การสั่งงานอุปกรณ์ไฟฟ้าผ่านทางคู่สายโทรศัพท์โดยใช้เสียง

REMOTE TELEPHONE VOICE COMMANDER

โดย

นายกิติกรณ์ ตัณยายุทธ์

นายวันประชา เชาวลิตวงศ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2541

การสั่งงานอุปกรณ์ไฟฟ้าผ่านทางคู่สายโทรศัพท์โดยใช้เสียง

REMOTE TELEPHONE VOICE COMMANDER

โดย

นายกิติกรณ์ ตัณยายุทธ์ 38014029

นาวันประชา เชาวลิตวงศ์ 38014453

อาจารย์ที่ปรึกษา

รศ.ดร.กอบชัย เดชหาญ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2541

ปริญญานิพนธ์ปีการศึกษา 2541

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การสั่งงานอุกรณ์ไฟฟ้าผ่านทางคู่สายโทรศัพท์โดยใช้เสียง

Remote Telephone Voice Commander

ผู้จัดทำ

1.นายกิติกรณ์ ตัณยายุทธ์ 38014029

2.นายวันประชา เชาวลิตวงศ์ 38014453

อาจารย์ที่ปรึกษา

(รศ.ดร.กอบชัย เดชหาญ)

การสั่งงานอุปกรณ์ไฟฟ้าผ่านทางคู่สายโทรศัพท์โดยใช้เสียง

Remote Telephone Voice Commander

โดย นายกิติกรณ์ ตัณฑายุทธ์ 38014029

นายวันประชา เชาวลิตวงศ์ 38014453

อาจารย์ที่ปรึกษา รศ.ดร.กอบชัย เดชหาญ

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้มีจุดประสงค์เพื่อที่จะเพิ่มประสิทธิภาพในการเชื่อมต่อระหว่างเครื่อง PC กับโทรศัพท์ โครงงานนี้จึงถูกคิดขึ้นมาเพื่อทำให้เกิดความสะดวกสบายในชีวิตประจำวัน โดยที่เราสามารถที่จะสั่งงานเปิดปิดอุปกรณ์ไฟฟ้าผ่านทางคู่สายโทรศัพท์โดยใช้เสียงพูดของเราได้ โดยเราสามารถที่จะควบคุมได้แม้ว่าเราจะออกไปนอกบ้าน

หลักการโดยทั่วไปที่ใช้ก็มีดังนี้ 1.การเชื่อมต่อระหว่างคู่สายโทรศัพท์กับคอมพิวเตอร์ เราได้สร้างเครื่องตอบรับโทรศัพท์อัตโนมัติใช้ในการตอบรับ และเราจะใช้ทรานฟอร์มเมอร์เพื่อเอาสัญญาณเสียงเข้าไปยังคอมพิวเตอร์ผ่านทางซาวน์การ์ด 2.การจดจำเสียง เราจะใช้วิธีการหาค่าองค์ประกอบของเสียงโดยใช้วิธีการ LPC(Linear Predictive Coding) และใช้การจัดระดับเวคเตอร์ VQ(Vector Quantization) และใช้การสร้างแบบจำลองของเสียงด้วยวิธีการ HMM(Hidden Markov Model) และในการสั่งงานเราจะใช้การหาค่าขนาดของพลังงานของเสียงใช้ในการตัดคำที่ใช้สั่งงาน 3.การอินเตอร์เฟสระหว่างอุปกรณ์ไฟฟ้ากับคอมพิวเตอร์เราจะใช้การ์ดโปรโตไทป์นำข้อมูลในบัสของ PC เพื่อไปเชื่อมต่อกับ ชิป 8255 แล้วนำสัญญาณไปขับรีเลย์

ABSTRACT

The propose of this project is to optimize the ability of telephone and computer. This project can command the electrical devices by phoning to our home and say the command words that have been already set.

General principles that are used as follws; 1.Principle of connection between telephone line and computer. This project creates an answering machine for answering the calls and using the transformer for sending our voices via sound-card to computer. 2.Principle of Speech Recognition. This project uses LPC(Liner Predictive Coding) VQ(Vector Quantization) and HMM(Hidden Markov Model). This project can also split the command words by using speech energy. 3.Principle of interfacing between electrical devices and computer. This project uses Prototype card for sending data at PC's data bus to 8255 which connects with relay.

สารบัญ

สารบัญรูปภาพ

สารบัญตาราง

บทที่1
บทนำ

เนื่องจากความเจริญก้าวหน้าทางเทคโนโลยีในปัจจุบัน ทำให้การติดต่อสื่อสารเป็นไปได้อย่างสะดวกรวดเร็วและประหยัด ทำให้อุปกรณ์อำนวยความสะดวกมีราคาถูก ดังนั้นเราจึงสามารถนำเทคโนโลยีใหม่ๆ เข้ามาช่วยอำนวยความสะดวกในชีวิตประจำวัน ของเราได้

ในอดีตมีการวิเคราะห์ว่าการสอนให้คอมพิวเตอร์สามารถรู้จำเสียงพูดของมนุษย์นั้นทำไม่ได้ เนื่องจากการพูดของมนุษย์มีความซับซ้อนและมีความแตกต่างกันในแต่ละบุคคล ความเร็วในการพูด รวมถึงเสียงสูงต่ำของคนแต่ละคน แต่ก็มีการวิจัยเพื่อให้คอมพิวเตอร์สามารถรู้จำเสียงพูดเรื่อยมา จนในปัจจุบันการรู้จำเสียงพูดสามารถจะใช้งานได้ดี และมีการนำไปใช้กับอุปกรณ์ต่าง ๆ เช่น ATM (Automatic Teller Machine) และเครื่องตอบรับโทรศัพท์อัตโนมัติ ถึงแม้คอมพิวเตอร์จะไม่สามารถรู้จำเสียงได้เท่ากับมนุษย์ แต่เราสามารถสอนให้คอมพิวเตอร์รู้จำได้ในขีดจำกัดระดับหนึ่ง โดยมีการกำหนดขอบเขตของการรู้จำ เช่นจำกัดคำที่จะสามารถรับรู้ได้

ปริญญานิพนธ์นี้ได้มีการศึกษาและทดลองนำเอาส่วนของการรู้จำเสียงที่เป็นทฤษฎีมาใช้งานจริง โดยทำในรูปการใช้เสียงสั่งงานเครื่องใช้ไฟฟ้าผ่านทางคู่สายโทรศัพท์ให้ ซึ่งจะได้กล่าวต่อไป

วัตถุประสงค์ของปริญญานิพนธ์ มีดังนี้

- ศึกษาเกี่ยวกับรายละเอียดและสัญญาณต่างๆของคู่สายโทรศัพท์และทำการศึกษาและสร้างเครื่องตอบรับโทรศัพท์อัตโนมัติ และทำการนำเอาสัญญาณจากคู่สายโทรศัพท์ผ่านเข้าเครื่อง PC
- แยกคำพูดของมนุษย์ออกเป็นคำเดี่ยวๆ ได้
- ศึกษาการวิเคราะห์เสียงพูดแบบไม่ขึ้นกับผู้พูด
- นำงานวิจัยเกี่ยวกับการรู้จำเสียงภาษาไทยที่ได้มีผู้พัฒนาไว้ในระดับหนึ่งแล้วมาทำการศึกษาและพัฒนาต่อ โดยจะเน้นพิจารณาและพัฒนาในด้านการนำไปใช้งานจริง
- เพื่อพัฒนาโปรแกรมส่วนต่างๆที่ทำให้คอมพิวเตอร์สามารถรู้จำเสียงพูด และนำโปรแกรมมาใช้งานได้
- ทำการปรับปรุงวิธีการทางทฤษฎีให้สามารถนำมาใช้ในทางปฏิบัติได้ เช่นหาวิธีที่จะช่วยเพิ่มความเร็วในการวิเคราะห์เสียง หรือปรับปรุงขั้นตอนบางอย่างเพื่อลดความผิดพลาดในการวิเคราะห์ เป็นต้น
- เพื่อเป็นพื้นฐานในการประยุกต์ใช้งานเพิ่มเติมต่อไป
- ศึกษาการต่อบอร์ดโปรโตไทป์จากเครื่อง PC
- ศึกษาการทำงานของชิป 8255
- ศึกษาและออกแบบวงจรขับรีเลย์

ขอบเขตของปริญญานิพนธ์

เพื่อให้สามารถใช้เสียงสั่งคอมพิวเตอร์ให้สั่งงานควบคุมอุปกรณ์ไฟฟ้าโดยผ่านทางคู่สายโทรศัพท์ ซึ่งมีขั้นตอนพอสังเขปดังนี้

1.เปิดเครื่อง PC ไว้ แล้วเปิดโปรแกรมสั่งงานอุปกรณ์ไฟฟ้าอัตโนมัติ

2.เมื่ออยู่นอกบ้านก็สามารถที่จะโทรศัพท์เข้ามาที่บ้าน โดยที่จะมีเครื่องตอบรับโทรศัพท์อัตโนมัติคอยตอบรับไว้

3.รับเสียงพูดผ่านทาง Microphone ของโทรศัพท์ โดยที่นำสัญญาณเสียงต่อเข้า Sound Card ของเครื่อง PC แล้วนำมาประมวลผล และส่งผลที่ได้ไปสั่งงานอุปกรณ์ไฟฟ้านั้น

4.เสียงที่ใช้สั่งไม่ขึ้นกับผู้พูด เสียงของคำที่มีความหมายเดียวกันจากผู้พูดต่างบุคคลก็จะได้ผลการรับรู้เหมือนกัน โดยขั้นต้นจะมีการเก็บตัวอย่างเสียงของคำนั้นๆ ที่มากและหลากหลายพอไว้ก่อน เพื่อผลการรับรู้ที่ถูกต้อง

5. เป็นการวิเคราะห์เสียงพูดแบบต่อเนื่อง ช่วงวิเคราะห์สัญญาณ โดยที่ความยาวของคำสั่งนั้นเราได้ทำการออกแบบไว้ คือคำสั่งควรจะสั้นเพื่อในการประมวลผลจะรวดเร็วขึ้น ตัวอย่างคำสั่งที่ได้ออกแบบไว้ ได้แก่ "เปิด-หนึ่ง" ก็หมายความว่าสั่งให้เปิดอุปกรณ์ตัวที่ 1 , "ปิด-แปด" หมายความว่า ให้ทำการปิดอุปกรณ์ตัวที่ 8 โดยที่ในโครงงานนี้ได้มีการออกแบบสวิตช์ไว้ 10 ตัว คือจะมีสวิตช์ เบอร์ 0-9

ในการศึกษาเรื่องการสั่งงานคอมพิวเตอร์ด้วยเสียง ส่วนของขั้นตอนต่างๆ ที่ใช้ในการวิคราะห์เสียงและการควบคุมอุปกรณ์ต่างๆ นั้น จะต้องมีความรู้พื้นฐานพอสมควรในเรื่องต่างๆ ดังต่อไปนี้

- ลักษณะต่างๆ ของเสียงพูดมนุษย์ เช่น ความถี่ , ความดัง , ความแตกต่างของเสียงชายและหญิง , เสียงที่เปล่งออกมาทางปากหรือออกมาทางจมูก , เสียงรบกวนที่เกิดขึ้น และกราฟต่างๆ ของเสียง เป็นต้น
- พื้นความรู้ทางด้านคณิตศาสตร์ที่สำคัญ เช่น ฟังก์ชันการถ่ายโอน , การประมาณเชิงเส้น , ผลรวมกำลังสองของความคลาดเคลื่อน , เวกเตอร์หลายมิติ และเมตริกซ์ เป็นต้น
- พื้นความรู้ทางด้านสถิติ เช่น ความน่าจะเป็น , ความน่าจะเป็นแบบมีเงื่อนไข
- ลักษณะการทำงานของ Sound Card ว่าเก็บเสียงมาได้อย่างไร มีการแปลงสัญญาณจากสัญญาณอนาล็อกเป็นดิจิตอลอย่างไร สัญญาณเสียงที่ได้มีการจัดเก็บและนำมาใช้งานได้ อย่างไร
- หลักการเชื่อมต่อบอร์ดโปรโตไทป์กับเครื่อง PC
- หลักการของชิป 8255
- หลักการเขียนโปรแกรมภาษา C หรือ C++

ขั้นตอนในการศึกษาและดำเนินงานมีดังนี้

1. ศึกษารูปแบบการวิเคราะห์เสียงพูดแบบไม่ขึ้นกับผู้พูด และไม่ต่อเนื่อง
2. ศึกษาขั้นตอนการประมาณเชิงเส้นของสัญญาณเสียงพูด โดยวิธีออโตคอร์รีเลชัน เพื่อ นำมาใช้ในการวิเคราะห์เสียงพูดมนุษย์ สาเหตุที่ใช้การประมาณเชิงเส้นเพราะสามารถนำมาประยุกต์ใช้กับการสื่อสารทางเสียงได้หลายด้านและการจดจำเสียงพูด และเป็นวิธีการบีบอัดข้อมูลได้เป็นอย่างดีโดยมีการสูญเสียน้อยมาก

3. วิเคราะห์การหาค่าพารามิเตอร์ LPC และ เซปสตรัม (Cepstrum) ของเสียง

4. สร้างแบบอ้างอิง (Codebook) โดยใช้การจัดระดับเวกเตอร์ (Vector Quantization) ของพารามิเตอร์ที่ได้จากการประมาณเชิงเส้น

5. สร้างแบบจำลองของ มาร์คอฟ (Hidden Markov Models) เพื่อใช้ในการรู้จำเสียงพูด

6. เขียนโปรแกรมตามรูปแบบการวิเคราะห์เสียงที่ได้ศึกษามา

7. ศึกษาและเขียนโปรแกรมติดต่อกับ Sound Card เพื่อนำสัญญาณเสียงเข้ามาวิเคราะห์

บทที่ 2
ทฤษฎีและหลักการ

2.1 ความรู้เบื้องต้นเกี่ยวกับโทรศัพท์

เครื่องโทรศัพท์ เป็นอุปกรณ์ปลายทางซึ่งทำหน้าที่รับและส่งสัญญาณเสียงพูดระหว่างผู้เช่า โดยการทำงานคือ ⎓ ทางด้านส่งจะแปลงพลังงานเสียงให้เป็นพลังงานไฟฟ้าแล้วส่งไปตามคู่สายโทรศัพท์ ทางด้านรับก็จะได้รับสัญญาณไฟฟ้า แล้วก็จะทำการเปลี่ยนพลังงานไฟฟ้านั้นให้เป็นพลังงานเสียง โดยจะทำการรับส่งอย่างนี้ไปเรื่อยๆ โดยที่ในช่วงปกติ(สายว่าง)จะมีแหล่งจ่ายไฟร่วม โดยทางชุมสายโทรศัพท์จะจ่ายไฟกระแสตรง(DC)ขนาด 48โวลต์ให้แต่ละลูปของผู้ใช้ โดยคู่สายโทรศัพท์นั้นจะมีตัวนำ 2 เส้น มีชื่อว่า ทิป (Tip) และ ริง (Ring) โดยที่ทิปจะต่อลงกราวน์ และริงจะต่อกับไฟ -48 โวลต์ เมื่อมีการเรียกเลขหมายนี้เข้ามา ทางชุมสายจะจ่ายสัญญาณกระดิ่งไปยังผู้ถูกเรียกคือ จากสภาวะปกติจะมีไฟตกคร่อมคู่สาย 48 โวลต์ จะมีสัญญาณกระดิ่งซึ่งเป็นแรงดันกระแสสลับประมาณ 100 Vp เป็นเวลา 2 วินาที และหยุด 4 วินาที เป็นจังหวะให้กระดิ่งโทรศัพท์ทำงาน เมื่อผู้ใช้ปลายทาง (ผู้ถูกเรียก) ยกหูโทรศัพท์ขึ้น สวิตช์ภายในโทรศัพท์จะทำการต่อคู่สายเข้ากับวงจรภายในที่มีความต้านทานกระแสตรงต่ำ จะเกิดการครบวงจรและทำให้แรงดันไฟ 48 โวลต์ตกลงเหลือ 5 ถึง 10 โวลต์ ทั้งนี้ขึ้นอยู่กับวงจรภายในของเครื่องโทรศัพท์นั้นๆ การที่แรงดันไฟฟ้าตกลงนั้นจะเป็นการทำให้ชุมสายได้รู้ว่า ผู้ใช้ปลายทางได้ยกหูแล้ว ดังนั้นจึงทำการสวิตช์เชื่อมต่อคู่สายทั้งสองเข้าด้วยกัน ทำให้สามารถติดต่อกันได้

2.1.1 ส่วนประกอบของโทรศัพท์

ในเครื่องโทรศัพท์ตามบ้านเรามีส่วนประกอบที่สำคัญดังต่อไปนี้

2.1.1.1 วงจรเสียงพูด (Speech Network) ทำหน้าที่แปลงสัญญาณจากระบบทูไวร์ (Two Wire) ในคู่สายโทรศัพท์ให้เป็นระบบโฟร์ไวร์ (Four Wire) ขยายสัญญาณเข้าไปในหูฟัง และไมโครโฟน โดยตัวที่จะทำการแปลงสัญญาณนั้นเราจะใช้ ทรานฟอร์มเมอร์ (Transformer) และยังสามารถควบคุมระดับไซด์โทน (Side Tone) ได้อีกด้วย

2.1.1.2 ไดอัลโทน (Dial Tone) ทำการกำเนิดสัญญาณหมายเลขโทรศัพท์ เพื่อส่งไปยังชุมสายโทรศัพท์

2.1.1.3 วงจรส่งเสียงพูด (Transmitter) เป็นส่วนที่รับสัญญาณจากผู้พูดต้นทางไปยังผู้ใช้ปลายทาง โดยจะใช้ไมโครโฟนแปลงสัญญาณเสียงให้เป็นสัญญาณไฟฟ้า

2.1.1.4 วงจรรับเสียงพูด (Receiver) เป็นตัวแปลงสัญญาณไฟฟ้าจากต้นทางให้กลายเป็นสัญญาณเสียง โดยจะใช้ลำโพงหรือไดอะแฟรม

2.1.1.5 วงจรกระดิ่ง (Ringer) ทำหน้าที่ตรวจจับสัญญาณจากชุมสายโทรศัพท์ เมื่อมีสัญญาณกระดิ่งเข้ามาก็จะทำการแปลงสัญญาณจากชุมสายให้กลายเป็นเสียงกระดิ่ง

2.1.1.6 ฮุกสวิตช์ (Hook Switch) ทำหน้าที่ตัดต่อเครื่องโทรศัพท์เข้ากับคู่สาย มี 2 สภาวะ คือ สภาวะการยกหู(Off Hook) และ สภาวะการวางหู (On Hook)

2.1.1.7 วงจรบริดจ์ (Polar Guard Bridge) ทำหน้าที่ผ่านกระแสไฟตรงจากคู่สาย ไปเลี้ยงวงจร และป้องกันการกลับขั้วของกระแสไฟ นอกจากนี้ยังทำหน้าที่ผ่านสัญญาณไฟฟ้าทั้งทางด้านบวก และลบไปในวงจร รวมทั้งเป็นส่วนสร้างกราวน์ให้แก่เครื่องโทรศัพท์

2.1.2 สัญญาณที่ส่งมาจากชุมสาย

สัญญาณที่ส่งมาจากชุมสายโทรศัพท์จะมี 6 ชนิด ดังนี้

2.1.2.1 สัญญาณให้หมุน (Dial Tone) เป็นสัญญาณที่บอกให้ทราบว่าขณะนี้อุปกรณ์ชุมสายว่าง พร้อมที่จะรับสัญญาณจากการหมุนหมายเลขจากผู้เรียกให้ผู้เรียกทำการส่งหมายเลขได้ สัญญาณนี้เป็นสัญญาณต่อเนื่อง มีความถี่ 425 เฮิร์ต มอดูเลตด้วยความถี่ 50 เฮิร์ต ผู้เช่าจะได้ยินเสียงสัญญาณโทนนี้ก็ต่อเมื่อ ทำการยกหูโทรศัพท์เพื่อทำการเรียก

2.1.2.2 สัญญาณไม่ว่าง (Busy Tone) เป็นสัญญาณที่บอกให้ทราบว่าอุปกรณ์ไม่ว่าง ตัวอย่าง เช่น ถ้าผู้เช่ายกหูโทรศัพท์แล้วได้ยินเสียงนี้ แทนที่จะเป็นสัญญาณให้หมุน แสดงว่าอุปกรณ์ชุมสายไม่ว่าง หรือถ้าได้ยินเสียงนี้หลังจากหมุนเลขหมายไปแล้ว แสดงว่าผู้เช่าฝ่ายถูกเรียกกำลังใช้โทรศัพท์อยู่นั่นก็คือสายไม่ว่าง หรืออุปกรณ์ที่ใช้สำหรับต่อออกชุมสายอื่นไม่ว่าง ในกรณีที่เป็นการต่อระหว่างชุมสาย สัญญาณที่ส่งนี้เป็นสัญญาณต่อเนื่องในรูปของคลื่นซายน์ ส่งเป็นช่วงๆ ละ 0.5 วินาที หยุด 0.5 วินาที สลับกันไป โดยมีความถี่ของสัญญาณ 425 เฮิร์ต

2.1.2.3 เสียงสัญญาณเรียก (Ringing Tone) เป็นเสียงสัญญาณที่ผู้เรียกได้ยิน หลังจากหมุนหมายเลขครบแล้ว เพื่อบอกให้ทราบว่าได้ทำการต่อสำเร็จ และขณะนี้ชุมสายได้ส่งสัญญาณเรียก (Ringing Signal) ไปยังผู้ถูกเรียกแล้ว สัญญาณนี้เป็นสัญญาณรูปซายน์ ส่งเป็นช่วงๆ โดยส่ง 1 วินาที หยุด 4 วินาที มีความถี่ 425 เฮิร์ต

2.1.2.4 สัญญาณเรียก (Ringing Signal) เป็นสัญญาณที่ส่งไปยังผู้เช่าซึ่งเป็นฝ่ายถูกเรียก ซึ่งจะได้ยินเสียงกระดิ่งหรือเสียงโทนดังขึ้น สัญญาณที่ใช้เป็นสัญญาณรูปซายน์ ความถี่ 25 เฮิร์ต ค่าแรงดันประมาณ 100-110 Vp ส่งเป็นสัญญาณเดียวกับ เสียงสัญญาณเรียก (Ringing Tone)

2.1.2.5 สัญญาณช่องสัญญาณไม่ว่าง (Unobtainable Tone) เป็นสัญญาณบอกให้ทราบว่าเลขหมายที่เรียกไปเป็นเลขหมายว่าง ยังไม่มีการติดตั้ง หรือถูกยกเลิกไป

2.1.2.6 เสียงตอบจากเครื่องบันทึกเสียง (Announcement Machine) เป็นเสียงตอบจากเครื่องบันทึก ใช้ในการบอกข่าวสารขององค์การโทรศัพท์ให้แก่ผู้ที่ต่อเข้ามา เช่น "เลขหมายนี้ได้ถูกยกเลิกไปแล้ว หากอยากทราบหมายเลขที่เปลี่ยนแปลง กรุณาติดต่อหมายเลข 505-1000" เป็นต้น

2.1.3 สัญญาณที่รับส่งระหว่างผู้เช่ากับชุมสาย

สัญญาณที่ใช้ในการรับส่งระหว่างผู้เช่าและชุมสายจะมี 3 ชนิดดังต่อไปนี้

2.1.3.1 สัญญาณวางหู (On hook) หมายถึง สภาพที่ผู้เช่าวางหู หรือสภาพว่าง (Idle) สายมีสถานะภาพเป็นวงจรเปิดมีความต้านทานสูง (Open Loop High Impedance)

2.1.3.2 สัญญาณช่วงยกหู (Off hook) หมายถึง สภาพที่ผู้เช่ายกหูโทรศัพท์ขึ้นมา สายมีสภาพเป็นวงจรปิดมีความต้านทานต่ำ (Close Loop Low Impedance)

2.1.3.3 สัญญาณหมุน (Dialing Tone) เกิดขึ้นเมื่อผู้เช่าทำการหมุนหมายเลขเครื่องแบบที่ใช้ระบบพัลส์ จะทำการให้อิมพีแดนซ์ต่ำ และสูงสลับกันไป ส่วนโทรศัพท์ที่ใช้ระบบโทนก็จะทำการส่งสัญญาณ DTMF ออกไป

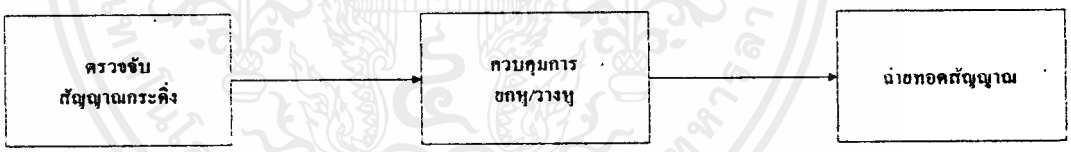
2.2 เครื่องตอบรับโทรศัพท์อัตโนมัติ

เครื่องตอบรับโทรศัพท์อัตโนมัติจะมีการทำงาน 3 ขั้นตอน ได้แก่

2.2.1 ส่วนตรวจจับสัญญาณกระดิ่ง

2.2.2 ส่วนของการควบคุมการยกหู/วางหู

2.2.3 ส่วนของการถ่ายทอดสัญญาณ



รูปที่ 2.1 Block Diagram ในส่วนของวงจรตอบรับโทรศัพท์อัตโนมัติ

เมื่อมีสัญญาณเรียกเข้ามา จะมีสัญญาณกระดิ่งส่งมาจากชุมสายโทรศัพท์ โดยเมื่อสัญญาณกระดิ่งเข้ามา ส่วนตรวจจับสัญญาณกระดิ่งก็จะสร้างพัลส์สัญญาณไปกระตุ้นส่วนควบคุมการยกหู/วางหู ซึ่งส่วนนี้จะทำงานที่ขอบขาขึ้น มีหน้าที่ยกหูโทรศัพท์เวลามีสัญญาณกระดิ่งเข้ามา การทำงานในส่วนของการยกหู/วางหูนี้ อาศัยความรู้พื้นฐานว่า เมื่อคู่สายโทรศัพท์อยู่ในสภาวะสายว่าง ชุมสายจะจ่ายแรงดันไฟ 48 โวลต์ ซึ่งเป็นแรงดันที่ให้กับคู่สายโทรศัพท์นั้น เมื่อมีผู้เรียกเลขหมายนั้น ชุมสายก็จะทำการจ่ายแรงดันไฟฟ้ากระแสสลับซึ่งมีแรงดันประมาณ 100 Vpจ่ายเป็นเวลา 2 วินาที และหยุดเป็นเวลา 4 วินาที เป็นจังหวะ และจากหลักการที่ว่า เมื่อมีการยกหูโทรศัพท์ สวิตช์ภายในตัวโทรศัพท์จะทำการต่อคู่สายเข้ากับวงจรภายในที่มีความต้านทานทางกระแสตรงต่ำ ทำให้

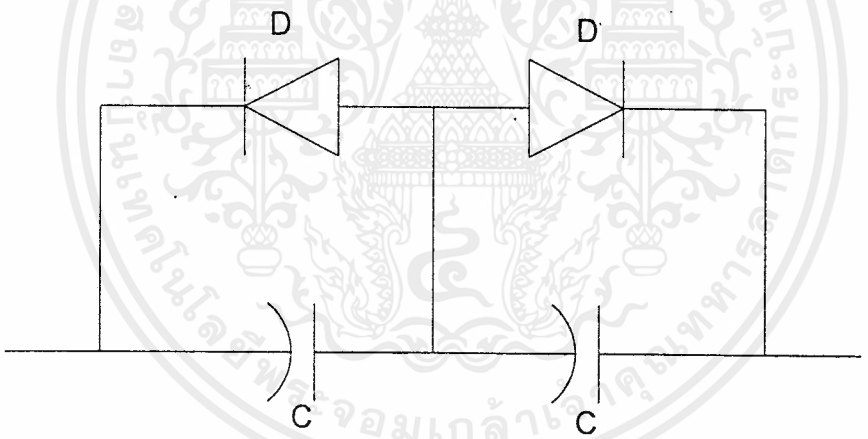
ครบวงจร คือ แรงดัน 48 โวลต์ จากคู่สาย ก็จะตกลงมาเหลือประมาณ 5-10 โวลต์ (ค่าแรงดันจะมากหรือน้อยขึ้นอยู่กับชนิดของโทรศัพท์) เมื่อเกิดสภาวะความต้านทานต่ำขึ้นระหว่างคู่สาย ก็จะเสมือนการยกหูโทรศัพท์ แล้วชุมสายก็จะต่อส่วนถ่ายทอดสัญญาณเข้ากับคู่สายโทรศัพท์ ดังนั้นเมื่อส่วนตรวจจับสัญญาณกระดิ่งมีพัลส์สัญญาณเข้ามา ส่วนควบคุมการยกหู/วางหูจะใช้หลักการดังกล่าวมายกหูโทรศัพท์ ต่อจากนั้นเราสามารถถ่ายทอดสัญญาณเสียงพูดของเราเข้าไปในคอมพิวเตอร์ได้โดยการเชื่อมต่อทางซาวน์การ์ด

2.2.1 ส่วนของวงจรตรวจจับสัญญาณกระดิ่ง

ในส่วนของวงจรตรวจจับสัญญาณกระดิ่งจากหลักการจะมีวงจรอยู่ 3 วงจรด้วยกัน ได้แก่

2.2.1.1 วงจรถ่ายทอดสัญญาณกระดิ่ง

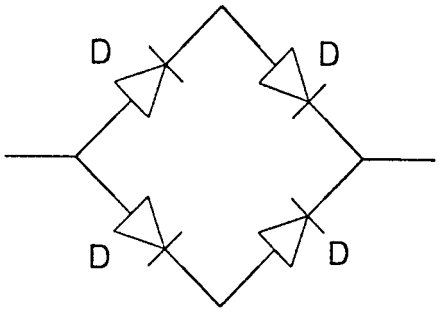
วงจรถ่ายทอดสัญญาณกระดิ่งจะทำหน้าที่ในการให้ไฟกระแสสลับผ่านเท่านั้น โดยไฟกระแสตรงจะไม่สามารถผ่านวงจรนี้ไปได้ เพราะฉะนั้นจะมีแค่สัญญาณเสียงกระดิ่งที่มาจากชุมสายเท่านั้นที่จะสามารถไหลผ่านวงจรนี้ไปได้ โดยไฟกระแสสลับขนาด 100 Vp จะผ่านวงจรนี้ไปยังวงจรอื่นต่อไป



รูปที่ 2.2 วงจรถ่ายทอดสัญญาณกระดิ่ง

2.2.1.2 วงจรบริดจ์เรคติไฟเออร์ (Bridge Rectifier Circuit)

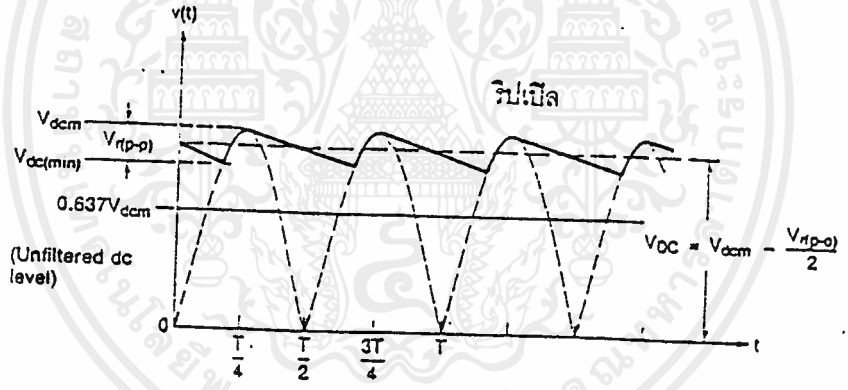
เนื่องจากการใช้งานในวงจร เราจำเป็นที่จะต้องใช้ไฟกระแสตรงเท่านั้น แต่จากวงจรถ่ายทอดสัญญาณกระดิ่งเราจะได้ไฟกระแสสลับผ่านมาจากวงจรถ่ายทอดสัญญาณกระดิ่ง ดังนั้นเราจึงจำเป็นที่จะต้องนำไปผ่านวงจรบริดเรคติไฟเออร์ เพื่อที่จะทำการแปลงไฟกระแสสลับให้เป็นไฟกระแสตรง สำหรับนำไปใช้งานต่อไป



รูปที่ 2.3 วงจรบริดจ์เรคติไฟเออร์

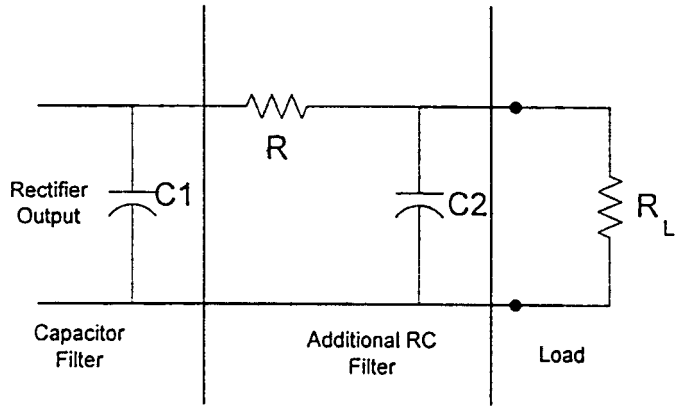
2.2.1.3 วงจรกรองสัญญาณ (Filter)

เนื่องสัญญาณที่ได้จากวงจรเรคติไฟเออร์ไม่สามารถนำไปใช้เป็นสัญญาณไฟตรงได้ทันที เนื่องจากสัญญาณที่ออกจากวงจรเรคติไฟเออร์จะมีการกระเพื่อมของสัญญาณ เรียกว่า ริปเปิ้ล (Ripple) แสดงดังรูป



รูปที่ 2.4 แสดงการกระเพื่อมของสัญญาณ

ริปเปิ้ลเป็นค่ากระเพื่อมของแรงดันไฟตรงเอาท์พุท ที่กระเพื่อมไปจากค่าแรงดันไฟเฉลี่ยหรือแรงดันไฟตรงของสัญญาณ แรงดันริปเปิ้ลจะมีผลเสมือนการจ่ายกระแสไฟให้โหลดเป็นรูปพัลส์ ดังนั้นการเกิดเช่นนี้จะทำให้ค่ากระแสและแรงดันที่จ่ายให้กับวงจรมีค่าไม่คงที่ ดังนั้นเราจึงจำเป็นที่จะต้องทำวงจรกรองสัญญาณมาต่ออีกที ซึ่งผลที่ได้คือ จะได้ค่าแรงดันที่เรียบมากขึ้นและกระแสจะมีค่าคงที่ โดยวงจรกรองสัญญาณเราจะใช้วงจรกรองสัญญาณแบบ RC โดยตัวต้านทาน (R) จะทำหน้าที่รักษากระแสที่ถูกเรคติไฟร์ให้มีค่าคงที่และลดองค์ประกอบของไฟกระแสสลับ ส่วนตัวเก็บประจุ(C) จะช่วยลดค่าริปเปิ้ลอีกครั้งและทำให้อิมพีแดนซ์ของแหล่งจ่ายไฟลดลง



รูปที่ 2.5 วงจรกรองสัญญาณ RC

2.2.2 ส่วนของวงจรยกหู/วางหู

โดยในส่วนของวงจรการยกหูและวางหูเราจะใช้รีเลย์ (Relay) เป็นตัวที่ทำการยกหู/วางหู โดยเราจะนำไปต่อกับ ดีฟลิปฟลอป (D-Flip Flop) โดยเราจะมีวงจรที่เพิ่มเข้ามาในการที่จะนำไปประยุกต์ใช้กับ ดีฟลิปฟลอป โดยในส่วนของการทำงานของการยกหู เราจะใช้ทรานซิสเตอร์เป็นตัวทำให้ยกหูตอบรับอัตโนมัติ ส่วนการทำงานของการวางหู เราจะใช้วงจร DTMF และทรานซิสเตอร์เข้ามาช่วยในการวางหู โดยการวางหูจะสามารถทำได้โดยกดปุ่ม # เครื่องตอบรับก็จะทำการวางหูให้เอง

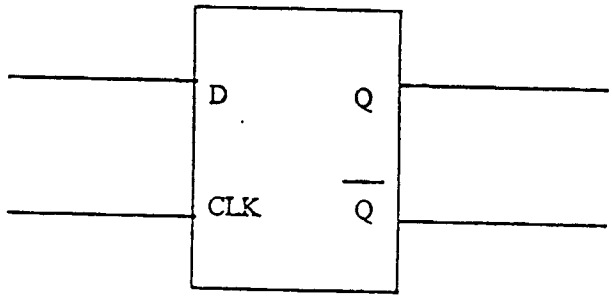
2.2.2.1 หลักการของ ดี ฟลิปฟลอป

อินพุท(D) และสัญญาณนาฬิกาทำงานขอบขาขึ้น (Clock Operation)		
D	Q	Q
0	0	1
1	1	0

ตารางที่ 2.1 แสดงค่าความจริงของดีฟลิปฟลอป

ดี ฟลิปฟลอป หรือ ฟลิปฟลอปข้อมูล (Data - Flip Flop) เป็นวงจรที่ใช้กันมาในการพักข้อมูลชั่วคราวในวงจรนับความถี่ หรือโวลต์มิเตอร์แบบดิจิตอล การทำงานของดีฟลิปฟลอปจะให้สัญญาณเอาท์พุทที่มีสถานะเปลี่ยนตามอินพุทที่ขาดี (D) เมื่อมีการเปลี่ยนสถานะของสัญญาณนาฬิกา (Clock) จากต่ำไปสูง หรือกล่าวคือจะมีการทำงานที่ขอบขาขึ้น

เอาท์พุท (Q) จะมีสถานะต่ำถ้าอินพุท (D) มีสถานะต่ำ และมีการเปลี่ยนแปลงสถานะของสัญญาณนาฬิกาจากต่ำไปสูง และอีกกรณีคือ เอาท์พุท (Q) จะมีสถานะสูงถ้าอินพุท (D) มีสถานะสูง และมีการเปลี่ยนสถานะของสัญญาณนาฬิกาจากต่ำไปสูง



รูปที่ 2.6 สัญลักษณ์ทางตรรกะของดีฟลิปฟลอป

2.2.2.2 หลักการของ DTMF (MT8870)

