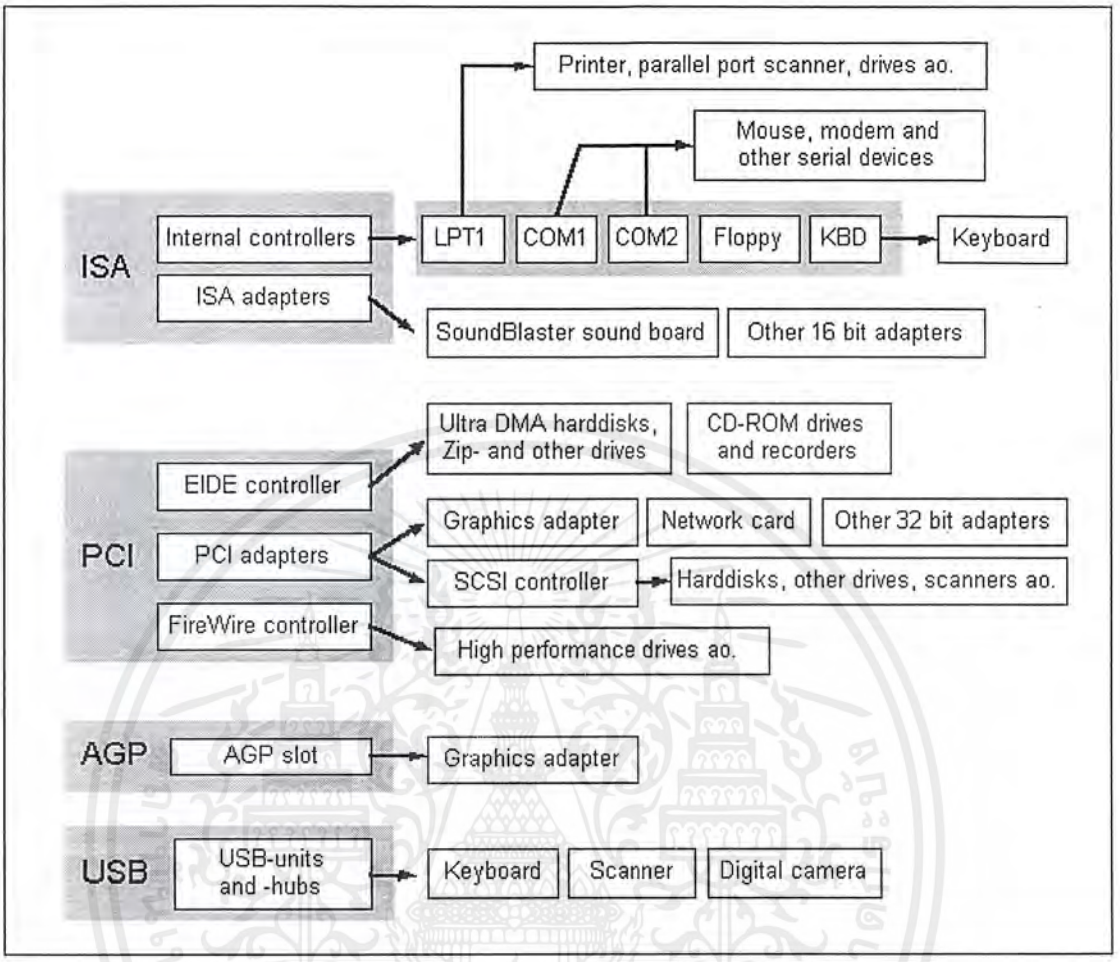
2.4 หลักการเบื้องต้นในการติดต่อกับไมโครคอมพิวเตอร์

หลักการทั่วไปในการติดต่อระว่างอุปกรณ์อินพุทเอาท์พุท กับเครื่องไมโครคอมพิวเตอร์ซึ่งเป็นคอมพิวเตอร์ส่วนบุคคล ซึ่งสามารถติดต่อได้โดยผ่านทางพอร์ทมาตรฐานที่มีอยู่แล้ว เช่น พอร์ทคอม1 (com1) , คอม2 (com2) , แอลพีที1 (LPT1) , แอลพีที2 (LPT2) เป็นต้น และยังมีอีกวิธีนึ่งคือการติดต่อผ่านทางสล๊อต (SLOT) ซึ่งมีช่องสัญญาณมาตรฐานต่าง ๆ ให้ติดต่อ (รูปที่ 2.8)

สัญญาณที่ใช้ติดต่อผ่านทางสล๊อตคอมพิวเตอร์

1. สัญญาณที่ใช้ติดต่อบนสล๊อต
2. การอินเตอร์รัพท์ (Interrupt) หน่วยประมวลผล
3. การอ้างตำแหน่งพอร์ทอุปกรณ์อินพุทเอาท์พุทของหน่วยประมวลผล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

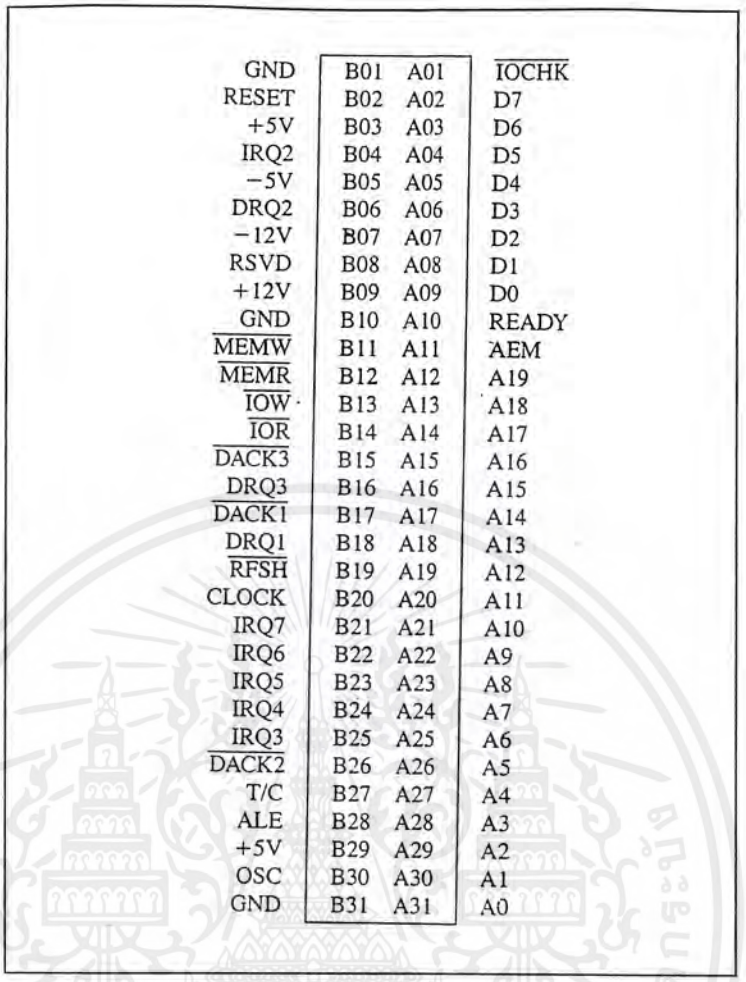


รูป 2.8 โครงสร้างการติดต่อบัสกับอุปกรณ์แบบต่างๆ

2.4.1 สัญญาณต่าง ๆ บนสล๊อต

ภายใน IBM/PC ได้มีการออกแบบให้สามารถที่จะเพิ่มเติมวงจรเชื่อมต่อเข้าไปในภายหลังได้ โดยผ่านทางสล๊อตที่อยู่บนเมนบอร์ด (Main Board) สำรับสล๊อตบนเมนบอร์ดนี้จะมีจำนวน 5 สล๊อต ซึ่งแต่ละสล๊อตจะมีจำนวนขาทั้งสิ้น 62 ขา แบ่งออกเป็น 2 ข้าง ข้างละ 31 ขา ส่วนการเรียกตำแหน่งขาของสล๊อตเหล่านี้จะขึ้นอยู่กับว่าขานั้นอยู่ข้างใด (ซ้ายหรือขวา) ของสล๊อต โดยขาที่อยู่ทางด้านซ้ายของสล๊อตจะเรียกโดยใช้อักษร "B" นำหน้าเลขตำแหน่งของขา เช่น ขา B16 ก็คือขาทางด้านซ้ายของสล๊อตขาที่ 16 (นับจากทางด้านท้ายของเครื่อง) ส่วนขาที่อยู่ทางด้านขวาของสล๊อตจะเรียกโดยใช้อักษร "A" นำหน้าเลขตำแหน่งของขา เช่น ขา A24 ก็คือขาทางด้านขวาของสล๊อต ขาที่24 (นับจากทางด้านท้ายของเครื่อง) (รูปที่ 2.9)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



GND	B01	A01	$\overline{\text{IOCHK}}$
RESET	B02	A02	D7
+5V	B03	A03	D6
IRQ2	B04	A04	D5
−5V	B05	A05	D4
DRQ2	B06	A06	D3
−12V	B07	A07	D2
RSVD	B08	A08	D1
+12V	B09	A09	D0
GND	B10	A10	READY
$\overline{\text{MEMW}}$	B11	A11	AEM
$\overline{\text{MEMR}}$	B12	A12	A19
$\overline{\text{IOW}}$	B13	A13	A18
$\overline{\text{IOR}}$	B14	A14	A17
$\overline{\text{DACK3}}$	B15	A15	A16
DRQ3	B16	A16	A15
$\overline{\text{DACK1}}$	B17	A17	A14
DRQ1	B18	A18	A13
$\overline{\text{RFSH}}$	B19	A19	A12
CLOCK	B20	A20	A11
IRQ7	B21	A21	A10
IRQ6	B22	A22	A9
IRQ5	B23	A23	A8
IRQ4	B24	A24	A7
IRQ3	B25	A25	A6
$\overline{\text{DACK2}}$	B26	A26	A5
T/C	B27	A27	A4
ALE	B28	A28	A3
+5V	B29	A29	A2
OSC	B30	A30	A1
GND	B31	A31	A0

รูปที่ 2.9 แสดงโครงสร้างของสล้อตคอมพิวเตอร์แบบ ISA BUS

แต่ละขาของสล๊อตเหล่านี้จะเชื่อมต่อกับเส้นสัญญาณต่าง ๆ บนเมนบอร์ด ทำให้การสร้างวงจรอิน-เทอร์เฟสกับ IBM/PC สามารถทำได้โดยสะดวก ซึ่งเส้นสัญญาณที่เชื่อมต่อกับขาของสล๊อตเหล่านี้จะประกอบไปด้วยเส้นสัญญาณของบัสแอดเดรส (Address Bus) , บัสข้อมูล (Data Bus) , บัสควบคุมสำหรับการเขียน อ่านข้อมูลจากหน่วยความจำ หรือพอร์ท ไอโอ (I/O) , เส้นสัญญาณสำหรับการขออินเทอร์รัพท์ของวงจรเชื่อมต่อ , เส้นสัญญาณแสดงการรีเฟรชหน่วยความจำและสัญญาณสำหรับการตรวจสอบความผิดพลาด (I/O CHCK)

นอกจากสัญญาณเหล่านี้แล้ว สล๊อตบนเมนบอร์ดยังเชื่อมต่อกันแหล่งจ่ายไฟต่าง ๆ ที่ใช้ในระบบอีกด้วย คือ +5Vcd , -5Vcd , +12Vcd และ –12 Vcd

แต่จะขอกล่าวเฉพาะสัญญาณที่จำเป็นต้องใช้เท่านั้นคือ

ขาสัญญาณทั้ง 20 ขานี้เป็นเอาท์พุท ซึ่งใช้สำหรับกำหนดแอดเดรสของหน่วยความจำหรืออุปกรณ์ I/O ที่ 8088 ต้องการติดต่อด้วย โดยที่สัญญาณ A_0 จะมีนัยสำคัญต่ำสุด (Least Significant Bit) และ A_{19} จะมีนัยสำคัญสูงสุด (Most Significant Bit) สำหรับค่าแอดเดรสบนบัสแอดเดรส A_0 - A_{19} นี้จะถูกกำหนดโดย 8088 ในระหว่างขบวนการอ่าน เขียนข้อมูลลงในหน่วยความจำหรืออุปกรณ์ อุปกรณ์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

อินพุทเอาท์พุท แต่ในช่วงของขบวนการ DMA นั้น DMA – Controller จะเป็นผู้กำนดค่าแอดเดรสบนบัสแอดเดรสเอง (ในระหว่างนี้ 8088 จะถูกตัดออกจากระบบ)

จะเห็นได้ว่าจำนวนเส้นแอดเดรสนี้จะมีอยู่ 20 เส้น ซึ่งสามารถที่จะอ้างแอดเดรสของหน่วยความจำได้ถึง 1 เมกะไบท์ แต่อย่างไรก็ตามจะมีแอดเดรสบางแอดเดรสที่ถูกใช้งานโดย IBM/PC อยู่ก่อนแล้ว คือแอดเดรสของหน่วยความจำแรม (RAM) บนเมนบอร์ดที่ถูกใช้โดยระบบจำนวน 64 กิโลไบท์ (สำหรับ IBM PC/XT จะเป็นจำนวน 256 กิโลไบท์) และแอดเดรสสำหรับหน่วยความจำรอม (ROM) อีก 48 กิโลไบท์ ซึ่งถูกจัดในช่วงของแอดเดรสบนสุดใน 1 เมกะไบท์ คือ OFCOOH จนถึง OFFFFFH (สำหรับ IBM PC/XT จะเป็น 64 กิโลไบท์)

สำหรับการอ้างแอดเดรสของพอร์ท I/O นั้น จะใช้เส้นแอดเดรสเพียง 16 เส้น คือ A_0 - A_{15} ซึ่งจะทำให้อ้างแอดเดรสของพอร์ทได้ 64 กิโลพอร์ท โดยผ่านทางชุดคำสั่ง IN และ OUT ส่วนเส้นแอดเดรสที่เหลือคือ $A_{16} – A_{19}$ นั้นจะไม่ถูกใช้งาน อย่างไรก็ตามภายใน IBM/PC จะใช้เส่นแอดเดรสในการอ้างแอดเดรสของพอร์ทเพียง 10 เส้น คือจาก A_0 - A_9 และค่าแอดเดรสที่ใช้งานจะต้องอยู่ในช่วง 0200H จะถึง 03FFH เท่านั้น

D_0 - D_7 (Data Bus; ขา A9-A2) :

ขาสัญญาณนี้จะเป็นแบบส่งข้อมูลด้สองทิศทาง (Bi – Directional) ซึ่งต่อกับบัสข้อมูลของระบบ เพื่อทำหน้าที่ในการส่งผ่านข้อมูลระหว่างพอร์ท I/O กับ IBM/PC โดยบิต D_0 จะมีนัยสำคัญต่ำสุด และบิต D_7 จะมีนัยสำคัญสูงสุด

สำหรับในบัสไซเคิลของการเขียนข้อมูลที่สร้างขึ้นโดย 8088 นั้น ข้อมูลจะถูกส่งออกมาบนบัสข้อมูลก่อนที่สัญญาณ $\overline{IOW}$ (ในกรณีที่ต้องการส่งข้อมูลให้กับพอร์ท) หรือ $\overline{MAMW}$ (ในกรณีที่ต้องการส่งข้อมูลให้กับหน่วยความจำ) จะเปลี่ยนจากลอจิก "0" เป็นลอจิก "1" (ขอบขาขึ้น) ซึ่งโดยทั่วไปขอบขาขึ้นของสัญญาณ $\overline{IOW}$ หรือ $\overline{MAMW}$ นี้จะถูกใช้เพื่อสั่งให้พอร์ท I/O หรือหน่วยความจำที่มีแอดเดรสตรงกับค่าแอดเดรสบนบัสแอดเดรสนั้นรับข้อมูลไปเก็บไว้

สำหรับในบัสไซเคิลของการอ่านข้อมูลที่สร้างขึ้นโดย 8088 นั้น พอร์ท I/O หรือหน่วยความจำที่ถูกอ้างอิงถึงจะต้องส่งข้อมูลออกมาบนบัสข้อมูล ก่อนที่สัญญาณ $\overline{IOR}$ (ในกรณีที่ต้องการอ่านข้อมูลจากพอร์ท) หรือ $\overline{MEMR}$ (ในกรณีต้องการอ่านข้อมูลจากหน่วยความจำ) จะเปลี่ยนจากลอจิก "0" (ขอบขาขึ้น)

ALE (Address Latch Enable; ขา B28) :

ขาสัญญาณนี้เป็นสัญญาณเอาท์พุทที่ 8288 Bus Controller สร้างขึ้นเพื่อใช้สำหรับแสดงการเริ่มต้นของบัสไซเคิล และแสดงให้อุปกรณ์ภายนอกทราบว่าแอดเดรสที่ 8088 ต้องการจะติดต่อด้วยนั้นถูกส่งออกมาบนบัสแอดเดรสแล้ว โดยที่สัญญาณ ALE นี้จะเปลี่ยนจากลอจิก "1" เป็น "0" เมื่อค่าแอดเดรสที่ถูกต้องถูกส่งออกมาบนบัสข้อมูลเรียบร้อยแล้ว ดังนั้นขอบขาลงของสัญญาณ ALE นี้จะถูกใช้ในการแลทช์ค่าบัสแอดเดรส/ข้อมูล (Address/Data Bus; $AD_0 – AD_7$) ของ 8088 ทำให้สามารถ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

แยกค่าแอดเดรส (A_0 – A_{19}) และข้อมูล (A_0 – A_7) ออกจากกันได้ อย่างไรก็ตามสัญญาณ ALE จะแอกทีฟเฉพาะที่บัสไซเคิลที่สร้างขึ้นโดย 8088 เท่านั้น โดยจะไม่แอกทีฟในระหว่างขบวนการ DMA

$\overline{IOW}$ **(I/O Write; ขา B13) :**

ขาสัญญาณนี้เป็นเอาท์พุทแอคทีฟที่ลอจิก "0" ซึ่งถูกสร้างขึ้นโดย 8288 Bus Controllr เพื่อใช้แสดงว่าบัสไซเคิลที่เกิดขึ้นนี้เป็นบัสไซเคิลของการเขียนข้อมูลลงบนพอร์ท I/O เพื่อให้พอร์ท I/O ที่มีแอดเดรสตรงกับแอดเดรสบนบัสแอดเดรสนั้น รับข้อมูลที่อยู่บนบัสข้อมูลไปเก็บไว้ อย่างไรก็ตามเนื่องจากในช่วงเวลาที่สัญญาณ $\overline{IOW}$ นี้แอคทีฟ (ลอจิก "0") นั้นข้อมูลบนบัสข้อมูลอาจจะยังไม่สมบูรณ์ ดังนั้นในการออกแบบจึงควรใช้ขอบขาขึ้นของสัญญาณ $\overline{IOW}$ แทนขอบขาลงในการทำให้พอร์ท I/O ที่เกี่ยวข้องรับข้อมูลไปเก็บไว้ เพื่อให้ข้อมูลบนบัสข้อมูลสมบูรณ์เสียก่อน สำหรับในขบวนการ DMA – Controller จะทำการสร้างสัญญาณ $\overline{IOW}$ เอง โดยที่ค่าแอดเดรสที่อยู่บนบัสแอดเดรสจะเป็นค่าแอดเดรสของหน่วยความจำที่พอร์ท I/O ที่ขอ DMA ต้องการจะอ่านข้อมูล

$\overline{IOR}$ **(I/O Read; ขา B14) :**

ขาสัญญาณนี้เป็นเอาท์พุทแอคทีปที่ลอจิก "0" ที่สร้างขึ้นโดย 8288 Bus Controller เพื่อใช้ในการแสดงว่าบัสไซเคิลที่เกิดขึ้นนี้ เป็นบัสไซเคิลของการอ่านข้อมูลจากพอร์ท I/O ที่มีแอดเดรสตรงกับแอดเดรสบนบัสแอดเดรสนั้นส่งข้อมูลออกมาบนบัสข้อมูลโดยข้อมูลต้องถูกส่งออกมาบนบัสข้อมูลก่อนขอบขาขึ้นของสัญญาณ $\overline{IOR}$ ประมาณ 30 นาโนวินาที เพื่อได้มั่นใจได้ว่า 8088 สามารถรับข้อมูลได้ถูกต้อง สำหรับในขบวนการ DMA 8237A – DMA Controller จะทำการสร้างสัญญาณ $\overline{IOR}$ เองโดยที่ค่าแอดเดรสที่อยู่บนบัสแอดเดรสจะเป็นค่าแอดเดรสของหน่วยความจำ (แทนที่จะเป็นแอดเดรสของพอร์ท I/O) ที่พอร์ท I/O ที่ของ DMA ต้องการจะนำข้อมูลไปเก็บที่พอร์ทใดจะส่งข้อมูลออกมาบนบัสข้อมูลนั้น จะอาศัยสัญญาณ DACK จาก DMA Controller เป็นตัวกำหนดเช่นกรณีที่สัญญาณ DACK แอคทีฟก็จะแสดงว่าพอร์ท I/O ที่จะต้องส่งข้อมูลออกมาบนบัสข้อมูลก็คือพอร์ท I/O ที่ขอ DMA ผ่านทางแขนแนลที่ 1 (DRQ_1) เป็นต้น

$\overline{IOW}$ **(I/O Write; ขา B13) :**

ขาสัญญาณนี้เป็นเอาท์พุทแอคทีฟที่ลอจิก "0" ซึ่งถูกสร้างขึ้นโดย 8288 Bus Controllr เพื่อใช้แสดงว่าบัสไซเคิลที่เกิดขึ้นนี้เป็นบัสไซเคิลของการเขียนข้อมูลลงบนพอร์ท I/O เพื่อให้พอร์ท I/O ที่มีแอดเดรสตรงกับแอดเดรสบนบัสแอดเดรสนั้น รับข้อมูลที่อยู่บนบัสข้อมูลไปเก็บไว้ อย่างไรก็ตามเนื่องจากในช่วงเวลาที่สัญญาณ $\overline{IOW}$ นี้แอคทีฟ (ลอจิก "0") นั้นข้อมูลบนบัสข้อมูลอาจจะยังไม่สมบูรณ์ ดังนั้นในการออกแบบจึงควรใช้ขอบขาขึ้นของสัญญาณ $\overline{IOW}$ แทนขอบเขาลงในการทำให้พอร์ท I/O ที่เกี่ยวข้องรับข้อมูลไปเก็บไว้ เพื่อให้ข้อมูลบนบัสข้อมูลสมบูรณ์เสียก่อน สำหรับในขบวนการ DMA-Controller จะทำการสร้างสัญญาณ $\overline{IOW}$ เอง โดยที่ค่าแอดเดรส

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ที่อยู่บนบัสแอดเดรสจะเป็นค่าแอดเดรสของหน่วยความจำที่พอร์ท I/O ที่ขอ DMA ต้องการจะอ่านข้อมูล

2.5 การเรียกใช้งาน Function ในการทำงานเกี่ยวกับเสียง ด้วยฟังก์ชัน API

ฟังก์ชัน API ด้านมัลติมีเดียในกลุ่มของ Command-Message และ Command-String เป็นฟังก์ชันที่เหมาะกับโปรแกรมประเภท VB/Win เป็นอย่างมาก กลุ่มฟังก์ชันระดับล่าง (Low Level Waveform Audio Function) ซึ่งเป็นกลุ่มฟังก์ชัน API ด้านมัลติมีเดียที่ได้รับการออกแบบสำหรับภาษาซีโดยเฉพาะ และประกอบด้วยฟังก์ชันที่สามารถใช้จัดการกับไฟล์ .WAV เช่น การอัดเสียง (Recording) การทำเทคนิคเสียงพิเศษ (Sound effect) การควบคุมระดับโทนเสียง (Pitch Control) เป็นต้น

2.5.1 รายละเอียดคำนำหน้าฟังก์ชัน

aux ฟังก์ชันด้าน Auxiliary Audio

joy ฟังก์ชันด้าน Joystick

mci ฟังก์ชันด้าน Media Control Interface

midi ฟังก์ชันด้าน Low-Level MIDI Audio

mmio ฟังก์ชันด้านการจัดการ I/O ของไฟล์มัลติมีเดีย

snd ฟังก์ชันด้าน High-Level Sound

time ฟังก์ชันด้าน Timer Event

wave ฟังก์ชันด้าน Low-Level Waveform Audio

ฟังก์ชันที่นำหน้าด้วยคำว่า wave ที่เป็นฟังก์ชันระดับล่างที่จัดการดับไฟล์ .wav ซึ่งสามารถจัดกลุ่มได้ดังนี้

2.5.2 กลุ่มของฟังก์ชัน

ไมโครซอฟท์ได้ทำการแบ่งกลุ่มของฟังก์ชันระดับล่างด้านเสียงออกตามลักษณะหน้าที่การให้บริการ การแยกกลุ่มของฟังก์ชันด้านมัลติมีเดียซึ่งเป็นรูปแบบเฉพาะของไมโครซอฟท์โดยไมโครซอฟท์ได้ใช้คำนำหน้า (prefix) นำหน้าชื่อของฟังก์ชันเพื่อกำหนดความสัมพันธ์ของแต่ละฟังก์ชันดังต่อไปนี้

2.5.2.1 กลุ่มฟังก์ชันด้าน Querying

ก่อนจะทำการเล่นหรืออัดเสียงด้วยไฟล์ .wav จะต้องทำการสำรวจความสามารถของฮาร์ดแวร์ในเครื่องคอมพิวเตอร์เสียก่อน ทั้งนี้เพื่อให้สามารถจัดการกับไฟล์ .wav ได้อย่างถูกต้อง ซึ่งประกอบด้วยฟังก์ชันดังต่อไปนี้

waveInGetNumDevs รายงานจำนวนอุปกรณ์รับสัญญาณเสียง(wavform) ที่ติดตั้งในระบบ

waveInGetDevCaps รายงานความสามารถของแต่ละอุปกรณ์รับสัญญาณเสียง

waveOutGetNumDevs รายงานจำนวนอุปกรณ์ส่งสัญญาณเสียง (waveform) ที่ติดตั้งในระบบ

waveOutGetDevCaps รายงานความสามารถของแต่ละอุปกรณ์ส่งสัญญาณเสียง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.5.2.2 กลุ่มฟังก์ชันด้านการเปิดและปิดอุปกรณ์

ก่อนการเล่นกลับหรือการอัดเสียงในรูปแบบ .wav ทุกครั้ง จะต้องทการเปิดอุปกรณ์ที่เกี่ยวข้องเสียก่อน เพื่อเป็นการอ้างสิทธิ์การใช้งานสำหรับแอพพลิเคชั่นนั้นๆ และควรจะปิดอุปกรณ์เมื่อไม่ต้องการใช้งานอีกต่อไป เพื่อให้แอพพลิเคชั่นอื่นๆ สามารถอ้างสิทธิ์การใช้งานได้ต่อไป ซึ่งประกอบด้วยฟังก์ชันดังต่อไปนี้

waveInOpen	เปิดอุปกรณ์รับสัญญาณเสียงสำหรับการอัดเสียง
waveInClose	ปิดอุปกรณ์รับสัญญาณเสียงที่กำหนด
waveOutOpen	เปิดอุปกรณ์ส่งสัญญาณเสียงสำหรับการเล่นกลับ
waveOutClose	ปิดอุปกรณ์ส่งสัญญาณเสียงที่กำหนด

2.5.2.3 กลุ่มของฟังก์ชันด้านการอ่านค่า ID ของอุปกรณ์จัดการสัญญาณเสียง

เนื่องจากการจัดการออบเจ็คใดๆ ของวินโดวส์ มันจะใช้รหัส ID ของแต่ละออบเจ็คในการควบคุมการทำงานหรือสั่งการ ซึ่งอุปกรณ์ด้านมัลติมีเดียก็เช่นกัน ก่อนที่วินโดวส์จะมีการติดต่อกับอุปกรณ์เหล่านี้ก็จะมีการกำหนดหมายเลข ID (device handle) ให้เสียก่อน ซึ่งสามารถอ่านหมายเลข ID โดยใช้ฟังก์ชันต่อไปนี้

waveInGetID	อ่านหมายเลข ID ของอุปกรณ์รับสัญญาณเสียง
waveOutGetID	อ่านหมายเลข ID ของอุปกรณ์ส่งสัญญาณเสียง

2.5.2.4 กลุ่มฟังก์ชันด้านการเล่นข้อมูลในรูปแบบสัญญาณเสียง

ฟังก์ชันในกลุ่มนี้ได้เตรียมไว้สำหรับการจัดเตรียมกลุ่มของข้อมูล (block) เพื่อใช้ส่งต่อให้กับอุปกรณ์ส่ง สัญญาณเสียงต่อไปนี้ ทั้งนี้เนื่องจากการจัดเก็บข้อมูลในรูปแบบสัญญาณเสียงลงในไฟล์นั้น ยังไม่เหมาะสำหรับการจัดส่งให้กับอุปกรณ์ส่งสัญญาณเสียงที่เดียว เพราะข้อมูลในไฟล์ .wav นอกจากจะมีการจัดเก็บข้อมูลของสัญญาณเสียงแล้ว ยังมีการจัดเก็บค่าต่างๆ เอาไว้ในส่วนของ header เพื่อใช้เป็นข้อมูลสำหรับการจัดการกับบัฟเฟอร์หรือข้อมูลในไฟล์ต่อไป โดยหัวข้อเกี่ยวกับไฟล์ในรูปแบบของ RIFF ซึ่งประกอบด้วยฟังก์ชันดังต่อไปนี้

waveOutPrepareHeader	เตรียมความพร้อมของดีไวซ์ไดรเวอร์สำหรับการส่งสัญญาณเสียง และเป็นการบอกดีไวซ์ไดรเวอร์ให้ทราบว่ากลุ่มของข้อมูลที่จะส่งมาเป็นข้อมูลสำหรับการเล่นกลับ
waveOutPrepareHeader	เขียนกลุ่มของข้อมูลให้กับอุปกรณ์ส่งสัญญาณเสียง
waveOutUnprepreHeader	บอกดีไวซ์ไดรเวอร์ส่งสัญญาณเสียงสามารถยกเลิกหรือลบทิ้ง การเตรียมการกับกลุ่มของข้อมูลที่ถูกส่งมาเป็นข้อมูลสำหรับการเล่นกลับ

2.5.2.5 กลุ่มของฟังก์ชันด้านการอัดสัญญาณเสียง

นอกจากจะสามารถเล่นกับไฟล์ .wav อัดสัญญาณเสียงที่ได้รับจากอุปกรณ์รับสัญญาณเสียงลงในไฟล์ .wav ด้วยฟังก์ชันในกลุ่มนี้ ซึ่งเสียงที่รับเข้ามาจะถูกจัดเก็บเอาไว้ในบัฟเฟอร์ซึ่งเป็นพื้นที่ในหน่วยความจำซึ่งเตรียมเอาไว้สำหรับการขนถ่ายข้อมูลในระหว่างการอัดสัญญาณเสียง ซึ่งประกอบด้วยฟังก์ชันดังต่อไปนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

-waveInAddBuffer ส่งบัฟเฟอร์ให้กับอุปกรณ์สำหรับการส่งสัญญาณเสียง ซึ่งบัฟเฟอร์นี้จะถูกใช้จัดเก็บข้อมูลในรูปแบบสัญญาณเสียงและจะถูกส่งกลับมายังแอพพลิเคชันต้นทางต่อไป

-waveInPrepareHeader บอกให้อุปกรณ์รับสัญญาณเสียงทราบว่า กลุ่มของข้อมูลที่จะส่งมาเป็นข้อมูลสำหรับการอัดเสียง

-waveInUnprepareHeader บอกดีไวซ์ไดร์เวอร์สำหรับการรับสัญญาณเสียงสามารถยกเลิกหรือลบทิ้งการเตรียมการกับกลุ่มของข้อมูลที่ถูกส่งมาเป็นข้อมูลสำหรับการอัดเสียง

2.5.2.6 กลุ่มฟังก์ชันด้านการอ่านตำแหน่งปัจจุบันของอุปกรณ์สัญญาณเสียง

ในขณะที่ทำการเล่นกลับหรืออัดเสียงด้วยไฟล์ .wav จะสามารถอ่านตำแหน่งการจัดการข้อมูลในปัจจุบันจากอุปกรณ์ได้โดยใช้ฟังก์ชันในกลุ่มนี้ ดังต่อไปนี้

-waveInGetPosition อ่านตำแหน่งการอัดเสียงปัจจุบันจากอุปกรณ์รับสัญญาณเสียง

-waveOutGetPoiton อ่านตำแหน่งการอัดเสียงปัจจุบันจากอุปกรณ์ส่งสัญญาณเสียง

2.5.2.7 กลุ่มฟังก์ชันด้านการควบคุมการเล่นกลับ

ฟังก์ชันในกลุ่มนี้จะทำหน้าที่ควบคุมวิธีการเล่นกลับทั้งหมดของอุปกรณ์ส่งสัญญาณเสียง เช่น Pause, Stop หรือ Play เป็นต้น ซึ่งการเล่นกลับนั้นจะเกิดขึ้นทันทีที่มีการเขียนโค้ดเพื่อทำการส่งกลุ่มของข้อมูลไปยังอุปกรณ์ส่งสัญญาณเสียง ซึ่งประกอบด้วยฟังก์ชันดังต่อไปนี้

-waveOutBreakLoop หลุดลูปของอุปกรณ์ส่งสัญญาณเสียง

-waveOutPause หยุดการเล่นกลับชั่วคราวของอุปกรณ์ส่งสัญญาณเสียง

-waveOutRestart เริ่มการเล่นกลับต่อจากตำแหน่งที่หยุดชั่วคราวของอุปกรณ์ส่ง สัญญาณเสียง

-waveOutReset หยุดการเล่นกลับของอุปกรณ์ส่งสัญญาณเสียง

2.5.2.8 กลุ่มฟังก์ชันด้านการควบคุมการอัดสัญญาณเสียง

ฟังก์ชันในกลุ่มนี้ทำหน้าที่ควบคุมวิธีการอัดเสียงลงในไฟล์ .wav ซึ่งจะมีความแตกต่างจากฟังก์ชันด้านการอัดสัญญาณเสียง ซึ่งประกอบด้วยฟังก์ชันต่อไปนี้

-waveInStart เริ่มต้นการอัดเสียงของอุปกรณ์รับสัญญาณเสียง

-waveInStop หยุดการอัดเสียงชั่วคราวของอุปกรณ์รับสัญญาณเสียง

-waveInReset หยุดการอัดเสียงของอุปกรณ์ส่งสัญญาณเสียง

2.5.2.9 กลุ่มฟังก์ชันด้านการเปลี่ยนเสียงสูง-ต่ำและอัตราความเร็วด้านการเล่นกลับ

สามารถใช้ฟังก์ชันในกลุ่มนี้สำหรับการตบแต่งสัญญาณเสียงให้มีความแตกต่างจากข้อมูลต้นฉบับในไฟล์ .wav ได้ โดยการปรับระดับเสียงสูง-ต่ำ(Pitch) หรือการปรับอัตราความเร็วด้านการเล่นกลับซึ่งในแอพพลิเคชันด้านมัลติมีเดียชั้นนำก็ล้วนแต่มีคำสั่งสำหรับงานด้านนี้รวมอยู่ด้วย ซึ่งประกอบด้วยฟังก์ชันดังต่อไปนี้

-waveOutGetPitch อ่านค่าสเกลของระดับเสียงสำหรับอุปกรณ์ส่งสัญญาณเสียง

-waveOutGetPlaybackRate อ่านสเกลของระดับเสียงของการเล่นกลับสำหรับอุปกรณ์ส่งสัญญาณเสียง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

-waveOutSetPitch กำหนดค่าสเกลของระดับเสียงสำหรับอุปกรณ์ส่งสัญญาณเสียง

-waveOutSetPlaybackRate กำหนดค่าสเกลของระดับเสียงของการเล่นกลับสำหรับอุปกรณ์ส่งสัญญาณเสียง

2.5.2.10 กลุ่มฟังก์ชันด้านการเปลี่ยนความดังของเสียง

ฟังก์ชันในกลุ่มนี้จะทำหน้าที่อ่านและกำหนดระดับความดังของสัญญาณเสียง (volume) สำหรับการเล่นกลับเท่านั้น ซึ่งประกอบด้วยฟังก์ชันดังต่อไปนี้

2.5.3 การใช้ฟังก์ชันและข้อมูลโครงสร้าง

ในที่นี้จะขออธิบายเฉพาะฟังก์ชันในกลุ่มฟังก์ชันด้าน Querying เท่านั้น แต่ก่อนที่จะอธิบายถึงฟังก์ชันเหล่านี้ จะขออธิบายถึงข้อมูลโครงสร้างที่เกี่ยวข้องเสียก่อน ดังต่อไปนี้

2.5.3.1 ข้อมูลแบบโครงสร้าง WAVEINCAPS

เป็นโครงสร้างสำหรับฟังก์ชัน waveInGetDevCaps ซึ่งประกอบด้วยสมาชิกที่บอกถึงคุณสมบัติหรือความสามารถของอุปกรณ์รับสัญญาณเสียงเช่น รหัสประจำตัวของผู้ผลิต รูปแบบสัญญาณเสียงหรือจำนวนแชลแนล เป็นต้น โดยที่แต่ละสมาชิกมีรายละเอียดดังต่อไปนี้

```
TypeWAVEINCAPS
Wmid                As Integer              ' manufacturer ID
Wpid                As Integer              ' product ID
VdriverVersion      As Integer              ' version of the driver
SzPname             As String*MAXPNAMELEN   ' product name (NULL Terminated
                                              string
DwFormats           As Long                 ' formats supported
Wchannels           As Integer              ' number of channels supported

End Type
```

wMid และ wDeviceID

ข้อมูลชนิดเลขจำนวนเต็ม บอกถึงรหัสประจำตัวของผู้ผลิตและรหัสประจำตัวผลิตภัณฑ์ตามลำดับ ซึ่งโดยปกติผู้ผลิตการ์ดเสียงต่างๆ จะทำการอัพเดตรหัสของตนเองในซอฟท์แวร์ โดยการแจ้งการลงทะเบียนไปยังหน่วยงาน Multimedia System Group ของไมโครซอฟท์ สำหรับข้อมูลในตารางที่ 2.2 เป็นตัวอย่างของรหัสประจำตัวที่ได้รับการลงทะเบียนและอัพเดตในซอฟท์แวร์แล้ว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รหัสประจำตัวผู้ผลิต (Manufacturer ID)		
ค่าคงที่	ตัวเลข	รายละเอียด
MM_MICROSOFT	1	ไดรเวอร์ที่ถูกพัฒนาโดยบริษัทไมโครซอฟท์
รหัสประจำตัวผลิตภัณฑ์ (Product ID)		
ค่าคงที่	ตัวเลข	รายละเอียด
MM_MIDI_MAPPER	1	ไมโครซอฟท์ MIDI MAPPER
MM_SNDBLST_MIDIOUT	3	พอร์ตส่ง MIDI ของ Sound Blaster
MM_SNDBLST_MIDIN	4	พอร์ตรับ MIDI ของ Sound Blast
MM_SNDBLST_SYNTH	5	ซินธีไซเซอร์ภายในของ Sound Blaster
MM_SNDBLST_WAVEOUT	6	พอร์ตส่ง waveform ของ Sound Blaster
MM_SNDVLST_WAVEIN	7	พอร์ตรับ waveform ของ Sound Blaster
MM_ADLIB	9	ซินธีไซเซอร์ที่คอมแพททิเบิลของ Ad Lib
MM_MPU401_MIDIOUT	10	พอร์ตส่ง MIDI ของ MPU401
MM_MPU401_MIDIIN	11	พอร์ตรับ MIDI ของ MPU401
MM_PC_JOYSTICK	12	การ์ดควบคุมเกมของบริษัท IBM

ตารางที่ 2.2 ตัวอย่างของรหัสประจำตัวที่ได้รับการลงทะเบียนและอัพเดตในซอฟท์แวร์

vDriver Version

ข้อมูลชนิดเลขจำนวนเต็มกำหนดหมายเลขเวอร์ชันของดีไวซ์ไดร์เวอร์สำหรับอุปกรณ์รับสัญญาณเสียง ซึ่งมีการจัดเก็บในรูปแบบเลขฐานสิบหก 2 ไบต์ โดยที่ไบต์ลำดับสูงจะเป็นหมายเลขหลัก (major number) และไบต์ลำดับล่างจะเป็นหมายเลขรอง (minor number) เช่น ค่า 200H จะหมายถึงหมายเลขเวอร์ชัน 2.00 เป็นต้น

szPname

ข้อมูลชนิดสตริงมีความยาวเท่ากับ 32 ไบต์ (MAXPNAMELEN) สำหรับจัดเก็บรายละเอียดของผู้ผลิตหรือผลิตภัณฑ์ก็ได้ โดยที่สตริงที่ได้จะเป็น ASCIIZ ซึ่งหมายถึงเป็นสตริงที่สิ้นสุดด้วยตัวอักษร ASCII 0

dwFormats

ข้อมูลชนิดเลขจำนวนเต็ม long integer ที่บอกถึงรูปแบบของการจัดเก็บสัญญาณเสียงมาตรฐานที่อุปกรณ์รับ สัญญาณเสียงสนับสนุน ซึ่งโดยปกติอุปกรณ์หนึ่งๆ จะสนับสนุนมากกว่า 1 รูปแบบ โดยค่าที่จัดเก็บลงในสมาชิกนี้ จึงจัดเก็บในรูปแบบของค่าแฟลก โดยใช้ตัวปฏิบัติการ Or ดังนั้นในการตรวจสอบค่าของ dwFormats จึงต้องใช้ตัวปฏิบัติการ And ในการทดสอบแต่ละแฟลก ซึ่งรูปแบบมาตรฐานที่เป็นไปได้มี (ตารางที่ 2.3)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ค่าคงที่	ตัวเลข	รายละเอียด
WAVE_INVALIDFORMAT	&HO	รูปแบบไม่เป็นจริง (โดยส่วนใหญ่หมายถึงการเกิดข้อผิดพลาด
WAVE_FORMAT_1M08	&H1	รูปแบบ 11.025 kHz, Mono, 8-bit
WAVE_FORMAT_1S08	&H2	รูปแบบ 11.025 kHz, Stereo, 8-bit
WAVE_FORMAT_1M16	&H4	รูปแบบ 11.025 kHz, Mono, 16-bit
WAVE_FORMAT_1S16	&H8	รูปแบบ 11.025 kHz, Stereo, 16-bit
WAVE_FORMAT_2M08	&H10	รูปแบบ 22.05 kHz, Mono, 8-bit
WAVE_FORMAT_2S08	&H20	รูปแบบ 22.05 kHz, Stereo, 8-bit
WAVE_FORMAT_2M16	&H40	รูปแบบ 22.05 kHz, Mono, 16-bit
WAVE_FORMAT_2S16	&H80	รูปแบบ 22.05 kHz, Stereo, 16-bit
WAVE_FORMAT_4M08	&H100	รูปแบบ 44.1 kHz, Mono, 16-bit
WAVE_FORMAT_4S08	&H200	รูปแบบ 44.1 kHz, Stereo, 16-bit
WAVE_FORMAT_4M16	&H400	รูปแบบ 44.1 kHz, Mono , 16-bit
WAVE_FORMAT_4S16	&H800	รูปแบบ 44.1 kHz, Stereo , 16-bit

ตารางที่ 2.3 แสดงรายละเอียดของฟังก์ชันของไฟล์เวฟ

ข้อมูลชนิดเลขจำนวนเต็ม ที่บอกถึงจำนวนแชลแนลที่อุปกรณ์รับสัญญาณเสียงสนับสนุนซึ่งมีค่าตั้งแต่ 1 ถึง 2 เท่านั้น โดยที่ 1 หมายถึงโมโน และ 2 หมายถึง สเตอริโอ

2.5.3.2 ข้อมูลแบบโครงสร้าง WAVEOUTCAPS

เป็นโครงสร้างสำหรับฟังก์ชัน wavwOutGetDevCaps ซึ่งประกอบด้วยสมาชิกที่บอกถึงคุณภาพหรือความสามารถของอุปกรณ์ส่งสัญญาณเสียง เช่น รหัสประจำตัวของผู้ผลิต รูปแบบสัญญาณเสียง หรือจำนวนแชลแนลเป็นต้น โดยที่มีสมาชิกที่เหมือนกับสมาชิกของโครงสร้าง WAVEINCAPS แต่จะแตกต่างกันก็เฉพาะสมาชิก dwSupport เท่านั้นที่ไม่มีในโครงสร้าง WAVEINCAPS ดังนั้นผู้อ่านสามารถอ่านรายละเอียดของสมาชิกที่เหมือนกันได้จากโครงสร้าง WAVEINCAPS

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

WaveInGetNumDevs	
Declaration	Declare Functio waveInGetNumDevs Lib 'MMSYSTEM' () As Integer
Description	ทำหน้าที่รายงานจำนวนอุปกรณ์รับสัญญาณเสียง (waveform) ที่ติดตั้งในระบบ
Parameters	ไม่มี
Example	Dim wInNumDevs% WlnNumDevs% = waveInGetNumDevs ( ) Print "Number of Input Device : " ; wInNumDevs%
Comment	ค่าที่ส่งกลับโดยฟังก์ชันนี้จะถูกนำไปใช้ในการกำหนดอุปกรณ์รับสัญญาณเสียงสำหรับฟังก์ชัน waveInGetDevCaps ต่อไป

Type WAVEOUTCAPS		
wMid	As Integer	' manufacturer ID
wPid	As Integer	' product ID
vDriverVersion	As Integer	' version of the driver
szPname	As String * MAXPNAMELEN	' product name (NULL terminated string)
dwFormats	As Long	' format supported
wChannels	As Integer	' number of sources supported
dwSupport	As Long	' functionality supported by driver
Find Type		

dwSupport

ข้อมูลชนิดเลขจำนวนเต็ม long ที่บอกถึงฟังก์ชันที่แต่ละอุปกรณ์ส่งสัญญาณเสียงสนับสนุน (ตารางที่ 2.4)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ค่าคงที่	ตัวเลข	รายละเอียด
WAVECAPS_PITCH	&H1	สนับสนุนการควบคุมระดับโทนเสียง (pitch)
WAVECAPS_PLAYBACKRATE	&H2	สนับสนุนการควบคุมอัตราความเร็วในการเล่นกลับ
WAVECAPS_VOLUME	&H4	สนับสนุนการควบคุมระดับความดังของเสียง (volume)
WAVECAPS_LRVOLUME	&H8	สนับสนุนการแยกการควบคุมระดับความดังด้านซ้ายขวา
WAVECAPS_SYNC	&H10	แสดงถึงการ synchronous ของไดรเวอร์

WaveInGetDevCaps

Declaration	Declare Function waveInGetDevCaps Lib "MMSYSTEM" (ByVal udeviceid As_Integer, lPcaps As WAVECAPS, ByVal uSize As Integer) As Integer	
Description	ทำหน้าที่รายงานความสามารถและคุณสมบัติของแต่ละอุปกรณ์รับสัญญาณเสียงตามที่กำหนด	
Parameters	udeviceid	ตัวแปรชนิดเลขจำนวนเต็ม กำหนดหมายเลข ID ของอุปกรณ์รับสัญญาณเสียงที่ต้องการอ่านข้อมูลรายละเอียด โดยที่หมายเลข ID จะได้จากฟังก์ชัน waveInGetNumDevsและหมายเลข ID ที่กำหนดให้กับตัวแปรนี้จะสามารถมีค่าได้ตั้งแต่ 0 ถึง 1 เท่านั้น
	lpCaps	ตัวแปรชนิดข้อมูลโครงสร้าง WAVEINCAPS
	uSize	ตัวแปรชนิดเลขจำนวนเต็ม กำหนดของขนาดข้อมูลโครงสร้าง WAVEINCAPS ที่ส่งให้กับฟังก์ชันนี้ ดังนั้น ในการเขียนโค้ดจึงอ่านจำนวนไบต์ของข้อมูลโครงสร้างWAVEINCAPS โดยใช้ฟังก์ชัน Len

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Example

```
Dim WavIn As WAVEINCAPS
    Dim wInNumDevs%
    Dim wInDevsFlag%
    wInNumDevs% = waveInGetNumDevs( )
    Print "Waveform Audio Device Information : "
    Print "Number of Input Device : " ; wInNumDevs%
    wInDevsFlag% = waveInGetDevCaps ( (wInNumDevs%-1),
    WanIn, Len (WavIn) )

Print " Waveform Audio Input Device Information : "
    Print "ManuFacTurer ID : " ; WavIn.wMid
    Print "Product ID : ' ; WavIn.wPid
    Print "Driver Version : " ; sPhrase
    Version(WavIn.vDriverVersion)
    Print "Formats Supported : " ; sFormatDesc(WavIn.dwFormats)
    Print "Number of Channels : " ; WavIn.wChannels
```

Comment

ค่าที่ส่งกลับโดยฟังก์ชันนี้จะบอกถึงความสำเร็จของฟังก์ชันนี้

0 หมายถึง ฟังก์ชันทำงานสำเร็จ

<> 0 หมายถึง ฟังก์ชันทำงานล้มเหลว โดยที่รหัสข้อผิดพลาดที่ สามารถเป็นไปได้มีดังนี้

MMSYSERR_BADDEVICEID หมายถึง รหัส ID ของอุปกรณ์ไม่ถูก

MMSYSERR_NODRIVER หมายถึง ไม่มีการติดตั้งไดรเวอร์

uSize ตัวแปรชนิดเลขจำนวนเต็ม กำหนดขนาดของข้อมูลโครงสร้าง WAVEOUTCAPS ที่ส่งให้กับฟังก์ชันนี้ ดังนั้นในการเขียนโค้ดจึงอ่านจำนวนไบต์ของข้อมูลโครงสร้าง WAVEOUTCAPS โดยใช้ฟังก์ชัน Len

ตารางที่ 2.4 แสดงฟังก์ชันการควบคุมเสียง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Example	Dim WavOut As WAVEOUTCAPS Dim woutNumDevs% Dim woutDevsFlag% wOutNumDevs% = waveOutGetDevs( ) Print "Waveform Audio Device Internation : " Print "Number of Output Device : " wOutNumDevs% wOutDevsFlag% = waveOutGetNumDevCaps ((wOutNumDevs% - 1), WavOut, Len (Wavout)) Print "Waveform Audio Output Device Information Print "Manufacturer ID : " ; WavOut.wMid Print "Product ID : " ; WavOut.wpid Print "Driver Version : "; sPhraseVersion(WavOut.vDriverVersion) Print "Product Name : " ; sZtrim(WavOut.szPname) Print "Formats Supported : "sFormatDesc(WavOutdwFormat) Print "Number of Channels : ";WavOut.wChannels Print "Functionality Supported : " ; sFuncSupport (WavOut.dwSupport)
Comment	ค่าที่ส่งกลับโดยฟังก์ชันนี้จะบอกถึงความสำเร็จของฟังก์ชัน ดังนี้ 0 หมายถึง ฟังก์ชันทำงานสำเร็จ <> 0 หมายถึง ฟังก์ชันทำงานล้มเหลว โดยที่รหัสข้อผิดพลาดที่สามารถเป็นไปได้มีดังนี้ MMSYSERR_BADDEVICEID หมายถึง รหัส ID ของอุปกรณ์ไม่ถูกต้อง MMSYSERR_NODRIVER หมายถึง ไม่มีการติดตั้งไดรเวอร์

2.6 โพรโตคอลทีซีพี/ไอพี (TCP/IP Protocol)

ในหัวข้อนี้จะกล่าวถึงลักษณะการทำงานของอินเตอร์เน็ต ซึ่งประกอบไปด้วยเรื่องเกี่ยวกับ TCP/IP, การกำหนดชื่อ และเลข IP อินเตอร์เน็ตนับได้ว่าเป็นเครือข่ายบนเครือข่าย ที่เปิดโอกาสให้เครือข่ายคอมพิวเตอร์อื่นๆ เชื่อมโยงเข้ามาใช้งาน หรือเป็นศูนย์กลางเชื่อมโยงเครือข่ายคอมพิวเตอร์อื่นๆ อีก แต่ปัญหาที่เกิดขึ้นในการเชื่อมโยงเครือข่ายคอมพิวเตอร์เข้าด้วยกันคือ แต่ละเครือข่ายใช้คอมพิวเตอร์ต่างชนิด ต่างยี่ห้อ และต่างระบบปฏิบัติการ มาตรฐานของ TCP/IP จึงถูกใช้เป็นกุญแจสำคัญในการแก้ปัญหาเหล่านี้ โดยจะกลายเป็นระบบเปิดที่สมบูรณ์แบบที่มีการเชื่อมโยงคอมพิวเตอร์ได้ตั้งแต่พีซีจนถึงเมนเฟรม และไม่จำกัดระบบปฏิบัติการที่ใช้ TCP/IP จึงเป็นมาตรฐานที่ทั่วโลกยอมรับ มีอุปกรณ์และซอฟต์แวร์ผลิตออกมาสนับสนุน TCP/IP มากมาย ดังนั้นจึงนับได้ว่า TCP/IP เป็นหัวใจของอินเตอร์เน็ตเลยทีเดียว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

TCP/IP คืออะไร

TCP/IP เป็นข้อกำหนดเกี่ยวกับรูปแบบการเชื่อมโยงในเครือข่าย (Networking Protocol) จัดทำขึ้นเพื่อใช้เป็นกฎเกณฑ์ให้เครื่องคอมพิวเตอร์ใช้งานร่วมกัน ในลักษณะของระบบเปิด (Open System) คือไม่ว่าจะเป็นคอมพิวเตอร์ชนิดใดหรือระบบใดก็ตาม จะสามารถติดต่อสื่อสารและแลกเปลี่ยนข้อมูลกันได้เช่น เครื่องคอมพิวเตอร์ของ Digital Equipment ซึ่งเป็นระบบ Mini ติดต่อสื่อสารกับ Compaq ซึ่งเป็นเครื่อง PC ได้ เมื่อดำเนินการด้วย TCP/IP

TCP/IP เป็นการกำหนดรูปแบบการสื่อสารระหว่างซอฟต์แวร์ การจัดการโอนย้ายข้อมูล การแสดงสถานะของเครื่องคอมพิวเตอร์ที่อยู่บนเครือข่าย ตลอดจนกฎระเบียบต่าง ๆ ที่กำหนดให้ทำ เมื่อเกิดความผิดพลาดหรือต้องทำ เพื่อห้องกันไม่ให้เกิดความผิดพลาด

TCP/IP เกิดจากการนำข้อกำหนดของรูปแบบต่าง ๆ กันมาใช้ร่วมกัน TCP และ IP ต่างก็เป็นรูปแบบหนึ่งของชุดข้อกำหนดนี้ (แต่เรียกชุดข้อกำหนดรูปแบบนี้ว่า TCP/IP) ถูกออกแบบมาเพื่อใช้รับส่งหรือโอนย้ายข้อมูลระหว่างคอมพิวเตอร์ที่อยู่บนระบบเครือข่ายเดียวกัน หรือต่างเครือข่ายกันก็ได้ และมีการจัดเตรียมข้อมูลสถานะของเครือข่ายขึ้นได้ภายในตังข้อกำหนดรูปแบบเอง ในการสร้างซอฟต์แวร์ของระบบเครือข่ายจะใช้ TCP/IP เป็นส่วนสนับสนุนได้ทั้งระบบเครือข่ายเฉพาะบริเวณ (Local Area Network) และเครือข่ายบริเวณกว้าง (Wide Area Network) ไม่ได้ใช้งานเฉพาะกับอินเตอร์เน็ตเท่านั้น

ส่วนประกอบของ TCP/IP

จากที่กล่าวมาแล้งข้างต้นว่า TCP/IP ประกอบไปด้วยชุดข้อกำหนดรูปแบบต่าง ๆ ซึ่งแบ่งเป็นกลุ่มได้ดังนี้

กลุ่มข้อกำหนดรูปแบบการขนส่ง (Transport Protocols)

ทำหน้าที่ควบคุมการเคลื่อนย้ายข้อมูลระหว่างคอมพิวเตอร์สองเครื่อง แบ่งย่อยออกได้เป็นสองชนิดคือ

1. TCP (Transmission Control Protocols) เป็นการบริการแบบ Connection Based Service ซึ่งคอมพิวเตอร์ด้านผู้รับและผู้ส่งต้องต่อถึงกันอยู่ตลอดเวลาในระหว่างการสื่อสาร ถ้าเปรียบเทียบก็คล้ายกับระบบโทรศัพท์ที่ต้องติดต่อกันให้ได้ก่อนจะพูดคุยกันได้

2. UDP (User Datagram Protocol) เป็นการให้บริการแบบ Connections Service คอมพิวเตอร์ด้านผู้ส่งไม่จำเป็นต้องติดต่อกับด้านผู้รับก่อน เพียงรู้ที่อยู่ของด้านผู้รับแล้วใส่ที่อยู่นั้นไปกับข้อมูลที่ส่งออก ข้อมูลจะเดินทางตามเส้นทางต่าง ๆ เพื่อไปถึงปลายทางตามที่อยู่ คล้ายกับการส่งจดหมายที่ไปรษณีย์ จะส่งให้ตามที่อยู่ที่จ่าหน้าซองจดหมาย โดยผู้ส่งและผู้รับไม่ต้องติดต่อกัน

กลุ่มข้อกำหนดเกี่ยวกับรูปแบบเส้นทาง (Routing Protocol)

ทำหน้าที่พิจารณาเส้นทางที่ดีที่สุดที่ใช้ส่งข้อมูลและถ้ามีข้อมูลเป็นจำนวนมากหรือมีขนาดใหญ่ กลุ่มข้อกำหนดรูปแบบนี้ก็จะทำการแบ่งย่อยข้อมูลให้มีขนาดเหมาะสม แล้วส่งออกไป เมื่อถึงผู้รับ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ปลายทาง กลุ่มข้อกำหนดนี้ก็จะทำหน้าที่ตรงข้าม คือรวบรวมข้อมูลย่อยให้ถูกต้องก่อนการแสดงผลกลุ่มข้อกำหนดรูปแบบกลุ่มนี้ประกอบด้วย

1. IP (Internet Protocol) เป็นข้อกำหนดรูปแบบการส่งข้อมูล

2. ICMP (Internet Control Message Protocol) เป็นข้อกำหนดรูปแบบของข้อมูลข่าวสารเกี่ยวกับสถานะของ IP เช่น ข่าวสารความผิดพลาดและผลกระทบต่อเส้นทางเมื่อมีการเปลี่ยนแปลงฮาร์ดแวร์ในเครือข่าย

3. RIP (Routing Information Protocol) ข้อกำหนดรูปแบบหนึ่งที่ใช้สำหรับทำการพิจารณาวิธีการเลือกเส้นทางเพื่อให้ได้เส้นทางที่ดีที่สุดเหมาะสมกับข้อมูลมากที่สุด

4. OSPF (Open Shortest Path First) ข้อกำหนดรูปแบบอีกประเภทหนึ่งที่ใช้ตัดสินเลือกเส้นทางโดยพิจารณาจากเส้นทางที่สั้นที่สุดก่อน

กลุ่มข้อกำหนดรูปแบบเกี่ยวกับที่อยู่เครือข่าย (Network Address)

ทำหน้าที่พิจารณาที่อยู่ของเครือข่ายและเครื่องคอมพิวเตอร์ไม่ว่าจะเป็นลักษณะตัวเลขหรือชื่อก็ตาม เพื่อความถูกต้องของข้อมูลที่จะไปยังผู้รับปลายทาง โดยที่ไม่ว่าเครือข่ายจะใหญ่โตสักเพียงใดหรือมีเครื่องคอมพิวเตอร์จำนวนมากก็ตาม ที่อยู่จะต้องไม่ซ้ำกัน กลุ่มข้อกำหนดรูปแบบกลุ่มนี้มีดังนี้

1. ARP (Address Resolution Protocol) ข้อกำหนดรูปแบบที่พิจารณาตัวเลขที่อยู่เพื่อไม่ให้เกิดที่อยู่ซ้ำกัน

2. DNS (Domain Name System) ข้อกำหนดรูปแบบที่พิจารณาตัวเลขที่อยู่เมื่อรู้ชื่อของเครือข่ายหรือเครื่องคอมพิวเตอร์ เพราะในการใช้งานจริงนั้นใช้เพียงที่อยู่ที่เป็นตัวเลข แต่ระบบชื่อจัดทำขึ้นเพื่อให้สะดวกต่อการใช้งานของผู้ใช้

3. RARP (Reverse Address Resolution Protocol) ข้อกำหนดรูปแบบที่พิจารณาตัวเลขที่อยู่เช่นเดียวกับ ARP แต่จะทำตรงกันข้ามกับ ARP

2.6.1 กลุ่มข้อกำหนดรูปแบบเกี่ยวกับเส้นทางการสื่อสารระหว่างเครือข่าย (Gateway Protocols)

สนับสนุนข้อมูลสถานะเพื่อนำไปใช้เลือกเส้นทางที่เหมาะสม ข้อกำหนดรูปแบบเหล่านี้ประกอบด้วย

1. EGP (Exterior Gateway Protocol) ข้อกำหนดรูปแบบนี้จะทำการถ่ายโอน ข้อมูลเส้นทางกันระหว่าง Gateway กับเครือข่ายภายนอกเพื่อทำการสื่อสาร

2.GGP (Gateway-to-Gateway Protocol) เป็นข้อกำหนดรูปแบบที่ทำงานถ่ายโอนข้อมูลเส้นทางกันระหว่าง Gateway กับ Gateway

3.IGP (Interior Gateway Protocol) ข้อกำหนดรูปแบบที่ถ่ายโอนข้อมูลเส้นทางกันภายในเครือข่ายเดียวกัน

Gateway เป็นอุปกรณ์หรือเครื่องคอมพิวเตอร์ที่ทำหน้าที่เสมือนประตูสื่อสาร เป็นช่องสัญญาณเข้าหรือออกไปยังระบบสื่อสารอื่น หรือภายในเครือข่ายเดียวกันติดต่อกัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.6.2 กลุ่มข้อกำหนดรูปแบบเกี่ยวกับการบริการผู้ใช้ (User Services)

ผู้ใช้สามารถใช้ข้อกำหนดรูปแบบได้โดยตรง ข้อกำหนดรูปแบบนี้ประกอบด้วย

1. B00TP (BOOT Protocol) เมื่อผู้ใช้เปิดเครื่องคอมพิวเตอร์บนเครือข่ายให้เริ่มทำงาน ข้อกำหนดรูปแบบนี้จะอ่านโปรแกรมควบคุมการทำงานจากคอมพิวเตอร์ให้บริการ (Server Computer) มาให้

2. FTP (File Transfer Protocol) ข้อกำหนดรูปแบบที่ให้บริการถ่ายโอน ไฟล์ข้อมูลระหว่างเครื่องคอมพิวเตอร์ ซึ่งอาจจะอยู่บนเครือข่ายเดียวกันหรือต่างเครือข่ายกันก็ได้

3. TELNET เป็นข้อกำหนดรูปแบบที่ให้บริการเกี่ยวกับการควบคุมการติดต่อระยะทางไกล (Remote Login)

กลุ่มข้อกำหนดรูปแบบอื่นที่นอกเหนือจากกลุ่มที่จัดไว้และบริการที่สำคัญ ๆ จัดทำไว้ให้บนระบบเครือข่ายที่สนใจมีดังนี้

- NFS (Network File System) เป็นข้อกำหนดรูปแบบที่ทำให้ผู้ใช้คอมพิวเตอร์เครื่องหนึ่งสามารถเข้าไปดูไฟล์ข้อมูลและใช้งานไฟล์ข้อมูล ซึ่งอยู่ในคอมพิวเตอร์เครื่องอื่นได้
- NIS (Network Information Services) เป็นข้อกำหนดรูปแบบที่ให้บริการกับ User Accounts ข้ามเครือข่าย เช่น Logins และ Password
- RPC (Remote Procedure Call) ข้อกำหนดรูปแบบที่อำนวยความสะดวกให้กับโปรแกรมประยุกต์ที่ใช้งานกับการควบคุมระยะทางไกล
- SMTP (Simple Mail Transfer Protocol) ข้อกำหนดรูปแบบที่ให้บริการถ่ายโอนจดหมายอิเล็กทรอนิกส์ (Electronic Mail) ระหว่างคอมพิวเตอร์
- SNMP (Simple Network Management Protocol) ข้อกำหนดรูปแบบที่ให้บริการข่าวสารต่าง ๆ ที่แสดงสถานะของเครือข่ายและอุปกรณ์ที่ต่ออยู่บนเครือข่าย

2.6.3 สถาปัตยกรรม TCP/IP

TCP/IP ออกแบบเป็นชุดกำหนดรูปแบบ แบ่งการบริหารออกเป็นกลุ่มๆ ภายในชุดข้อกำหนดรูปแบบเช่น กลุ่มบริการผู้ใช้ กลุ่มการขนส่ง และกลุ่มเกี่ยวกับเครือข่าย ดังกล่าวมาแล้ว จากกลุ่มต่างๆ ที่แบ่งไว้นำไปพัฒนาเป็นสถาปัตยกรรมซึ่งมีลักษณะเป็นระดับชั้น (Layer) ตามกลุ่มการบริการ

สถาปัตยกรรมแบบระดับชั้น มีข้อดีอยู่หลายประการ แต่ละชั้นมีอิสระไม่ขึ้นต่อกัน ทำให้การเปลี่ยนแปลงการบริการของชั้นใดๆ ไม่ก่อปัญหากับบริการชั้นอื่น การเพิ่มเติมการบริการใหม่ทำได้โดยไม่ต้องเปลี่ยนแปลงโปรแกรมระบบเดิมและสิ่งสำคัญก็คือการทำระดับชั้นนั้นทำให้ตัวโปรแกรมมีขนาดเล็ก สามารถระบุส่วนที่จะต้องปรับปรุงได้แน่นอน ไม่ต้องวิตกกังวลถึงโปรแกรมส่วนอื่นทำให้การพัฒนาประสิทธิภาพของระบบทำได้ง่ายและดียิ่งขึ้น (ตารางที่ 2.5)

ชุดข้อกำหนดรูปแบบ TCP/IP เมื่อนำไปใช้งานจะอยู่ระหว่างระดับชั้นที่เป็นฮาร์ดแวร์กับระดับชั้นที่เป็นซอฟต์อื่น เมื่อติดตั้งกับบางระบบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Application	
TCP/IP	User Interface (Windows,X,ete.)
Operating System	
Hardware	

ตารางที่ 2.5 แสดงสถาปัตยกรรมของ TCP/IP

TCP/P เมื่อนำไปใช้งานกับระบบ Windows ต้องเพิ่ม Drivers พิเศษเข้าไปด้วยเพื่อให้ Windows รู้จักข่าวสารที่เป็นแพ็กเกจ (Packet) และส่งแพ็กเกจเข้าสู่ระบบเครือข่ายได้ แต่ถ้าเป็นซอฟต์แวร์ที่ใช้ชุดข้อกำหนดรูปแบบ TCP/IP อยู่แล้วข่าวสารจากโปรแกรมประยุกต์จะผ่านมายังระดับชั้น TCP/IP ทำการประมวลส่งเข้าเครือข่ายโดยให้ระบบปฏิบัติการทำการส่ง

การกำหนดชื่อและเลข IP

เครื่องคอมพิวเตอร์ที่ต่อเชื่อมกับอินเตอร์เน็ตเครือข่ายที่ใช้ TCP/IP มีอยู่เป็นจำนวนมาก เครื่องคอมพิวเตอร์แต่ละเครื่องต้องสามารถระบุ หรืออ้างอิงได้โดยไม่เกิดความซ้ำซ้อนกับเครื่องคอมพิวเตอร์อื่นๆ มิฉะนั้นแล้วข่าวสารที่เครือข่ายรับมาจะไม่สามารถส่งไปให้กับเครื่องคอมพิวเตอร์ที่ต้องการข่าวสารนั้นได้ จึงต้องมีการจัด ระบบที่ดี เครือข่ายอินเตอร์เน็ตหรือเครือข่ายที่ใช้ TCP/IP ได้ออกแบบการจัดการระบบตรงส่วนนี้ไว้แล้ว

รหัสเลขประจำเครื่อง IP

เครือข่ายอินเตอร์เน็ตใช้รหัสหมายเลขมากำหนดให้แต่ละเครือข่ายและเครื่องคอมพิวเตอร์ภายในเครือข่ายที่เชื่อมโยง เรียกรหัสหมายเลขเหล่านี้ว่า **อินเตอร์เน็ตแอดเดรส (Internet Address)** หรือนิยมเรียกกันทั่วไปว่า **IP แอดเดรส**

IP แอดเดรส ประกอบด้วยเลขฐานสองจำนวน 32 บิต แบ่งออกเป็น 4 ส่วน แต่ละส่วนมี 8 บิต เมื่อดูเฉพาะแต่ละส่วนเป็นฐานสิบจะได้เลขจำนวน 256 ค่าไม่ซ้ำกัน (0-256) IP แอดเดรส จะนำเอาหมายเลขทั้ง 4 ส่วนมารวมกัน โดยแยกแต่ละส่วนด้วยจุด ดังนั้นหมายเลขทั้งหมดที่เป็นไปได้ โดยค่าไม่ซ้ำกัน คือ 256^4 หรือ 4294967296 จำนวนมีค่าหมายเลขจาก 000.000.000.000 จนถึง 255.255.255.255 หมายเลขเหล่านี้เองที่อินเตอร์เน็ต ใช้กำหนดให้กับเครือข่าย และ เครื่องคอมพิวเตอร์เพื่อใช้อ้างอิงถึง

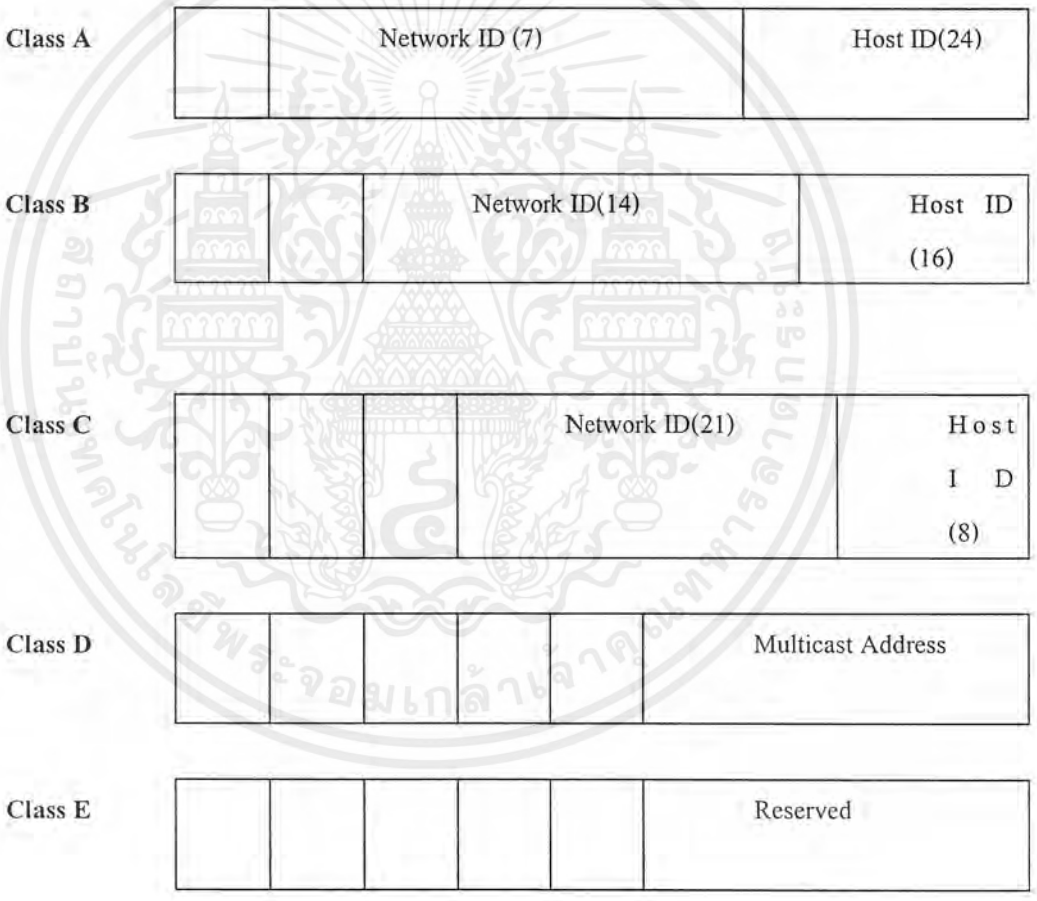
IP แอดเดรส บางหมายเลขสงวนไว้ใช้ด้วยจุดหมายกรณีพิเศษ ทำให้ IP แอดเดรส ที่ใช้งานทั่วไป ลดลง จากจำนวนที่เป็นไปได้ ความหมายของ IP แอดเดรส จะแบ่งได้เป็น 2 กลุ่มดังนี้

- กลุ่มที่ใช้เป็นรหัสประจำเครือข่าย
- กลุ่มที่ใช้เป็นรหัสประจำเครื่องคอมพิวเตอร์ที่อยู่ภายในเครือข่าย (Host Computers)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

IP แอดเดรส ในกลุ่มรหัสประจำเครื่องคอมพิวเตอร์สามารถซ้ำกันได้ แต่กลุ่มรหัสประจำเครือข่ายจะซ้ำกัน ไม่ได้ ดังนั้นรหัสเครื่องที่ซ้ำกันจึงไม่มีผลต่อการอ้างอิงถึง

นอกจากนี้เพื่อความเหมาะสมในการกำหนด IP แอดเดรสให้กับผู้ขอ ทางผู้บริหารอินเตอร์เน็ตแบ่งคลาสของผู้ขอ IP แอดเดรสตามขนาดของเครือข่าย เพื่อให้ทรัพยากรส่วนนี้ถูกใช้อย่างคุ่มค่าที่สุด องค์กรขนาดใหญ่ ก็จะจัดให้อยู่ในคลาสที่สามารถกำหนด IP แอดเดรสให้กับเครื่องคอมพิวเตอร์ในเครือข่ายได้มาก การแบ่งคลาสจะแบ่งได้ (รูปที่ 2.10)



รูปที่ 2.10 แสดงคลาสของไอพี

การกำหนดหมายเลขของเครือข่ายและเครื่องคอมพิวเตอร์ นอกจากจะแบ่ง IP แอดเดรสเป็นคลาสทั้ง 5 ประเภทแล้ว ยังมีข้อกำหนดปลีกย่อยอีกหลายประการที่ผู้วางระบบต้องรู้ไว้ เพื่อที่จะกำหนดแอดเดรสใช้งานได้อย่างถูกต้องไม่ผิดพลาด ได้แก่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

IP แอดเดรสที่เป็น "0" ทุกบิต จะไม่ใช้งานทั่วไป แต่นำไปใช้กับอุปกรณ์ หาเส้นทาง (Router) เพื่อกำหนด "Default Route"

IP แอดเดรสในแต่ละส่วนคือ ส่วนหมายเลขเครือข่ายและส่วนหมายเลขประจำเครื่องคอมพิวเตอร์จะเป็น "0" หรือ "1" ทุกบิตไม่ได้

1. ถ้าส่วนของหมายเลขประจำเครื่องคอมพิวเตอร์เป็น "0" ทุกบิตหมายถึง หมายเลขเครือข่ายนั้น ใช้งานร่วมกับอุปกรณ์หาเส้นทางหรือเพื่อบอกให้เครื่องคอมพิวเตอร์รู้ว่าตัวเองอยู่ในเครือข่ายใด

2. ส่วนของหมายเลขประจำเครื่องคอมพิวเตอร์เป็น "1" ทุกบิต หมายถึง หมายเลขเครือข่ายนั้นใช้สำหรับกระจายข่าวภายในเครือข่าย (Broadcast Address)

3. ถ้า IP แอดเดรสทั้งสองส่วนเป็น "1" ทุกบิต หมายถึงกระจายข่าวหรืออีกนัยหนึ่งเครื่องคอมพิวเตอร์ที่ส่ง แอดเดรสนี้เข้าสู่ระบบ ไม่ทราบว่าตนเองอยู่ในเครือข่ายใด

IP แอดเดรสของคลาส A หมายเลข 127.0.0.0 จะสงวนไว้ใช้งานเฉพาะอย่าง เช่น IPC (Inter-Process Communication)

จากรูป 2.10 จะแสดง ประเภทของ IP แอดเดรสและข้อกำหนดปลีกย่อยต่างๆ เราสามารถทราบได้ว่าเครือข่ายขององค์กรถูกจัดอยู่ใน Class ใด โดยดูจากค่าหมายเลขของ 8bit แรกซ้ายสุด ดังนี้

Class A : IP แอดเดรสอยู่ในช่วง 1 ถึง 126

Class B : IP แอดเดรสอยู่ในช่วง 128 ถึง 191

Class C : IP แอดเดรสอยู่ในช่วง 192 ถึง 223

ตัวอย่าง เช่น IP แอดเดรส เป็น 147.14.87.23 หมายถึง เครือข่ายนี้อยู่ใน Class B มีหมายเลขประจำเครือข่ายเป็น 147.14 และมีหมายเลขเครื่องคอมพิวเตอร์บนเครือข่ายเป็น 87.23 ซึ่งเครื่องคอมพิวเตอร์บนเครือข่าย 147.14 จะมีหมายเลขเครื่องคอมพิวเตอร์ 87.23 ได้เพียงเครื่องเดียวเท่านั้น

2.7 ประวัติของอินเตอร์เน็ต

จุดกำเนิดของอินเตอร์เน็ตมาจากความคิดเชิงยุทธศาสตร์ทางการทหาร ในช่วงปี พ.ศ. 2512 เป็นช่วงสงครามเย็นระหว่างสหรัฐอเมริกากับรัสเซีย ทำให้เกิดความวิตกว่าศูนย์คอมพิวเตอร์ที่ใช้ควบคุมสั่งการและควบคุมด้านการส่งกำลังบำรุงถูกทำลายแล้วศูนย์คอมพิวเตอร์อื่นจะไม่สามารถทำงานทดแทนได้ เนื่องมาจากระบบคอมพิวเตอร์ที่ใช้อยู่มีหลายรูปแบบหลายยี่ห้อจากผู้ผลิตเครื่องคอมพิวเตอร์ เช่น IBM, VAX, Apple อีกทั้งระบบปฏิบัติการ (Operation System) ที่ใช้ก็ต่างกัน เช่น UNIX, VM, MVS เป็นต้น ดังนั้น จึงเกิดความคิดที่จะเชื่อมโยงคอมพิวเตอร์ต่างๆเหล่านี้เข้าด้วยกัน ให้สามารถทำงานได้ตลอดเวลาแม้ว่าระบบคอมพิวเตอร์บางส่วนจะถูกทำลายไป แต่ส่วนที่เหลือต้องทำงานต่อได้

จากแนวความคิดดังกล่าวทำให้มีการจัดตั้งหน่วยงานร่วมระหว่างสถาบันการศึกษากับฝ่ายวิจัยพัฒนาทางทหารขึ้นมา เพื่อพัฒนาระบบสื่อสาร และเครือข่ายคอมพิวเตอร์ภายใต้โครงการ "US Advance Research Projects Agency" เราเรียกระบบเครือข่ายนี้ว่า ARPANET (Advanced Research

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Projects Agency Network) โดยใช้โพรโตคอล (Protocol) NCP (Network Control Protocol) แต่ต่อมาพบว่าโพรโตคอล NCP นี้มีข้อจำกัดอยู่มาก จึงมีการพัฒนาโพรโตคอลตัวใหม่ขึ้นมา ซึ่งมีชื่อเรียกว่า TCP/IP (Transmission Control Protocol/Internet Protocol) จนกระทั่งเป็นรูปแบบมาตรฐานสำหรับ ARPANET แทน NCP

การพัฒนาเครือข่ายได้ดำเนินต่อมาเรื่อยๆ จนกระทั่งกระทรวงกลาโหมของสหรัฐอเมริกา ซึ่งเป็นผู้ริเริ่มได้ยกเลิกการสนับสนุนโดยให้ทางกลุ่มสถาบันการศึกษาดำเนินการเอง จากจุดนี้เองทำให้อินเตอร์เน็ตมีความหลากหลาย และขยายตัวจนเรียกว่า เป็นเครือข่ายคอมพิวเตอร์ที่เสมือนใยแมงมุมครอบคลุมโลกเอาไว้

2.8 วินโดวส์ ซ้อกเก็ต (Windows Sockets)

กุญแจที่จะนำไปสู่การเขียนโปรแกรมบนอินเตอร์เน็ต (Internet Programming) โดยใช้วินโดว์คือ "ซ้อกเก็ต" (Sockets) โดยทั่วไปแล้วซ้อกเก็ต หมายถึงเซต เซตหนึ่ง (a set) ของฟังก์ชันที่ใช้ทำการติดต่อแบบสองทาง (Two-ay Communication) โดยใช้โปรโตคอล TCP/IP หรือโปรโตคอลอื่นๆ

ส่วนถ้าหากเราพิจารณาคำว่าวินโดวส์ ซ้อกเก็ต (Windows Sockets) แล้วจะหมายความถึง API มาตรฐานที่ถูกพัฒนาโดยกลุ่มบริษัทกว่า 20 บริษัทและถูกใช้เป็นมาตรฐานการเปิดเครือข่ายเพื่อจัดการการติดต่อโดยใช้โปรโตคอล TCP/IP (ถึงแม้ว่าจะม่การเผื่อไว้สำหรับใช้กับโปรโตคอลอื่นภายหลังก็ตาม) ถ้าหากเราต้องการเขียนโปรแกรมบนอินเตอร์เน็ตสำหรับวินโดวส์แล้ว เราต้องรู้จักซ้อกเก็ตและวินโดว์ซ้อกเก็ต

2.8.1 ซ้อกเก็ตคืออะไร

ซ้อกเก็ตมาจากระบบปฏิบัติการยูนิกซ์ (UNIX) และโดยเฉพาะอย่างยิ่งมาจากเบิร์กเลย์ยูนิกส์ (Berkley UNIX: BSD) ช่วงประมาณปีคริสตศักราช 1980 ซ้อกเก็ตหนึ่งซ้อกเก็ตคือชิ้นส่วนหนึ่ง (a piece) ของซอฟแวร์ (Software) ซึ่งสามารถรับและส่งข้อมูลบนเครือข่าย TCP/IP ได้ ส่วนของข่าวสารข้อมูล (Information) เกี่ยวกับซ้อกเก็ตที่สำคัญมีอยู่ 3 ส่วนคือ

1. IP Addess ซ้อกเก็ตกำลังติดต่ออยู่
2. พอร์ต ที่ซ้อกเก็ตใช้ติดต่อกับแอดเดรสนั้น
3. ชนิดของซ้อกเก็ต

สิ่งหนึ่งที่เกี่ยวกับพอร์ตที่จำเป็นต้องทำความเข้าใจให้ถูกต้องนั่นคือ พอร์ตในที่นี้ไม่เหมือนพอร์ตขนานหรือพอร์ตอนุกรม พอร์ตเหล่านี้ไม่ได้เป็นฮาร์ดแวร์ (Hardware) ใดๆ ทั้งสิ้นพอร์ตในที่นี้เป็นเพียงสภาวะ (convention) ที่ถูกใช้โดยโปรแกรมที่รันอยู่บนเครื่องที่กำลังทำการติดต่อสื่อสาร โปรโตคอล TCP/IP จะให้เครื่องหนึ่งสามารถติดต่อกับอีกหลายๆ เครื่องในขณะเดียวกันได้ ยกตัวอย่างเช่น เราสามารถเปิดเวบบราวเซอร์ให้โหลดเพจขนาดใหญ่ๆ ขณะเดียวกันก็ให้นิวส์รีดเดอร์จัดเรียงและอินเดกซิ่งบทความขนาดใหญ่ และโปรแกรม FTP ก็กำลังดาวน์โหลดไฟล์ขนาดใหญ่ไฟล์หนึ่ง และทันใดนั้นก็มีอีเมล์ส่งมาถึงพอดี เราก็เลยหันไปอ่านอีเมล์โดยปล่อยให้มีการโหลด,อินเดกซิ่งและอื่นๆ ต่อไป สภาวะต่างๆ ของการติดต่อบนอินเทอร์เนตระหว่าง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เครื่องของเรากับเครื่องอื่น สภาวะเหล่านี้จะเกิดขึ้นบนพอร์ตที่แตกต่างกัน บริการมาตรฐานบนอินเตอร์เน็ตเช่น เวบนิวส์ FTP และเมล์ เหล่านี้จะมีพอร์ตมาตรฐานของแต่ละบริการ

ซ้อกเก็ตอยู่ 2 ชนิดได้แก่ สตรีมซ้อกเก็ต (stream socket) และดาต้าแกรมซ้อกเก็ต (datagram socket) ซึ่งทั้งสองชนิดใช้กับโปรโตคอลหลักสองประเภทของ TCP/IP คือ TCP และ UDP โปรโตคอล TCP มีกระบวนการคล้ายกับการที่เครื่องสองเครื่องสร้างการติดต่อหนึ่งช่องการติดต่อ (form a connection) จากนั้นข้อมูลจะถูกแลกเปลี่ยนผ่านช่องการติดต่อนี้ เมื่อเสร็จก็ทำการปิดช่องการติดต่อนั้น ในทางตรงกันข้าม โปรโตคอล UDP จะไม่เกี่ยวข้องกับการสร้างการติดต่อเครื่องหนึ่งส่งข้อมูลให้เครื่องอื่นโดยไม่สร้างช่องทางการติดต่อ สตรีมซ้อกเก็ตใช้สร้างโปรโตคอล TCP และถูกใช้กับงานที่จะต้องแลกเปลี่ยนข้อมูลจำนวนมากๆ หรืองานที่เน้นว่าลำดับก่อนหลังของข้อมูลที่เข้ามามีความสำคัญ ตัวอย่างของการใช้งาน TCP ที่เก่าแก่ซึ่งก็ใช้สตรีมซ้อกเก็ตนั่นคือ โปรโตคอล FTP แต่ดาต้าแกรมซ้อกเก็ตใช้สร้างโปรโตคอล UDP และถูกใช้กับงานที่มีการแลกเปลี่ยนข้อมูลน้อยๆ หรืองานที่ไม่สนใจลำดับก่อนหลังของข้อมูล ตัวอย่างของการใช้งาน UDP ที่เก่าแก่เห็นจะเป็นคล็อกอัพ เดทเตอร์ (clock-updater) ซึ่งจะบรอดคาสท์เวลาของระบบไปสู่เครื่องอื่นๆ

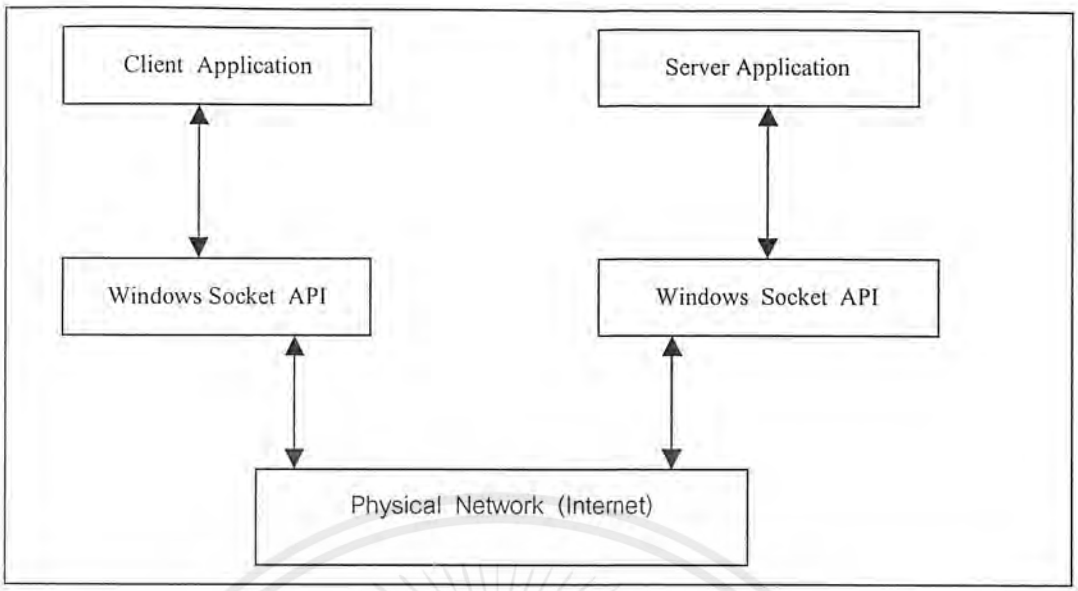
2.8.2 การได้รับ API ของวินโดวส์ ซ้อกเก็ต

เมื่อมีคนพูดถึง API ของวินโดวส์ซ้อกเก็ต คนๆ นั้นจะหมายถึงข้อตกลงที่เป็นที่ยอมรับของทุกๆ บริษัท ซึ่งก็คือชื่อของฟังก์ชันต่างๆ และหน้าที่การทำงานของฟังก์ชันเหล่านั้น ยกตัวอย่างเช่นข้อตกลงระบุว่า มีฟังก์ชันหนึ่งชื่อ connect() ซึ่งทำหน้าที่ "สร้างช่องการติดต่อระหว่างซ้อกเก็ตที่กำหนด"

อย่างไรก็ตามเราไม่สามารถที่จะเอ็กซิคิวต์ "ข้อตกลง" ได้ เราต้องมีซอสโค้ด หรือโค้ดที่คอมไพล์แล้ว หลายบริษัทจึงจัดหาไดนามิกไลบรารี่เรียกว่า WINSOCK.DLL ซึ่งทำให้เราสามารถเรียกฟังก์ชันได้ทุกฟังก์ชันที่ระบุไว้ในข้อตกลงดังกล่าว ตัวอย่างเช่นชุดของโปรแกรมซึ่งโดยทั่วไปเรียกว่า ทรัมเป็ด (Trumpet) ของปีเตอร์ แทตแทม (Peter Tattam) จะมีไฟล์ WINSOCK.DLL ซึ่งจะมีฟังก์ชั่นเหล่านี้และฟังก์ชันอื่นที่ใช้ในการสร้างการเชื่อมต่อกับอินเตอร์เน็ต แต่ถ้าเป็นวินโดวส์ 95 และวินโดวส์ NT จะมี WINSOCK.DLL ต่างกันเล็กน้อย

API ของวินโดวส์ซ้อกเก็ตจะเป็นตัวที่อยู่ระหว่างแอพพลิเคชันกับเน็ตเวิร์ก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.11 การติดต่อของวินโดวส์แอพพลิเคชันผ่านเครือข่ายอินเตอร์เน็ต

รูปนี้แสดงการติดต่อของวินโดวส์แอพพลิเคชันสองแอพพลิเคชันคืออินเตอร์เน็ตไคลเอนต์กับอินเตอร์เน็ตเซิร์ฟเวอร์ ซึ่งทั้งสองติดต่อผ่านเครือข่ายอินเตอร์เน็ตทางวินโดวส์ซ้อกเก็ต เป็นที่แน่นอนว่าโปรแกรมที่เราติดต่ออยู่อาจไม่เป็นวินโดวส์แอพพลิเคชัน ซ้อกเก็ตสร้างกระบวนการติดต่อแบบเดียวกันในหลายระบบปฏิบัติการ

2.8.3 การเรียกใช้ซ้อกเก็ตในโปรแกรม C++

ข้อกำหนด Winsock ระบุฟังก์ชันและพารามิเตอร์ของแต่ละฟังก์ชัน ส่วนไฟล์ WINSOCK.DLL ทำให้เราสามารถเรียกใช้ฟังก์ชันเหล่านั้นจากส่วนใดก็ได้ของโปรแกรมของเราเราจะเรียนรู้เกี่ยวกับฟังก์ชั่นเหล่านี้ในความหมายโดยทั่วไป จากนั้นสร้างขึ้นเป็นคลาสของซ้อกเก็ตหนึ่งคลาสซึ่งมี API อยู่ข้างใน

2.8.4 ฟังก์ชันซ้อกเก็ตของเบิร์กเลย์

สำหรับโปรแกรมเมอร์ที่เคยเขียนซ้อกเก็ตของระบบปฏิบัติการอื่นมาแล้วจะคุ้นเคยกับฟังก์ชันเหล่านี้เป็นอย่างดี เพราะฟังก์ชันเหล่านี้มีชื่อเดียวกัน และมีการทำงานเหมือนกันกับฟังก์ชันในซ้อกเก็ตของระบบยูนิกซ์ดั้งเดิมซึ่งถูกสร้างที่เบิร์กเลย์ ฟังก์ชันเหล่านี้ได้แก่

- closesocket
- connect
- getpeername
- getsockname
- getsockopt, setsockopt
- htonl, htons, ntohl, ntohs

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- inet_addr, inet_ntoa
- ioctlsocket
- listen
- recv, recvfrom
- select
- send, sendto
- shutdown
- socket

ถ้าหากเราไม่เคยมีประสบการณ์การเขียนโปรแกรมซ้อกเก็ตมาก่อน ฟังก์ชันทั้งสี่คือ htonl, ntohl และ ntohs มีความพิเศษคือใช้สำหรับใช้แปลงตัวเลขซึ่งแทนอินเตอร์เน็ตแอดเดรส อินเตอร์เน็ตแอดเดรสสามารถแทนด้วยตัวเลขจำนวน 4 ชุดซึ่งคั่นด้วยเครื่องหมาย "." เช่น 198.53.145.3 ซึ่งตัวเลขแต่ละจำนวนมีค่าตั้งแต่ 0 ถึง 255 สามารถซึ่งแทนด้วยรหัส 8 บิต ดังนั้นเลขทั้งสี่จะแทนด้วยรหัส 32 บิต เป็นที่น่าเสียดายว่าการแทนเลข 32 บิตสามารถทำได้สองวิธี

วิธีแรกคือแทนตัวเลขซ้ายสุด (ในกรณีนี้คือ 198) แทนด้วย 8 บิตบนสุด (8 most significant bits) ตัวเลขถัดมาทางขวาก็แทนด้วย 8 บิตที่อยู่ถัดมาข้างล่าง ทำอย่างนี้จนครบ 32 บิต เป็นการใส่จากบิตสูงมาบิตต่ำเรียกว่า "การเรียงลำดับแบบบิกอินเดียน" (big-Endian order) อีกวิธีหนึ่งจะแทนตัวเลขซ้ายสุดด้วย 8 บิตที่อยู่ต่ำสุดแล้วไล่ขึ้นมาจนครบ 32 บิต วิธีนี้เรียก "การเรียงลำดับแบบลิตเติลอินเดียน" (little-Endian order) ยกตัวอย่างเช่น แอดเดรส 0.0.0.1 จะถูกแปลงเป็น 1 ในการเรียงแบบบิกอินเดียนหรือ 16,777,216 (เท่ากับ 2^{24}) ถ้าหากเครื่องสองเครื่องที่ใช้การเรียงลำดับต่างกันจำเป็นต้องแลกเปลี่ยนแอดเดรสกัน เครื่องหนึ่งส่งตัวเลขแบบบิกอินเดียนแต่อีกเครื่องส่งตัวเลขแบบลิตเติลอินเดียน และนี่เป็นปัญหาที่ชัดเจนที่สุดของการสร้างช่องทางการติดต่อ

ระบบอินเตอร์เน็ตใช้การเรียงลำดับแบบบิกอินเดียน แต่เครื่องของเราใช้การเรียงลำดับแบบลิตเติลอินเดียน (เครื่องที่ใช้ไมโครโปรเซสเซอร์ของอินเทลใช้การเรียงลำดับแบบลิตเติลอินเดียน ขณะที่เครื่องที่ใช้ไมโครโปรเซสเซอร์ของโมโตโรล่า รวมทั้งเครื่องแมกอินทอชใช้การเรียงลำดับแบบบิกอินเดียน) ซ้อกเก็ตฟังก์ชัน hton() แปลง 32 บิตโฮสแอดเดรสไปเป็นเน็ตเวิร์กแอดเดรส (32 บิตเรียงลำดับแบบอินเดียน) ทั้งนี้โฮสแอดเดรสเดิมจะเป็นการเรียงลำดับแบบใด ก็จะมี DLL ที่ออกแบบมาเหมาะสมเพื่อใช้ในการแปลงนี้ ส่วนฟังก์ชัน hton() ก็ทำแบบเดียวกันแต่กับ 16 บิตโฮสแอดเดรส และการแปลงกลับเป็นโฮสแอดเดรสก็ใช้ฟังก์ชัน nton() และ ntons()

โดยทั่วไปฟังก์ชันที่ใช้สำหรับแปลงค่าแอดเดรสเหล่านี้จะถูกนำไปใช้เมื่อมีเครื่องหนึ่งต้องการผ่านค่าอินเตอร์เน็ตแอดเดรสหรือค่าพอร์ตไปให้เครื่องอื่น และอาจถูกนำไปใช้ในกรณีอื่น (ซึ่งหายาก)

นอกจากนี้ฟังก์ชัน inet_addr() แปลงจากสตริงของตัวอักขระ เช่น "198.53.145.3" ไปเป็นเน็ตเวิร์กแอดเดรสและในทางกลับกันฟังก์ชัน inet_ntoa() เปลี่ยนเน็ตเวิร์กแอดเดรสไปเป็นสตริงของตัวอักขระ ASCII

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ส่วนฟังก์ชันของเบิร์กเลย์ที่เหลือจะได้กล่าวถึงต่อไป

2.8.5 ฟังก์ชันฐานข้อมูล

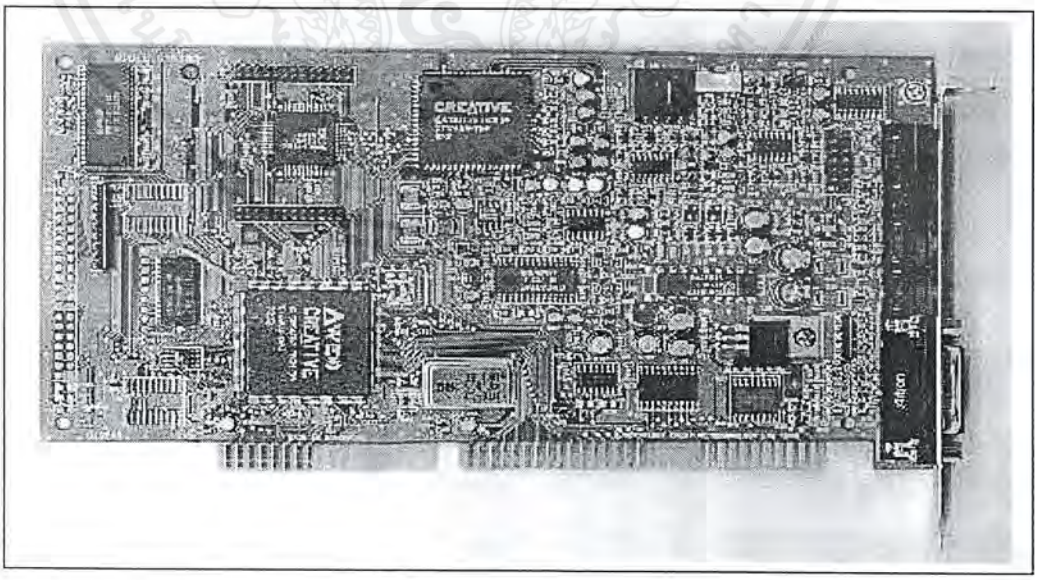
ถึงแม้ว่าฟังก์ชันเหล่านี้เรียกว่าเป็นฟังก์ชันฐานข้อมูล แต่ก็ไม่จำเป็นที่จะใช้หาข้อมูลในฐานข้อมูลเสมอไป ยกตัวอย่างเช่นอาจใช้ส่งรีเควสไปเครื่องอื่นที่อยู่บนอินเตอร์เน็ตเพื่อที่จะค้นหาเครื่องที่ต้องการ

- gethostbyaddr
- gethostname
- gethostbyname
- getprotobyname
- getservbyname
- getservbyport

ฟังก์ชันเหล่านี้สามารถแปลงกลับไปกลับมาได้ระหว่างตัวเลขที่เข้ารหัสไว้ (cryptic number) กับโค้ด (code) และชื่อเครื่อง ตัวอย่างเช่นฟังก์ชัน gethostbyname() จะให้ค่า 32

2.9 ข้อมูลการ์ดเสียง

ต้นตำรับของเสียงสังเคราะห์เกิดจากนายโรเบิร์ต มูก เป็นผู้ริเริ่มนำเอาเสียงสังเคราะห์มาใช้โดยการรวมคีย์บอร์ดเข้ากับวงจรอิเล็กทรอนิกส์ เพื่อให้ได้เสียงที่แตกต่างจากออร์แกนอิเล็กทรอนิกส์ นายโรเบิร์ต ใช้วงจรออสซิลเลเตอร์เป็นวงจรกรองสัญญาณ และวงจรขยายสำหรับสังเคราะห์เสียง และสามารถใช้ได้จริงสำหรับมืออาชีพ ยังสามารถพัฒนาให้สังเคราะห์เสียงดนตรีต่างๆ เพิ่มเติมนอกเหนือจากเปียโน และออร์แกน (รูปที่ 2.12)

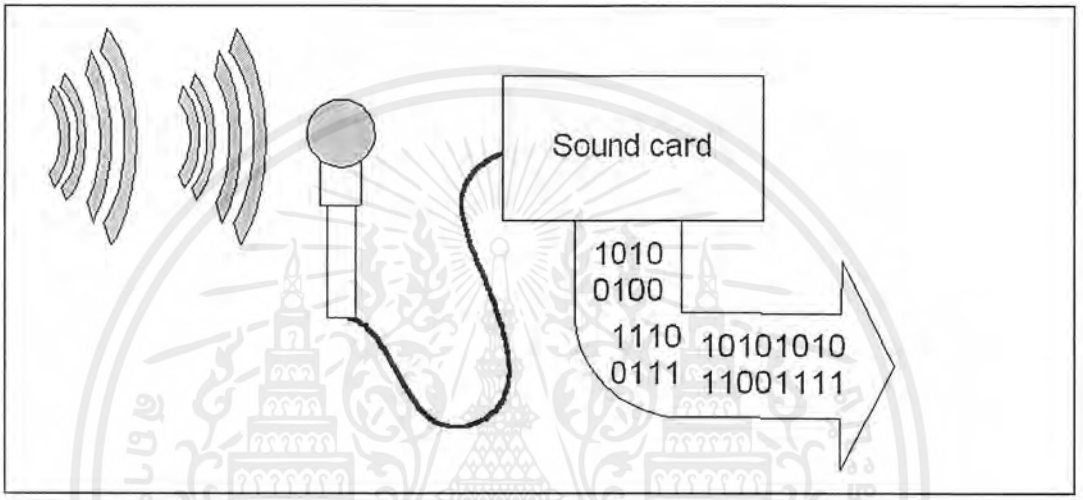


รูปที่ 2.12 ลักษณะของการ์ดเสียง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

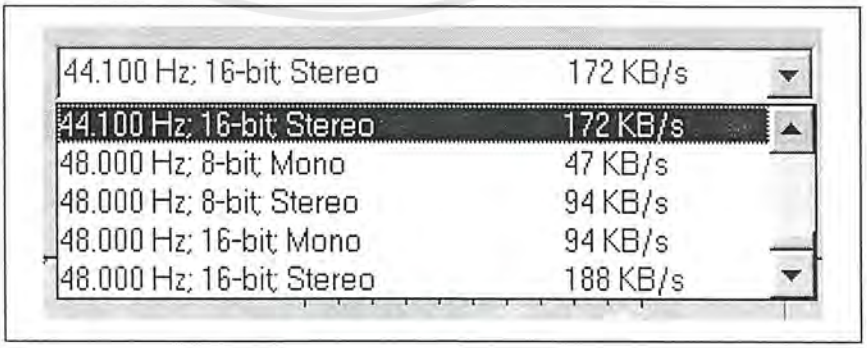
คุณสมบัติของเสียงโดยทั่วไป มักเป็นสัญญาณอนาล็อก ที่มีค่าขนาดของสัญญาณต่อเนื่องกันไป โดยสามารถแทนค่าขนาดของสัญญาณด้วยแรงดันไฟฟ้า และความถี่เสียงด้วยความดังมากกว่าสัญญาณที่มีความถี่ สูงกว่า แต่ถ้าเป็นสัญญาณดิจิตอล จะเก็บ และประมวลผลไว้ในลักษณะของตัวเลขฐานสอง คือเลข 0 และ 1

ดังนั้นการเก็บเสียงในคอมพิวเตอร์ จึงต้องแปลงสัญญาณเสียง ให้เป็นสัญญาณดิจิตอลก่อน เพื่อนำไปเก็บไว้ในหน่วยความจำของคอมพิวเตอร์ เมื่อต้องการรับฟังเสียงที่บันทึกไว้ก็จะนำข้อมูลดิจิตอลมาแปลงกลับให้เป็นสัญญาณอนาล็อกเสียก่อน (รูปที่ 2.13)



รูปที่ 2.13 กระบวนการแปลงสัญญาณของการ์ดเสียง

กระบวนการแปลงสัญญาณอนาล็อกเป็นสัญญาณดิจิตอล ในการ์ดเสียงจะเป็นหน้าที่ของชิพ ADC โดยจะทำหน้าที่วัดขนาดสัญญาณที่คลื่นเสียงตามระยะเวลาที่คงที่ เรียกว่า อัตราแซมปลิ้ง (Sampling Rate) และเรียกเวลาในการแปลงขนาดของสัญญาณเป็นข้อมูลตัวเลข เพื่อนำไปประมวลผลของชิพ ADC ว่าคาบเวลาของการแซมปลิ้ง (Sampling Time) ค่าตัวเลขที่ได้เป็นเลขฐานสิบ ส่วนเอาท์พุทจริงจากชิพ ADC เป็นเลขฐานสอง



รูปที่ 2.14 ค่าที่ได้จากการแซมปลิ้งสัญญาณ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ดังนั้นจึงต้องมีกระบวนการแปลงเลขฐานสิบเป็นเลขฐานสอง ค่าที่เก็บจะเริ่มจากเลขศูนย์ตามด้วยเลขฐานสองต่างๆ กรรมวิธีนี้เป็นกระบวนการมอดูเลชั่นแบบหนึ่งเรียกว่า พัลส์โค้ดมอดูเลชั่น (Pulse Code Modulation : PCM)

ความแม่นยำในการวัดขนาดของสัญญาณอนาล็อกกำหนดได้ โดยจำนวนบิตของเลขฐานสอง ในระบบดิจิตอลของชิพ ADC ถ้าหากใช้ชิพขนาด 8 บิต จะได้ขนาดของสัญญาณที่วัดได้ช่วงละ 256 ค่า และถ้าเพิ่มเป็น 12 บิตจะได้รายละเอียดเพิ่มเป็น 4096 ค่า และเมื่อเพิ่มขึ้นเป็น 16 บิต จะมีรายละเอียดเพิ่มเป็น 65535 ค่า และยิ่งเพิ่มความละเอียดมากขึ้นก็ต้องใช้เนื้อที่ของหน่วยความจำมากขึ้น และการแปลงสัญญาณข้อมูลดิจิตอลกลับมาเป็นสัญญาณอนาล็อก (Digital to Analog Converter : DAC) โดยใช้ความเร็วในการแปลงกลับเท่ากับการแซมปลิ้ง เอาท์พุทที่ได้จะมีลักษณะเป็นขั้นบันไดต่อเนื่องกันไป จากนั้นต้องผ่านกระบวนการแปลงคลื่นขั้นบันไดให้เป็นคลื่นปกติ โดยการใช้วงจรกรองความถี่ชิพ ADC และ DAC ใช้งานควบคู่กันเพื่อการบันทึก และสังเคราะห์เสียงออกมา เป็นผลให้ระดับเสียงแบบสเตอริโอ ใช้หน่วยความจำที่บรรจุข้อมูล เป็นสองเท่าของระบบโมโน โดยปกติแล้วอัตราการแซมปลิ้งสัญญาณที่เหมาะกับการ์ดเสียงที่ติดตั้งชิพ ADC และ DAC จะใช้ค่า 5.0125 และ 11.025 KHz ทำให้คุณภาพเสียงเท่ากับเสียงจากโทรศัพท์ ส่วนการแซมปลิ้ง 16 บิต เป็นขนาดมาตรฐานในวงการค้าของเครื่องเล่นซีดี และ งานบันทึกเสียงระบบดิจิตอลออดิโอเทป (Digital Audio Tape : DAT) เครื่องเล่นซีดีมีอัตราการแซมปลิ้งที่ 44.1 KHz และ DAT จะใช้ 48 KHz การ์ดอะแดปเตอร์เสียงส่วนใหญ่ใช้ 8 บิต มีอัตราการแซมปลิ้งที่ 22.05 KHz

การสังเคราะห์เสียง

การสังเคราะห์เสียงดิจิตอลซนิดเอฟเอ็ม สามารถเลียนเสียงต่างได้มากมาย แต่ไม่ทั้งหมดเสียงที่มีฮาร์โมนิคเปลี่ยนแปลงรวดเร็ว จะไม่สามารถทำการสังเคราะห์เสียงได้ แต่บันทึกเสียงเหล่านั้นไว้แล้วแปลงเป็นข้อมูลดิจิตอลจากนั้นมาเล่นกลับ ภายหลังจากการสังเคราะห์เสียงต่างๆ ด้วยการบันทึกลงซีดี ทำได้โดยนำเสียงดนตรีแต่ละชิ้นทำการแซมปลิ้งเสียงลงแผ่นซีดี 16 บิต และเก็บไว้ด้วยความเร็ว 44.1 KHz โดยการปรับเปลี่ยนอัตราการแซมปลิ้งทำให้พิตช์เปลี่ยน และสร้างโน๊ตเสียงใหม่ๆ ได้อีกมากมาย หรือถ้าเปลี่ยนอัตราความเร็วในการเล่นเหลือเพียงครึ่งหนึ่ง หรือเพิ่มความเร็วเป็นสองเท่าก็ได้ ซึ่งเป็นผลให้เพิ่มหรือลดระดับเสียงสูงจนถึงต่ำสุดของเสียงอีกหนึ่งอ๊อคเตป (Octave)

คุณสมบัติของการ์ดเสียง

การ์ดเสียงโดยทั่วไปมีข้อกำหนดไว้ 5 ข้อ คือ

1. สามารถเล่นไป – กลับ โดยใช้ข้อมูลที่เก็บไว้ในรูปพัลส์โค้ดมอดูเลชั่น
2. สามารถสังเคราะห์เสียงได้ 128 เสียง
3. ส่วนมิกเซอร์ (Mixer) เสียง จะทำหน้าที่ควบคุมแหล่งจ่ายเสียง และระดับสัญญาณเสียงที่ลำโพง ในบางการ์ดอาจมีวงจรควบคุมเสียงทุ่ม และเสียงแหลมเพิ่มเติมด้วย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4. การบันทึกเสียงจะบันทึกเสียงดนตรี หรือเสียงต่างๆ ในรูปของพัลส์โค้ดมอดูเลชั่น อาจบันทึกเป็นแบบสเตอริโอ หรือโมโน และสามารถควบคุมอัตราการแซมปลิ้งที่เหมาะสมกับความคมชัดของเสียงได้

5. ต้องมีอินพุท และเอาท์พุท เพื่อให้สามารถใช้การ์ดเสียงกับเครื่องดนตรีมิดี้ภายนอกได้ ปกติ การ์ดจะมีแอดเดรสสำหรับต่อกับจอยสติ๊ก, แอดเดรสมิดี้ และแอดเดรสสำหรับต่อกับซีดีรอมไดร์ฟ การที่มีแอดเดรสจอยสติ๊กช่วยให้แน่ใจว่า สามารถต่อกับคอมพิวเตอร์ที่คอมแพตติเบิลกันได้ด้วยแอดเดรสอินพุท และ เอาท์พุทของ 8086 มักมีคอนเนคเตอร์หลายขาที่เรียกว่า เฮดเดอร์ (Header) เพื่ออินเตอร์เฟสกับซีดีรอมได้ บางทีก็แบ่งเป็นการ์ดเล็กต่างหากแล้วมีขาสำหรับเสียงต่อกับเมนบอร์ด

2.10 ทฤษฎีและวิธีการในการย่อขนาดของข้อมูล

The Nyquist Criterion

การเปลี่ยนรูปแบบสัญญาณ Analog ไปยังสัญญาณ Digital จะมีพื้นฐานอยู่บนหลักการของ Nygquist คือจะต้องทำการสุ่มของสัญญาณอย่างน้อย 2 ครั้งของความถี่ที่สูงที่สุดของสัญญาณความถี่นั้นๆ จึงจะสามารถทำการสร้างรูปสัญญาณนั้นๆ ให้กลับคืนเหมือนสัญญาณต้นฉบับได้