โครงสร้างภายในของ MT8870 ประกอบไปด้วยวงจรกรองความถี่และวงจรถอดรหัสฟังก์ชันทางดิจิตอล เป็นไอซีที่สร้างโดยใช้เทคโนโลยี ISO-CMOS ในส่วนของวงจรกรองความถี่ใช้เทคนิคของสวิตช์คาปาซิเตอร์ฟิลเตอร์ สำหรับกรองความถี่สูงและต่ำ ส่วนวงจรถอดรหัสใช้เทคนิคการนับทางดิจิตอลเพื่อตรวจจับและถอดรหัสทั้ง 16 ความถี่ออกเป็นเลขฐานสองขนาด 4 บิท และเช็คช่วงเวลาที่สัญญาณเข้ามา ส่วนภาคอินพุทเป็นออปแอมป์ ซึ่งสามารถปรับอัตราขยายได้โดยต่ออุปกรณ์ภายนอกเอาท์พุทเป็นวงจรแลตช์ 3 สถานะ

ฟังก์ชันการทำงานภายใน MT8870

ภายใน MT8870 ประกอบด้วยส่วนสำคัญ 5 ส่วน คือ

1. ภาคกรองความถี่ (Filter Section)

ในส่วนนี้จะแยกสัญญาณ DTMF ที่เข้ามาออกเป็น 2 กลุ่มความถี่ คือช่วงความถี่สูงและช่วงความถี่ต่ำโดยใช้วงจรกรองความถี่อันดับ 6 ชนิดสวิตช์คาปาซิเตอร์(six-order switched capacitor band pass filter) ซึ่งความถี่ที่แยกได้มี 2 ช่วง คือช่วงความถี่สูงและช่วงความถี่ต่ำ

2. ภาคถอดรหัส (Decoder Section)

ความถี่ DTMF ที่ถูกกรองเรียบร้อยแล้วจะผ่านเข้าวงจรถอดรหัสความถี่ออกเป็นตัวเลขโดยใช้เทคนิคการนับแบบดิจิตอล และมีการตรวจสอบความถี่ที่เข้ามาว่าเป็นความถี่มาตรฐาน DTMF หรือไม่ เพื่อป้องกันความถี่อื่นเข้ามาผสมเมื่อตรวจสอบว่าความถี่นั้นถูกต้อง สัญญาณที่ขา Est (early steering) ก็จะแอคทีฟสำหรับค่าที่ถอดรหัสได้จากความถี่ต่างๆนั้น

3.ภาคตรวจสอบสัญญาณ (Steering Circuit)

ก่อนที่จะมีการถอดรหัสความถี่ออกไปที่เอาท์พุท จะมีตรวจสอบช่วงความถี่ที่เข้ามาว่ามีระยะเวลาตามที่กำหนดหรือไม่ โดยสังเกตจากระยะเวลาการกดปุ่มโทรศัพท์ ซึ่งต้องกดปุ่มให้มีความถี่ออกมาเป็นช่วงเวลาพอสมควรมิฉะนั้นวงจรส่วนนี้จะไม่รับ โดยถือว่าสัญญาณนั้นไม่ถูก

ต้อง ส่วนช่วงเวลายาวเท่าใดสามารถตั้งได้โดยใช้ RC ต่อภายนอก สัญญาณที่ขา Est จะเป็น "High" นานใกล้เคียงกับระยะเวลาที่มีความถี่ DTMF เข้ามา เมื่อขา Est เป็น "High" วงจรถอดรหัสจึงจะถอดรหัสออกเป็นตัวเลขขนาด 4 บิท

สำหรับคำว่าการ์ดไทม์ (Guard time) นั้นหมายถึงช่วงคาบเวลาของความถี่ที่เข้ามา ซึ่งจะต้องนานเท่ากับหรือมากกว่าช่วงเวลาที่เราตั้งไว้ จึงจะได้รับการยอมรับว่าสัญญาณความถี่นั้นถูกต้องหรือพูดได้ว่า เวลาที่เราตั้งไว้โดย RC ก็คือการ์ดไทม์นั้นเอง เมื่อสัญญาณความถี่เข้ามานานเท่ากับหรือมากกว่าเวลาที่ตั้งไว้จึงจะสามารถแปลงเป็นตัวเลขได้ ถ้าสัญญาณความถี่เข้ามาสั้นกว่าก็จะไม่มีการถอดรหัสเป็นตัวเลขออกไป

4.ภาคขยายสัญญาณความแตกต่าง (Differential Input)

วงจรส่วนอินพุทของ MT8870 เป็นภาคขยายออปแอมป์ที่สามารถปรับอัตราขยายโดยต่อวงจรภายนอกเพิ่มเข้าไป

5.ภาคกำเนิดความถี่ (Oscillator)

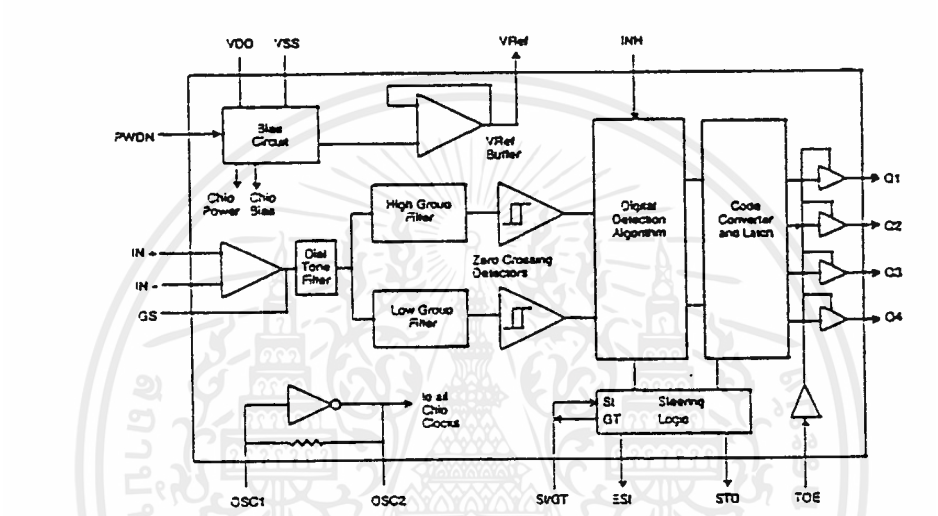
ในภาคนี้ภายในไอซีจะมีวงจรเวลาอยู่ภายใน เพียงแต่ต่อคริสตอลขนาด 3.57MHz ก็สามารถนำไปใช้งานได้ทันที

f1	f2	ตัวเลข	Std	Q4	Q3	Q2	Q1
697	1209	1	H	0	0	0	1
697	1336	2	H	0	0	1	0
697	1477	3	H	0	0	1	1
770	1209	4	H	0	1	0	0
770	1336	5	H	0	1	0	1
770	1477	6	H	0	1	1	0
852	1209	7	H	0	1	1	1
852	1336	8	H	1	0	0	0
852	1477	9	H	1	0	0	1
941	1336	0	H	1	0	1	0
941	1209	*	H	1	0	1	1
941	1477	#	H	1	1	0	0

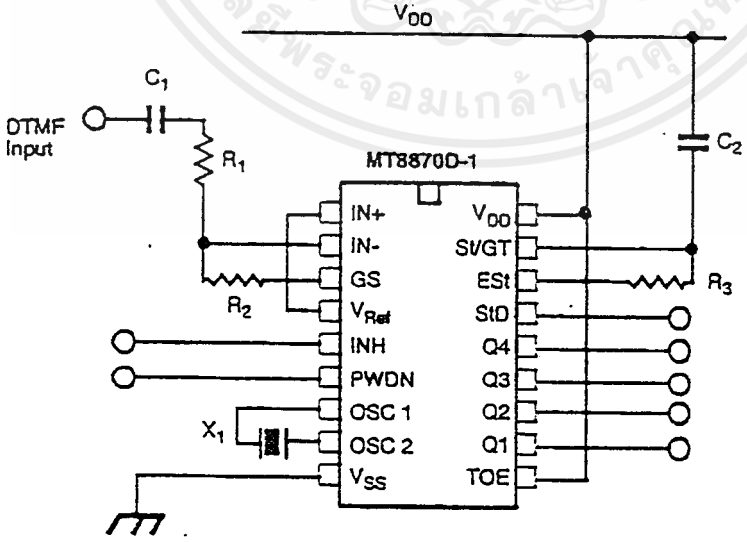
ตารางที่ 2.2 แสดงค่าที่ถอดรหัสได้จากความถี่ต่างๆ

697 Hz	1	2	3
770 Hz	4	5	6
852 Hz	7	8	9
941 Hz	*	0	#
	1209 Hz	1336 Hz	1477 Hz

รูปที่ 2.7 แสดงค่าความถี่ของโทรศัพท์ชนิดกดปุ่ม



รูปที่ 2.8 แสดงโครงสร้างภายในของ MT8870



NOTES:
$R_1 = 102K\Omega \pm 1\%$
$R_2 = 71.5K\Omega \pm 1\%$
$R_3 = 390K\Omega \pm 1\ \%$
$C_1, C_2 = 100\ nF \pm 5\%$
$X_1 = 3.579545\ MHz \pm 0.1\%$
$V_{DD} = 5.0V \pm 5\%$

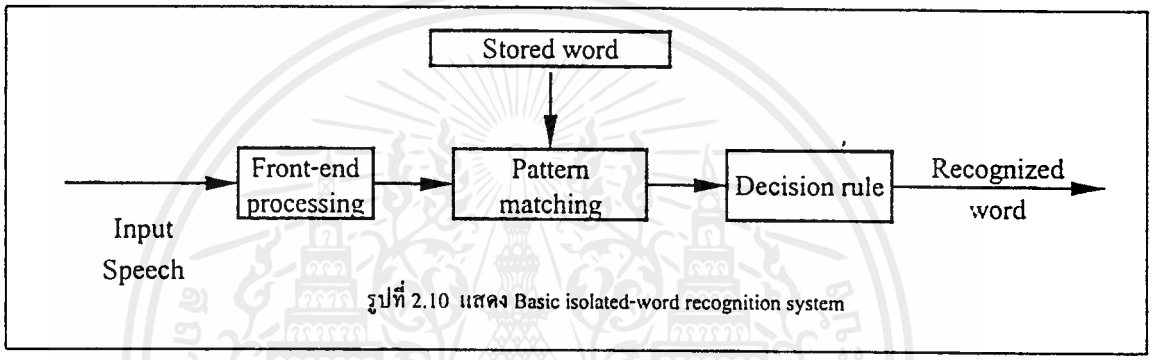
รูปที่ 2.9 แสดงวงจรใช้งานเบื้องต้นของ MT8870

2.2.3 ส่วนของการถ่ายทอดสัญญาณ

เราจะเชื่อมต่อ และถ่ายทอดสัญญาณเสียงให้ผ่านเข้าไปยังคอมพิวเตอร์ โดยจะผ่านทางซาวน์การ์ดของคอมพิวเตอร์ โดยในส่วนนี้เมื่อเราทำการยกหูอัตโนมัติเรียบร้อยแล้ว ก็จะสามารถถ่ายทอดสัญญาณได้อย่างต่อเนื่อง โดยจะส่งสัญญาณเสียงจากคู่สายโทรศัพท์ผ่านทางทรานฟอร์มเมอร์ (Transformer) โดยทั้งสัญญาณเข้าและออกเราจะใช้ทรานฟอร์มเมอร์ตัวเดียวกัน

2.3 หลักการวิเคราะห์เสียง

จากการศึกษารูปแบบการรู้จำเสียงพูดแบบคำเดี่ยว (Isolated word recognition) มีรูปแบบและหลักการพื้นฐานดังรูปที่ 2.10



รูปที่ 2.10 แสดง Basic isolated-word recognition system

สัญญาณที่เข้าโมเดลเป็นคลื่นเสียง ส่วนผลลัพธ์เป็นเสียงวิเคราะห์ที่ได้จากสัญญาณเข้า โมเดลนี้มีการใช้กันอย่างแพร่หลาย แบ่งเป็นส่วนต่าง ๆ ดังนี้

- Front-end processing เป็นขั้นตอนการวิเคราะห์เสียงที่ได้รับมาให้อยู่ในรูปแบบที่สามารถนำไปวิเคราะห์ได้
- Pattern matching เป็นขั้นการสร้างโมเดลของเสียง
- Decision rule เป็นขั้นตอนการตัดสินใจในการเลือกโมเดลเสียงที่ใกล้เคียงกับเสียงทดสอบมากที่สุด

2.3.1 Front-end processing

Front-end processing เป็นขั้นตอนที่ทำการแปลงข้อมูลที่มีอยู่มากมายเป็นส่วนเล็กๆ ที่สามารถแสดงคุณสมบัติของคลื่นเสียงนั้นๆ โดยผ่านขั้นตอน ดังนี้

2.3.1.1 การประมาณเชิงเส้น (Linear predictive coding : LPC)

2.3.1.2 การจัดระดับเวคเตอร์ (Vector Quantization : VQ)

2.3.1.1 การประมาณเชิงเส้น (Linear predictive coding)

เป็นการวิเคราะห์เพื่อหาค่าพารามิเตอร์ที่เหมาะสมเป็นข้อมูลในการวิเคราะห์เสียง โดยแบ่งสัญญาณเสียงพูดที่จะวิเคราะห์ออกเป็นส่วนๆ แต่ละส่วนใช้ระยะเวลาช่วงสั้นๆ ประมาณ 15-20 มิลลิวินาที ซึ่งช่วงนี้สัญญาณเสียงพูดจะมีการเปลี่ยนคุณลักษณะอย่างช้าๆ จนอาจถือว่าระบบกำเนิดเสียงมีคุณลักษณะที่คงที่ (stationary)

ก่อนที่จะนำข้อมูลเสียงพูดมาทำการประมาณเชิงเส้นนั้นจะต้องทำการปรับแต่งข้อมูลให้เหมาะสม โดยการตัดส่วนหัวและส่วนท้ายคำพูดที่เป็นส่วนเกินออก เนื่องจากเป็นส่วนของสัญญาณรบกวน จากนั้นจึงนำข้อมูลการผ่านขั้นตอนส่วนต่างๆ ดังนี้

2.3.1.1.1 การพรีเอมฟาซิส (pre-emphasis)

2.3.1.1.2 การแบ่งช่วงสัญญาณ (frame blocking)

2.3.1.1.3 การวินโดว์ (windowing)

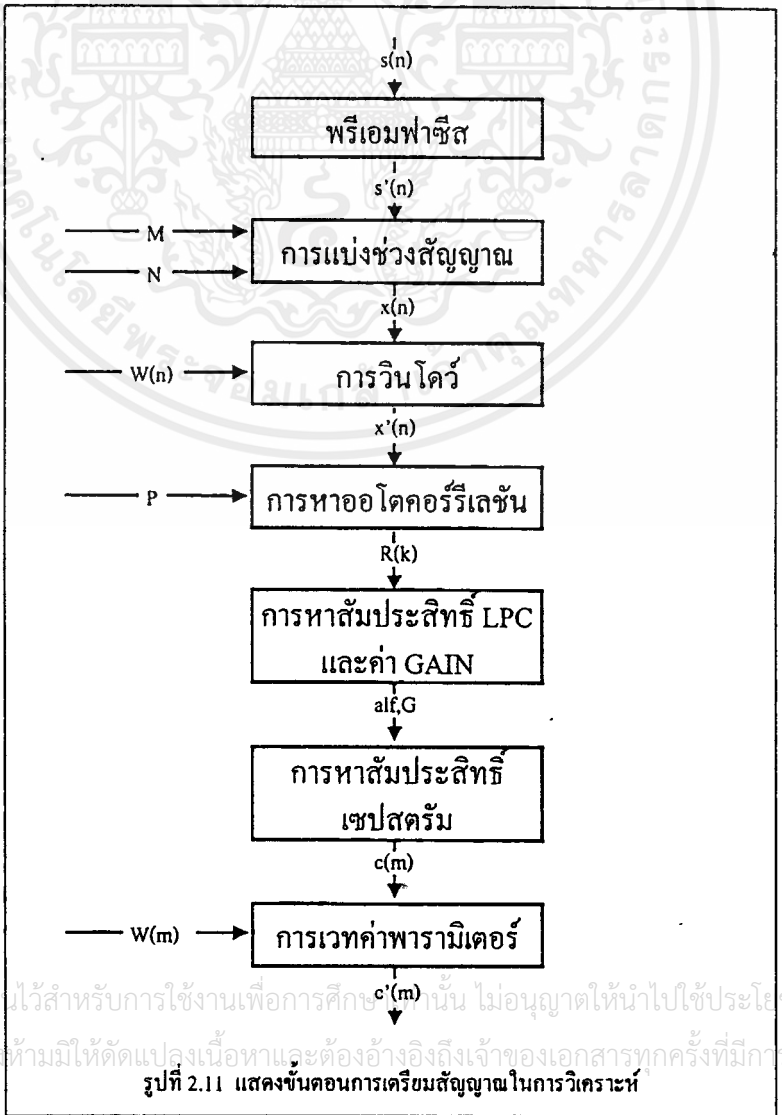
2.3.1.1.4 การวิเคราะห์ออโตคอร์รีเลชัน (autocorrelation analysis)

2.3.1.1.5 การหาอัตราการขยาย G

2.3.1.1.6 การสัมประสิทธิ์เซปสตรัม (cepstrum)

2.3.1.1.7 การเวทค่าพารามิเตอร์ (parameter weighting)

ดังแสดงในรูปที่ 2.11



รูปที่ 2.11 แสดงขั้นตอนการเตรียมสัญญาณในการวิเคราะห์

การประมาณเชิงเส้น จะทำการประมาณค่าสัญญาณจากผลรวมเชิงเส้นของสัญญาณก่อนหน้า โดยใช้หลักผลรวมกำลังสองของความคลาดเคลื่อนที่มีค่าต่ำสุด ซึ่งการประมาณเชิงเส้นมีอยู่หลายวิธีได้แก่

- วิธีโควาเรียนซ์ (Co-variance Method)
- วิธีออโตคอร์รีเลชัน (Auto-correlation Method)
- วิธีแลตทิส (Lattice Method)

และอื่นๆ อีกหลายวิธี แต่วิธีที่นิยมใช้คือวิธีออโตคอร์รีเลชัน หรือ วิธีอัตตสัมพันธ์ (Auto-correlation)หลังจากผ่านขั้นตอนดังกล่าวจะได้ค่าสัมประสิทธิ์ LPC (α) และอัตราการขยาย (G) ซึ่งในแต่ละขั้นตอนอธิบายได้ดังนี้

2.3.1.1.1 การพรีเอมฟาซิส (Preemphasis)

เนื่องจากสัญญาณเสียงพูดของมนุษย์ มีองค์ประกอบส่วนใหญ่อยู่ในช่วงความถี่ต่ำ เมื่อเทียบกับแถบความถี่ที่ปฏิบัติงาน (bandwidth) ไม่เกิน 5 กิโลเฮิรตซ์ ดังนั้น เพื่อให้อัตราส่วนสัญญาณเสียงต่อสัญญาณรบกวน (Signal to noise ratio : SNR) มีค่าค่อนข้างคงที่ ตลอดช่วงความถี่ที่ปฏิบัติงานนี้ เราจึงต้องมีการพรีเอมฟาซิส เพื่อเน้นความถี่สูงให้มีขนาดสูงขึ้น โดยการกรองสัญญาณด้วยวงจรกรองความถี่สูง (high pass filter) ซึ่งจะใช้วงจร กรองดิจิตอลแบบ first order มีรูปแบบสมการดังนี้

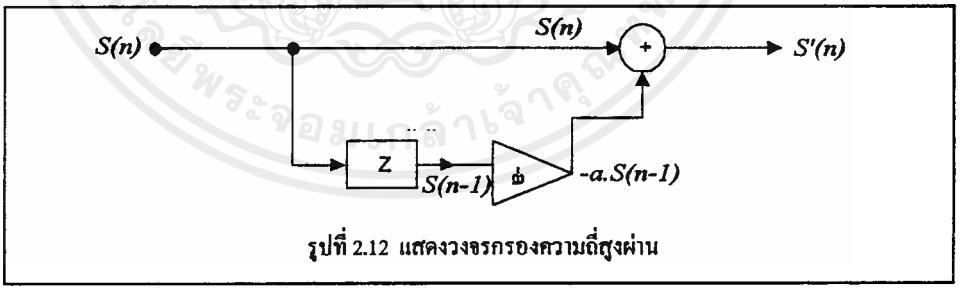
$$y(n) = a_0x(n) - b_1y(n-1) \qquad (2.1)$$

มีฟังก์ชันถ่ายโอนเป็น

$$H(z) = 1 - a.z^{-1} \;;\; 0.9 < a < 1.0 \qquad (2.2)$$

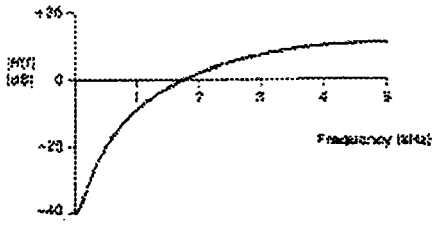
สมมติว่าสัญญาณเดิมเป็น $S(n)$ เมื่อประมาณค่าสัญญาณแล้วจะเป็น $S'(n)$

จะได้ว่า $$S'(n) = S(n) - a.S(n-1) \qquad (2.3)$$



รูปที่ 2.12 แสดงวงจรกรองความถี่สูงผ่าน

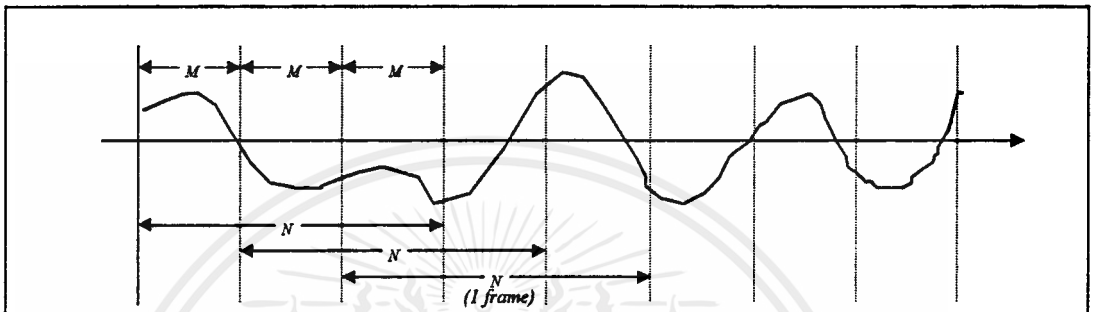
ยิ่งค่า α ใกล้ 1 เท่าใดความถี่สูงก็จะถูกขยายมากขึ้นเท่านั้น ค่า α ที่ควรใช้การพรีเอมฟาซิสคือ 0.9375 ถ้านำฟังก์ชันการพรีเอมฟาซิสมาพล็อตกราฟของ ขนาด กับ ความถี่จะได้กราฟดังรูป



รูปที่ 2.13 แสดงกราฟแอมปลิจูดกับความถี่ เมื่อ $\alpha = 0.9375$

2.3.1.1.2 การแบ่งช่วงสัญญาณ (block into frames)

สัญญาณที่ผ่านการพรีเอมฟาซิสแล้วจะถูกตัดแบ่งออกเป็นช่วงๆ หรือ เฟรมช่วงละ N ตัวอย่างสัญญาณ การวิเคราะห์จะวิเคราะห์ทีละช่วงของแต่ละ N ตัวอย่างสัญญาณ โดยช่วงในการวิเคราะห์แต่ละช่วงจะถูกเลื่อนไปเป็นระยะ M ช่วงสัญญาณ จะเห็นว่าถ้าค่า M โตกว่าค่า N ในการเลื่อนของช่วงในการวิเคราะห์ ก็จะเป็นการสูญเสียส่วนหนึ่งทำให้ผลที่ได้ไม่ถูกต้องเท่าที่ควร ถ้าค่า M เล็กกว่า N ก็จะทำให้ตัวอย่างสัญญาณทุกตัวถูกนำมาวิเคราะห์ ยิ่งค่า M เล็กเท่าใด ความแม่นยำในการวิเคราะห์ก็จะสูงยิ่งขึ้นเท่านั้น แต่ก็จะทำให้การคำนวนช้าลง

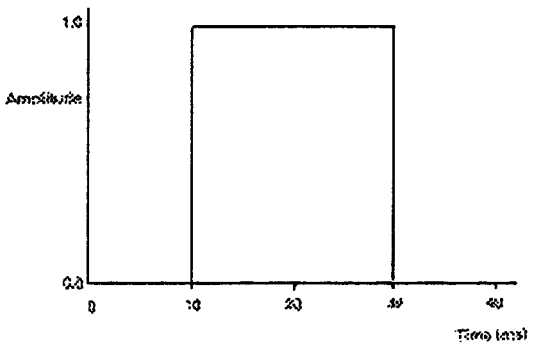


รูปที่ 2.14 แสดงการแบ่งช่วงสัญญาณ

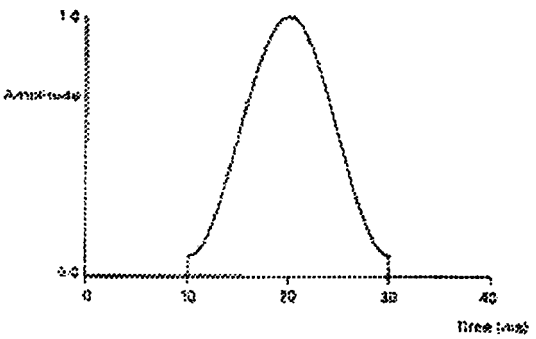
2.3.1.1.3 การวินโดว์ (Windowing)

พิจารณาช่วงสัญญาณ N ตัวอย่างสัญญาณของช่วงใดๆที่ตัดมาวิเคราะห์จะเห็นว่าที่ขอบของเฟรมที่ตัดมานี้มีความไม่ต่อเนื่องของสัญญาณ ถ้ามองในโดเมนความถี่ที่สูงเหล่านี้ เราจะคูณด้วยฟังก์ชันวินโดว์เพื่อลดความไม่ต่อเนื่องของสัญญาณที่ขอบ และไม่ทำให้สเปคตรัมของสัญญาณในช่วงความถี่ต่ำเปลี่ยนไปมากนัก

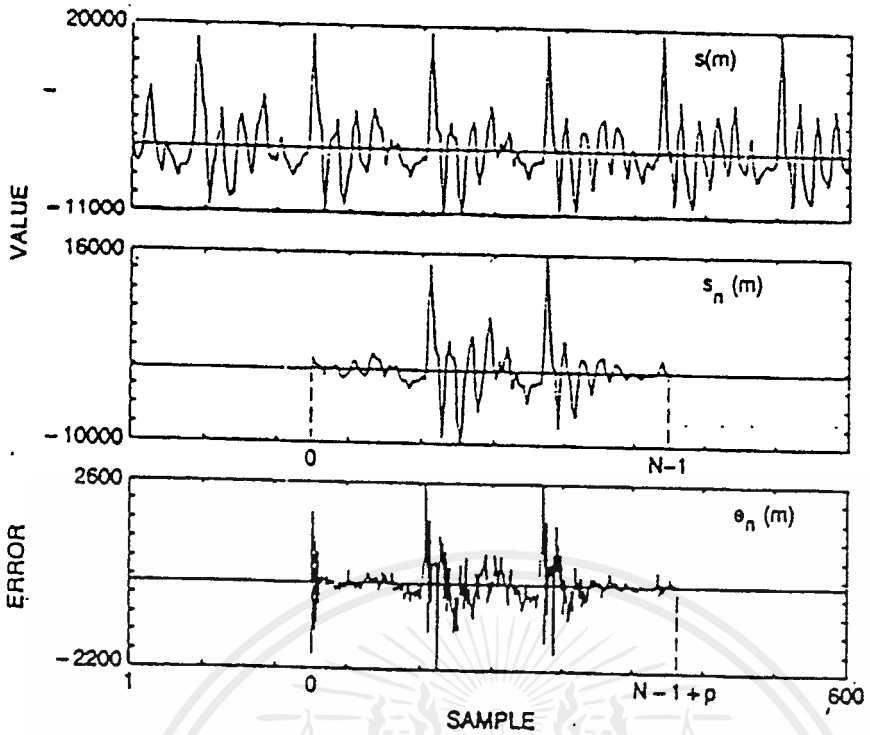
ฟังก์ชันวินโดว์ที่ใช้ในการรวมกับสัญญาณมีหลายชนิด เช่น วินโดว์แบบสี่เหลี่ยม (rectangular window) มีลักษณะดังรูปที่ 2.15 สำหรับฟังก์ชันวินโดว์ที่เหมาะสมที่ใช้กันคือวินโดว์แบบแฮมมิ่ง (Hamming Window) ซึ่งมีลักษณะดังรูปที่ 2.16



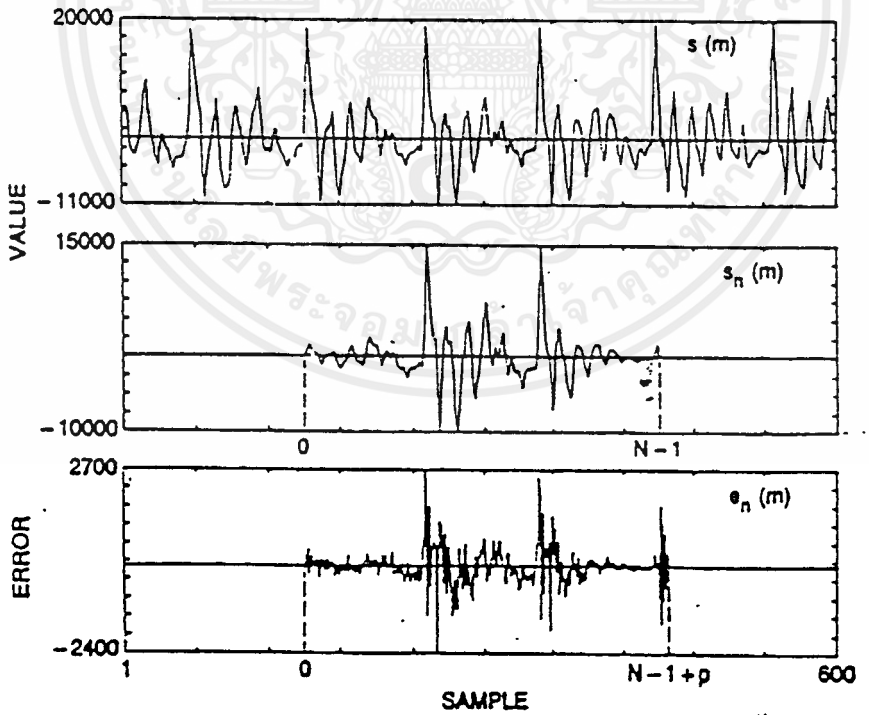
รูปที่ 2.15 แสดง Rectangular window function



รูปที่ 2.16 แสดง Hamming window function



รูปที่ 2.17 แสดงการเกิดความไม่ต่อเนื่องของสัญญาณที่ขอบส่วนต้นของเฟรมที่ตัดมาวิเคราะห์



รูปที่ 2.18 แสดงการเกิดความไม่ต่อเนื่องของสัญญาณที่ขอบส่วนท้ายของเฟรมที่ตัดมาวิเคราะห์

ฟังก์ชั่นวินโดว์แฮมมิ่งมีสมการดังนี้

$$w(n) = 0.54 - 0.46\cos(2\pi n/(N-1)) \quad \text{เมื่อ } n = 0,1,2, \ldots, N-1 \qquad (2.4)$$

ในการวิเคราะห์เสียงโดยใช้ฟังก์ชันวินโดว์ จะพบว่าสัญญาณเสียง ที่ผ่านการกรองโดยวินโดว์นั้นจะมีการแกว่งขึ้นลงมากน้อยขึ้นกับช่วงเวลาของวินโดว์ (ความกว้างของวินโดว์) คือถ้าช่วงเวลาของการวินโดว์สั้น จะมีการแกว่งขึ้นลงอย่างรวดเร็ว และถ้าช่วงเวลาของการวินโดว์มากจะมีการแกว่งขึ้นลงช้าๆ ดังนั้นในการเลือกช่วงเวลาของการวินโดว์ ต้องให้อยู่ในช่วงที่เหมาะสม คือไม่ให้เอาท์พุทของสัญญาณแกว่งช้าหรือเร็วเกินไป อยู่ในช่วงระหว่าง 10 - 30 ms เมื่อคูณกับฟังก์ชันวินโดว์แล้ว ก็จะได้

$$x'(n) = w(n)x(n) \qquad (2.5)$$

2.3.1.1.4 การวิเคราะห์ออโตคอร์รีเลชัน (auto -correlation)

สมมติว่าสัญญาณเดิมเป็น S(n) การประมาณ ค่าสัญญาณเป็น S'(n) ดังนั้นสามารถอธิบายการประมาณเชิงเส้นด้วยสมการต่อไปนี้

$$s'(n) = \sum_{k=1}^{p} \alpha_k s(n--k) \qquad (2.6)$$

เมื่อ α_k เป็นค่าคงที่ เรียกวิธีการนี้ว่าการประมาณเชิงเส้นอันดับ p โดยมีเงื่อนไขว่า ค่า α_k ที่ใช้ในการประมาณจะต้องทำให้ ผลรวมของกำลังสองของความคลาดเคลื่อน $\{s(n)-s'(n)\}^2$ มีค่าน้อยที่สุด นั่นคือ $\sum e^2(n) = \sum\{s(n) - s'(n)\}^2$ มีค่าต่ำที่สุด ซึ่งจะใช้การประมาณเชิงเส้นวิธีออโตคอร์รีเลชั่น (Autocorreiation Method) หรือ วิธีอัตตสัมพันธ์

การคำนวณออโตคอร์รีเลชั่น เป็นวิธีการ หาสัมประสิทธิ์ LPC โดยฟังก์ชั่นออโตคอร์รีเลชั่น ซึ่งเป็นการเปรียบเทียบสัญญาณกับสัญญาณของตัวมันเองที่ถูกเลื่อนไปตามแกนเวลา ที่ใช้วิธีนี้ เนื่องจากเป็นการคำนวณที่มีการแก้สมการที่น้อยกว่าวิธีอื่นๆ และมีความแน่นอนในด้านเสถียรภาพ อีกทั้งมีการเก็บข้อมูลที่น้อยกว่า

การประมาณเชิงเส้นอันดับ p ของอันดับสัญญาณ S(n) และผลรวมเชิงเส้นของสัญญาณนี้ที่ถูกกำหนดด้วยการถ่วงน้ำหนักด้วยค่า α_1 ถึง α_p จะใช้ในการประมาณค่าของสัญญาณถัดไป S'(n) เขียนได้ในรูปสมการ

$$S'(n) = \sum_{k=1}^{p} \alpha_k S(n-k) \quad ; 10 \le p \le 26$$

ผลต่างของค่าสัญญาณประมาณ S'(n) กับ ค่าของสัญญาณปัจจุบันเรียกว่าค่าความคลาดเคลื่อน e(n) (residual signal)

$$e(n) = S(n) - S'(n) \qquad (2.7)$$

การวิเคราะห์การประมาณเชิงเส้นคือการหาค่า α_k ที่ทำให้ค่ากำลังสองของความคลาดเคลื่อนมีค่าน้อยที่สุด ซึ่งสามารถหาได้จากสมการ

$$R(k) = \sum_{n=0}^{N-1-k} S(n)\, S(n+k) \quad ; 0 \le k \le p \qquad (2.8)$$

และแทนค่าในเมตริกซ์เพื่อหา α_k

$$\begin{bmatrix} R_n(0) & R_n(1) & \cdots & R_n(p-1) \\ R_n(1) & R_n(0) & \cdots & R_n(p-2) \\ \vdots & \vdots & \ddots & \vdots \\ R_n(p-1) & R_n(p-2) & \cdots & R_n(0) \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_p \end{bmatrix} = \begin{bmatrix} R_n(1) \\ R_n(2) \\ \vdots \\ R_n(p) \end{bmatrix}$$

หรือ $$R_n \cdot \alpha = r_n \qquad (2.9)$$

เมื่อ $$R_n = \begin{bmatrix} R_n(0) & R_n(1) & \cdots & R_n(p-1) \\ R_n(1) & R_n(0) & \cdots & R_n(p-2) \\ \vdots & \vdots & \ddots & \vdots \\ R_n(p-1) & R_n(p-2) & \cdots & R_n(0) \end{bmatrix}, \quad \alpha = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_p \end{bmatrix} \text{ และ } r_n = \begin{bmatrix} R_n(1) \\ R_n(2) \\ \vdots \\ R_n(p) \end{bmatrix}$$

2.3.1.1.5 การหาอัตราการขยาย G

อัตราการขยายสามารถหาได้โดยตรงจากสมการ

$$G^2 = \frac{R_n(0) - \sum_{k=1}^{p} \alpha_k R_n(k)}{\sum_{m=0}^{N-1} u^2(m)} \qquad (2.10)$$

2.3.1.1.6 การสัมประสิทธิ์เซปสตรัม (Cepsturm)

หลังจากที่หาสัมประสิทธิ์ LPC และGain ใน 1 เฟรมแล้ว จะเปลี่ยนให้เป็นสัมประสิทธิ์เซปสตรัม เนื่องจากการรู้จำเสียงพูดนั้น สัมประสิทธิ์เซปสตรัมนี้เป็นพารามิเตอร์ที่มีลักษณะน่าเชื่อถือได้ดีกว่าสัมประสิทธิ์ LPC ทั้งยังมีความสัมพันธ์ใกล้ชิดกับการรับรู้เสียง ตามความรู้สึกของมนุษย์โดยแท้จริง สัมประสิทธิ์เซปสตรัมสามารถหาได้โดยตรงจากสัมประสิทธิ์ LPC ดังนี้

$C_0 = \ln G$ เป็นสัมประสิทธิ์ตัวแรก ซึ่งเป็นเกน

$Q \approx \frac{3}{2} p$ โดย p = 12 ดังนั้น Q = 18 รวมกับเกน (C_0)

จะได้ว่า สัมประสิทธิ์เซปสตรัมใน 1 เฟรม = 19 ตัว

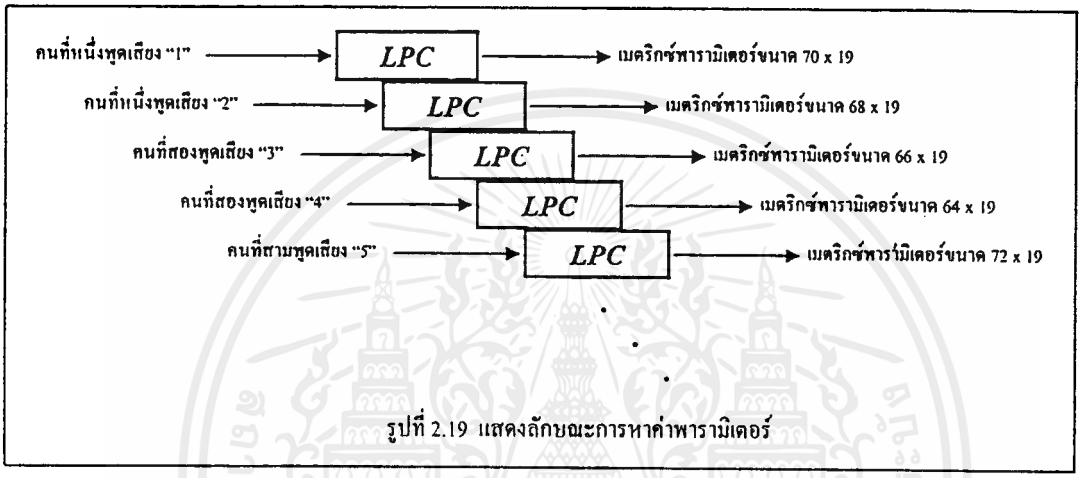
$$C_m = a_m + \sum_{k=1}^{m-1} \left(\frac{k}{m}\right) C_k a_{m-k} \quad , 1 \le m \le p \qquad (2.12)$$

$$C_m = \sum_{k=1}^{m-1} \left(\frac{k}{m}\right) C_k a_{m-k} \quad , m > p \qquad (2.13)$$

2.3.1.1.7 การเวทค่าพารามิเตอร์ (parameter weighting)

เนื่องจาก สัมประสิทธิ์เซปสตรัมที่ได้นั้น ช่วงลำดับต้นๆ และลำดับท้ายๆ ของเฟรมที่นำมาวิเคราะห์จะเกิดความคลาดเคลื่อนมากกว่าบริเวณส่วนอื่น เพราะฉะนั้น จึงทำการถ่วงน้ำหนัก เพื่อลดค่าความคลาดเคลื่อนดังกล่าวนี้ ด้วยฟังก์ชั่นเวทติ้ง ดังนี้คือ

$$W_m = [1 + \frac{Q}{2}\sin(\frac{\pi m}{Q})] \qquad ,1 \le m \le Q \qquad (2.14)$$



รูปที่ 2.19 แสดงลักษณะการหาค่าพารามิเตอร์

จะได้พารามิเตอร์สุดท้าย คือ

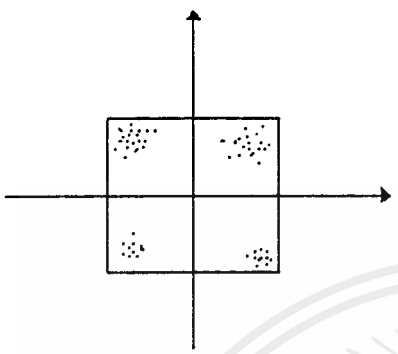
$$C_m' = C_m * W_m$$

จากนั้นก็พิจารณาให้ครบทุกเฟรมของข้อมูล เมื่อพิจารณาเรียบร้อยแล้วก็จะนำไปจัดกลุ่มเสียงและสร้างแบบจำลองเสียงเพื่อใช้ในการเปรียบเทียบต่อไป

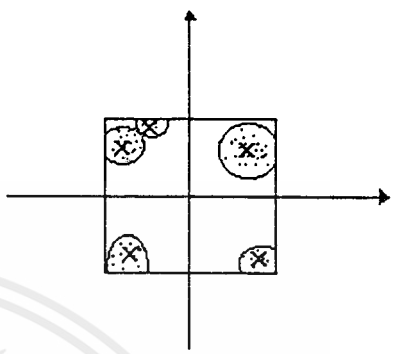
2.3.1.2 การจัดระดับเวคเตอร์ (Vector Quantization)

เวคเตอร์ ควอนไตซ์เซชัน เป็นวิธีการลดไดเมนชั่น (Dimension) หรือจำนวนของข้อมูล เวคเตอร์อินพุท หรือ เซตเทรนนิ่ง หรือ พารามิเตอร์ที่ได้จากขั้น LPC จะถูกเลือกมากลุ่มหนึ่งซึ่งใช้เป็นตัวแทนของข้อมูลจำนวนหนึ่งหรือเรียกกว่าการหา Codebook อินพุทที่เข้ามาจะถูกทำการเปรียบเทียบกับ codebook ที่มีอยู่ โดยจะพิจารณาว่าอินพุทที่เข้ามานั้นห่างจาก codebook ใดน้อยที่สุด อินพุทดังกล่าวจะถูกแทนด้วยเวคเตอร์โค้ด (index) นั้น อินพุททุกตัวที่เป็นสมาชิกของเวคเตอร์โค้ดใดๆ จะถูกนำมาหาจุดศูนย์กลางร่วมใหม่ และนำจุดศูนย์กลางที่ได้นี้ไปทำการหาความคลาดเคลื่อนกับสมาชิกทุกตัว ถ้าค่าความคลาดเคลื่อนที่ได้มีค่ามากกว่าค่าที่กำหนดไว้ค่าหนึ่งหรือค่าที่ยอมรับได้ ก็จะนำศูนย์กลางใหม่นั้นไปเป็น codebook แทน และจะทำการจัดกลุ่มอินพุทเข้ากับ codebook ใหม่ที่ได้และหาความคลาดเคลื่อนอีกครั้งทำอย่างนี้ซ้ำๆ จนการทั้งค่าความคลาดเคลื่อนมีค่าน้อยถึงค่าที่ยอมรับได้ ก็จะถือว่าได้ codebook ที่ดีที่

สุดที่จะเป็นตัวแทนของอินพุททั้งหมด จะสังเกตุได้ว่าทุกครั้งที่มีการหา codebook ใหม่นั้น ค่าความคลาดเคลื่อนที่ได้จะมีค่าลดลงทุกครั้งด้วย



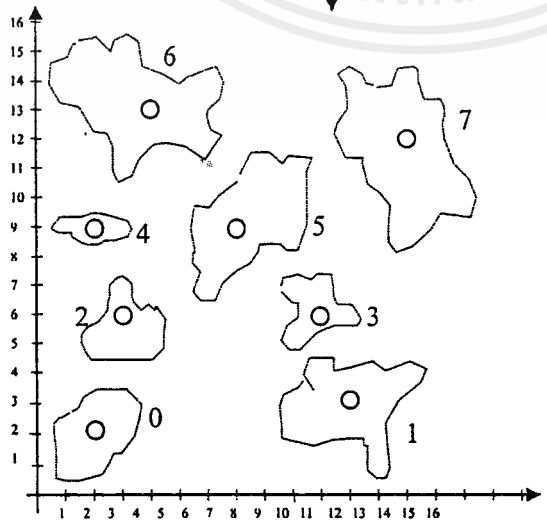
รูปที่ 2.20 แสดงการกระจายเฟรมของเสียงพูด แต่ละจุดแทนเฟรมของเสียง



รูปที่ 2.21 แสดงการรวมกลุ่มของเฟรมเสียงเพื่อไปสร้าง codebook X แทนเวคเตอร์ศูนย์กลางเพื่อแบ่งแยกเวคเตอร์

Training

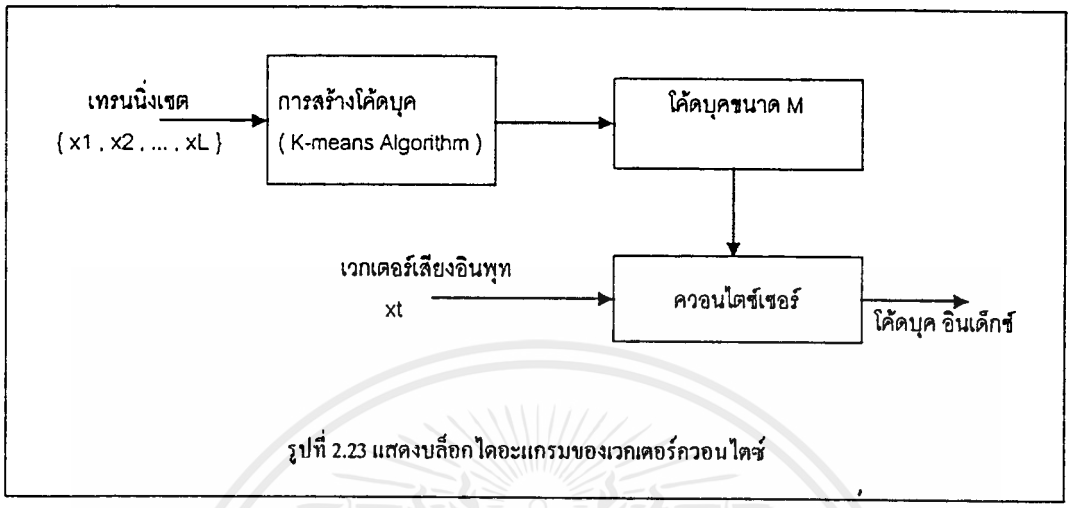
(1,1)	(3,5)	(6,7)	(9,9)	(12,1)
(1,2)	(3,6)	(6,8)	(9,11)	(12,3)
(1,9)	(3,7)	(6,9)	(10,2)	(12,11)
(1,14)	(3,9)	(6,12)	(10,3)	(12,13)
(2,1)	(3,11)	(6,14)	(10,4)	(13,4)
(2,2)	(3,13)	(7,8)	(10,7)	(13,9)
(2,3)	(3,14)	(7,10)	(11,2)	(13,11)
(2,5)	(3,15)	(8,11)	(11,3)	(13,12)
(2,9)	(4,5)	(9,2)	(11,4)	(13,14)
(2,13)	(4,6)	(9,3)	(11,6)	(14,12)
(2,15)	(4,12)	(9,5)	(11,12)	(14,13)
(3,2)	(4,14)	(9,7)	(11,14)	(15,10)
(3,3)	(5,13)			



Codebook	
Index	Vector
0	2,2
1	11,3
2	3,6
3	10,6
4	2,9
5	7,9
6	4,13

รูปที่ 2.22 แสดงตัวอย่างการหา Codebook

ตัวอย่างเวกเตอร์ควอนไตซ์เซชั่น สมมุติให้เวกเตอร์แต่ละตัวมี 2 มิติ และทำการหา Codebook ขนาด 8 เวกเตอร์



รูปที่ 2.23 แสดงบล็อกไดอะแกรมของเวกเตอร์ควอนไตซ์

เวกเตอร์ทั้งหมดจะถูกจัดเข้าไปในกลุ่มของ Codebook ต่างๆ แล้วทำการหาจุดศูนย์ใหม่โดยการเฉลี่ยค่าเวกเตอร์สมาชิกทุกตัวที่อยู่ในกลุ่มเดียวกัน ผลที่ได้คือ Codebook 8 ตัวที่เป็นตัวแทนของเวกเตอร์ทั้งหมด

การทำงานของการควอนไตซ์แบบเวคไตซ์ แบ่งเป็น 2 ขั้นตอนดังนี้

2.3.1.2.1 การสร้างโค้ดบุค (codebook) โดยใช้วิธี K-means

จากขั้นตอนการประมาณเชิงเส้นของเสียงตัวอย่างจำนวนมากจะได้เทรนนิ่งเซตซึ่งประกอบด้วยเวคเตอร์สเปคตรัมจำนวน L เฟรม ; $x = \{ x_i , 1 \leq i \leq L \}$ เฟรมละ P มิติ ; $x = [x_1, x_2, \ldots, x_p]$ แล้วนำข้อมูลที่ได้มาทำการสร้างเป็นกลุ่มของแบบอ้างอิง

ในระบบการรับรู้เสียงพูดแบบต่างบุคคล จะใช้แบบอ้างอิงของคำหนึ่งๆ จากผู้พูดจำนวนมาก เพื่อที่จะได้ครอบคลุมถึงความแปรปรวนต่างๆ ที่เกิดขึ้นระหว่างผู้พูดแต่ละคน เนื่องจากถ้าใช้แบบอ้างอิงจำนวนมาก เวลาที่ใช้ในการตอบสนองจะมาก เนื้อที่ในหน่วยความจำสำรองที่ใช้เก็บแบบอ้างอิงจะเพิ่ม และเมื่อเพิ่มแบบอ้างอิงไปจนถึงระดับหนึ่ง ความถูกต้องในการรับรู้จะเริ่มคงที่ ดังนั้นจึงมีการจัดกลุ่มของแบบอ้างอิงใหม่เพื่อให้ได้แบบอ้างอิงที่เหมาะสม และสามารถใช้เป็นตัวแทนของแบบอ้างอิงที่มีอยู่ทั้งหมดได้ อัลกอริทึมที่ใช้ได้แก่ K-means Algorithm ซึ่ง ขั้นตอนที่ใช้ในการสร้างโค้ดบุคมีดังนี้

2.3.1.2.1.1 นำเทรนนิ่งเซตมาใช้ในการสร้างโค้ดบุค

ขนาดโค้ดบุคของการควอนไตซ์แบบเวคเตอร์ คือ $M = 2^B$ เวคเตอร์ (B - bit codebook) และเพื่อที่จะหาเซตของ M โค้ดบุคที่ดีที่สุด จำนวนเวคเตอร์อินพุทจะต้องมากกว่าขนาดโค้ดบุคมาก ($L >> M$)

2.3.1.2.1.2 การสุ่มค่าเริ่มต้น

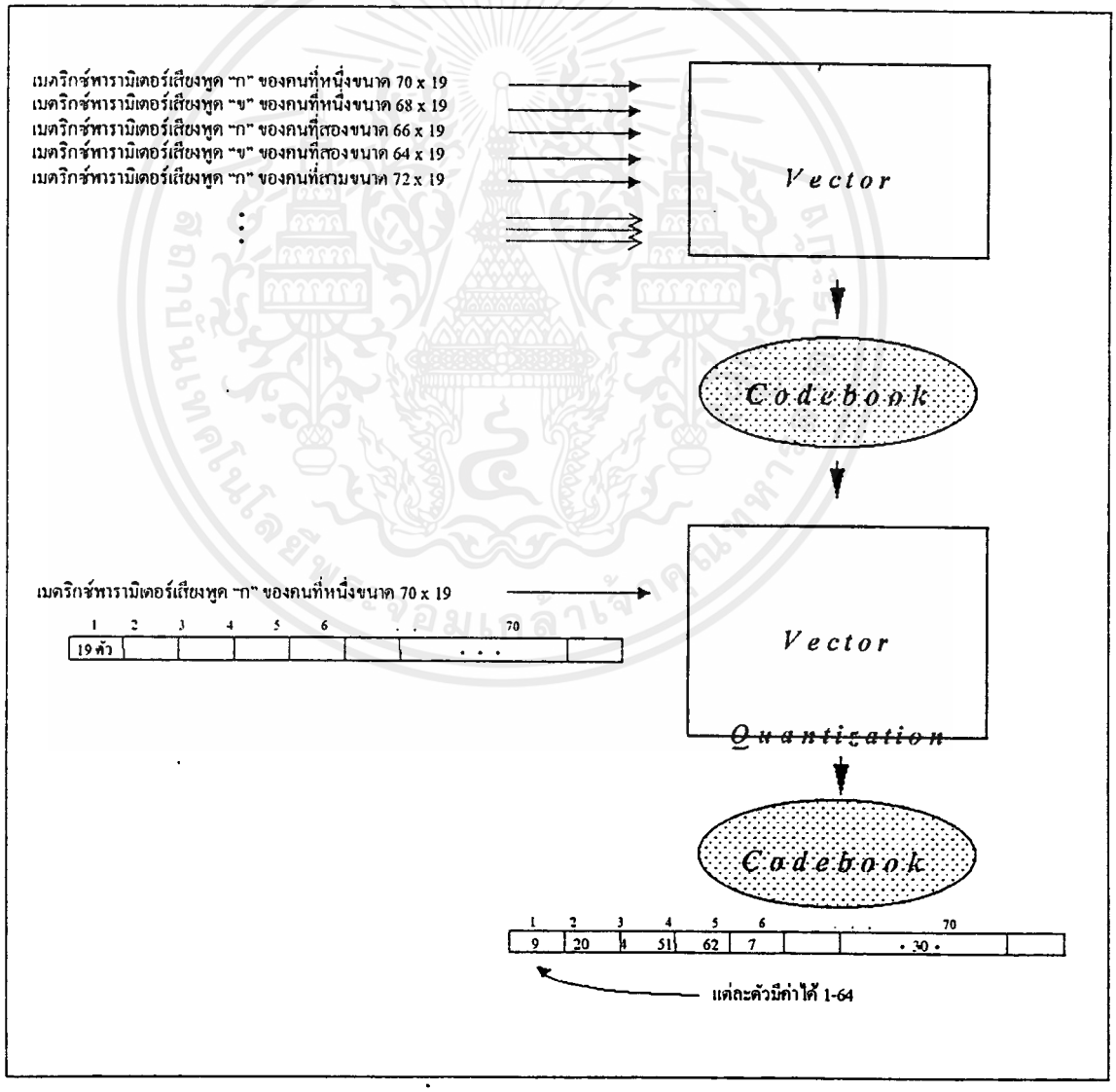
การสุ่มค่าเริ่มต้น เป็นวิธีหนึ่งในการออกแบบโค้ดบุค ซึ่งคือ การเลือกค่าเริ่มต้นของโค้ดบุค เรียกโค้ดบุคที่ได้จากการสุ่มค่าเริ่มต้นนี้ว่า แรนดอมโค้ดบุค (random codebook) ถึงแม้วิธีนี้จะไม่ใช่วิธีที่ดีนัก แต่โค้ดบุคที่ได้จากการสุ่มก็ให้ผลเป็นที่ยอมรับ

2.3.1.2.1.3 การหาความคลาดเคลื่อน

การวัดความคลาดเคลื่อน เป็นส่วนที่จำเป็นและเป็นประโยชน์ต่อการออกแบบโค้ดบุค สมการทางพีชคณิตที่ใช้ในการหาระยะทาง มีหลายวิธี แต่วิธีที่นำมาใช้ คือ การหาความคลาดเคลื่อนกำลังสองรวม (Total square error) ซึ่งเป็นวิธีการคำนวณที่ง่ายและรวดเร็ว

ถ้าสัญญาณมี P มิติ สามารถหาระยะห่างระหว่างสัญญาณอินพุท (x) กับเวคเตอร์โค้ด (y) โดยสมการ

$$d(v_1,v_2) = \left\| v_1 - v_2 \right\|^2 = \sum_{i=0}^{k-1}(x_i - y_i)^2 \qquad (2.15)$$



รูปที่ 2.24 ขั้นตอนของเวกเตอร์ควอนไตเซชั่น

2.3.1.2.1.4 การจัดกลุ่ม (classification) และการหาจุดศูนย์กลางของกลุ่ม (center cluster)

การจัดกลุ่มเป็นการแบ่งเวคเตอร์อินพุทเข้าไปตามกลุ่มต่างๆ ของแรนดอมโค้ดบุค โดยพิจารณาระยะทาง หรือความคลาดเคลื่อนน้อยที่สุด ของแต่ละเวคเตอร์อินพุท x กับเวคเตอร์ โค้ดบุค y ซึ่งเป็นโค้ดบุค จากนั้นจะทำการหาค่าเฉลี่ยของแต่ละกลุ่ม เพื่อเป็นค่ากลางของกลุ่มนั้นๆ จะได้

$$\overline{Y} = \frac{1}{L}\sum_{i=1}^{L} x_i$$

$\overline{Y}$ เป็นจุดศูนย์กลางซึ่งเป็นเวคเตอร์ที่อยู่ตรงกลางของ $\{x_i\}_{i=1}^{L}$ ซึ่งแต่ละมิติจะไม่ขึ้นแก่กันหมายความว่า แต่ละ y_k เป็นค่ากลางของ $\{x_j\}_{i=1}^{L}$

ทำ 2 ขั้นตอนนี้ซ้ำ จนกว่าจะเกิดการลู่เข้า (convergent) โดยความคลาดเคลื่อนรวมจะต่ำกว่าค่าหนึ่งๆ ซึ่งค่าความคลาดเคลื่อนรวมจะลดลงทุกครั้งที่มีการคำนวณซ้ำใหม่ จึงขึ้นกับค่าที่กำหนดว่าต้องการให้ความคลาดเคลื่อนรวมน้อยเท่าใด ค่ากลางดังกล่าวของแต่ละกลุ่มจะถูกเก็บเป็นเวคเตอร์โค้ดจะได้ว่า y เป็นควอนไตซ์ของค่า x

$$y = q(x)$$

โดย q(.) เป็นโอเปอร์เรเตอร์ของควอนไตซ์ y ถูกเรียกว่าเอาท์พุทเวคเตอร์ของค่า x โดย y เป็นค่าใดค่าหนึ่งใน $Y = \{ y_i, 1 \le i \le M \}$ โดย $y_i = [y_{i1}\ y_{i2} \ldots y_{iP}]$ Y เป็นเซตของโค้ดบุค M เป็นขนาดของโค้ดบุค และ $\{ y_i \}$ เป็นเซตของเวคเตอร์โค้ด y_i อาจเรียกว่าเป็นโค้ดอ้างอิง และ M อาจเรียกว่าจำนวนระดับขั้น จะทำการแบ่งเวคเตอร์ x ไปใน M เซล $\{ C_i, 1 \le i \le M \}$ เมื่อ x อยู่ในเซล C_i

$$q(x) = Y_i \quad \text{ถ้า} \quad x \in C_i$$

2.3.1.2.2 การเปรียบเทียบ

เวคเตอร์ควอนไตเซชั่นที่ใช้เพื่อการออกแบบการรับรู้เสียงพูดนั้น มีจำนวนควอนไตเซอร์ M ตัว ซึ่งหมายถึง มี M ระดับเสียงเพื่อการรับรู้ แต่ละระดับเสียงพิจารณามาจากเซตของข้อมูลเทรนนิ่ง ซึ่ง M เป็นดัชนีระดับ แต่ละเซตของเทรนนิ่งในแต่ละระดับจะเก็บเสียงที่อยู่ในระดับเดียวกัน

เมื่อมีเสียงที่เราไม่ทราบ (unknown) ; x_t เข้ามา จะเป็นอินพุทเข้าไปยังทุกๆ ควอนไตเซอร์ ค่าดัชนีระดับ (index) ที่ถูกเลือก จะเป็นระดับที่มีความคลาดเคลื่อน เฉลี่ยน้อยที่สุด D (c) เมื่อ I = 1,2,...,M ซึ่งความคลาดเคลื่อนเฉลี่ยนั้นหาได้จากการวัดระยะทาง โดยใช้วิธีการหาความคลาดเคลื่อนกำลังสองรวม

2.3.2 Pattern Matching

วิธีการในการสร้างและหารูปแบบของคำพูดที่เหมือนมีอยู่สองแนวทาง

- Dynamic time warping (DTW)
- Hidden Markov Models (HMM)

ในที่นี้จะเลือกใช้แบบ HMM เนื่องจากเหมาะกับการวิเคราะห์คำพูดทั้งแบบต่อเนื่องและไม่ต่อเนื่องได้ดี

Hidden Markov Models

แบบจำลองมาร์คอฟ เป็นแบบจำลอง (model) ทางสถิติซึ่งพัฒนาเพื่อแบ่งกลุ่มของอนุกรมทางเวลา หรือสัญญาณที่ไม่คงที่ นั่นคือ ใช้สำหรับจัดกลุ่มของสัญญาณที่ไม่รู้จัก (Unknown signal) ให้ไปอยู่ในกลุ่มใดกลุ่มหนึ่งของสัญญาณ ซึ่งแบบจำลองมาร์คอฟ ได้ถูกนำมาประยุกต์ใช้ในการรู้จำเสียงพูด

แบบจำลองมาร์คอฟ แบ่งออกเป็น 2 ประเภทคือ แบบต่อเนื่อง (Continuous) และแบบไม่ต่อเนื่อง (Discrete-time) ในที่นี้จะเลือกใช้แบบไม่ต่อเนื่องเพราะเป็นวิธีการที่ซับซ้อนน้อยกว่าและใช้ได้ดีกับคำพูดสั้นๆ

2.3.2.1 ส่วนประกอบของแบบจำลอง HMM

2.3.2.1.1 N คือจำนวนสเตทในแบบจำลอง โดยสามารถย้ายจากสเตทหนึ่งไปยังอีกสเตทหนึ่งได้ เราให้เซตของสเตทเป็น { 1,2,...,N } และสเตทที่เวลา t ใดๆเป็น q_t

2.3.2.1.2 M คือจำนวนของค่าปรากฏต่อหนึ่งสเตท แทนด้วยสัญลักษณ์ $V = \{ V_1, V_2, ..V_M \}$

2.3.2.1.3 $A = \{a_{ij}\}$ คือความน่าจะเป็นในการเปลี่ยนสเตทที่ $a_{ij} = P\{ q_t=j \mid q_{t-1}=i \}$ เมื่อ $1 \le i,j \le N$

2.3.2.1.4 $B = \{b_j(k)\}$ คือความน่าจะเป็นของการเกิดค่าปรากฏ ที่ $b_j(k)=P\{ O_t=V_k|q_t=j \}$ เมื่อ j=1,2,...,N

2.3.2.1.5 π_i คือความน่าจะเป็นที่แต่ละสเตทจะเป็นสเตทเริ่มต้น เมื่อ $\pi_i = P(q_i$ ที่เวลา t=1)

2.3.2.2 โครงสร้างของแบบจำลองHMM

แบ่งตามลักษณะการเปลี่ยนสเตท(transition)ของเมตริกซ์ A

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$$

2.3.2.2.1 แบบ Egordic Model หรือ Fully Connected Model แบบจำลองนี้ทุกสเตทสามารถเปลี่ยนสเตทไปยังสเตทอื่นๆได้ทุกๆสเตท

2.3.2.2.2 แบบ Left - Right Model หรือ Bakis Model แบบจำลองนี้ การเปลี่ยนสเตทจะเปลี่ยนจากซ้ายไปขวา มีคุณสมบัติการเปลี่ยนสเตทดังนี้

2.3.2.2.2.1 $a_{ij}=0$, $j<1$ หมายความว่า เมื่อผ่านสเตทใดไปแล้วจะไม่มีการย้อนกลับมายังสเตทนั้นอีก

2.3.2.2.2.2 $\pi_i = \{ 0$ เมื่อ $i \neq 1$; 1เมื่อ $i=1\}$ หมายความว่า ลำดับของสเตทต้องเริ่มต้นที่สเตทที่ 1 สเตทที่เหลือจึงมีความน่าจะเป็นที่จะเป็นสเตทเริ่มต้นเท่ากับศูนย์

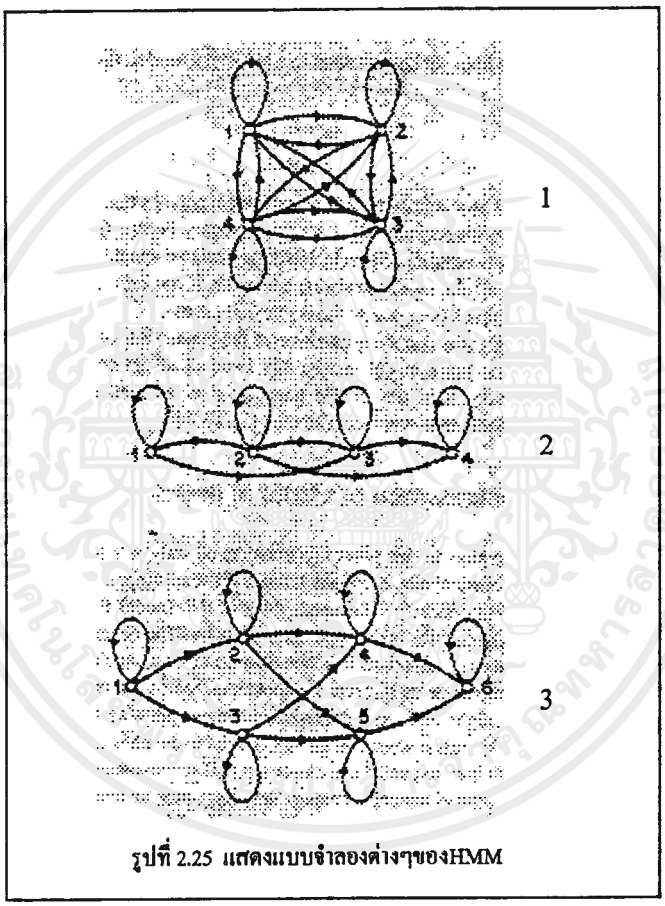
Left-Right Model นี้มีกฎข้อบังคับการเปลี่ยนสเตทดังนี้

$a_{ij}=0$ เมื่อ $i > i+\Delta i$ โดยค่าของ $\Delta i=2$ หมายความว่า การเปลี่ยนสเตทจะสามารถเปลี่ยนได้เกิน 2 สเตท จะได้เมตริกซ์การเปลี่ยนสเตทเป็น

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 \\ 0 & a_{22} & a_{23} & a_{24} \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & a_{44} \end{bmatrix}$$

จะเห็นว่าสเตทสุดท้ายมีสัมประสิทธ์การเปลี่ยนสเตทเป็น $a_{NN}=1$, $a_{Ni}=0$ เมื่อ $i<N$ แบบจำลองนี้จึงเหมาะกับสัญญาณที่มีการเปลี่ยนแปลงอย่างต่อเนื่องเช่น คำพูด

2.3.2.2.3 แบบ Parallel Left-Right Model มีคุณสมบัติการเปลี่ยนสเตทคล้ายแบบที่ 2 แต่มีความยืดหยุ่นมากกว่า



รูปที่ 2.25 แสดงแบบจำลองต่างๆของHMM

ปัญหาของHMM

ปัญหาของ HMM มี 3 ข้อซึ่งต้องใช้อัลกอริทึมวิธีต่างๆในการคำนวณเพื่อแก้ปัญหานี้

ปัญหาที่ 1 เมื่อลำดับของค่าปรากฏ $O=\{O_1,O_2,...,O_T\}$ และมีโมเดล $\lambda=(A,B,\pi)$ เราจะคำนวณ หาค่า$P(O|\lambda)$ ของลำดับของค่าปรากฏได้อย่างไร

ปัญหาที่ 2 เมื่อมีลำดับของค่าปรากฏ $O=\{O_1,O_2,...,O_T\}$ และแบบจำลอง $\lambda=(A,B,\pi)$ เราจะหาลำดับสเตท $q=\{q_1q_2...q_T\}$ ที่เหมาะสมในการให้ค่าปรากฏนั้นได้อย่างไร

ปัญหาที่ 3 จะหาแบบจำลอง $\lambda=(A,B,\pi)$ ที่ให้ค่า$P(O|\lambda)$ มากที่สุดได้อย่างไร

ลำดับของค่าปรากฏที่ใช้ปรับค่าพารามิเตอร์ A,B และ π เพื่อให้ได้แบบจำลองที่ดีที่สุดนั้นเรียกว่า ลำดับเทรนนิ่ง(training sequence)

2.3.2.3 การคำนวณเพื่อแก้ปัญหาของHMM

2.3.2.3.1 การแก้ปัญหาที่ 1 เป็นการคำนวณว่าแบบจำลอง λ ให้ความน่าจะเป็นที่จะได้ลำดับค่าปรากฏมากน้อยเพียงใด มีวิธีการเพื่อช่วยแก้ปัญหาโดยใช้กระบวนการต่อไปนี้

2.3.2.3.1.1 กระบวนการไปข้างหน้า (Forward Procedure)

เมื่อกำหนดให้ตัวแปรไปข้างหน้า(forward variable) $\alpha_t(i) = P(O_1,O_2,....,O_T,q_t=i|\lambda)$ หมายถึง ความน่าจะเป็นของการเกิดลำดับค่าปรากฏ $O_1,O_2,....,O_T$ ที่จะอยู่ที่สเตท i ณ เวลา t โดยมีแบบจำลองเป็น λ โดยสามารถหา $\alpha_t(i)$ ได้ดังนี้

2.3.2.3.1.1.1 <u>การเริ่มต้น (initializa tion)</u>

เมื่อกำหนด $\alpha_t(i) = \pi_i b_i(O_1)$ ที่เวลาเริ่มต้น t=1 และเหตุการณ์เริ่มต้น O_1 เมื่อ $1 \le i \le N$

2.3.2.3.1.1.2 <u>การเหนี่ยวนำ (induction)</u>

$\alpha_{t+1}(j) = [\sum_{i=1}^{N} \alpha_t(i)\, a_{ij}]\, b_j(o_{t+1})$ เมื่อ $1 \le t \le T-1$ และ $1 \le j \le N$ หมายถึง ความน่าจะเป็นของสเตท j ที่เวลา t+1 ได้มาจากสเตท i ที่เป็นไปได้ถึง N สเตท ที่เวลา t ดังรูปที่ 3.14

2.3.2.3.1.1.3 <u>การสิ้นสุด (termination)</u>

$P(O|\lambda) = \sum_{i=1}^{N} \alpha_t(i)$ ความน่าจะเป็นของลำดับค่าปรากฏ O ได้จากผลรวมของ $\alpha_t(i)$ จากทุกๆ สเตท เมื่อ $1 \le j \le N$

2.3.2.3.1.2 กระบวนการย้อนกลับ (Backward Procedure)

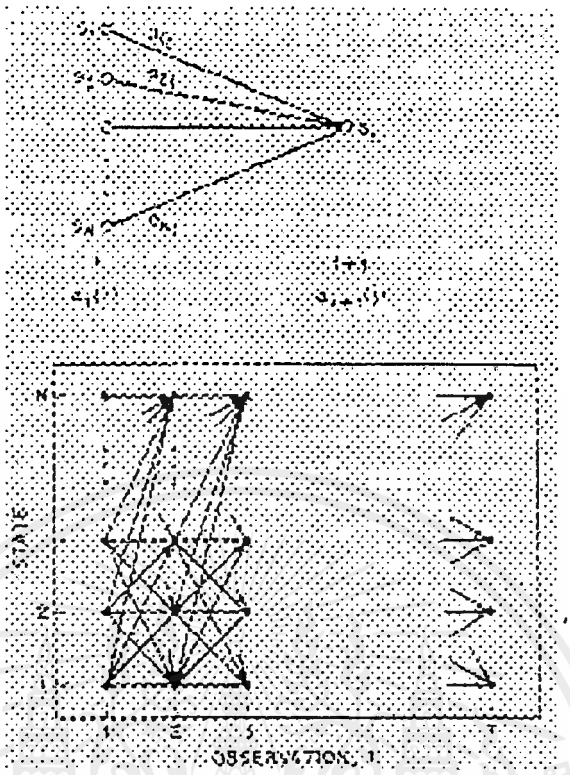
เมื่อกำหนดให้ตัวแปรย้อนกลับ(backward variable) $\beta_T(i) = P(o_{t+1}o_{t+2}....o_T,q_t=i\,|\lambda)$ หมายถึง ความน่าจะเป็นของลำดับค่าปรากฏส่วนหลัง จากเวลา t+1 ไปจนจบ โดยกำหนดว่าต้องอยู่ที่สเตท i ที่เวลา t และมีแบบจำลองเป็น λ เราจะคำนวณหา $\beta_T(i)$ ได้ดังนี้

2.3.2.3.1.2.1 การเริ่มต้น (initialization)

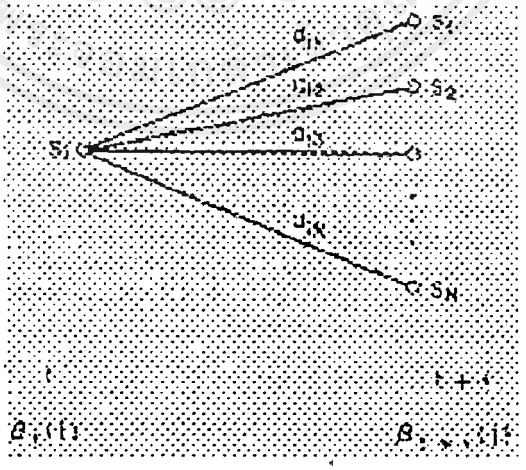
$\beta_T(i) = 1$ เมื่อ $1 \le j \le N$

2.3.2.3.1.2.2 การเหนี่ยวนำ(induction)

$$\beta_T(i) = \sum_{j=1}^{N} a_{ij}\, b_j(o_{t+1})\beta_{t+1}(i) \text{ เมื่อ } t=T-1,T-2,...1,\ 1 \le i \le N \quad (2.16)$$



รูปที่ 2.26 แสดงกระบวนการไปข้างหน้า



รูปที่ 2.27 แสดงกระบวนการถอยหลัง

2.3.2.3.2 การแก้ปัญหาที่ 2 เพื่อหาลำดับสเตทที่เหมาะสม

เราจะใช้วิธี วิทเทอร์บี อัลกอริทึม (Viterbi Algorithm) เพื่อหาลำดับสเตทที่ดีที่สุด ณ เวลา t หนึ่งๆ เมื่อกำหนดลำดับเหตุการณ์ $O = (o_1 o_2 ....o_t)$ โดยนิยามให้

$$\delta_t(i) = \max_{q_1,q_2,...,q_t} P[q_1 q_2 ....q_{t-1}, q_t=i, o_1 o_2 ....o_t | \lambda]$$

หมายถึง ความน่าจะเป็นสูงสุดของเส้นทาง (path) ณ เวลา t ซึ่งเริ่มนับจากเหตุการณ์ที่เวลาเริ่มต้นจนถึงเวลา t ที่สเตท i และ โดยการอาศัยคุณสมบัติการเหนี่ยวนำ (induction) เราจะได้

$$\delta_{t+1}(i) = [\max_{i} \delta_t(i)\, a_{ij}]\, b_j(o_{t+1})$$

เราสามารถหาลำดับสเตทที่ดีที่สุดได้โดยใช้กระบวนการต่อไปนี้ เมื่อกำหนดให้ $\psi_t(i)$ เป็น อาเรย์ (array)

2.3.2.3.2.1 การเริ่มต้น (initialization)

$$\delta_t(i) = \pi_i b_i(o_1) \text{ เมื่อ } 1 \le j \le N$$

$$\psi_t(i) = 0$$

2.3.2.3.2.2 การย้อนกลับ (recursion)

$$\delta_t(j) = [\max_{1 \le i \le N} \delta_{t-1}(i)\, a_{ij}]\, b_j(o_t) \qquad \text{เมื่อ } 2 \le t \le T,\ 1 \le j \le N$$

$$\psi_t(j) = \arg \max_{1 \le i \le N} [\delta_{t-1}(i)\, a_{ij}] \qquad \text{เมื่อ } 2 \le t \le T,\ 1 \le j \le N$$

2.3.2.3.2.3 การสิ้นสุด (termination)

$$P^* = \max_{1 \le i \le N} [\delta_T(i)]$$

$$q_T^* = \max_{1 \le i \le N} [\delta_T(i)]$$

2.3.2.3.2.4 เส้นทางเดินย้อนกลับ (Path backtracking)

$$q_t^* = \psi_{t+1}(q_{t+1}^*) \text{ เมื่อ } t = T-1, T-2, ..., 1$$

2.3.2.3.3 การแก้ปัญหาที่ 3 เพื่อหาโมเดลที่จะให้ผลตามลำดับค่าปรากฏิหนึ่งๆโดยเลือกค่าพารามิเตอร์ A,B,π ที่ดีที่สุด โดยใช้ กระบวนการทำซ้ำ (Iterative) วิธีที่เราเลือกใช้ คือวิธี บาม-เวลช์ (Baum-Welch) หรือ EM (Expectation-Maximization)