สัญญาณ F(t) สามารถที่จะปลูกสร้างกลับจากตัวอย่างที่ถูกเก็บไว้ได้ โดยใช้ Low Pass Filter ตาม ทฤษฎีแล้วค่าข้อมูลที่ได้จากการสุ่มตัวอย่างคือค่าแอมพลิจูด ตามเวลาที่กำหนดไว้และสัญญาณที่ต่อเนื่องกันจะถูกสร้างใหม่โดยใช้ วงจร Low Pass Filter ในการใช้งานจริงกับเสียงพูด จำนวนของการสุ่มตัวอย่างนั้นไม่เพียงพอต่อคุณภาพที่ได้รับ แต่เราก็สามารถที่จะปรับเปลี่ยนได้ ในปี ค.ศ. 1933 Harry Nyquist ได้กำหนดจำนวนต่ำสุดของการสุ่มตัวอย่างคือ

$$fs > 2\ bw$$

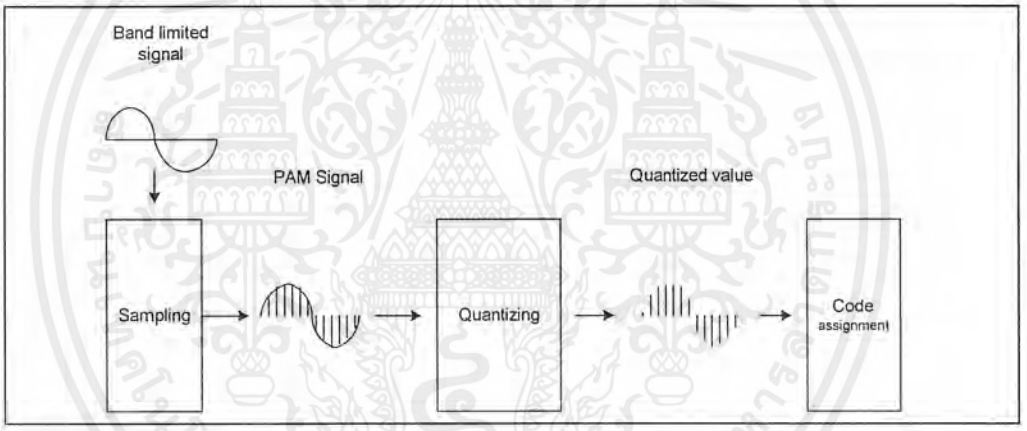
โดยที่ fs = ความถี่ของการสุ่มตัวอย่าง bw = ความถี่ที่สูงที่สุดของสัญญาณ input ตามทฤษฎีของ Nmyquist ซึ่งแสดงถึงว่าการ Sampling จะถูกพิจารณาด้วย Pluse ที่ต่อเนื่อง และมีความถี่ที่เท่าๆ กัน รวมถึงขนาดของ Pluse ด้วยแหล่งสัญญาณที่จะนำมาทำการ Sampling จะผ่านระบบที่เรียกว่า Band Limit Filter เพื่อจุดประสงค์ 2 ประการคือ เพื่อจำกัดความถี่ที่เกินจากความสามารถที่จะได้จากขั้นตอน PAM ยังคงอยู่ในรูปของสัญญาณ Analog เพราะว่ามันยังมี Amplitude หลายค่าและขั้นตอนต่อไป คุณลักษณะนี้จะถูกเปลี่ยนโดยการแทนค่าของสัญญาณ ด้วยจำนวนเลขซึ่งจะแทนค่าของ Amplitude อันนั้น ค่าของเลขจำนวนจะถูกทำให้อยู่ในรูปของจำนวน binary ซึ่งมีค่าสองค่าคือ 1 และ 0 เทคนิคนี้เราเรียกว่า Pulse code Modulation (PCM) วิธีการเฉพาะที่จะใช้ในการกำหนดค่าของแต่ละครั้งจากการ Sampling จะถูกเรียกว่า Quantizing และแต่ละครั้งของ PAM จะมีการประมาณค่าโดยใช้เลข Binary จำนวนเป็น Bit เพื่อแทนค่า

การทำ Quantizing แบบดั้งเดิมจะนำมาซึ่งข้อผิดพลาดเพราะค่าของการแทนขนาดของการ Sampling แต่ละค่ามีความแตกต่างกัน แต่อย่างไรก็ตามการ Quantizing 1 ครั้งสัญญาณจะถูกหน่วงให้ช้าลงในทุกๆ ช่วงโดยปราศจากการลดทอนของสัญญาณโดยการควบคุมจำนวนขั้นของการ Quantizing

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จำนวนของค่า n นั้นก็ค่อนข้างจะสำคัญ (ในรูป A กำหนดให้ความสัมพันธ์แบบคงที่ระหว่างสัญญาณภาพ PAM กับ PCM) ค่าของ n จะถูกระบบแบ่งให้มีขนาดครึ่งหนึ่งของ Amplitude ของ PAM หรือในอีกทางหนึ่งคือค่าของจำนวน Bit จะถูกแบ่งโดยขอบเขตของ 2n เช่น การใช้ค่า 8 bit ในการแทนจะมีความเที่ยงตรงน่าเชื่อถือมากกว่า

ระบบ PCM แบบเก่าจะใช้จำนวน bit เท่ากับ 7 bit แต่ในระบบใหม่จะใช้ค่า n เท่ากับ 8 ถ้าการ Quantize กำหนดจำนวนของขั้น ออกเป็น 128 ขั้น เราจะใช้ค่า n เท่ากับ 7 bit ในการแทนค่าแต่ละครั้งของการ Sampling (2^7 เท่ากับ 256) นอกจากนี้ถ้าเราจะใช้การ Quantize ที่ 128 ระดับ เราต้องการส่งข้อมูลเท่ากับ 56,000 bit / วินาที (8,000 x 7 เท่ากับ 56,000) และถ้าต้องการให้มีจำนวนของการ Quantize ที่ 256 ระดับ เราต้องส่งข้อมูลมีปริมาณถึง 64,000 bit ต่อวินาที (8,000 x 8 เท่ากับ 64,000) จำนวน bit ของข้อมูลที่มากขนาดนี้ต้องการช่องสัญญาณที่ส่งข้อมูลได้ดี แต่ในระบบของ Modem เราต้องทำการลดจำนวนของข้อมูลจึงจะสามารถส่งข้อมูลได้ เนื่องจาก Modem มีความเร็วในการส่งข้อมูลไม่มากพอ



รูปที่ 2.15 แสดงกระบวนการการแปลงสัญญาณอะนาลอกโดย PCM

2.10.1 Sampling และ Quantizing Error

การส่งสัญญาณแบบ Digital ใช่ว่าจะขจัดปัญหาได้ สัญญาณ Digital สามารถถูกทอนสัญญาณได้จากหลายๆ ทาง คือความถี่ในการ Sampling ปัญหานี้สามารถใช้แก้ไขได้โดยเพิ่มความถี่ของการ Sampling แต่แน่นอนมันต้องการอุปกรณ์ที่มีราคาแพง เพื่อรองรับกับจำนวนข้อมูลที่เพิ่มขึ้นและไม่มีเทคนิคใดที่สามารถขจัดปัญหาและถูกลดทอนสัญญาณให้หมดไปได้ ส่วนปัญหาที่ 2 คือ ปัญหาที่เกิดจาก Quantizing Error คือขบวนการที่การ Quantizing ไม่สามารถแทนค่าที่ได้มาจาก PAM ได้อย่างแม่นยำ มันเป็นเพียงค่าประมาณเท่านั้น ผลที่ตามมาคือสัญญาณที่จะถูกสร้างกลับไม่เหมือนกับสัญญาณต้นฉบับ ปัญหานี้เราสามารถแก้ไขได้โดยการเพิ่มจำนวน Step ในการทำ Quantizing แต่การเพิ่มจำนวน Step นั้น ก็นำมาซึ่งปริมาณข้อมูลที่เพิ่มขึ้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.10.2 Companding

เทคนิคที่สามารถลดขนาด Amplitute ลงเพื่อลดจำนวนของระดับการ Quantizig หรือในอีกทางหนึ่งคือแอมพลิจูดที่มีขนาดเล็ก จะถูกขยายเป็นการเพิ่มจำนวนของระดับการ Quantizing และลดจำนวนของระดับการ Quantize โดยรวมลง หลังจากที่สัญญาณถูกตีความ มันจะกลับไปอยู่ในรูปแบบเดิม การผสมผสานระหว่างการบีบอัด (Compression) และการขยายช่วง (Expanding) จึงเรียกวิธีนี้ว่า Companding การ Quantize สัญญาณที่มีความสูงมากจะถูกกำหนดให้มีระดับการ Quantize ต่ำ (จำนวนขั้นของการทำ Quantize น้อย) แต่การ Quantize สัญญาณที่มีความสูงน้อยจะถูกกำหนดให้มีการ Quantize มาก (จำนวนขั้นของการทำ Quantize มาก) ตามความเหมาะสม

2.10.3 G.728 Low Delay CELP Codec

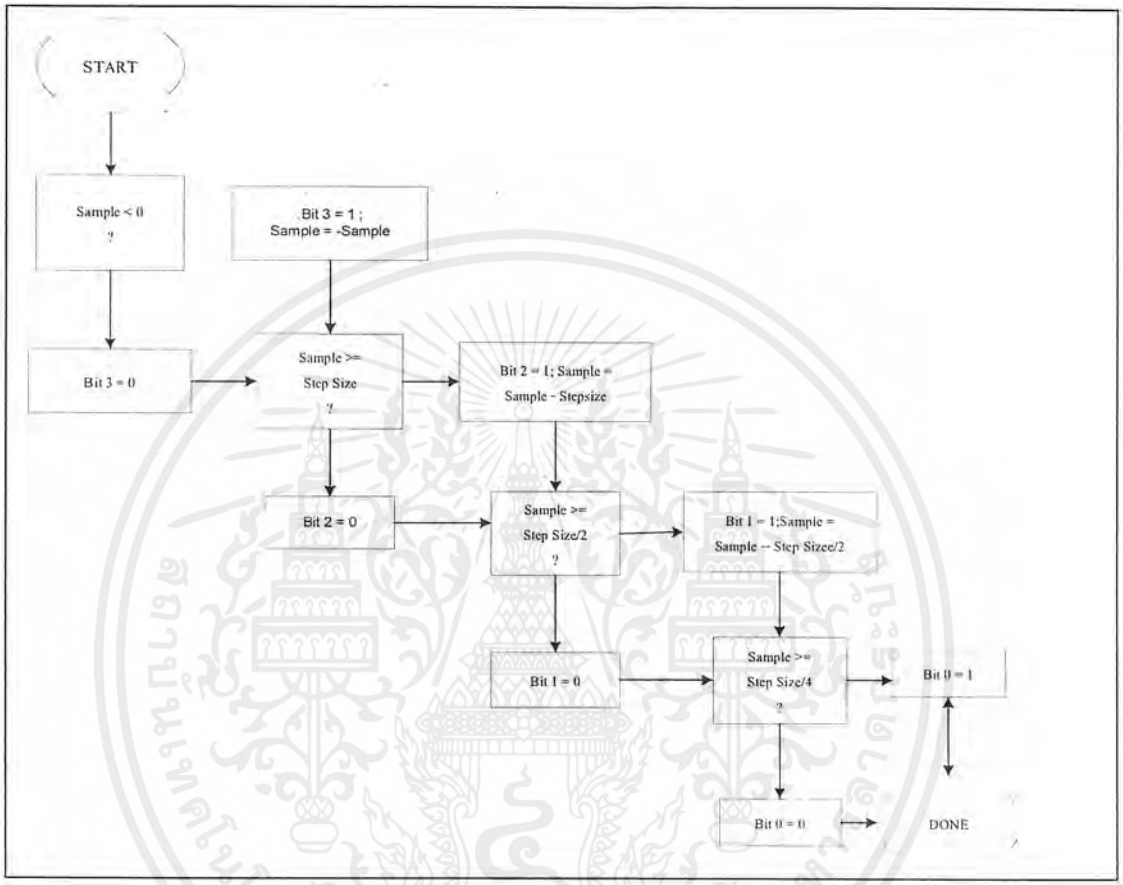
ที่จำนวนของข้อมูลประมาณ 16KBITS/S คุณภาพของสัญญาณจะตกลงอย่างมาก ดังนั้นอัตราของการเข้า และถอดรหัส โดยเฉพาะอย่างยิ่ง ตามวิธีการของ CELP ที่ถูกดัดแปลงมามีแนวโน้มที่จะนำมาใช้ได้ แต่อย่างไรก็ตามเนื่องจากการตัดสินใจล่วงหน้าของส่วน short term ที่ใช้ในการเข้า และถอดรหัสมักเกิดความล่าช้า ความล่าช้าในการเข้า และถอดรหัสนี้จะถูกกำหนดโดยเวลา เมื่อสัญญาณเสียงพูดมาถึงส่วนของชุด Codec จะตอบสนองต่อตัวอย่างสัญญาณเสียงที่ได้รับ โดยสร้างเอาท์พุทออกมา ปริมาณของข้อมูลที่เกิดจากการเข้ารหัสจะถูกส่งไปยังตัวถอดรหัส (Decoder) โดยปกติการเข้า และถอดรหัสเสียงพูดจะใช้เวลาอยู่ในช่วง 50-100 มิลลิวินาที และความล่าช้าที่เกิดขึ้นจะเป็นปัญหาสำคัญ ดังนั้นในปี ค.ศ.1988 CCITT ได้กำหนดมาตรฐานที่ปริมาณข้อมูลเท่ากับ 16 KBITS/S ออกมาเป็นมาตรฐาน ความต้องการส่วนใหญ่ของการเข้า และถอดรหัสข้อมูลควรจะมีคุณภาพเทียบได้กับ G.721 32 KBITS/S ADPCM และควรมีความล่าช้าน้อยกว่า 50 มิลลิวินาที และทางอุดมคติควรน้อยกว่า 2 มิลลิวินาที ความต้องการของ CCITT นี้ได้ถูกบรรจุอยู่ในวิธีการเข้ารหัสแบบ CELP ซึ่งวิธีการนี้ได้ถูกพัฒนาต่อโดย AT&T Bell Lab และถูกตั้งเป็นมาตรฐานในปี ค.ศ.1992 ได้ชื่อว่า G.728

วิธีการเข้า และถอดรหัสวิธีนี้จะใช้การคำนวณค่าที่เคยมีอยู่ เพื่อสร้างส่วนของ short term filters ที่มีประสิทธิภาพ นั่นหมายความว่า ต้องใช้ Buffer ที่สามารถเก็บข้อมูลได้มากกว่า 20 มิลลิวินาที หรือมากกว่านั้น สัญญาณเสียงที่เข้ามาจะถูกคำนวณค่าการ filter ที่เหมาะสมจากสัญญาณเสียงที่ถูกสร้างไปแล้ว นั่นหมายความว่า การเข้า และถอดรหัสโดยวิธีนี้สามารถใช้ความยาวของ Frame ข้อมูลได้สั้นกว่าวิธีการ CELP ที่มีอยู่เดิมได้ และ G.278 จะใช้ความยาวของ Frame ที่มีค่าเพียง 5 ตัว และมีความล่าช้าเพียง 2 มิลลิวินาที ที่ปริมาณความต้องการสูงๆ การคาดคะเนแบบสั้นจะถูกนำมาใช้ ฉะนั้นเราจึงไม่มีความจำเป็นที่จะต้องใช้การคาดคะเนแบบ long term ดังนั้นข้อมูลจำนวน 10 Bits จะถูกนำมาใช้สำหรับค่า 5 ตัวที่ 16 KBPS โดยใช้แทนวิธีการของ Fixed Code Book และ อีก 3 Bits จะถูกใช้แทนขนาด และการใช้ค่าที่เคยเก็บไว้ เพื่อช่วยในการเทียบระดับของ Quntize และที่ขบวนการถอดรหัสจะใช้ค่า filter แบบเดิม เพื่อเพิ่มคุณภาพของสัญญาณเสียง และทั้งหมดจะนำสู่ปริมาณข้อมูลเข้า และถอดรหัสที่ 16 KBPS และมีความล่าช้าน้อยกว่า 2 มิลลิวินาที นอกจากนี้คุณภาพของเสียงพูดจะเท่ากับ G.721

2.10.4 IMA ADPCM COMPRESSION

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การบีบอัดข้อมูลโดยวิธี IMA ADPCM นี้จะทำการคำนวณความแตกต่างระหว่างค่าของการสุ่มปัจจุบัน X(N) กับค่าการทำนายของการสุ่มครั้งก่อน X(N-1) และจะใช้ขนาดความแตกต่างนี้ในการคำนวณระดับของการ Quantize โดยวิธีการของ IMA ALGORITHM แต่ละครั้งของการ Quantize ระดับของ Quantize จะถูกแทนด้วยจำนวนเลข 4 Bits และ Bit ที่ 4 จะเป็น Bit เครื่องหมาย



รูปที่ 2.16 ไดอะแกรมแสดง Quantization

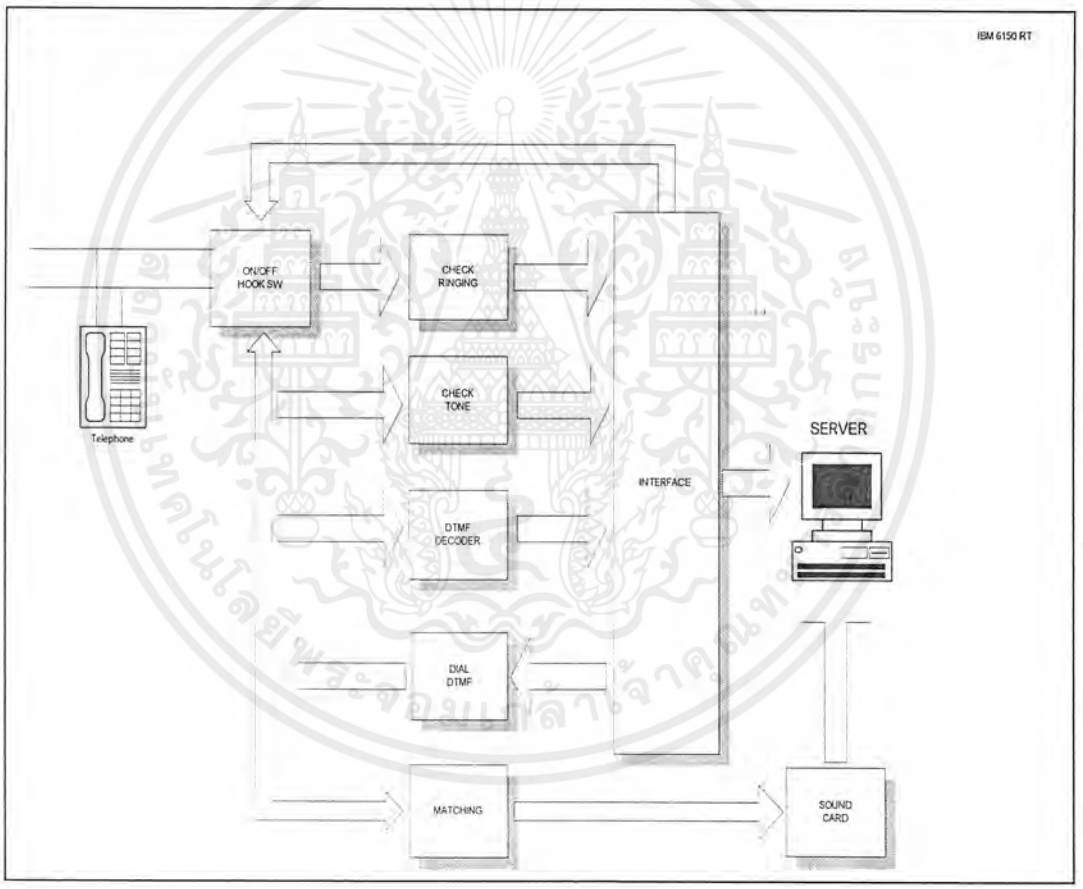
จาก Diagram แสดงขบวนการ Quantization ซึ่งสามารถแสดงให้เห็นถึงผลลัพธ์ของความแตกต่าง และระยะห่างของ X(N) และX(N-1) ซึ่งคือระยะห่างของแต่ละ Stepในปัจจุบันนั่นเอง ระยะห่างที่กล่าวถึงนี้จะถูกแทนด้วยค่า Bit ที่ 3 ขบวนการจัดแบ่งขนาด Step ออกเป็นครึ่งหนึ่ง และทำการเปรียบเทียบอีกครั้งกับระดับความแตกต่างจริงที่พบ ถ้าระดับความแตกต่างไม่เท่ากัน Bit ที่ 2 จะถูกกำหนดเป็น 1 ถ้าค่าของการ Sampling เท่ากับค่าของ Step Size ตัวเข้ารหัสจะทำการแบ่ง Step Size ออกเป็นครึ่งหนึ่งแล้วทำการเปรียบเทียบอีก และถ้ามีความแตกต่างกันอีก Bit ที่ 1 จะถูกกำหนดให้เป็น 1 ดังนั้น 3 Bits นี้จะแทนระดับการ Quantize

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 3
การคำนวณ และการสร้าง

การออกแบบ

การออกแบบโครงงานนี้ จะแยกเป็นสองส่วน คือ ส่วนของฮาร์ดแวร์และซอฟต์แวร์ซึ่งส่วนของฮาร์ดแวร์ได้แก่ วงจรอินเตอร์เฟส วงจรถอดรหัสสัญญาณความถี่คู่ วงจรตรวจสอบสัญญาณความถี่เสียง วงจรตรวจสอบสัญญาณกระดิ่ง วงจรหมุนรหัสความถี่คู่ วงจรไฮบริดจ์ และวงจรควบคุมการยกหูและวางหู ในส่วนของซอฟท์แวร์จะเป็นโปรแกรมควบคุมการทำงานของตอบรับและโอนสายโทรศัพท์อัตโนมัติ โดยการทำงานทั้งหมดแสดงการทำงานของชุดตอบรับและโอนสายโทรศัพท์อัตโนมัติโดยใช้ไมโครคอมพิวเตอร์ ในรูปที่ 3.1 มีการทำงานดังนี้



รูปที่ 3.1 ผังการทำงานทั้งหมดแสดงการทำงานของชุดตอบรับและโอนสายโทรศัพท์อัตโนมัติ

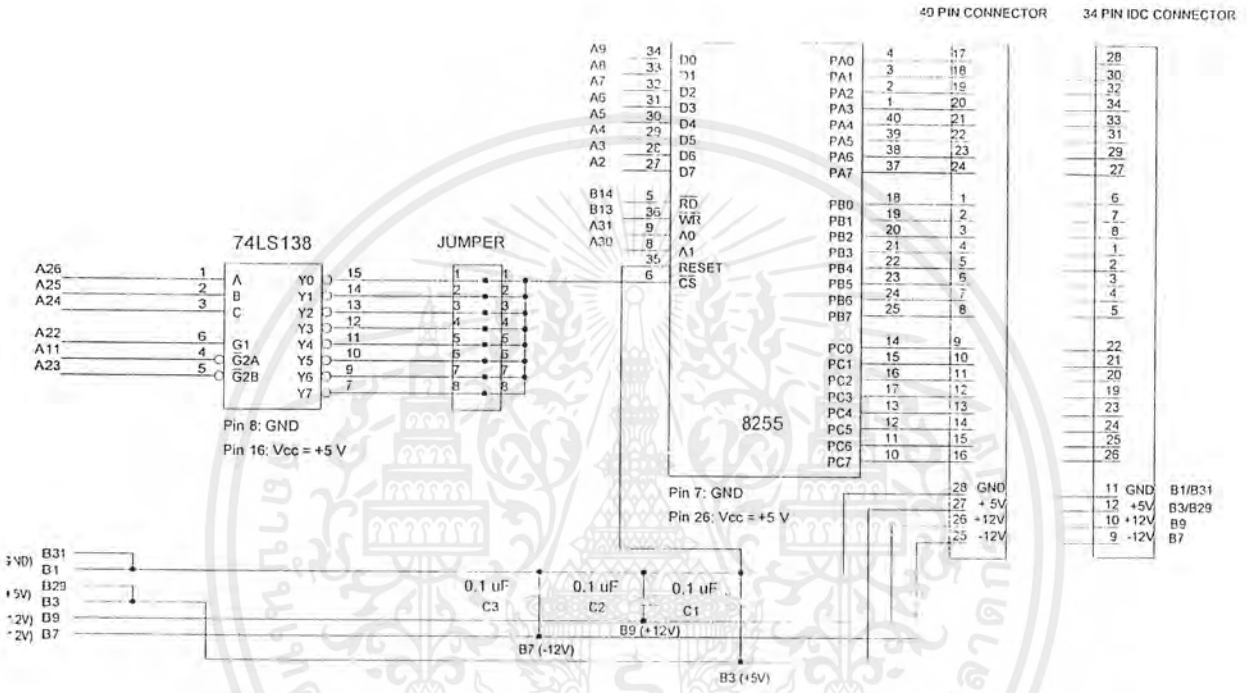
3.1 ส่วนของการเชื่อมต่อกับระบบคอมพิวเตอร์ (INTERFACE CARD)

ในภาคของการเชื่อมต่อระหว่างคอมพิวเตอร์กับส่วนของอุปกรณ์ภายนอกจะผ่านทาง I/O พอร์ต ที่เป็นสล็อตบนคอมพิวเตอร์ ซึ่งในโครงงานนี้ใช้การเชื่อมต่อ (interface) โดยการ์ด I/O เสียบลง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บนคอมพิวเตอร์ตั้งแต่รุ่น 486 จนถึงปัจจุบัน อาศัยบัสตำแหน่ง (addess bus) , บัสข้อมูล (data bus) , สัญญาณควบคุมการอ่านและเขียน , สัญญาณอินพุตและเอาท์พุต , ไฟเลี้ยงจากเครื่องคอมพิวเตอร์

โครงงานนี้ในส่วนของภาคอินเตอร์เฟส สามารถแบ่งออกเป็น 2 ส่วนใหญ่ๆ 2 ส่วนคือ ส่วนของฮาร์ดแวร์ และส่วนของซอฟแวร์ โดยในส่วนของฮาร์ดแวร์ คือตัวการ์ด I/O จะใช้ IC 8255 A เป็นหลักสายสัญญาณอินพุตและเอาท์พุตทั้ง 24 เส้นของ 8255 A มีระดับสัญญาณ TTL อินเตอร์เฟสกับวงจรลอจิกอื่นๆ ดังรูปที่ 3.2



รูปที่ 3.2 วงจรอินเตอร์เฟส

ในส่วนของวงจรอินเตอร์เฟส จะใช้ ไอซ์ 8255A ในตัวเหล็กโดยจะมาเชื่อมต่อกับสายสัญญาณอินพุตและเอาท์พุตจากอุปกรณ์ดีเทค (DETECT) สัญญาณโทรศัพท์จำนวน 24 เส้น สัญญาณ จะมีระดับสัญญาณ TTL จึงเป็นการง่ายในการใช้ 8255A อินเตอร์เฟสกับวงจรลอจิกอื่นๆ รวมทั้งวงจรโดย IC CMOS จากวงจรข้างต้นจะใช้ไอพี 74LS138 นำมาใช้ร่วมกับ 8255A เพื่อใช้ในการถอดรหัสตำแหน่งควบคุมให้การ์ด I/O ทำงาน ตามเบอร์พอร์ตที่ได้ทำการออกแบบไว้

การทำงานหลักจะแบ่งออกเป็น 2 ส่วนใหญ่ๆ คือ

1. ฮาร์ดแวร์ คือ ตัวการ์ดอินเตอร์เฟส ที่กำลังกล่าวถึง
2. ซอฟแวร์ คือ โปรแกรมที่จะมาควบคุมการทำงานของการ์ด

ส่วนประกอบของฮาร์ดแวร์นี้จะสามารถเลือกแอดเดรสของการ์ดได้ให้ตรงกับแอดเดรส (Addess) ในคอมพิวเตอร์ที่ว่างอยู่ ซึ่งในเครื่องคอมพิวเตอร์เครื่องอื่นต้องเซตค่าแอดเดรสใหม่ได้ที่ดิสท์สวิตซ์ตามตำแหน่งต่างๆ (ตารางที่ 3.1)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตำแหน่งสวิตช์ที่ "ON"	ช่องแอดเดรส
S_1	$XXD
S_2	$XX3NC
S_3	$XX9
S_4	$XX7NC
S_5	$XX5
S_6	$XXBNC
S_7	$XX1
S_8	$XXFNC
S_9	$27X
S_{10}	$26X
S_{11}	$25X
S_{12}	$24X
S_{13}	$23X
S_{14}	$22X
S_{15}	$21X
S_{16}	$20X

ตารางที่ 3.1 แสดงตำแหน่งสวิตช์ในการควบคุมพอร์ต

สมมุติเราต้องการตั้งแอดเดรสใหม่ที่ตำแหน่ง 20DH ก่อนอื่นจะต้องเลื่อนดิสท์สวิตช์ไปไว้ที่ตำแหน่ง "OFF" ก่อน จากนั้นจึงเลื่อนดิสท์สวิตช์ S_1 และ S_{16} ไปอยู่ตำแหน่ง "ON" ก็จะได้ค่าของแอดเดรสใหม่ตามที่ต้องการสังเกตุ S_2, S_4, S_6 และ S_8 ค่าแอดเดรสจะลงท้ายด้วย NC หมายถึงห้ามต่อกับวงจร

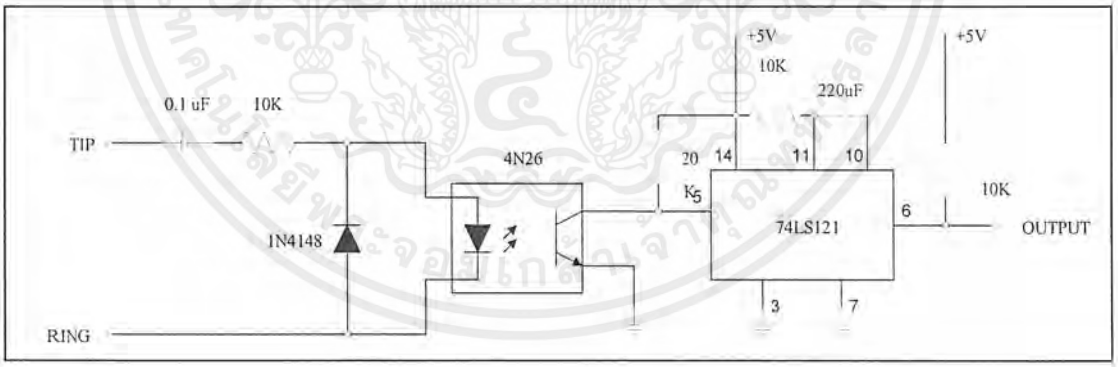
การควบคุมการ์ดนี้ใช้ IC_1 และ IC_2 เป็นตัวถอดรหัสการเลือกการ์ดจากเครื่อง PC เพื่อส่งสัญญาณ $\overline{CS}$ เลือกชิพ 8255A ใช้ทำงานโดย $\overline{CS}$ จะแอคทีฟ "0" ตามวงจรในรูปที่ 3.1 เครื่อง PC จะต้องส่งแอดเดรสมาที่บัสแอดเดรส (addess bus) $A_0 - A_{EN}$ เป็นค่า 010000000XX IC_1 และ IC_2 จะไม่นำไปใช้ แต่ 8255A จะนำไปใช้เอง เมื่อ CS แอคทีฟ "0" 8255A ก็จะทำงานโดยรับสัญญาณจาก $A_0 - A_1$ มาถอดรหัสโดยค่า A_0 และ A_1 นี้ IC_1 และ IC_2 จะไม่นำมาใช้ แต่ 8255A ก็จะทำงานโดยรับสัญญาณจาก $A_0 - A_1$ มาถอดรหัสเลือกพอร์ตใช้งานพร้อมกับการรับหรือส่งข้อมูลระหว่างพอร์ตที่ถูกเลือกใช้กับบัสข้อมูล ตามโหมดการทำงานที่ตั้งไว้ตอนเริ่มใช้งาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.2 วงจรตรวจสอบสัญญาณกระดิ่ง (Check Ringing)

โดยวงจรตรวจสอบสัญญาณกระดิ่งจะคอยตรวจสอบสัญญาณกระดิ่งที่เข้ามา ซึ่งเมื่อมีสัญญาณความถี่คู่เข้ามา จะผ่านวงจรตรวจจับซึ่งจะทำการเปลี่ยนไปอยู่บนรูปสัญญาณไบนารี ผ่านไปให้ภาคอินเตอร์เฟส เชื่อมกับส่วนซอฟแวร์ ตอบรับเสียง โทรศัพท์ขึ้นพร้อมทั้งเชื่อมต่อโทรศัพท์ ในข่ายให้ตามหมาย เลขเฉพาะส่งสัญญาณเสียงตอบรับออกมาจากแผ่นวงจรเสียงผ่านวงจรไฮบริดออกไปเพื่อบอกขั้นตอนการทำงานของระบบ ให้ผู้ที่เรียกฟังและปฏิบัติตาม ขณะนี้วงจรตรวจสอบสัญญาณกระดิ่งจะถูกตัดออกจากคู่สายโทรศัพท์ และนำคู่สายโทรศัพท์มาต่อเข้ากับวงจรตรวจสอบสัญญาณความถี่เสียง วงจรถอดรหัสสัญญาณความถี่คู่ วงจรหมุนรหัสสัญญาณ ความถี่คู่ และวงจรไฮบริด โดยวงจรถอดรหัส สัญญาณความถี่คู่จะคอยรอรับสัญญาณความถี่คู่ที่เข้ามาทางคู่สายโทรศัพท์ แล้วแปลงเป็นสัญญาณดิจิตอลขนาด 4 บิต ส่งไปให้กับคอมพิวเตอร์โดยผ่านวงจรอินเตอร์เฟส เพื่อทำการเชื่อมต่อวงจรให้ทำงานร่วมกับคอมพิวเตอร์ได้

จากนั้นคอมพิวเตอร์จะประมวลผลค่าสัญญาณที่รับเข้ามา และส่งให้วงจรทำงานตามคำสั่งนั้น ๆ คือ ถ้าหากผู้เรียกต้องการติดต่อกับบุคคลภายใน ชุดตอบรับก็จะโดนสายโทรศัพท์ให้ผู้เรียกภายนอกและบุคคลภายในตามที่กดเลขหมายมา สัญญาณเลขหมายความถี่คู่ที่กดเข้ามาจะผ่านคู่สายโทรศัพท์เข้ามายังวงจรถอดรหัสสัญญาณความถี่คู่ เพื่อแปลงเป็นสัญญาณดิจิตอลส่งไปให้คอมพิวเตอร์ จากนั้น โปรแกรมจะสั่งให้วงจรตรวจสอบสัญญาณความถี่เสียง ตรวจสอบสภาวะสัญญาณปลายทาง หากสัญญาณปลายทางเป็นสัญญาณให้หมุน โปรแกรมก็จะสั่งให้วงจรแปลงสัญญาณความถี่คู่แปลงสัญญาณดิจิตอลกลับมาเป็นสัญญาณความถี่คู่ แล้วส่งออกไป



รูปที่ 3.3 วงจรตรวจสอบสัญญาณกระดิ่ง

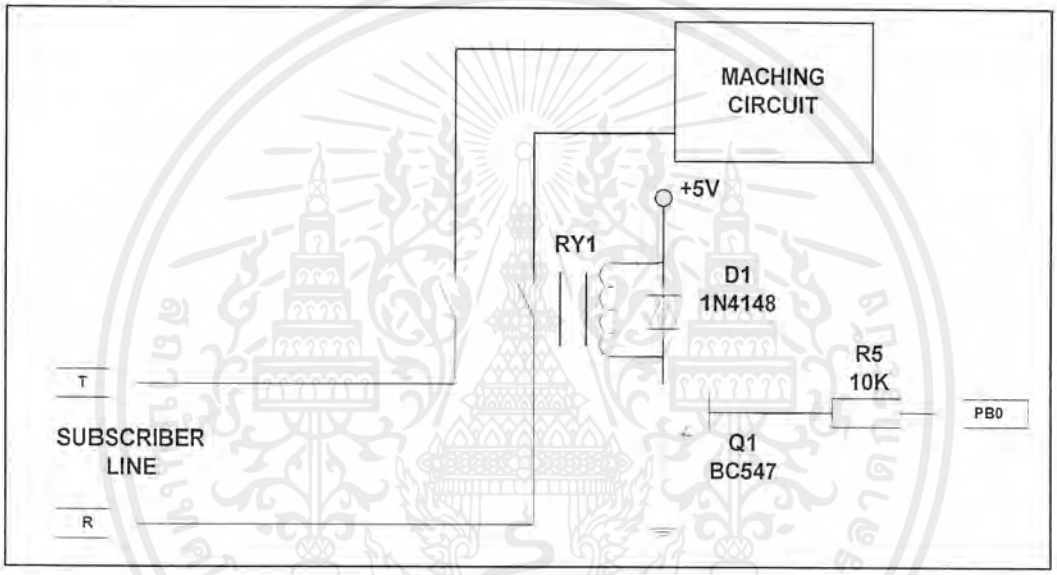
ในส่วนของวงจรตรวจสอบสัญญาณกระดิ่งจะใช้ออปโต้ทรานซิสเตอร์เป็นหลักสัญญาณจากคู่สายโทรศัพท์จะต่อเข้าวงจรโดยผ่านตัวต้านทาน R1 และ C1 ไปยังไดโอดบริด เพื่อเปลี่ยนระดับแรงดันของสัญญาณกระดิ่งให้เป็นแรงดันไฟตรงโดยมี C2 และ ZD1 ทำหน้าที่รักษาระดับแรงดันให้คงที่ โปรแกรมตอบรับอัตโนมัติทำงานวงจรนับจะเป็นรหัสไบนารี่ขนาด 4 บิต

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากรูปที่ 3.3 การทำงานเริ่มเมื่อมีสัญญาณกระดิ่งเข้ามา วงจรบริดจะทำการเปลี่ยน ระดับแรงดันของสัญญาณกระดิ่งให้เป็นระดับแรงดันไฟตรง เมื่อแรงดันซีกบวกเข้ามาจะทำให้ ออปโต้ทรานซิสเตอร์ทำงานเปลี่ยนระดับแรงดันจาก 5 โวลท์ เป็น 0.3 โวลท์ (ขอบขาลง) ส่งผลให้วงจรกำเนิดสัญญาณพัลส์ทำงาน โปรแกรม

3.3 วงจรยกและวางหูโทรศัพท์

วงจรยกและวางหูโทรศัพท์แบ่งเป็น 3 ชุด โดยแต่ละชุดทำหน้าที่แตกต่างกัน คือในชุดที่ 1 ทำหน้าที่ยกหูโทรศัพท์อัตโนมัติ ชุดที่ 2 ทำหน้าที่ตัดต่อวงจรตรวจสอบสัญญาณตอบกลับ และชุดที่ 3 ทำหน้าที่ตัดต่อระหว่างไมโครโฟนกับลำโพง



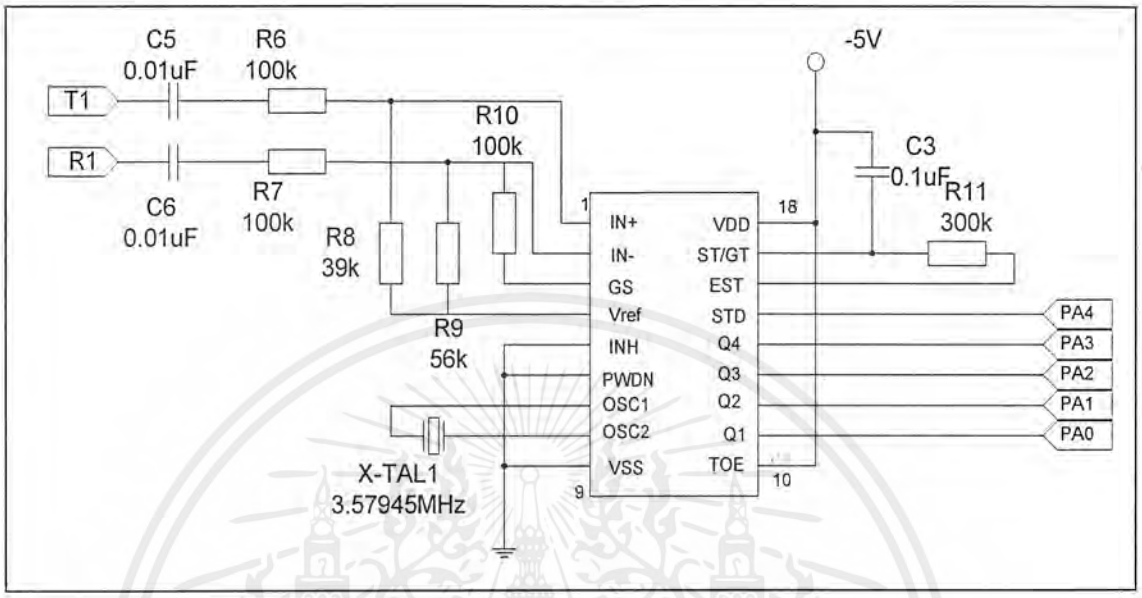
รูปที่ 3.4 วงจรควบคุมรีเลย์

จากรูป 3.4 ในสภาวะเริ่มแรก หน้าสัมผัสของรีเลย์จะอยู่ในสถานะที่ไม่ต่อกับวงจรไฮบริดจ์ คืออยู่ในสถาวะออฟ เมื่อมีการเรียกเข้าสามารถตรวจจับสัญญานกระดิ่ง คอมพิวเตอร์จะส่งสัญญาณจากคอมพิวเตอร์มากระตุ้นที่ขาเบสของแตะกับขาอีกด้านหนึ่ง เป็นการเปลี่ยนแปลงการทำงานทรานซิสเตอร์ Q1 ทำให้ขาคอลเลคเตอร์มีแรงดันตกคร่อมประมาณ 0 โวลท์ ทรานซิสเตอร์อยู่ในสภาวะของสวิทช์ออน ทำให้หน้าสัมผัสของรีเลย์ก็จะถูกสลับต่อเข้ากับวงจรไฮบริด ต่อวงจรเข้ากับส่วนของคอมพิวเตอร์ เมื่อมีการเลิกการติดต่อคอมพิวเตอร์ PB0 จะมีสถานะลอจิก " 0 " ทำให้หน้าสัมผัสของรีเลย์ก็จะถูกสลับกลับมาที่เดิม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.4 วงจรถอดรหัสสัญญาณความถี่คู่ดีทีเอ็มเอฟ (DTMF Decoder)

วงจรถอดรหัสสัญญาณความถี่คู่นี้ทำหน้าที่ถอดรหัสสัญญาณความถี่คู่ ให้เป็นรหัส ไบนารี่ขนาด 4 ก่อนส่งไปยังคอมพิวเตอร์



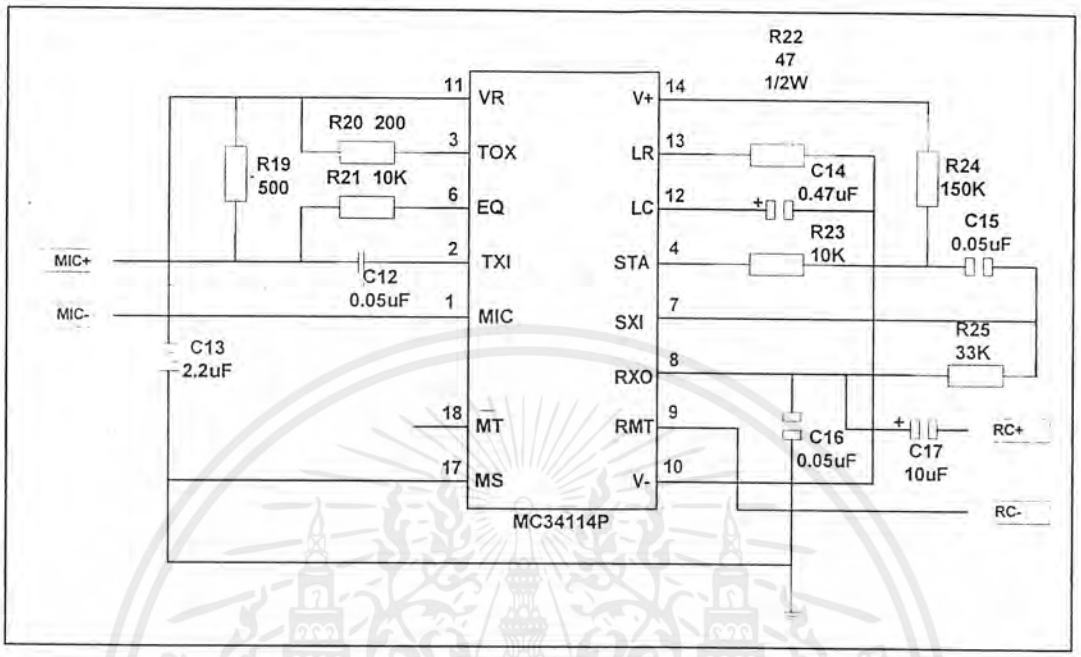
รูปที่ 3.5 วงจรถอดรหัสความถี่คู่ดีทีเอ็มเอฟ

จากรูปที่ 3.5 เมื่อมีสัญญาณความถี่คู่ เข้ามา C5 และ C6 จะเป็นตัวส่งผ่านสัญญาณ แล้วส่งผ่าน R6 และ R7 มาเข้าที่ขาอินพุตของไอซี MT8870 ซึ่งเป็นอุปกรณ์ที่สำคัญของ วงจร ส่วน R6 กับ R10 เป็นตัวกำหนดอัตราขยายของออปแอมป์ที่อยู่ภายในไอซี ให้มีอัตราการ ขยายเท่ากับ 1 ส่วน R 8 กับ R9 เป็นตัวกำหนดแรงดันอ้างอิง และขา OSC1 และ OSC2 จะต่อกับแร่คริสตอลขนาด 3.57954 MHz เพื่อกำเนิดความถี่เป็นสัญญาณนาฬิกาให้กับ ไอซี และวงจรจะมีการตรวจสอบ ช่วงความถี่ที่เข้ามาว่ามีระยะเวลาตามที่กำหนดหรือไม่โดยสังเกตจากระยะเวลาการกดปุ่มโทรศัพท์ ซึ่งจะต้องมีช่วงเวลานานพอสมควร คือให้ระยะเวลานานเท่ากัน หรือมากกว่าช่วงเวลาที่ตั้งไว้จึงจะยอมรับ และถือว่าสัญญาณนั้นถูกต้อง โดยที่ขา EST จะเป็น 1 นานใกล้เคียงกับระยะเวลาที่มีความถี่คู่ ทำให้แรงดันที่ขา ST/GT สูงขึ้น ตัวเก็บประจุ C ซึ่งเก็บประจุ เต็มก็จะคายประจุออกมา ทำให้แรงดันที่ขา ST/GT สูงขึ้นจนถึงค่าเทรสโฮลด์ วงจรถอดรหัสจึงจะถอดรหัสออกมาเป็นตัวเลขขนาด 4 บิท ทางขา Q1, Q2, Q3 และ Q4 แต่ถ้าเปรียบเทียบกันแล้วช่วง เวลาน้อยกว่าที่ตั้งไว้ก็จะไม่มีการถอดรหัสเป็นตัวเลขขนาด 4 บิทออกไป ซึ่งช่วงเวลานี้เราสามารถตั้งได้โดยการกำหนดค่าของ R11 และ C3 ส่วนขา TOE นั้นจะทำงานคล้าย ๆ กับเป็นขา Enable คือ ในสภาวะปกติไม่มีสัญญาณเอาต์พุตจะทำให้ที่ขาเอาต์พุตนี้มีค่าอิมพิแดนซ์สูงมาก ทำให้สัญญาณ ต่าง ๆ ไม่สามารถผ่านขา Q ได้ แต่เมื่อมีสัญญาณเอาต์พุตมาจะมีอิมพิแดนซ์ต่ำลง แล้วจึงส่งสัญญาณ เอาต์พุตออกไปได้ สำหรับขา STD ในวงจรนี้ก็เป็นขาเอาต์พุตอีกขาหนึ่ง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.5 วงจรไฮบริด

วงจรไฮบริดทำหน้าที่เป็นวงจรต่อผ่านสัญญาณจากปากพูดไปยังหูฟัง และต่อผ่านสัญญาณจากคู่สายเข้ายังวงจรโทรศัพท์



รูปที่ 3.6 วงจรไฮบริด

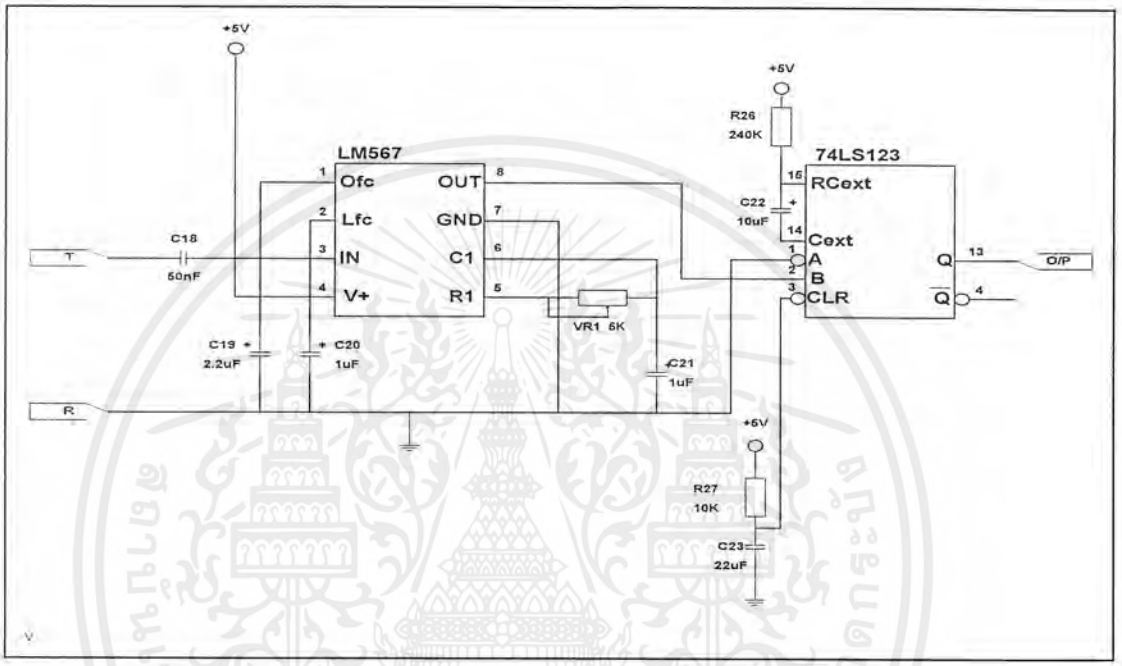
จากรูปที่ 3.6 จะใช้ไอซี MC341114P เป็นตัวหลักของวงจร โดยสัญญาณจากไมโครโฟนจะต่อเข้าที่ขา MIC ซึ่งมีแรงดัน 1.2 โวลท์ จากขา VR มาทำการไบอัสให้กับไมโครโฟน เพื่อขยายสัญญาณจากไมโครโฟนให้ดังมากขึ้น จากนั้นจะนำสัญญาณนี้ ส่งเข้าที่ขา TXI ส่งเข้าไปทำการขยายแล้วส่งไปยังชุดปรับแต่งสัญญาณ (Equalization) เพื่อจัดระดับของสัญญาณแล้วป้อนกลับมาที่อินพุตขา TXI เพื่อขยายสัญญาณแรงขึ้นแล้วส่งต่อไปยังชุดไซด์โทน (Sidetone Amp) เพื่อส่ง สัญญาณออกยังขา STA แล้วป้อนกลับมาที่ขา RXI จากนั้นทำการขยายและส่งออกทางขา RXO ผ่าน C17 ออกทางลำโพง และในขณะเดียวกันสัญญาณก็จะถูกส่งต่อไปยังคู่สายโทรศัพท์ ทำให้ทั้ง ตัวเราเอง และผู้ที่เราสนทนาด้วยได้ยินเสียงที่เราพูดออกไป และในทางกลับกันเมื่อคู่สนทนาของเรานั้นพูดมาบ้าง สัญญาณเสียงของเขาก็จะถูกส่งผ่านชุดไดโอดบริดเข้ามายังขา RXI ทำการขยายสัญญาณเสียงแล้วส่งออกทางขา RXO ผ่าน C17 คัปปลิ้งออกลำโพง ทำให้เราได้ยินเสียงของผู้ที่เรา สนทนาด้วย และในวงจรนี้จะมีขา MS เป็นขาที่ให้เลือกสัญญาณที่เป็นสัญญาณพัลส์และโทน ซึ่งใน วงจรนี้เลือกใช้สัญญาณความถี่เสียง และที่ขา MT จะส่งสัญญาณมาทำการเปรียบเทียบ ถ้าสัญญาณมี ค่าแรงดันน้อยกว่า 1 V ก็จะคิดเป็นสภาวะ 0 และตัดสัญญาณเสียงส่งสัญญาณความถี่เสียงออกไป แต่ถ้ามีแรงดันของสัญญาณมากกว่า VDD - 0.3 โวลท์ ก็จะคิดเป็นสภาวะ 1 และตัดสัญญาณโทนส่งสัญญาณเสียงออกมาที่ขา RMT เพื่อส่งออกไปทางลำโพง และออกไปยังคู่สายโทรศัพท์ และหากต้องการจะโทรศัพท์ออก สัญญาณความถี่คู่ที่กดมานั้นจะ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ถูกป้อนเข้ามาที่ขา TXI สัญญาณนั้น ๆ จะถูกขยายแล้วส่งออกไปที่ขา V+ เพื่อส่งออกไปยังคู่สายต่อไป โดยผ่านไดโอดบริดออกไป

3.6 วงจรตรวจสอบสัญญาณความถี่เสียง (Check Tone)

วงจรตรวจสอบสัญญาณความถี่เสียง ทำหน้าที่ตรวจสอบสัญญาณโทนภายในคู่สาย เพื่อบอกให้ทราบว่าเป็นสัญญาณให้หมุนเลขหมาย สัญญาณไม่ว่าง หรือ สัญญาณตอบกลับ



รูปที่ 3.7 วงจรตรวจสอบสัญญาณความถี่เสียง

จากรูป 3.7 ไอซี LM567 เป็นหัวใจหลักของวงจรนี้ ทำหน้าที่เป็นวงจรโทน-ดีโค้ดเดอร์ ซึ่งจะทำงานในลักษณะตรงกันข้ามกับสัญญาณความถี่เสียงที่เข้ามา คือ ถ้าไม่มีสัญญาณความถี่เสียงเข้ามาทางอินพุตขา 3 เอาต์พุตที่ขา 8 จะเป็นสภาวะ 1 แต่ถ้ามีสัญญาณความถี่เสียงเข้ามาทางด้านอินพุต ก็จะทำให้ทางด้านเอาต์พุตมีค่าเป็นสภาวะ 0 ส่วนไอซี 74LS123 เป็นวงจรโมโนสเตเบิลมัลติไวเบรเตอร์ที่มีขาอินพุต 3 ขา ซึ่งจะผ่านเข้ามาเป็นอินพุตของแอนเกท ขา 1 และขา 2 ที่อยู่ ภายในตัวไอซี จากนั้นก็จะส่งค่าเอาต์พุตจากแอนเกท ออกไปเป็นเอาต์พุตของวงจรที่ขา Q (ขา 13) ของไอซี

เมื่อมีสัญญาณความถี่เสียงเข้ามา C18 จะทำหน้าที่คัปปลิ้งสัญญาณความถี่เสียง ไปยัง ขา 3 ของไอซี LM567 และผ่านเข้าไปยังภาคเฟสล็อคลูปที่อยู่ภายในตัวไอซี โดยมี VR1 และ C21 เป็นตัวกำหนดความถี่ของวงจร และ C20 ทำหน้าที่กรองความถี่ แบบความถี่ต่ำผ่าน และในขณะเดียวกันสัญญาณความถี่เสียง อีกส่วนหนึ่งจะแยกไปเข้ายังภาคแยกเฟส 90 องศา (Quadrature Phase Detector) ที่อยู่ภายในตัวไอซี ทำการเปรียบเทียบความถี่และเฟสของสัญญาณอินพุต จากนั้นส่งสัญญาณเอาต์พุตไปทำการขยายและส่งออกเป็นสภาวะ 0 ทางขาเอาต์พุตที่ขา 8 จากนั้นต่อเข้ายังวงจรโมโนสเตเบิลมัลติ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