เมื่อนิยามให้

1. $\gamma_t(i) = P(q_t=i | O, \lambda)$

หมายถึง ความน่าจะเป็นที่จะอยู่ที่สเตท i ณ เวลา t โดยกำหนดลำดับเหตุการณ์ O และแบบจำลอง λ ให้ สามารถแสดงค่า $\gamma_t(i)$ ได้ดังนี้

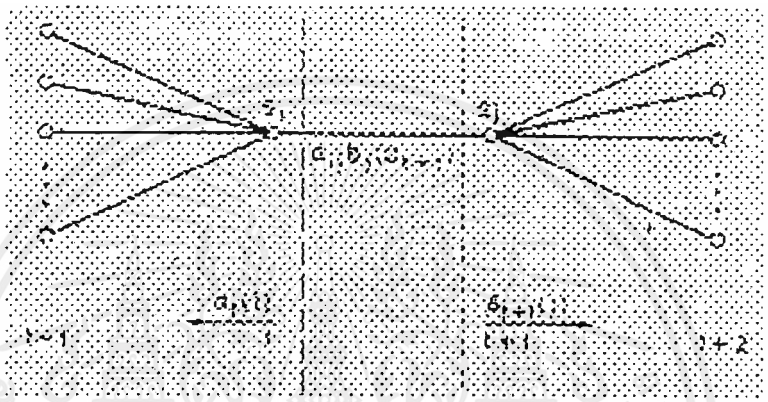
$$\gamma_t(i) = \frac{p(O, q_t = i|\lambda)}{p(O|\lambda)} = \frac{p(O, q_t = i|\lambda}{\sum_{i=1}^{N} P(O, q_t = i|\lambda)}$$

เนื่องจาก $P(O,q_t=i|\lambda)$ มีค่าเท่ากับ $\alpha_t(i)\beta_t(i)$ จึงได้

$$\gamma_t(i) = \frac{\alpha(i)\beta_t(i)}{\sum_{i=1}^{N}\alpha(i)\beta_t(i)}$$

2. $\varepsilon_t(i,j) = P(q_t=i, q_{t+1}=j \mid O,\lambda)$

หมายถึง ความน่าจะเป็นที่จะอยู่ที่สเตท i ที่เวลา t และสเตท j ที่เวลา t+1 เมื่อกำหนดแบบจำลองและลำดับค่าปรากฏให้



รูปที่ 2.28 แสดงถึงค่าปรากฏที่จะอยู่ที่สเตท i ที่เวลา t โดยคำนึงถึงลำดับค่าปรากฏจากเวลา t+1 ซึ่งต้องพิจารณาสเตท j ที่จะเป็นไปได้ทั้งหมด ณ เวลา t+1 โดยจะขึ้นอยู่กับค่า a_{ij} และ $b_j(o_{t+1})$

ซึ่งจากนิยามของตัวแปรไปข้างหน้าและตัวแปรย้อนกลับ สามารถนำมาสัมพันธ์กับ $\varepsilon_t(i,j)$ ได้ดังนี้

$$\varepsilon_t(i,j) = \frac{P(q_t = i, q_t + 1 = j, O|\lambda)}{P(O|\lambda)}$$

$$= \frac{\alpha_t(i)a_{ij}b_j(o_t + 1)\beta_t + 1(j)}{P(O|\lambda)}$$

$$= \frac{\alpha_t(i)a_{ij}b_j(o_t + 1)\beta_t + 1(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_t(i)a_{ij}b_j(o_t + 1)\beta_t + 1(j)}$$

และจะได้ความสัมพันธ์ของ $\gamma_t(i)$ กับ $\varepsilon_t(i,j)$ ดังนี้

$$\gamma_t(i) = \sum_{j=1}^{N} \varepsilon_t(i,j)$$

และ

$\sum_{t=1}^{T-1} \gamma_t(i)$ = จำนวนของการเปลี่ยนสเตทออกจากสเตท i ในลำดับค่าปรากฏ O

$\sum_{t=1}^{T-1} \varepsilon_t(i,j)$ = จำนวนของการเปลี่ยนสเตทจากสเตท i ไป j ในลำดับค่าปรากฏ O

ดังนั้นสามารถหาค่าพารามิเตอร์ได้ดังนี้

$\pi_i' = \gamma_1(I)$ เมื่อ $1 \le i \le N$

$$a_{ij}' = \frac{\sum_{t=1, O_t = v_k}^{T} n(j)}{\sum_{t=1}^{T-1} n(i)}$$

$$b_j(k) = \frac{\sum_{t=1, O_t = v_k}^{T} n(j)}{\sum_{t=1}^{T} n(j)}$$

จากกระบวนการข้างต้น ถ้าเราจะคำนวณซ้ำๆ โดยให้ $\lambda' = (A', B', \pi')$ แทน $\lambda = (A, B, \pi)$ซึ่งเป็นแบบจำลองเริ่มต้นแล้ว จะทำให้ความน่าจะเป็นของการเกิดลำดับค่าปรากฏ O ดีขึ้น จนกระทั่งถึงจุดวิกฤตจึงหยุด ซึ่งเราจะได้จุดวิกฤตของฟังก์ชันความน่าจะเป็นในกรณีที่ $\lambda' = \lambda$ หรือถ้า λ' มีความน่าจะเป็นมากกว่าแบบจำลอง λ ในลักษณะที่ $P(O|\lambda') > P(O|\lambda)$ นั่นก็คือ เราก็จะได้ แบบจำลอง λ' ใหม่ ที่น่าจะทำให้เกิดลำดับค่าปรากฏ 0 ได้ดีกว่า

2.3.2.4 การปรับค่าพารามิเตอร์ของ HMM

2.3.2.4.1 การสเกลลิ่ง (Scaling)

เนื่องจาก $\alpha_t(i)$ จะประกอบด้วยผลรวมของเทอมจำนวนมาก ซึ่งก็คือ

$$\left(\prod_{s=1}^{t-1} a_{q_s q_{s+1}} \quad \prod_{s=1}^{t} b_{q_s}(O_s) \right)$$

และเนื่องจากแต่ละเทอมของ a และ b มีค่าน้อยกว่า 1 อยู่แล้ว เมื่อพิจารณาผลรวมของการคูณค่ายิ่งน้อยลงเรื่อยๆ แสดงว่าเมื่อ t มากขึ้น แต่ละเทอมของ $\alpha_t(i)$ จะเข้าสู่ศูนย์ ทำให้ Dynamic Range ของการคำนวณ $\alpha_t(i)$ เกิน Range ของคอมพิวเตอร์ ทำให้ค่าที่ได้ไม่ถูกต้อง จึงได้มีการสเกลลิ่งขึ้นเพื่อทำให้ $\alpha_t(i)$ อยู่ภายใน Dynamic Range ของคอมพิวเตอร์ การสเกลลิ่งทำได้โดยการคูณ $\alpha_t(i)$ โดยสัมประสิทธิ์การสเกลลิ่ง ซึ่งสัมประสิทธิ์นี้ไม่ขึ้นกับ i การสเกลลิ่ง $B_t(i)$ ก็เช่นเดียวกัน หลังการคำนวณค่าการสเกลลิ่งก็จะตัดกันหมดไปเอง พิจารณาจาก สมการ

$$\overline{a_{ij}} = \frac{\sum_{t=1}^{T-1} \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\sum_{t=1}^{T} \sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)} \qquad (2.17)$$

เมื่อเราให้ $\alpha_t(i)$ แทน α ที่ยังไม่ได้สเกลลิ่ง

$\alpha_t(i)$ แทน α ที่สเกลลิ่งแล้ว

$\alpha_t(i)$ แทนเวอร์ชันของ α ก่อนการสเกลลิ่ง

เมื่อ $t = 1$ จะได้ $\alpha_t(i) = c_t \alpha_t(i)$

เมื่อ $c_t = \frac{1}{\sum_{i=1}^{N} \alpha_t(i)}$

เมื่อ $2 \leq t \leq T$ คำนวณ $\alpha_t(i)$ จาก สมการ 2.16 ในเทอมของ $\alpha_t(i)$ ค่าก่อน

$$\alpha_t(i) = \sum_{j=1}^{N} \hat{\alpha}_{t-1}(j) a_{ji} b_i(O_t) \qquad (2.18)$$

เมื่อ สัมประสิทธิ์ การสเกลลิ่ง เป็น

$$c_t = \frac{1}{\sum_{i=1}^{N} \hat{\alpha}(i)}$$

เมื่อให้ $\alpha_t(i) = c_t \alpha_t(i)$

$$\alpha_t(i) = \frac{\sum_{j=1}^{N} \hat{\alpha}_{t-1}(j) a_{ji} b_j(O_t)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \hat{\alpha}_{t-1}(j) a_{ji} b_j(O_t)}$$

และโดยการเหนี่ยวนำ จะได้

$$\alpha_{t-1}(j) = (\prod_{i=1}^{t-1} c_\tau) \alpha_{t-1}(i)$$

จะได้ว่า

$$\alpha_t(i) = \frac{\sum_{i=1}^{N} \alpha_{t-1}(i) (\prod_{\tau=1}^{t-1} c_\tau) a_{ij} b_i(O_t)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_{t-1}(j) (\prod_{\tau=1}^{t-1} c_\tau) a_{ij} b_i(O_t)} = \frac{\alpha_t(i)}{\sum_{i=1}^{N} \alpha_t(i)}$$

นั่นคือจะสเกล $\alpha_t(i)$ ได้โดยหารด้วยผลรวมของ $\alpha_t(i)$ ทั้งหมด และสเกล $B_t(i)$ ด้วยค่าเดียวกันนี้ ในเทอมของการสเกลนี้สมการ (2.17) จะเป็น

$$\overline{a_{ij}} = \frac{\sum_{t=1}^{T-1} \hat{\alpha}_t(i) a_{ij} b_j(O_{t+1}) \hat{\beta}_{t+1}(j)}{\sum_{t=1}^{T-1} \sum_{j=1}^{N} \hat{\alpha}_t(i) a_{ij} b_j(O_{t+1}) \hat{\beta}_{t+1}(j)} \qquad (2.19)$$

โดยแต่ละ $\alpha_t(i).\hat{\beta}_{t+1}(j)$ จะได้เป็น

$$\alpha_t(i) = \left[\prod_{s=1}^{t} c_s\right] \alpha_t(i) = C_t \alpha_t(i)$$

$$\hat{\beta}_{t+1}(j) = \left[\prod_{s=1}^{T} c_s\right] \beta_{t+1}(j) = D_{t+1} \beta_{t+1}(j)$$

ดังนั้นสมการ (2.19) จะเขียนได้เป็น

$$\overline{a_{ij}} = \frac{\sum_{t=1}^{T-1} C_t \alpha_t(i) a_{ij} b_j(O_{t+1}) D_{t+1} \beta_{t+1}(j)}{\sum_{t=1}^{T-1} \sum_{j=1}^{N} C_t \alpha_t(i) a_{ij} b_j(O_{t+1}) D_{t+1} \beta_{t+1}(j)} \qquad (2.20)$$

ซึ่งเทอม C_tD_{t+1} จะเขียนได้ในเทอม

$$C_t D_{t+1} = \prod_{s=1}^{t} c_s \prod_{s=t+1}^{T} c_s = \prod_{s=1}^{T} c_s = C_T$$

ซึ่งไม่ขึ้นกับเวลา t ดังนั้น C_tD_{t+1} จะถูกตัดทิ้ง ทั้งเศษและส่วนของสมการ (2.20) ซึ่งทำให้ได้สูตรการคำนาณซ้ำ ๆ (reestimate) เดิมกลับคืนมา

กระบวนการ สเกลลิ่ง ดังกล่าานี้สามารถใช้ได้กับสัมประสิทธิ์ β และ π ในการสเกลลิ่งนี้จะทำให้การคำนาณค่า $P(O|\lambda)$ เปลี่ยนไป เราจะไม่สามารถหาได้จากการรวมกับของเทอม $\hat{}_T(i)$ แต่จะหาจากคุณสมบัติ

$$\prod_{t=1}^{T} c_t \sum_{i=1}^{N} \alpha_T(i) = C_T \sum_{i=1}^{N} \alpha_T(i) = 1$$

ดังนั้นจะได้

$$\prod_{t=1}^{T} c_t P(O|\lambda) =$$

$$P(O|\lambda) = \frac{1}{\prod_{t=1}^{T} c_t}$$

ทำให้อยู่ในรูป log ของ P เพื่อไม่ให้เกิน dinamic range ของคอมพิวเตอร์

$$\log[P(O|\lambda)] = \sum_{t=1}^{T} \log c_t$$

2.3.2.5 ลำดับของค่าปรากฏหลายเหตุการณ์ (Multiple Observation Sequence)

ในการใช้แบบจำลองแบบ Left-Right นั้น การเทรนแบบจำลองต้องใช้หลาย ๆ เหตุการณ์ของลำดับค่าปรากฏเข้ามาเทรน เพื่อให้ได้ค่าพารามิเตอร์ที่ถูกต้องมากขึ้น

ถ้าให้เซตของ v ลำดับค่าปรากฏเป็น

$O=[O^{(1)},O^{(2)},.....,O^{(k)}]$

เมื่อ $O^{(v)}=O_1^{(v)}O_2^{(v)}.....O_{Tv}^{(v)}$ เป็นลำดับค่าปรากฏของเหตุการณ์ที่ v โดยให้แต่ละเหตุการณ์ เป็นอิสระต่อกัน จะได้

$$P(O|\lambda)=\prod_{v=1}^{V}P$$

$$=\prod P_v$$

นำเอาจำนวนเหตุการณ์ของการเกิดค่าปรากฏแต่ละเหตุการณ์มารวมกันจะได้สูตรหา a_{ij},$b_j(k)$ เป็น

$$\bar{a}_i=\frac{\sum_{v=1}^{V}\frac{1}{P_v}\sum_{t=1}^{T_v-1}\alpha_t^v(i)a_ij(O_{t+1}^v)\beta_{t+1}^v(j)}{\sum_{v=1}^{V}\frac{1}{P_v}\sum_{t=1}^{T_v-1}\alpha_t^v(i)\beta_t^v(i)}$$

$$\bar{b}_i(k)=\frac{\sum_{v=1}^{V}\frac{1}{P_v}\sum_{t=1,O=k}^{T_v-1}\alpha_t^v(i)\beta_t^v(i)}{\sum_{v=1}^{V}\frac{1}{P_v}\sum_{t=1}^{T_v-1}\alpha_t^v(i)\beta_t^v(i)}$$

ส่วน π_iไม่ต้องคำนวณเนื่องจาก $\pi_1=1$□, $\pi_i=0,i\neq1$

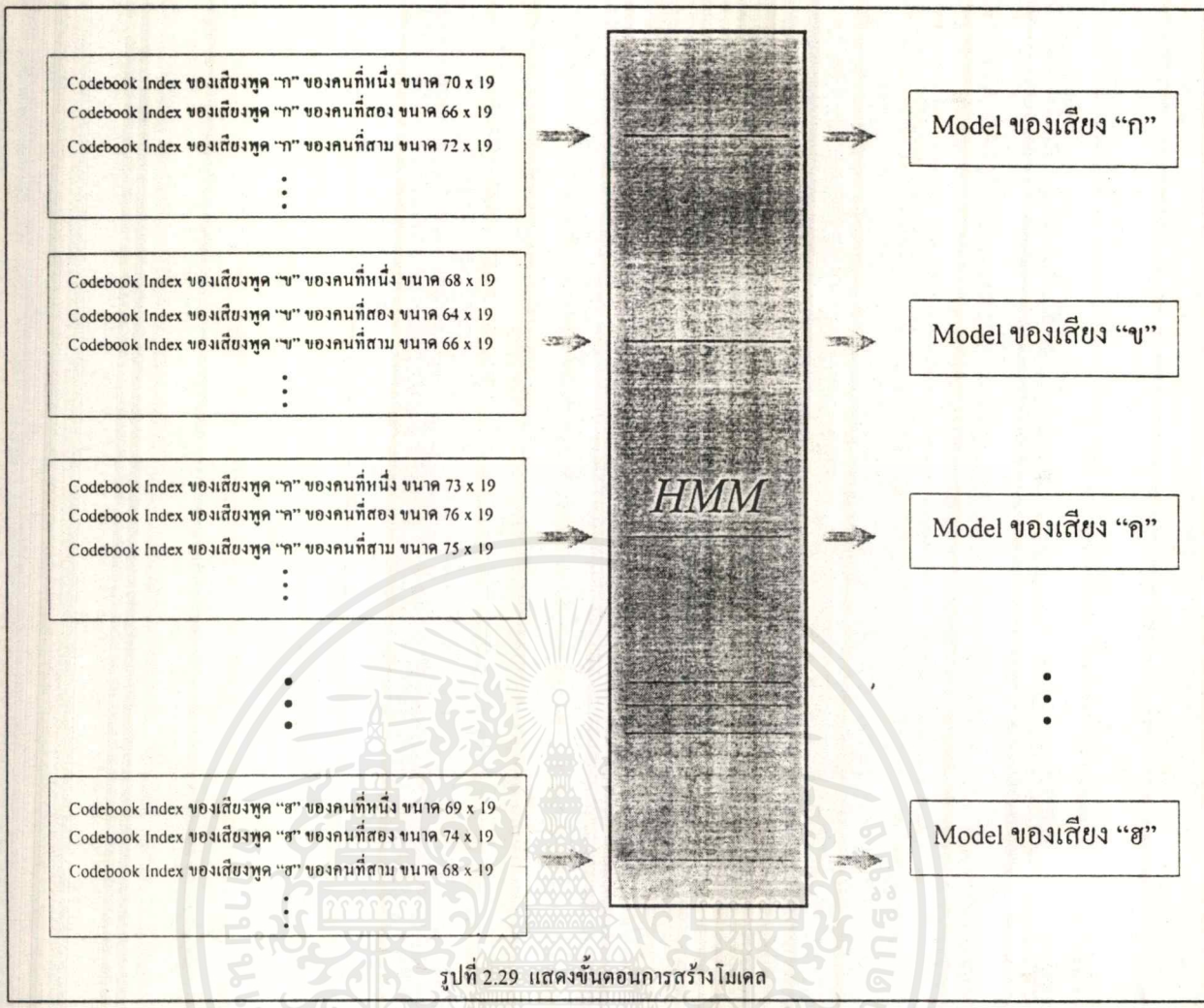
จะได้การสเกลลิ่ง ที่เหมาะสมของสมการ(5.55)-(5.56)

$$\bar{a}_i=\frac{\sum_{v=1}^{V}\sum_{t=1}^{T_v-1}\hat{\alpha}^v a_ij(O_{t+1}^v)\hat{\beta}_{t+1}^v(j)}{\sum_{v=1}^{V}\sum_{t=1}^{T_v-1}\sum_{j=1}^{N}\hat{\alpha}_t^v(i)a_ij(O_{t+1}^v)\hat{\beta}_{t+1}^v(j)}$$

$$\bar{b}_j(k)=\frac{\sum_{v=1}^{V}\sum_{t=1,O_t=k}^{T_v-1}\hat{\alpha}^v a_ij(O_{t+1}^v)\hat{\beta}_{t+1}^v(j)}{\sum_{v=1}^{V}\sum_{t=1}^{T_v-1}\sum_{j=1}^{N}\hat{\alpha}_t^v(i)a_ij(O_{t+1}^v)\hat{\beta}_{t+1}^v(j)}$$

2.3.2.5 ขั้นตอนการสร้างโมเดล

การสร้างโมเดลโดยวิธี HMM นั้นเป็นการนำทฤษฎีความน่าจะเป็นมาอธิบายการเกิดของตัวแปร 2 ตัว คือ สถานะ หรือ State ซึ่งจะเก็บข้อมูลการเปล่งเสียงส่วนหนึ่งของพยางค์ที่เวลาต่างๆ ไว้

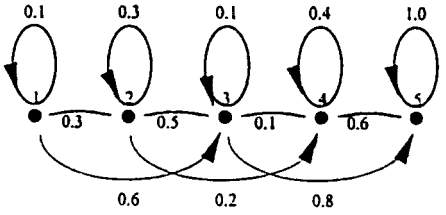


รูปที่ 2.29 แสดงขั้นตอนการสร้างโมเดล

และ ค่าปรากฏ คือ sequance ของค่า Index ของเสียงที่ได้จากขั้นตอน VQ ที่เวลาต่างๆ ผู้สังเกตจะเห็นเพียงค่าปรากฏของแต่ละ state แต่จะไม่ทราบแน่ชัดว่าอยู่ที่ state ใด จึงเรียกว่า Hiddan Markov Models อาจเปรียบเทียบกับการโยนเหรียญ สมมุติว่าโยนเหรียญ 3 เหรียญ แล้วออก "ก้อย" 1 เหรียญ ในที่นี้เหรียญ 3 เหรียญก็คือ จำนวน state 3 state และค่าปรากฏก็คือ "ก้อย" เราทราบเพียงว่าออก "ก้อย" แต่ไม่ทราบว่าเหรียญใดแน่ที่ออก "ก้อย"

การใช้วิธี HMM สามารถบอกได้ว่าค่าปรากฏนั้นอยู่ที่ state ใดโดยการใช้เทคนิคความน่าจะเป็นเข้ามาช่วย โมเดลที่สร้างจาก HMM จะประกอบด้วยข้อมูลของสถานะต่าง ๆ ที่เชื่อมเข้าหากันโดยใช้เส้นที่หมายถึงการเปลี่ยนแปลงสถานะของข้อมูลความน่าจะเป็น เพื่อใช้บอกว่าสถานะใดจะเกิดต่อจากสถานะปัจจุบัน

ชนิดของโมเดลที่ใช้เป็นแบบ Left - Right Model ซึ่งมีลักษณะการย้าย state จากซ้ายไปขวา และที่ state ใดๆ จะสามารถย้ายไป state ถัดไปได้มากสุด 2 state ดังแสดงในรูปที่ 2.30



รูปที่ 2.30 แสดงโมเดลของเสียงๆ หนึ่ง

จากรูปตัวอย่างโมเดลของเสียงนี้ จะประกอบด้วยสเตท 5 สเตท แต่ละเส้นของการย้าย state จะกำหนดด้วยค่าความน่าจะเป็นค่าหนึ่ง ซึ่งแต่ละโมเดลจะมีค่าต่างกัน ที่สเตทใดๆค่าความน่าจะเป็นรวมของการย้ายสเตทจะเท่ากับ 1

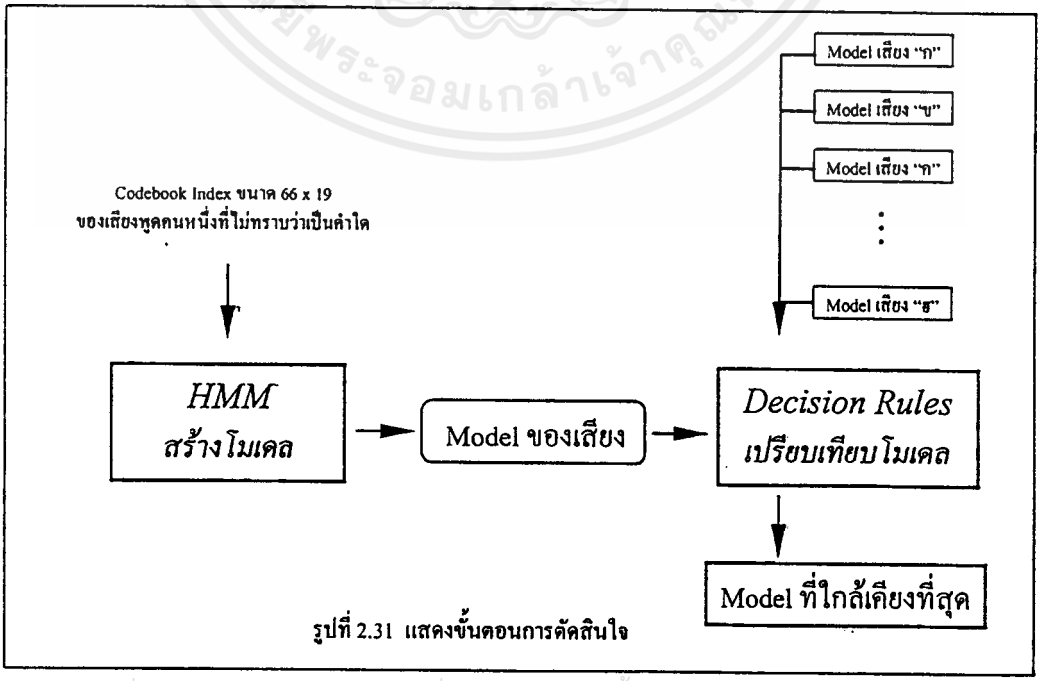
2.3.3 Decision rule

ดังที่ได้กล่าวไว้ในส่วนการแก้ปัญหาข้อที่ 2 ของ HMM โดยใช้วิธีตาม Viterbi algorithm

2.3.3.1 Viterbi Algorithm

ใช้ในการระยะทางที่สั้นที่สุดจากระยะทางที่เป็นไปได้ ของโมเดลที่มีอยู่กับโมเดลของเสียงที่เข้ามา โดยเลือกโมเดลที่มีความเป็นไปได้ในการเกิดเหตุการณ์ ถ้าโมเดลของคำใดมีค่าความน่าจะเป็นสูงกว่าโมเดลของคำอื่นๆ จะแสดงว่าคำที่ไม่ทราบว่าเป็นเสียงใด ก็คือคำนั้นนั่นเอง

ขั้นการเปรียบเทียบโมเดลเป็นขั้นที่นำเสียง Input ที่ไม่ทราบว่าเป็นคำใด เข้ามาผ่านกระบวนการสร้างโมเดลเฉพาะของเสียงนั้น แล้วนำโมเดลที่ได้นี้ไปเปรียบเทียบกับโมเดลของเสียงทุกเสียงที่ได้เก็บไว้แล้ว



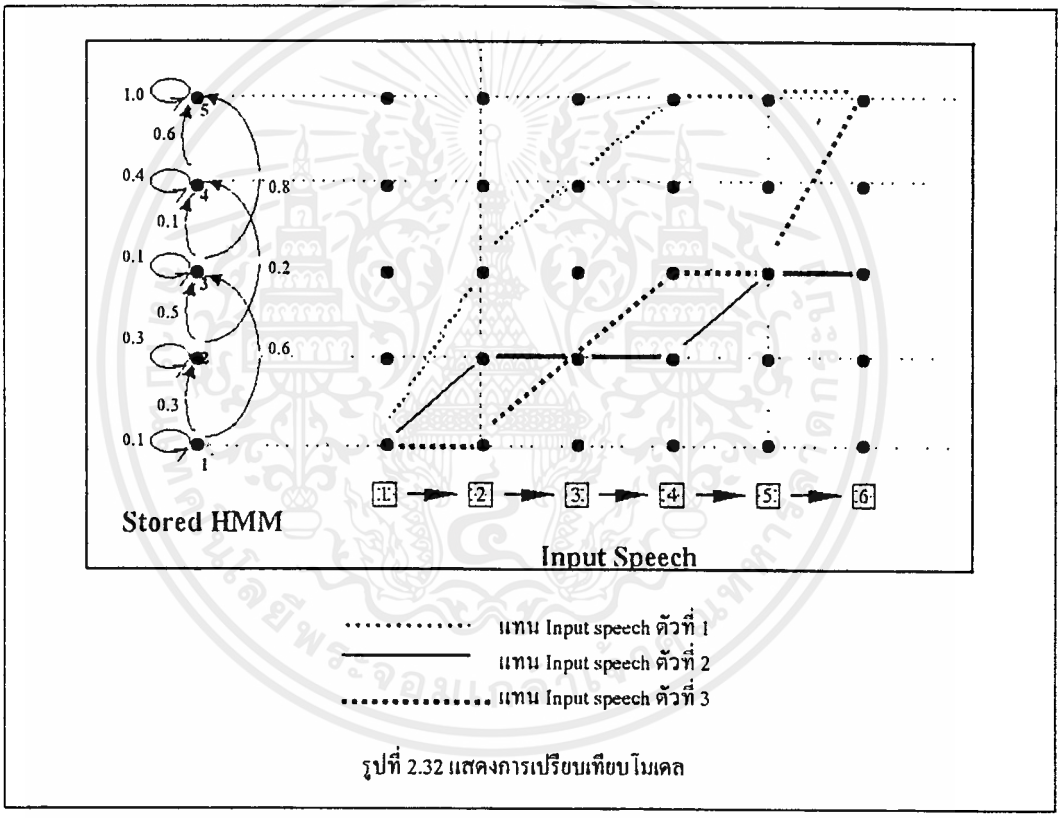
รูปที่ 2.31 แสดงขั้นตอนการตัดสินใจ

การเปรียบเทียบโมเดลนั้นจะใช้วิธี Viterbi Algorithm (VA) ซึ่งจะเลือกเส้นทางที่สั้นที่สุดของโมเดลเสียงที่ไม่ทราบกับโมเดลที่เก็บไว้ทุกตัว ผลของการเลือกจะทำให้ทราบว่าเสียงนั้นเป็นเสียงใด

ความสั้นยาวของเส้นทางที่กล่าวถึงก็คือความน่าจะเป็นของการเปลี่ยนสเตทนั่นเอง ค่าปรากฏที่เกิดจากโมเดลของเสียงที่ไม่ทราบค่าจะเป็นตัวกำหนดสเตทที่เกิดขึ้นในแต่ละช่วงเวลา

การตัดสินใจว่าเสียงนั้นตรงกับโมเดลใดได้มาจากการเปรียบเทียบค่าที่น้อยที่สุดของ ' *ความยาวรวมของเส้นทางที่ได้จากสเตททั้งหมด* ' ที่เปรียบเทียบกับแต่ละโมเดล

ตัวอย่างนี้มี Input Speech 3 ตัวเทียบกับแบบจำลอง HMM 1 ตัว Input ตัวที่มีระยะทางห่างจากแบบจำลองนี้น้อยที่สุด ก็จะถือว่าเป็นคำนี้



รูปที่ 2.32 แสดงการเปรียบเทียบโมเดล

2.4 หลักการตัดเสียงให้เป็นคำ เดี่ยว

หลักการตัดเสียงให้เป็นคำเดี่ยวเพื่อใช้ในการวิเคราะห์เสียง จากหลักการของการวิเคราะห์เสียงจำเป็นที่จะต้องเป็นคำเดี่ยวเท่านั้นจึงจะทำให้การจำเสียงได้ดี ดังนั้นเราจึงจำเป็นที่จะต้องแยกเสียงที่เข้ามาให้เป็นคำเดี่ยวก่อน จากหลักการประมวลผลในเชิงเวลานั้นคือ การวิเคราะห์สัญญาณของเสียงพูดโดยตรงจากรูปคลื่นสัญญาณ (wave form) ซึ่งแตกต่างจากการประมวลผลเชิงความถี่ (Frequency-Domain) ตัวอย่างของการประมวลผลของสัญญาณเสียงพูดเชิงเวลา เช่น การหาค่าอัตราการตัดศูนย์เฉลี่ย (Zero-

crossing Rate) , การหาค่าพลังงาน (Energy) และค่าออโตคอร์รีเลชั่น (Autocorrelation) โดยการประมวลผลหาค่าเหล่านี้เป็นไปโดยง่ายมีขั้นตอนที่ไม่ซับซ้อน แต่ให้องค์ประกอบสำคัญของสัญญาณเสียง แต่จากการศึกษาในการคำนวณหาคุณลักษณะของเสียงที่ใช้ในการตัดคำแล้ว เราจะเห็นได้ว่าวิธีการใช้การหาค่าพลังงานของเสียง นั่นคือการหาขนาดกำลังสองของสัญญาณเฉลี่ยในช่วงเวลาสั้นๆ เป็นวิธีการที่สามารถทำการตัดคำได้ดี และยังใช้เวลาในการคำนวณน้อยเมื่อเปรียบเทียบกับการหาค่าอื่นๆ นั่นจึงทำให้การหาค่าพลังงานสามารถที่จะนำมาใช้งานจริงได้ โดยจะมีหลักการที่ใช้ศึกษาดังนี้

2.4.1 การประมวลผลเสียงพูดโดยขึ้นกับเวลา

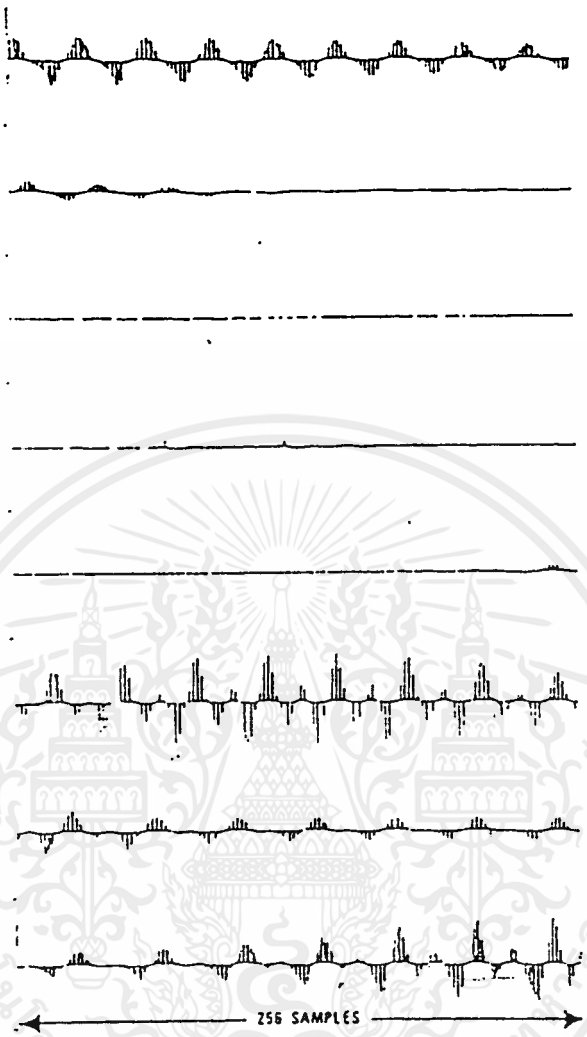
จากรูปที่ 2.33 ซึ่งเป็นสัญญาณเสียงพูดที่มีอัตราการสุ่มตัวอย่างของสัญญาณเป็น 8000 ค่าใน 1 วินาที จะเห็นได้ชัดว่าคุณสมบัติของสัญญาณเสียงพูดจะมีลักษณะที่เปลี่ยนแปลงไปตามเวลา ตัวอย่างเช่น การเปลี่ยนแปลงสัญญาณเสียงพูดระหว่างเสียงที่เป็นเสียงก้องหรือเสียงโฆษะ(voice) และเสียงไม่ก้องหรือเสียงอโฆษะ(unvoice) โดยที่เสียงจะมีการเปลี่ยนแปลงที่เห็นได้ชัดจากขนาด(magnitude) ของสัญญาณ และยังมีการเปลี่ยนแปลงของความถี่มูลฐานของเสียง(fundamental frequency) ภายในช่วงที่เป็นโฆษะด้วย โดยการที่แสดงรูปแบบสัญญาณเป็นรูปคลื่นนี้สามารถทำให้ง่ายต่อการสังเกตถึงองค์ประกอบและคุณลักษณะของเสียงนั้น เช่น ความเข้มของเสียง(Intensity), ชนิดของเสียง(เสียงก้องหรือเสียงไม่ก้อง) หรือ excitation mode,ความถี่มูลฐาน(pitch หรือ fundamental frequecy) และอาจรวมถึงสัมประสิทธิ์ในอวัยวะกำเนิดเสียง เช่น ความถี่กำทอน(resonant หรือ formant frequency)

สมมติฐานที่ใช้ในการประมวลผลสัญญาณเสียงส่วนมากก็คือการที่ถือเอาว่าคุณลักษณะของเสียงนั้นมีการเปลี่ยนแปลงช้ามากเมื่อเทียบกับเวลา และสมมติฐานนี้ทำให้เกิดการประมวลผลที่เรียกว่าการประมวลผลแบบช่วงเวลาสั้นๆ(short time) โดยที่ส่วนเล็กๆแต่ละส่วนของเสียงพูดนั้นที่ถูกแยกออกมาและจะถูกประมวลผลโดยแต่ละส่วนของเสียงพูดนี้เรียกว่าการวิเคราะห์เฟรม(analysis frames) ซึ่งเฟรมแต่ละเฟรมจะมีส่วนที่ซ้อนทับกัน และผลลัพธ์ที่ได้ของแต่ละส่วนนั้นอาจเป็นตัวเลขตัวเดียว หรืออาจเป็นกลุ่มของตัวเลขก็ได้ ดังนั้นกระบวนการนี้ก็จะทำให้เกิดลำดับของค่าใหม่ในเชิงของเวลา ซึ่งแสดงถึงลักษณะของสัญญาณเสียง

กระบวนการวิเคราะห์เสียงแบบช่วงเวลาสั้นๆ นั้นเกือบทั้งหมดสามารถแสดงในรูปของสมการทางคณิตศาสตร์ที่มีรูปแบบ คือ

$$Q_n = \sum_{m=-\infty}^{\infty} T[X(m)]W(n-w)$$

สมการนี้สัญญาณเสียงจะถูกกรองเอาเฉพาะย่านความถี่ที่ต้องการและจะถูกกระทำโดยฟังก์ชั่น T[ ] ซึ่งอาจเป็นเชิงเส้นหรือไม่ขึ้นอยู่กับสัมประสิทธิ์ที่จะทำการคำนวณ และหลังจากนั้นลำดับที่ได้จะถูกคูณด้วยลำดับวินโดว์ ในตำแหน่งที่สัมพันธ์กับเวลาของตัวอย่างสัญญาณที่ n และผลคูณก็จะถูกรวมกัน โดยทั่วไปแล้วผลลัพธ์จะเกิดจากลำดับที่จำกัด



รูปที่2.33 แสดงรูปสัญญาณเสียงที่มีอัตราสุ่มเป็น 8 kHz

2.4.2 ค่าขนาดกำลังสองของสัญญาณเฉลี่ยในช่วงเวลาสั้นๆ

จากรูปที่ 2.33 เราสามารถสังเกตได้ว่าค่าขนาดของสัญญาณ(Amplitude) นั้นจะมีการเปลี่ยนแปลงไปตามเวลา โดยเฉพาะขนาดของสัญญาณของเสียงไม่ก้อง มักจะมีค่าต่ำกว่าขนาดของสัญญาณของเสียงก้องมาก ซึ่งการใช้ค่ากำลังสองของสัญญาณเฉลี่ยในช่วงสั้นๆ สามารถแสดงให้เห็นถึงความแตกต่างนี้ได้อย่างชัดเจนได้มากกว่าการใช้ขนาดโดยมีรูปแบบสมการของการใช้ในการคำนวณ คือ

$$E_n = \sum_{m=-\infty}^{\infty} [X(m)W(n-m)]^2$$

หรือสามารถเขียนได้เป็น

$$E_n = \sum_{m=-\infty}^{\infty} X^2(m)h(n-m)$$

โดยที่

$$h(n) = W^2(m)$$

สมการนี้สามารถแสดงเป็นบล็อคไดอะแกรมได้ดังรูปที่ 2.34 สัญญาณ X2(m) จะถูกกรองโดยตัวกรองแบบเชิงเส้น h(n) ก็คือวินโดว์ (window) ลองมาดูว่าวินโดว์ที่แตกต่างกันจะมีผลอย่างไรกับค่าพลังงานผลลัพธ์ที่ได้ โดยถ้า h(n) ให้ค่าขนาดของสัญญาณคงที่ในช่วงเวลาที่ยาวนานแล้วค่า E จะมีการเปลี่ยนแปลงน้อยมาก เนื่องจากเราต้องการให้ค่ากำลังสองในช่วงเวลาสั้นๆ แสดงให้เห็นถึงการเปลี่ยนแปลงของขนาดเสียงในสัญญาณเสียง ดังนั้นจึงต้องใช้วินโดว์ที่มีช่วงเวลาสั้นเกินไปก็จะทำให้ไม่ได้ค่าที่มีความเรียบพอเพียง โดยตัวฟังชั่นก์วินโดว์มีรูปแบบดังนี้

วินโดว์แบบสี่เหลี่ยมผืนผ้า $h(n) = 1$ เมื่อ

$$0 \le n \le N-1$$

$h(n) = 0$ อื่น ๆ

วินโดว์แบบแฮมมิ่ง

$$h(n) = 0.54 - 0.46\cos\left(\frac{2n\pi}{(N-1)}\right)$$

เมื่อ

$$0 \le n \le N-1$$

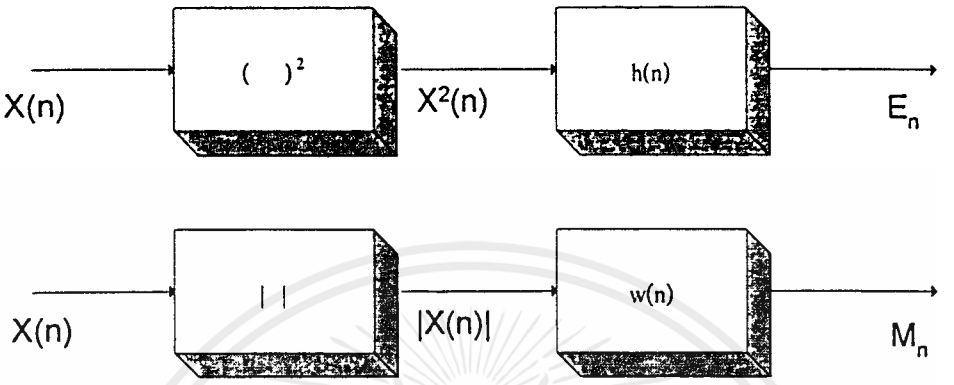
$h(n) = 0$ อื่น ๆ

จากสมการจะเห็นได้ว่า วินโดว์แบบสี่เหลี่ยม นั้นจะมีการถ่วงน้ำหนัก(weight) ค่าเท่ากันตลอดทุกสัญญาณ(Sample) ในช่วง(n-N+1)ถึง n ส่วนผลตอบสนองทางความถี่ของวินโดว์แบบสี่เหลี่ยมนั้นจะได้ตามสมการดังนี้

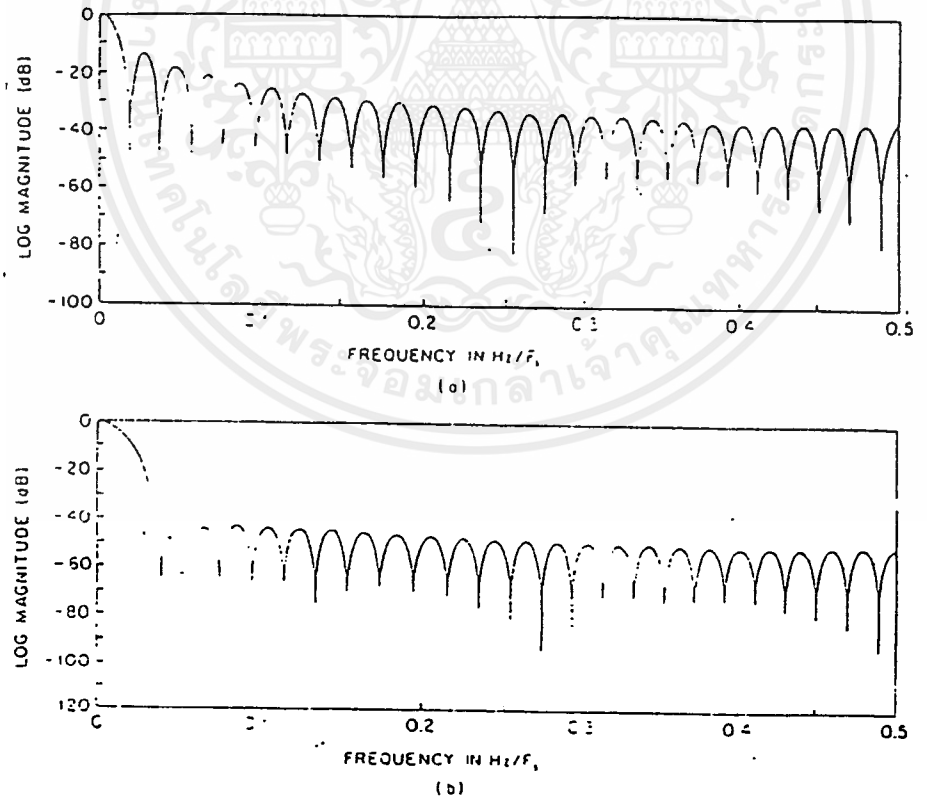
$$H\left(e^{j\Omega T}\right) = \frac{\sin(\Omega NT/2)}{\sin(\Omega T/2)} e^{-j\Omega T(N-1)/2}$$

ส่วนแฮมมิ่งวินโดว์ นั้นจะใช้ประโยชน์ได้ดีสำหรับการวิเคราะห์เชิงความถี่ เพราะให้ค่าลดทอนภายนอกย่านมากกว่าดังรูปที่ 2.35

จากที่กล่าวมาแล้วว่าถ้าค่า N ในวินโดว์มีค่าน้อยเกินไปก็จะมีการเปลี่ยนแปลงของค่า En มากเกินไป แต่ถ้าค่า N มากเกินไปก็จะทำให้สูญเสียคุณสมบัติของเสียงไป แต่เป็นการยากที่จะกำหนดค่า N เพียงค่าเดียวให้เหมาะสมกับเสียงทั้งหมด เนื่องจากคาบเวลาของเสียงที่เรียกว่าพิทช์(pitch) จะแตกต่างกันเช่น เด็กหรือ ผู้หญิงอาจมีค่าเพียง 20 ตัวอย่าง ที่อัตราการสุ่มตัวอย่าง 10kHz แต่ผู้ชายอาจสูงถึง 250 ตัวอย่าง แต่ค่า N ที่ใช้กันนั้นอยู่ในช่วงประมาณ 100-200 ตัวอย่างที่อัตราการสุ่ม 10kHz (10-20msec)

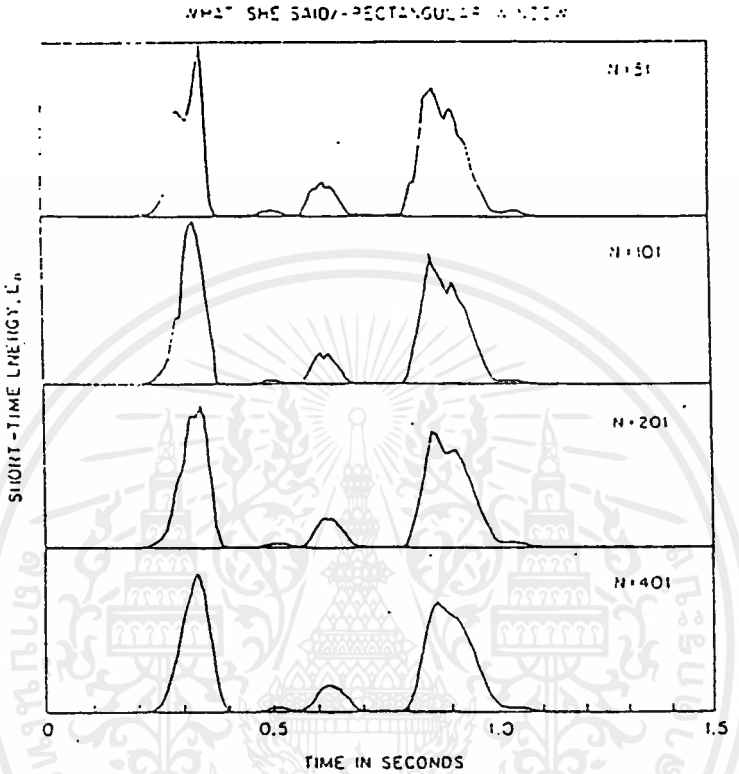


รูปที่ 2.34 แสดงบล็อกไดอะแกรม (a) ขนาดกำลังสองในช่วงเวลาสั้น(b) ขนาดในช่วงเวลาสั้น ๆ



รูปที่ 2.35 แสดงการแปลงฟูเรียร์ของ (a) วินโดว์สี่เหลี่ยม (b) วินโดว์แบบแฮมมิ่ง

จากรูปที่ 2.36 จะแสดงให้เห็นผลจากการเปลี่ยนแปลงค่า N ในวินโดว์ ซึ่งจะสังเกตได้ชัดเจนว่าเมื่อค่า N เพิ่มก็จะทำให้ค่าขนาดกำลังสองในช่วงเวลาสั้น ๆ มีผลออกมามีความราบเรียบมากยิ่งขึ้น



รูปที่ 2.36 แสดงค่าขนาดกำลังสองช่วงเวลาสั้นๆสำหรับวินโดว์แบบสี่เหลี่ยมเมื่อเปลี่ยนค่า N

สิ่งสำคัญที่ได้จากค่า E_nนั้นคือ การที่เราสามารถแยกแยะส่วนของเสียงก้อง และเสียงไม่ก้อง ออกจากกันได้โดยส่วนของเสียงที่ไม่ก้อง นั้นจะมีค่า E_n ที่ต่ำกว่าเสียงก้องอย่างเห็นได้ชัด ซึ่งจะทำให้สามารถทำการกำหนดจุดอย่างคร่าวๆได้ว่าตำแหน่งไหนเป็นจุดที่เสียงได้ทำการเปลี่ยนจากเสียงก้องหรือในทางกลับกันและสัญญาณที่มีค่า S/N สูงๆนั้นค่า E_n จะทำให้เราสามารถแยกเสียงพูดออกจากเสียงเงียบ(silence)ได้ด้วย

2.4.3 การแยกเสียงพูดออกจากเสียงเงียบ(silence) โดยใช้ค่าขนาดกำลังสองและค่าอัตราการตัดศูนย์เฉลี่ย

ในระบบการประมวลผลสัญญาณเสียงพูดนั้นสิ่งสำคัญสิ่งหนึ่งคือ การแยกแยะและการหาขอบเขตทั้งจุดเริ่มต้นและจุดสิ้นสุดของเสียงออกจากสัญญาณรบกวน (Background Noice) โดยเฉพาะในกระบวนการรู้จำเสียงพูดชนิดคำโดด (Isolated Word) จำเป็นอย่างยิ่งที่จะต้องกำหนดจุดเริ่มต้นและจุดสิ้นสุดของคำให้ได้อย่างมีประสิทธิภาพ

การใช้ค่าพลังงานและอัตราการตัดศูนย์เฉลี่ย จากในรูปที่ 2.37 เป็นตัวอย่างแรก จะเห็นได้ว่ารูปคลื่นนี้สามารถแยกแยะเสียงออกจากสิ่งแวดล้อมได้ง่ายโดยใช้ค่าขนาดกำลังสองส่วนอีกตัวอย่างหนึ่งในรูปที่ 2.38 เป็นเสียงที่ค่าพลังงานของเสียงใกล้เคียงกับสัญญาณรบกวนมากไม่สามารถแยกแยะออกได้ แต่เมื่อมาดูที่ค่าอัตราการศูนย์เฉลี่ยจะมีความแตกต่างกับสัญญาณรบกวนมากพอสมควร

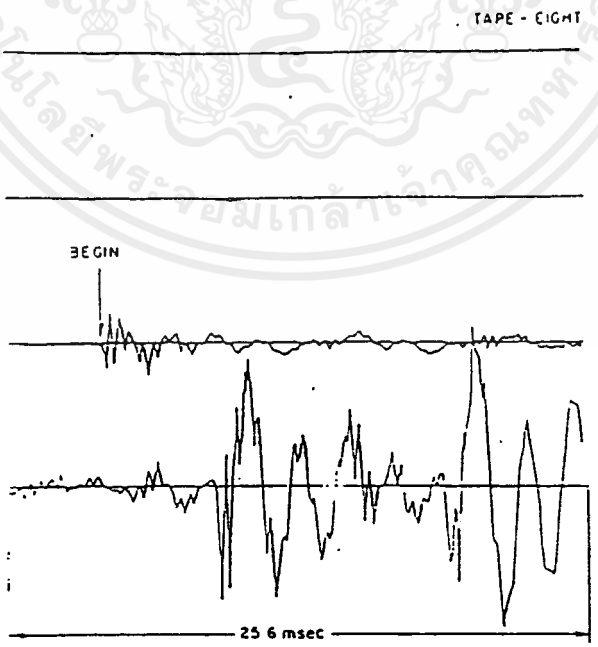
จากรูปที่ 2.39 เป็นตัวอย่างที่ยากมากที่จะทำการกำหนดจุดเริ่มต้นและสิ้นสุดของเสียง ซึ่งจุดเริ่มต้นของเสียงที่มีลักษณะพ่นออกมา(Fricative) ซึ่งจะทำให้ค่าพลังงานมีค่าต่ำมาก ซึ่งลักษณะของเสียงที่ยากต่อการกำหนดขอบเขต มีดังเช่น

1.ลักษณะของเสียงที่พ่นลมออกมา ณ จุดเริ่มต้นหรือสิ้นสุด (Weak fricative)

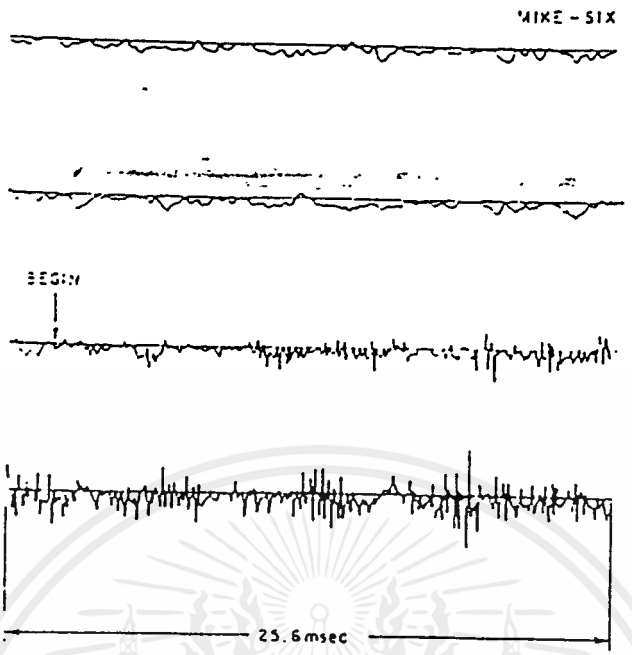
2.ลักษณะของเสียงที่เป็นการระเบิดออกมา(Weak plosive bursts)

3.ลักษณะของเสียงที่เป็นเสียงนาสิก(Nasals)

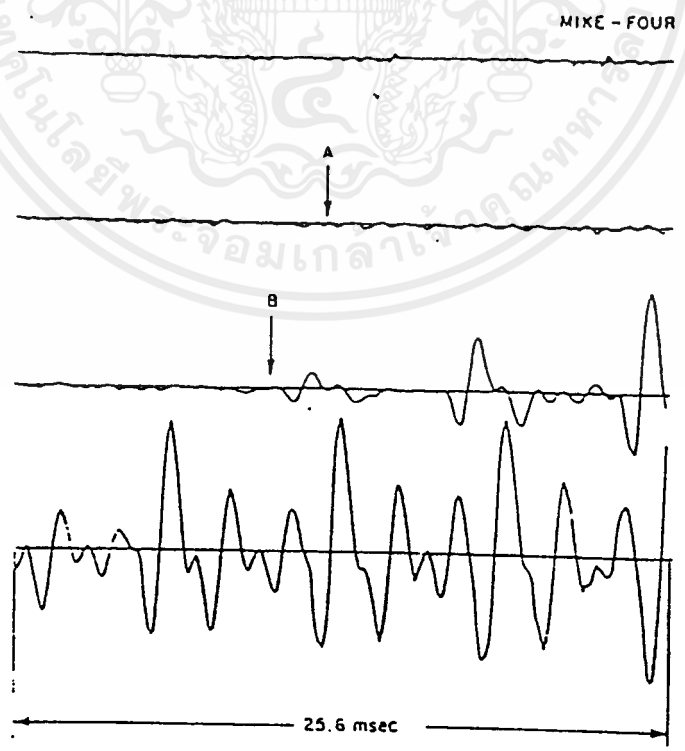
ในการที่จะกำหนดจุดเริ่มต้นและสิ้นสุดของเสียงเหล่านี้ เราสามารถใช้กระบวนการของการหาค่าขนาดกำลังสอง และอัตราการตัดศูนย์เฉลี่ยร่วมกันได้ ตัวอย่างในระบบรู้จำเสียงพูดแบบคำโดด การบันทึกเสียงจะบันทึกส่วนที่เป็นเสียงพูดและส่วนที่ผู้พูดไม่ได้พูดไว้ด้วย แล้วใช้ระบบหาจุดเริ่มต้นและสิ้นสุดของคำเพื่อที่จะทำให้ผ่านเข้ากระบวนการรู้จำเสียงโดยส่งเฉพาะส่วนที่เป็นเสียงพูดเข้าไปประมวลผลต่อไป



รูปที่ 2.37 แสดงรูปคลื่นของจุดเริ่มต้นเสียงคำว่า /eight/



รูปที่ 2.38 แสดงรูปคลื่นของจุดเริ่มต้นของเสียงคำว่า /six/



รูปที่ 2.39 แสดงรูปคลื่นของจุดเริ่มต้นเสียงคำว่า/four/

การทำงานของระบบนี้จะใช้การหาค่าอัตราการศูนย์เฉลี่ยที่มีขนาดเฟรมเท่ากับ 10 msec และการหาค่าพลังงาน และอัตราการตัดศูนย์เฉลี่ยร่วมกัน โดยใช้วินโดว์ขนาด 10 msec ซึ่งทั้ง 2 กระบวนการจะทำตลอดช่วงของสัญญาณด้วยอัตรา 100 ครั้งต่อวินาที โดยการสมมติให้ช่วงเวลา 100 msec แรกไม่มีเสียงพูด หมายความว่าค่าอัตราการตัดศูนย์ และค่าพลังงานในช่วงนี้จะเป็นค่าของสัญญาณรบกวน และทำการคำนวณค่าพลังงาน และอัตราการตัดศูนย์เฉลี่ยที่ใช้เป็นเกณฑ์ในการตัดสินใจค่าระดับของพลังงาน

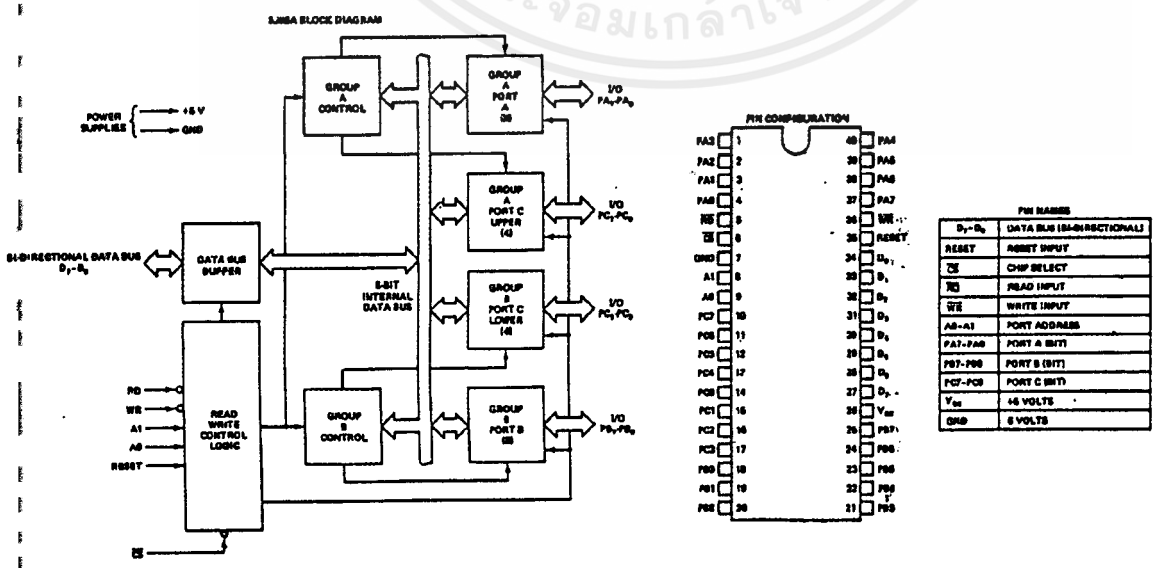
ค่าระดับที่ใช้ส่วนมากจะให้ค่าโดยประมาณซึ่งในความเป็นจริงแล้วจุดเริ่มต้น และจุดสิ้นสุดนั้นจะอยู่ภายนอกจากค่าระดับนี้จึงมีการทำซ้ำโดยดูจากจุดที่กำหนดโดยค่าระดับในครั้งแรกและลดลงจนถึงค่าระดับอีกค่าหนึ่งจึงถือได้ว่าเป็นจุดเริ่มต้นและจุดสิ้นสุดของคำ ขอบเขตที่ได้จากการใช้ค่าระดับค่าแรกนั้นเป็นที่น่าเชื่อถือว่าภายในขอบเขตนี้จะไม่มีจุดเริ่มต้นหรือจุดสิ้นสุดของคำอยู่

กระบวนการถัดไปคือ การย้อนกลับจากจุดเริ่มต้นที่ได้จากค่าระดับค่าแรกแล้วทำการเปรียบเทียบค่าอัตราการตัดศูนย์เฉลี่ยกับค่าระดับการตัดศูนย์เฉลี่ยของมันซึ่งได้จากสัญญาณรบกวน โดยถ้าค่าอัตราการตัดศูนย์เฉลี่ยมีค่ามากกว่าค่าระดับเกิน 3 ครั้ง แล้วจะถือว่าจุดเริ่มต้นเป็นจุดที่มีค่าอัตราการตัดศูนย์เฉลี่ยเกินค่าระดับเป็นครั้งแรก

2.5 การอินเตอร์เฟซพื้นฐาน

เครื่องคอมพิวเตอร์ทุกรุ่นจะมีหมายเลขพอร์ตสำหรับใช้งานต่างๆ ดังตารางที่ 2.1 จะเห็นว่ามีบางพอร์ตที่ไม่ได้ถูกใช้งาน (สงวนไว้) เช่น 360-36F, 3C0-3CF เราสามารถที่จะนำหมายเลขพอร์ตเหล่านี้ไปประยุกต์ใช้งานได้ หรือจะใช้งานหมายเลขพอร์ตที่ถูกกำหนดไว้แล้ว แต่เครื่องไมโครคอมพิวเตอร์ของเราไม่ได้ต่ออุปกรณ์ใช้งานกับพอร์ตนั้น เช่น พอร์ตหมายเลข 278-27F (เครื่องพิมพ์ขนานพอร์ต 2) เพราะโดยทั่วไปจะนิยมต่อเครื่องพิมพ์ที่พอร์ตหมายเลข 3BC หรือ 378

2.5.1 8255 พอร์ตอินพุท/เอาท์พุทของระบบ (Programmable Peripheral Interface)



รูปที่ 2.40 แสดงแผนผังและการจัดขาของ 8255A-5

หมายเลขพอร์ต	การใช้งาน
000-01F	ตัวควบคุมดีเอ็มเอ 1, 8237A-5
020-03F	ตัวควบคุมอินเตอร์รัปต์ 1,8259 (มาสเตอร์)
040-05F	ตัวควบคุมไทเมอร์เคาน์เตอร์ 8254-2
060-06F	ตัวควบคุมพอร์ตขนานและคีย์บอร์ด 8042
070-07F	Real Time Clock, NMI ของระบบ
080-09F	ดีเอ็มเอเพจรีจิสเตอร์ 74LS612
0A0-0BF	ตัวควบคุมอินเตอร์รัปต์ 2,8259 (สเลฟ)
0C0-0DF	ตัวควบคุมดีเอ็มเอ 2, 8237A-5
0F0	เคลียร์ไมโครโปรเซสเซอร์
0F1	รีเซตไมโครโปรเซสเซอร์
0F8-0FF	ไมโครโปรเซสเซอร์ 80287
1F0-1FB	ฮาร์ดดิสก์
200-207	เกมอินพุท/เอาท์พุท
278-27F	เครื่องพิมพ์ขนาน พอร์ต 2
2F8-2FF	เครื่องพิมพ์อนุกรม พอร์ต 2
300-31F	การ์ดโปรโตไทป์ (protype)
360-36F	สงวนไว้
378-37F	เครื่องพิมพ์ขนาน พอร์ต 1
380-38F	SDLC ไบต์ซิงค์โครไนซ์ 1
3A0-3AF	ไบต์ซิงค์โครไนซ์ 1
3B0-3BF	อะแดปเตอร์โมโนโครม และเครื่องพิมพ์
3C0-3CF	สงวนไว้
3D0-3DF	อะแดปเตอร์สี/กราฟฟิก
3F0-3F7	ตัวควบคุมดิสก์ไดรฟ์
3F8-3FF	พอร์ตอนุกรม 1

ตารางที่ 2.3 แสดงการจัดตำแหน่งพอร์ตของระบบ

8255 เป็นชิปซับพอร์ตที่ทำหน้าที่เป็นพอร์ตขยาย สามารถรับส่งข้อมูลแบบขยายได้รวดเร็ว โดยมีพอร์ตให้ใช้งาน 3 พอร์ตด้วยกันคือ พอร์ต A,พอร์ต B และพอร์ต C นอกจากนั้นยังมีพอร์ตควบคุมอีก 1 พอร์ต ในพอร์ต A และพอร์ต B จะมีขนาด 8 บิท ส่วนพอร์ต C จะถูกแบ่งออกเป็น 4 บิทบนและล่าง โดย 4 บิทบน(PC4-PC7) จะถูกควบคุมด้วยพอร์ต A ส่วน 4 บิทล่าง (PC0 – PC3) จะถูกควบคุมโดยพอร์ต B

การใช้งานพอร์ตสามารถทำได้โดยการเขียนโปรแกรมควบคุมพอร์ตนั้นๆ สามารถจะใช้ภาษาเบสิก ภาษาซี ภาษาปาสคาล และแอสเซมบลี ก่อนอื่นจะต้องทราบหมายเลขพอร์ตที่ใช้งาน แล้วศึกษารายละเอียดของตัวอุปกรณ์ที่ทำงานอยู่ในพอร์ตนั้นว่ามีการทำงานอย่างไร จากนั้นจึงเขียนโปรแกรมควบคุมการทำงานของพอร์ตนั้น บนเครื่องไมโครคอมพิวเตอร์จะมีพอร์ตที่ทำหน้าที่ควบคุมการรับคีย์บอร์ด สแกนโค้ดเป็นพอร์ตขนาน แล้วยังผลิตเสียงได้ด้วย คือพอร์ตหมายเลข 060-06F พอร์ตนี้จะใช้ชิป 8255 ในการทำงาน

ขาสัญญาณ	หน้าที่
PA, PB, PC (Port A, B, C)	เป็นขาสัญญาณของพอร์ตทั้ง 3 ของ 8255 คือ พอร์ต A, พอร์ต B และพอร์ต C การเลือกใช้งาน 1 ใน 3 พอร์ต จะใช้แอดเดรส A0, A1 เลือกอีกทีหนึ่ง
CS (Chip Select)	สัญญาณเลือกชิป 8255 ก่อนจะทำการโปรแกรม ต้องให้สัญญาณนี้แอคทีฟ คือ เป็น 0 ด้วย
RD (Read)	เป็นสัญญาณอินพุท เพื่อใช้อ่านข้อมูลภายในพอร์ตของ 8255 สัญญาณนี้จะต้องแอคทีฟพร้อม CS
WR (Write)	เป็นสัญญาณอินพุท เพื่อใช้อ่านข้อมูลภายในพอร์ตของ 8255 สัญญาณนี้จะต้องแอคทีฟพร้อม CS
D0-D7 (Data Bus)	เป็นขาสัญญาณแบบไบไดเรกแชนเนล คือ 2 ทิศทาง สามารถใช้รับ/ส่งข้อมูลจากซีพียูได้
A0-A1 (Address)	เป็นสัญญาณอินพุทใช้งานร่วมกับสัญญาณ RD และ WR เพื่อเลือกและควบคุม 1 ใน 3 พอร์ต หรือคอนโทรลเวิร์ดรีจิสเตอร์
Reset	เป็นสัญญาณอินพุทใช้รีเซตแก่ 8255 เพื่อทำการเคลียร์สถานะต่าง ๆ ของ 8255 และทำให้พอร์ตทั้ง 3 เป็นอินพุททั้งหมด

ตารางที่ 2.4 หน้าที่และขาสัญญาณต่าง ๆ ของ 8255

2.5.2 การต่อบอร์ดโปรโตไทป์กับ PC

บอร์ดโปรโตไทป์ถูกออกแบบเอาไว้เพื่อให้ผู้ใช้สามารถทำการพัฒนาการเชื่อมต่อกับ PC โดยจะทำการติดต่อทางบัส(bus) โดยที่บอร์ดโปรโตไทป์จะมีการเชื่อมต่อกันได้มากถึง 64 พอร์ต โดยจะมี 32 พอร์ตอินพุท และ 32 พอร์ตเอาท์พุท โดยที่รายละเอียดของการเชื่อมต่อแต่ละพอร์ตมีดังนี้

A0 ถึง A19 ทั้ง 20 พินนี้จะเก็บไว้สำหรับหน่วยความจำและอินพุทและเอาท์พุทแอดเดรส โดยที่พอร์ต A0 จะเป็น LSB(least-significant bit) และพอร์ต A19 จะเป็น MSB(most-significant bit)

D0 ถึง D7 ทั้ง 8 พินนี้เป็นบัสข้อมูลแบบ 2 ทิศทาง โดยที่ D0 เป็น LSB และ D7 เป็น MSB

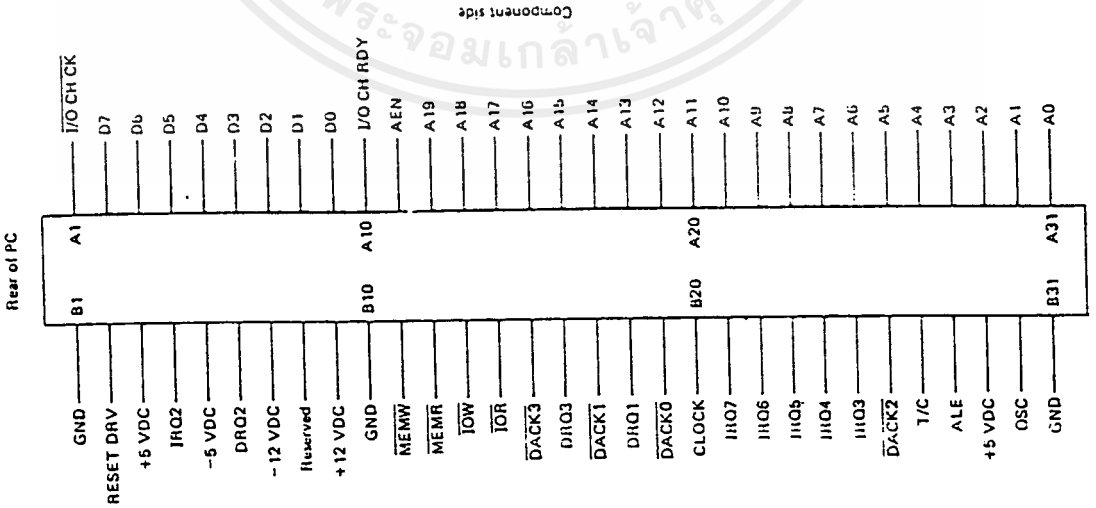
MEMR,MEMW,IOR,IOW เป็นพอร์ตที่ควบคุมการอ่านและเขียน

ALE(address latch enable) สำหรับบัสของระบบ ALE จะเป็นตัวชี้ของการเริ่มของไซเคิ้ลของบัส และเมื่อปรากฏสัญญาณขึ้นและบัสข้อมูลของระบบไม่มีการบรรจุข้อมูลของแอดเดรส ดังนั้นฟังก์ชั่นของ ALE จะไม่ถูกดีมัลติเพล็กซ์

AEN(address enable) สัญญาณนี้จะถูกกำหนดโดย ตัวควบคุมDMA เพื่อที่จะทำการชี้ว่าไซเคิ้ลของ DMA กำลังอยู่ในกระบวนการทำงาน โดยปกติจะใช้สำหรับยกเลิกการถอดรหัสของพอร์ตอินพุทและเอาท์พุท

OSC(oscillator),CLOCK OSC คือ คล็อกความเร็วสูงของระบบ ซึ่งจะมีคาบ 70 ns หรือ 14.31818 MHz หรือมีค่า 50% ของรอบการทำงาน ส่วน CLOCK จะมีความถี่ 1 ใน 3 ของความถี่ของ oscillator ซึ่งก็คือ 4.77 MHz ซึ่งจะมีคาบการทำงาน 210 ns หรือ 33% ของรอบการทำงาน

IRQ2 ถึง IRQ7 อุปกรณ์อินพุทและเอาท์พุทจะใช้สายอินพุททั้ง 6 เส้นนี้ในการสร้างการเรียกการ interrupt โดยมันจะกำหนดให้ IRQ2 เป็นตัวที่ดีที่สุด และ IRQ7 เป็นตัวต่ำที่สุด



รูปที่ 2.41 แสดงบัสของระบบของเครื่อง PC

2.5.3 การโปรแกรม 8255

การใช้งาน 8255 จะต้องทำการโปรแกรมเสียก่อน โดยการส่งค่าคอนโทรลไบต์ให้แก่พอร์ตควบคุมจะเป็นคำสั่งขนาด 8 บิท หรือ 1 ไบต์ ซึ่งแต่ละบิทจะมีความหมายและการใช้งานต่างกัน คอนโทรลไบต์นี้จะเป็นคำสั่งกำหนดโหมดการทำงานของ 8255 และการกำหนดให้พอร์ตทั้ง 3 (A,B,C) เป็นอินพุท หรือเอาท์พุท ดังแสดงในรูปที่ 2.18 และมีแอดเดรส A0,A1 เป็นตัวกำหนดการติดต่อกับอินพุทเอาท์พุทพอร์ตทั้ง 3 คือพอร์ต A,B และC ตามลำดับโดยมีการกำหนดแอดเดรสดังนี้

A1	A0	ชื่อพอร์ต
0	0	พอร์ต A
0	1	พอร์ต B
1	0	พอร์ต C
1	1	คอนโทรลพอร์ต

แต่ละบิทของคอนโทรลไบต์ มีความหมายดังนี้

บิท D0 = ข้อมูลในบิทนี้กำหนดให้พอร์ต C ล่าง (PC0-PC3) เป็นอินพุท หรือเอาท์พุท ถ้าบิทนี้เป็น 1 จะเป็นอินพุท แต่ถ้าเป็น 0 จะเป็นเอาท์พุท

บิท D1 = ข้อมูลในบิทนี้กำหนดให้พอร์ต B เป็นอินพุทหรือเอาท์พุท ถ้าบิทนี้เป็น 1 จะเป็นอินพุท แต่ถ้าเป็น 0 จะเป็นเอาว์พุท