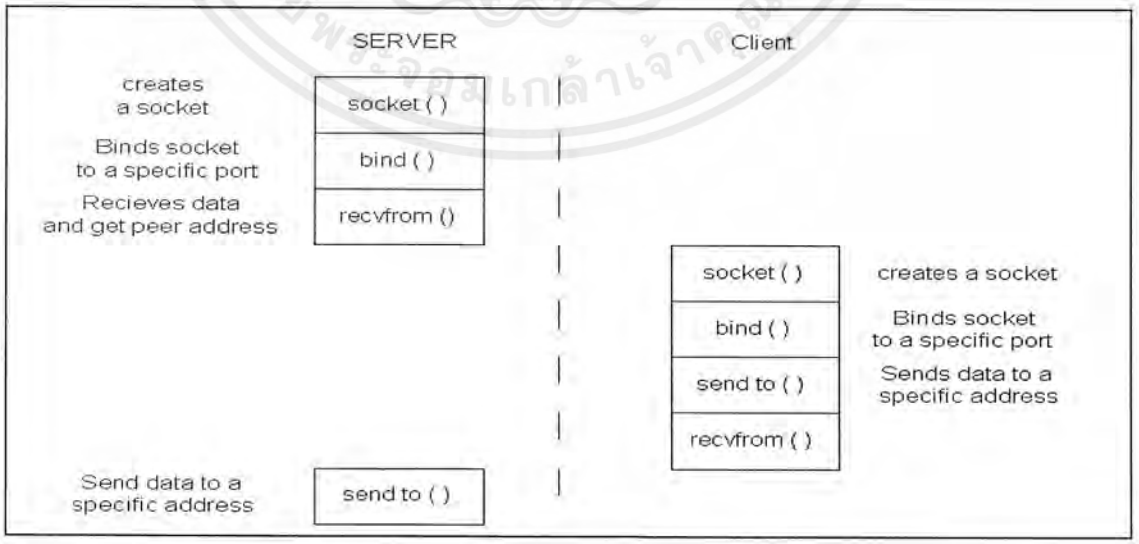
ไวเบรเตอร์ ซึ่งมีขาอินพุต 3 ขา คือขา 1, 2, และ 3 ซึ่งที่ขา 2 จะได้สัญญาณเป็นสภาวะ 0 ส่วนขา 1 และขา 3 จะได้สัญญาณเป็นสภาวะ 0 เช่นกัน โดยที่ขา 1 ได้สภาวะ 0 จากกราวด์ของวงจร และที่ขา 3 ได้สภาวะ 0 จากชุดปรับตั้งแรงดันใหม่ (ช่วงจ่ายไฟป้อนเข้าวงจร) และสัญญาณสภาวะ 0 ทั้ง 2 ขานี้จะผ่านน็อตเกท ภายในตัวไอซี สัญญาณจึงเปลี่ยนจากสภาวะ 0 เป็นสภาวะ 1 อินพุตของแอนเกท จึงเป็นสภาวะ 101 ทำให้เอาต์พุตของแอนเกท เป็นสภาวะ 0 ส่งออกไปยังขา Q (ขา 13) ซึ่งเป็นเอาต์พุตของตัวไอซี (โดยปกติแล้ววงจรนี้ที่ขา 1 และ ขา 3 ของแอนเกท จะเป็นสภาวะ 1 อยู่ตลอดเวลาที่มีการจ่ายไฟป้อนเข้าวงจร ดังนั้น มีเพียงขา 2 เท่านั้นที่จะเป็นตัวเปลี่ยนแปลง และมีผลต่อเอาต์พุตของวงจร) ซึ่งลักษณะการทำงานของวงจรจะ แตกต่างกันไปตามสัญญาณความถี่เสียงที่เข้ามา

3.7 การออกแบบในส่วนของซอฟท์แวร์ (Software)

การออกแบบของซอฟท์แวร์ในเวลานี้จะทำในส่วนของการติดต่อกันระหว่างเครื่องคอมพิวเตอร์สองเครื่อง หลังจากทำการเชื่อมต่อได้แล้วจะทดลองทำการติดต่อกัน โดยการส่งข้อความคุยกัน จากนั้นจึงจะพัฒนาต่อไปโดยเปลี่ยนจากการติดต่อโดยใช้ข้อความเป็นการติดต่อโดยเสียง ในส่วนของซอฟต์แวร์ควบคุมการทำงานของอินเตอร์เฟสการ์ด และซอฟต์แวร์ควบคุมการทำงานของการสื่อสารแพ็กเกตเสียงบนเครือข่ายคอมพิวเตอร์ ซึ่งในส่วนของโปรแกรมมิ่งที่ใช้ในการเขียน จะใช้โปรแกรม Visual C++ เวอร์ชัน 6.0

เป็นโปรแกรมที่สามารถเข้าไปควบคุมถึงฮาร์ดแวร์ระดับล่างๆได้ดี เพราะในโครงงานนี้จำเป็นต้องควบคุมการอินเตอร์เฟส กับอุปกรณ์ภายนอก และมีการควบคุมการทำงานของซาวด์การ์ด ซึ่งได้เสนอฟังก์ชัน ในการติดต่อ API ของวินโดวส์ไปแล้วในบทที่ 2

ในส่วนนี้จะทำการออกแบบเพื่อติดต่อสื่อสารกันระหว่างคอมพิวเตอร์ โดยใช้ UDP โปรโตคอล โดยมีลักษณะการติดต่อดังนี้



3.8 รูปแสดงขั้นตอนการสื่อสารของ UDP โปรโตคอล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.7.1 ขั้นตอนการทำงานของโปรแกรมส่วนของด้านส่ง (sender)

1. เตรียมซ็อกเกต (socket) เพื่อติดต่อกับเครื่องคอมพิวเตอร์ปลายทาง (client) เพื่อรอการติดต่อ

2. กำหนดฟอร์แมทของไฟล์และค่าต่างๆ เช่น ค่าแซมปลิ้ง กำหนดแชนแนล บิทเรท (bit rate) ในการส่งซึ่งเป็นไปตามฟังก์ชันของซาวด์การ์ด ในการอัดเสียง

3. เมื่อเตรียมอุปกรณ์เรียบร้อย จะกำหนดฟังก์ชันเพื่อเตรียมบัฟเฟอร์ (Buffer) คือหน่วยความจำส่วนที่จะเก็บข้อมูลเสียงลงไปในเครื่องคอมพิวเตอร์ ในที่นี้เรากำหนดไว้ 8 ตัว ตามขนาดของฟอร์แมทที่เราต้องการบันทึก ซึ่งการเซ็ตบัฟเฟอร์ขึ้นอยู่กับการบันทึกในฟอร์แมทต่าง ๆ

4. เตรียมเฮดเดอร์ของบัฟเฟอร์ เฮดเดอร์ของบัฟเฟอร์จะทำหน้าที่เก็บรูปแบบของๆไฟล์เสียง ซึ่งมีค่าของอุปกรณ์ในการแซมปลิ้ง, อัตราการส่ง ก่อนที่จะทำการบันทึกให้พร้อมสำหรับการบันทึก เพื่อเตรียมการบันทึกข้อมูล โดยเริ่มบันทึกลงในบัฟเฟอร์ตัวแรกก่อน เมื่อเต็มก็จะเปลี่ยนเป็นตัวที่สอง และตัวต่อไป

5. เมื่อข้อมูลในบัฟเฟอร์ตัวที่หนึ่งเต็มก็จะส่งไปที่เครื่องปลายทาง ผ่านทางแบบ UDP winsock

6. เคลียร์เฮดเดอร์ของบัฟเฟอร์ที่ใช้งานแล้ว

7. ทำการวนลูปโปรแกรมนี้จนกระทั่งผู้ใช้ยกเลิกการติดต่อการติดต่อกับปลายทาง

3.7.2 ขั้นตอนการทำงานของโปรแกรมด้านรับ (Reciever)

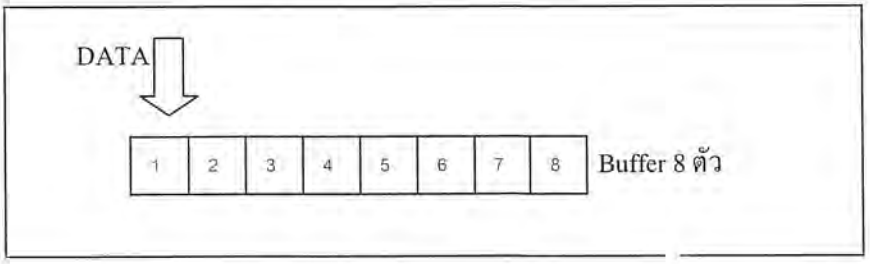
1. เตรียมซ็อกเกตของเครื่องคอมพิวเตอร์ปลายทางแบบ UDP winsock เพื่อรอรับข้อมูลจากเซิร์ฟเวอร์

2. กำหนดฟอร์แมทของเวปที่จะรับเข้ามาให้ตรงกับทางด้านส่ง กำหนดอัตราการรับข้อมูล

3. ถามไปยังวินโดวส์ ว่าซาวด์การ์ดที่อยู่ในเครื่องซัพพอร์ทการเล่นไฟล์เวฟในฟอร์แมทที่ด้านส่งกำหนดหรือไม่

4. ถ้าซัพพอร์ทกับฟอร์แมทไฟล์เวฟ ก็จะเปิดช่องสัญญาณติดต่อกับซาวด์การ์ด และเตรียมบัฟเฟอร์ด้านรับ เพื่อรับข้อมูลจาก socket ที่กำหนดไว้ โดยขนาดของบัฟเฟอร์จะขึ้นอยู่กับขนาดของรูปคลื่นที่รับมาจากเซิร์ฟเวอร์

5. รับข้อมูลที่ได้จากเซิร์ฟเวอร์ผ่าน socket เข้ามาเก็บลงในบัฟเฟอร์ทั้ง 8 ตัว โดยใช้หลักการเดียวกันกับทางด้านส่ง



รูปที่ 3.9 แสดงบัฟเฟอร์ในการรับข้อมูล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6. เตรียมเฮดเดอร์ของบัฟเฟอร์ เพื่อให้วินโดวส์เตรียมเปิดไฟล์เวฟขึ้นมา (play back)

7. ส่งข้อมูลจากบัฟเฟอร์ผ่านช่องสัญญาณที่เตรียมไว้ไปยังซาวด์การ์ด เพื่อเล่นกลับ (play back)

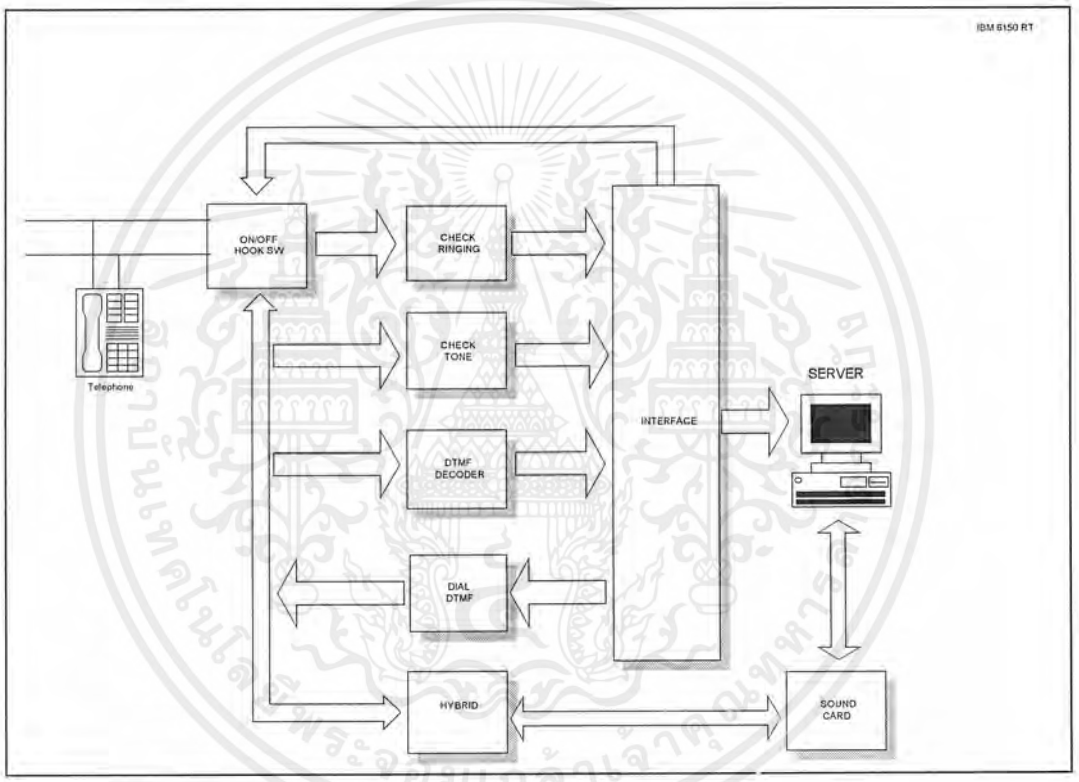
8. เคลียร์เฮดเดอร์ของบัฟเฟอร์ ที่มีการเล่นกลับออกไปแล้ว ทำการวนลูปไปสเตปที่ 5 คือ เริ่มนำข้อมูลจาก socket มาเก็บไว้ที่บัฟเฟอร์ใหม่ วนไปเรื่อย ๆ จนกรทั่งหยุดการสื่อสาร

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 4
การทดลอง และผลการทดลอง

จากการออกแบบวงจรในส่วนของฮาร์ดแวร์ และโปรแกรมการติดต่อสื่อสารจะทำการทดลองแยกกเป็น 2 ส่วน คือ ส่วนที่หนึ่งจะเป็นของตัวการ์ดอินเตอร์เฟสกับวงจรดีเทคสัญญาณของโทรศัพท์ต่างๆ และส่วนของโปรแกรมควบคุมการทำงานการติดต่อสื่อสารและส่วนของโปรแกรมควบคุมการ์ดอินเตอร์เฟส

4.1 ฮาร์ดแวร์ (Hardware)

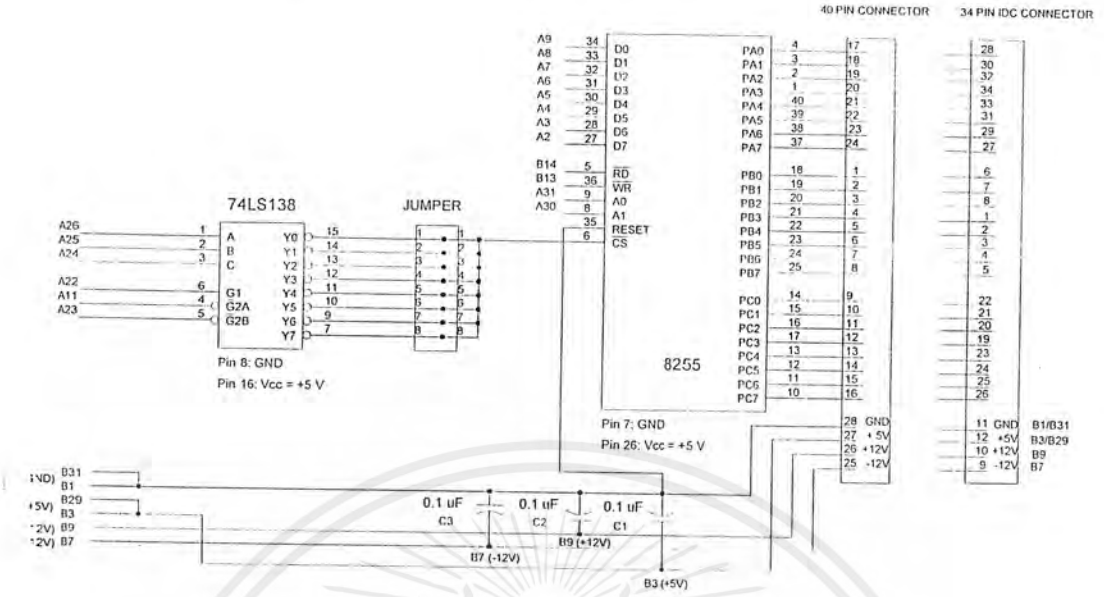


รูปที่ 4.1 Block diagram card interface

จากบล็อกไดอะแกรมจะเป็นโครงสร้างการทำงานของ card interface ซึ่งประกอบด้วยวงจรย่อย ๆ 6 ส่วน ซึ่งจะแยกการทำการทดลอง และเขียนโปรแกรมควบคุมในส่วนของ IC 8255 ที่เป็นตัวอินเตอร์เฟสกับคอมพิวเตอร์ (รูปที่ 4.1)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.1.1 ชุดอินเตอร์เฟส (Interface card)



รูปที่ 4.2 วงจรอินเตอร์เฟส

ขั้นตอนการทดลอง

1. ประกอบวงจรตามรูปที่ 4.2 เข้ากับการ์ด ISA BUS ซึ่งเป็นการ์ดอินเตอร์เฟสเอนกประสงค์ควรตรวจเช็คบัสของข้อมูลของคอมพิวเตอร์ให้ตรงกับการ์ดอินเตอร์เฟส โดยเฉพาะขาไฟของคอมพิวเตอร์บนการ์ดกับสล็อตบนคอมพิวเตอร์ เพราะอาจจะทำให้คอมพิวเตอร์เสียหาย

2. หลังจากตรวจสอบความเรียบร้อยแล้ว ทำการติดตั้ง jumper ไปที่ตำแหน่ง port 240H เนื่องจากการทดลองนี้ใช้กับคอมพิวเตอร์ที่มี Address port ว่างที่ port 240H

3. นำการ์ดเสียบลงบนสล็อตของคอมพิวเตอร์

4. ในการทดสอบวงจรอินเตอร์เฟส (Interface) จะทำการทดสอบชุดนี้ด้วยโปรแกรมภาษาเบสิกบนดอส เพราะสามารถรันกับเครื่องรุ่นเก่าได้ ซึ่งการทดสอบในครั้งแรกจะใช้โปรแกรมภาษาเบสิกแล้วสามารถทำงานได้จะเปลี่ยนเป็นเครื่องที่ประสิทธิภาพสูงขึ้น

5 . เปิดคอมพิวเตอร์เข้าโปรแกรมภาษาเบสิกแล้วพิมพ์โปรแกรมดังต่อไปนี้

```
100   REM 8255  PP1 SET  PORT  A,B,C  TO OUTPUT
110   BASEADDR  =  576: REM  240H(JUMPER AT NUMBER 5)
120   PORT A  =  BASEADDR
130   CNTRL  =  BASEADDR + 3
140   OUT  CNTRL,128
```

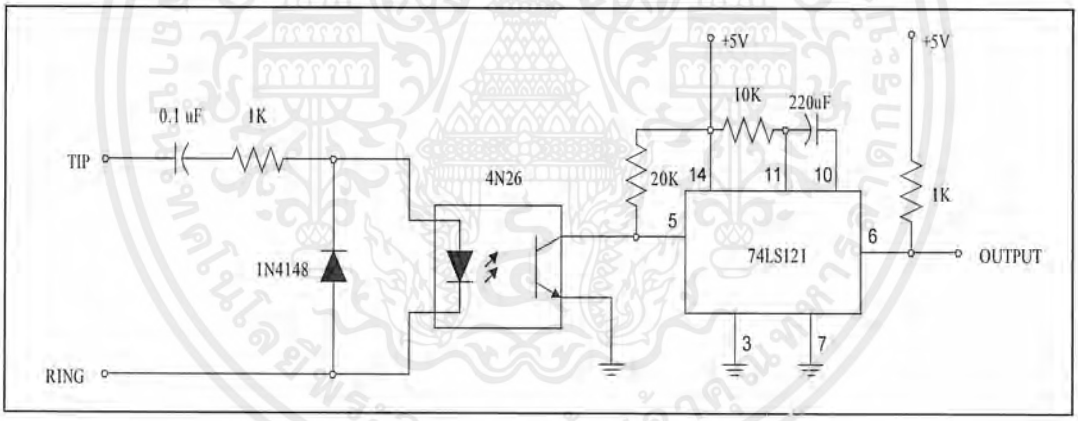
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
150     FOR SGNAL = 0 TO 255
160     OUT PORTA,SGNAL: PRINT " DECIMAL = "
        ; SGNAL
170     FOR DELAY = 1 TO 500 : NEXTDELAY
180     NEXT SGNAL
900     END
```

6. ได้ผลดังต่อไปนี้

ตัวโปรแกรมออกแบบเพื่อควบคุมการ์ดอินเตอร์เฟส (interface card) โดยจะเซตให้ทุกพอร์ตบน IC 8255 มีสถานะเป็นเอาท์พุตและทำการให้พอร์ต A นับเลขไบนารี 8 บิท เพื่อให้พอร์ต A แสดงค่าตั้งแต่ 0 – 255 ถ้า LED มาต่อ จะแสดงสถานะได้ดีที่สุด แต่การทดลองใช้มิเตอร์วัดสถานะแทน

4.1.2 วงจรตรวจสอบสัญญาณกระดิ่ง

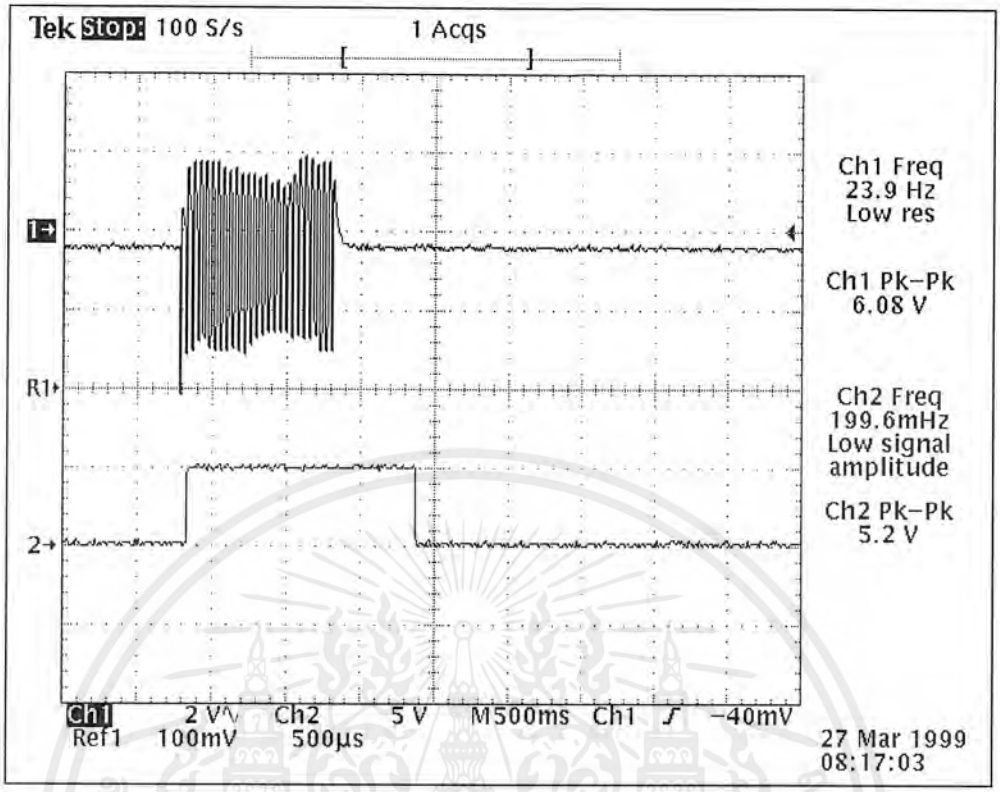


รูปที่ 4.3 วงจรตรวจสอบสัญญาณกระดิ่ง

ขั้นตอนการทดลอง

1. ประกอบวงจร (รูปที่ 4.3) ตรวจสอบอุปกรณ์และระบบไฟ
2. นำวงจรมาต่อเข้ากับคู่สายโทรศัพท์ ทำการโทรศัพท์เข้ามาที่คู่สายที่ต่อกับวงจร
3. นำมิเตอร์มาวัด ที่เอาท์พุตของ IC 74LS121 ที่ขา 6 จะต้องมีสถานะเป็นลอจิก "1" เมื่อมีสัญญาณกระดิ่งเข้ามาที่คู่สาย
4. นำวงจรมาวัดสัญญาณที่สโคบ เพื่อวัดรูปสัญญาณเปรียบเทียบกันระหว่างอินพุทที่เป็นสัญญาณกระดิ่งกับเอาท์พุตที่เป็นพัลส์ที่ OUTPUT ของวงจรโมโนสเตเบิ้ล IC74LS121 ซึ่งผลการทดลองจะนำไปทำงานร่วมกับ การ์ด อินเตอร์เฟส 8255 (รูปที่ 4.4)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูป 4.4 แสดงผลการทดลองของวงจรตรวจจับสัญญานกระดิ่ง

5. จากรูปที่ 4.4 จะเป็นการทดลองทางด้านตัวฮาร์ดแวร์อย่างเดียว ขั้นตอนนี้จะผนวกให้ส่วนนี้ข้ากับการ์ดอินเตอร์เฟส 8255 โดยเราจะเขียนโปรแกรมควบคุมการทำงาน โดยจะคอยเช็คสัญญานที่พอร์ตที่ $242 หรือพอร์ต C ของ 8255 โดยกำหนดว่าถ้ามีสัญญานกระดิ่งมา จะเขียนโปรแกรมรอรับดังนี้

```
// STARTThread message handlers

int STARTThread::Run()
  {
  int  i=0,j;
  char text[100];
  BYTE input0,input1,select;
  static CWnd *pStatusbar=AfxGetMainWnd()->
GetDescendantWindow(AFX_IDW_STATUS_BAR);

  m_pOwner->Outport(0x241,0x00);
  m_pOwner->Outport(0x243,0x99);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

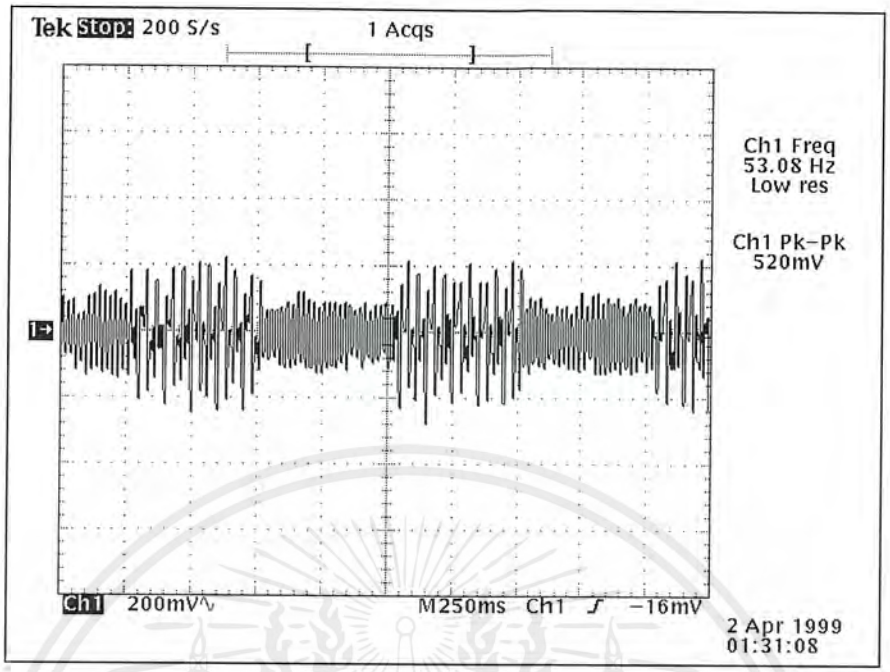
```
for(;;)
    {
    input0=m_pOwner->Inport(0x242);
    if(input0==255)
      {
      select=0;

for(j=0;j<1;j++)
        {
        m_pOwner->Outport(0x241,0x01);
        sprintf(text,"Waiting input digit : %d",j+1);
        AfxGetMainWnd()->SetWindowText(text);
```

จากโปรแกรมข้างต้น จะรอรับค่าที่พอร์ต $242 ถ้ามีค่าเท่ากับ 255 โปรแกรมจะทำงานสั่งให้ส่งค่าลอจิก 1 ไปที่บิทที่ 1 ของพอร์ตเบอร์ $241 เพื่อทำให้วงจรควบคุมรีเลย์ทำงาน

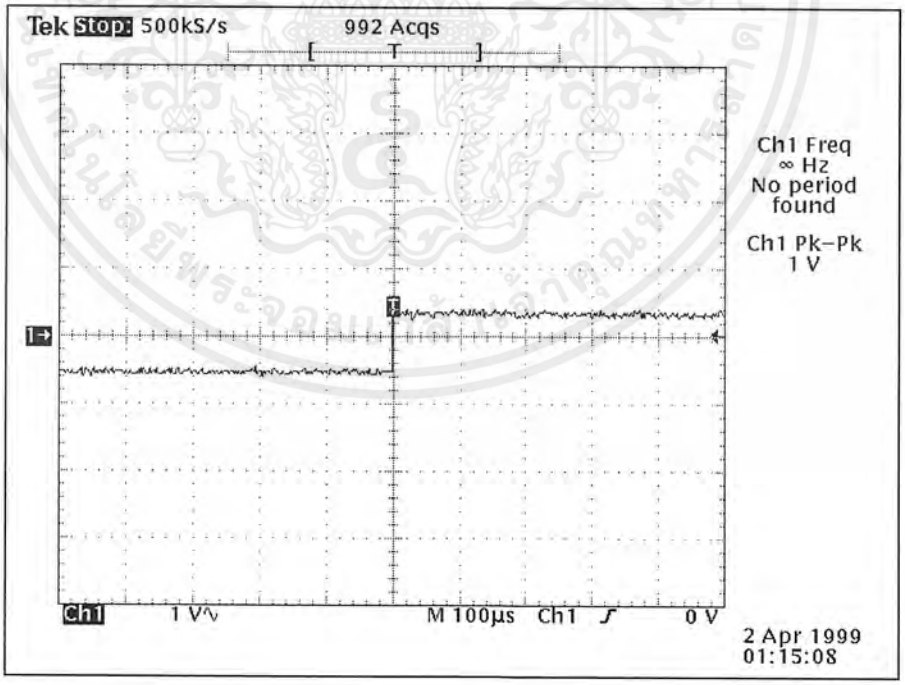
4.1.3 การทดลองส่วนวงจรตรวจจับสัญญาณเสียง

จากการออกแบบในบทที่ 3 ตัววงจรจะมีไอซี LM 567 เป็นไอซีที่ทำหน้าที่เป็นโทนดีโค้ดเดอร์ความถี่เสียง โดยจะมีวงจรเฟสล็อกหลู่ ซึ่งสามารถคำนวณความถี่ที่ต้องการตรวจจับ ซึ่งสามารถคำนวณได้ในบทที่ 3 ในการออกแบบในโครงงานนี้เรากำหนดความถี่ที่ 400 เฮิร์ทซ (Hz) คือเป็นวงจรตรวจจับสัญญาณไม่ว่าง (BUSY TONE) และสัญญาณเรียกกลับ (RING BACK) ซึ่งเป็นสัญญาณความถี่ 400 เฮิร์ทซ โดยในการทดลองจะประกอบวงจรดังรูปที่ 4.5 แล้วป้อนอินพุทเป็นคู่สายโทรศัพท์ (SUBSCRIBER LINE) ต้องขนานกับเครื่องโทรศัพท์ แล้วทำการทดลองยกหูโทรศัพท์ขึ้นทั้ง 2 ด้าน จะพบว่ามีสัญญาณไม่ว่างเกิดขึ้น นำออสซิลโลสโคบ (OSCILOSCOPE) วัดเอาท์พุตที่ขา 8 ของเอาท์พุตเป็นลักษณะพัลซ์ (SQUARE WAVE) โดยจะเป็นลอจิก " 1 " เมื่อมีความถี่ 400 เฮิร์ทซเข้ามาและจะเป็นลอจิก " 0 " แล้วนำค่าที่ได้มาป้อนให้กับวงจรโมโนสเตเบิ้ลของ IC 74LS121 เมื่อไม่มีสัญญาณหรือไม่ใช่ความถี่ 400 เฮิร์ทซเข้ามาและทำการทดลองสัญญาณเรียกกลับ (RING BACK) โดยทำการขนานวงจรเข้ากับเครื่องโทรศัพท์ แล้วกดหมายเลขผู้รับวัดสัญญาณ ขณะที่ปลายทางยังไม่รับสายซึ่งเอาท์พุตมีลักษณะเช่นเดียวกับสัญญาณไม่ว่าง (BUSY TONE) โดยจะได้สัญญาณการทดลอง และสัญญาณเอาท์พุทของโมโนสเตเบิ้ลเพื่อป้อนให้กับวงจรอินเตอร์เฟส (รูปที 4.5)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูป 4.5 แสดงสัญญานไม่ว่าง (BUSY TONE)



รูปที่ 4.6 แสดงรูปเอาท์พุทของโมโนสเบิ้ล

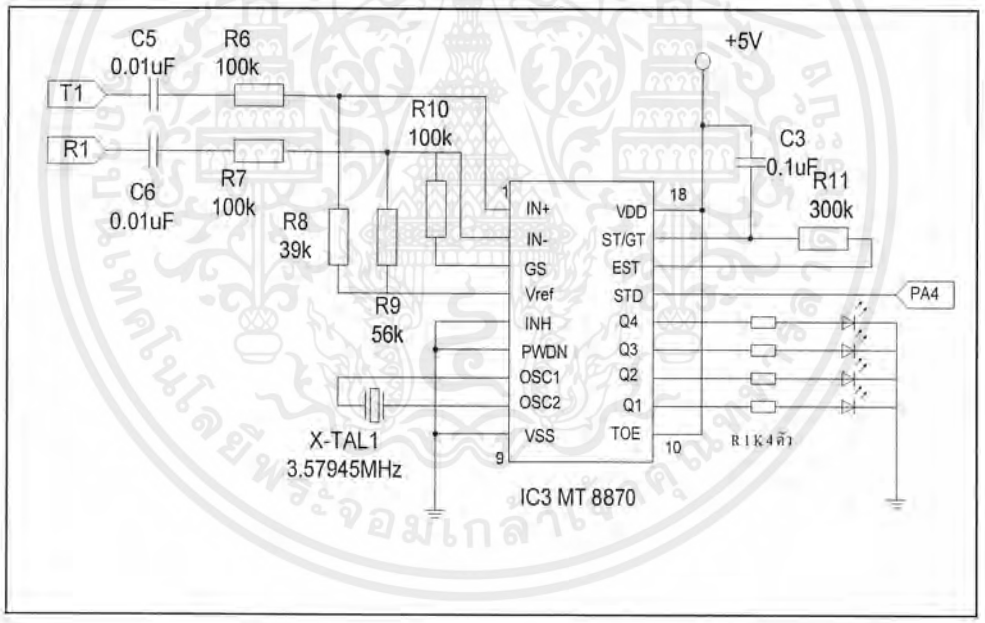
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.1.4 ผลการทดลองวงจรดีโคดเดอร์สัญญาณ DTMF

ในส่วนของภาคดีโค้ดเดอร์สัญญาณ DTMF จะใช้ไอซีเบอร์ MT 8770 ซึ่งเป็นไอซีทำหน้าที่แปลงสัญญาณความถี่คู่ (DTMF) ออกมาเป็นเอาท์พุตที่เป็นสัญญาณไบนารี 4 บิท เพื่อนำไปเก็บไว้ในตัวแปรในโปรแกรม ทำการทดลองโดยสองขั้น โดยการทดลองขั้นที่1 จะนำส่วนของวงจรมาทดสอบกับ LED ให้เห็นเอาท์พุตเป็นแสงของ LED โดยถ้าเอาท์พุตเป็นลอจิก " 1 " จะแสดงผลให้ LED การทดลอง จะนำวงจรมาขนานกับเครื่องรับโทรศัพท์แล้วทำการกดหมายเลขบนเครื่องโทรศัพท์แล้วไอซีจะทำการถอดรหัส DTMF ซึ่งเอาท์พุตจะเป็นเลขไบนารี่ (รูปที่ 4.6)

ขั้นตอนทำการทดลอง

1. ประกอบวงจรตามรูปที่ 4.5 พร้อมทั้งต่อขนานเข้ากับไลน์โทรศัพท์
2. ยกหูโทรศัพท์ที่ทดลองแล้วกดหมายเลข LED ต้องแสดงผลตามตารางที่ 4.1 เป็นเลขไบนารี่



รูปที่ 4.7 การวงจรทดสอบวงจรดีโค้ดเดอร์ DTMF

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

LOW FREQUENCY	HIGH FREQUENCY	NUMBER	Q_4	Q_3	$Q1_2$	Q_1
697	1209	1	0	0	0	1
697	1336	2	0	0	1	0
697	1877	3	0	0	1	1
770	1209	4	0	1	0	0
770	1336	5	0	1	0	1
770	1477	6	0	1	1	0
852	1209	7	0	1	1	1
852	1336	8	1	0	0	0
852	1477	9	1	0	0	1
941	1336	0	1	0	1	0
941	1209	*	1	0	1	1
941	1477	#	1	1	0	0

ตาราง4.1 ผลการทดลองวงจรดีโค้ดเดอร์

จากตารางที่ 4.1 จะแสดงผลการถอดรหัส DTMF ส่วนอีกขั้นตอนจะทำการทดสอบกับโปรแกรมซึ่งส่วนของการทดลองกับโปรแกรมเพื่อรับค่าดีโคดเดอร์บนการ์ดอินเตอร์เฟสโดยโปรแกรม visual c++ โดยจะยกเฉพาะที่เป็นส่วนของการควบคุมมาทำการทดลองดังนี้

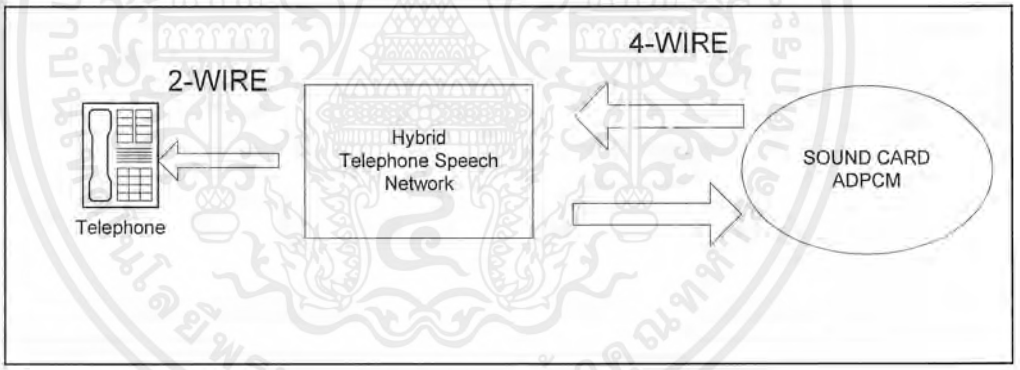
```
for(;;)
          {
          input1=m_pOwner->Inport(0x240);
          if(input1&0x0010==0x0010)
            {
            input1&=0x000F;
            if(j!=0) select+=((j*10)*input1);
            else     select+=input1;
            sprintf(text,"%d digit :
%d",j+1,select);
            pStatusbar->SetWindowText(text);
            break;
            }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากโปรแกรมข้างต้นนี้จะรอรับอินพุท โดยเราใช้พอร์ท $240 ของตัวรับเอาท์พุทจาก ไอซี 8870 ซึ่งต่ออยู่บนการ์ดอินเตอร์เฟส ทำการทดสอบโดยต่อขนานกับโทรศัพท์เครื่องที่ทำการทดลองแล้วทำการกดหมายเลขบนหน้าปัดของโทรศัพท์ ซึ่งโปรแกรมจะรอรับ ค่าที่ กดแล้วทำการเลขไบนารี่ 4 บิท เป็นเลขฐานสิบเช่น เมื่อทำการทดลองจะกดหมายเลข 1-9 ค่าหมายเลขจะแสดงที่ละตัวบนสเตตัสบาร์ของโปรแกรมตามที่เรากด

4.1.5 การทดลองวงจรไฮบริดแบบอิเล็กทรอนิกส์

ในส่วนของภาคไฮบริดจ์ เพื่อแปลงจากวงจร 2 คู่สาย เพื่อเป็นวงจร 4 คู่สายเพื่อทำให้สามารถติดต่อกันได้ แบบฟูลดูเพล็ก โดยในโทรศัพท์แบบเก่าจะใช้หม้อแปลงไฮบริดจ์ ซึ่งในการทดลองครั้งแรกๆ ในการทดลอง เราจะใช้หม้อแปลงไฮบริดจ์เป็นตัวแมชชิ่งแปลงวงจร 2 คู่สายเป็น 4 คู่สาย แต่พบปัญหาทางด้าน อิมพิแดนซ์และการควบคุมอัตราขยาย ทำให้ได้สัญญาณที่เบาและมีสัญญาณรบกวน แต่ทางเราได้ลองปรับเปลี่ยนมาใช้ไอซี MC 134114P ซึ่งเป็นไอซีที่ทำงานเป็นวงจรไฮบริดจ์แบบอิเล็คทรอนิกส์ ทำให้แก้ปัญหาของสัญญาณรบกวนและอัตราขยายได้ในการทดลองจะเป็นการส่งสัญญาณเสียงเป็นรูปแรงดันกระแสสลับ และต่อเข้ากับเครื่องคอมพิวเตอร์โดยผ่านซาวด์การ์ด สัญญาณเสียงที่ได้มีสัญญาณรบกวนค่อนข้างมากแต่เมื่อทำการจัดา ารปรับแต่งสายและวงจรทำให้สัญญาณเสียงที่ได้ชัดเจนขึ้น



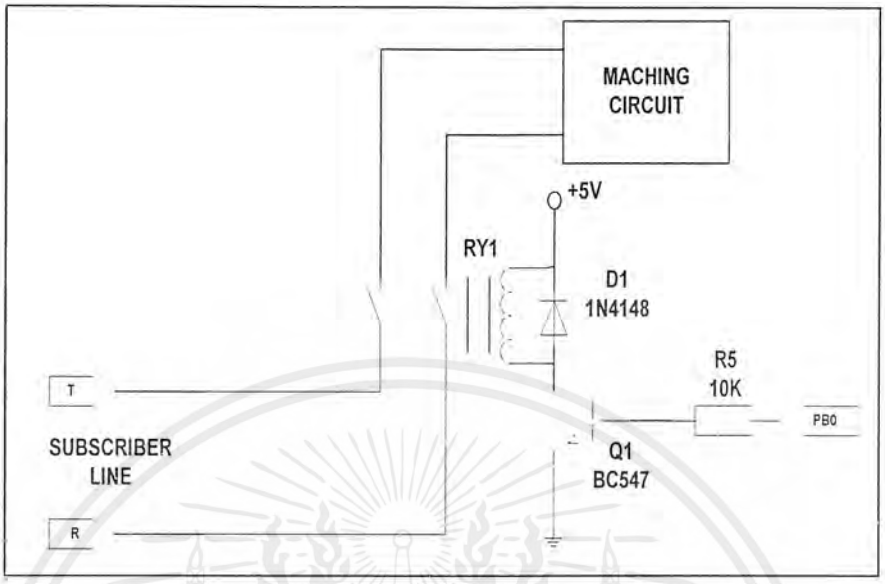
รูป 4.8 แสดงการต่อทดสอบวงจรภาคไฮบริดจ์

4.1.6 ชุดยกหูวางหูโทรศัพท์และวงจรแมชชิ่งโทรศัพท์

ขั้นตอนการทดลอง

1. ประกอบวงจรตามรูปที่ 4.7 พร้อมตรวจเช็คระบบไฟ
2. นำขาอินพุทของวงจรหรือขา B ของทรานซิสเตอร์ต่อเข้ากับชุดอินเตอร์เฟสการ์ด ที่ขา PB0 ของ IC 8255

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.9 ชุดยกหูวางหูโทรศัพท์อัตโนมัติ

3. ทดสอบโปรแกรมคุมดังนี้

```
        // STARTThread message handlers

int STARTThread::Run()
  {
  int  i=0,j;
  char text[100];
  BYTE input0,input1,select;
  static CWnd *pStatusbar=AfxGetMainWnd()->
GetDescendantWindow(AFX_IDW_STATUS_BAR);

  m_pOwner->Outport(0x243,0x99);

  m_pOwner->Outport(0x241,0x00);

}
```

4.ผลการทดลองเมื่อรันโปรแกรมแล้วจะทำให้มีสัญญาณลอจิก "1" มาที่มาขอ ทรานซิสเตอร์ เพื่อทำให้เป็นสวิตช์ on เพราะอยู่ในสถานะ saturate ทำให้ รีเลย์ทำงาน ต่อวงจรเข้ากับภาคไฮบริดจ์ของชุดอินเตอร์เฟสการ์ด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.2 การทดลองพัฒนาในส่วนของการรับส่งข้อมูลบนเครือข่ายคอมพิวเตอร์

ในการสร้างโปรแกรมสำหรับการทดลองรับส่งข้อมูลผ่านเครือข่ายคอมพิวเตอร์ เราได้ใช้โปรแกรม Visual C++ 6.0 เป็นโปรแกรมการพัฒนาการสื่อสารบนระบบเน็ตเวิร์ค ซึ่งการทดสอบจะทำการติดตั้งโปรแกรมที่พัฒนา

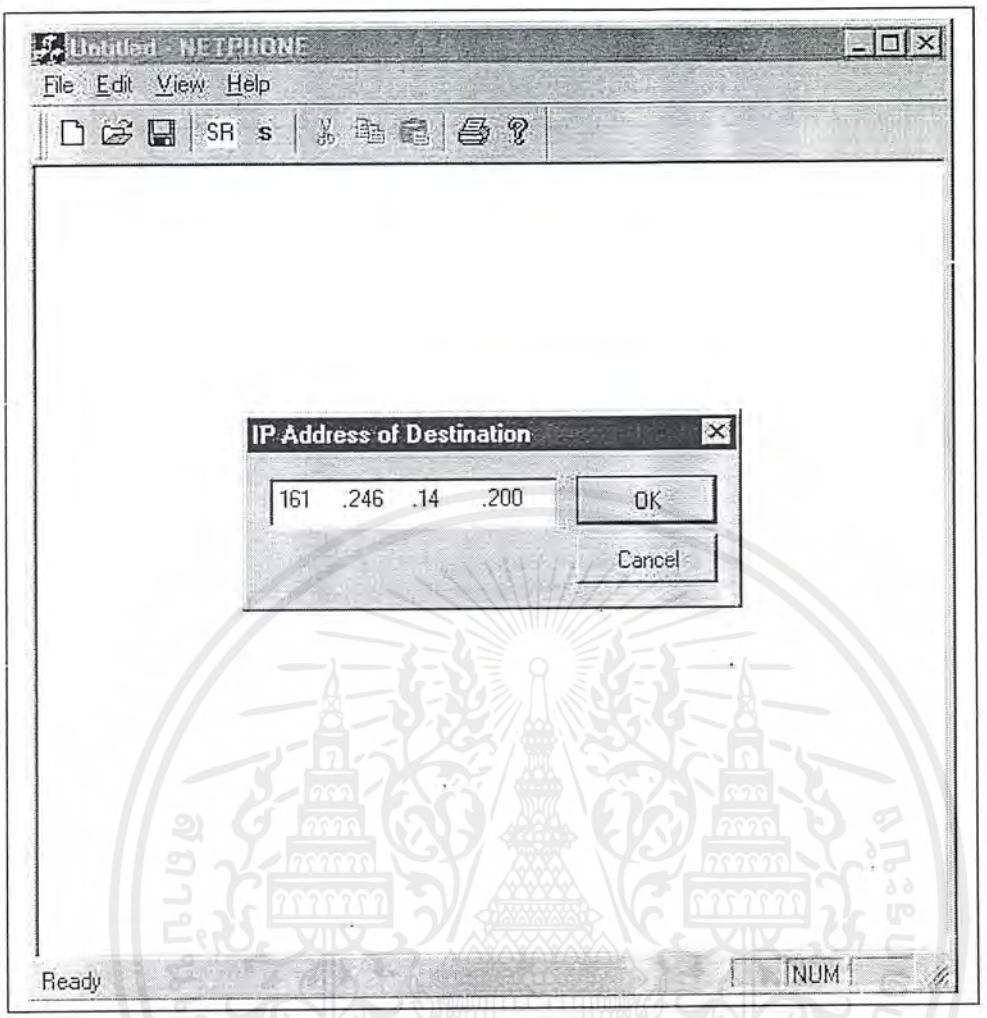
ขั้นตอนทำการทดลอง

1. ทำการตั้งค่าฟอร์แมทของเสียงที่จะทำการบันทึก อัตราการส่ง จำนวนบิทในการแซมเปิ้ล และขนาดของบัฟเฟอร์ในเก็บข้อมูลโดยในการทดลองเริ่มใช้ ค่าความถี่ในการแซมปลิ้งเท่ากับ 8,000 เฮิร์ท จำนวนบิทในการแซมเปิ้ลใช้ PCM ชนิด 8 บิท ดังโปรแกรมด้านล่าง

```
// CNETPHONEDoc construction/destruction

CNETPHONEDoc::CNETPHONEDoc()
  {
  nchannels=1;
  bitspersample=8;
  samplepersec=8000;
  samplesize=nchannels*(bitspersample/8);
  bytesperbuffer=(samplesize*samplepersec)/4;

  ipaddress.Empty();


CNETPHONEDoc::~CNETPHONEDoc()
  {
  receivesocket.Close();

  }
```

2. ในการทดลองครั้งแรกจะตั้งค่าขนาดของบัฟเฟอร์ในการเก็บข้อมูลขนาด 2000 ไบต์ ต่อบัฟเฟอร์ 1 ตัว โดยใช้อัตราการส่งข้อมูลเท่ากับ 64 Kb/s
3. ทำการ คอมไพล์โปรแกรมทั้งหมดแล้วติดต้องทดสอบกับคอมพิวเตอร์ทั้งสองฝั่ง โดยลองกับเครื่องคอมพิวเตอร์ที่คณะครุศาสตร์อุตสาหกรรรม โดยมีเงื่อนไขต้องอยู่บนเครื่องคอมพิวเตอร์ที่อยู่บนเน็ตเวิร์กของลาดกระบัง
4. ทำการติดตั้งพร้อมเซ็ตค่าไอพี แอดเดรสของที่เครื่องคอมพิวเตอร์ปลายทาง โดยเซ็ตค่าไอพีที่ฝั่งใดฝั่งหนึ่งดังรูป ที่ 4.8

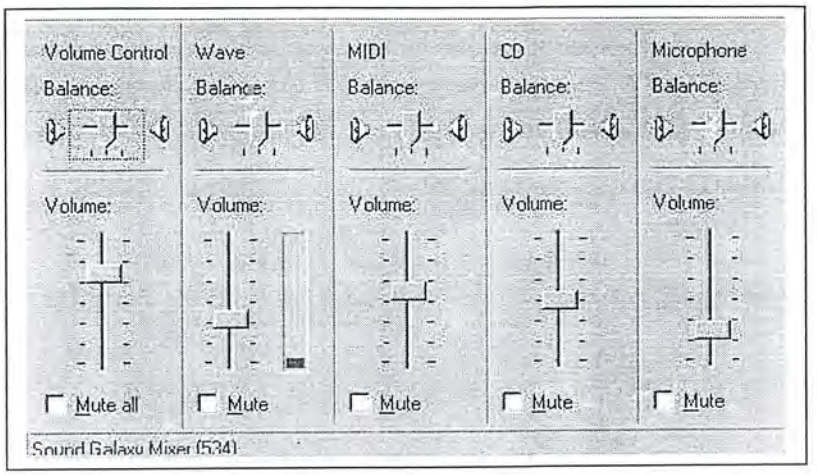
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูป 4.10แสดงการเซ็ตค่าไอพีเครื่องคอมพิวเตอร์ปลายทาง

5. ทำการเปิดโปรแกรมเดียวกันที่เครื่องคอมพิวเตอร์ปลายทาง แต่ไม่ต้องเซ็ตค่าไอพีอีกซึ่งซึ่งในตัวโปรแกรมนี้จะทำงาน แบบฟูลดูเพล็ก สามารถสื่อสารได้ทั้งสองทางพร้อมกัน
6. ในโปรแกรมเวอร์ชั่นแรกที่ทำการพัฒนาจะไม่มี การคอนฟิกค่าได้ในโปรแกรมที่คอมไพล์แล้วต้องแก้ค่าคอนฟิก ในซอร์ดโคด
7. เตรียมอุปกรณ์ทางด้านระบบ มัลติมีเดีย โดยจะมี ไมโครโฟนและลำโพงให้พร้อมก่อนทำการเซ็ตระดับสัญญานของ ไมโครโฟนและระดับเอาท์พุต ดังรูปที่ 4.9

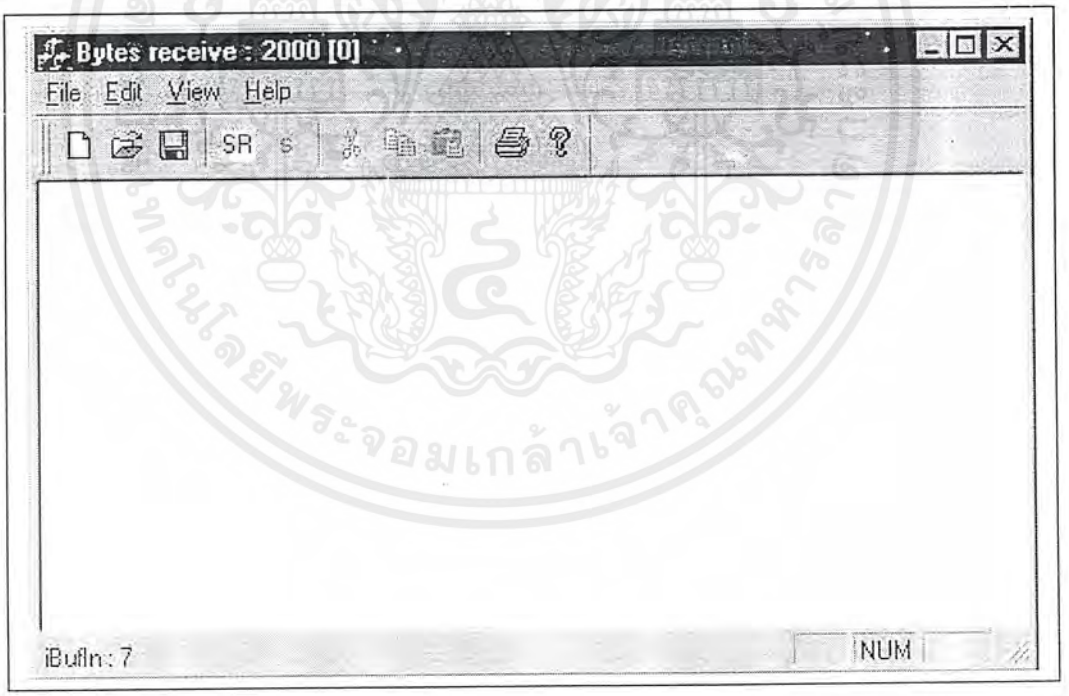
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูป 4.11 แสดงการปรับระดับสัญญานให้เหมาะสม

ผลการทดลอง

1. ทำการติดต่อสื่อสารโดยครั้งแรกยังไม่ทำการอัดเสียงจะ เห็นการค่าของการอัดข้อมูลลงในบัฟเฟอร์ทั้งตัวรับและตัวส่งแม้จะยังไม่ทำการพูดที่สเตตัสบาร์ทั้งด้านบนและด้านล่าง(รูปที่ 4.10)



รูปที่ 4.12 แสดงการรอรับข้อมูลลงในบัฟเฟอร์

2. จากการทดลองครั้งแรกจะตั้งค่าขนาดของบัฟเฟอร์ในการเก็บข้อมูลขนาด 2000 ไบต์ ต่อบัฟเฟอร์ 1 ตัว โดยใช้อัตราการส่งข้อมูลเท่ากับ 64 Kb/s โดยตั้งค่า แซมเปิ้ล ที่ 8000 แซมเปิ้ลต่อวินาที ทำการ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ส่งเสียง ได้ค่อนข้างชัดเจน แต่จะมีการดีเลย์ทั้งทางด้านฝั่งส่งและรับข้อมูลและเกิดการสะท้อนกลับของสัญญานเสียง (echo cancelled)

3. ทำการเปลี่ยนขนาดของบัฟเฟอร์ลงเพื่อทดลองอาจทำให้การดีเลย์ลดลง โดยเพิ่มตัวหารขึ้นจาก
4 เป็น 8 ทำให้ขนาดของบัฟเฟอร์เหลือตัวละ 1000 ไบต์

```
// CNETPHONEDoc construction/destruction

CNETPHONEDoc::CNETPHONEDoc()
  {
  nchannels=1;
  bitspersample=8;
  samplepersec=8000;
  samplesize=nchannels*(bitspersample/8);
  bytesperbuffer=(samplesize*samplepersec)/8; ***********

  ipaddress.Empty();


CNETPHONEDoc::~CNETPHONEDoc()
  {
  receivesocket.Close();