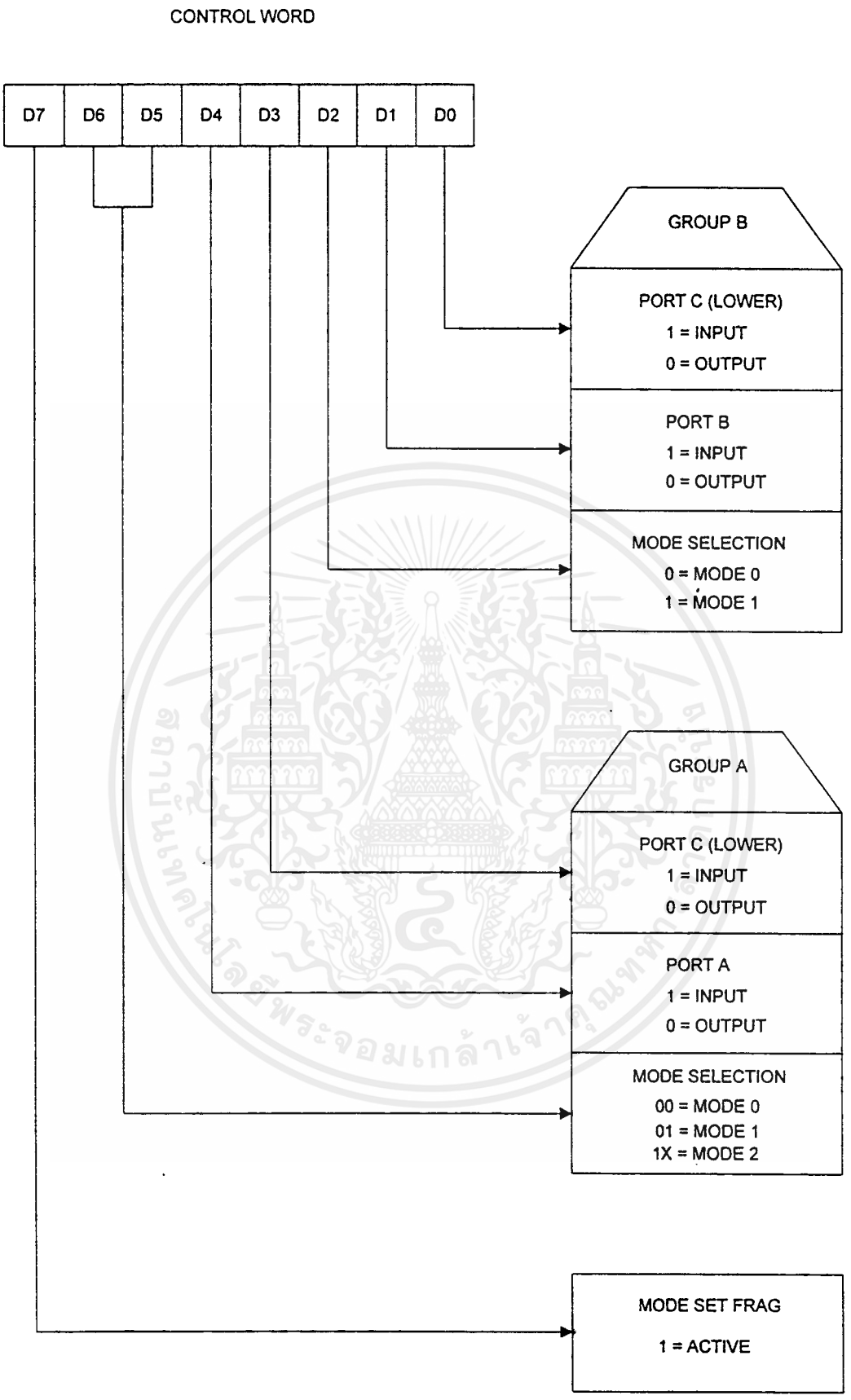
บิท D2 = ข้อมูลในบิทนี้กำหนดการเลือกโหมดของกลุ่ม B ถ้าบิทนี้เป็น 1 จะทำงานในโหมด 1 ถ้าบิทนี้เป็น 0 จะทำงานในโหมด 0

บิท D3 = ข้อมูลในบิทนี้กำหนดให้พอร์ต C บน (PC4-PC7) เป็นอินพุทหรือเอาท์พุท ถ้าบิทนี้เป็น 1 จะเป็นอินพุท แต่ถ้าเป็น 0 จะเป็นเอาท์พุท

บิท D4 =ข้อมูลในบิทนี้กำหนดให้พอร์ท A เป็นอินพุทหรือเอาท์พุท ถ้าบิทนี้เป็น 1 จะเป็นอินพุท แต่ถ้าเป็น 0 จะเป็นเอาท์พุท

บิท D5,D6 =ข้อมูลทั้ง 2 บิทนี้เป็นตัวเลือกโหมดการทำงานของกลุ่ม A ถ้ามีค่าเป็น 00 จะทำงานโหมด 0 หรือถ้ามีค่าเป็น 01 จะทำงานโหมด 1 แต่ถ้ามีค่าเป็น 1X (10 และ 11)จะทำงานในโหมด 2

บิท D7 = ข้อมูลในบิทนี้เกี่ยวกับการเซ็ตเฟลกใน 8255



ตารางที่ 2.5 แสดงรูปแบบการกำหนดค่าของคอนโทรลไบต์

2.5.4 โหมดการทำงานของ 8255

8255 มีโหมดการทำงานอยู่ 3 โหมดด้วยกัน คือ

2.5.4.1 โหมด 0 (อินพุท/เอาท์พุทพื้นฐาน)

2.5.4.2 โหมด 1 (อินพุท/เอาท์พุทสโตรบ)

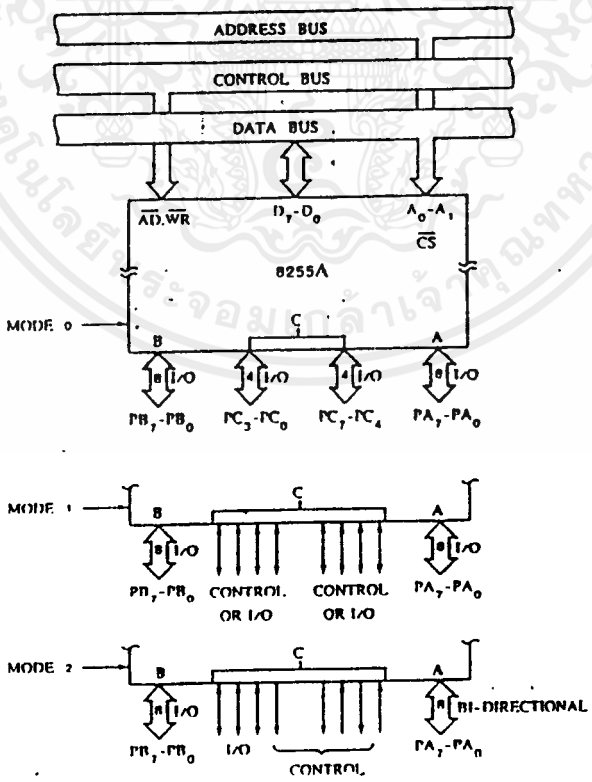
2.5.4.3 โหมด 2 (บัสแบบสองทิศทาง)

2.5.4.1 โหมด 0 (โหมดอินพุท/เอาท์พุทพื้นฐาน)

ในโหมด 0นี้จะกำหนดให้พอร์ต A,B และ C เป็นอินพุท/เอาท์พุทก็ได้ ไม่มีสัญญาณแฮนเชกกิ้ง (Handshaking)เอาท์พุทที่ออกจากพอร์ท A,B และC จะแลช์ (Latched) ค่าไว้ด้วยสามารถจัดรูปแบบของอินพุท/เอาท์พุทได้ 16 แบบด้วยกันดังรูปที่ 2.39

2.5.4.2 โหมด 1(โหมดอินพุท/เอาท์พุทสโตรบ)

การทำงานในโหมด 1 มีการตรวจสอบสัญญาณแฮนด์เชกกิ้ง คือเป็นการทำงานระหว่างอุปกรณ์ภายนอกซีพียู และชิป 8255 โดยให้พอร์ต 8255 เป็นตัวกลางในการทำงาน สามารถใช้งานได้ 2 พอร์ต คือ พอร์ต A และพอร์ต B เป็นได้ทั้งอินพุทและเอาท์พุท โดยใช้พอร์ต C บน (PC4-PC7) ทำการตรวจสอบสัญญาณแฮนเชกกิ้งของพอร์ต B

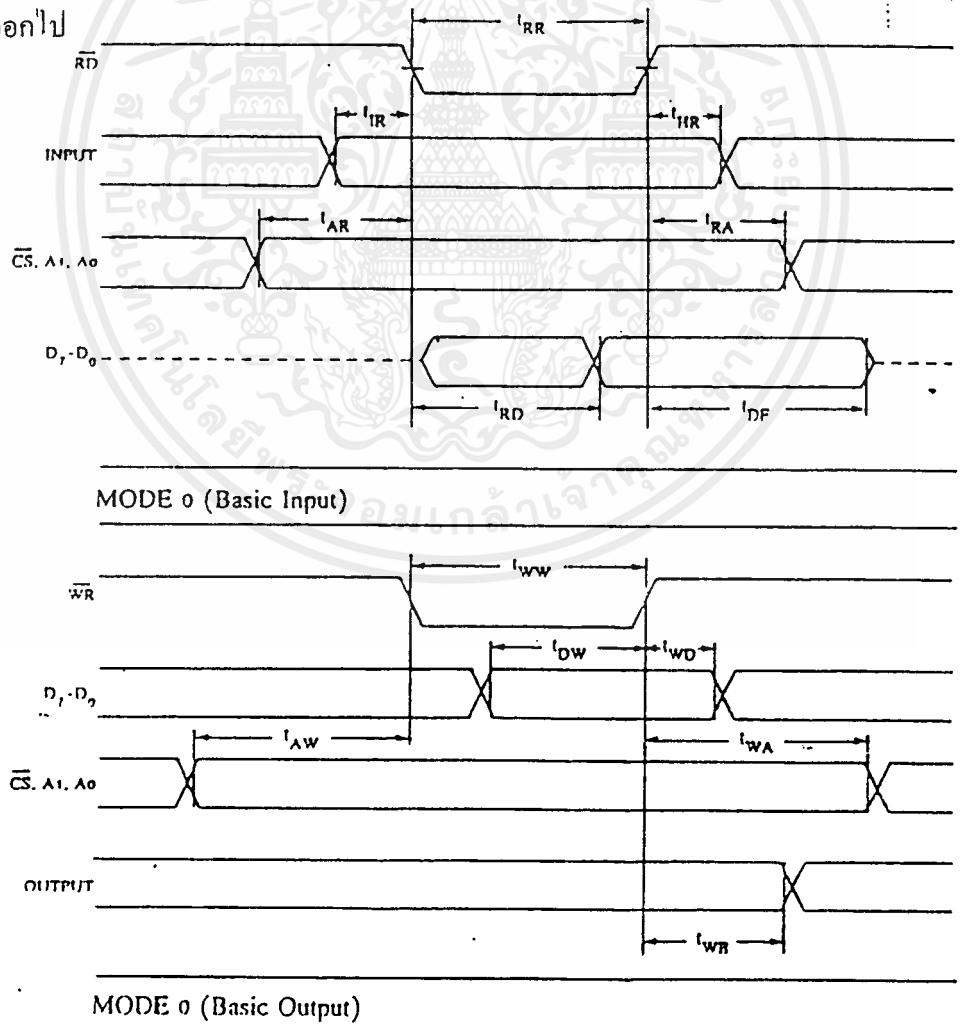


รูปที่ 2.42 แสดงโหมดการทำงานทั้ง 3 พอร์ต และการเชื่อมต่อระบบบัส

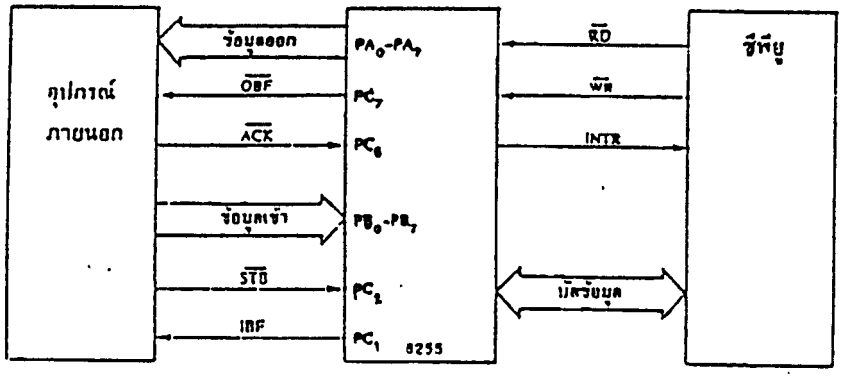
การใช้งาน 8255 ในโหมด 1 นี้ จะมีประโยชน์ในกรณีที่อุปกรณ์ภายนอกไม่สามารถจะทำงานได้เร็วทันการทำงานของซีพียู และเพื่อให้การรับส่งของข้อมูลเป็นไปอย่างถูกต้อง จากรูปที่ 2.44 จะเห็นว่า 8255 ส่งสัญญาณอินเตอร์รัปต์ (INTR) ไปให้ซีพียูเพื่อให้ซีพียูทำการอ่าน หรือเขียนข้อมูลกับอุปกรณ์ภายนอก

เมื่ออุปกรณ์ภายนอกต้องการจะส่งข้อมูลให้ซีพียูส่งสัญญาณ STB (Strobe Input) มาให้ชิป 8255 (รูปที่ 2.45) พร้อมทั้งส่งข้อมูลเข้าไปเก็บไว้ในพอร์ต จากนั้นชิป 8255 จะส่งสัญญาณ IBF (Input Buffer Full) ไปยังอุปกรณ์ภายนอกเพื่อไม่ให้ส่งข้อมูลเข้าไปอีก และ 8255 ก็จะส่งสัญญาณ $INTR_B$ ไปให้ซีพียู จากนั้นซีพียูจะส่งสัญญาณ RD เพื่อทำการอ่านข้อมูลจากพอร์ต B เมื่อเสร็จแล้วสัญญาณ $INTR_B$ จะไม่ทำงาน และสัญญาณ IBF จะไม่ทำงานด้วย เป็นการบอกให้อุปกรณ์ภายนอกส่งข้อมูลชุดใหม่มาได้

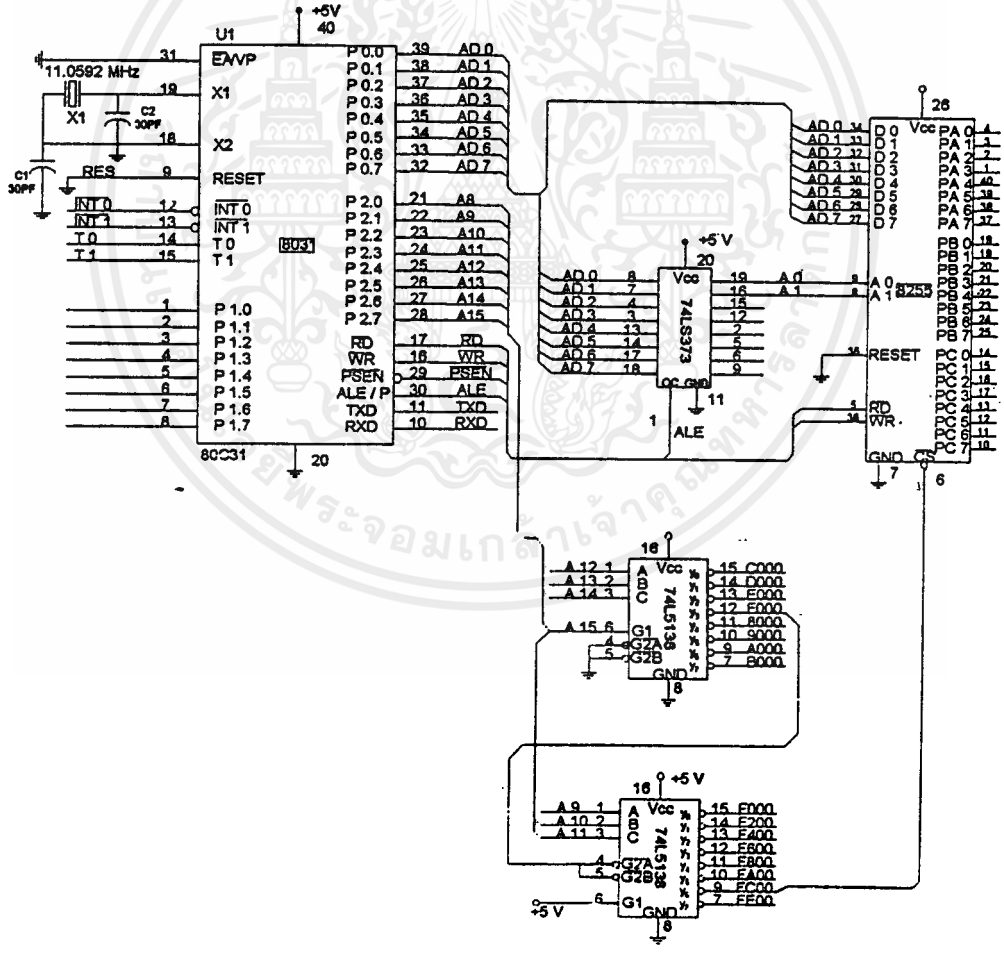
ถ้าซีพียูต้องการจะส่งข้อมูลออกไปให้อุปกรณ์ภายนอกสามารถทำได้โดยส่งสัญญาณ WR ออกมา (รูปที่ 2.46) พร้อมทั้งข้อมูล 8255 จะรับข้อมูลจากซีพียูไว้ในพอร์ต แล้วส่งสัญญาณ OBF (Output Buffer Full) ให้อุปกรณ์ภายนอกรับข้อมูลไปได้ เมื่ออุปกรณ์ภายนอกรับข้อมูลไปแล้วจะส่งสัญญาณ ACK (Acknowledge) ไปให้ 8255 ว่ารับข้อมูลเรียบร้อยแล้ว ทำให้สัญญาณ $INTR_A$ ทำงานเพื่อให้ซีพียูส่งข้อมูลชุดใหม่ออกไป



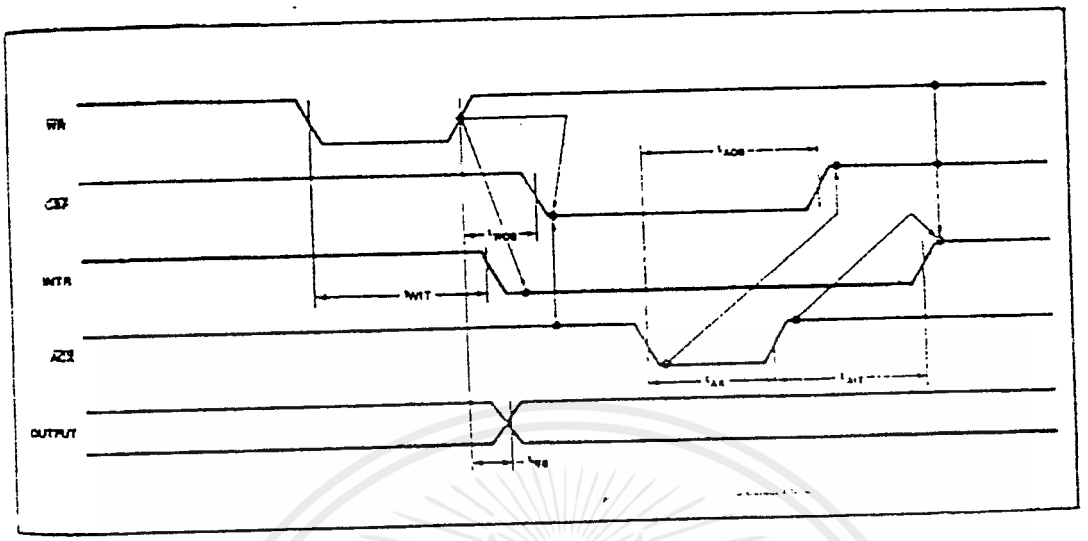
รูปที่ 2.43 แสดงบัสไซเคิลในโหมด 0 ของ 8255



รูปที่ 2.44 แสดงการใช้งานในโหมด 1 ของชิป 8255



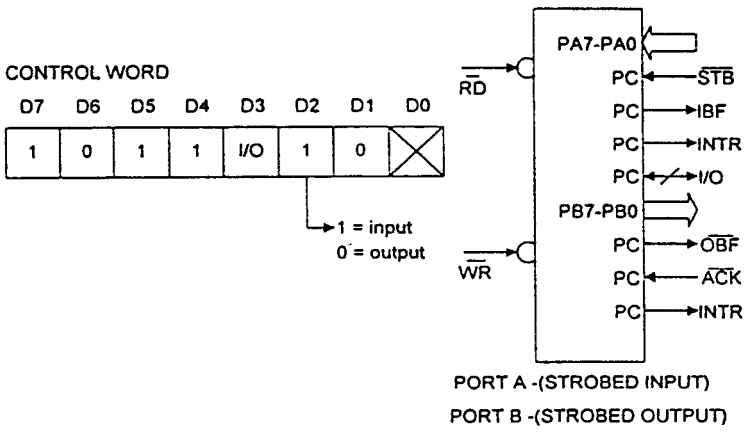
รูปที่ 2.45 แสดงการเชื่อมต่อ 8255 เข้ากับ CPU

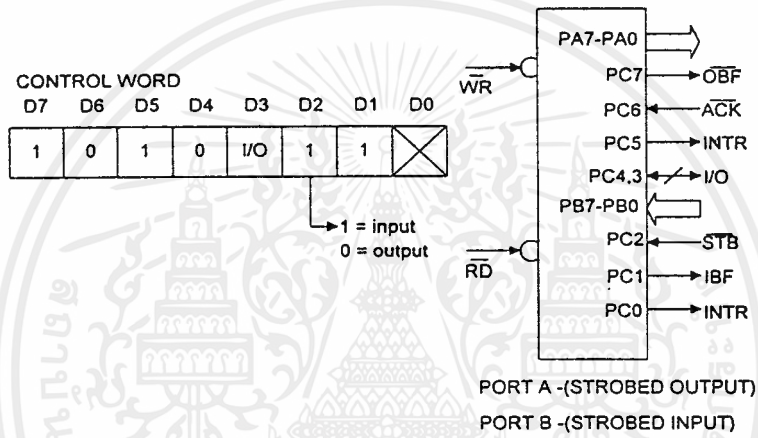


รูปที่ 2.46 แสดงบัสไซเคิลของการแฮนเชกกิ้งขณะเป็นเอาท์พุทพอร์ต

ขา	อินพุท	เอาท์พุท
Pc0	INTRB	INTRB
Pc1	IBFB	OBFB
Pc2	STBB	ACKB
Pc3	INTRA	INTRA
Pc4	STBA	I/O
Pc5	IBFA	I/O
Pc6	I/O	ACKA
Pc7	I/O	OBFA

ตารางที่ 2.6 แสดงหน้าที่สัญญาณของพอร์ต C ในโหมด 1

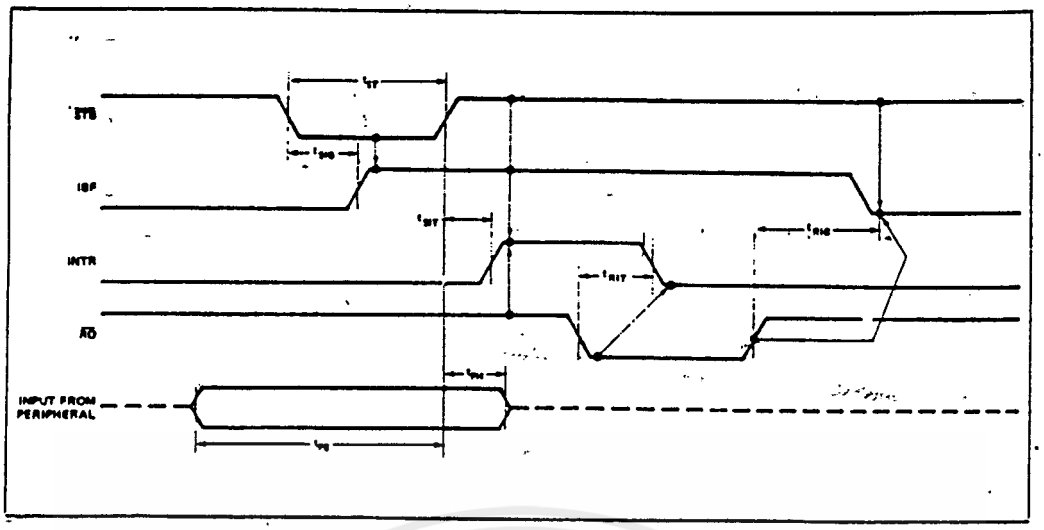




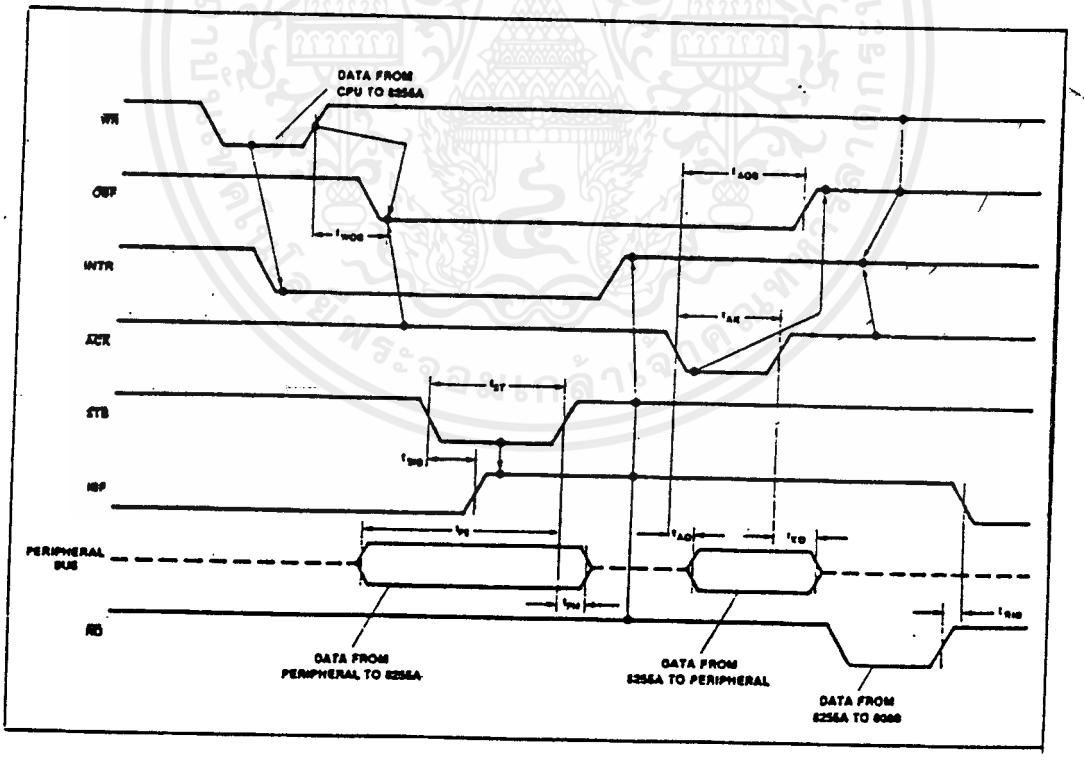
รูปที่ 2.47 แสดงการจัดให้พอร์ต A และพอร์ต B เป็นอินพุท/เอาท์พุทพอร์ตในโหมด 1

2.5.4.3 โหมด 2 (บัสแบบสองทิศทาง)

โหมดสองนี้ยังคงใช้สัญญาณการตรวจสอบแฮนเชกกิ้งอยู่ เป็นบัสแบบสองทิศทาง จัดได้เฉพาะพอร์ต A โดยให้พอร์ต A เป็นทั้งอินพุทและเอาท์พุทพอร์ตจะใช้งานเกี่ยวกับการสื่อสารระหว่างอุปกรณ์ 2 ตัวที่สลับกันรับ/ส่งข้อมูล และใช้พอร์ต C ขนาด 2 บิท ในการตรวจสอบสัญญาณ สัญญาณที่เข้า/ออกจากพอร์ต A จะแลชท์ค่าไว้ด้วยกาทำงานจะคล้ายๆกับโหมด 1 ของชิป 8255



รูปที่ 2.48 แสดงบัสไซเคิ้ลของการแฮนเชกกิ้งขณะที่เป็นอินพุทพอร์ต



รูปที่ 2.49 แสดงบัสไซเคิ้ลของการแฮนเชกกิ้งในโหมด 2

บทที่ 3
การคำนวณและการสร้าง

โครงงานนี้สามารถที่จะแบ่งการทำงานออกเป็นส่วนหลักๆได้ 3 ส่วน ได้แก่

3.1 ส่วนของเครื่องตอบรับโทรศัพท์อัตโนมัติ ในส่วนนี้จะมีการทำงานย่อยอีก 3 ส่วน ดังนี้

3.1.1 ส่วนตรวจจับสัญญาณกระดิ่ง

3.1.2 ส่วนควบคุมการยกหูและวางหู

3.1.3 ส่วนถ่ายทอดสัญญาณเสียง

3.2 ส่วนของโปรแกรมการจดจำเสียง ซึ่งสามารถที่จะแบ่งย่อยได้อีก 2 ส่วน ได้แก่

3.2.1 ส่วนของการตัดคำให้เป็นคำเดี่ยว

3.2.2 ส่วนของการเรียนรู้

3.2.3 ส่วนของการวิเคราะห์และจดจำเสียง

3.2.4 การใช้งานโปรแกรม Voice Commander

3.3 ส่วนของการเชื่อมต่อกับอุปกรณ์ไฟฟ้า ซึ่งสามารถที่จะแบ่งออกได้อีก 2 ส่วนดังนี้

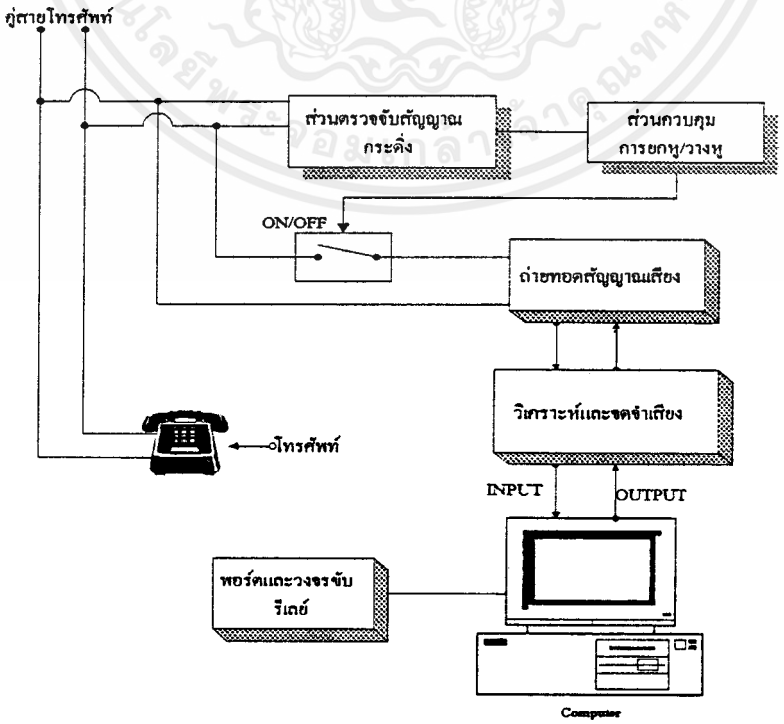
3.3.1 ส่วนของการต่อพอร์ตและการ์ดโปรโตไทป์

3.3.2 ส่วนของวงจรขับรีเลย์

รูปที่ 3.1 แสดงบล็อคไดอะแกรมการทำงานของการสั่งงานเครื่องใช้ไฟฟ้าผ่านทางโทรศัพท์โดยใช้เสียง

3.1 ส่วนของเครื่องตอบรับโทรศัพท์อัตโนมัติ

การทำงานของเครื่องตอบรับโทรศัพท์สามารถแบ่งได้ 3 ส่วน

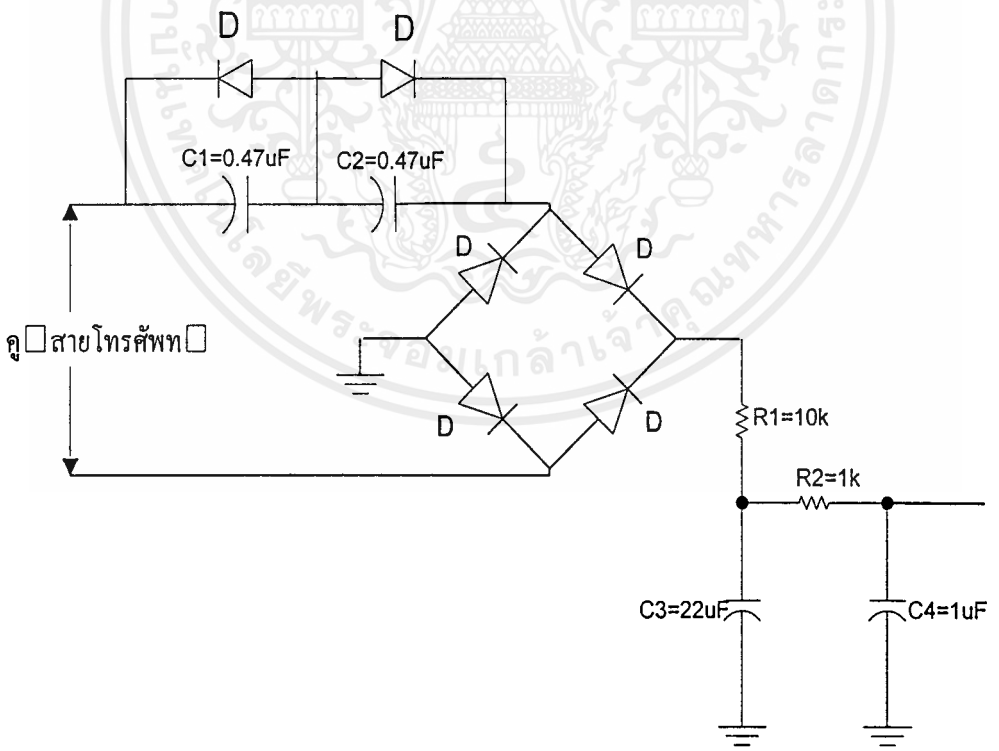


รูปที่ 3.1 แสดงบล็อคไดอะแกรมการทำงานของการสั่งงานเครื่องใช้ไฟฟ้าผ่านทางโทรศัพท์โดยใช้เสียง

3.1.1 ส่วนตรวจจับสัญญาณกระดิ่ง

เมื่อมีสัญญาณกระดิ่งจากชุมสายโทรศัพท์ ส่วนของวงจรตรวจจับสัญญาณกระดิ่งโดยมี C_1,C_2 และ D_1,D_2 ต่อประกอบกันเป็นวงจรถ่ายทอดสัญญาณกระดิ่ง โดยยอมให้กระแสสลับไหลผ่านเท่านั้น โดยจะกั้นไม่ให้ไฟตรงจากชุมสายผ่านวงจรนี้ กล่าวคือขณะที่สายว่างจะมีไฟตรงจากชุมสายตกคร่อมอยู่ 48 โวลต์ วงจรในส่วนนี้ก็จะเป็นตัวกั้นไฟตรง แต่เมื่อมีสัญญาณกระดิ่งซึ่งเป็นไฟกระแสสลับเข้ามาก็จะยอมให้ไฟกระแสสลับนั้นผ่านเข้าไปได้ สัญญาณกระดิ่งจะผ่าน C_1,C_2 มาได้ก็จะถูกเรคติไฟร์ โดย D_3-D_6ที่ต่อเป็นวงจรบริดจ์เรคติไฟร์เออร์ สัญญาณกระดิ่ง 16 Hz+-100V ก็จะถูกแปลงเป็นสัญญาณพัลส์ 32 Hz+-100V จะเห็นว่า ไฟกระแสสลับของสัญญาณกระดิ่งเมื่อผ่านวงจรบริดจ์เรคติไฟเออร์ในส่วนของสัญญาณซีกลบจะโดนกลับเฟสให้มาอยู่ในซีกบวกหมด เพราะฉะนั้นจึงทำให้มีความถี่เพิ่มขึ้น แต่เราจะเห็นได้ว่าแรงดันมีค่าสูงมากจึงต้องลดทอนลงด้วย R_1 ให้มีแรงดันที่เหมาะสม

ตัวเก็บประจุ C_3,C_4 และตัวต้านทาน R_2 ต่อเข้าด้วยกันเป็นวงจรกรองสัญญาณ ซึ่งมีหน้าที่ในการกรองสัญญาณ กล่าวคือสัญญาณที่ออกมาจากวงจรบริดจ์เรคติไฟเออร์จะยังมีการกระเพื่อมของสัญญาณอยู่ เพราะฉะนั้นเราจึงต้องใส่วงจรกรองสัญญาณเพื่อลดการกระเพื่อมของสัญญาณ และยังทำให้แรงดันที่ขาเบส-อีมิตเตอร์ของทรานซิสเตอร์ Q_1 ค่อยๆเพิ่มขึ้นเมื่อเพิ่มถึงระดับหนึ่ง ทรานซิสเตอร์ Q_1 ก็จะนำกระแส



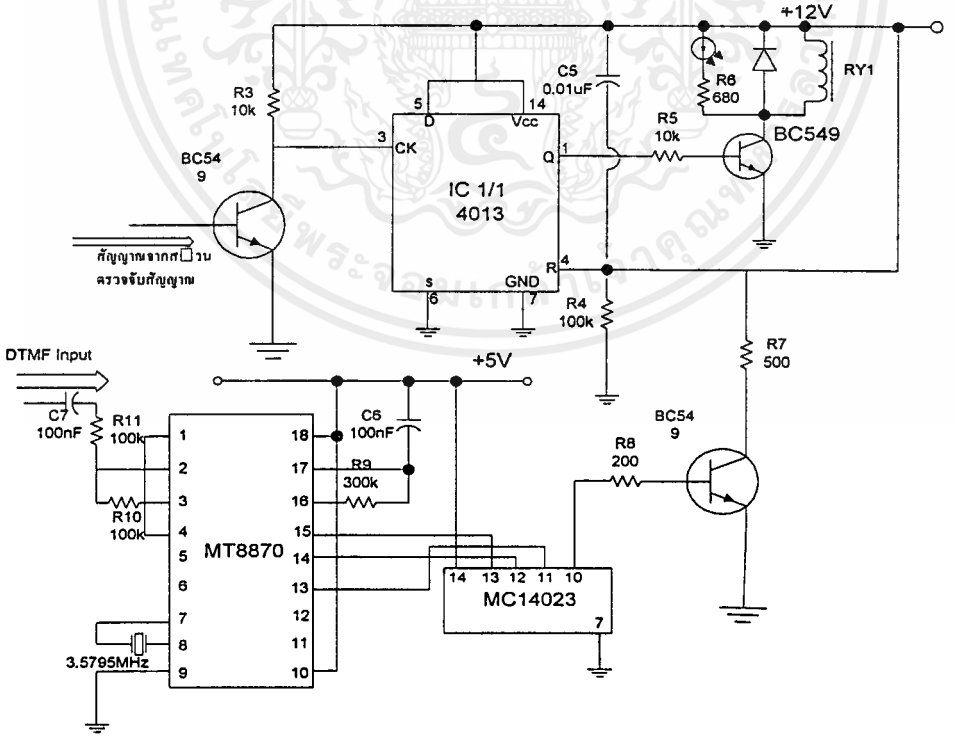
รูปที่ 3.2 แสดงวงจรในส่วนตรวจจับสัญญาณกระดิ่ง