  }
```

5. ผลการทดลองที่พบ ก็คือจะพบการดีเลย์ของเสียงลดลง แต่คุณภาพของเสียงจะต่ำกว่าของเดิมเล็กน้อย และยังคงมีเสียง Echo อยู่บ้าง ซึ่งการเข้าไปปรับค่าใน Volume control ให้เหมาะสมก็จะช่วยอีกทางหนึ่งด้วย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 5
บทวิจารณ์และบทสรุป

5.1 บทวิจารณ์

เนื่องจากในโครงงานนี้ ทางกลุ่มพยายามศึกษาข้อมูล และเก็บข้อมูลเกี่ยวกับระบบไอพีเทเลโฟน ซึ่งเป็นเทคโนโลยีค่อนข้างใหม่ในประเทศไทย โดยในต่างประเทศได้มีการใช้งานกันอย่างกว้างขวาง จึงเป็นอุปสรรคที่สำคัญในการหาข้อมูล เพื่อนำมาดัดแปลงกับระบบคอมพิวเตอร์ในสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

โดยผลการทดลอง หลักๆ จะเป็นการดีเทคสัญญาณออกจากวงจรต่างๆ เพื่อนำมาเข้าวงจรอินเตอร์เฟส ปัญหาที่พบส่วนใหญ่เกี่ยวกับสัญญาณเสียงในภาควงจรไฮบริดจ์ที่มีคุณภาพต่ำกว่าที่ตั้งเป้าหมายไว้ จึงได้มีการแก้ไขปัญหาโดยเปลี่ยนจากหม้อแปลงไฮบริดจ์มาเป็น วงจรไฮบริดจ์อิเลคทรอนิกส์ ทำให้ได้สัญญาณเสียงที่สมบูรณ์ขึ้น

5.2 สรุปผลการทดลอง

จากการทดลองส่งสัญญาณเสียงบนเครือข่ายของลาดกระบัง โดยกำหนดอัตราการส่งอยู่ที่ 64 kb/s ซึ่งเป็นอัตราความเร็วค่อนข้างสูง ซึ่งในบางช่วงของการทดลองจะเกิดการดีเลย์ของเสียงมาก โดยเฉพาะในเวลากลางวัน แต่จะดีมากในช่วงเวลากลางคืน เพราะทราฟฟิก (Traffic) ของการใช้งานจะน้อยกว่าช่วงเวลาตอนกลางวัน

จากการทดลองจะได้สัญญาณเสียงค่อนข้างชัดเจนจนเป็นที่ยอมรับได้ ด้วยอัตราการแซมปลิ้งที่ความถี่ 8 kHz แบบพัลซ์โคดมอดดูเลชั่น (PCM) เมื่อทำการเปลี่ยนอัตราการแซมปลิ้งเป็น 16 kHz จะได้คุณภาพเสียงที่ดีขึ้น แต่จะมีการดีเลย์สูงในการรับ และส่งเสียง จึงเลือกค่าแซมปลิ้งที่ 8 kHz มาใช้ในการรับ และส่งเสียงเพราะให้คุณภาพของเสียงเป็นที่ยอมรับได้

เมื่อนำคู่สายโทรศัพท์มาต่อเข้ากับการ์ดที่ทำขึ้นมา พบว่าจะเกิดสัญญาณรบกวนจากคอมพิวเตอร์เข้าไปรบกวนในสายโทรศัพท์ ซึ่งได้ตรวจเช็คระบบกราวด์ และการแมชระหว่างอิมพีแดนซ์ของการ์ดเสียงกับวงจรไฮบริดจ์ บนการ์ดอินเตอร์เฟส ที่เปลี่ยนจากการใช้หม้อแปลงแมชชิ่งมาเป็นไอซีไฮบริดจ์ ทำให้ได้คุณภาพเสียงที่ดีขึ้น

5.3 ปัญหาที่พบจากการทดลอง

1. สัญญาณรบกวนในคู่สายโทรศัพท์ซึ่งเกิด
2. ในการกำหนดขนาดของบัฟเฟอร์พยายามที่จะลดขนาดของบัฟเฟอร์ลง ให้มีขนาดเล็ก เพื่อลดจำนวนของหน่วยความจำจากเดิมที่ใช้บัฟเฟอร์ตัวละ 2000ในการเก็บข้อมูล และเพื่อให้การส่งข้อมูลมีค่าดีเลย์ไทม์น้อยที่สุด
3. การบีบอัด พยายามที่จะลดขนาดของข้อมูลลง เพื่อให้สามารถส่งไปเครือข่ายได้ในอัตราความเร็วต่ำกว่า 64 kb/s
4. การทดลองยังไม่เคยได้ทดสอบผ่านโมเด็ม และเครือข่ายคอมพิวเตอร์ที่อื่น ๆ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

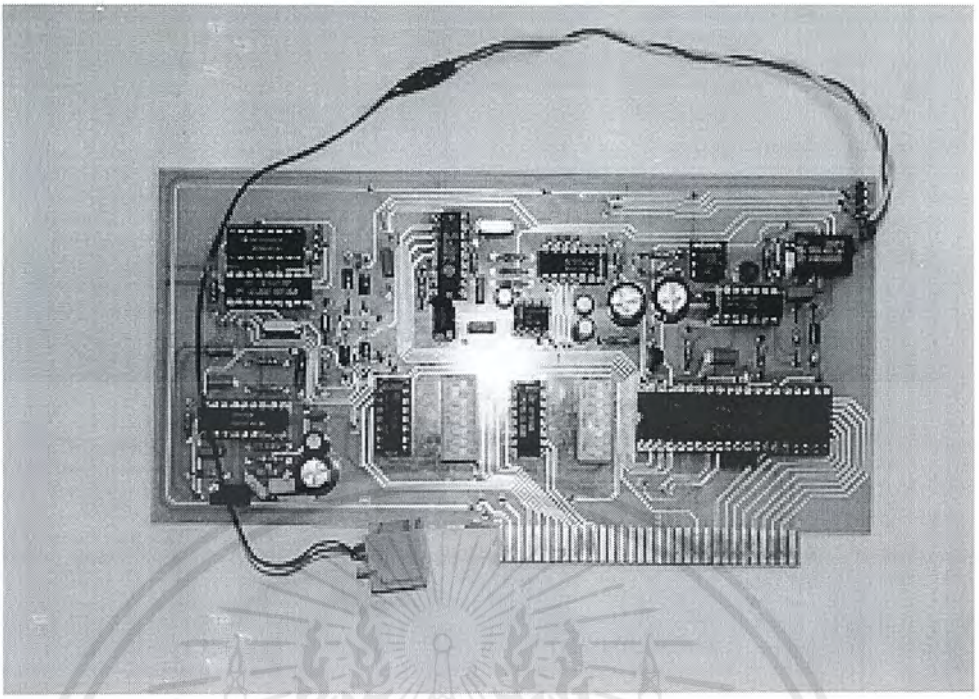
5.4 แนวทางในการพัฒนา

1. การลดขนาดของข้อมูลในการส่ง (Data Compression) ถ้าสามารถนำข้อมูลเสียงมาทำการบีบอัดโดยใช้ฟังก์ชัน ACM ใน Windows จะทำให้สามารถส่งผ่านโมเด็มได้
2. ด้านฮาร์ดแวร์ ควรใช้ชิพ DSP มาช่วยในการลดแอคโค่ และเรื่องคุณภาพเสียง(จากบทความของ อ.ยืน ภู่วรวรรณ เรื่องเครือข่ายระบบการสื่อสารแพ็กเก็ตเสียง

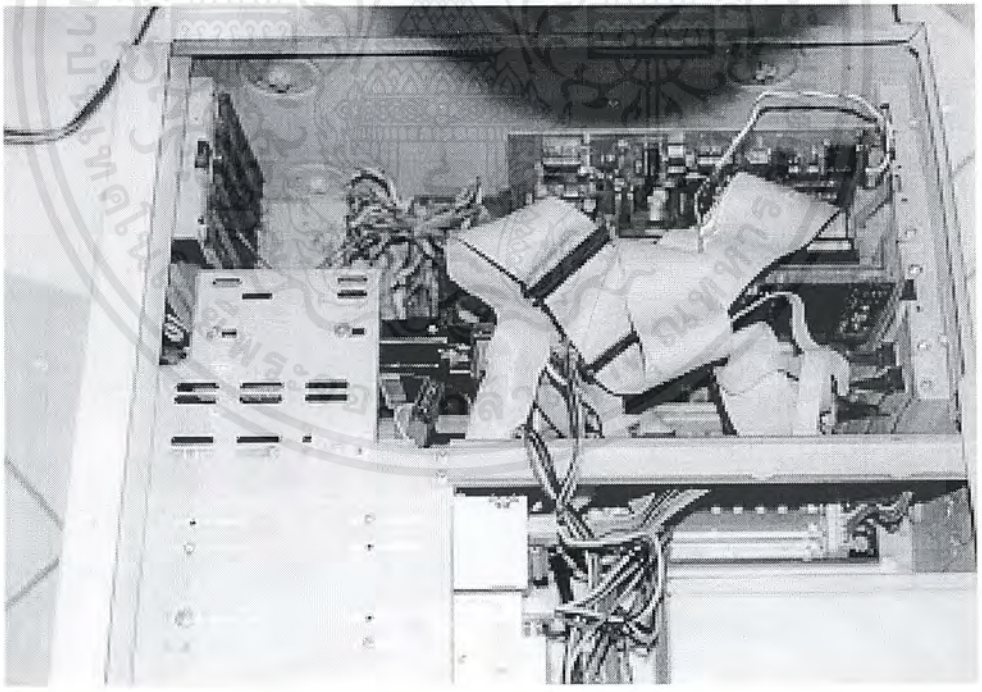
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

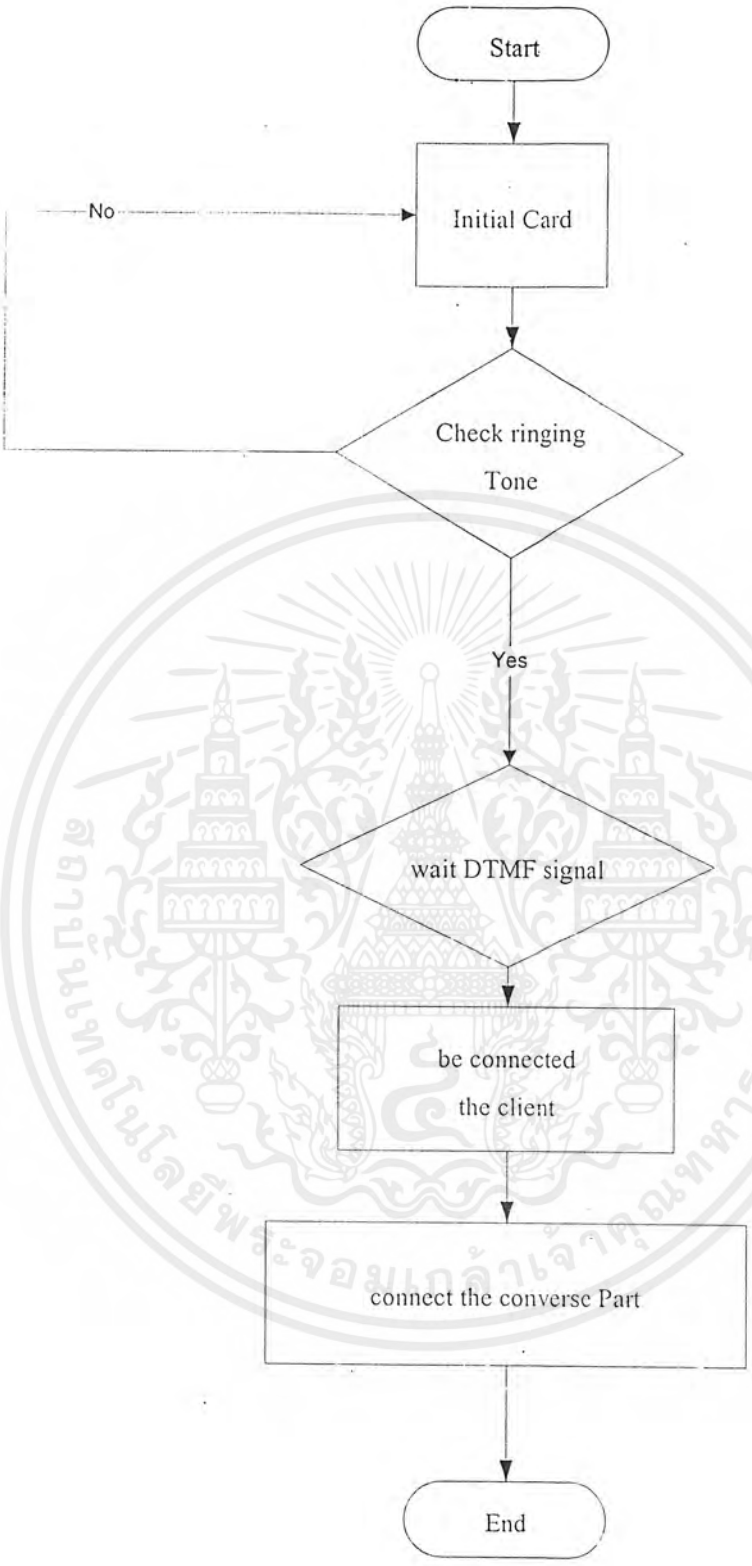


รูป แสดงการ์ดชิ้นงาน



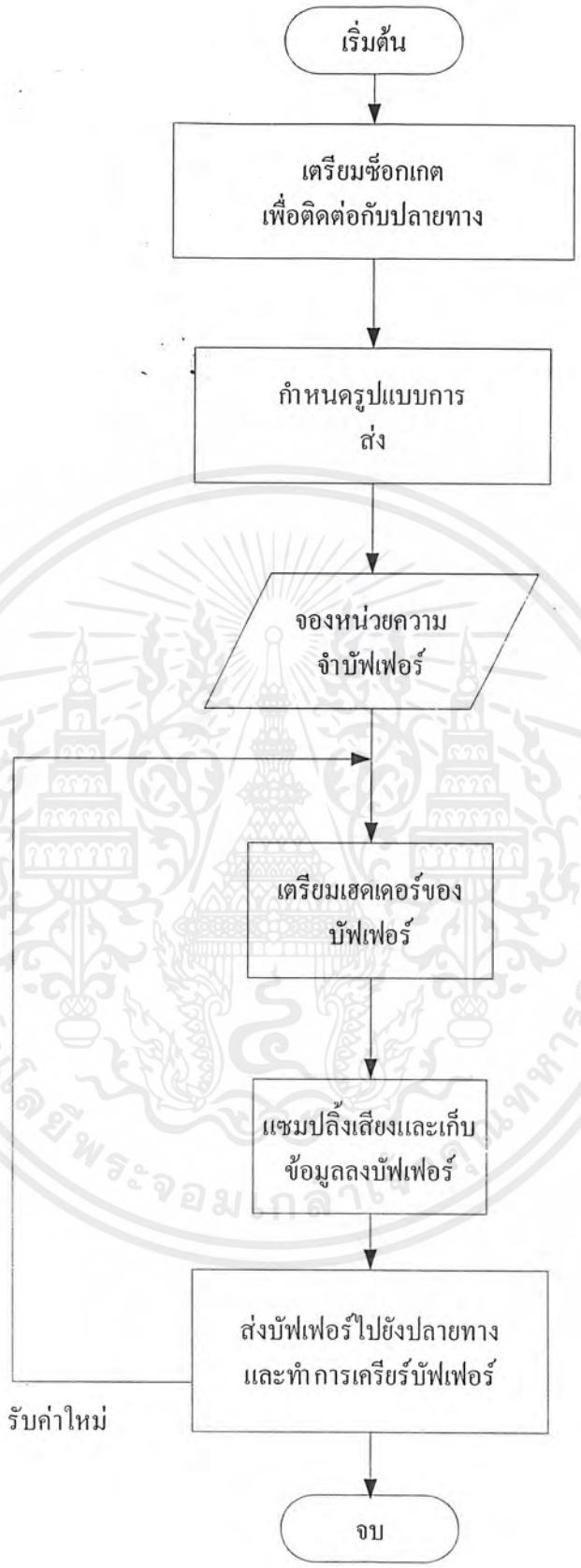
รูป แสดงการติดตั้งการ์ดขณะใช้งานจริง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



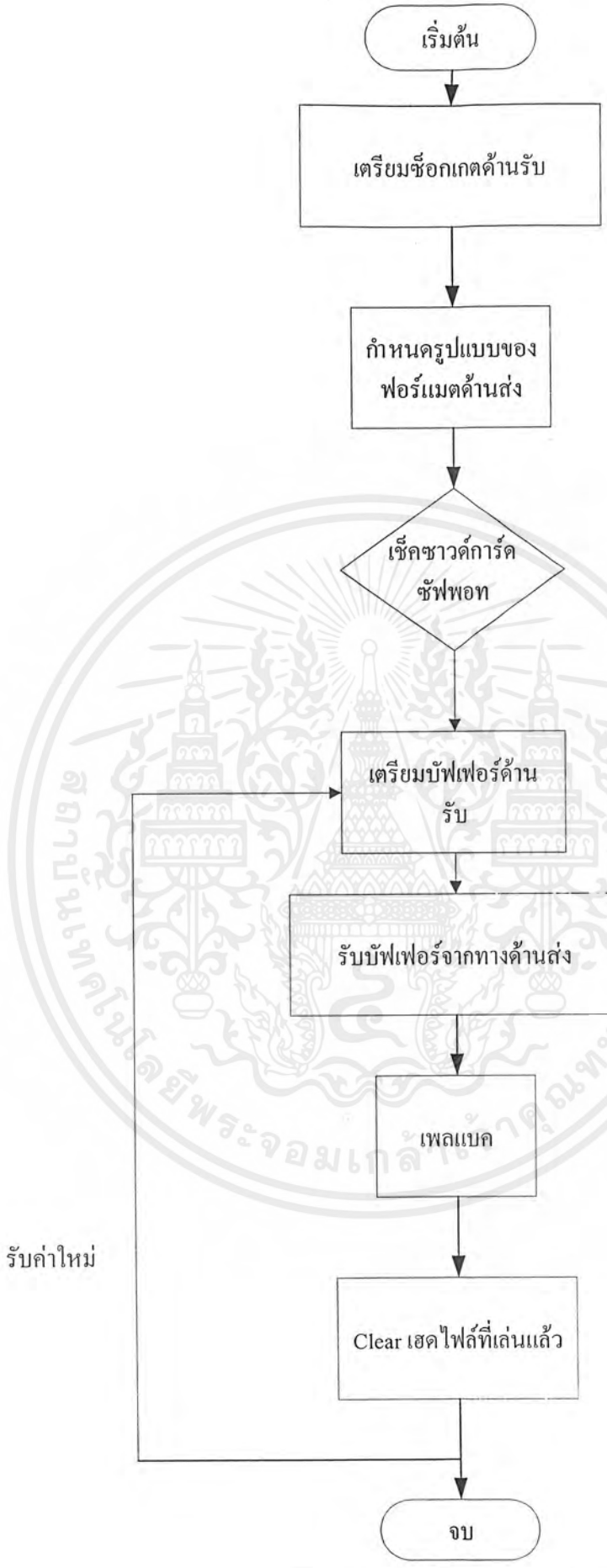
รูปแสดงบล็อคการทำงานของฮาร์ดแวร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



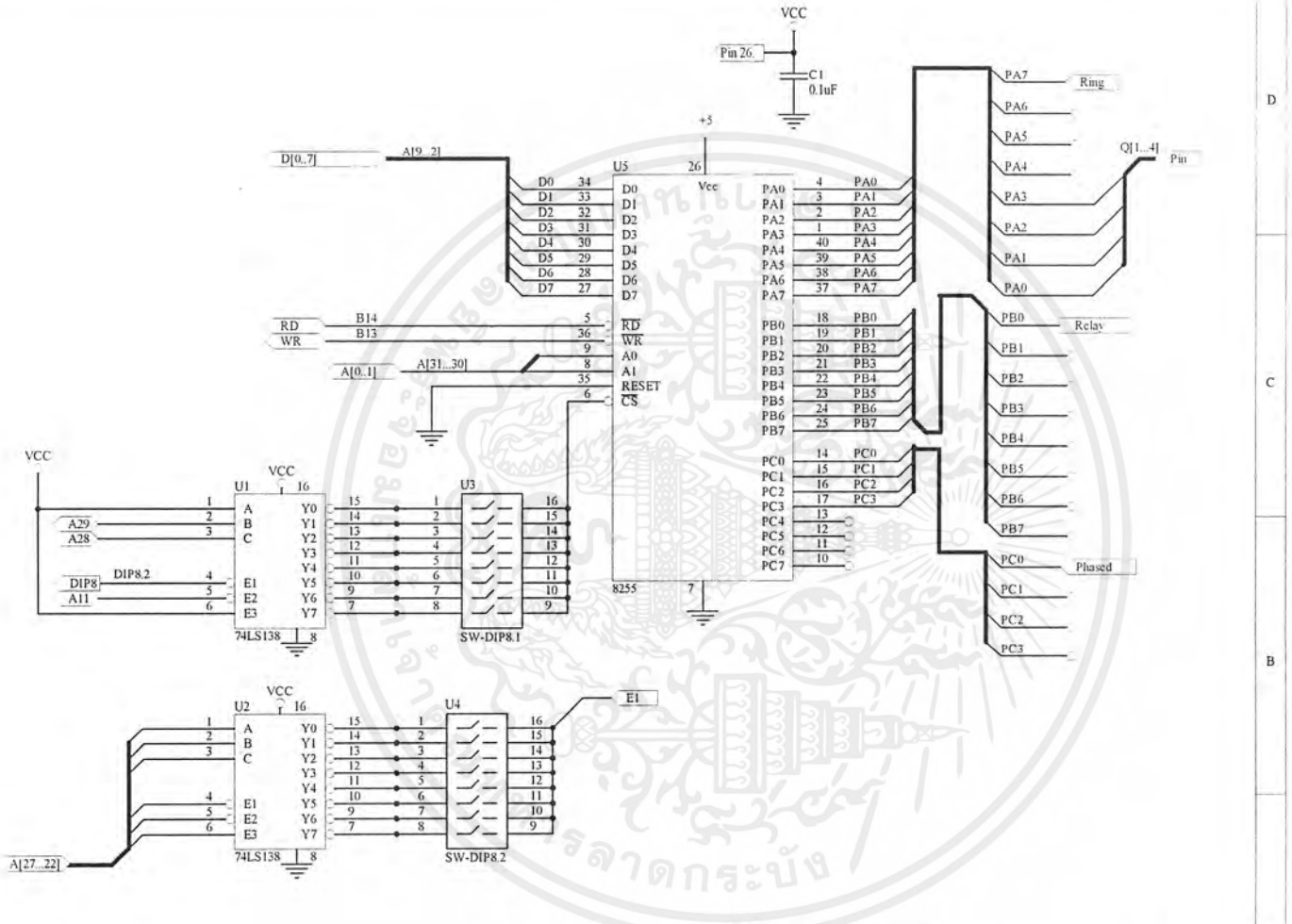
รูปแสดงบล็อคไดอะแกรมทางด้านส่ง

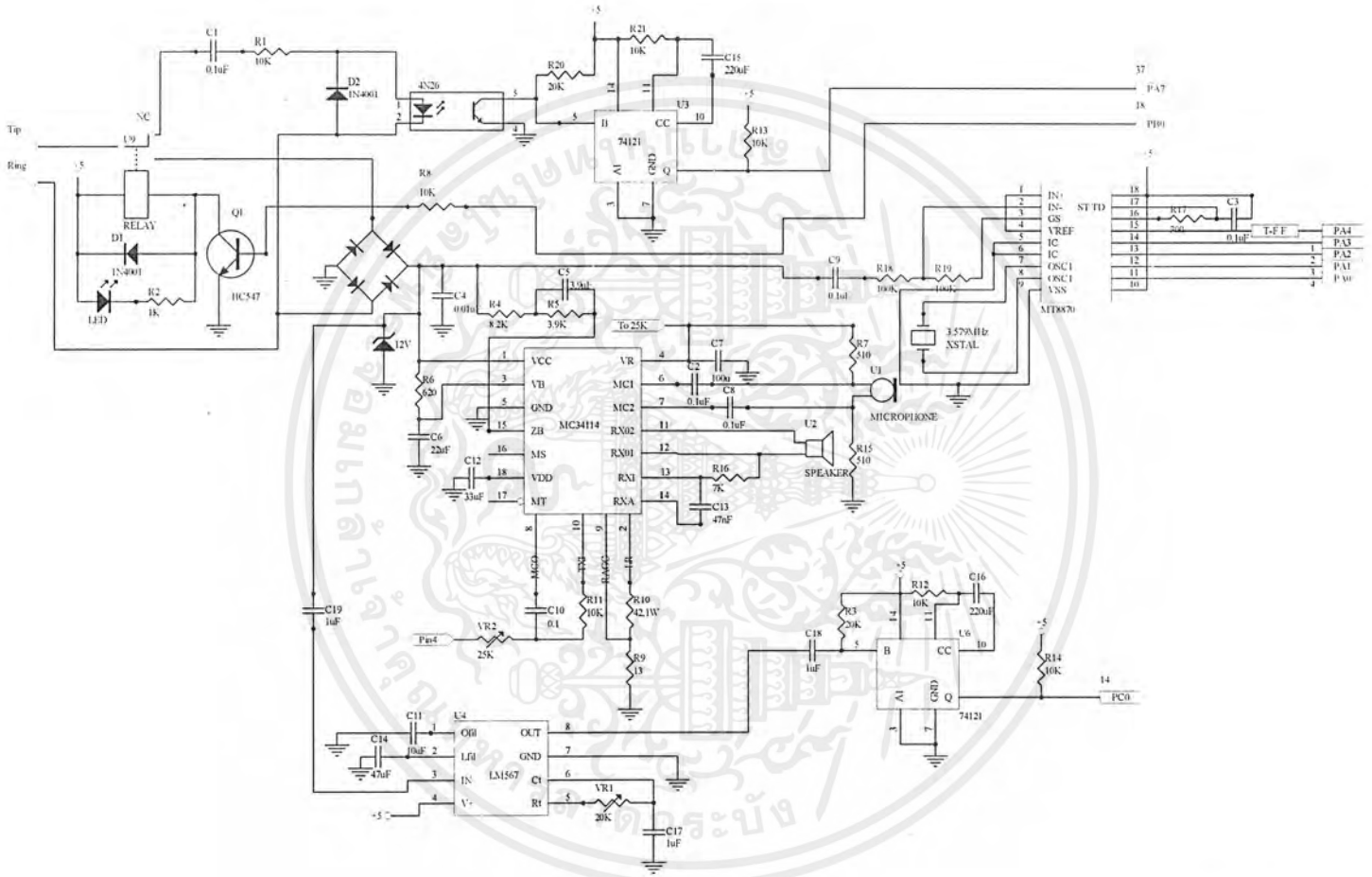
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปแสดงบลอกไดอะแกรมทางด้านรับ

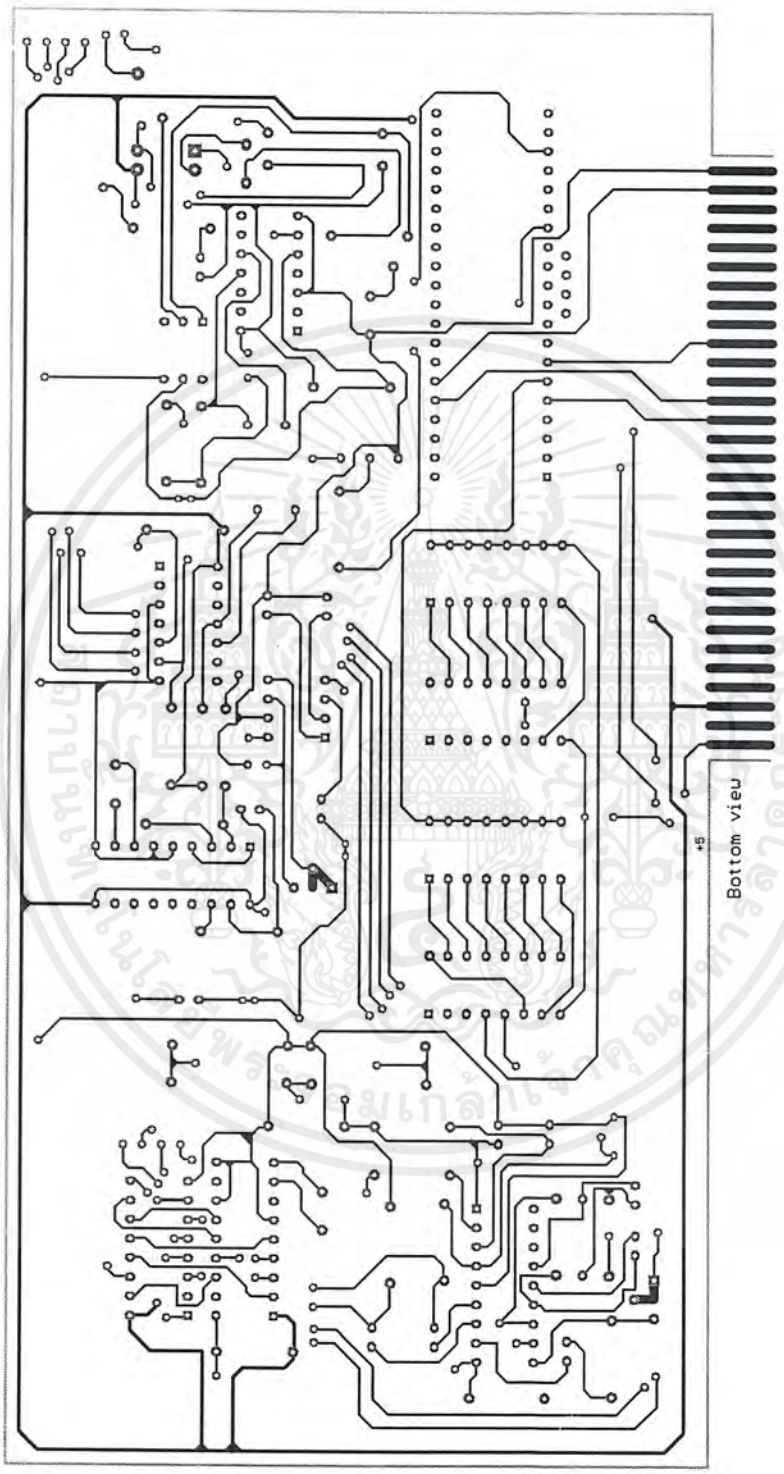
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้


VCC
Pin 26
C1
0.1uF
+5
D[0..7]
A19..21
U5
Vcc
D0
D1
D2
D3
D4
D5
D6
D7
RD
WR
A0
A1
RESET
CS
8255
PA0
PA1
PA2
PA3
PA4
PA5
PA6
PA7
PB0
PB1
PB2
PB3
PB4
PB5
PB6
PB7
PC0
PC1
PC2
PC3
PC4
PC5
PC6
PC7
B14
B13
A[0..1]
A[31...30]
Q[1...4]
Pin
Ring
Relay
Phased
U1
74LS138
U2
74LS138
U3
SW-DIP8.1
U4
SW-DIP8.2
A29
A28
DIP8
DIP8.2
A11
A[27...22]
E1
Title
Interface Part
Size
A
Number
Revision
Date:
1-Apr-1999
Sheet of
File:
C:\CLIENT\SCH3\PPI.SCH
Drawn By:

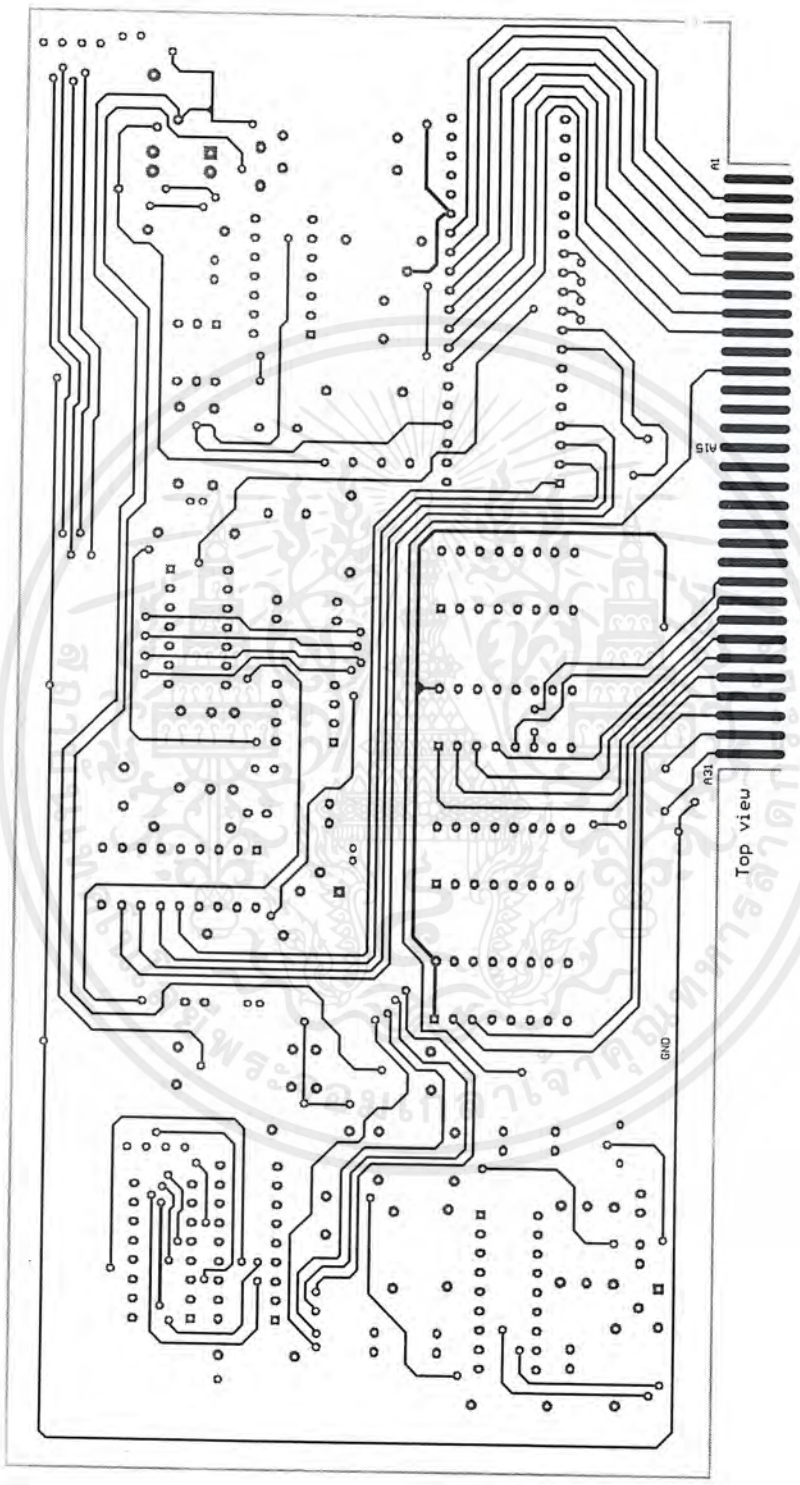


Title: *Telephone Part*

Size	Number	Revision
B		
Date:	1-Apr-1999	Sheet of
File:	C:\CLIENT\SCH\PROJ.SCH	Drawn By:



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
//=============================================================================
// IPDialog.h : header file
//-----------------------------------------------------------------------------

#ifndef _IPDIALOG_
#define _IPDIALOG_

#if
!defined(AFX_IPDIALOG_H__F02A183B_E81E_11D2_9E06_0080C8E98B2A__INCLU
DED_)
#define
AFX_IPDIALOG_H__F02A183B_E81E_11D2_9E06_0080C8E98B2A__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

// IPDialog dialog

class IPDialog : public CDialog
  {
  // Construction
  public:
    IPDialog(CWnd* pParent=NULL); // standard constructor

  // Dialog Data
    //{{AFX_DATA(IPDialog)
      enum {IDD=IDD_IPDIALOG};
      CIPAddressCtrl m_ipaddress;
    //}}AFX_DATA

  // Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(IPDialog)
  protected:
    virtual void DoDataExchange(CDataExchange* pDX); // DDX/DDV
support
    //}}AFX_VIRTUAL

  // Implementation
  protected:

  // Generated message map functions
    //{{AFX_MSG(IPDialog)
      virtual void OnOK();
      //}}AFX_MSG
  DECLARE_MESSAGE_MAP()
  };

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations
immediately before the previous line.

#endif //
!defined(AFX_IPDIALOG_H__F02A183B_E81E_11D2_9E06_0080C8E98B2A__INCLU
DED_)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
//
/////////////////////////////////////////////////////////////////////////////
//////

#if
!defined(AFX_MAINFRM_H__B13ABF7C_C11D_11D2_9DD2_0080C8E98B2A__INCLUDED_
)
#define AFX_MAINFRM_H__B13ABF7C_C11D_11D2_9DD2_0080C8E98B2A__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000

class CMainFrame : public CFrameWnd
  {
  protected: // create from serialization only
    CMainFrame();
    DECLARE_DYNCREATE(CMainFrame)

  // Attributes
  public:

  // Operations
  public:

  // Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CMainFrame)
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
    //}}AFX_VIRTUAL

  // Implementation
  public:
    virtual ~CMainFrame();
  #ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
  #endif

  protected:  // control bar embedded members
    CStatusBar m_wndStatusBar;
    CToolBarEx m_wndToolBar;

  // Generated message map functions
  protected:
    //{{AFX_MSG(CMainFrame)
    afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
    //}}AFX_MSG
  DECLARE_MESSAGE_MAP()
  };

/////////////////////////////////////////////////////////////////////////////
//////
//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations
immediately before the previous line.
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
#endif //
!defined(AFX_MAINFRM_H__B13ABF7C_C11D_11D2_9DD2_0080C8E98B2A__INCLUDED_
)
// NETPHONE.h : main header file for the NETPHONE application
//

#if
!defined(AFX_NETPHONE_H__B13ABF78_C11D_11D2_9DD2_0080C8E98B2A__INCLUDED
_)
#define AFX_NETPHONE_H__B13ABF78_C11D_11D2_9DD2_0080C8E98B2A__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000

#ifndef __AFXWIN_H__
  #error include 'stdafx.h' before including this file for PCH
#endif

#include "resource.h"       // main symbols
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
/////////////////////////////////////////////////////////////////////////////
//////
// CNETPHONEApp:
// See NETPHONE.cpp for the implementation of this class
//

class CNETPHONEApp : public CWinApp
  {
  public:
    CNETPHONEApp();

  // Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CNETPHONEApp)
  public:
    virtual BOOL InitInstance();
    //}}AFX_VIRTUAL

  // Implementation
    //{{AFX_MSG(CNETPHONEApp)
    afx_msg void OnAppAbout();
    //}}AFX_MSG
  DECLARE_MESSAGE_MAP()
  };

/////////////////////////////////////////////////////////////////////////////
//////
//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations
immediately before the previous line.

#endif //
!defined(AFX_NETPHONE_H__B13ABF78_C11D_11D2_9DD2_0080C8E98B2A__INCLUDED
_)



// NETPHONEDoc.h : interface of the CNETPHONEDoc class
//
/////////////////////////////////////////////////////////////////////////////
//////

#if
!defined(AFX_NETPHONEDOC_H__B13ABF7E_C11D_11D2_9DD2_0080C8E98B2A__INCLU
DED_)
#define
AFX_NETPHONEDOC_H__B13ABF7E_C11D_11D2_9DD2_0080C8E98B2A__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000

#include "StartThread.h"
#include "Thread.h"
#include "Playthread.h"
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
#include "Soundsocket.h"

class CNETPHONEDoc : public CDocument
  {
  protected: // create from serialization only
    CNETPHONEDoc();
    DECLARE_DYNCREATE(CNETPHONEDoc)

  // Attributes
  public:
     STARTThread  *m_pSTARTThread;
     RECORDThread *m_pRECORDThread;
     PLAYThread   *m_pPLAYThread;

  // Operations
  public:

  // Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CNETPHONEDoc)
  public:
    virtual BOOL OnNewDocument();
    virtual void Serialize(CArchive& ar);
    //}}AFX_VIRTUAL

  // Implementation
  public:
    virtual ~CNETPHONEDoc();
    void    Send();
    BYTE    Inport(unsigned short port);
    void    Outport(unsigned short port,BYTE data);
  #ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
  #endif

  protected:

  // Generated message map functions
  protected:
    //{{AFX_MSG(CNETPHONEDoc)
      afx_msg void OnSend();
    afx_msg void OnStart();
    afx_msg void OnUpdateSend(CCmdUI* pCmdUI);
    //}}AFX_MSG
  DECLARE_MESSAGE_MAP()
  };

/////////////////////////////////////////////////////////////////////////////
//////
//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations
immediately before the previous line.

#endif //
!defined(AFX_NETPHONEDOC_H__B13ABF7E_C11D_11D2_9DD2_0080C8E98B2A__INCLU
DED_)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
// NETPHONEView.h : interface of the CNETPHONEView class
//
/////////////////////////////////////////////////////////////////////////////
//////

#if
!defined(AFX_NETPHONEVIEW_H__B13ABF80_C11D_11D2_9DD2_0080C8E98B2A__INCL
UDED_)
#define
AFX_NETPHONEVIEW_H__B13ABF80_C11D_11D2_9DD2_0080C8E98B2A__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000

class CNETPHONEView : public CView
  {
  protected: // create from serialization only
    CNETPHONEView();
    DECLARE_DYNCREATE(CNETPHONEView)

  // Attributes
  public:
    CNETPHONEDoc* GetDocument();

  // Operations
  public:

  // Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CNETPHONEView)
  public:
    virtual void OnDraw(CDC* pDC);  // overridden to draw this view
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
  protected:
    virtual BOOL OnPreparePrinting(CPrintInfo* pInfo);
    virtual void OnBeginPrinting(CDC* pDC, CPrintInfo* pInfo);
    virtual void OnEndPrinting(CDC* pDC, CPrintInfo* pInfo);
  //}}AFX_VIRTUAL

  // Implementation
  public:
    virtual ~CNETPHONEView();
  #ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
  #endif

  protected:
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
  // Generated message map functions
  protected:
    //{{AFX_MSG(CNETPHONEView)
    //}}AFX_MSG
  DECLARE_MESSAGE_MAP()
  };

#ifndef _DEBUG  // debug version in NETPHONEView.cpp
  inline CNETPHONEDoc* CNETPHONEView::GetDocument() {return
(CNETPHONEDoc*)m_pDocument;}
#endif

////////////////////////////////////////////////////////////////////////
//////
//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations
immediately before the previous line.

#endif //
!defined(AFX_NETPHONEVIEW_H__B13ABF80_C11D_11D2_9DD2_0080C8E98B2A__INCL
UDED_)




//=====================================================================
====
// PLAYThread.h : header file
//---------------------------------------------------------------------
----

#ifndef _PLAYThread_
#define _PLAYThread_

#if
!defined(AFX_PLAYTHREAD_H__6F1E7ACA_E5E9_11D2_9E04_0080C8E98B2A__INCLUD
ED_)
#define
AFX_PLAYTHREAD_H__6F1E7ACA_E5E9_11D2_9E04_0080C8E98B2A__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

class CNETPHONEDoc;

////////////////////////////////////////////////////////////////////////
//////
// PLAYThread thread

class PLAYThread : public CWinThread
  {
  DECLARE_DYNCREATE(PLAYThread)
  protected:
    PLAYThread(); // protected constructor used by dynamic creation
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
// Attributes
public:
   CNETPHONEDoc *m_pOwner;

// Operations
public:
  void SetOwner(CNETPHONEDoc* pOwner) {m_pOwner=pOwner;};
  void KillThread();
  void ClearUpThread();

// Overrides
  // ClassWizard generated virtual function overrides
  //{{AFX_VIRTUAL(PLAYThread)
    public:
  virtual BOOL InitInstance();
  virtual int ExitInstance();
    virtual int Run();
    //}}AFX_VIRTUAL

// Implementation
protected:
  virtual ~PLAYThread();
  static LRESULT CALLBACK waveOutProc(HWAVEOUT hwi,UINT uMsg,DWORD
dwInstance,DWORD dwParam1,DWORD dwParam2);

// Generated message map functions
  //{{AFX_MSG(PLAYThread)
  //}}AFX_MSG

DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////
//////
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately
before the previous line.

#endif //
!defined(AFX_PLAYTHREAD_H__6F1E7ACA_E5E9_11D2_9E04_0080C8E98B2A__INCLUD
ED_)

#endif // _PLAYThread_


//{{NO_DEPENDENCIES}}
// Microsoft Developer Studio generated include file.
// Used by Netphone.rc
//
#define IDD_ABOUTBOX                    100
#define IDP_SOCKETS_INIT_FAILED         104
#define IDR_MAINFRAME                   128
#define IDR_NETPHOTYPE                  129
#define IDD_IPDIALOG                    130
#define IDC_IPADDRESS                   1000
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
#define ID_SEND                         32771
#define ID_START                        32772

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_3D_CONTROLS                     1
#define _APS_NEXT_RESOURCE_VALUE        135
#define _APS_NEXT_COMMAND_VALUE         32775
#define _APS_NEXT_CONTROL_VALUE         1001
#define _APS_NEXT_SYMED_VALUE           101
#endif
#endif
```


เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
// SoundSocket.h : header file
//

#if
!defined(AFX_SOUNDSOCKET_H__D83E6FC7_DAEA_11D2_845E_0080C80D1CC6__INCLU
DED_)
#define
AFX_SOUNDSOCKET_H__D83E6FC7_DAEA_11D2_845E_0080C80D1CC6__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

class CNETPHONEDoc;

// SoundSocket command target

class SoundSocket : public CAsyncSocket
  {
  // Attributes
  public:
     CNETPHONEDoc *m_pOwner;

  // Operations
  public:
    SoundSocket();
    virtual ~SoundSocket();
    void SetOwner(CNETPHONEDoc* pOwner) {m_pOwner=pOwner;};

  // Overrides
  public:
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(SoundSocket)
  public:
    virtual void OnReceive(int nErrorCode);
    //}}AFX_VIRTUAL

  // Generated message map functions
    //{{AFX_MSG(SoundSocket)
    //}}AFX_MSG

  // Implementation
  protected:
  };

/////////////////////////////////////////////////////////////////////////////
//////
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately
before the previous line.

#endif //
!defined(AFX_SOUNDSOCKET_H__D83E6FC7_DAEA_11D2_845E_0080C80D1CC6__INCLU
DED_)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
// STARTThread.h : header file
//-------------------------------------------------------------------------
----

#ifndef _STARTThread_
#define _STARTThread_

class CNETPHONEDoc;

#if
!defined(AFX_STARTTHREAD_H__F02A183F_E81E_11D2_9E06_0080C8E98B2A__INCLU
DED_)
#define
AFX_STARTTHREAD_H__F02A183F_E81E_11D2_9E06_0080C8E98B2A__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

/////////////////////////////////////////////////////////////////////////
//////
// STARTThread thread

class STARTThread : public CWinThread
  {
  DECLARE_DYNCREATE(STARTThread)
  protected:
    STARTThread(); // protected constructor used by dynamic creation

  // Attributes
  public:
    CNETPHONEDoc *m_pOwner;

  // Operations
  public:
    void SetOwner(CNETPHONEDoc* pOwner) {m_pOwner=pOwner;};
    void KillThread();
    void ClearUpThread();

  // Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(STARTThread)
      public:
    virtual BOOL InitInstance();
    virtual int ExitInstance();
    virtual int Run();
    //}}AFX_VIRTUAL

  // Implementation
  protected:
    virtual ~STARTThread();

  // Generated message map functions
    //{{AFX_MSG(STARTThread)
    //}}AFX_MSG

  DECLARE_MESSAGE_MAP()
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
};

/////////////////////////////////////////////////////////////////////////////
//////
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately
before the previous line.

#endif //
!defined(AFX_STARTTHREAD_H__F02A183F_E81E_11D2_9E06_0080C8E98B2A__INCLU
DED_)

#endif // _STARTThread_
```


เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
// stdafx.h : include file for standard system include files,
//  or project specific include files that are used frequently, but
//      are changed infrequently
//

#if
!defined(AFX_STDAFX_H__B13ABF7A_C11D_11D2_9DD2_0080C8E98B2A__INCLUDED_)
#define AFX_STDAFX_H__B13ABF7A_C11D_11D2_9DD2_0080C8E98B2A__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000

#define VC_EXTRALEAN            // Exclude rarely-used stuff from Windows
headers

#include <afxwin.h>         // MFC core and standard components
#include <afxext.h>         // MFC extensions
#include <afxdisp.h>        // MFC OLE automation classes
#ifndef _AFX_NO_AFXCMN_SUPPORT
#include <afxcmn.h>                     // MFC support for Windows Common
Controls
#endif // _AFX_NO_AFXCMN_SUPPORT

#include <afxmt.h>
#include <afxsock.h>            // MFC socket extensions
#include <vfw.h>
#include "toolbar.h"
#pragma comment(lib,"vfw32.lib")
#pragma comment(lib,"winmm.lib")

//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations
immediately before the previous line.

#endif //
!defined(AFX_STDAFX_H__B13ABF7A_C11D_11D2_9DD2_0080C8E98B2A__INCLUDED_)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
//=============================================================================
// THREAD.h : header file
//-----------------------------------------------------------------------------

#ifndef _RECORDThread_
#define _RECORDThread_

class CNETPHONEDoc;

/////////////////////////////////////////////////////////////////////////////
//////
// DECODEThread thread

class RECORDThread : public CWinThread
  {
  DECLARE_DYNCREATE(RECORDThread)
  protected:
    RECORDThread(); // protected constructor used by dynamic creation

  // Attributes
  public:
    CNETPHONEDoc *m_pOwner;

  // Operations
  public:
    void SetOwner(CNETPHONEDoc* pOwner) {m_pOwner=pOwner;};
    void KillThread();
    void ClearUpThread();

  // Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(DECODEThread)
  public:
    virtual BOOL InitInstance();
    virtual int  ExitInstance();
    virtual int  Run();
    //}}AFX_VIRTUAL

  // Implementation
  protected:
    virtual ~RECORDThread();
    static LRESULT CALLBACK waveInProc(HWAVEIN hwi,UINT uMsg,DWORD
dwInstance,DWORD dwParam1,DWORD dwParam2);

  // Generated message map functions
  //{{AFX_MSG(RECORDThread)
    // NOTE - the ClassWizard will add and remove member functions
here.
  //}}AFX_MSG
  DECLARE_MESSAGE_MAP()
  };

#endif // _RECORDThread_
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
// MainFrm.cpp : implementation of the CMainFrame class
//

#include "stdafx.h"
#include "NETPHONE.h"
#include "MainFrm.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
//////
// CMainFrame

IMPLEMENT_DYNCREATE(CMainFrame,CFrameWnd)

BEGIN_MESSAGE_MAP(CMainFrame,CFrameWnd)
  //{{AFX_MSG_MAP(CMainFrame)
  ON_WM_CREATE()
  //}}AFX_MSG_MAP
END_MESSAGE_MAP()

static UINT indicators[]={ID_SEPARATOR,
                          ID_INDICATOR_CAPS,
                          ID_INDICATOR_NUM,
                          ID_INDICATOR_SCRL,};

/////////////////////////////////////////////////////////////////////////////
//////
// CMainFrame construction/destruction

CMainFrame::CMainFrame()

  {
  }

CMainFrame::~CMainFrame()
  {
  }

int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
  {
  if(CFrameWnd::OnCreate(lpCreateStruct)==-1) return(-1);

  if(!m_wndToolBar.Create(this) || !m_wndToolBar.LoadToolBar
(IDR_MAINFRAME))
    {
    TRACE0("Failed to create toolbar\n");
    return(-1);
    }

  if(!m_wndStatusBar.Create(this) || !m_wndStatusBar.SetIndicators
(indicators,sizeof(indicators)/sizeof(UINT)))
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    {
    TRACE0("Failed to create status bar\n");
    return(-1);
    }

m_wndToolBar.SetBarStyle(m_wndToolBar.GetBarStyle()|CBRS_TOOLTIPS|CBRS_
FLYBY|CBRS_SIZE_DYNAMIC);

  m_wndToolBar.EnableDocking(CBRS_ALIGN_ANY);
  EnableDocking(CBRS_ALIGN_ANY);
  DockControlBar(&m_wndToolBar);
  return(0);
  }

BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)
  {
  cs.x=20; cs.y=20;
  cs.cx=::GetSystemMetrics(SM_CXFULLSCREEN)-(2*cs.x);
  cs.cy=::GetSystemMetrics(SM_CYFULLSCREEN)-(2*cs.y);
  return CFrameWnd::PreCreateWindow(cs);
  }

////////////////////////////////////////////////////////////////////////////
//////
// CMainFrame diagnostics

#ifdef _DEBUG
void CMainFrame::AssertValid() const
  {
  CFrameWnd::AssertValid();
  }

void CMainFrame::Dump(CDumpContext& dc) const
  {
  CFrameWnd::Dump(dc);
  }

#endif //_DEBUG

////////////////////////////////////////////////////////////////////////////
//////
// CMainFrame message handlers
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
// IPDialog.cpp : implementation file
//

#include "stdafx.h"
#include "netphone.h"
#include "IPDialog.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

extern CString ipaddress;

/////////////////////////////////////////////////////////////////////////////
//////
// IPDialog dialog

IPDialog::IPDialog(CWnd* pParent):CDialog(IPDialog::IDD, pParent)
  {
  //{{AFX_DATA_INIT(IPDialog)
  //}}AFX_DATA_INIT
  }

void IPDialog::DoDataExchange(CDataExchange* pDX)
  {
  CDialog::DoDataExchange(pDX);
  //{{AFX_DATA_MAP(IPDialog)
  DDX_Control(pDX,IDC_IPADDRESS,m_ipaddress);
  //}}AFX_DATA_MAP
  }

BEGIN_MESSAGE_MAP(IPDialog,CDialog)

  //{{AFX_MSG_MAP(IPDialog)
      //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
//////
// IPDialog message handlers

void IPDialog::OnOK()
  {
  if(m_ipaddress.IsBlank()==0)
    {
    char ipadd[100];
    BYTE ip0,ip1,ip2,ip3;

    if(m_ipaddress.GetAddress(ip0,ip1,ip2,ip3)==4)
      {
      sprintf(ipadd,"%d.%d.%d.%d",ip0,ip1,ip2,ip3);
      ipaddress=ipadd;
      }
    else ipaddress.Empty();
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
  }
CDialog::OnOK();
}
```


เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
// NETPHONEDoc.cpp : implementation of the CNETPHONEDoc class
//

#include "stdafx.h"
#include "NETPHONE.h"
#include "NETPHONEDoc.h"
#include "soundsocket.h"
#include "IPDialog.h"
#include <conio.h>

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

extern int     nchannels;      // number of channel
extern int     samplepersec;   // sampling frequency
extern int     bitspersample;  // bits per sample
extern int     samplesize;     // bytes per sample
extern int     bytesperbuffer; // bytes per buffer
extern CString ipaddress;

HANDLE      hEventDeadStart;
HANDLE      hEventKillStart;

HANDLE      hEventIn[2];
HANDLE      hEventDoneIn;
HANDLE      hEventDeadIn;
HANDLE      hEventKillIn;

HANDLE      hEventOut[2];
HANDLE      hEventDoneOut;

HANDLE      hEventDeadOut;
HANDLE      hEventKillOut;

SoundSocket receivesocket;

/////////////////////////////////////////////////////////////////////////////
//////
// CNETPHONEDoc

IMPLEMENT_DYNCREATE(CNETPHONEDoc, CDocument)

BEGIN_MESSAGE_MAP(CNETPHONEDoc, CDocument)
  //{{AFX_MSG_MAP(CNETPHONEDoc)
  ON_COMMAND(ID_SEND, OnSend)
  ON_COMMAND(ID_START, OnStart)
  ON_UPDATE_COMMAND_UI(ID_SEND, OnUpdateSend)
  //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
//////
// CNETPHONEDoc construction/destruction
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
CNETPHONEDoc::CNETPHONEDoc()
  {
  nchannels=1;
  bitspersample=8;
  samplepersec=8000;
  samplesize=nchannels*(bitspersample/8);
  bytesperbuffer=(samplesize*samplepersec)/4;

  ipaddress.Empty();

  if(!receivesocket.Create(2000,SOCK_DGRAM)) AfxMessageBox("Can't
create receivesocket");

  hEventDeadStart=CreateEvent(NULL,FALSE,FALSE,NULL);
  hEventKillStart=CreateEvent(NULL,FALSE,FALSE,NULL);

  hEventDoneIn=CreateEvent(NULL,FALSE,FALSE,NULL);
  hEventDeadIn=CreateEvent(NULL,FALSE,FALSE,NULL);
  hEventKillIn=CreateEvent(NULL,FALSE,FALSE,NULL);

  hEventDoneOut=CreateEvent(NULL,FALSE,FALSE,NULL);
  hEventDeadOut=CreateEvent(NULL,FALSE,FALSE,NULL);
  hEventKillOut=CreateEvent(NULL,FALSE,FALSE,NULL);

  hEventIn[0]=hEventKillIn;
  hEventIn[1]=hEventDoneIn;

  hEventOut[0]=hEventKillOut;
  hEventOut[1]=hEventDoneOut;

  m_pSTARTThread=NULL;
  m_pRECORDThread=NULL;
  m_pPLAYThread=NULL;

  m_pPLAYThread=(PLAYThread*)AfxBeginThread(RUNTIME_CLASS
(PLAYThread),THREAD_PRIORITY_NORMAL,0,CREATE_SUSPENDED);
  m_pPLAYThread->SetOwner(this);
  m_pPLAYThread->ResumeThread();

  receivesocket.SetOwner(this);
  }

CNETPHONEDoc::~CNETPHONEDoc()
  {
  receivesocket.Close();

  if(m_pSTARTThread!=NULL) m_pSTARTThread->KillThread();
  if(m_pRECORDThread!=NULL) m_pRECORDThread->KillThread();
  if(m_pPLAYThread!=NULL) m_pPLAYThread->KillThread();

  CloseHandle(hEventDeadStart);
  CloseHandle(hEventKillStart);

  CloseHandle(hEventDoneIn);
  CloseHandle(hEventDeadIn);
  CloseHandle(hEventKillIn);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
  CloseHandle(hEventDoneOut);
  CloseHandle(hEventDeadOut);
  CloseHandle(hEventKillOut);
  }

BOOL CNETPHONEDoc::OnNewDocument()
  {
  if(!CDocument::OnNewDocument()) return(FALSE);
  return(TRUE);
  }

/////////////////////////////////////////////////////////////////////////////
//////
// CNETPHONEDoc serialization

void CNETPHONEDoc::Serialize(CArchive& ar)
  {
  if(ar.IsStoring())
    {
    }
  else
    {
    }
  }

/////////////////////////////////////////////////////////////////////////////
//////
// CNETPHONEDoc diagnostics

#ifdef _DEBUG
void CNETPHONEDoc::AssertValid() const
  {
  CDocument::AssertValid();
  }

void CNETPHONEDoc::Dump(CDumpContext& dc) const
  {
  CDocument::Dump(dc);
  }
#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
//////
// CNETPHONEDoc commands

void CNETPHONEDoc::OnStart()
  {

  if(m_pSTARTThread==NULL)
    {
    m_pSTARTThread=(STARTThread*)AfxBeginThread(RUNTIME_CLASS
(STARTThread),THREAD_PRIORITY_NORMAL,0,CREATE_SUSPENDED);
    m_pSTARTThread->SetOwner(this);
    m_pSTARTThread->ResumeThread();
    }
  }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
void CNETPHONEDoc::Send()
  {
  if(m_pRECORDThread==NULL)
    {
    m_pRECORDThread=(RECORDThread*)AfxBeginThread(RUNTIME_CLASS
(RECORDThread),THREAD_PRIORITY_NORMAL,0,CREATE_SUSPENDED);
    m_pRECORDThread->SetOwner(this);
    m_pRECORDThread->ResumeThread();
    }
  }

void CNETPHONEDoc::OnSend()
  {
  IPDialog ip;
  if(ip.DoModal()==IDOK)
    {
    if(ipaddress.IsEmpty()==0)
      {
      m_pRECORDThread=(RECORDThread*)AfxBeginThread(RUNTIME_CLASS
(RECORDThread),THREAD_PRIORITY_NORMAL,0,CREATE_SUSPENDED);
      m_pRECORDThread->SetOwner(this);
      m_pRECORDThread->ResumeThread();
      }
    }
  }

void CNETPHONEDoc::OnUpdateSend(CCmdUI* pCmdUI)
  {
  pCmdUI->Enable(m_pRECORDThread==NULL);
  }


BYTE CNETPHONEDoc::Inport(unsigned short port)
  {
  BYTE result;
  _asm
    {
    push dx
    mov dx,port
    in al,dx
    mov result,al
    pop dx
    }
  return(result);
  }

void CNETPHONEDoc::Outport(unsigned short port,BYTE data)
  {
  _asm
    {
    push dx
    mov dx,port
    mov al,data
    out dx,al
    pop dx
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
// NETPHONE.cpp : Defines the class behaviors for the application.
//

#include "stdafx.h"
#include "NETPHONE.h"

#include "MainFrm.h"
#include "NETPHONEDoc.h"
#include "NETPHONEView.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
//////
// CNETPHONEApp

BEGIN_MESSAGE_MAP(CNETPHONEApp, CWinApp)
  //{{AFX_MSG_MAP(CNETPHONEApp)
  ON_COMMAND(ID_APP_ABOUT, OnAppAbout)
  //}}AFX_MSG_MAP
  // Standard file based document commands
  ON_COMMAND(ID_FILE_NEW, CWinApp::OnFileNew)
  ON_COMMAND(ID_FILE_OPEN, CWinApp::OnFileOpen)
  // Standard print setup command
  ON_COMMAND(ID_FILE_PRINT_SETUP, CWinApp::OnFilePrintSetup)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
//////
// CNETPHONEApp construction


CNETPHONEApp::CNETPHONEApp()
  {
  }

/////////////////////////////////////////////////////////////////////////////
//////
// The one and only CNETPHONEApp object

CNETPHONEApp theApp;

/////////////////////////////////////////////////////////////////////////////
//////
// CNETPHONEApp initialization

BOOL CNETPHONEApp::InitInstance()
  {
  if(!AfxSocketInit())
    {
    AfxMessageBox(IDP_SOCKETS_INIT_FAILED);
    return(FALSE);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    }

  AfxEnableControlContainer();

  // Standard initialization
  // If you are not using these features and wish to reduce the size
  //  of your final executable, you should remove from the following
  //  the specific initialization routines you do not need.