3.1.2 ส่วนควบคุมการยกหูและวางหู จะมีการทำงาน 2 ส่วน ดังนี้

3.1.2.1 ส่วนควบคุมการยกหู มีหลักการดังนี้

เมื่อมีการนำกระแสที่ทรานซีสเตอร์ Q_1ซึ่งจะทำให้เสมือนว่าทรานซีสเตอร์ Q_1 ได้ทำการช็อต (Short) ขาคอลเล็กเตอร์เข้ากับขาอีมิตเตอร์ จะทำให้ขา CK ของ $IC_{1/1}$ ที่เป็น D-Flip Flop ได้ต่อลงกราวน์ ซึ่งก็คือขา CK ของ $IC_{1/1}$ ที่เป็น D-Flip Flop ได้รับสัญญาณนาฬิกาที่เป็นขอบขาลงคือ เดิมจะมีแรงดันตกคร่อมที่ขา CK 12 V แต่เมื่อมีการช็อตขา CK ลงกราวน์ก็จะทำให้แรงดันตกคร่อมเหลือ 0 V จึงยังไม่มีผลต่อ $IC_{1/1}$ เนื่องจาก $IC_{1/1}$ จะทำงานเมื่อขา CK เป็นขอบขาขึ้น เมื่อสัญญาณกระดิ่งลูกแรกหมดลง ก็จะไม่มีการนำกระแสที่ทรานซีสเตอร์ Q_1 ซึ่งจะทำให้ไม่มีการช็อตระหว่างขาคอลเล็กเตอร์และขาอีมิตเตอร์ แรงดันที่ขา CK ก็จะเปลี่ยนจาก 0 V เป็น 12 V ก็จะทำให้ $IC_{1/1}$ ทำงานโดยให้เอาท์พุตออกที่ขา Q เป็นลอจิกเดียวกับที่ขา D ในทันทีที่ขา CK เกิดการเปลี่ยนแปลงจากลอจิก “0” เป็น “1” และขา D ขณะนี้จะมีลอจิก “1” ดังนั้นที่ขา Q ขณะนี้จะมีลอจิก “1” ไปเป็นไบอัสให้ Q_2 นำกระแส รีเลย์ Ry_1 ที่ต่ออยู่ด้วยจึงทำงาน ขณะนี้หน้าคอนแทครีเลย์จะทำการต่อขดไพรมารีเข้ากับคู่สายโทรศัพท์เป็นการยกหูโทรศัพท์และเชื่อมต่อวงจรถ่ายทอดสัญญาณเข้ากับคู่สายไปในเวลาเดียวกัน

ทำให้ผู้ที่เรียกเข้ามาสามารถทราบได้ว่าโทรศัพท์ได้ยกหูแล้ว โดยผู้ที่โทรศัพท์มาสั่งงานเสียงที่สั่งงานจะส่งไปยัง Sound Card และถูกส่งไปประมวลผลภายในคอมพิวเตอร์ต่อไป เพื่อให้ทำงานตามคำสั่ง



รูปที่ 3.3 แสดงวงจรในส่วนควบคุมการยกหู/วางหู

3.1.2.2 ส่วนควบคุมการวางหู มีหลักการดังนี้

เมื่อผู้ที่เรียกเข้ามาต้องการยกเลิกการติดต่อทำได้โดยการกดปุ่ม # เมื่อกดปุ่ม # สัญญาณ DTMF ที่มีความถี่ 941Hz กับ 1477 Hz จะถูกส่งไปยังวงจร MT8870 วงจร MT8870 เมื่อได้รับสัญญาณ DTMF ก็จะทำการแปลงสัญญาณ DTMF นั้นออกมาเป็นรหัสจำนวน 4 บิต โดยรหัสที่ออกมานี้จะไม่เหมือนกันในการกดปุ่มแต่ละปุ่ม โดยผลที่เกิดจากการกดปุ่ม # จะได้เอาท์พุทที่ขา

$Q_1 = 0$

$Q_2 = 0$

$Q_3 = 1$

$Q_4 = 1$

และที่ขา "Std" จะมีค่าเป็น 1 เสมอเมื่อมีการกดปุ่มไม่ว่าปุ่มใด

เราจะทำการเลือกรหัสจำนวน 3 บิต ต่อเข้าไปยัง MC14203 ซึ่งทำหน้าที่เป็นแนนซ์เกต เอาท์พุทที่ออกจากแนนซ์เกตเมื่อกดปุ่มอื่นหรือก่อนที่จะมีการกดปุ่ม # จะมีค่าลอจิกเป็น "1" เสมอ ซึ่งจะมีไฟขนาด 5 V ออกมาจากแนนซ์เกต ซึ่งจะมีการนำกระแสจึงทำให้ทรานซิสเตอร์ Q_2 ทำงาน คือจะมีการช็อตระหว่างขาคอลเล็กเตอร์และอีมิตเตอร์ ทำให้ขารีเซ็ต (Reset) ของ D-FlipFlop เสมือนกับว่าต่อลงกราน์ดลอดซึ่งก็คือจะไม่มีการทำการรีเซ็ต ดังนั้นเราจะเลือกเอาขา Std,Q_3 และ Q_4 ไปเชื่อมต่อเข้ากับแนนซ์เกต คือเมื่อมีการกดปุ่ม # จะมีค่าลอจิก 1,1 และ 1 เข้าไปยังแนนซ์เกต ซึ่งจะทำให้ค่าลอจิกที่ออกมาจากแนนซ์เกตมีค่าเป็น "0" ดังนั้นจึงไม่มีการเกิดกระแสเหนี่ยวนำที่ทรานซิสเตอร์ Q_2 ทำให้ไม่มีการช็อตระหว่างขาคอลเล็กเตอร์และอีมิตเตอร์ ดังนั้นจึงทำให้มีแรงดันตกคร่อมที่ขารีเซ็ตมีค่าเท่ากับ 12 V ซึ่งทำให้ไปทริกซ์ IC4013 ก็จะทำการตัดการติดต่อ โดยจะไม่มีกระแสไหลผ่านรีเลย์ก็จะทำให้ทำการวางหู

ในส่วนควบคุมการวางหูเราจำเป็นที่จะต้องใส่วงจรขยายสัญญาณ เพราะจากการทดลองสัญญาณ DTMF ที่เข้ามาที่ขาอินพุทของ MT8870 มีความแรงของสัญญาณน้อยเกินไป เนื่องจากการนำสัญญาณจากคู่สายโทรศัพท์เราต้องผ่านทรานฟอร์มเมอร์ก่อนซึ่งทำให้ความแรงของสัญญาณน้อยลง โดยในส่วนของวงจรขยายสัญญาณนี้เราเลือกใช้ วงจร นอน-อินเวอร์ตติ้ง แอมปลิไฟเออร์ (Non-Inverting Amplifier)

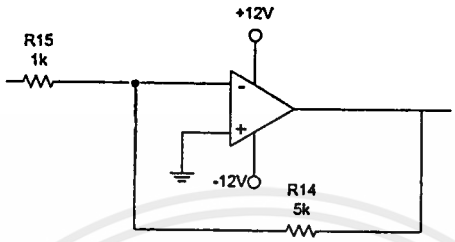
โดยตัวไอซีที่ใช้เราใช้ LF353 ซึ่งเราจำเป็นที่จะต้องจ่ายไฟเลี้ยง 12 V และ –12 V โดยจากรูปที่ 3.4 Rf = 5k และ Rs = 1k โดยเราสามารถคำนวณหาค่าอัตราการขยายของสัญญาณได้จากสมการดังต่อไปนี้

อัตราการขยาย (Gain) = - Rf/Rs

จากสมการเราจะเห็นได้ว่าค่าอัตราการขยายมีการติดลบเพราะว่า มีการกลับเฟสของสัญญาณ ซึ่งจากการพิจารณาก็จะไม่มีผลต่อ การทำงานของวงจร DTMF โดยเราจะคำนึงถึงความถี่เฟสจะไม่มีผลต่อการทำงาน

อัตราการขยาย (Gain) = -5k/1k = -5

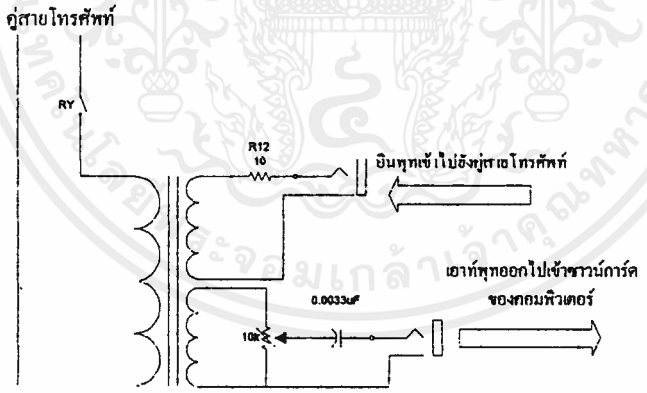
โดยจากวงจรแล้ว เราจะได้ค่าอัตราการขยาย (Gain) = -5 โดยระดับของสัญญาณเอาท์พุทมีความแรงของสัญญาณมากกว่าความแรงของสัญญาณอินพุท 5 เท่า และจะมีการกลับเฟสกัน



รูปที่ 3.4 แสดงวงจรขยายสัญญาณ

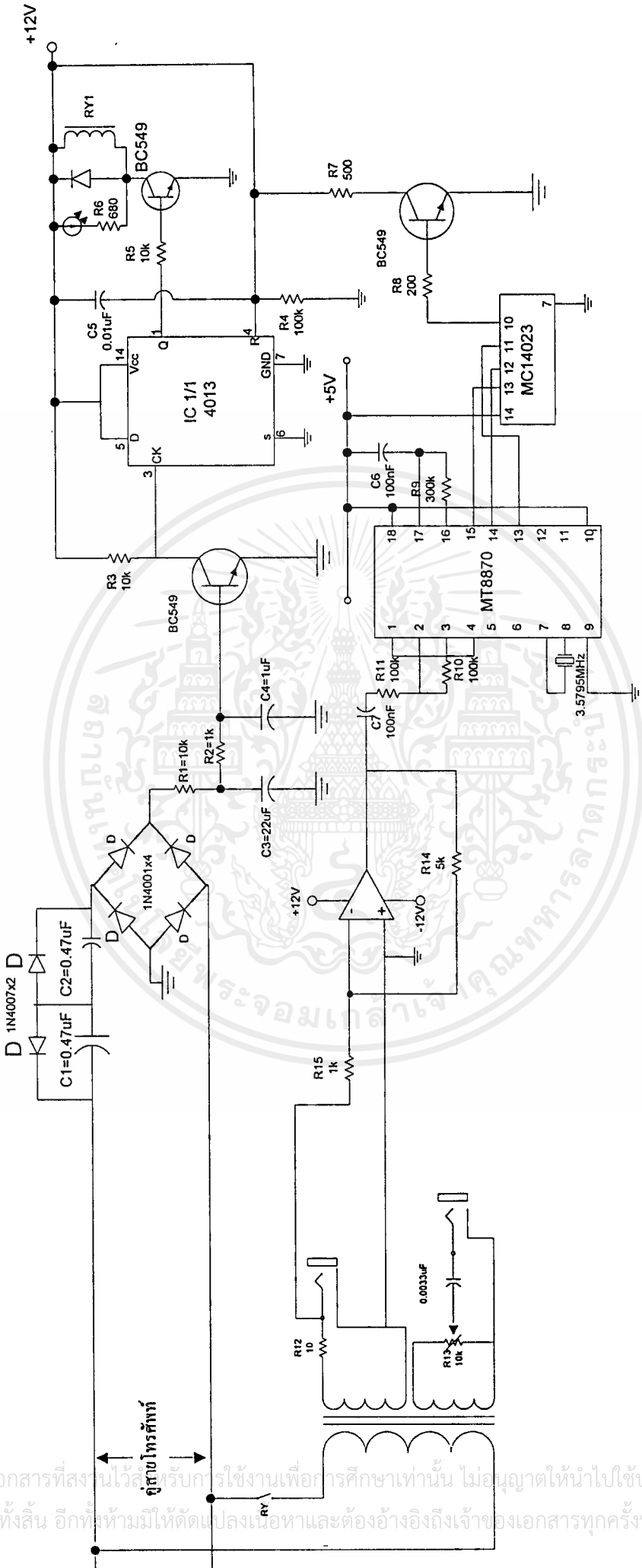
3.1.3 ส่วนถ่ายทอดสัญญาณเสียง

ส่วนถ่ายทอดสัญญาณเสียงจะมีรูปวงจรดังนี้



รูปที่ 3.5 แสดงวงจรถ่ายทอดสัญญาณเสียง

โดยการถ่ายถอดสัญญาณเราจะใช้ทรานฟอร์มเมอร์ (Transformer) เป็นตัวแปลงสัญญาณเสียงจากคู่สายโทรศัพท์เข้าไปยังคอมพิวเตอร์ และแปลงสัญญาณเสียงจากคอมพิวเตอร์เข้าไปยังคู่สายโทรศัพท์ โดยจะมีแจ็ค (Jack) เป็นตัวเชื่อมต่อ โดยสัญญาณจากคู่สายโทรศัพท์สามารถลดทอนได้ โดยเราจะใช้วิธีการ Voltage Divider เพื่อไม่ให้สัญญาณที่เข้าซาวน์การ์ดของคอมพิวเตอร์มีความแรงจนเกินไป เราจะสามารถปรับได้ที่ตัวต้านทานปรับค่าได้ 10 kΩ



รูปที่ 3.6 แสดงวงจรรวมทั้งหมดของส่วนของเครื่องตอบรับโทรศัพท์อัตโนมัติ

3.2 ส่วนของโปรแกรมการจดจำเสียง

ในส่วนของโปรแกรมการจดจำเสียงจะประกอบด้วย 2 ส่วนหลักได้แก่

3.2.1 ส่วนของการตัดคำให้เป็นคำเดี่ยว

3.2.2 ส่วนของการเรียนรู้

3.2.3 ส่วนของการวิเคราะห์และจดจำ

3.2.4 การใช้งานโปรแกรม Voice Commander

โปรแกรมทั้งสองส่วนนี้เขียนด้วยภาษาซี โดยรันโปรแกรมบน Visual C++

3.2.1 ขั้นตอนในการตัดคำให้เป็นคำเดี่ยว

ในขั้นตอนนี้จะทำการหาค่าขนาดกำลังสอง เพื่อใช้ในการกำหนดขอบเขตของคำ และทำการตัดคำให้เป็นคำเดี่ยวเพื่อใช้ในการจดจำ โดยจะมีขั้นตอนย่อยๆ ดังนี้

3.2.1.1 ในขั้นตอนนี้เราจะทำการกำหนดค่าพารามิเตอร์ต่างๆที่ใช้ในการคำนวณ เช่น ขนาดของเฟรม ขนาดความกว้างที่ใช้ในการเลื่อนเฟรม โดยค่าพารามิเตอร์จะถูกนำไปใช้ในการหาค่าขนาดกำลังสองของสัญญาณเสียงที่ป้อนเข้าไป

3.2.1.2 อ่านข้อมูลของสัญญาณเสียง และทำการคำนวณหาค่าเฟรมทั้งหมดของเสียงนั้น

3.2.1.3 คำนวณหาค่าขนาดกำลังสองในแต่ละเฟรมจนครบทุกเฟรม

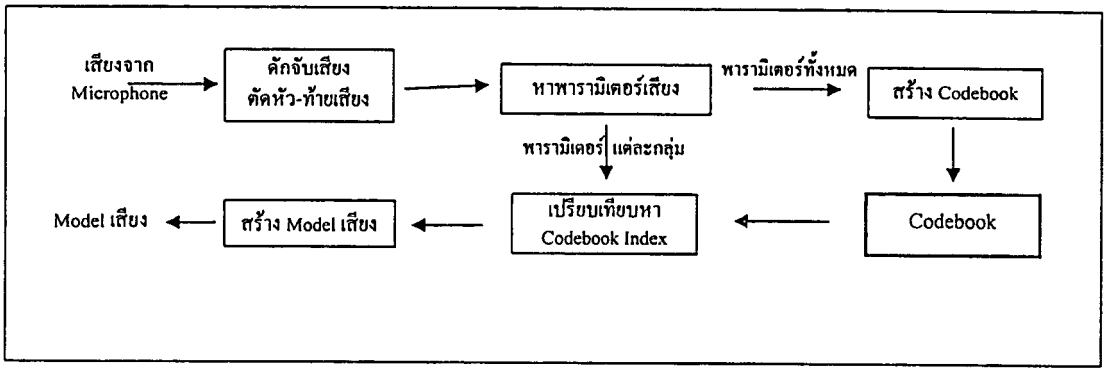
3.2.1.4 ทำการหาค่าขนาดกำลังสองเฉลี่ยจากเสียงเพื่อที่จะทำการกำหนดขอบเขตของคำแต่ละคำอย่างคร่าวๆ แต่การกำหนดขอบเขตของคำนั้นจะไม่สามารถคลอบคลุมคำได้ทั้งหมด

3.2.1.5 ทำการหาค่าขนาดกำลังสองเฉลี่ยระดับต่ำ เพื่อช่วยในการขยายขอบเขตของคำ เพื่อให้สามารถคลอบคลุมขอบเขตของคำได้เกือบหมด

3.2.1.6 ทำการขยายขอบเขตของคำโดยใช้ค่าขนาดกำลังสองเฉลี่ยระดับต่ำอีกหนึ่งที

3.2.1.7 นำขอบเขตคำที่ได้มาใช้ในการตัดแบ่งคำให้เป็นคำเดี่ยว

3.2.2 ขั้นตอนการเรียนรู้



รูปที่ 3.7 แสดงขั้นตอนการเรียนรู้

เป็นขั้นตอนที่ต้องการสร้างแบบจำลองของเสียงขึ้นมา เพื่อนำไปใช้ในขั้นตอนการรู้จำเสียงพูดต่อไป ประกอบด้วยส่วนย่อยๆ ดังนี้

3.2.2.1 การวิเคราะห์สัญญาณเสียงเบื้องต้น

มีการเลือกใช้ ค่าต่างๆ ในการคำนวณและออกแบบโปรแกรมดังนี้

3.2.2.1.1) การพรีเอมฟาซีส

ใช้วงจรอันดับหนึ่ง ซึ่งมีฟังก์ชั่นถ่ายโอน คือ

$$H(z) = 1 - \alpha z^{-1}$$

ค่า α ที่ใช้คือ 15/16 = 0.9375

3.2.2.1.2) การแบ่งช่วงสัญญาณ

ขนาดของช่วงสัญญาณมีเงื่อนไขในการเลือก คือ

- ค่า N ต้องสั้นพอที่คุณสมบัติของเสียงไม่ เปลี่ยนแปลง
- ค่า N ต้องยาวพอที่จำนวนของตัวอย่างมี เพียงพอสำหรับการหาสัมประสิทธิ์
- การเลื่อนในการวิเคราะห์ (ค่า M) ต้องไม่ข้ามข้อมูล

ดังนั้นค่า M จะน้อยกว่า N แต่ถ้าค่า M มีขนาดเล็กเกินไปจะทำให้การคำนวณช้าลง ดังนั้นจึงเลือกค่า M = 100 แซมเปิ้ล และค่า N = 300 แซมเปิ้ล

3.2.2.1.3) ความถี่ที่ใช้ในการสุ่มสัญญาณ

เนื่องจาก ความถี่ที่ใช้ในการแซมปลิ้ง มากกว่าหรือเท่ากับ สองเท่าของความถี่เสียง ($f_s \geq f_N$) ดังนั้น $f_s \geq$ 8 KHz แต่เนื่องจากใน wave studio มีความถี่ที่ใกล้เคียงที่สุด คือ 11.025 Khz

ดังนั้น ช่วงเวลาที่ใช้ในการวิเคราะห์แต่ละเฟรม คือ 300/11.025 = 27.21 ms และระยะที่ใช้ในการเลื่อนเฟรมคือ 100/11.025 = 9.07 ms

3.2.2.1.4) การเลือกวินโดว์ที่เหมาะสมสำหรับการวิเคราะห์เสียง

โดยพิจารณาลักษณะสเปคตรัม คือ

- ความถี่เรโซลูชั่นสูง (high frequency resolution) คือ มีโลบหลักแคบและแหลม
- การลดทอน (Attenuation) นอกช่วงความถี่ที่ผ่านได้ ต่ำ คือ ไซด์โลบมีค่าน้อยฟังก์ชันวินโดว์มีหลายชนิด แต่ชนิดที่เหมาะสมที่สุดที่นำมาใช้ได้แก่ แฮมมิ่งวินโดว์ ซึ่งมีค่าไดโลบต่ำ (- 40 dB) และเมนโลบแคบพอใช้

3.2.2.1.5) การหาคุณลักษณะของเสียง

ใช้การประมาณพันธะเชิงเส้นในการวิเคราะห์หาค่าสัมประสิทธิ์ LPC ซึ่งการประมาณเชิงเส้นที่เลือกใช้ คือ วิธีอัตสัมพัทธ์ (autocorrelation method) ซึ่งวิธีนี้มีการคำนวณที่ง่ายกว่าวิธีอื่นๆ และมีความแน่นอนด้านเสถียรภาพ อีกทั้งมีวิธีการเก็บข้อมูลที่น้อยกว่า

เนื่องจากเป็นการวิเคราะห์โดยวิธีอัตสัมพัทธ์ อันดับการประมาณเชิงเส้น (P) ที่มากจะทำให้การประมาณเสียงมีความใกล้เคียงมากยิ่งขึ้น แต่ถ้าอันดับ P มีค่ามากเกินไปจะทำให้การคำนวณมีความยุ่งยาก และใช้เวลานาน ดังนั้น เพื่อความเหมาะสมค่าอันดับ P ที่ใช้ คือ 12

3.2.2.2 การสร้างโค้ดบุค

จากการทดสอบ (L.R. Rabiner, S.E. Levinson และ Sondhi ,1982) จะได้ว่าที่ขนาดโค้ดบุคเท่ากับ 64 จะมีค่าความคลาดเคลื่อนเฉลี่ยประมาณ 0.2 ซึ่งเป็นค่าที่น้อยมากสำหรับเวคเตอร์ควอนไตซ์เซชั่น

การวัดค่าความคลาดเคลื่อน ใช้วิธีการคำนวณแบบ square error distortion ในการหาระยะทาง เนื่องจากเป็นวิธีที่ง่าย และรวดเร็ว

การสร้างโค้ดบุค โดยนำเทรนนิ่งเซตที่ได้จากการประมาณเชิงเส้นมาผ่านกระบวนการดังนี้

3.2.2.2.1) สุ่มค่าโค้ดบุคเริ่มต้นมา 64 ตัว ตัวละ 19 มิติ

3.2.2.2.2) หาระยะทางระหว่างโค้ดบุคกับเทรนนิ่งเซตแต่ละตัวโดยใช้ความคลาดเคลื่อนกำลังสอง

3.2.2.2.3) จัดกลุ่มของเวคเตอร์อินพุทโดยพิจารณาจากระยะทางที่น้อยที่สุด

3.2.2.2.4) หาจุดศูนย์กลางของกลุ่ม

3.2.2.2.5) ทำขั้นตอน 3 และ 4 ซ้ำจนกว่าความคลาดเคลื่อนรวมจะน้อยกว่า 0.001 ซึ่งจุดศูนย์กลางที่ได้ก็คือโค้ดบุคนั่นเอง

3.2.2.3 การสร้างแบบจำลอง HMM ของเสียง มีขั้นตอนดังนี้

3.2.2.3.1) สุ่มค่าเริ่มต้น a, b และ กำหนดให้ π = [100000] ตามเงื่อนไขในการใช้แบบจำลองแบบ Left - Right

3.2.2.3.2) หาค่า α ,β จากค่า a,b เริ่มต้น และลำดับค่าปรากฏ $O = \{ O_1 O_2 O_3 \ldots O_T \}$ ซึ่งเรียกว่าลำดับเทรนนิ่ง ตามวิธีของ Forward - Backward Procedure โดยใช้ลำดับของค่าปรากฏหลายๆ ลำดับเข้ามาเทรน เพื่อความถูกต้องมากขึ้น

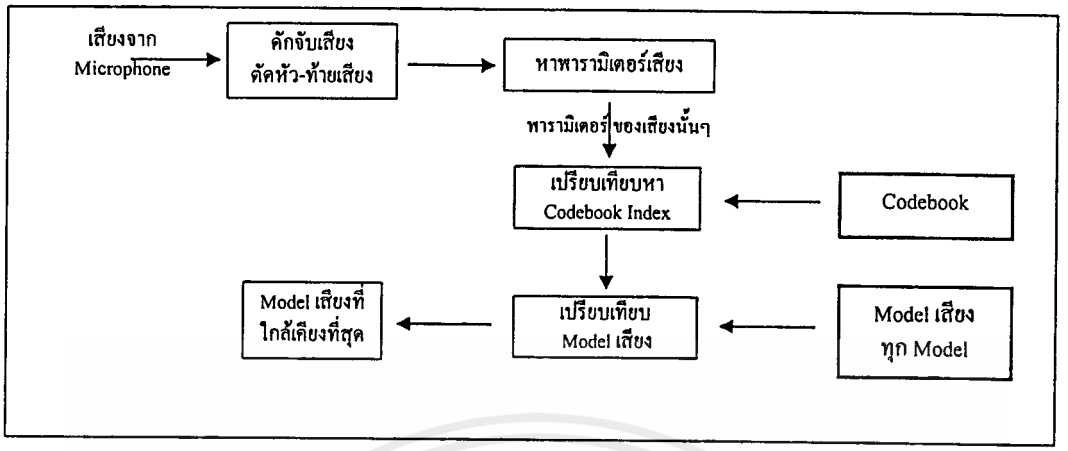
3.2.2.3.3) ทำการสเกลลิ่ง α ,เพื่อให้ค่าอยู่ในย่านที่คอมพิวเตอร์สามารถคำนวณได้อย่างถูกต้อง

3.2.2.3.4) หาค่าพารามิเตอร์ a,b,π ที่ให้ค่าความน่าจะเป็นสูงสุด ที่จะเป็นแบบจำลอง λ ที่เหมาะสมของคำ

3.2.2.3.5) ตรวจสอบค่าพารามิเตอร์ของแบบจำลองที่ได้ ว่า ลู่เข้าหรือยัง โดยใช้วิธีการคำนวณค่า a,b ซ้ำประมาณ 50 รอบ เมื่อมีการเปลี่ยนแปลงน้อยมากจนเป็นที่พอใจตามระดับค่าที่ตั้งไว้ในที่นี้ใช้ค่าเท่ากับ 10^{-5} ก็จะหยุด และได้ค่าพารามิเตอร์ a,b และ π ของแบบจำลองที่ต้องการ

3.2.2.3.6) เก็บค่าพารามิเตอร์ a,b,π ที่ได้จากข้อ 5. เป็นพารามิเตอร์ของแบบจำลองไว้

3.2.3 ขั้นตอนการวิเคราะห์และจดจำ



รูปที่ 3.8 แสดงขั้นตอนการวิเคราะห์และจดจำ

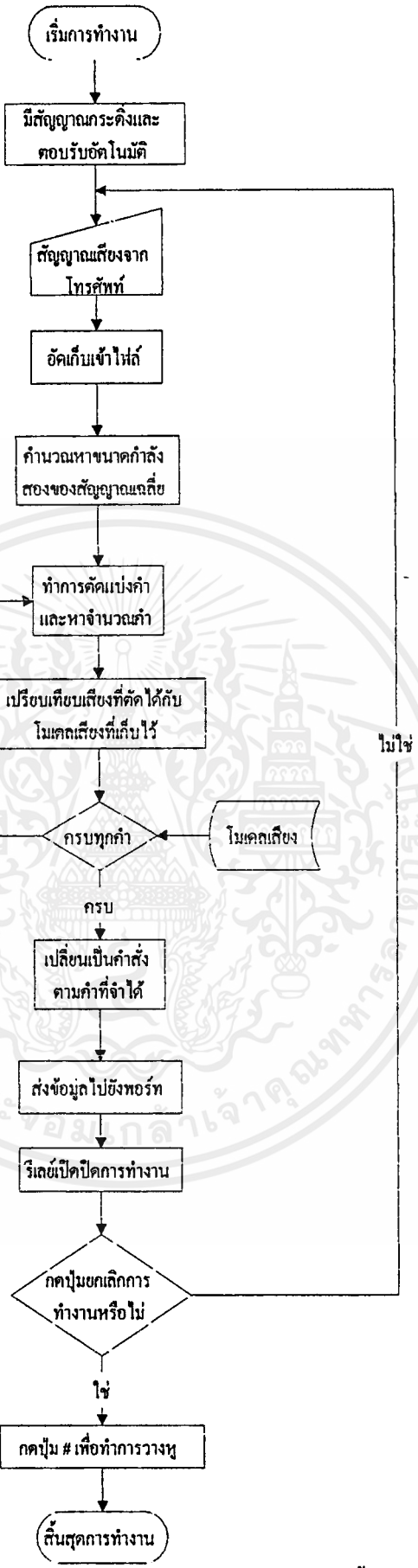
ในขั้นตอนนี้จะมีขั้นตอนย่อยๆอีก 2 ขั้นตอน ดังนี้

3.2.3.1 การหาดัชนีโค้ดบุค

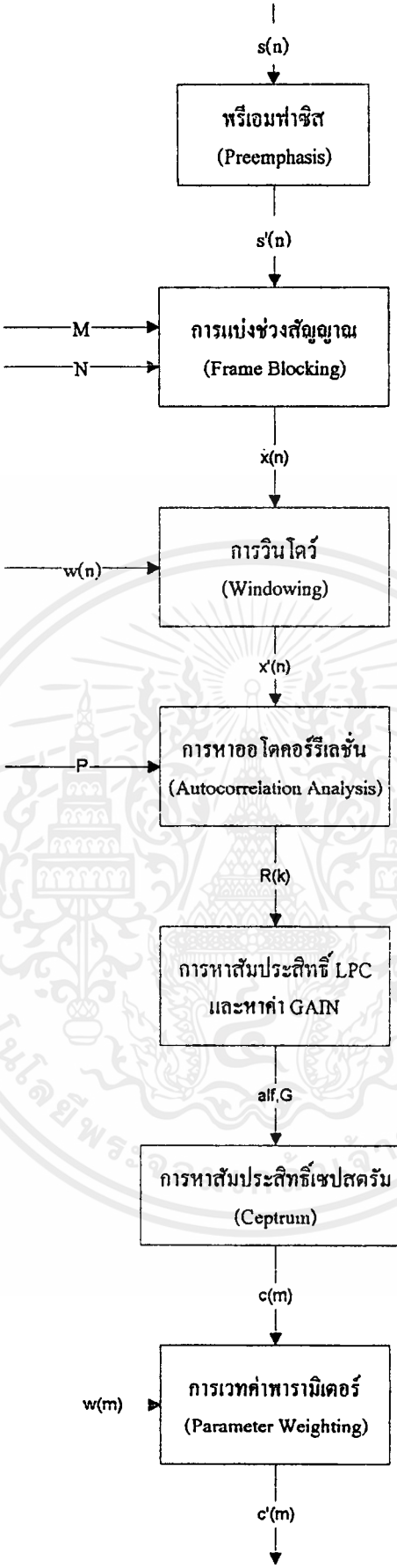
โดยการนำเวกเตอร์เสียงจากการประมาณเชิงเส้นมาทีละเสียง แล้วเปรียบเทียบกับโค้ดบุคที่ได้จากการสร้างในขั้นตอนการเรียนรู้ ทีละเฟรม โดยวิธีความคลาดเคลื่อนกำลังสอง เวกเตอร์เสียงห่างจากโค้ดบุคใดน้อยที่สุดจะได้ว่าเป็นดัชนีโค้ดบุคของเฟรมเสียงนั้น และเก็บดัชนีโค้ดบุคของแต่ละเฟรมในแต่ละเสียงไว้เป็นลำดับค่าปรากฏ (observation sequence) สำหรับการสร้างแบบจำลองต่อไป

3.2.3.2 การรู้จำเสียง

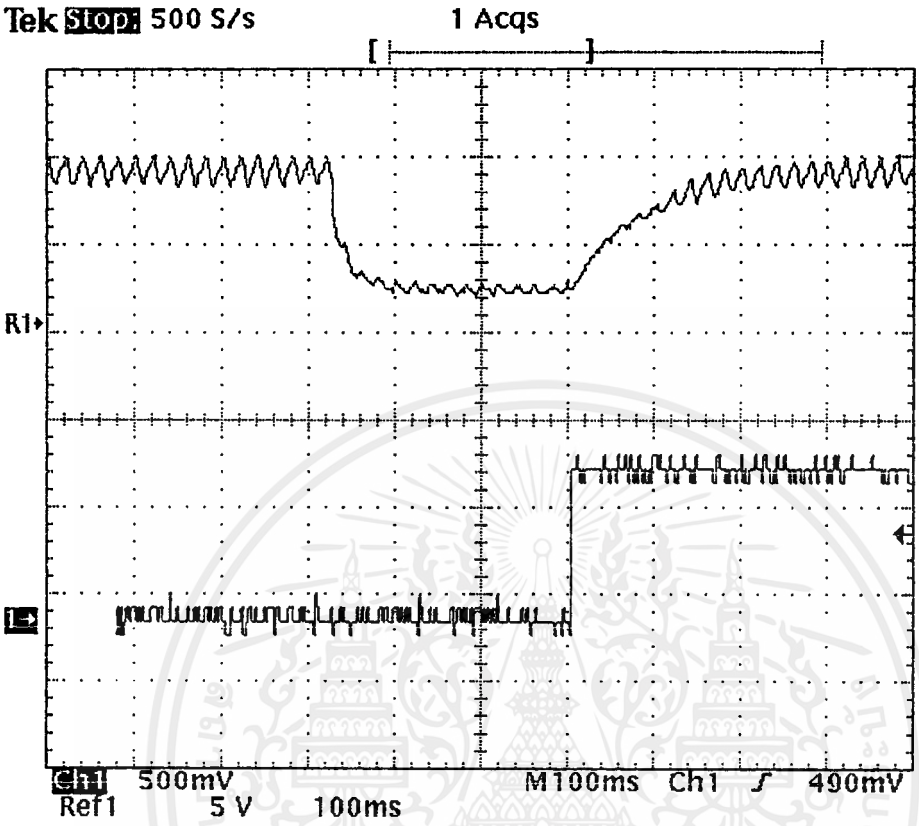
หลังจากที่ได้แบบจำลอง HMM ของแต่ละคำศัพท์แล้ว เมื่อมีลำดับของค่าปรากฏ $O = \{ O_1 O_2 O_3 \ldots O_T \}$ ของเสียง unknown ซึ่งเป็นเสียงที่ต้องการทดสอบเข้ามา เราจะทำการคำนวณหาความน่าจะเป็น $P(O|\lambda)$ ทุกแบบจำลองของแต่ละคำศัพท์โดยใช้วิธี viterbi algorithm แล้วเลือกเอาคำศัพท์ที่มีความน่าจะเป็นสูงสุด ซึ่งก็คือ คำศัพท์ที่แบบจำลองจำได้นั่นเอง



รูปที่ 3.9 แสดง flow chart ของการทำงานของโครงงานทั้งหมด



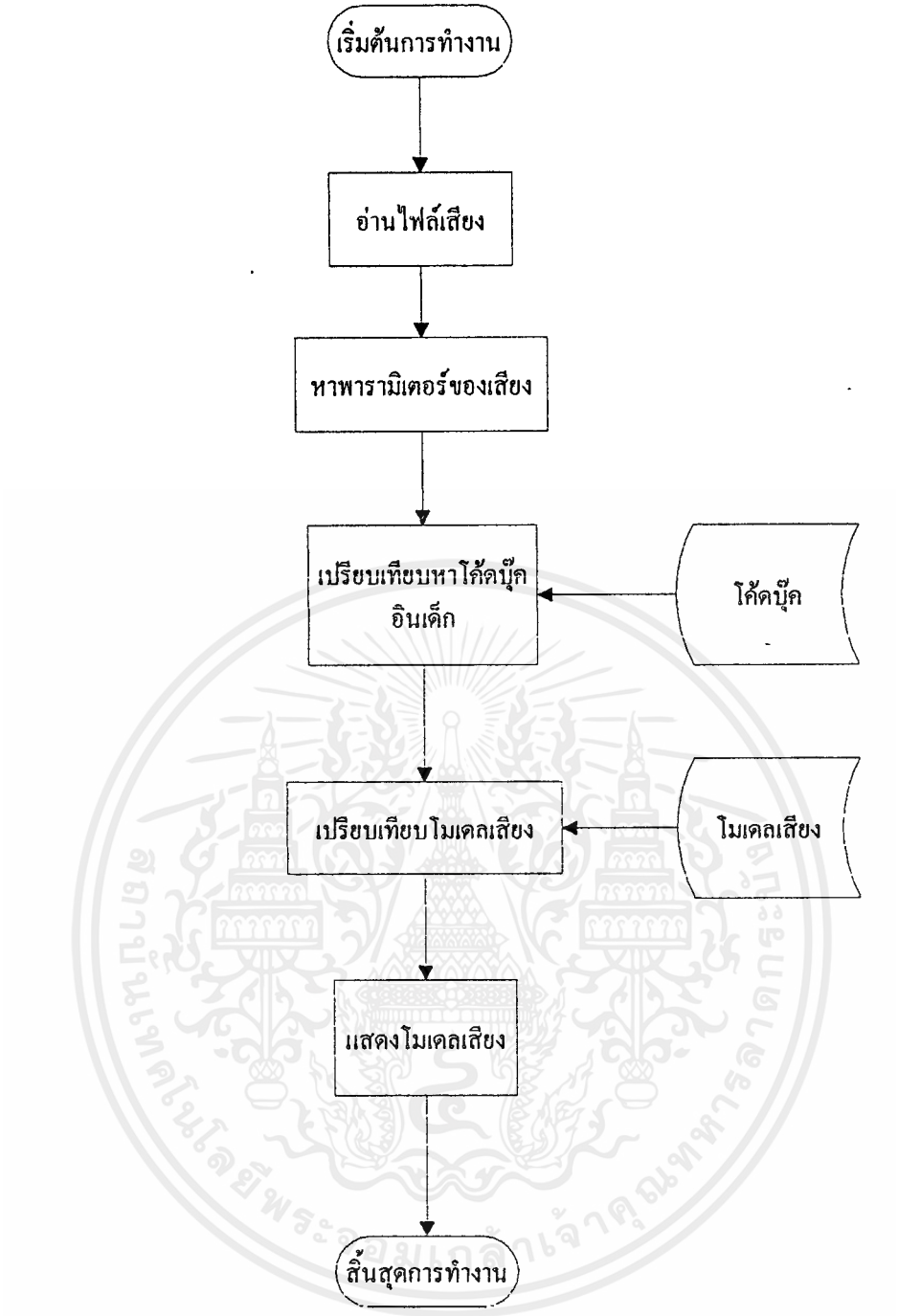
รูปที่ 3.10 แสดงขั้นตอนการเตรียมสัญญาณในการวิเคราะห์



รูปที่ 4.4 แสดงสัญญาณที่เข้าไปในขา CK ของ ดีฟลิปฟลอป และ
แรงดันที่เปลี่ยนไปของขา Q ของดีฟลิปฟลอป
Ref 1 คือ สัญญาณอินพุทที่เข้ามาที่ขา CK ของ ดีฟลิปฟลอป
Ch 1 คือ สัญญาณเอาท์พุทที่ขา Q ของ ดีฟลิปฟลอป