  #ifdef _AFXDLL
     Enable3dControls();        // Call this when using MFC in a shared
DLL
  #else
     Enable3dControlsStatic(); // Call this when linking to MFC
statically
  #endif


  // Change the registry key under which our settings are stored.
  // You should modify this string to be something appropriate
  // such as the name of your company or organization.
  SetRegistryKey(_T("Local AppWizard-Generated Applications"));

  LoadStdProfileSettings(); // Load standard INI file options
(including MRU)

  // Register the application's document templates.  Document templates
  //  serve as the connection between documents, frame windows and
views.

  CSingleDocTemplate* pDocTemplate;
  pDocTemplate = new CSingleDocTemplate(IDR_MAINFRAME,
                                        RUNTIME_CLASS(CNETPHONEDoc),
                                        RUNTIME_CLASS(CMainFrame),
                                        RUNTIME_CLASS(CNETPHONEView));
  AddDocTemplate(pDocTemplate);

  // Parse command line for standard shell commands, DDE, file open
  CCommandLineInfo cmdInfo;
  ParseCommandLine(cmdInfo);

  // Dispatch commands specified on the command line
  if(!ProcessShellCommand(cmdInfo)) return(FALSE);

  // The one and only window has been initialized, so show and update
it.
  m_pMainWnd->ShowWindow(SW_SHOW);
  m_pMainWnd->UpdateWindow();
  return(TRUE);
  }

////////////////////////////////////////////////////////////////////////////
//////
// CAboutDlg dialog used for App About
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
class CAboutDlg : public CDialog
  {
  public:
    CAboutDlg();

  // Dialog Data
    //{{AFX_DATA(CAboutDlg)
    enum { IDD=IDD_ABOUTBOX };
    //}}AFX_DATA

  // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CAboutDlg)
  protected:
    virtual void DoDataExchange(CDataExchange* pDX); // DDX/DDV support
    //}}AFX_VIRTUAL

  // Implementation
  protected:
    //{{AFX_MSG(CAboutDlg)
    //}}AFX_MSG
  DECLARE_MESSAGE_MAP()
  };

CAboutDlg::CAboutDlg():CDialog(CAboutDlg::IDD)
  {
  //{{AFX_DATA_INIT(CAboutDlg)
  //}}AFX_DATA_INIT
  }

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
  {
  CDialog::DoDataExchange(pDX);
  //{{AFX_DATA_MAP(CAboutDlg)

  //}}AFX_DATA_MAP
  }

BEGIN_MESSAGE_MAP(CAboutDlg,CDialog)
  //{{AFX_MSG_MAP(CAboutDlg)
  //}}AFX_MSG_MAP
END_MESSAGE_MAP()

// App command to run the dialog
void CNETPHONEApp::OnAppAbout()
  {
  CAboutDlg aboutDlg;
  aboutDlg.DoModal();
  }

/////////////////////////////////////////////////////////////////////////////
//////
// CNETPHONEApp commands
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
//=============================================================================
// PLAYThread.cpp : implementation file
//-----------------------------------------------------------------------------

#include "stdafx.h"
#include "NETPHONE.h"
#include "NETPHONEDoc.h"
#include "PLAYThread.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

#define NUMBUFFER 8

extern int       nchannels;      // number of channel
extern int       samplepersec;   // sampling frequency
extern int       bitspersample;  // bits per sample
extern int       samplesize;     // bytes per sample
extern int       bytesperbuffer; // bytes per buffer
extern MMRESULT  result;
extern char      data[NUMBUFFER+2][8000];

extern HANDLE     hEventOut[2];
extern HANDLE     hEventDoneOut;
extern HANDLE     hEventDeadOut;
extern HANDLE     hEventKillOut;
extern SoundSocket receivesocket;

int          iBufPlayed;
int          iBufOut,prevBufOut;

HWAVEOUT     handleOut;
WAVEHDR      waveheaderOut[NUMBUFFER];
WAVEFORMATEX outputwaveformat;

/////////////////////////////////////////////////////////////////////////////
// PLAYThread

IMPLEMENT_DYNCREATE(PLAYThread,CWinThread)

PLAYThread::PLAYThread()
  {
  m_pOwner=NULL;
  iBufPlayed=0;
  }

PLAYThread::~PLAYThread()
  {
  waveOutReset(handleOut);
  for(int i=0;i<NUMBUFFER;i++)
if(waveheaderOut[i].dwFlags&WHDR_PREPARED) waveOutUnprepareHeader
(handleOut,&waveheaderOut[i],sizeof(WAVEHDR));
  waveOutClose(handleOut);
  m_pOwner->m_pPLAYThread=NULL;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
}

BOOL PLAYThread::InitInstance()
{
return TRUE;
}

int PLAYThread::ExitInstance()
{
return CWinThread::ExitInstance();
}

BEGIN_MESSAGE_MAP(PLAYThread,CWinThread)
//{{AFX_MSG_MAP(PLAYThread)
//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// PLAYThread message handlers

int PLAYThread::Run()
{
outputwaveformat.wFormatTag      =WAVE_FORMAT_PCM;
outputwaveformat.nChannels       =nchannels;
outputwaveformat.nSamplesPerSec =samplepersec;
outputwaveformat.nAvgBytesPerSec=samplepersec*nchannels*
(bitspersample/8);
outputwaveformat.nBlockAlign     =samplesize;
outputwaveformat.wBitsPerSample =bitspersample;
outputwaveformat.cbSize          =0;

result=waveOutOpen(&handleOut,WAVE_MAPPER,&outputwaveformat,DWORD
(waveOutProc),0,CALLBACK_FUNCTION);
switch(result)
  {
  case MMSYSERR_NOERROR: break;
  case MMSYSERR_ALLOCATED: AfxMessageBox("Specified resource is
already allocated."); return(FALSE); break;
  case MMSYSERR_BADDEVICEID: AfxMessageBox("Specified device
identifier is out of range."); return(FALSE); break;
  case MMSYSERR_NODRIVER: AfxMessageBox("No device driver is
present."); return(FALSE); break;
  case MMSYSERR_NOMEM: AfxMessageBox("Unable to allocate or lock
memory."); return(FALSE); break;
  case WAVERR_BADFORMAT: AfxMessageBox("Attempted to open with an
unsupported waveform-audio format."); return(FALSE); break;
  case WAVERR_SYNC: AfxMessageBox("The device is synchronous but
waveOutOpen was called without using the WAVE_ALLOWSYNC flag."); return
(FALSE); break;
  }
WaitForSingleObject(hEventKillOut,INFINITE);
ClearUpThread();
SetEvent(hEventDeadOut);
return CWinThread::Run();
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
LRESULT CALLBACK PLAYThread::waveOutProc(HWAVEOUT hwi,UINT uMsg,DWORD
dwInstance,DWORD dwParam1,DWORD dwParam2)
  {
  if(uMsg==WOM_DONE)
    {

waveOutUnprepareHeader(handleOut,&(waveheaderOut[iBufPlayed]),sizeof
(WAVEHDR));
    iBufPlayed++; if(iBufPlayed==NUMBUFFER) iBufPlayed=0;
    }
  return(0);
  }

void PLAYThread::KillThread()
  {
  SetEvent(hEventKillOut);
  SetThreadPriority(THREAD_PRIORITY_HIGHEST);
  WaitForSingleObject(hEventDeadOut,1000);
  }

void PLAYThread::ClearUpThread()
  {
  m_pOwner->m_pPLAYThread=NULL;
  AfxEndThread(0);
  }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
// SoundSocket.cpp : implementation file
//

#include "stdafx.h"
#include "NETPHONE.h"
#include "SoundSocket.h"
#include "Netphonedoc.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

#define NUMBUFFER 8

extern int nchannels;      // number of channel
extern int samplepersec;   // sampling frequency
extern int bitspersample;  // bits per sample
extern int samplesize;     // bytes per sample
extern int bytesperbuffer; // bytes per buffer
extern CString ipaddress;

extern int          bytesperbuffer;
extern int          iBufOut,prevBufOut;
extern char         *pBufferOut;
extern HWAVEOUT     handleOut;
extern WAVEHDR      waveheaderOut[NUMBUFFER];
extern WAVEFORMATEX outputwaveformat;

char     data[NUMBUFFER+2][8000];
MMRESULT resultOut;


/////////////////////////////////////////////////////////////////////////////
// SoundSocket

SoundSocket::SoundSocket()
  {
  iBufOut=0;
  }

SoundSocket::~SoundSocket()
  {
  }

// Do not edit the following lines, which are needed by ClassWizard.
#if 0
BEGIN_MESSAGE_MAP(SoundSocket,CAsyncSocket)
  //{{AFX_MSG_MAP(SoundSocket)
  //}}AFX_MSG_MAP
END_MESSAGE_MAP()
#endif // 0

/////////////////////////////////////////////////////////////////////////////
// SoundSocket member functions
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
void SoundSocket::OnReceive(int nErrorCode)
  {
  static BOOL messagebox=FALSE;
  static int nReceive;
  static char text[100];
  static CWnd *window=AfxGetMainWnd();

  if(nErrorCode==0)
    {
    if(m_pOwner->m_pRECORDThread==NULL)

      {
      UINT ipport;


nReceive=ReceiveFrom(data[iBufOut],bytesperbuffer,ipaddress,ipport);

      m_pOwner->m_pRECORDThread=(RECORDThread *)AfxBeginThread
(RUNTIME_CLASS(RECORDThread),THREAD_PRIORITY_NORMAL,0,CREATE_SUSPENDED)
;
      m_pOwner->m_pRECORDThread->SetOwner(m_pOwner);
      m_pOwner->m_pRECORDThread->ResumeThread();
      }
    else nReceive=Receive(data[iBufOut],bytesperbuffer);

    if(nReceive==bytesperbuffer)
      {
      waveheaderOut[iBufOut].lpData          =data[iBufOut];
      waveheaderOut[iBufOut].dwBufferLength=bytesperbuffer;
      waveheaderOut[iBufOut].dwFlags         =0;
      waveheaderOut[iBufOut].dwLoops         =0;
      waveOutPrepareHeader(handleOut,&waveheaderOut[iBufOut],sizeof
(WAVEHDR));
      waveOutWrite(handleOut,&waveheaderOut[iBufOut],sizeof(WAVEHDR));
      sprintf(text,"Bytes receive : %d [%d]",nReceive,iBufOut);
      window->SetWindowText(text);
      iBufOut++; if(iBufOut==NUMBUFFER) iBufOut=0;
      }
    }
   CAsyncSocket::OnReceive(nErrorCode);
   }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
// STARTThread.cpp : implementation file
//

#include "stdafx.h"
#include "netphone.h"
#include "STARTThread.h"
#include "Netphonedoc.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

extern CString ipaddress;
extern HANDLE  hEventDeadStart;
extern HANDLE  hEventKillStart;

/////////////////////////////////////////////////////////////////////////////
// STARTThread

IMPLEMENT_DYNCREATE(STARTThread, CWinThread)

STARTThread::STARTThread()
  {
  }

STARTThread::~STARTThread()
  {
  }

BOOL STARTThread::InitInstance()
  {

  return TRUE;
  }

int STARTThread::ExitInstance()
  {
  return CWinThread::ExitInstance();
  }

BEGIN_MESSAGE_MAP(STARTThread,CWinThread)
  //{{AFX_MSG_MAP(STARTThread)
  //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// STARTThread message handlers

int STARTThread::Run()
  {
  int  i=0,j;
  char text[100];
  BYTE input0,input1,select;
  static CWnd *pStatusbar=AfxGetMainWnd()->GetDescendantWindow
 (AFX_IDW_STATUS_BAR);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
m_pOwner->Outport(0x241,0x00);
m_pOwner->Outport(0x243,0x99);

for(;;)
  {
  input0=m_pOwner->Inport(0x242);
  if(input0==255)
    {
    select=0;


    // play wave

    for(j=0;j<1;j++)
      {
      m_pOwner->Outport(0x241,0x01);
      sprintf(text,"Waiting input digit : %d",j+1);
      AfxGetMainWnd()->SetWindowText(text);

      for(;;)
        {
        input1=m_pOwner->Inport(0x240);
        if(input1&0x0010==0x0010)
          {
          input1&=0x000F;
          if(j!=0) select+=((j*10)*input1);
          else     select+=input1;
          sprintf(text,"%d digit : %d",j+1,select);
          pStatusbar->SetWindowText(text);
          break;
          }
        else
          {
          sprintf(text,"%d digit : %d",j+1,select);
          pStatusbar->SetWindowText(text);
          }
       Sleep(333);

       if(WaitForSingleObject(hEventKillStart,0)==WAIT_OBJECT_0)
          {
          ClearUpThread();
          SetEvent(hEventDeadStart);
          return CWinThread::Run();
          }

        }
      }
     switch(select)
       {
       case 0: ipaddress="161.246.14.180"; break;
       case 1: ipaddress="161.246.14.200"; break;
       default: ipaddress="161.246.14.200"; break;
       }
     m_pOwner->Send();
     ClearUpThread();
     SetEvent(hEventDeadStart);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
      return CWinThread::Run();
      }
    if(WaitForSingleObject(hEventKillStart,0)==WAIT_OBJECT_0)
      {
      ClearUpThread();
      SetEvent(hEventDeadStart);
      return CWinThread::Run();
      }
    Sleep(500);
    sprintf(text,"Waiting input phone : %d second",i++);
    AfxGetMainWnd()->SetWindowText(text);
    }
  return CWinThread::Run();
  }

void STARTThread::KillThread()
  {
  SetEvent(hEventKillStart);                    // reset the hEventKill
which signals the thread to shutdown
  SetThreadPriority(THREAD_PRIORITY_HIGHEST); // allow thread to run at
highest priority during kill process
  WaitForSingleObject(hEventDeadStart,1000);
  }

void STARTThread::ClearUpThread()
  {
  m_pOwner->m_pSTARTThread=NULL;
  AfxEndThread(0);
  }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
//=============================================================================
// THREAD.cpp : implementation file
//-----------------------------------------------------------------------------

#include "stdafx.h"
#include "THREAD.h"
#include "resource.h"
#include "netphonedoc.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

#define NUMBUFFER 8

int          iBufIn,prevBufIn;
int          nchannels;      // number of channel
int          samplepersec;   // sampling frequency
int          bitspersample;  // bits per sample
int          samplesize;     // bytes per sample
int          bytesperbuffer; // bytes per buffer
char         *pBufferIn;
HWAVEIN      handleIn;
MMRESULT     result;
WAVEHDR      waveheaderIn[NUMBUFFER];
WAVEFORMATEX inputwaveformat;
CString      ipaddress;

extern HANDLE      hEventIn[2];
extern HANDLE      hEventDoneIn;
extern HANDLE      hEventDeadIn;

extern HANDLE      hEventKillIn;
extern SoundSocket receivesocket;

BEGIN_MESSAGE_MAP(RECORDThread,CWinThread)
  //{{AFX_MSG_MAP(RECORDThread)
  //}}AFX_MSG_MAP
END_MESSAGE_MAP()

//////////////////////////////////////////////////////////////////////////////
// RECORDThread

IMPLEMENT_DYNCREATE(RECORDThread,CWinThread)

RECORDThread::RECORDThread()
  {
  m_pOwner=NULL;
  pBufferIn=NULL;
  }

RECORDThread::~RECORDThread()
  {
  waveInReset(handleIn);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
  for(int i=0;i<NUMBUFFER;i++)
if(waveheaderIn[i].dwFlags&WHDR_PREPARED) waveInUnprepareHeader
(handleIn,&waveheaderIn[i],sizeof(WAVEHDR));
  waveInClose(handleIn);
  if(pBufferIn!=NULL) {free(pBufferIn); pBufferIn=NULL;}
  m_pOwner->m_pRECORDThread=NULL;
  }

BOOL RECORDThread::InitInstance()
  {
  return(TRUE);
  }

int RECORDThread::ExitInstance()
  {
  return CWinThread::ExitInstance();
  }

/////////////////////////////////////////////////////////////////////////
// RECORDThread message handlers

int RECORDThread::Run()
  {
  CWnd *pStatusbar=AfxGetMainWnd()-
>GetDescendantWindow(AFX_IDW_STATUS_BAR);

  if(m_pOwner==NULL) return(-1);

  if(waveInGetNumDevs()==0) {AfxMessageBox("No sound card installed!");
return(FALSE);}

  inputwaveformat.wFormatTag      =WAVE_FORMAT_PCM;
  inputwaveformat.nChannels       =nchannels;
  inputwaveformat.nSamplesPerSec =samplepersec;
  inputwaveformat.nAvgBytesPerSec=samplepersec*nchannels*
(bitspersample/8);
  inputwaveformat.nBlockAlign     =samplesize;
  inputwaveformat.wBitsPerSample =bitspersample;
  inputwaveformat.cbSize          =0;


result=waveInOpen(&handleIn,0,&inputwaveformat,DWORD(waveInProc),0,CALL
BACK_FUNCTION);
  switch(result)
    {
    case MMSYSERR_NOERROR: break;
    case MMSYSERR_ALLOCATED: AfxMessageBox("Specified resource is
already allocated."); return(FALSE); break;
    case MMSYSERR_BADDEVICEID: AfxMessageBox("Specified device
identifier is out of range."); return(FALSE); break;
    case MMSYSERR_NODRIVER: AfxMessageBox("No device driver is
present."); return(FALSE); break;
    case MMSYSERR_NOMEM: AfxMessageBox("Unable to allocate or lock
memory."); return(FALSE); break;
    case WAVERR_BADFORMAT: AfxMessageBox("Attempted to open with an
unsupported waveform-audio format."); return(FALSE); break;
    }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
  pBufferIn=(char *)malloc(NUMBUFFER*bytesperbuffer);
  memset(pBufferIn,0,NUMBUFFER*bytesperbuffer);
  if(pBufferIn==NULL)
     {
     ClearUpThread();
     AfxMessageBox("Can't allocate memory for recorder buffer.");
     return(FALSE);
     }

  for(int i=0;i<NUMBUFFER;i++)
    {
    waveheaderIn[i].lpData         =&pBufferIn[i*bytesperbuffer];
    waveheaderIn[i].dwBufferLength=bytesperbuffer;
    waveheaderIn[i].dwFlags        =0;
    waveheaderIn[i].dwLoops        =0;
    waveInPrepareHeader(handleIn,&waveheaderIn[i],sizeof(WAVEHDR));
    waveInAddBuffer(handleIn,&waveheaderIn[i],sizeof(WAVEHDR));
    }
  iBufIn=0;
  waveInStart(handleIn);

  for(;;)
    {
    switch(WaitForMultipleObjects(2,hEventIn,FALSE,INFINITE)-
WAIT_OBJECT_0)
      {
      case 0: waveInStop(handleIn); ClearUpThread(); SetEvent
(hEventDeadIn); return(0); break;
      case 1: for(int i=0;i<NUMBUFFER;i++)
                {

                waveInUnprepareHeader(handleIn,&(waveheaderIn
[i]),sizeof(WAVEHDR));
                waveheaderIn[i].lpData         =&pBufferIn
[i*bytesperbuffer];
                waveheaderIn[i].dwBufferLength=bytesperbuffer;
                waveheaderIn[i].dwFlags        =0;
                waveheaderIn[i].dwLoops        =0;
                waveInPrepareHeader(handleIn,&waveheaderIn[i],sizeof
(WAVEHDR));
                waveInAddBuffer(handleIn,&waveheaderIn[i],sizeof
(WAVEHDR));
                }
              iBufIn=0;
              break;
      }
    }
  return(0);
  }

LRESULT CALLBACK RECORDThread::waveInProc(HWAVEIN hwi,UINT uMsg,DWORD
dwInstance,DWORD dwParam1,DWORD dwParam2)
  {
  static char text[100];
  static CWnd *pStatusbar=AfxGetMainWnd()->GetDescendantWindow
(AFX_IDW_STATUS_BAR);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
  if(uMsg==WIM_DATA)
    {
    receivesocket.SendTo(waveheaderIn[iBufIn].lpData,waveheaderIn
[iBufIn].dwBufferLength,2000,LPCTSTR(ipaddress));
    waveInUnprepareHeader(handleIn,&(waveheaderIn[iBufIn]),sizeof
(WAVEHDR));

    sprintf(text,"iBufIn : %d",iBufIn);
    pStatusbar->SetWindowText(text);

    iBufIn++; if(iBufIn==NUMBUFFER) {SetEvent(hEventDoneIn); iBufIn=0;}
    }

  return(0);
  }

void RECORDThread::KillThread()
  {
  SetEvent(hEventKillIn);                    // reset the hEventKill
which signals the thread to shutdown
  SetThreadPriority(THREAD_PRIORITY_HIGHEST); // allow thread to run at
highest priority during kill process
  WaitForSingleObject(hEventDeadIn,1000);
  }

void RECORDThread::ClearUpThread()
  {
  m_pOwner->m_pRECORDThread=NULL;
  AfxEndThread(0);
  }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
// stdafx.cpp : source file that includes just the standard includes
//    NETPHONE.pch will be the pre-compiled header
//    stdafx.obj will contain the pre-compiled type information

#include "stdafx.h"
```


เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
// NETPHONEView.cpp : implementation of the CNETPHONEView class
//

#include "stdafx.h"
#include "NETPHONE.h"

#include "NETPHONEDoc.h"
#include "NETPHONEView.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
//////
// CNETPHONEView

IMPLEMENT_DYNCREATE(CNETPHONEView,CView)

BEGIN_MESSAGE_MAP(CNETPHONEView,CView)
  //{{AFX_MSG_MAP(CNETPHONEView)
  //}}AFX_MSG_MAP
  // Standard printing commands
  ON_COMMAND(ID_FILE_PRINT,          CView::OnFilePrint)
  ON_COMMAND(ID_FILE_PRINT_DIRECT, CView::OnFilePrint)
  ON_COMMAND(ID_FILE_PRINT_PREVIEW,CView::OnFilePrintPreview)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
//////
// CNETPHONEView construction/destruction

CNETPHONEView::CNETPHONEView()

  {
  }

CNETPHONEView::~CNETPHONEView()
  {
  }

BOOL CNETPHONEView::PreCreateWindow(CREATESTRUCT& cs)
  {
  return CView::PreCreateWindow(cs);
  }

/////////////////////////////////////////////////////////////////////////////
//////
// CNETPHONEView drawing

void CNETPHONEView::OnDraw(CDC* pDC)
  {
  CNETPHONEDoc* pDoc=GetDocument();
  ASSERT_VALID(pDoc);
  }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
////////////////////////////////////////////////////////////////////////
//////
// CNETPHONEView printing

BOOL CNETPHONEView::OnPreparePrinting(CPrintInfo* pInfo)
  {
  // default preparation
  return DoPreparePrinting(pInfo);
  }

void CNETPHONEView::OnBeginPrinting(CDC* /*pDC*/, CPrintInfo*
/*pInfo*/)
  {
  }

void CNETPHONEView::OnEndPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
  {
  }

////////////////////////////////////////////////////////////////////////
//////
// CNETPHONEView diagnostics

#ifdef _DEBUG
void CNETPHONEView::AssertValid() const
  {
  CView::AssertValid();
  }

void CNETPHONEView::Dump(CDumpContext& dc) const
  {
  CView::Dump(dc);
  }

CNETPHONEDoc* CNETPHONEView::GetDocument() // non-debug version is
inline
  {
  ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CNETPHONEDoc)));
  return (CNETPHONEDoc*)m_pDocument;
  }
#endif //_DEBUG

////////////////////////////////////////////////////////////////////////
//////
// CNETPHONEView message handlers
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
// NETPHONEDoc.cpp : implementation of the CNETPHONEDoc class
//

#include "stdafx.h"
#include "NETPHONE.h"
#include "NETPHONEDoc.h"
#include "soundsocket.h"
#include "IPDialog.h"
#include <conio.h>

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

extern int     nchannels;      // number of channel
extern int     samplepersec;   // sampling frequency
extern int     bitspersample;  // bits per sample
extern int     samplesize;     // bytes per sample
extern int     bytesperbuffer; // bytes per buffer
extern CString ipaddress;

HANDLE      hEventDeadStart;
HANDLE      hEventKillStart;

HANDLE      hEventIn[2];
HANDLE      hEventDoneIn;
HANDLE      hEventDeadIn;
HANDLE      hEventKillIn;

HANDLE      hEventOut[2];
HANDLE      hEventDoneOut;

HANDLE      hEventDeadOut;
HANDLE      hEventKillOut;

SoundSocket receivesocket;

/////////////////////////////////////////////////////////////////////////////
//////
// CNETPHONEDoc

IMPLEMENT_DYNCREATE(CNETPHONEDoc,CDocument)

BEGIN_MESSAGE_MAP(CNETPHONEDoc,CDocument)
  //{{AFX_MSG_MAP(CNETPHONEDoc)
  ON_COMMAND(ID_SEND,OnSend)
  ON_COMMAND(ID_START,OnStart)
  ON_UPDATE_COMMAND_UI(ID_SEND,OnUpdateSend)
  //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
//////
// CNETPHONEDoc construction/destruction
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
CNETPHONEDoc::CNETPHONEDoc()
  {
  nchannels=1;
  bitspersample=8;
  samplepersec=8000;
  samplesize=nchannels*(bitspersample/8);
  bytesperbuffer=(samplesize*samplepersec)/4;

  ipaddress.Empty();

  if(!receivesocket.Create(2000,SOCK_DGRAM)) AfxMessageBox("Can't
create receivesocket");

  hEventDeadStart=CreateEvent(NULL,FALSE,FALSE,NULL);
  hEventKillStart=CreateEvent(NULL,FALSE,FALSE,NULL);

  hEventDoneIn=CreateEvent(NULL,FALSE,FALSE,NULL);
  hEventDeadIn=CreateEvent(NULL,FALSE,FALSE,NULL);
  hEventKillIn=CreateEvent(NULL,FALSE,FALSE,NULL);

  hEventDoneOut=CreateEvent(NULL,FALSE,FALSE,NULL);
  hEventDeadOut=CreateEvent(NULL,FALSE,FALSE,NULL);
  hEventKillOut=CreateEvent(NULL,FALSE,FALSE,NULL);

  hEventIn[0]=hEventKillIn;
  hEventIn[1]=hEventDoneIn;

  hEventOut[0]=hEventKillOut;
  hEventOut[1]=hEventDoneOut;

  m_pSTARTThread=NULL;
  m_pRECORDThread=NULL;
  m_pPLAYThread=NULL;

  m_pPLAYThread=(PLAYThread*)AfxBeginThread(RUNTIME_CLASS
(PLAYThread),THREAD_PRIORITY_NORMAL,0,CREATE_SUSPENDED);
  m_pPLAYThread->SetOwner(this);
  m_pPLAYThread->ResumeThread();

  receivesocket.SetOwner(this);
  }

CNETPHONEDoc::~CNETPHONEDoc()
  {
  receivesocket.Close();

  if(m_pSTARTThread!=NULL) m_pSTARTThread->KillThread();
  if(m_pRECORDThread!=NULL) m_pRECORDThread->KillThread();
  if(m_pPLAYThread!=NULL) m_pPLAYThread->KillThread();

  CloseHandle(hEventDeadStart);
  CloseHandle(hEventKillStart);

  CloseHandle(hEventDoneIn);
  CloseHandle(hEventDeadIn);
  CloseHandle(hEventKillIn);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
  CloseHandle(hEventDoneOut);
  CloseHandle(hEventDeadOut);
  CloseHandle(hEventKillOut);
  }

BOOL CNETPHONEDoc::OnNewDocument()
  {
  if(!CDocument::OnNewDocument()) return(FALSE);
  return(TRUE);
  }

/////////////////////////////////////////////////////////////////////////////
//////
// CNETPHONEDoc serialization

void CNETPHONEDoc::Serialize(CArchive& ar)
  {
  if(ar.IsStoring())
    {
    }
  else
    {
    }
  }

/////////////////////////////////////////////////////////////////////////////
//////
// CNETPHONEDoc diagnostics

#ifdef _DEBUG
void CNETPHONEDoc::AssertValid() const
  {
  CDocument::AssertValid();
  }

void CNETPHONEDoc::Dump(CDumpContext& dc) const
  {
  CDocument::Dump(dc);
  }
#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
//////
// CNETPHONEDoc commands

void CNETPHONEDoc::OnStart()
  {

  if(m_pSTARTThread==NULL)
    {
    m_pSTARTThread=(STARTThread*)AfxBeginThread(RUNTIME_CLASS
(STARTThread),THREAD_PRIORITY_NORMAL,0,CREATE_SUSPENDED);
    m_pSTARTThread->SetOwner(this);
    m_pSTARTThread->ResumeThread();
    }
  }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
void CNETPHONEDoc::Send()
  {
  if(m_pRECORDThread==NULL)
    {
    m_pRECORDThread=(RECORDThread*)AfxBeginThread(RUNTIME_CLASS
(RECORDThread),THREAD_PRIORITY_NORMAL,0,CREATE_SUSPENDED);
    m_pRECORDThread->SetOwner(this);
    m_pRECORDThread->ResumeThread();
    }
  }

void CNETPHONEDoc::OnSend()
  {
  IPDialog ip;
  if(ip.DoModal()==IDOK)
    {
    if(ipaddress.IsEmpty()==0)
      {
      m_pRECORDThread=(RECORDThread*)AfxBeginThread(RUNTIME_CLASS
(RECORDThread),THREAD_PRIORITY_NORMAL,0,CREATE_SUSPENDED);
      m_pRECORDThread->SetOwner(this);
      m_pRECORDThread->ResumeThread();
      }
    }
  }

void CNETPHONEDoc::OnUpdateSend(CCmdUI* pCmdUI)
  {
  pCmdUI->Enable(m_pRECORDThread==NULL);
  }


BYTE CNETPHONEDoc::Inport(unsigned short port)
  {
  BYTE result;
  _asm
    {
    push dx
    mov dx,port
    in al,dx
    mov result,al
    pop dx
    }
  return(result);
  }

void CNETPHONEDoc::Outport(unsigned short port,BYTE data)
  {
  _asm
    {
    push dx
    mov dx,port
    mov al,data
    out dx,al
    pop dx
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
// NETPHONE.cpp : Defines the class behaviors for the application.
//

#include "stdafx.h"
#include "NETPHONE.h"

#include "MainFrm.h"
#include "NETPHONEDoc.h"
#include "NETPHONEView.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
//
// CNETPHONEApp

BEGIN_MESSAGE_MAP(CNETPHONEApp, CWinApp)
  //{{AFX_MSG_MAP(CNETPHONEApp)
  ON_COMMAND(ID_APP_ABOUT, OnAppAbout)
  //}}AFX_MSG_MAP
  // Standard file based document commands
  ON_COMMAND(ID_FILE_NEW, CWinApp::OnFileNew)
  ON_COMMAND(ID_FILE_OPEN, CWinApp::OnFileOpen)
  // Standard print setup command
  ON_COMMAND(ID_FILE_PRINT_SETUP, CWinApp::OnFilePrintSetup)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
//
// CNETPHONEApp construction


CNETPHONEApp::CNETPHONEApp()
  {
  }

/////////////////////////////////////////////////////////////////////////////
//
// The one and only CNETPHONEApp object

CNETPHONEApp theApp;

/////////////////////////////////////////////////////////////////////////////
//
// CNETPHONEApp initialization

BOOL CNETPHONEApp::InitInstance()
  {
  if(!AfxSocketInit())
    {
    AfxMessageBox(IDP_SOCKETS_INIT_FAILED);
    return(FALSE);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    }

    AfxEnableControlContainer();

    // Standard initialization
    // If you are not using these features and wish to reduce the size
    //  of your final executable, you should remove from the following
    //  the specific initialization routines you do not need.

    #ifdef _AFXDLL
        Enable3dControls();       // Call this when using MFC in a shared
DLL
    #else
        Enable3dControlsStatic(); // Call this when linking to MFC
statically
    #endif


    // Change the registry key under which our settings are stored.
    // You should modify this string to be something appropriate
    // such as the name of your company or organization.
    SetRegistryKey(_T("Local AppWizard-Generated Applications"));

    LoadStdProfileSettings(); // Load standard INI file options
(including MRU)

    // Register the application's document templates.  Document templates
    //  serve as the connection between documents, frame windows and
views.

    CSingleDocTemplate* pDocTemplate;
    pDocTemplate = new CSingleDocTemplate(IDR_MAINFRAME,
                                          RUNTIME_CLASS(CNETPHONEDoc),
                                          RUNTIME_CLASS(CMainFrame),
                                          RUNTIME_CLASS(CNETPHONEView));
    AddDocTemplate(pDocTemplate);

    // Parse command line for standard shell commands, DDE, file open
    CCommandLineInfo cmdInfo;
    ParseCommandLine(cmdInfo);

    // Dispatch commands specified on the command line
    if(!ProcessShellCommand(cmdInfo)) return(FALSE);

    // The one and only window has been initialized, so show and update
it.
    m_pMainWnd->ShowWindow(SW_SHOW);
    m_pMainWnd->UpdateWindow();
    return(TRUE);
    }

////////////////////////////////////////////////////////////////////////
//////
// CAboutDlg dialog used for App About
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
class CAboutDlg : public CDialog
  {
  public:
    CAboutDlg();

  // Dialog Data
    //{{AFX_DATA(CAboutDlg)
    enum { IDD=IDD_ABOUTBOX };
    //}}AFX_DATA

  // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CAboutDlg)
  protected:
    virtual void DoDataExchange(CDataExchange* pDX); // DDX/DDV support
    //}}AFX_VIRTUAL

  // Implementation
  protected:
    //{{AFX_MSG(CAboutDlg)
    //}}AFX_MSG
  DECLARE_MESSAGE_MAP()
  };

CAboutDlg::CAboutDlg():CDialog(CAboutDlg::IDD)
  {
  //{{AFX_DATA_INIT(CAboutDlg)
  //}}AFX_DATA_INIT
  }

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
  {
  CDialog::DoDataExchange(pDX);
  //{{AFX_DATA_MAP(CAboutDlg)

  //}}AFX_DATA_MAP
  }

BEGIN_MESSAGE_MAP(CAboutDlg,CDialog)
  //{{AFX_MSG_MAP(CAboutDlg)
  //}}AFX_MSG_MAP
END_MESSAGE_MAP()

// App command to run the dialog
void CNETPHONEApp::OnAppAbout()
  {
  CAboutDlg aboutDlg;
  aboutDlg.DoModal();
  }

/////////////////////////////////////////////////////////////////////////////
//////
// CNETPHONEApp commands
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

MITEL®

Application Note MSAN-10
Applications of The MT887
Integrated DTMF Receive

ISSUE 1 June 198

Contents

Introduction

The purpose of this Application Note is to provide information on the operation and application of DTMF Receivers. The MT8870 Integrated DTMF Receiver will be discussed in detail and its use illustrated in the application examples which follow.

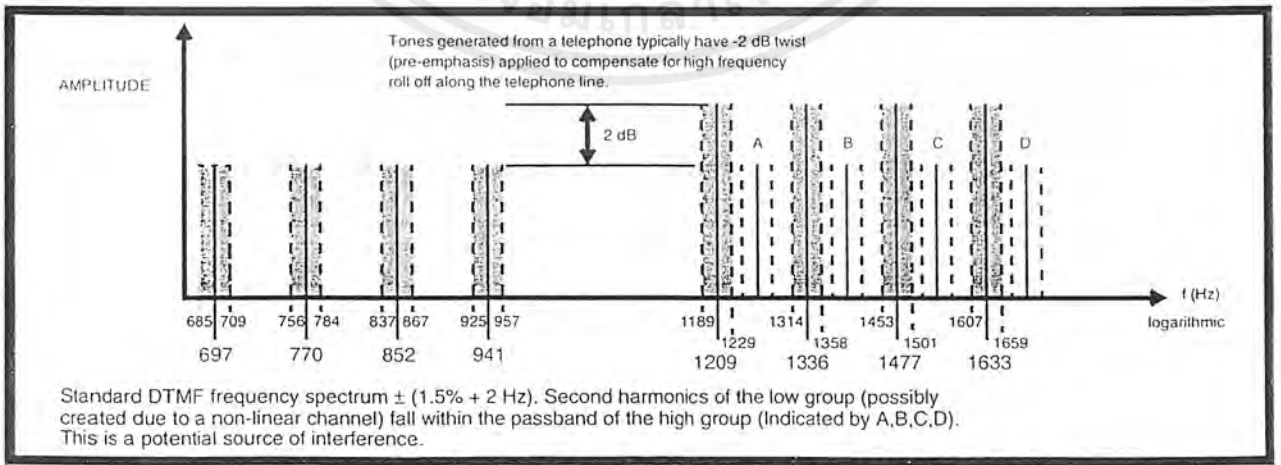
More than 25 years ago the need for an improved method for transferring dialling information through the telephone network was recognized. The traditional method, Dial pulse signalling, was not only slow, suffering severe distortion over long wire loops, but required a DC path through the communications channel. A signalling scheme was developed utilizing voice frequency tones and implemented as a very reliable alternative to pulse dialling. This scheme is known as DTMF (Dual Tone Multi-Frequency), Touch-Tone™ or simply, tone dialling. As its acronym suggests, a valid DTMF signal is the sum of two tones, one from a low group (697-941Hz) and one from a high group (1209-1633Hz) with each group containing four individual tones. The tone frequencies were carefully chosen such that they ar not harmonically related and that thei intermodulation products result in minimal signallin impairment (Fig. 1a). This scheme allows for 1 unique combinations. Ten of these codes represen the numerals zero through nine, the remaining si (*,#,A,B,C,D) being reserved for special signalling Most telephone keypads contain ten numeric pus buttons plus the asterisk (*) and octothorp (#). Th buttons are arranged in a matrix, each selecting its low group tone from its respective row and its hig group tone from its respective column (Fig. 1b).

The DTMF coding scheme ensures that each signa contains one and only one component from each o the high and low groups. This significantly simplifies decoding because the composite DTMF signal may be separated with bandpass filters, into its two single frequency components each of which may be handled individually. As a result DTMF coding has proven to provide a flexible signalling scheme o excellent reliability, hence motivating innovative and competitive decoder design.

Development

Early DTMF decoders (receivers) utilized banks of bandpass filters making them somewhat cumbersome and expensive to implement. This generally restricted their application to central offices (telephone exchanges).

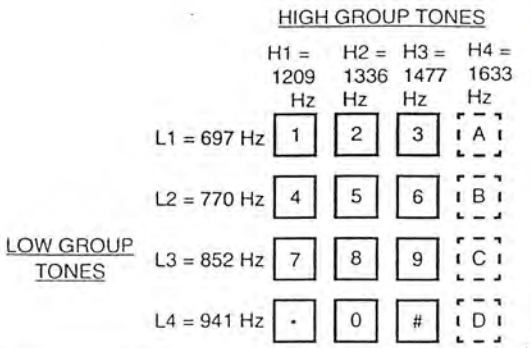
The first generation receiver typically used LC filters, active filters and/or phase locked loop techniques to



Standard DTMF frequency spectrum ± (1.5% + 2 Hz). Second harmonics of the low group (possibly created due to a non-linear channel) fall within the passband of the high group (Indicated by A,B,C,D). This is a potential source of interference.

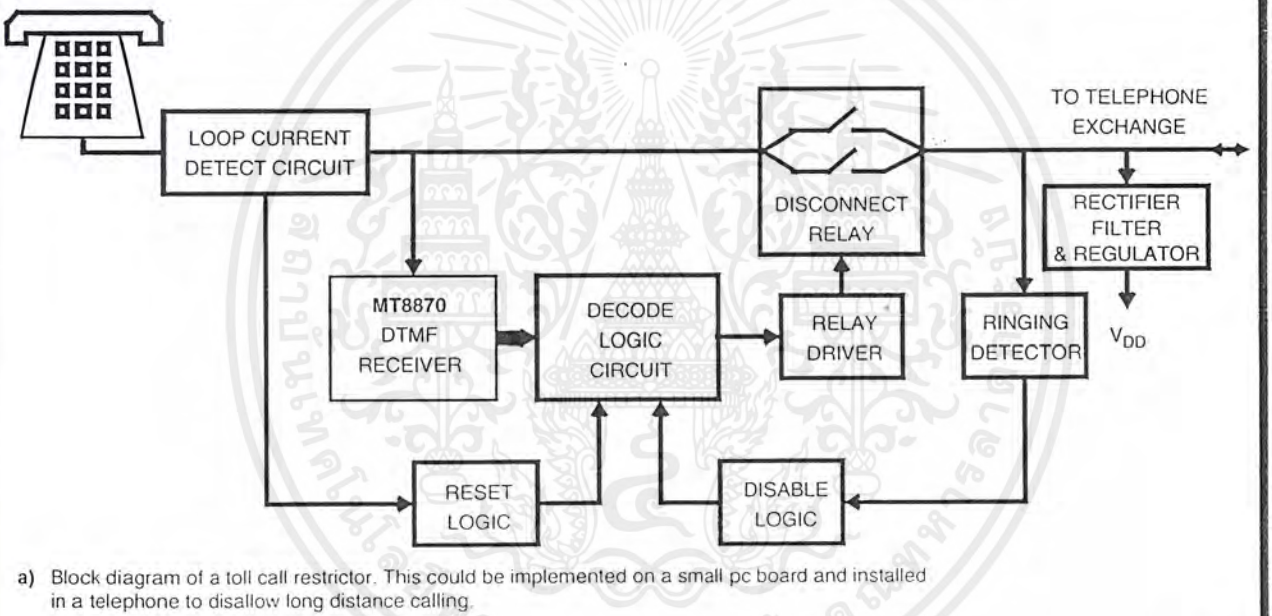
Figure 1a - The Dual Tone Multifrequency (DTMF) Spectrum

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

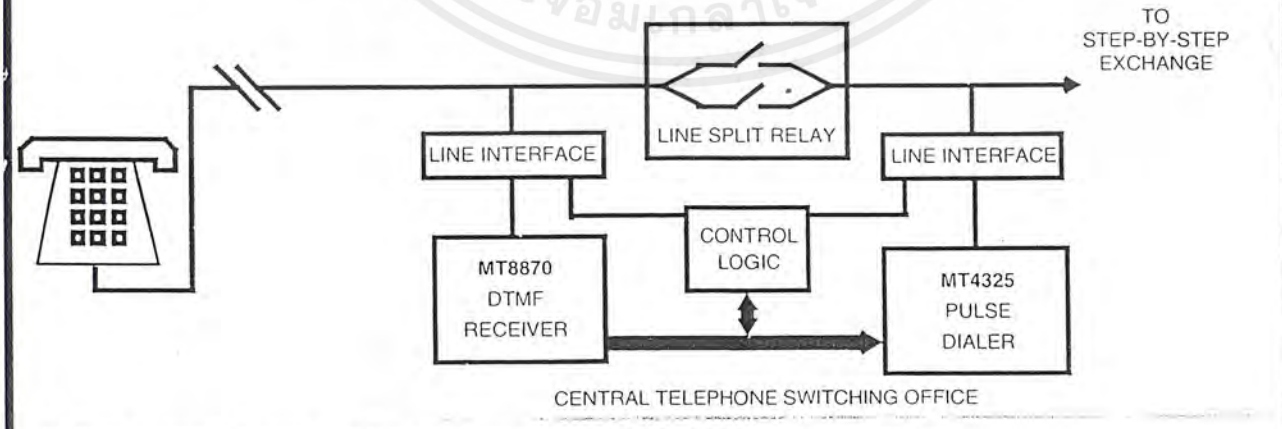


Telephone DTMF keypad matrix. Column H_4 is normally not available on a telephone keypad and is reserved for special signalling.

Figure 1b - The Dual Tone Multifrequency (DTMF) Keypad



a) Block diagram of a toll call restrictor. This could be implemented on a small pc board and installed in a telephone to disallow long distance calling.



b) Block diagram of a simple tome to pulse converter to allow TOUCH-TONE dialing into a step-by-step or crossbar exchange.

Figure 2 - Typical DTMF Receiver Applications

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

receive and decode DTMF tones. Initial functions were, commonly, phone number decoders and toll call restrictors. A DTMF receiver is also frequently used as a building block in a tone-to-pulse converter which allows Touch-Tone dialling access to mechanical step-by-step and crossbar exchanges (Fig. 2).

The introduction of MOS/LSI digital techniques brought about the second generation of tone receiver development. These devices were used to digitally decode the two discrete tones that result from decomposition of the composite signal. Two analog bandpass filters were used to perform the decomposition.

Totally self-contained receivers implemented in thick film hybrid technology depicted the start of third generation devices. Typically, they also used analog active filters to bandsplit the composite signal and MOS digital devices to decode the tones.

The development of silicon-implemented switched capacitor sampled filters marked the birth of the fourth and current generation of DTMF receiver technology. Initially single chip bandpass filters were combined with currently available decoders enabling a two chip receiver design. A further advance in integration has merged both functions onto a single chip allowing DTMF receivers to be realized in minimal space at a low cost.

The second and third generation technologies saw a tendency to shift complexity away from the analog circuitry towards the digital LSI circuitry in order to reduce the complexity of analog filters and their inherent problems. Now that the filters themselves can be implemented in silicon, the distribution of complexity becomes more a function of performance and silicon real estate.

Inside The MT8870

The MT8870 is a state of the art single chip DTMF receiver incorporating switched capacitor filter technology and an advanced digital counting/ averaging algorithm for period measurement. The block diagram (Fig. 3) illustrates the internal workings of this device.

To aid design flexibility, the DTMF input signal is first buffered by an input op-amp which allows adjustment of gain and choice of input configuration. The input stage is followed by a low pass continuous RC active filter which performs an antialiasing function. Dial tone at 350 and 440Hz is then rejected by a third order switched capacitor notch filter. The signal, still in its composite form, is then split into its individual high and low frequency components by two sixth order switched capacitor and pass filters. Each component tone is then smoothed by an output filter and squared up by a hard limiting comparator.

The two resulting rectangular waves are applied to digital circuitry where a counting algorithm measures and averages their periods. An accurate reference clock is derived from an inexpensive external 3.58MHz colourburst crystal.

The timing diagram (Fig. 4) illustrates the sequence of events which follow digital detection of a DTMF tone pair. Upon recognition of a valid frequency from each tone group the Early Steering (ESt) output is raised. The time required to detect the presence of two valid tones, t_{DP}, is a function of the decode algorithm, the tone frequency and the previous state of the decode logic. ESt indicates that two tones of proper frequency have been detected and initiates an RC timing circuit. If both tones are present for the minimum guard time, t_{GTP}, which is determined by the external RC network, the DTMF signal is decoded and the resulting data (Table 1) is latched in the output register. The Delayed Steering (StD) output is raised and indicates that new data is available. The time required to receive a valid DTMF signal, t_{REC}, is equal to the sum of t_{DP} and t_{GTP}.

f_{LOW}	f_{HIGH}	KEY	TOE	Q_4	Q_3	Q_2	Q_1
697	1209	1	1	0	0	0	1
697	1336	2	1	0	0	1	0
697	1477	3	1	0	0	1	1
770	1209	4	1	0	1	0	0
770	1336	5	1	0	1	0	1
770	1477	6	1	0	1	1	0
852	1209	7	1	0	1	1	1
852	1336	8	1	1	0	0	0
852	1477	9	1	1	0	0	1
941	1209	0	1	1	0	1	0
941	1336	*	1	1	0	1	1
941	1477	#	1	1	1	0	0
697	1633	A	1	1	1	0	1
770	1633	B	1	1	1	1	0
852	1633	C	1	1	1	1	1
941	1633	D	1	0	0	0	0
-	-	ANY	0	Z	Z	Z	Z

Table 1. MT8870 Output Truth Table

0=LOGIC LOW 1=LOGIC HIGH Z=HIGH IMPEDANCE
Output truth table. Note that key "0" is output as "1010_2 (ie:10_{10})" corresponding to standard telephony coding.

A simplified circuit diagram (Fig. 5) illustrates how the chip's steering circuit drives the external RC network to generate guard times. Pin 17, St/GT (Steering/Guard Time), is a bidirectional signal pin which controls StD, the output latches, and resets the timing circuit. When St/GT is in its input mode (St function) both Q_1 and Q_2 are turned off and the voltage level at St/GT is compared to the steering threshold voltage V_{TSt}. A transition from below to above V_{TSt} will switch the comparator's output from

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

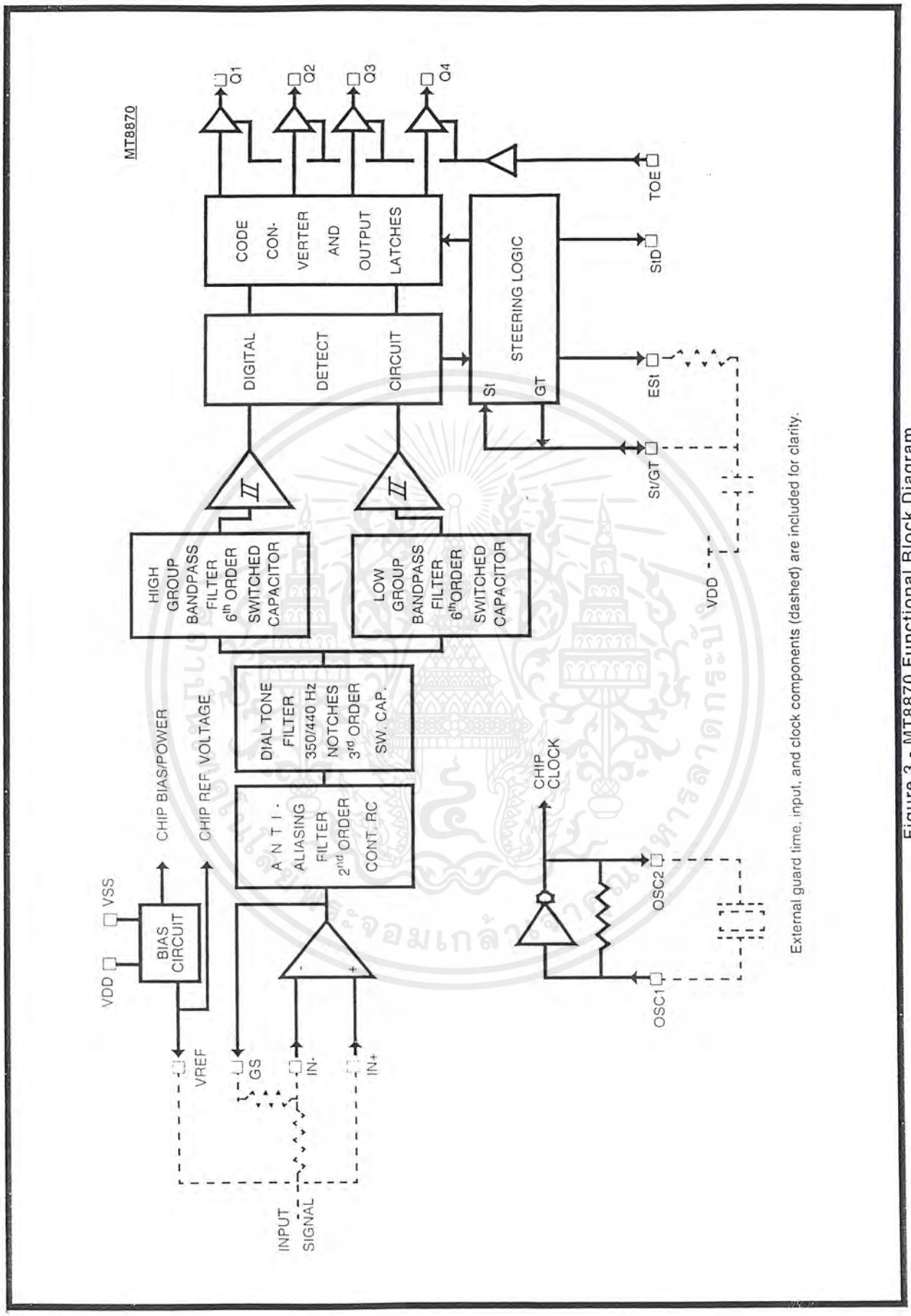


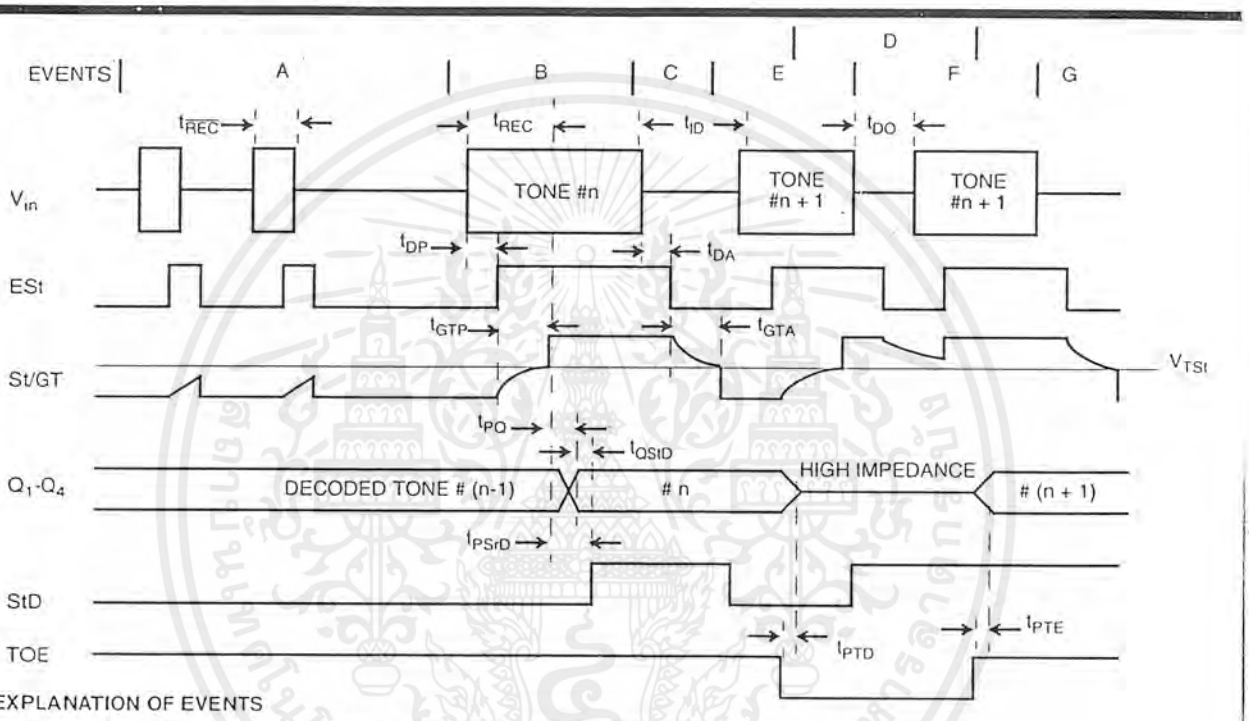
Figure 3 - MT8870 Functional Block Diagram

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

low to high strobing new data into the output latches, and raising the StD output. As long as an input level above V_{TSt} is maintained StD will remain high indicating the presence of a valid DTMF signal.

Initially, when no valid tone-pairs are present, capacitor C is fully charged applying a low voltage to St/GT. This causes a low at the comparator's output and since ESt is also low, Q_2 turns on ensuring that C is completely charged. In this condition St/GT is in its output mode (GT function). When a valid tone-pair is received ESt is raised turning off Q_2 which puts St/GT in its high impedance input mode and allows C to discharge through R. If this condition persists for the tone-present guard time, t_{GTP} tl voltage at St/GT rises above V_{TSt} raising StD whic indicates reception of a valid DTMF signal. If th tone pair drops out before the duration of t_{GTP} ESt lowered turning on Q_2 which charges C resetting th tone-present guard time.

Once a DTMF signal is recognized as valid both ES and the comparator output are high. This turns o Q_1 which discharges C and initializes the tone absent guard time, t_{GTA}. After the DTMF signal i removed, ESt is lowered, Q_1 turns off placing St/G in its input mode and C begins to charge through R.



EXPLANATION OF EVENTS

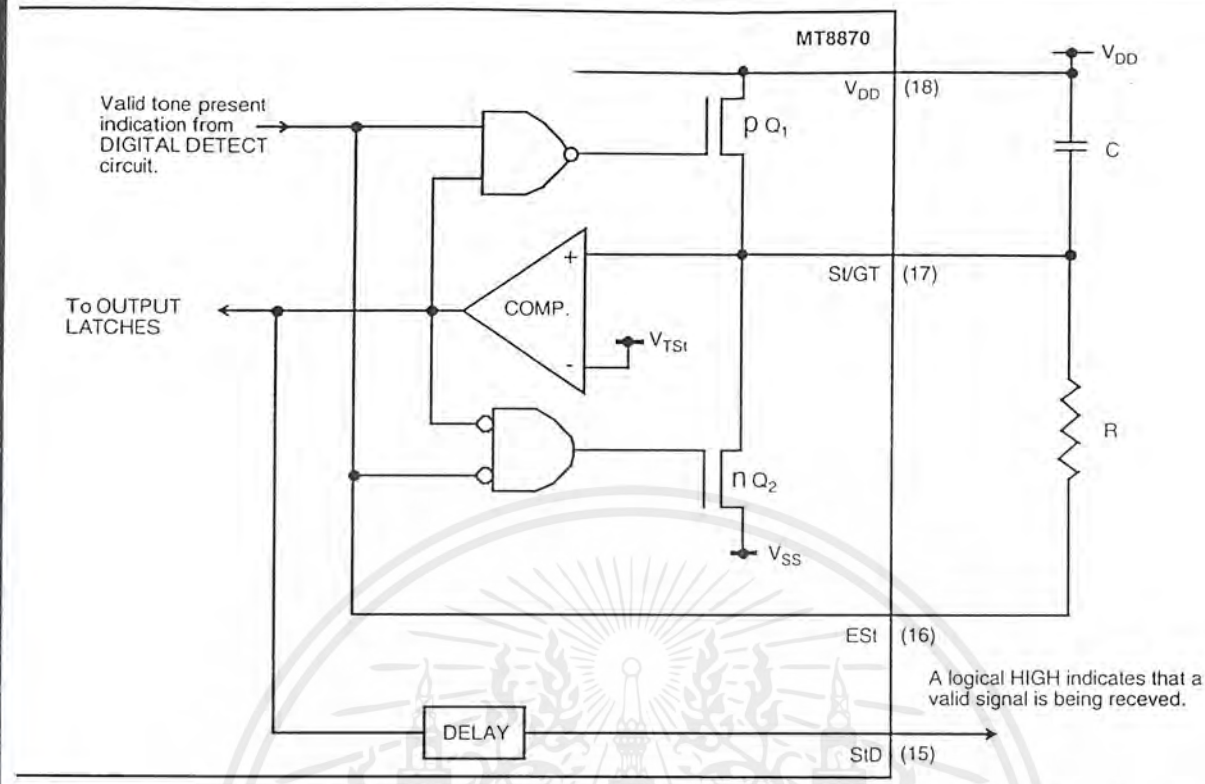
A)	TONE BURSTS DETECTED, TONE DURATION INVALID, OUTPUTS NOT UPDATED.
B)	TONE #n DETECTED, TONE DURATION VALID, TONE DECODED AND LATCHED IN OUTPUTS.
C)	END OF TONE #n DETECTED, TONE ABSENT DURATION VALID, OUTPUTS REMIAN LATCHED UNTIL NEXT VALID TONE.
D)	OUTPUTS SWITCHED TO HIGH IMPEDANCE STATE.
E)	TONE #n + 1 DETECTED, TONE DURATION VALID, TONE DECODED AND LATCHED IN OUTPUTS (CURRENTLY HIGH IMPEDANCE)
F)	ACCEPTABLE DROPOUT OF TONE #n + 1, TONE ABSENT DURATION INVALID, OUTPUTS REMAIN LATCHED.
G)	END OF TONE #n + 1 DETECTED, TONE ABSENT DURATION VALID, OUTPUTS REMAIN LATCHED UNTIL NEXT VALID TONE.

EXPLANATION OF SYMBOLS

V_{in}	DTMF COMPOSITE INPUT SIGNAL.
ESt	EARLY STEERING OUTPUT. INDICATES DETECTION OF VALID TONE FREQUENCIES.
St/GT	STEERING INPUT/GUARD TIME OUTPUT. DRIVES EXTERNAL RC TIMING CIRCUIT.
Q_1-Q_4	4-BIT DECODED TONE OUTPUT.
StD	DELAYED STEERING OUTPUT. INDICATES THAT VALID FREQUENCIES HAVE BEEN PRESENT/ABSENT FOR THE REQUIRED GUARD TIME THUS CONSTITUTING A VALID SIGNAL.
TOE	TONE OUTPUT ENABLE (INPUT). A LOW LEVEL SHIFTS Q_1-Q_4 TO ITS HIGH IMPEDANCE STATE.
t_{REC}	MAXIMUM DTMF SIGNAL DURATION NOT DETECED AS VALID
t_{REC}	MINIMUM DTMF SIGNAL DURATION REQUIRED FOR VALID RECOGNITION
t_{ID}	MAXIMUM TIME BETWEEN VALID DTMF SIGNALS
t_{DO}	MAXIMUM ALLOWABLE DROP OUT DURING VALID DTMF SIGNAL
t_{DP}	TIME TO DETECT THE PRESENCE OF VALID DTMF SIGNALS
t_{DA}	TIME TO DETECT THE ABSENCE OF VALID DTMF SIGNALS
t_{GTP}	GUARD TIME, TONE PRESENT.
t_{GTA}	GUARD TIME, TONE ABSENT.

Figure 4 - MT8870 Timing Diagram

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



Simplified steering circuit. Initially ESt is low, C is fully charged applying 0V to St/GT and Q_2 is on. Upon reception of a valid tone pair ESt is raised turning off Q_2 and allowing C to discharge through R which increases the voltae at St/GT. When VTSt is reached the comparator output goes high indicating a valid signal, latches the outputs and turns on Q_1 which discharges C. When the tone pair is lost ESt goes low Q_1 turns off and C charges through R decreasing the voltage at St/GT. When V_{TSt} is reached StD goes low and Q_2 turns on resetting the timing circuit.

STEERING TRUTH TABLE

ESt	St(/GT)	(St/)GT	StD
0	$<V_{TSt}$	0	0
0	$>V_{TSt}$	Z	1
1	$<V_{TSt}$	Z	0
1	$>V_{TSt}$	1	1

0 - LOGIC LOW
1 = LOGIC HIGH
Z = HiH IMPEDANCE
V_{TSt} = Threshold Voltqae
(typically 1/2 V_{DD})

Steering circuit truth table. Note that pin 17 (St/GT) acts as both an input and an output depending on the relative states of ESt and the comparator output.

Figure 5 - MT8870 Steering And GuardTime Circuit Operation

If the same valid tone-pair does not reappear before t_{GTA} then the voltage at St/GT falls below V_{TSt} which resets the timing circuit via Q_2 and prepares the device to receive another signal. If the same valid tone-pair reappears before t_{GTA}, ESt is raised turning on Q_1 and discharging C which resets t_{GTA}. In this case StD remains high and the tone dropout is disregarded as noise.

To provide good reliability in a typical telephony environment, a DTMF receiver should be designed to recognize a valid tone-pair greater than 40mS in duration and, to accept as successive digits, tone-pairs that are greater than 40mS apart. However in other environments, such as two-way radio, the optimum tone duration and intra-digit times may differ due to noise considerations.

By adding an extra resistor and steering diode (Fig. 6b, 6c) t_{GTP} and t_{GTA} can be set to different values. Guard time adjustment allows tailoring of noise immunity and talk-off performance to meet specific system needs.

Talk-off is a measure of errors that occur when the receiver falsely detects a tone pair due to speech or background noise simulating a DTMF signal. Increasing t_{GTP} improves talk off performance since

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

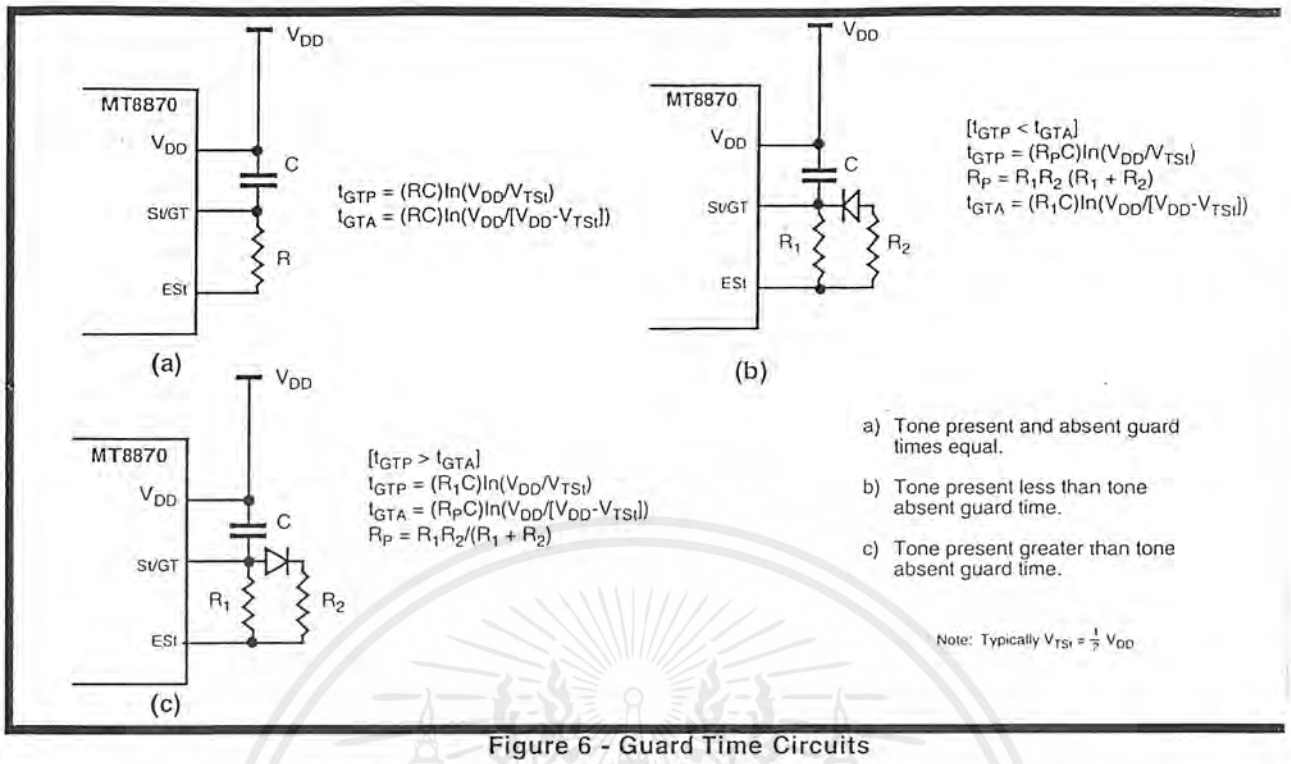


Figure 6 - Guard Time Circuits

it reduces the probability that speech will maintain DTMF simulation long enough to be considered valid. The trade-off here is decreased noise immunity because dropout (longer than t_{DA}) due to noise pulses will restart t_{GTP}. Therefore, for noisy environments, t_{GTP} should be decreased. The signal absent guard time, t_{GTA}, determines the minimum time allowed between successive DTMF signals. A dropout shorter than t_{GTA} will be considered noise and will not register as a successive valid tone detection. This guards against multiple reception of a single character. Therefore, lengthening t_{GTA} will increase noise immunity and tolerance to the presence of an unwanted third tone at the expense of decreasing the maximum signalling rate.

The intricacies of the digital detection algorithm have a significant impact on the overall receiver performance. It is here that the initial decision is made to accept the signal as valid or reject it as speech or noise.

Trade-offs must be made between eliminating talk off errors and eliminating the effects of unwanted third tone signals and noise. These are mutually conflicting events. On one hand valid DTMF signals present in noise must be recognized which requires relaxation of the detection criteria. On the other hand, relaxing the detection criteria increases the probability of receiving "hits" due to talk off errors.

Many considerations must be taken into account in evaluating criteria for noise rejection. In the telephony environment two sources of noise are predominant. These are, third tone interference which generally comes from dial tone harmonics and band-limited white noise . In the MT8870 a complex digital averaging algorithm provides excellent immunity to voice, third tone and noise signals which prevail in a typical voice bandwidth channel.

The algorithm used in the MT8870 combines the best features from two previous generations of Mitel digital decoders with improvements resulting from years of practical use within the telephone environment. The algorithm has evolved through a combination of statistical calculations and empirical "tweaks" to result in the realization of an extremely reliable decoder.

Applications

The proven reliability of DTMF signalling has created a vast spectrum of possible applications. Until recently, many of these applications were rendered ineffective due to cost or size considerations. Now that a complete DTMF receiver can be designed with merely a single chip and a few external passive components one can take full advantage of a highly developed signalling scheme as a small, cost effective signalling solution.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

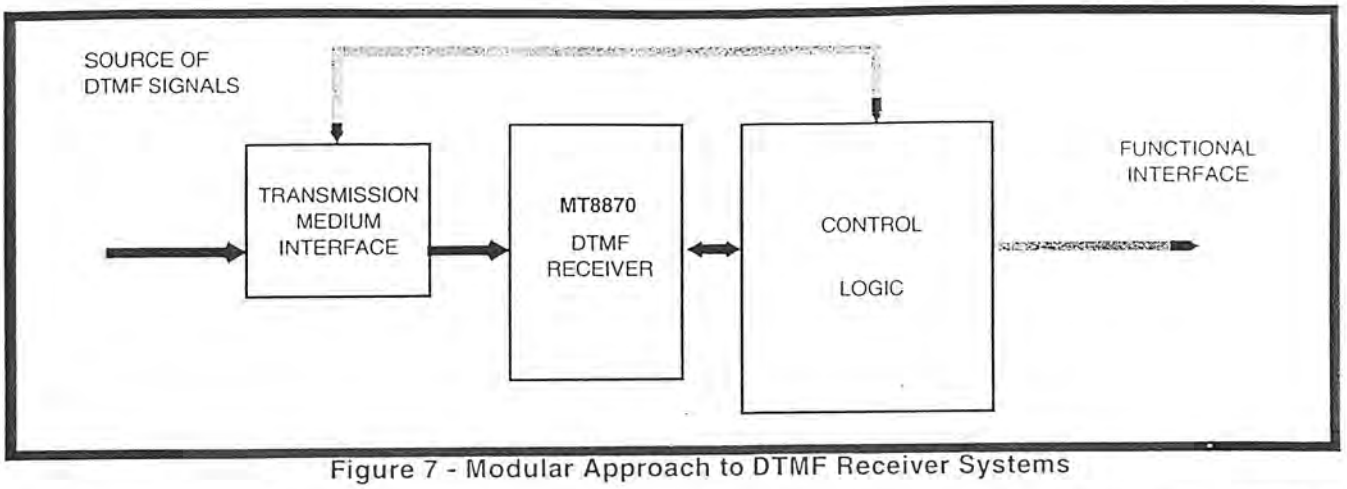


Figure 7 - Modular Approach to DTMF Receiver Systems

The design of a DTMF receiving system can generally be broken down into three functional blocks (Fig. 7). The first consideration is the interface to the transmission medium. This may be as simple as a few passive components to adequately configure the MT8870's input stage or as complex as some form of demodulation, multiplexing or analog switching system. The second functional block is the DTMF receiver itself. This is where the receiving system's parameters can be optimized for the specific signal conditions delivered from the "front end" interface. The third, and perhaps most widely varying function, is the output control logic. This may be as simple as a 4 to 16 line decoder, controlling a specific function for each DTMF code, or as complex as a full blown computer handling system protocols and adaptively varying the tone receiver's parameters to adjust for changing signal conditions. Several currently applied and conceptually designed applications are described subsequently but first let's consider the design of a typical input stage.

The input arrangement of the MT8870 provides a differential input op amp as well as a bias source (V_{REF}) which is used to bias the inputs at mid-rail. The output of this op amp is available to provide feedback for gain adjustment.

A typical single ended input configuration having unity gain is shown in Figure 8.

For balanced line applications good common mode rejection is offered by the differential configuration (Fig. 9). In both cases, the inputs are biased to 1/2 V_{DD} by V_{Ref}. Consider an input stage which will interface to a 600Ω balanced line. To reject common mode noise signals, a balanced differential amplifier input provides the solution.

With the input configured for unity gain the MT8870 will accept maximum signal levels of +1 dBm (into 600Ω). The lowest DTMF frequency that must be detected is approximately 685Hz. Allowing 0.1dB of

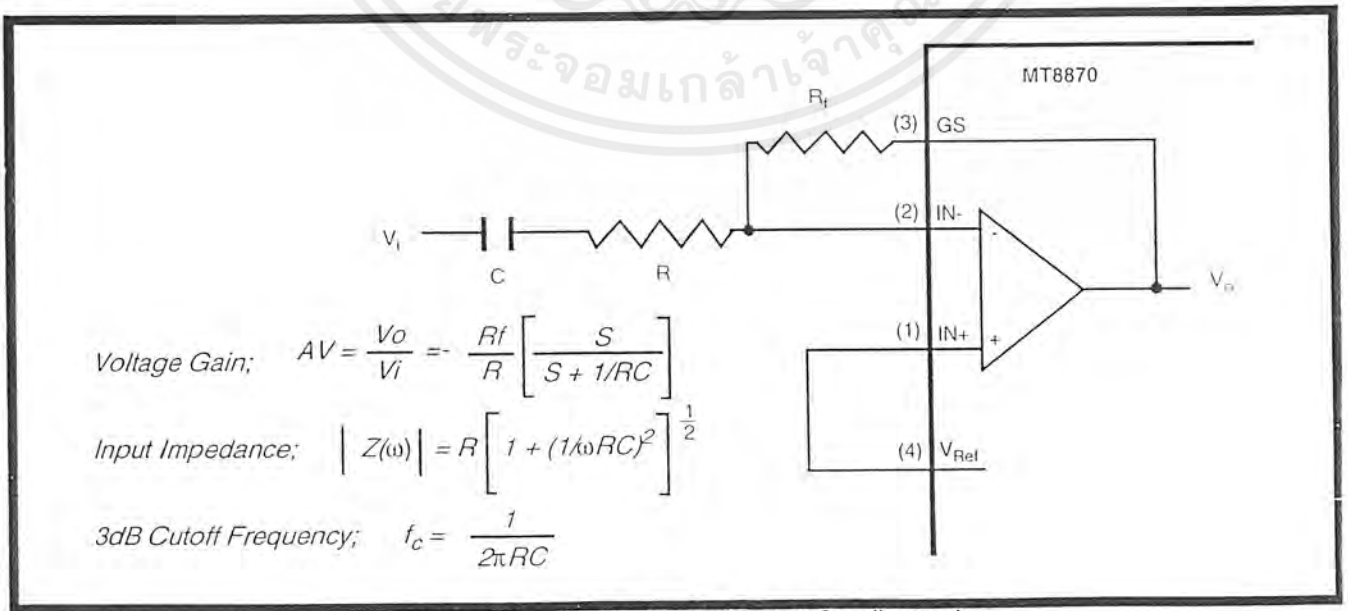


Figure 8 - Single Ended Input Configuration

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

National Semiconductor

February 1995

LM567/LM567C Tone Decoder

General Description

The LM567 and LM567C are general purpose tone decoders designed to provide a saturated transistor switch to ground when an input signal is present within the passband. The circuit consists of an I and Q detector driven by a voltage controlled oscillator which determines the center frequency of the decoder. External components are used to independently set center frequency, bandwidth and output delay.

Features

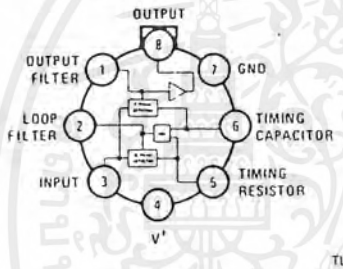
- 20 to 1 frequency range with an external resistor
- Logic compatible output with 100 mA current sinking capability
- Bandwidth adjustable from 0 to 14%
- High rejection of out of band signals and noise
- Immunity to false signals
- Highly stable center frequency
- Center frequency adjustable from 0.01 Hz to 500 kHz

Applications

- Touch tone decoding
- Precision oscillator
- Frequency monitoring and control
- Wide band FSK demodulation
- Ultrasonic controls
- Carrier current remote controls
- Communications paging decoders

Connection Diagrams

Metal Can Package

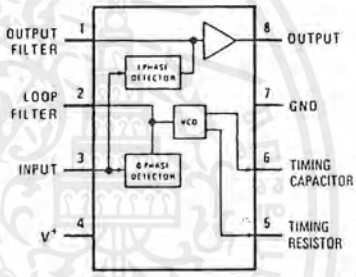


TL/H/6975–1

Top View

Order Number LM567H or LM567CH
See NS Package Number H08C

Dual-In-Line and Small Outline Packages



TL/H/6975–2

Top View

Order Number LM567CM
See NS Package Number M08A
Order Number LM567CN
See NS Package Number N08E

© 1995 National Semiconductor Corporation TL/H/6975

RRD-B30M115/Printed in U.S.A.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage Pin	9V
Power Dissipation (Note 1)	1100 mW
V_8	15V
V_3	−10V
V_3	V_4 + 0.5V
Storage Temperature Range	−65°C to +150°C
Operating Temperature Range	
LM567H	−55°C to +125°C
LM567CH, LM567CM, LM567CN	0°C to +70°C

Soldering Information

Dual-In-Line Package	
Soldering (10 sec.)	260°C
Small Outline Package	
Vapor Phase (60 sec.)	215°C
Infrared (15 sec.)	220°C

See AN-450 "Surface Mounting Methods and Their Effect on Product Reliability" for other methods of soldering surface mount devices.

Electrical Characteristics AC Test Circuit, T_A = 25°C, V^+ = 5V

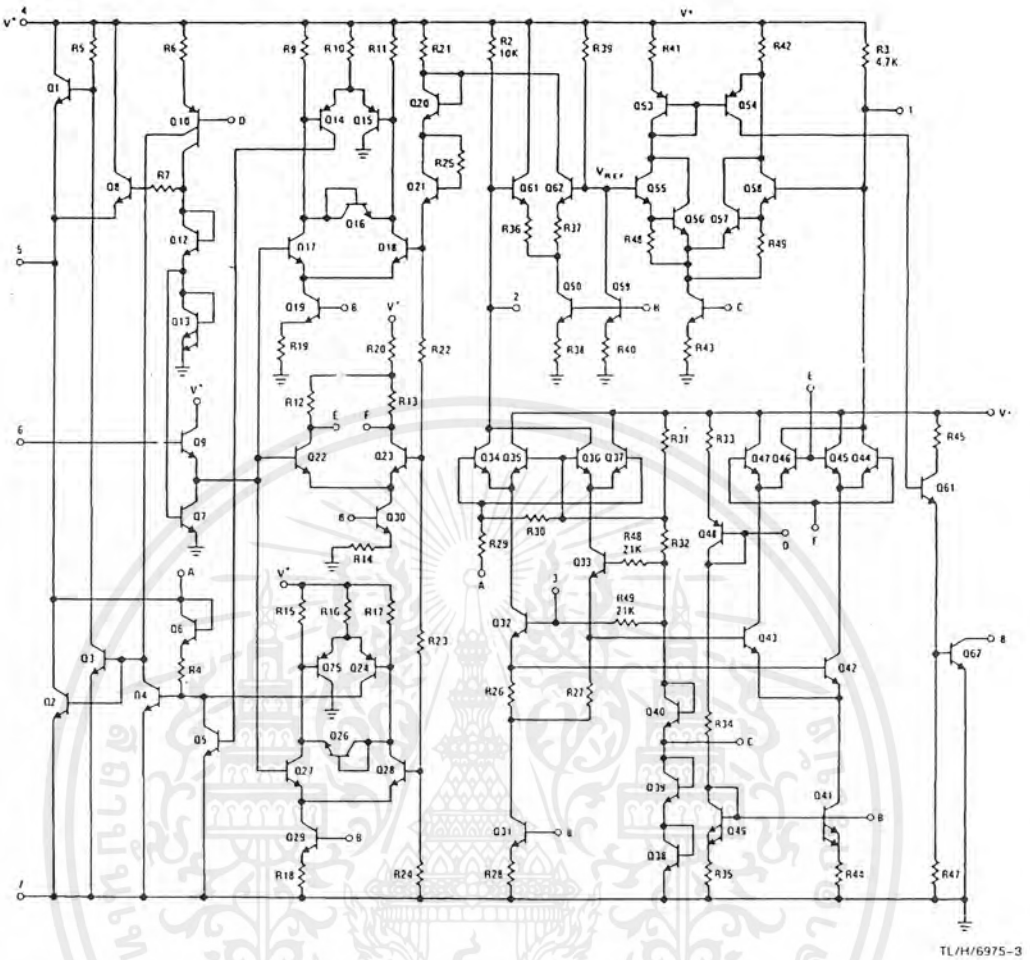
Parameters	Conditions	LM567			LM567C/LM567CM			Units
		Min	Typ	Max	Min	Typ	Max	
Power Supply Voltage Range		4.75	5.0	9.0	4.75	5.0	9.0	V
Power Supply Current Quiescent	R_L = 20k		6	8		7	10	mA
Power Supply Current Activated	R_L = 20k		11	13		12	15	mA
Input Resistance		18	20		15	20		kΩ
Smallest Detectable Input Voltage	I_L = 100 mA, $f_i = f_o$		20	25		20	25	mVrms
Largest No Output Input Voltage	I_C = 100 mA, $f_i = f_o$	10	15		10	15		mVrms
Largest Simultaneous Outband Signal to Inband Signal Ratio			6			6		dB
Minimum Input Signal to Wideband Noise Ratio	B_n = 140 kHz		−6			−6		dB
Largest Detection Bandwidth		12	14	16	10	14	18	% of f_o
Largest Detection Bandwidth Skew			1	2		2	3	% of f_o
Largest Detection Bandwidth Variation with Temperature			±0.1			±0.1		%/°C
Largest Detection Bandwidth Variation with Supply Voltage	4.75 − 6.75V		±1	±2		±1	±5	%V
Highest Center Frequency		100	500		100	500		kHz
Center Frequency Stability (4.75–5.75V)	$0 < T_A < 70$ $-55 < T_A < +125$		35 ± 60 35 ± 140			35 ±60 35 ± 140		ppm/°C ppm/°C
Center Frequency Shift with Supply Voltage	4.75V − 6.75V 4.75V − 9V		0.5	1.0 2.0		0.4	2.0 2.0	%/V %/V
Fastest ON-OFF Cycling Rate			f_o/20			f_o/20		
Output Leakage Current	V_8 = 15V		0.01	25		0.01	25	μA
Output Saturation Voltage	e_i = 25 mV, I_8 = 30 mA e_i = 25 mV, I_8 = 100 mA		0.2 0.6	0.4 1.0		0.2 0.6	0.4 1.0	V
Output Fall Time			30			30		ns
Output Rise Time			150			150		ns

Note 1: The maximum junction temperature of the LM567 and LM567C is 150°C. For operating at elevated temperatures, devices in the TO-5 package must be derated based on a thermal resistance of 150°C/W, junction to ambient or 45°C/W, junction to case. For the DIP the device must be derated based on a thermal resistance of 110°C/W, junction to ambient. For the Small Outline package, the device must be derated based on a thermal resistance of 160°C/W, junction to ambient.

Note 2: Refer to RETS567X drawing for specifications of military LM567H version.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

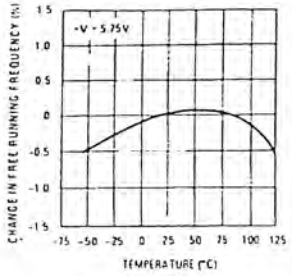
Schematic Diagram



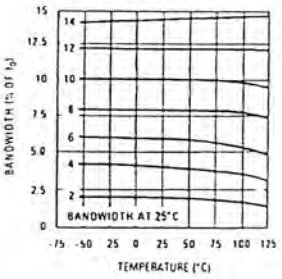
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Typical Performance Characteristics

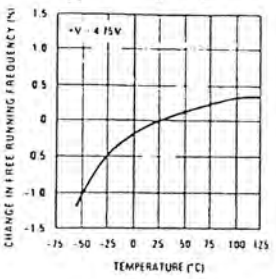
Typical Frequency Drift



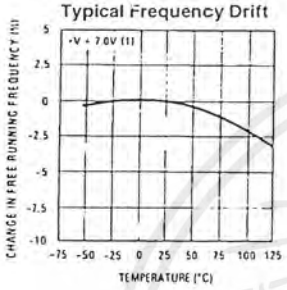
Typical Bandwidth Variation



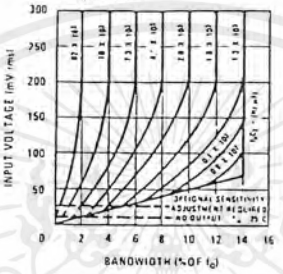
Typical Frequency Drift



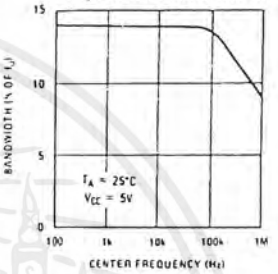
Typical Frequency Drift



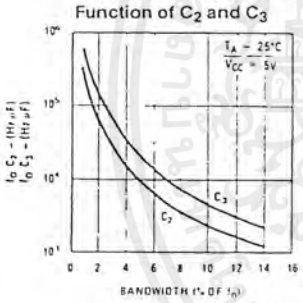
Bandwidth vs Input Signal Amplitude



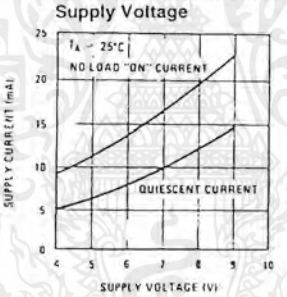
Largest Detection Bandwidth



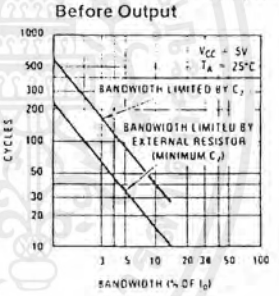
Detection Bandwidth as a Function of C_2 and C_3



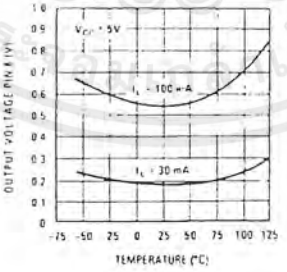
Typical Supply Current vs Supply Voltage



Greatest Number of Cycles Before Output



Typical Output Voltage vs Temperature

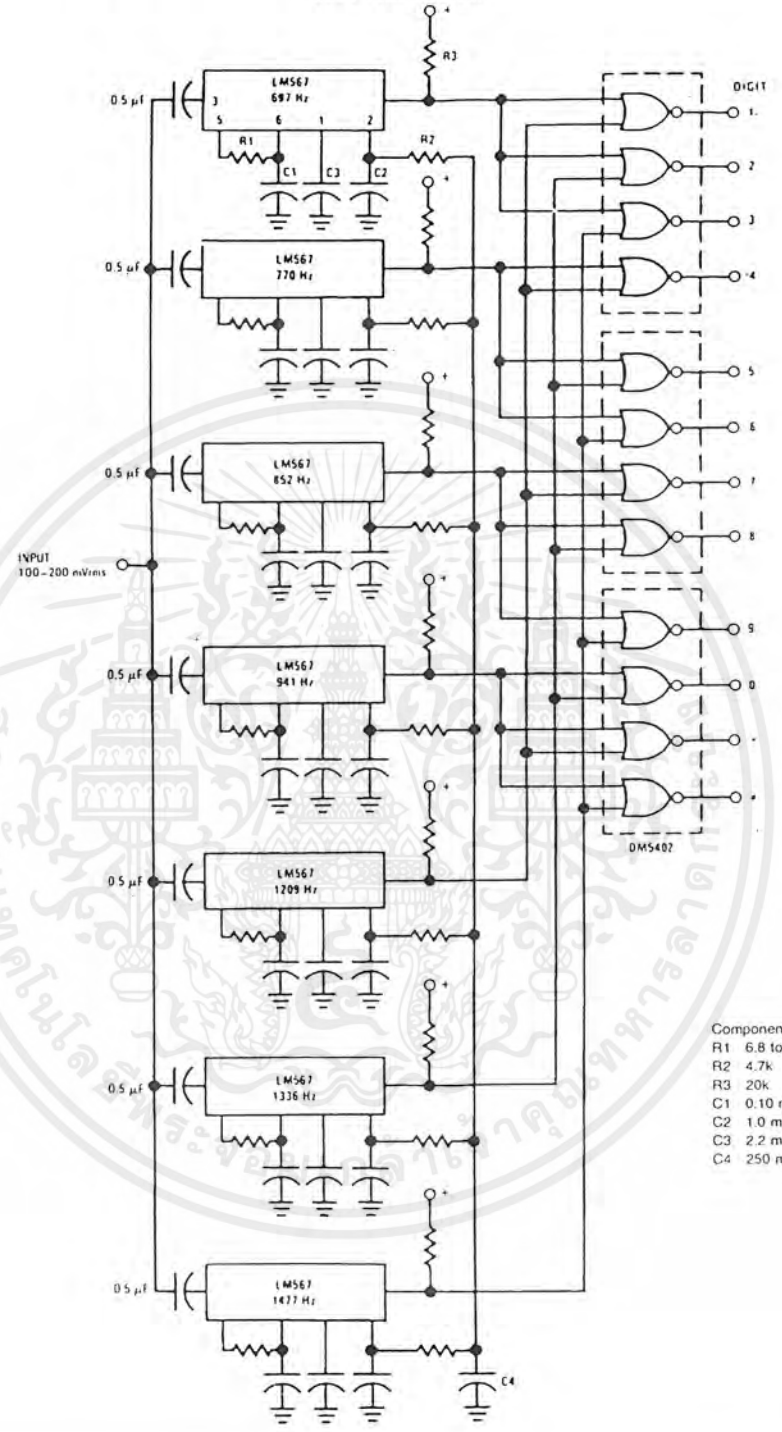


TL/H/6975–4

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Typical Applications

Touch-Tone Decoder



Component values (typ)

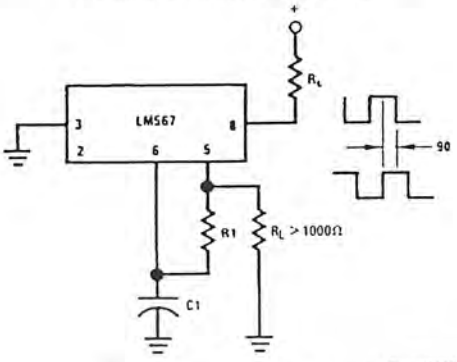
R1	6.8 to 15k
R2	4.7k
R3	20k
C1	0.10 mfd
C2	1.0 mfd 6V
C3	2.2 mfd 6V
C4	250 mfd 6V

TL/H/6975–5

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Typical Applications (Continued)

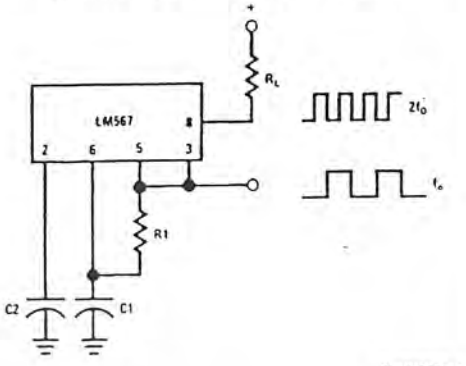
Oscillator with Quadrature Output



TL/H/6975-6

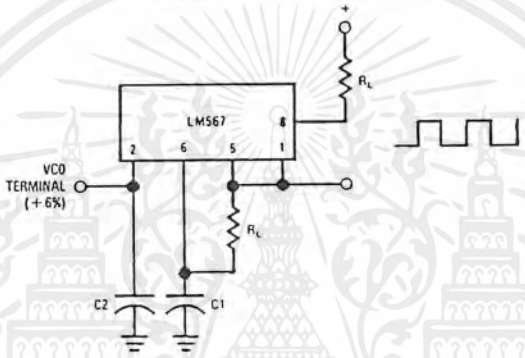
Connect Pin 3 to 2.8V to Invert Output

Oscillator with Double Frequency Output



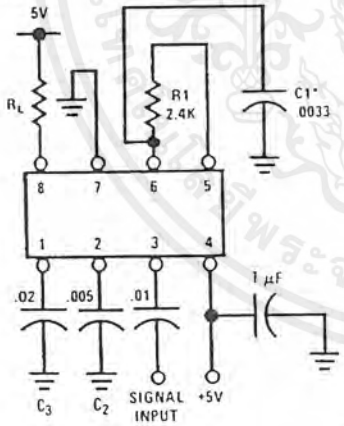
TL/H/6975-7

Precision Oscillator Drive 100 mA Loads



TL/H/6975-8

AC Test Circuit



TL/H/6975-9

f_i = 100 kHz ± 5V

*Note: Adjust for f_o = 100 kHz

Applications Information

The center frequency of the tone decoder is equal to the free running frequency of the VCO. This is given by

$$f_o \approx \frac{1}{1.1\, R_1 C_1}$$

The bandwidth of the filter may be found from the approximation

$$BW = 1070 \sqrt{\frac{V_i}{f_o C_2}} \text{ in \% of } f_o$$

Where:

V_i = Input voltage (volts rms). $V_i \leq 200$ mV

C_2 = Capacitance at Pin 2 (μF)

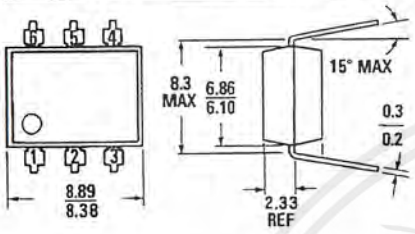
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

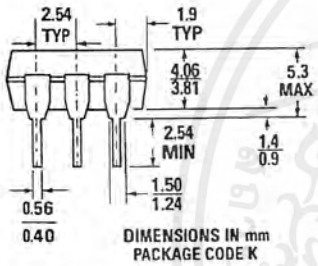
QT OPTOELECTRONICS

PHOTOTRANSISTOR OPTOCOUPLERS

4N25 4N27
4N26 4N28

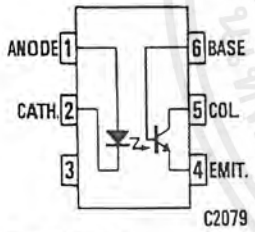
PACKAGE DIMENSIONS





DIMENSIONS IN mm
PACKAGE CODE K

ST1603A



Equivalent Circuit

DESCRIPTION

The 4N25, 4N26, 4N27, and 4N28 series of optocouplers have an NPN silicon planar phototransistor optically coupled to a gallium arsenide diode.

FEATURES & APPLICATIONS

- AC line/digital logic isolator
- Digital logic/digital logic isolator
- Telephone/telegraph line receiver
- Twisted pair line receiver
- High frequency power supply feedback control
- Relay contact monitor
- Power supply monitor
- Small package size and low cost
- Excellent frequency response
- UL recognized—File E90700

ABSOLUTE MAXIMUM RATINGS

TOTAL PACKAGE

*Storage temperature	−55°C to 150°C
*Operating temperature at junction	−55°C to 100°C
*Lead temperature (soldering, 10 sec)	260°C
*Total package power dissipation at 25°C ambient (LED plus detector)	250 mW
*Derate linearly from 25°C	3.3 mW/°C

INPUT DIODE

*Forward DC current continuous	80 mA
*Reverse voltage	3.0 V
*Peak forward current (300 μs, 2% duty cycle)	3.0 A
*Power dissipation at 25°C ambient	150 mW
*Derate linearly from 25°C	2.0 mW/°C

OUTPUT TRANSISTOR

*Collector emitter voltage (BV_{CEO})	30 V
*Collector base voltage (BV_{CBO})	70 V
*Emitter collector voltage (BV_{ECO})	7 V
*Power dissipation at 25°C ambient	150 mW
*Derate linearly from 25°C	2.0 mW/°C

*Indicates JEDEC Registered Data.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ELECTRO-OPTICAL CHARACTERISTICS
(25°C Free Air Temperature Unless Otherwise Specified)

INDIVIDUAL COMPONENT CHARACTERISTICS

CHARACTERISTICS	SYMBOL	MIN.	TYP.	GUAR. MAX.	UNITS	TEST CONDITIONS
INPUT DIODE						
*Forward voltage	V_F		1.20	1.50	V	I_F=10 mA
Capacitance	C		150		pF	V_F=0 V, f=1 MHz
*Reverse leakage current			.05	100	μA	V_R=3.0 V, R_L=1.0 MΩ
DETECTOR						
DC forward current gain	h_{FE}		250			V_{CE}=5 V, I_C=500 μA
*Collector to emitter breakdown voltage	BV_{CEO}	30	65		V	I_C=1.0 mA, I_B=0
*Collector to base breakdown voltage	BV_{CBO}	70	165		V	I_C=100 μA, I_E=0
*Emitter to collector breakdown voltage	BV_{ECO}	7	14		V	I_E=100 μA, I_B=0
*Collector to emitter leakage current (4N25, 4N26, 4N27)	I_{CEO}		3.5	50	nA	V_{CE}=10 V Base Open
*Collector to emitter leakage current (4N28)				100	nA	
*Collector to base leakage current	I_{CBO}		0.1	20	nA	V_{CB}=10 V Emitter Open

TRANSFER CHARACTERISTICS

DC CHARACTERISTICS	SYMBOL	MIN.	TYP.	GUAR. MAX.	UNITS	TEST CONDITIONS
*Collector output current (a) (4N25, 4N26)	I_C	2.0	5.0	—	mA	V_{CE}=10 V, I_F=10 mA, I_B=0
(4N27, 4N28)		1.0	3.0	—		
*Collector-emitter saturation	$V_{CE(SAT)}$		0.2	0.5	V	I_C=2.0 mA, I_F=50 mA

TRANSFER CHARACTERISTICS

AC CHARACTERISTICS	SYMBOL	TYP.	UNITS	TEST CONDITIONS
Non-saturated Collector Delay time	t_d	0.5	μs	R_L=100 Ω, I_C=2 mA, V_{CC}=10 V
Rise time	t_r	2.5	μs	(Fig. 10 and 11)
Fall time	t_f	2.6	μs	
Non-saturated Collector Delay time	t_d	2.0	μs	R_L=1kΩ, I_C 2 mA, V_{CC}=10 V
Rise time	t_r	15	μs	(Fig. 10 and 11)
Fall time	t_f	15	μs	

*Indicates JEDEC Registered Data.
(a) Pulse Test: Pulse Width=300 μs, Duty Cycle ≤2.0%
(b) For this test LED pins 1 and 2 are common and Phototransistor pins 4, 5 and 6 are common.
(c) If adjusted to yield I_C=2 mA and i_c=0.7 mA RMS; Bandwidth referenced to 10 kHz.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ELECTRO-OPTICAL CHARACTERISTICS
(25°C Free Air Temperature Unless Otherwise Specified) (Cont'd)

TRANSFER CHARACTERISTICS (Cont'd)

AC CHARACTERISTICS	SYMBOL	MIN.	TYP.	GUAR. MAX.	UNITS	TEST CONDITIONS
Saturated t_{on} (from 5 V to 0.8 V)	t_{on} (SAT)		5		μs	R_L=2kΩ, I_F=15 mA, V_{CC}=5 V
t_{of} (from SAT to 2.0 V)	t_{of} (SAT)		25		μs	R_B=Open (Fig. 10)
Saturated t_{on} (from 5 V to 0.8 V)	t_{on} (SAT)		5		μs	R_L=2kΩ, I_F=20 mA, V_{CC}=5 V
t_{of} (from SAT to 2.0 V)	t_{of} (SAT)		18		μs	R_B=100kΩ (Fig. 10)
Non-saturated Base—Collector photo diode Rise time	t_r		175		ns	R_L=1kΩ, V_{CB}=10 V
Fall time	t_f		175		ns	
Isolation voltage (b) (4N25, 4N26, 4N27, 4N28) *(4N26, 4N27) *(4N28)	V_{ISO}	5300 1500 500	— — —	— — —	V V V	$I_{I-O} \leq 1$ μA RMS, t=1 minute Peak Peak
Isolation resistance (b)			10^{11}		Ω	V=500 VDC
Isolation capacitance (b)			1.3		pF	V=0, f=1.0 MHz
Bandwidth (c) (also see note 2)	B_W		300		kHz	I_C=2.0 mA, R_L=100 Ω (Fig. 12)

*Indicates JEDEC Registered Data.
(a) Pulse Test: Pulse Width=300 μs, Duty Cycle ≤2.0%
(b) For this test LED pins 1 and 2 are common and Phototransistor pins 4, 5 and 6 are common.
(c) If adjusted to yield I_C=2 mA and i_c=0.7 mA RMS; Bandwidth referenced to 10 kHz.

TYPICAL ELECTRO-OPTICAL CHARACTERISTIC CURVES
(25°C Free Air Temperature Unless Otherwise Specified)

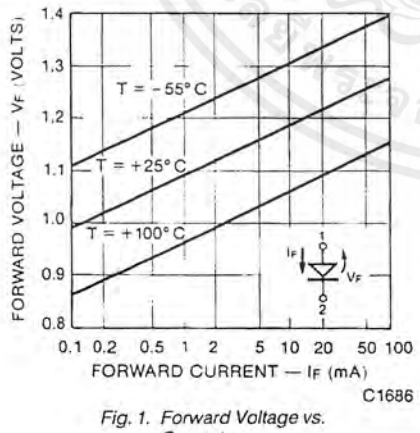


Fig. 1. Forward Voltage vs. Current

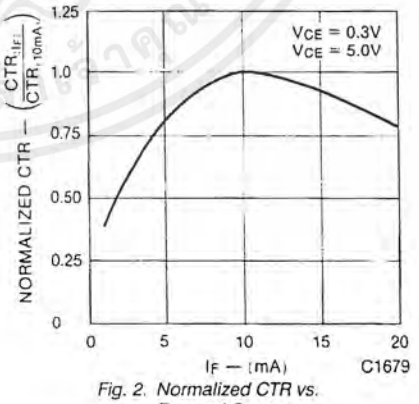


Fig. 2. Normalized CTR vs. Forward Current

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

TYPICAL ELECTRO-OPTICAL CHARACTERISTIC CURVES

(25°C Free Air Temperature Unless Otherwise Specified) (Cont'd)

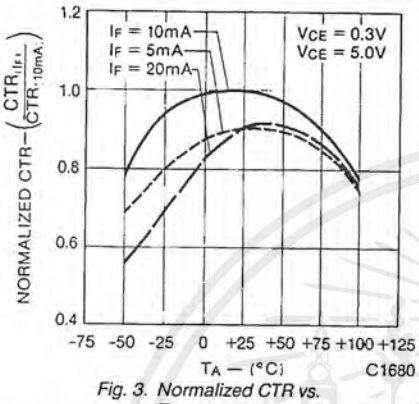


Fig. 3. Normalized CTR vs. Temperature

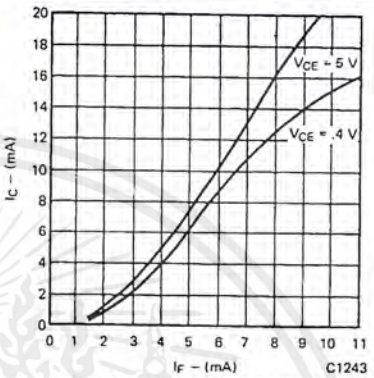


Fig. 4. Collector Current vs. Forward Current

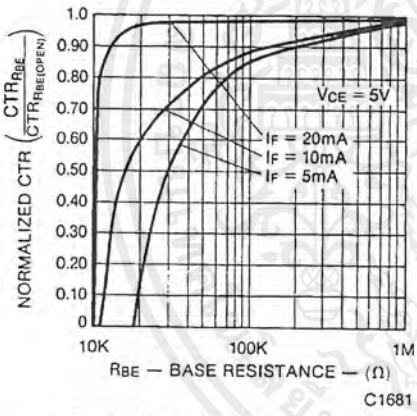


Fig. 5. CTR vs. RBE (Unsaturated)

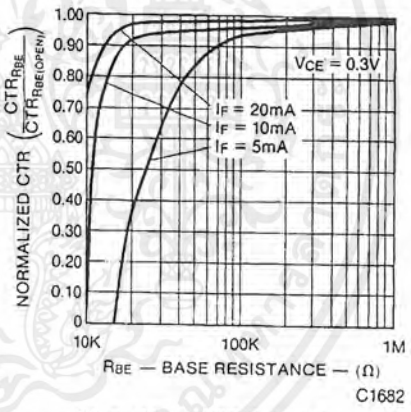


Fig. 6. CTR vs. RBE (Saturated)

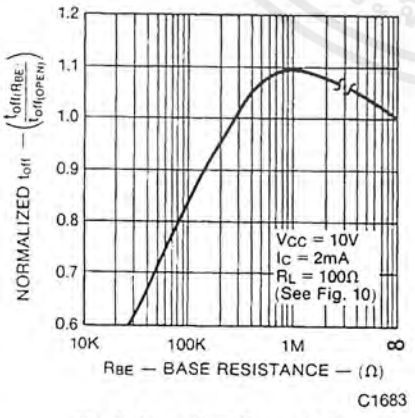


Fig. 7. Normalized T_{OFF} vs. RBE

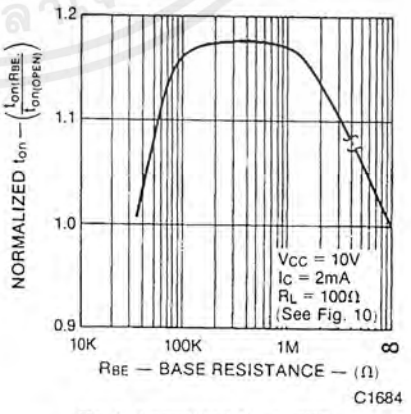


Fig. 8. Normalized T_{ON} vs. RBE

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

TYPICAL ELECTRO-OPTICAL CHARACTERISTIC CURVES
(25°C Free Air Temperature Unless Otherwise Specified) (Cont'd)

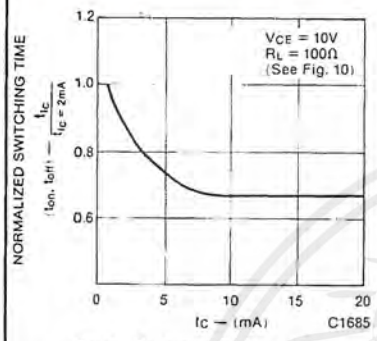


Fig. 9. Switching Time vs. IC

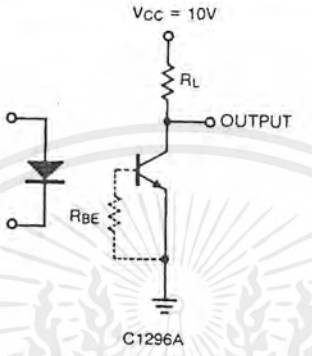


Fig. 10. Switching Time Test Circuit

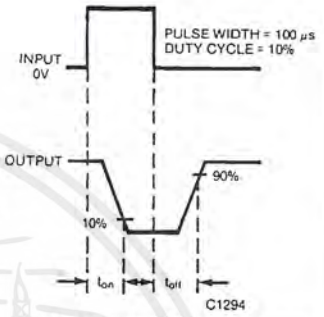


Fig. 11. Switching Time Waveforms

OPERATING SCHEMATICS

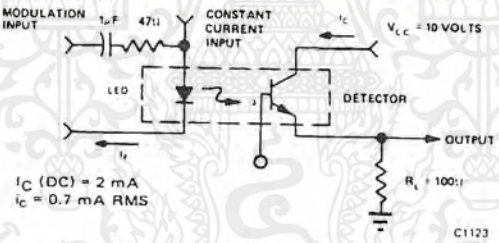


Fig. 12. Modulation Circuit Used to Obtain Output vs. Frequency Plot

NOTES

1. The current transfer ratio (I_C/I_F) is the ratio of the detector collector current to the LED input current with V_{CE} at 10 volts.
2. The frequency at which i_c is 3dB down from the 10 kHz value.
3. Rise time (t_r) is the time required for the collector current to increase from 10% of its final value to 90%.
   Fall time (t_f) is the time required for the collector current to decrease from 90% of its initial value to 10%.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

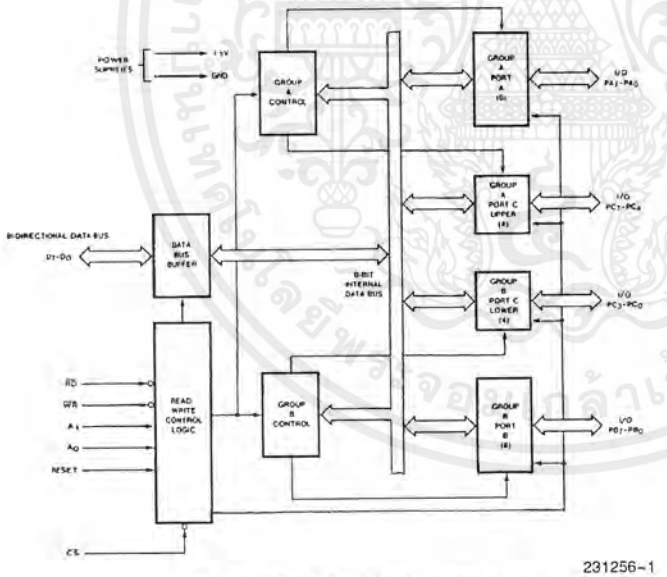
82C55A
CHMOS PROGRAMMABLE PERIPHERAL INTERFACE

- Compatible with all Intel and Most Other Microprocessors
- High Speed, "Zero Wait State" Operation with 8 MHz 8086/88 and 80186/188
- 24 Programmable I/O Pins
- Low Power CHMOS
- Completely TTL Compatible
- Control Word Read-Back Capability
- Direct Bit Set/Reset Capability
- 2.5 mA DC Drive Capability on all I/O Port Outputs
- Available in 40-Pin DIP and 44-Pin PLCC
- Available in EXPRESS
  - Standard Temperature Range
  - Extended Temperature Range

The Intel 82C55A is a high-performance, CHMOS version of the industry standard 8255A general purpose programmable I/O device which is designed for use with all Intel and most other microprocessors. It provides 24 I/O pins which may be individually programmed in 2 groups of 12 and used in 3 major modes of operation. The 82C55A is pin compatible with the NMOS 8255A and 8255A-5.

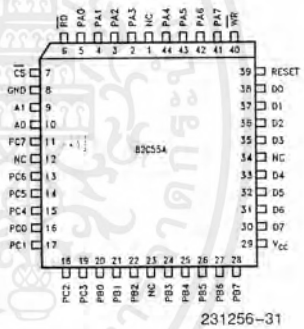
In MODE 0, each group of 12 I/O pins may be programmed in sets of 4 and 8 to be inputs or outputs. In MODE 1, each group may be programmed to have 8 lines of input or output. 3 of the remaining 4 pins are used for handshaking and interrupt control signals. MODE 2 is a strobed bi-directional bus configuration.

The 82C55A is fabricated on Intel's advanced CHMOS III technology which provides low power consumption with performance equal to or greater than the equivalent NMOS product. The 82C55A is available in 40-pin DIP and 44-pin plastic leaded chip carrier (PLCC) packages.

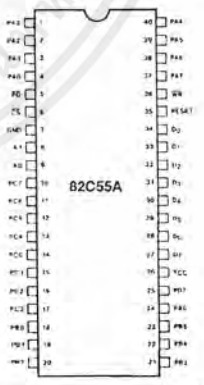


231256-1

Figure 1. 82C55A Block Diagram



231256-31



231256-2

Figure 2. 82C55A Pinout

Diagrams are for pin reference only. Package sizes are not to scale.

October 1995

Order Number: 231256-004

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Table 1. Pin Description

Symbol	Pin Number Dip	Pin Number PLCC	Type	Name and Function
PA_{3-0}	1–4	2–5	I/O	**PORT A, PINS 0–3:** Lower nibble of an 8-bit data output latch/buffer and an 8-bit data input latch.
$\overline{RD}$	5	6	I	**READ CONTROL:** This input is low during CPU read operations.
$\overline{CS}$	6	7	I	**CHIP SELECT:** A low on this input enables the 82C55A to respond to $\overline{RD}$ and $\overline{WR}$ signals. $\overline{RD}$ and WR are ignored otherwise.
GND	7	8		**System Ground**
A_{1-0}	8–9	9–10	I	**ADDRESS:** These input signals, in conjunction $\overline{RD}$ and $\overline{WR}$, control the selection of one of the three ports or the control word registers.

A_1	A_0	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	Input Operation (Read)
0	0	0	1	0	Port A - Data Bus
0	1	0	1	0	Port B - Data Bus
1	0	0	1	0	Port C - Data Bus
1	1	0	1	0	Control Word - Data Bus
					Output Operation (Write)
0	0	1	0	0	Data Bus - Port A
0	1	1	0	0	Data Bus - Port B
1	0	1	0	0	Data Bus - Port C
1	1	1	0	0	Data Bus - Control
					Disable Function
X	X	X	X	1	Data Bus - 3 - State
X	X	1	1	0	Data Bus - 3 - State

PC_{7-4}	10–13	11,13–15	I/O	**PORT C, PINS 4–7:** Upper nibble of an 8-bit data output latch/buffer and an 8-bit data input buffer (no latch for input). This port can be divided into two 4-bit ports under the mode control. Each 4-bit port contains a 4-bit latch and it can be used for the control signal outputs and status signal inputs in conjunction with ports A and B.
PC_{0-3}	14–17	16–19	I/O	**PORT C, PINS 0–3:** Lower nibble of Port C.
PB_{0-7}	18–25	20–22, 24–28	I/O	**PORT B, PINS 0–7:** An 8-bit data output latch/buffer and an 8-bit data input buffer.
V_{CC}	26	29		**SYSTEM POWER:** + 5V Power Supply.
D_{7-0}	27–34	30–33, 35–38	I/O	**DATA BUS:** Bi-directional, tri-state data bus lines, connected to system data bus.
RESET	35	39	I	**RESET:** A high on this input clears the control register and all ports are set to the input mode.
$\overline{WR}$	36	40	I	**WRITE CONTROL:** This input is low during CPU write operations.
PA_{7-4}	37–40	41–44	I/O	**PORT A, PINS 4–7:** Upper nibble of an 8-bit data output latch/buffer and an 8-bit data input latch.
NC		1, 12, 23, 34		No Connect

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

82C55A FUNCTIONAL DESCRIPTION

General

The 82C55A is a programmable peripheral interface device designed for use in Intel microcomputer systems. Its function is that of a general purpose I/O component to interface peripheral equipment to the microcomputer system bus. The functional configuration of the 82C55A is programmed by the system software so that normally no external logic is necessary to interface peripheral devices or structures.

Data Bus Buffer

This 3-state bidirectional 8-bit buffer is used to interface the 82C55A to the system data bus. Data is transmitted or received by the buffer upon execution of input or output instructions by the CPU. Control words and status information are also transferred through the data bus buffer.

Read/Write and Control Logic

The function of this block is to manage all of the internal and external transfers of both Data and Control or Status words. It accepts inputs from the CPU Address and Control busses and in turn, issues commands to both of the Control Groups.

Group A and Group B Controls

The functional configuration of each port is programmed by the systems software. In essence, the CPU "outputs" a control word to the 82C55A. The control word contains information such as "mode", "bit set", "bit reset", etc., that initializes the functional configuration of the 82C55A.

Each of the Control blocks (Group A and Group B) accepts "commands" from the Read/Write Control Logic, receives "control words" from the internal data bus and issues the proper commands to its associated ports.

Control Group A - Port A and Port C upper (C7–C4)
Control Group B - Port B and Port C lower (C3–C0)

The control word register can be both written and read as shown in the address decode table in the pin descriptions. Figure 6 shows the control word format for both Read and Write operations. When the control word is read, bit D7 will always be a logic "1", as this implies control word mode information.

Ports A, B, and C

The 82C55A contains three 8-bit ports (A, B, and C). All can be configured in a wide variety of functional characteristics by the system software but each has its own special features or "personality" to further enhance the power and flexibility of the 82C55A.

Port A. One 8-bit data output latch/buffer and one 8-bit input latch buffer. Both "pull-up" and "pull-down" bus hold devices are present on Port A.

Port B. One 8-bit data input/output latch/buffer. Only "pull-up" bus hold devices are present on Port B.

Port C. One 8-bit data output latch/buffer and one 8-bit data input buffer (no latch for input). This port can be divided into two 4-bit ports under the mode control. Each 4-bit port contains a 4-bit latch and it can be used for the control signal outputs and status signal inputs in conjunction with ports A and B. Only "pull-up" bus hold devices are present on Port C.

See Figure 4 for the bus-hold circuit configuration for Port A, B, and C.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

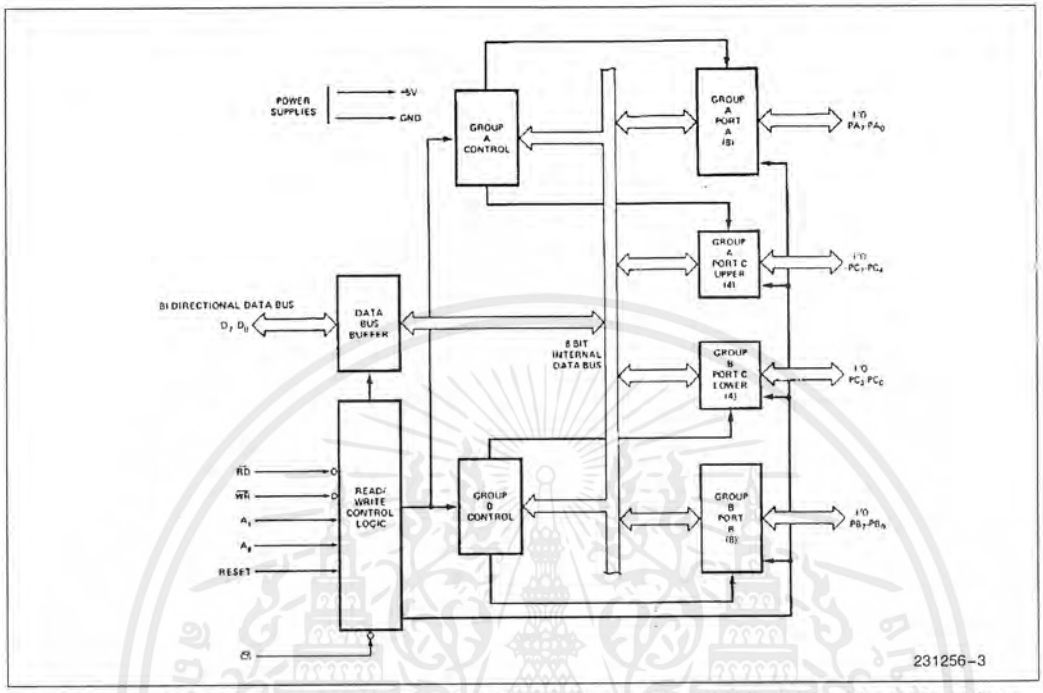
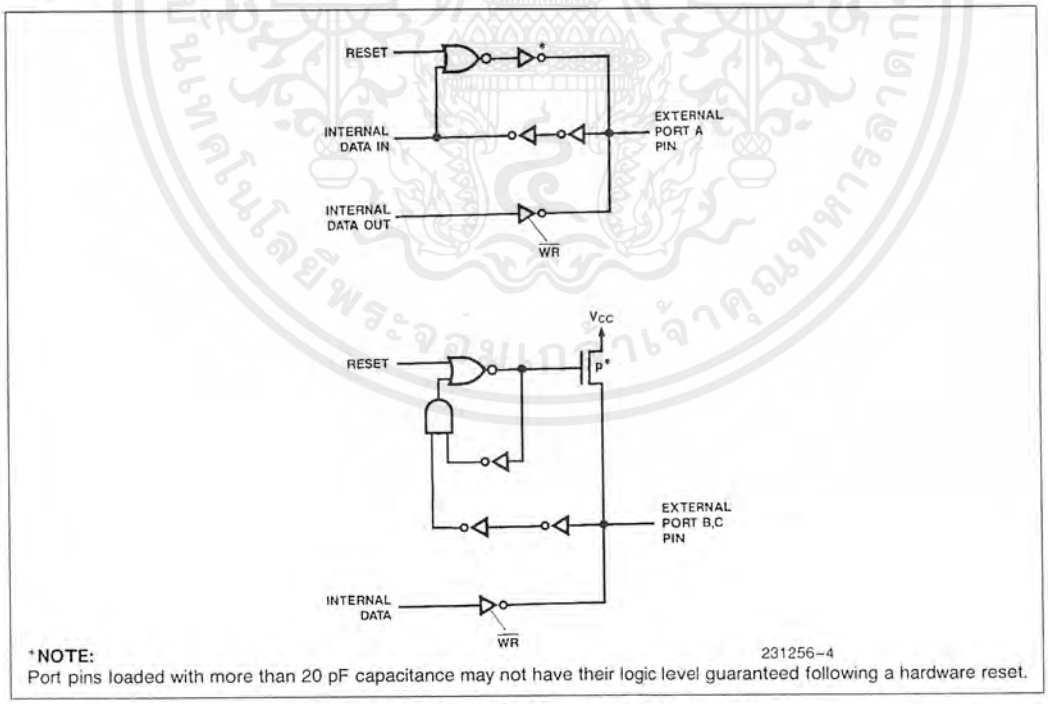


Figure 3. 82C55A Block Diagram Showing Data Bus Buffer and Read/Write Control Logic Functions



*NOTE:
Port pins loaded with more than 20 pF capacitance may not have their logic level guaranteed following a hardware reset.

Figure 4. Port A, B, C, Bus-hold Configuration

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

82C55A OPERATIONAL DESCRIPTION

Mode Selection

There are three basic modes of operation that can be selected by the system software:

Mode 0 — Basic input/output
Mode 1 — Strobed Input/output
Mode 2 — Bi-directional Bus

When the reset input goes "high" all ports will be set to the input mode with all 24 port lines held at a logic "one" level by the internal bus hold devices (see Figure 4 Note). After the reset is removed the 82C55A can remain in the input mode with no additional initialization required. This eliminates the need for pullup or pulldown devices in "all CMOS" designs. During the execution of the system program, any of the other modes may be selected by using a single output instruction. This allows a single 82C55A to service a variety of peripheral devices with a simple software maintenance routine.

The modes for Port A and Port B can be separately defined, while Port C is divided into two portions as required by the Port A and Port B definitions. All of the output registers, including the status flip-flops, will be reset whenever the mode is changed. Modes may be combined so that their functional definition can be "tailored" to almost any I/O structure. For instance; Group B can be programmed in Mode 0 to monitor simple switch closings or display computational results, Group A could be programmed in Mode 1 to monitor a keyboard or tape reader on an interrupt-driven basis.

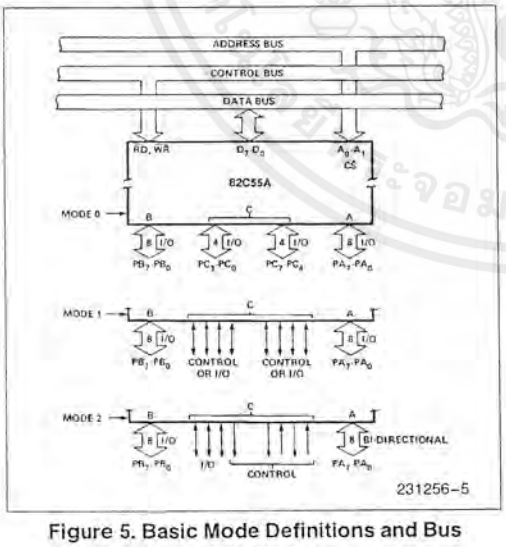


Figure 5. Basic Mode Definitions and Bus Interface

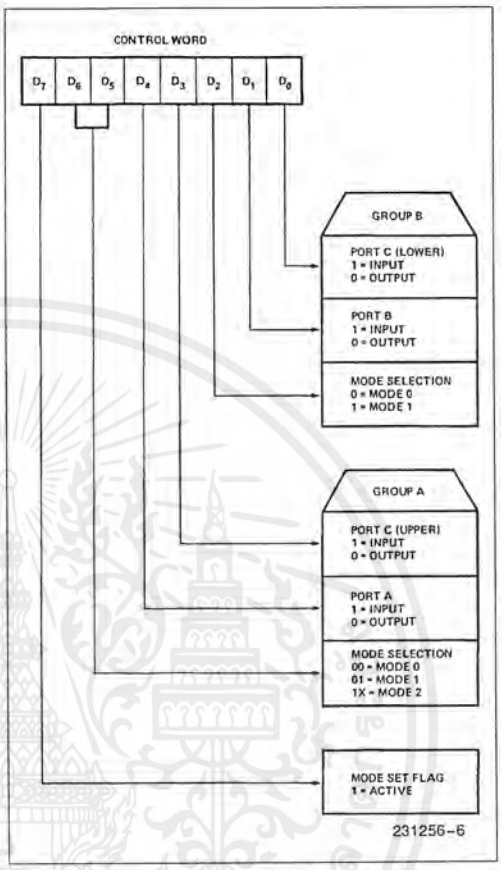


Figure 6. Mode Definition Format

The mode definitions and possible mode combinations may seem confusing at first but after a cursory review of the complete device operation a simple, logical I/O approach will surface. The design of the 82C55A has taken into account things such as efficient PC board layout, control signal definition vs PC layout and complete functional flexibility to support almost any peripheral device with no external logic. Such design represents the maximum use of the available pins.

Single Bit Set/Reset Feature

Any of the eight bits of Port C can be Set or Reset using a single OUTput instruction. This feature reduces software requirements in Control-based applications.

When Port C is being used as status/control for Port A or B, these bits can be set or reset by using the Bit Set/Reset operation just as if they were data output ports.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

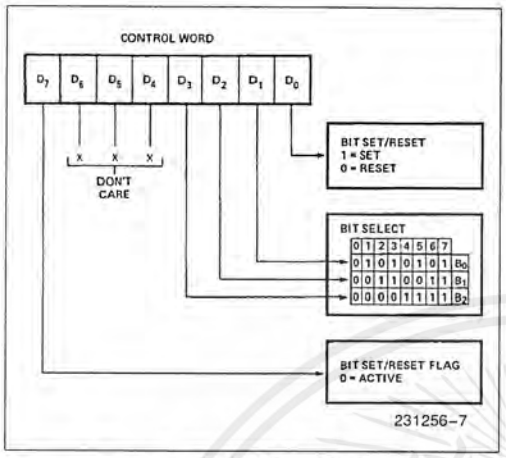


Figure 7. Bit Set/Reset Format

Interrupt Control Functions

When the 82C55A is programmed to operate in mode 1 or mode 2, control signals are provided that can be used as interrupt request inputs to the CPU. The interrupt request signals, generated from port C, can be inhibited or enabled by setting or resetting the associated INTE flip-flop, using the bit set/reset function of port C.

This function allows the Programmer to disallow or allow a specific I/O device to interrupt the CPU without affecting any other device in the interrupt structure.

INTE flip-flop definition:

(BIT-SET)—INTE is SET—Interrupt enable
(BIT-RESET)—INTE is RESET—Interrupt disable

Note:
All Mask flip-flops are automatically reset during mode selection and device Reset.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

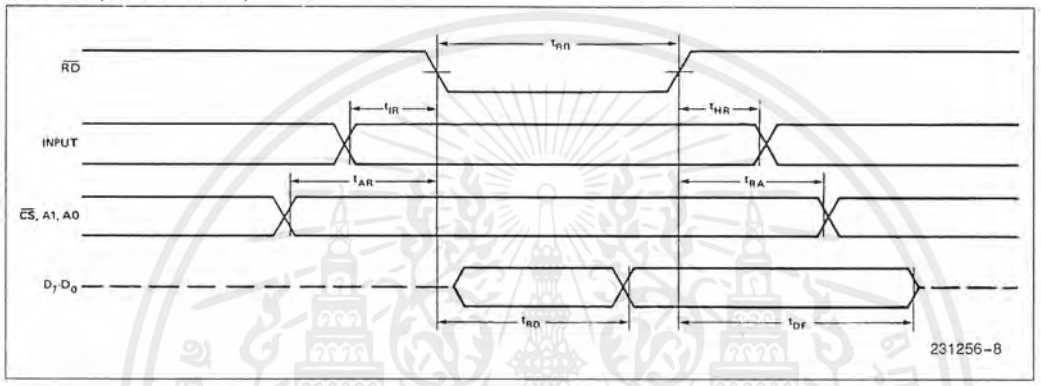
Operating Modes

Mode 0 (Basic Input/Output). This functional configuration provides simple input and output operations for each of the three ports. No "handshaking" is required, data is simply written to or read from a specified port.

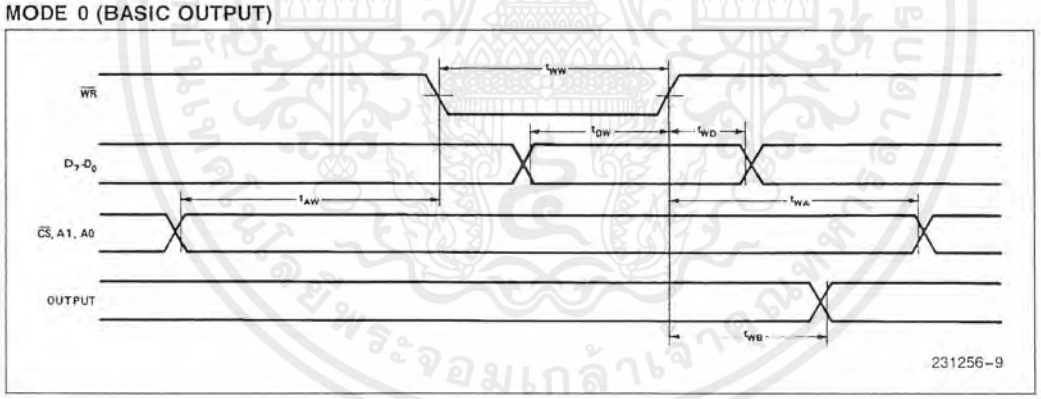
Mode 0 Basic Functional Definitions:

- Two 8-bit ports and two 4-bit ports.
- Any port can be input or output.
- Outputs are latched.
- Inputs are not latched.
- 16 different Input/Output configurations are possible in this Mode.

MODE 0 (BASIC INPUT)



MODE 0 (BASIC OUTPUT)

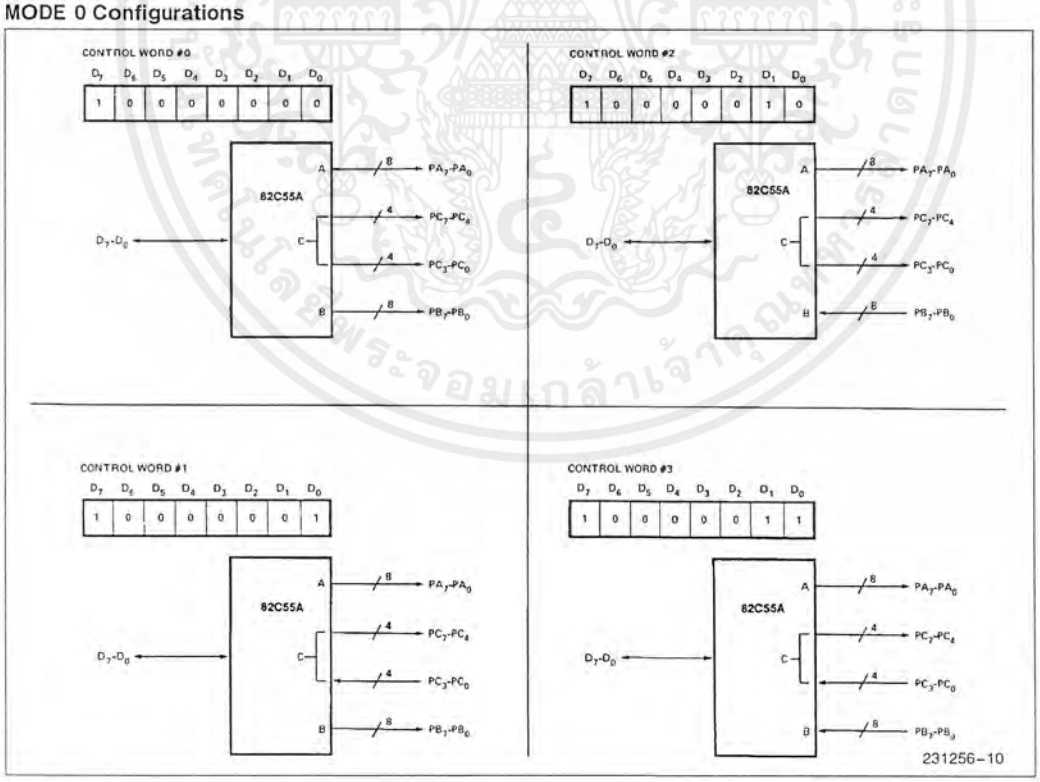


เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

MODE 0 Port Definition

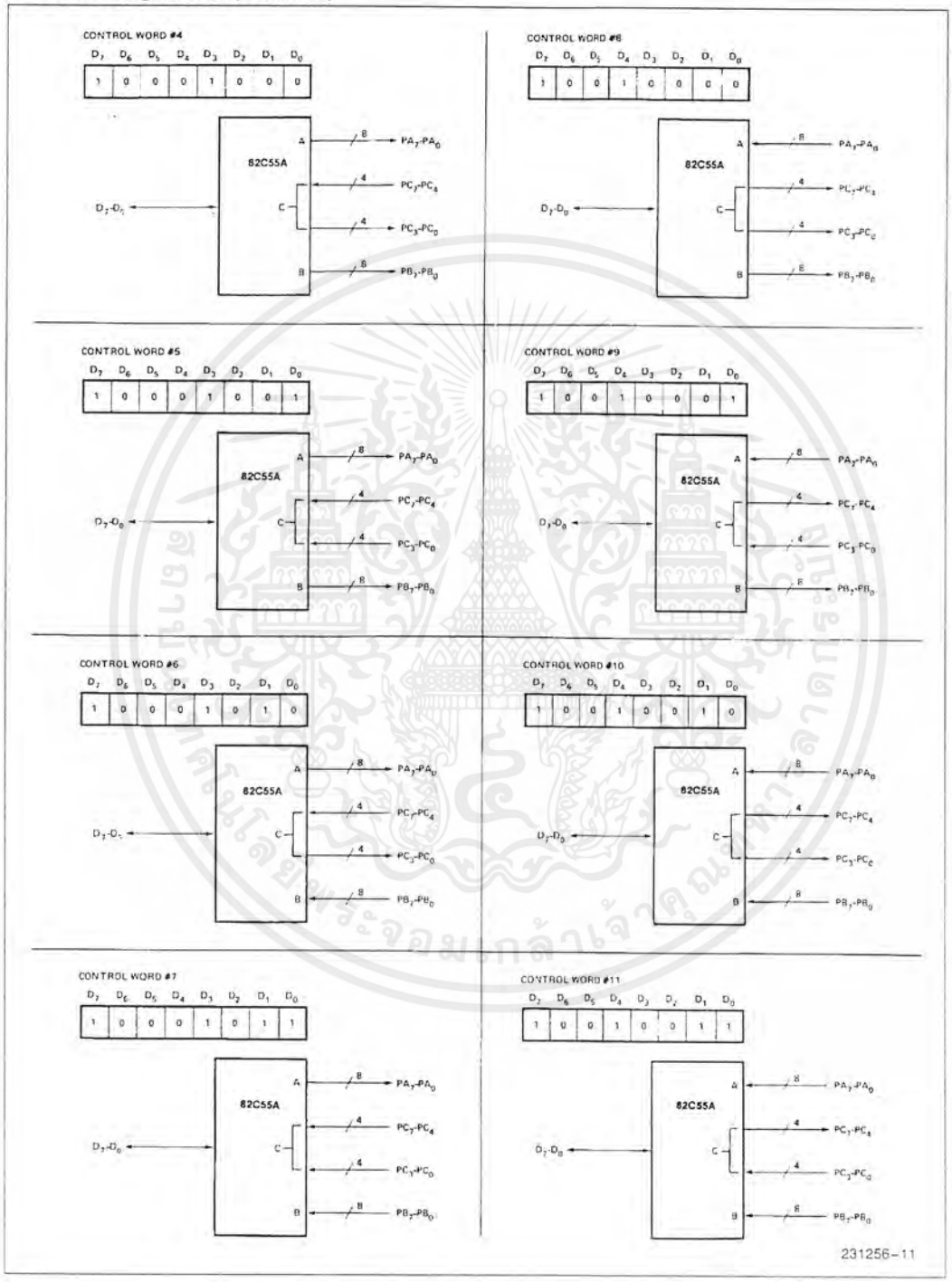
A		B		GROUP A			GROUP B	
D_4	D_3	D_1	D_0	PORT A	PORT C (UPPER)	#	PORT B	PORT C (LOWER)
0	0	0	0	OUTPUT	OUTPUT	0	OUTPUT	OUTPUT
0	0	0	1	OUTPUT	OUTPUT	1	OUTPUT	INPUT
0	0	1	0	OUTPUT	OUTPUT	2	INPUT	OUTPUT
0	0	1	1	OUTPUT	OUTPUT	3	INPUT	INPUT
0	1	0	0	OUTPUT	INPUT	4	OUTPUT	OUTPUT
0	1	0	1	OUTPUT	INPUT	5	OUTPUT	INPUT
0	1	1	0	OUTPUT	INPUT	6	INPUT	OUTPUT
0	1	1	1	OUTPUT	INPUT	7	INPUT	INPUT
1	0	0	0	INPUT	OUTPUT	8	OUTPUT	OUTPUT
1	0	0	1	INPUT	OUTPUT	9	OUTPUT	INPUT
1	0	1	0	INPUT	OUTPUT	10	INPUT	OUTPUT
1	0	1	1	INPUT	OUTPUT	11	INPUT	INPUT
1	1	0	0	INPUT	INPUT	12	OUTPUT	OUTPUT
1	1	0	1	INPUT	INPUT	13	OUTPUT	INPUT
1	1	1	0	INPUT	INPUT	14	INPUT	OUTPUT
1	1	1	1	INPUT	INPUT	15	INPUT	INPUT

MODE 0 Configurations



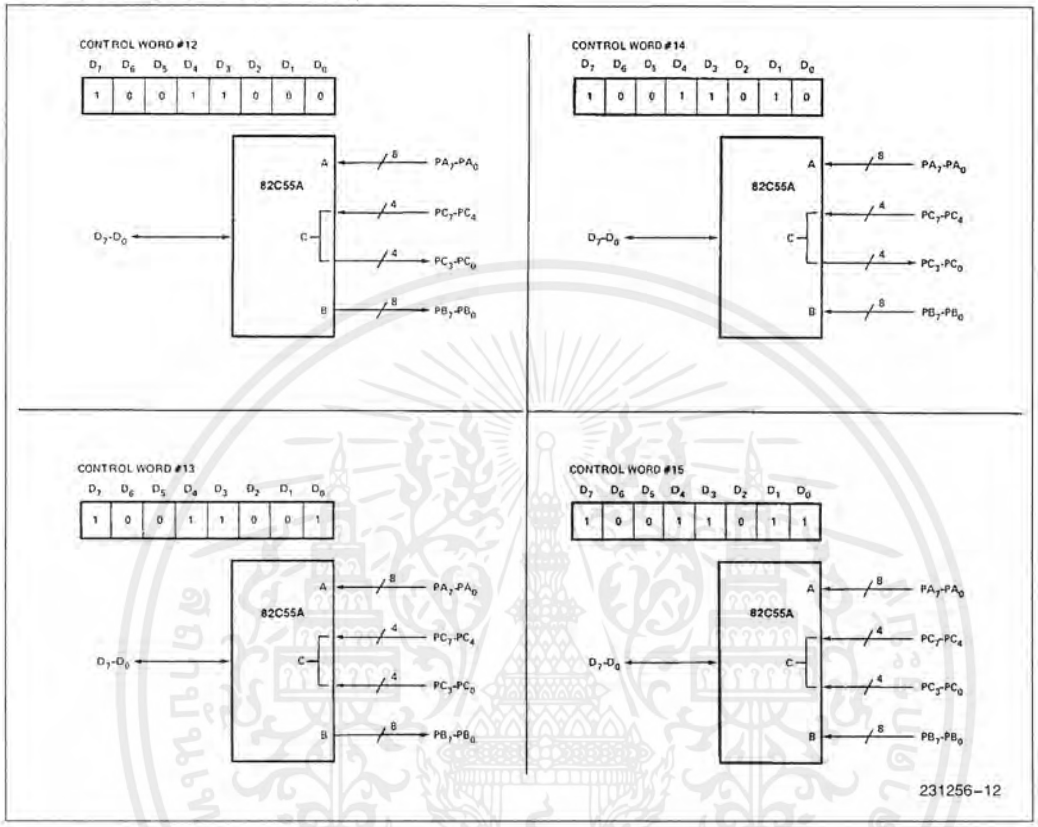
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

MODE 0 Configurations (Continued)



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

MODE 0 Configurations (Continued)



Operating Modes

MODE 1 (Strobed Input/Output). This functional configuration provides a means for transferring I/O data to or from a specified port in conjunction with strobes or "handshaking" signals. In mode 1, Port A and Port B use the lines on Port C to generate or accept these "handshaking" signals.

Mode 1 Basic functional Definitions:

- Two Groups (Group A and Group B).
- Each group contains one 8-bit data port and one 4-bit control/data port.
- The 8-bit data port can be either input or output Both inputs and outputs are latched.
- The 4-bit port is used for control and status of the 8-bit data port.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Input Control Signal Definition

$\overline{STB}$ (Strobe Input). A "low" on this input loads data into the input latch.

IBF (Input Buffer Full F/F)

A "high" on this output indicates that the data has been loaded into the input latch; in essence, an acknowledgement. IBF is set by $\overline{STB}$ input being low and is reset by the rising edge of the $\overline{RD}$ input.

INTR (Interrupt Request)

A "high" on this output can be used to interrupt the CPU when an input device is requesting service. INTR is set by the $\overline{STB}$ is a "one", IBF is a "one" and INTE is a "one". It is reset by the falling edge of $\overline{RD}$. This procedure allows an input device to request service from the CPU by simply strobing its data into the port.

INTE A

Controlled by bit set/reset of PC_4.

INTE B

Controlled by bit set/reset of PC_2.

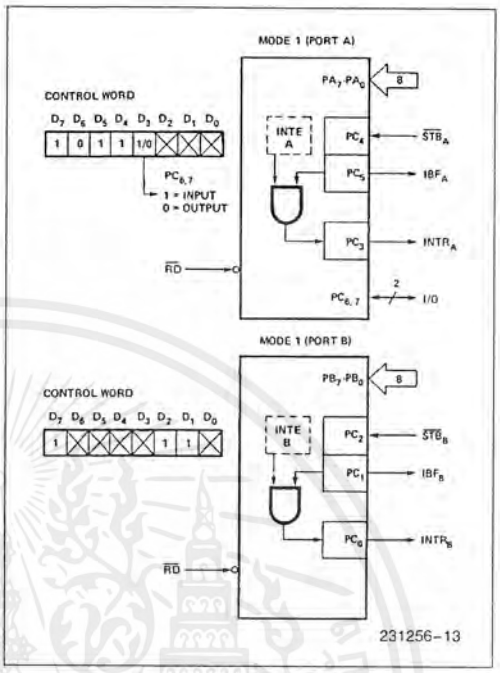


Figure 8. MODE 1 Input

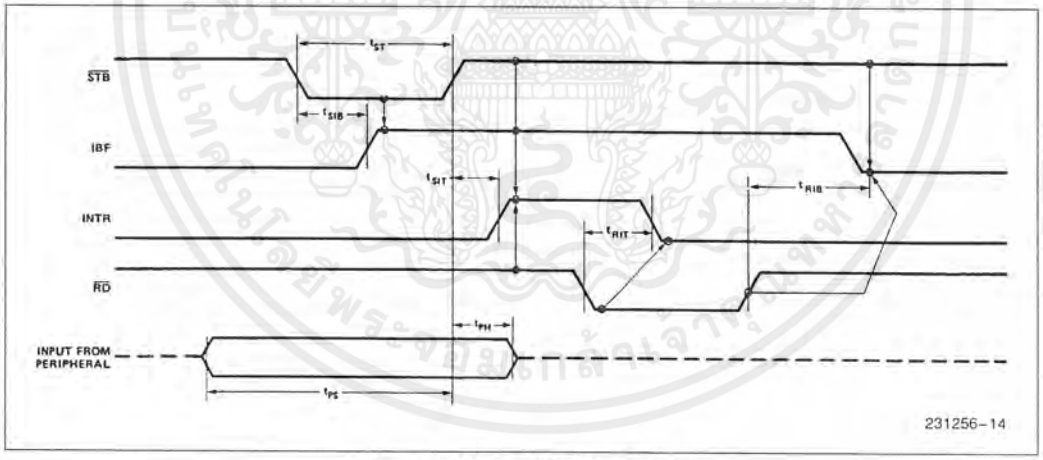


Figure 9. MODE 1 (Strobed Input)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Output Control Signal Definition

$\overline{OBF}$ (Output Buffer Full F/F). The $\overline{OBF}$ output will go "low" to indicate that the CPU has written data out to the specified port. The $\overline{OBF}$ F/F will be set by the rising edge of the $\overline{WR}$ input and reset by $\overline{ACK}$ Input being low.

$\overline{ACK}$ (Acknowledge Input). A "low" on this input informs the 82C55A that the data from Port A or Port B has been accepted. In essence, a response from the peripheral device indicating that it has received the data output by the CPU.

INTR (Interrupt Request). A "high" on this output can be used to interrupt the CPU when an output device has accepted data transmitted by the CPU. INTR is set when $\overline{ACK}$ is a "one", $\overline{OBF}$ is a "one" and INTE is a "one". It is reset by the falling edge of $\overline{WR}$.

INTE A

Controlled by bit set/reset of PC_6.

INTE B

Controlled by bit set/reset of PC_2.

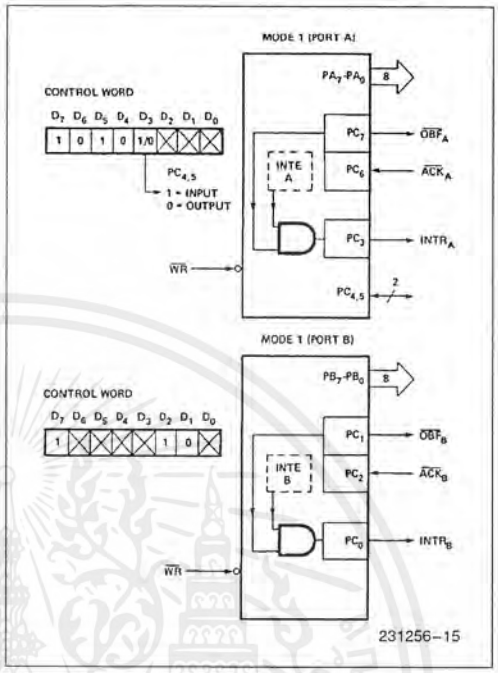


Figure 10. MODE 1 Output

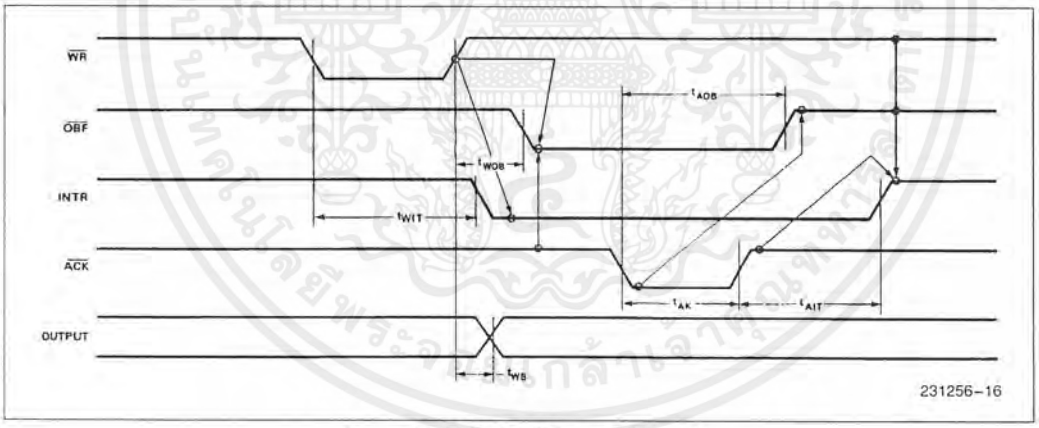


Figure 11. MODE 1 (Strobed Output)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Combinations of MODE 1

Port A and Port B can be individually defined as input or output in Mode 1 to support a wide variety of strobed I/O applications.

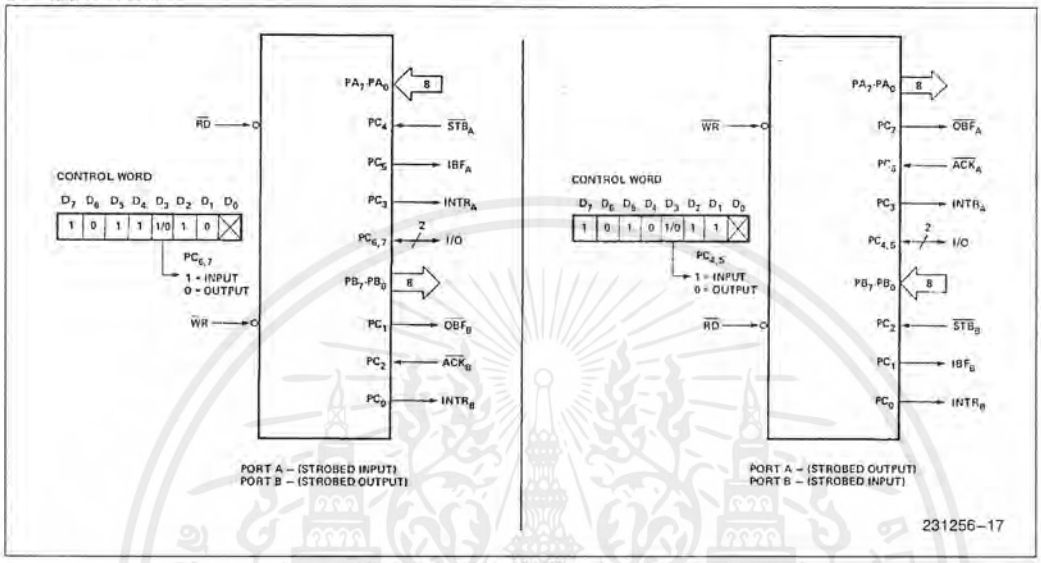


Figure 12. Combinations of MODE 1

Operating Modes

MODE 2 (Strobed Bidirectional Bus I/O). This functional configuration provides a means for communicating with a peripheral device or structure on a single 8-bit bus for both transmitting and receiving data (bidirectional bus I/O). "Handshaking" signals are provided to maintain proper bus flow discipline in a similar manner to MODE 1. Interrupt generation and enable/disable functions are also available.

MODE 2 Basic Functional Definitions:

- Used in Group A **only.**
- One 8-bit, bi-directional bus port (Port A) and a 5-bit control port (Port C).
- Both inputs and outputs are latched.
- The 5-bit control port (Port C) is used for control and status for the 8-bit, bi-directional bus port (Port A).

Bidirectional Bus I/O Control Signal Definition

INTR (Interrupt Request). A high on this output can be used to interrupt the CPU for input or output operations.

Output Operations

$\overline{OBF}$ (Output Buffer Full). The $\overline{OBF}$ output will go "low" to indicate that the CPU has written data out to port A.

$\overline{ACK}$ (Acknowledge). A "low" on this input enables the tri-state output buffer of Port A to send out the data. Otherwise, the output buffer will be in the high impedance state.

INTE 1 (The INTE Flip-Flop Associated with $\overline{OBF}$). Controlled by bit set/reset of PC_6.

Input Operations

$\overline{STB}$ (Strobe Input). A "low" on this input loads data into the input latch.

IBF (Input Buffer Full F/F). A "high" on this output indicates that data has been loaded into the input latch.

INTE 2 (The INTE Flip-Flop Associated with IBF). Controlled by bit set/reset of PC_4.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Combinations of MODE 1

Port A and Port B can be individually defined as input or output in Mode 1 to support a wide variety of strobed I/O applications.

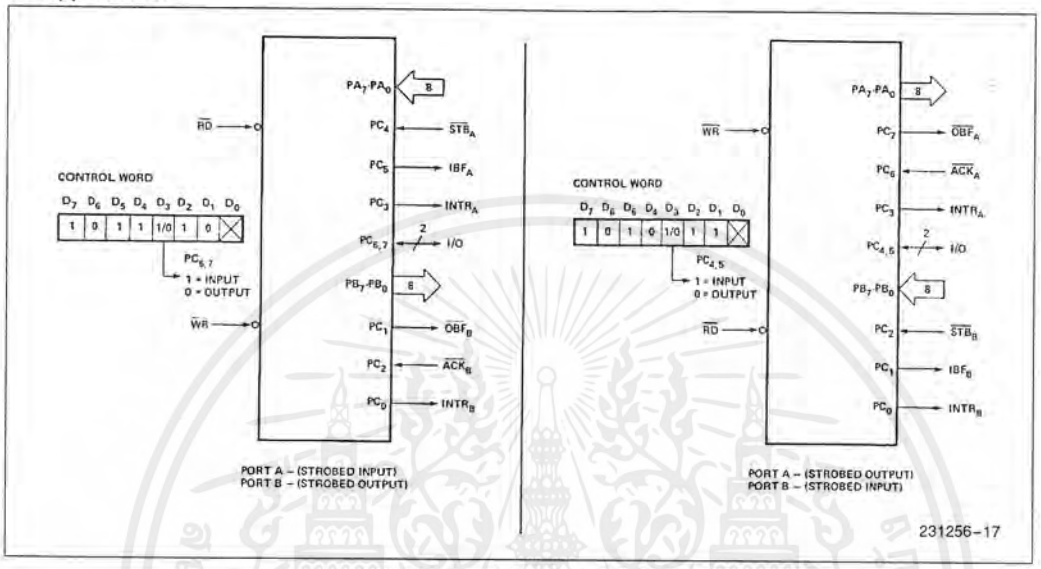


Figure 12. Combinations of MODE 1

Operating Modes

MODE 2 (Strobed Bidirectional Bus I/O). This functional configuration provides a means for communicating with a peripheral device or structure on a single 8-bit bus for both transmitting and receiving data (bidirectional bus I/O). "Handshaking" signals are provided to maintain proper bus flow discipline in a similar manner to MODE 1. Interrupt generation and enable/disable functions are also available.

MODE 2 Basic Functional Definitions:

- Used in Group A **only.**
- One 8-bit, bi-directional bus port (Port A) and a 5-bit control port (Port C).
- Both inputs and outputs are latched.
- The 5-bit control port (Port C) is used for control and status for the 8-bit, bi-directional bus port (Port A).

Bidirectional Bus I/O Control Signal Definition

INTR (Interrupt Request). A high on this output can be used to interrupt the CPU for input or output operations.

Output Operations

$\overline{OBF}$ (Output Buffer Full). The $\overline{OBF}$ output will go "low" to indicate that the CPU has written data out to port A.

$\overline{ACK}$ (Acknowledge). A "low" on this input enables the tri-state output buffer of Port A to send out the data. Otherwise, the output buffer will be in the high impedance state.

INTE 1 (The INTE Flip-Flop Associated with $\overline{OBF}$). Controlled by bit set/reset of PC_6.

Input Operations

$\overline{STB}$ (Strobe Input). A "low" on this input loads data into the input latch.

IBF (Input Buffer Full F/F). A "high" on this output indicates that data has been loaded into the input latch.

INTE 2 (The INTE Flip-Flop Associated with IBF). Controlled by bit set/reset of PC_4.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

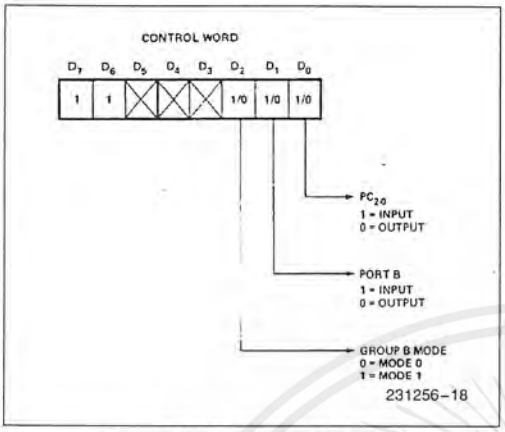


Figure 13. MODE Control Word

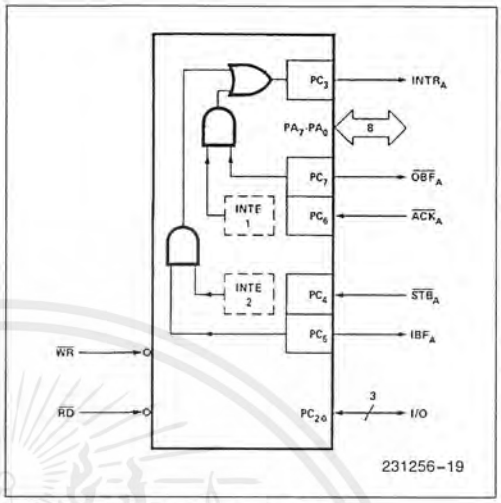


Figure 14. MODE 2

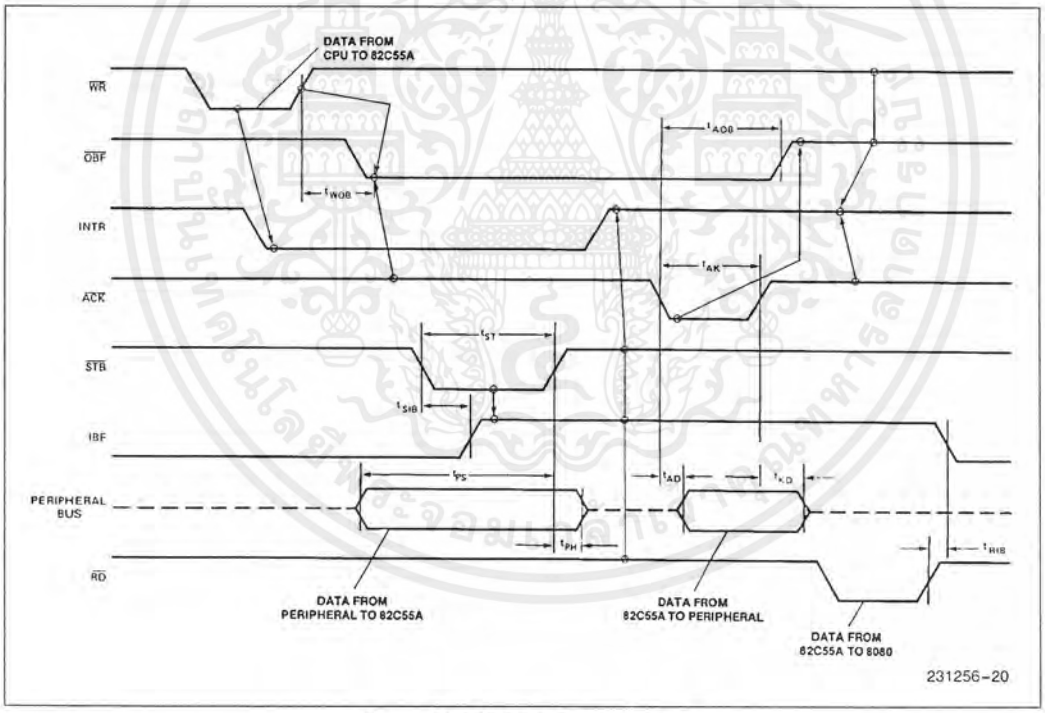


Figure 15. MODE 2 (Bidirectional)

NOTE:
Any sequence where $\overline{WR}$ occurs before $\overline{ACK}$, and $\overline{STB}$ occurs before $\overline{RD}$ is permissible.
($INTR = IBF \bullet \overline{MASK} \bullet \overline{STB} \bullet \overline{RD} + \overline{OBF} \bullet \overline{MASK} \bullet \overline{ACK} \bullet \overline{WR}$)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

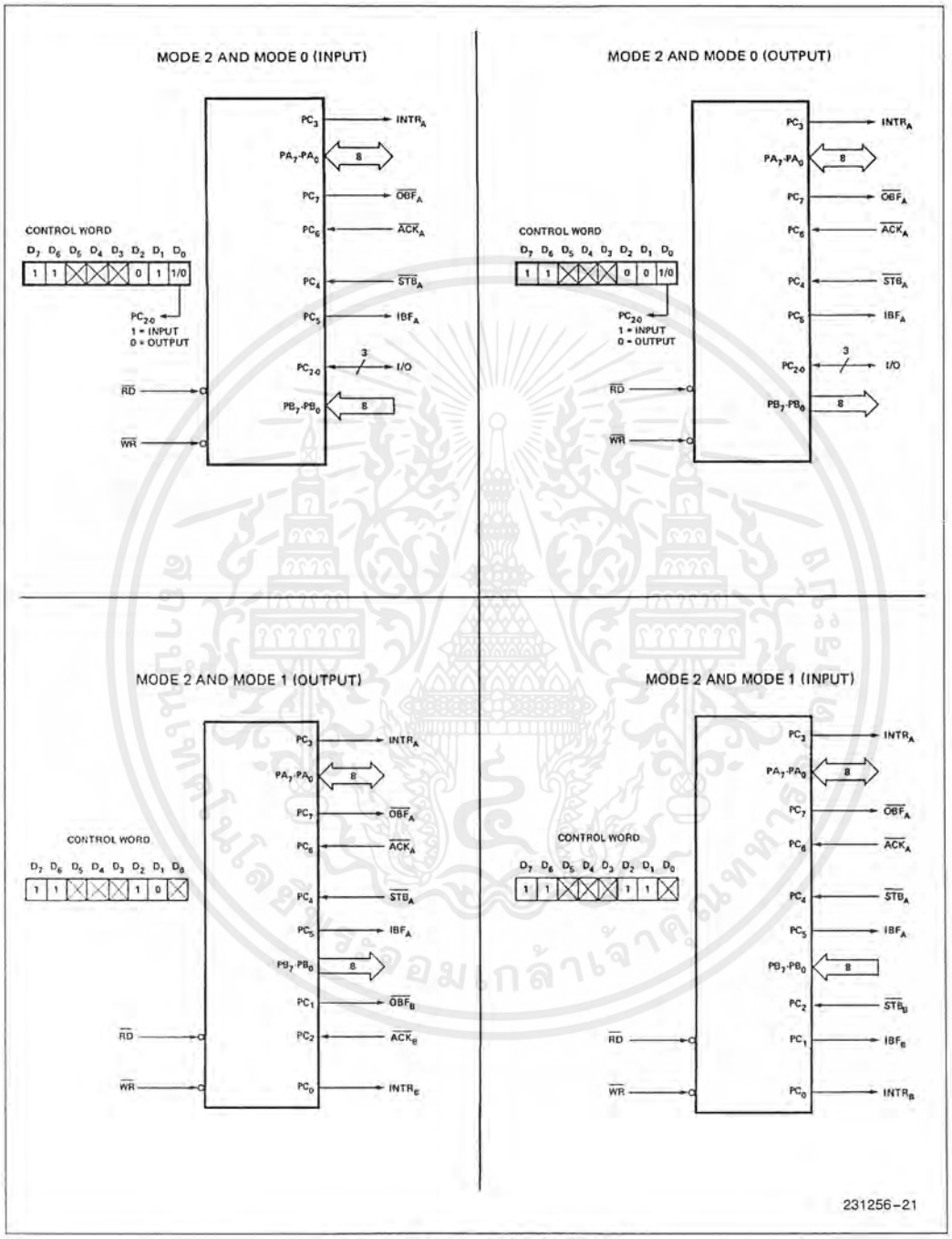


Figure 16. MODE 1/4 Combinations

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หนังสืออ้างอิง

[1] BILL MILNE, " THE SOUND DECISION ",DATA COMMUNICATION,PAGE 75 – 84 MARCH 1998.

[2] Pat Bonner , " NETWORK PROGRAMMING WITH WINDOWS SOCKETS ",Prentice-Hall,Inc.

[3] Viktor Toth , " Visual C++ 4 " ,First editon, Sams publishing,1996.

[4] จารุวรรณ ระวิภักดิ์, " รู้ลึก รู้จริง Style Borland Delphi ",บริษัท เฟิสท์ แปซิฟิก มีเดีย จำกัด.

[5] กิตติ ภักดีวัฒนะกุล, " NETSCAPE ALL IN ONE " , บริษัท ดวงกมลสมัย จำกัด 2539.

[6] ยืน ภู่วรวรรณ . " Voice Packet ", บทความจากไมโครคอมพิวเตอร์ หน้า 130 – 138 ฉบับที่ 153, บริษัท ซีเอ็ดยูเคชั่น จำกัด.

[7] ประพันธ์ อัศวภานุวัฒน์, " Communication : NetChat " , บทความจากไมโครคอมพิวเตอร์ หน้า 172 – 179 ฉบับที่ 153, บริษัท ซีเอ็ดยูเคชั่น จำกัด.

[8] พ.อ. เจนวิทย์ เหลืองอร่าม, " การใช้ Turbo C++ เขียนโปรแกรมภาษา C ",บริษัท เยลโลการพิมพ์ จำกัด,กรุงเทพมหานคร.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กิตติกรรมประกาศ

ปริญญานิพนธ์นี้ขอกราบขอบคุณทุกท่านที่กล่าวถึงต่อไปนี้เพราะนี้เป็นส่วนที่เป็นแรงผลักดันให้ข้าพเจ้ามีความตั้งใจและความพยายามในการทำโครงงานชิ้นนี้จนสมบูรณ์

- บิดา มารดาของข้าพเจ้า ที่คอยให้กำลังใจในการทำโครงงาน ให้สามารถฝ่าฟันอุปสรรคไปได้ด้วยดี
- อาจารย์ รศ.ดร. สุวิพล สิทธิชีวภาค และ ผศ. เกรียงไกร วงศ์โรจนภรณ์ ที่ช่วยให้คำปรึกษาทางด้านโปรเจค และคำแนะนำสิ่งดี ๆ ที่มีประโยชน์ต่อข้าพเจ้า
- คุณ ดวงพร ชนะอรรถกาล ช่วยให้งานพิมพ์ผ่านไปได้ด้วยดี และได้ผลงานที่ประณีต

คุณ เทอดศักดิ์ ธนกิจประภา พี่ปริญญาโทผู้เชี่ยวชาญในการเขียนโปรแกรมบนระบบเครือข่ายกับมัลติมีเดียที่ให้คำปรึกษาและสอนการเขียนโปรแกรม ตลอดคำแนะนำที่ดีตลอดการทำงาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้