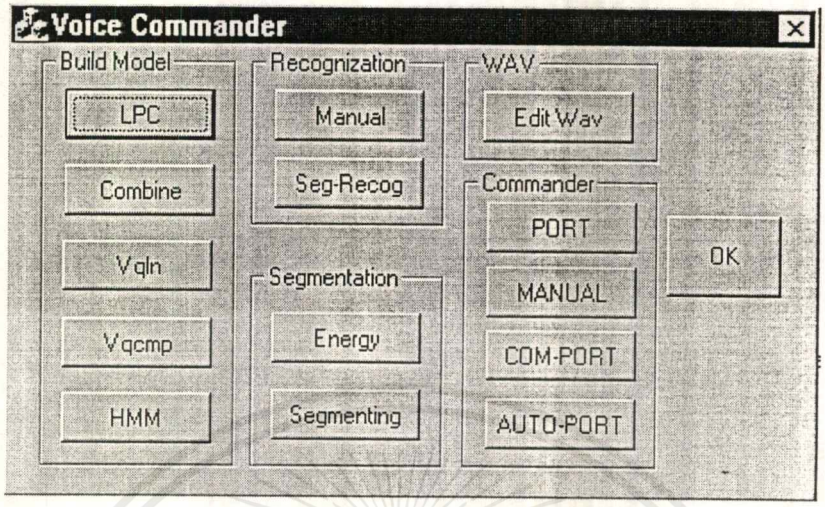
จากรูปที่ 4.4 เราจะเห็นได้ว่าสัญญาณที่เข้าไปชังขาคล็อคของ ดีฟลิปฟลอปจะมีการเปลี่ยนไป และจากทฤษฎีเราก็จะรู้ว่าดีฟลิปฟลอปมีการทำงานที่ขาลง เราจะเห็นได้ว่าเมื่อแรงดันของสัญญาณตกลงมา ขา Q ของดีฟลิปฟลอปก็จะทำการให้ลอจิกที่ขา Q เป็น "1" ก็หมายความว่ามีแรงดันไฟ 12 V ออกมาจากขา Q ของดีฟลิปฟลอป แล้วจากวงจรที่ได้ก็จะนำไปทำให้เกิดกระแสเหนี่ยวนำในทราซิสเตอร์ แล้วทำให้รีเลย์ทำงานต่อไป

<u>หมายเหตุ</u> จากกราฟผลการทดลองทุกรูปเราจะใช้ Probe x10 ทุกรูปดังนั้นค่าแรงดันไฟที่อ่านได้จะต้องคูณ 10 ทุกรูป



รูปที่ 3.12 แผนผังขั้นตอนการทำงานของโปรแกรมวิเคราะห์เสียง

3.2.4 การใช้งานโปรแกรม Voice Commander



รูปที่ 3.13 แสดงโปรแกรมหลักของโครงงาน

โดยที่โปรแกรมหลักจะสามารถแบ่งออกเป็น 5 ส่วนหลักด้วยกัน ดังนี้

3.2.4.1 ส่วนของการสร้างโมเดลเสียง

3.2.4.2 ส่วนของการจดจำเสียง

3.2.4.3 ส่วนของการหาค่าพลังงานและวิเคราะห์การตัดคำเดี่ยว

3.2.4.4 ส่วนของการตัดไฟล์เสียงออกเป็นไฟล์แต่ละคำ

3.2.4.5 ส่วนของการสั่งงานเครื่องใช้ไฟฟ้า

ในแต่ละส่วนจะมีการทำงานดังที่กล่าวมาตามรายละเอียดของโปรแกรม โดยที่ในแต่ละส่วนจะมีการทำงานแยกกันไป โดยจะกล่าวถึงรายละเอียดของการใช้งานดังต่อไปนี้

3.2.4.1 ส่วนของการสร้างโมเดลเสียง ก็จะมีโปรแกรมย่อยอีก 5 ส่วนย่อย ดังนี้

3.2.4.1.1 LPC

3.2.4.1.2 Combine

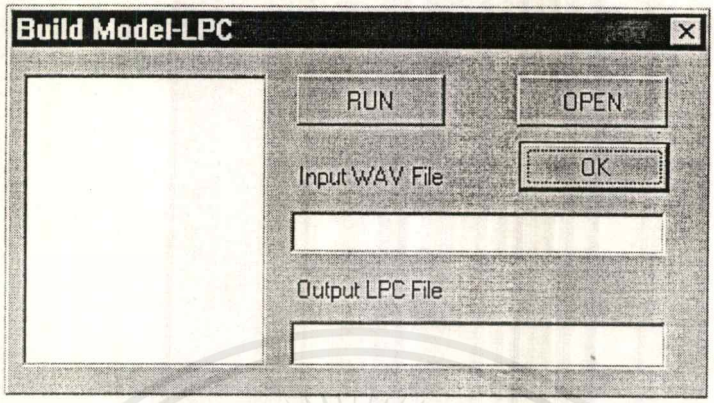
3.2.4.1.3 Vqln

3.2.4.1.4 Cqcmp

3.2.4.1.5 HMM

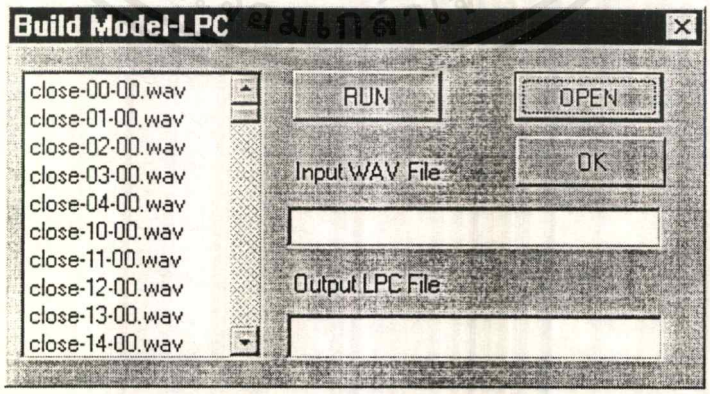
3.2.4.1.1 LPC

โปรแกรมนี้เป็นโปรแกรมย่อยนี้มีหน้าที่ในการหาค่าสัมประสิทธิ์ LPC โดยที่ไฟล์อินพุทจะเป็นไฟล์เสียง ซึ่งในส่วนนี้จะมีรูปร่างดังนี้

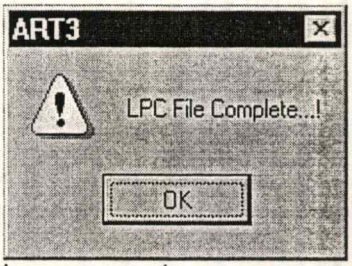


รูปที่ 3.14 แสดงรูปโปรแกรมในส่วนของการหาค่า LPC

การใช้งานนั้นจะมีดังนี้ เราจะเริ่มทำการคัดลอกไฟล์ที่ต้องการลงไปในไดเร็กทอรี่(directory) ที่มีโปรแกรมอยู่ แล้วเราก็กดปุ่ม OPEN โปรแกรมก็จะทำการแสดงไฟล์ที่มีอยู่ในไดเร็กทอรี่ของโปรแกรม โดยที่มันจะแสดงใน "LIST BOX"(บล็อคซ้ายสุด) หลังจากนั้นเราต้องการที่จะหาค่าสัมประสิทธิ์ของ LPC ที่ไฟล์ใดก็ "DOUBLE-CLICK" ที่ไฟล์นั้น ชื่อไฟล์นั้นก็จะขึ้นมาที่อินพุทไฟล์และเอาท์พุทเองอัตโนมัติ หลังจากนั้นเราก็กดปุ่ม RUN โปรแกรมก็จะเริ่มทำการหาค่าสัมประสิทธิ์ LPC พอเสร็จแล้วก็จะมีแมสเสจบล็อคออกมาแสดงให้เรารู้ว่า โปรแกรมได้ทำการคำนวณเสร็จแล้ว และโปรแกรมนี้ได้ถูกออกแบบไว้ให้พิมพ์ออกมาเป็นเท็กซ์ไฟล์(Text File) ด้วย

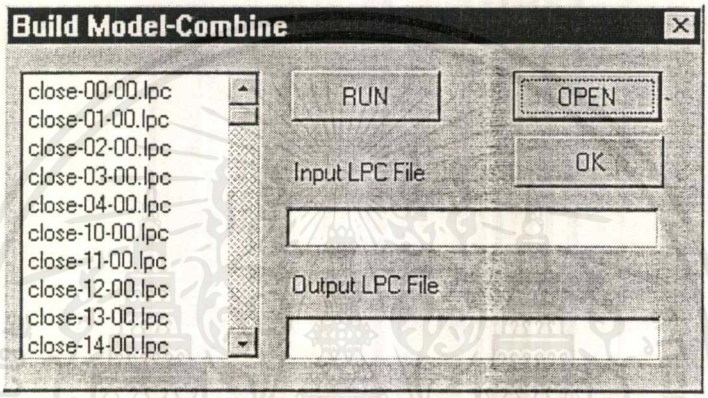


รูปที่ 3.15 แสดงไฟล์เมื่อเราทำการกดปุ่ม OPEN



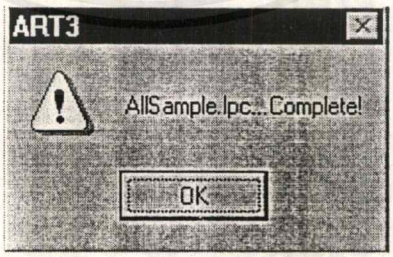
รูปที่ 3.16 แสดงแมสเสจบล็อคที่แสดงออกมาเมื่อโปรแกรมคำนวณหาค่าสัมประสิทธิ์ LPC เสร็จ

3.2.4.1.2 Combine โปรแกรมในส่วนนี้มีหน้าที่ในการรวมไฟล์ค่าสัมประสิทธิ์ LPC ทั้งหมดไว้ในไฟล์เดียว เพื่อใช้ในการคำนวณหาโค้ดบุคต่อไป



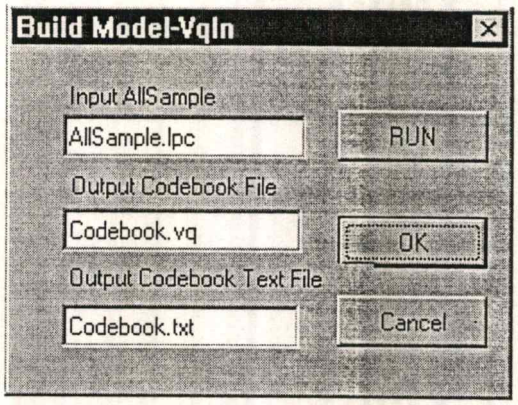
รูปที่ 3.17 แสดงรูปโปรแกรมส่วนของการรวมไฟล์ค่าสัมประสิทธิ์ LPC

ส่วนการใช้งานก็จะคล้ายกับโปรแกรมส่วนของ LPC ปุ่มที่ใช้งานการใช้งานก็จะเหมือนกัน และเมื่อโปรแกรมทำงานเสร็จก็จะมีแมสเสจบล็อคแสดงบอกเหมือนกัน



รูปที่ 3.18 แมสเสจบล็อคเมื่อโปรแกรมทำงานเสร็จ

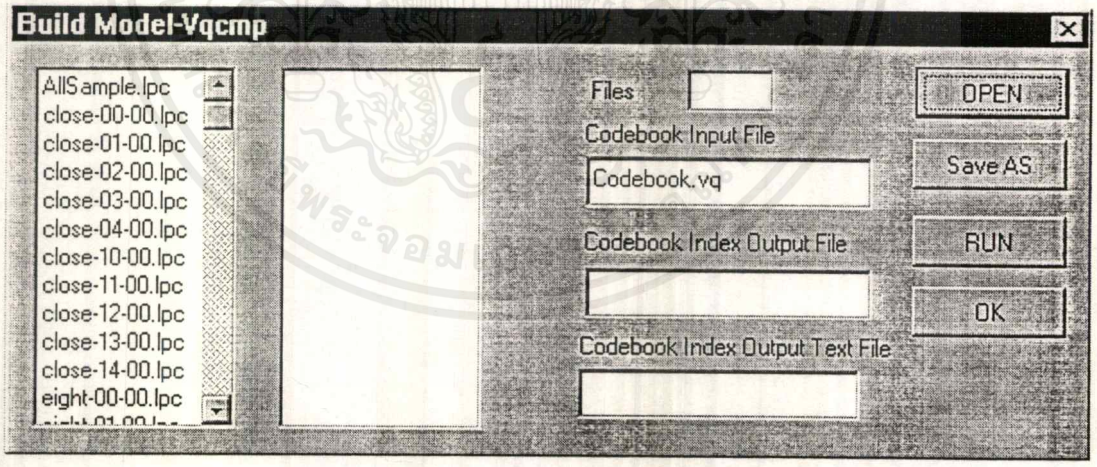
3.2.4.1.3 Vqln โปรแกรมส่วนนี้มีหน้าที่ในการหาค่าการจัดระดับเวคเตอร์ หรือที่เราเรียกว่า Vector Quantization โดยการทำงานในส่วนนี้จะนำเอาค่าไฟล์ที่รวมค่าสัมประสิทธิ์ LPC (AllSample.lpc) นำมาทำการหาโค้ดบุค



รูปที่ 3.19 แสดงรูปโปรแกรมย่อยของ Vqln

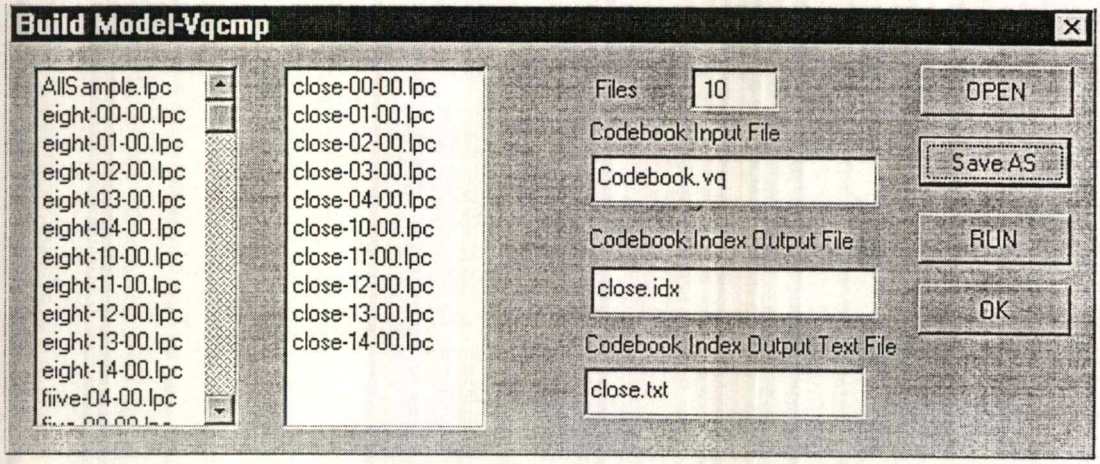
ในส่วนของการทำงานของโปรแกรมในส่วนนี้เราก็จะมีอินพุทไฟล์ "AllSample.lpc" ซึ่งเมื่อมีการเรียกโปรแกรมออกมาได้มีการโปรแกรมให้ขึ้นออกมาอยู่แล้วและเอาท์พุทที่ออกมาโดยปกติก็จะต้องให้ออกมาเป็นไฟล์ "Codebook.vq" และเมื่อโปรแกรมทำงานเสร็จก็จะมีแมสเสจบล็อคออกมาบอกเหมือนกัน

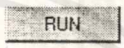
3.2.4.1.4 Vqcmp โปรแกรมในส่วนนี้เป็นการทำการเปรียบเทียบค่าสัมประสิทธิ์ LPC ของแต่ละเสียงกับโค้ดบุคที่ได้ทำการสร้างไว้ เพื่อที่จะได้โค้ดบุคอินเด็ก(Codebook Index) ออกมา



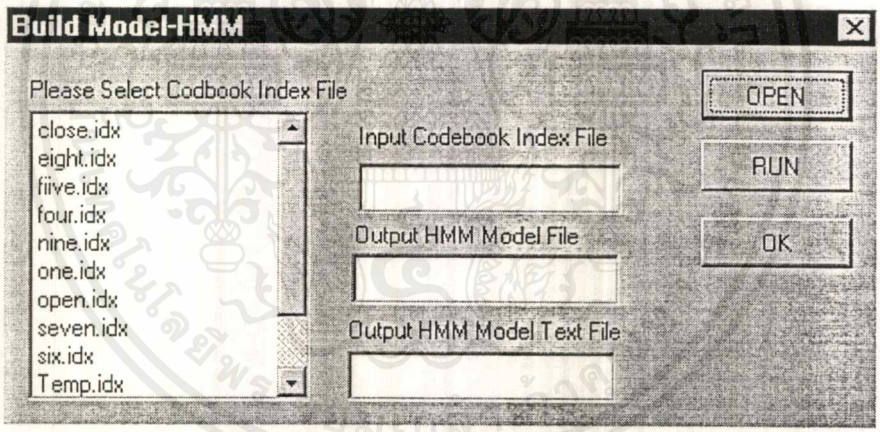
รูปที่ 3.20 แสดงโปรแกรมในส่วนของ Vqcmp

การใช้งานก็จะคล้ายๆกับโปรแกรมที่ได้กล่าวมาข้างต้นแต่จะซับซ้อนกว่าตรงที่ แต่ละเสียงเราอาจจะพูดหลายครั้งหรือหลายคนเราจึงต้องทำบล็อคเอาไว้แสดงไฟล์ต่างหาก และเราจำเป็นที่จะต้องทำการนับจำนวนไฟล์ว่าแต่ละเสียงนำมาคำนวณกี่เสียง และเมื่อเราเลือกไฟล์เป็นที่เรียบร้อยแล้ว เราก็จะทำการกดปุ่ม Save AS โปรแกรมก็จะทำการแสดงเอาท์พุทไฟล์อัตโนมัติ



รปที่ 3.21 แสดงการตั้งค่าต่างพร้อมที่จะกดปุ่ม 

3.2.4.1.5 HMM โปรแกรมย่อยในส่วนนี้จำมีหน้าที่ในการคำนวณหาโมเดลแบบเสียงเพื่อใช้ในการเปรียบเทียบในการจดจำเสียง โดยมีวิธีการแบบ Hidden Markov Model(HMM)



รูปที่ 3.22 แสดงโปรแกรมในส่วนของ HMM

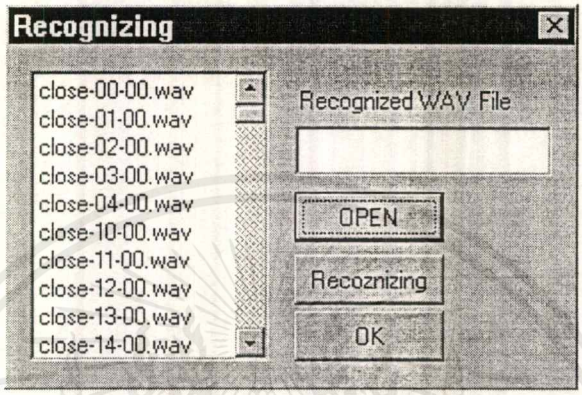
การใช้งานของโปรแกรมในส่วนนี้ก็จะคล้ายๆกับโปรแกรมก่อนหน้านี้ คือในการป้อนค่าอินพุทเราก็สามารถที่จะ Doble-Click ที่ไฟล์ได้เลย แต่ในส่วนของเอาท์พุทโปรแกรมในส่วนนี้ได้ออกแบบให้เราป้อนชื่อของโมเดลไฟล์ตามที่เราต้องการ โดยเราจะใช้นามสกุลของไฟล์เป็น ".hmm" เช่น "1.hmm" ในส่วนของเอาท์พุทที่เป็นเท็กซ์ไฟล์เราได้ออกแบบให้มีชื่อเดียวกับโมเดลไฟล์ โดยจะเป็นโดยอัตโนมัติโดยที่เราไม่ต้องป้อนชื่อไฟล์เอง โดยที่เอาท์พุทเท็กซ์ไฟล์จะมีนามสกุลเป็น ".txt" เช่น "1.txt"

3.2.4.2 ส่วนของการจำเสียง ก็จะมี 2 โปรแกรมย่อยด้วยกัน

3.2.4.2.1 Manual

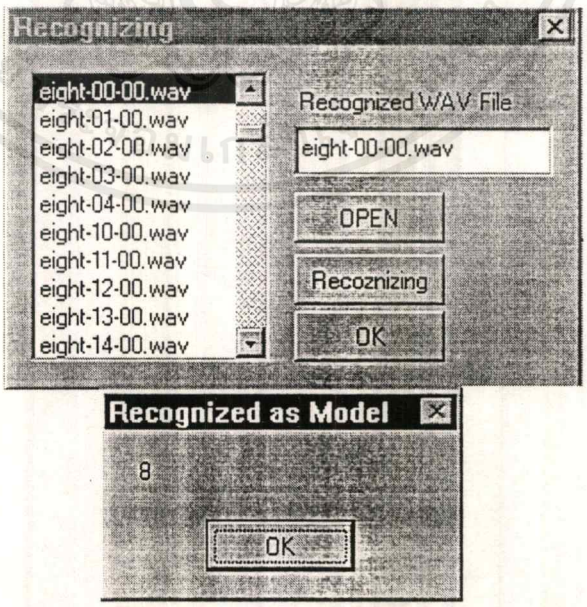
3.2.4.2.2 Seg-Recog

3.2.4.2.1 Manual โปรแกรมนี้เป็นการจดจำเสียงแบบที่เราต้องอัดไฟล์เสียงที่เราต้องการให้จดจำไปเองก่อน แล้วจึงค่อยเรียกโปรแกรมส่วนนี้มาใช้



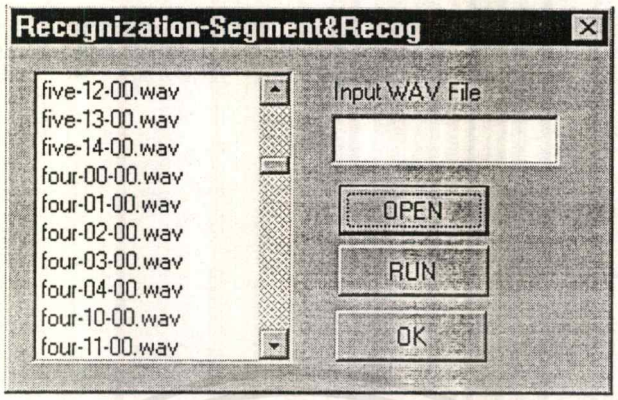
รูปที่ 3.23 แสดงโปรแกรมของการจดจำเสียงแบบ Manual

โดยที่การใช้งานก็จะคล้ายกับโปรแกรมก่อนหน้านี้ เมื่อเราทำการเลือกไฟล์เสียงที่เราต้องการจะให้จดจำได้แล้ว เราก็จะทำการกดปุ่ม Recoznizing เพื่อที่จะทำการจดจำเสียง



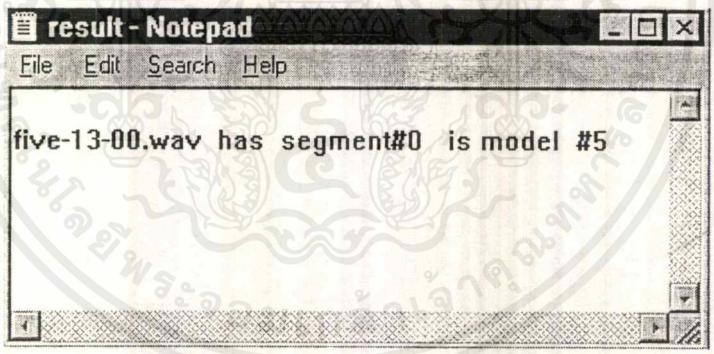
รูปที่ 3.24 แสดงผลลัพธ์ที่ได้จากการจำเสียง

3.2.4.2.2 Seg-Recog เป็นโปรแกรมที่มีหน้าที่ในการจดจำเสียง โดยที่เสียงที่เข้ามาไม่จำเป็นจะต้องเป็นคำเดี่ยวคือโปรแกรมนี้สามารถที่จะทำการตัดคำก่อนแล้วจึงค่อยทำการจดจำเสียง

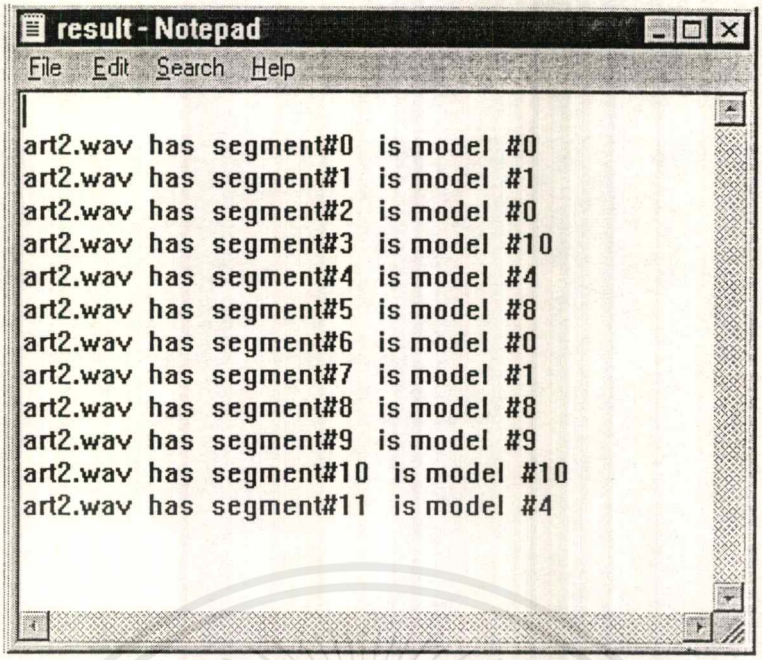


รูปที่ 3.25 แสดงโปรแกรม Seg-Recog

ในส่วนของการใช้งานก็จะเหมือนกับในส่วนของ Manual แต่จะแตกต่างตรงที่เอาท์พุทที่ออกมานั้นจะไม่เป็นแมสเสจบล็อคออกมาแสดงให้เห็นโดยที่ผลลัพท์ที่เราจำได้จะออกมาอยู่ในรูปของไฟล์ที่ชื่อ "result.txt" โดยเวลาที่เราต้องการจะดูผลลัพธ์ของการจำเสียงเราจึงต้องเข้าไปดูที่ไฟล์ตัวนี้



รูปที่ 3.26 แสดงผลลัพท์ที่ได้ออกมาใน "result.txt" ในกรณีอินพุทไฟล์เป็นคำเดี่ยว



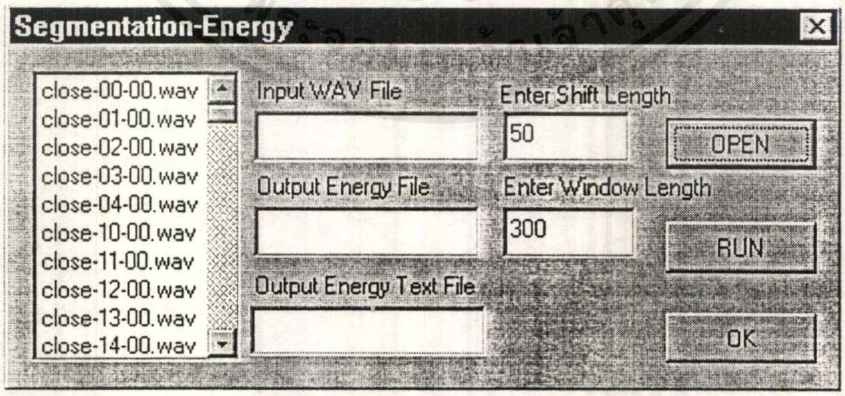
รูปที่ 3.27 แสดงผลลัพท์ที่ได้ออกมาใน "result.txt" ในกรณีอินพุทไฟล์ไม่เป็นคำเดี่ยว

3.2.4.3 ส่วนของการหาค่าพลังงานและวิเคราะห์การตัดคำเดี่ยว การทำงานในโปรแกรมส่วนนี้สามารถที่จะแบ่งออกเป็นโปรแกรมย่อยได้อีก 2 ส่วน

3.2.4.3.1 โปรแกรมในการหาค่าพลังงานของเสียง

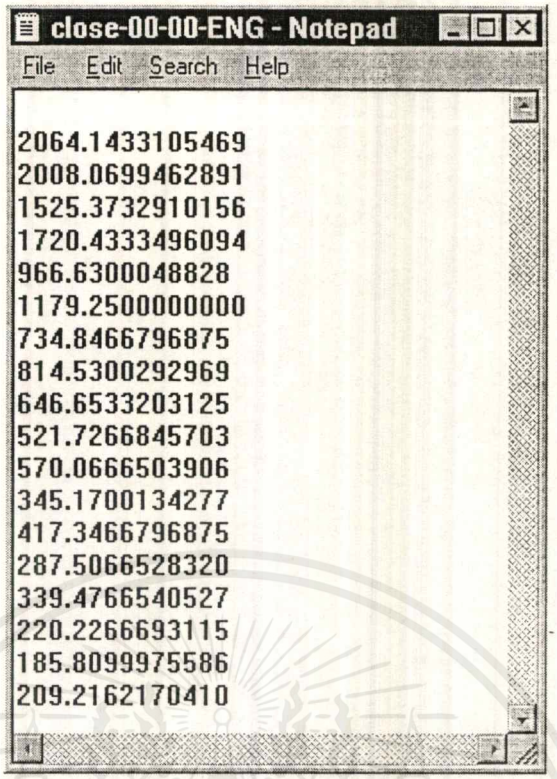
3.2.4.3.2 โปรแกรมในการตัดสินใจในการตัดแบ่งเป็นคำเดี่ยว

3.2.4.3.1 โปรแกรมในการหาค่าพลังงานของเสียง โปรแกรมนี้จะมีหน้าที่ในการหาค่าพลังงานของเสียงโดยใช้วิธีการหาค่าขนาดของเสียงกำลังสองเฉลี่ย



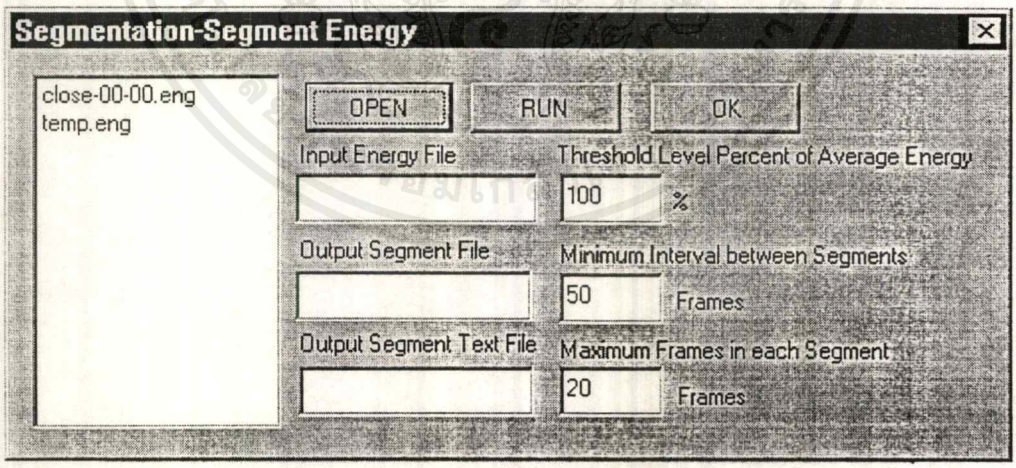
รูปที่ 3.28 แสดงโปรแกรมหาค่าพลังงานของเสียง

โปรแกรมหาค่าพลังงานของเสียง ได้มีการกำหนดค่าวินโดว์และค่าการเลื่อนของวินโดว์ที่เหมาะสมไว้อัตโนมัติ โดยที่เราสามารถเปลี่ยนแปลงแก้ไขค่าต่างๆได้เองตามต้องการ



รูปที่ 3.29 แสดงไฟล์ผลลัพธ์ของการหาค่าพลังงาน

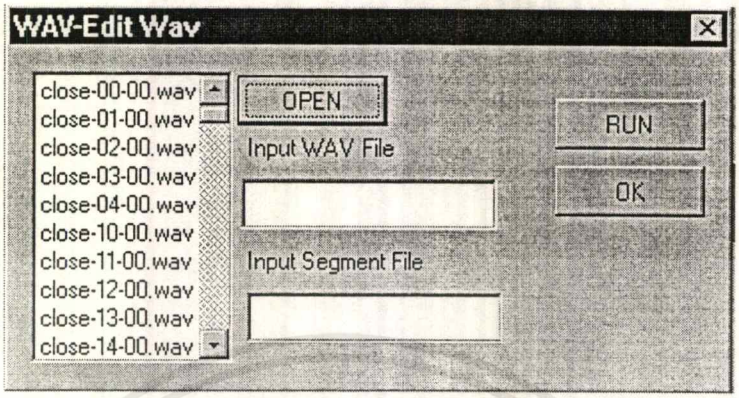
3.2.4.3.2 โปรแกรมในการตัดสินใจในการตัดแบ่งเป็นคำเดี่ยว โดยที่โปรแกรมนี้จะทำงานได้ก็ต่อเมื่อมีการหาค่าพลังงานมาแล้ว โปรแกรมจึงทำการตัดสินใจว่าในเสียงนั้นมีคำเดี่ยวอยู่กี่คำ



รูปที่ 3.30 แสดงโปรแกรม Segment

ในโปรแกรมนี้เราต้องมีการกำหนดค่า Threshold leevel percent of average energy, Minimum interval between segments, maximum frame in each segment โดยเราได้มีการใส่ค่าที่เหมาะสมไว้อยู่โดยอัตโนมัติ แต่เราก็สามารถที่จะกำหนดค่าได้เองก็ได้

3.2.4.4 ส่วนของการตัดไฟล์เสียงออกเป็นไฟล์แต่ละคำ โปรแกรมนี้มีหน้าที่ในการตัดแบ่งเสียงที่เข้ามาให้เป็นคำเดี่ยวย่อยๆได้ โดยที่ไฟล์เสียงของคำเดี่ยวนั้นจะไม่สามารถรับฟังได้แต่เราจะสามารถเอามาคำนวณหาค่าองค์ประกอบของเสียงได้



รูปที่ 3.31 แสดงโปรแกรมแยกเสียงออกเป็นคำเดี่ยว

ในการที่เราจะใช้งานโปรแกรมในส่วนนี้เราจำเป็นที่จะต้องคำนวณหาค่าต่างๆในโปรแกรมหาค่าพลังงานของเสียงและโปรแกรมตัดคำก่อนจึงจะสามารถที่จะทำการตัดแบ่งเป็นไฟล์คำเดี่ยวๆได้

3.3 ส่วนของการเชื่อมต่อกับอุปกรณ์ไฟฟ้า ซึ่งสามารถที่จะแบ่งออกได้อีก 2 ส่วนดังนี้

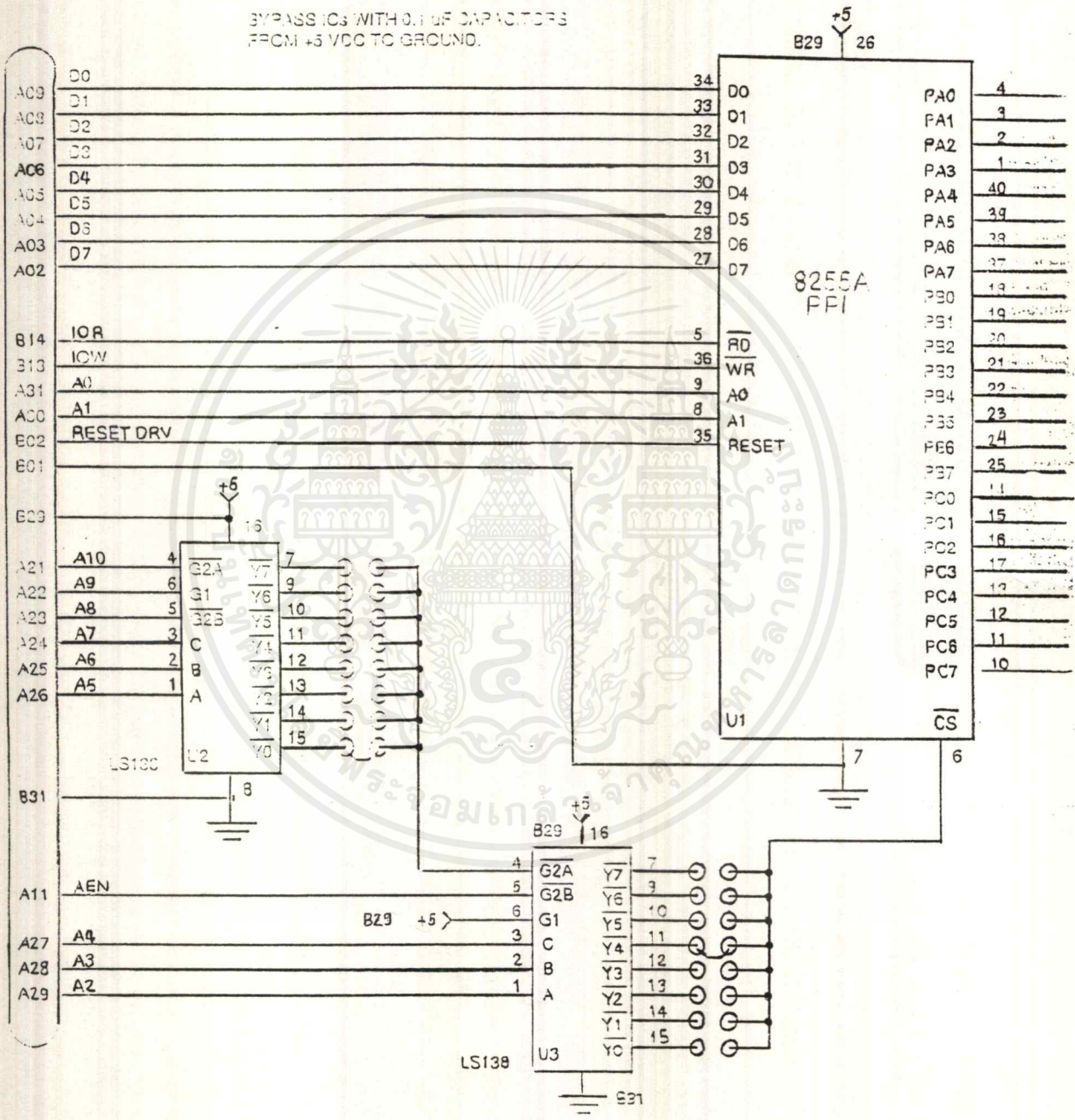
3.3.1 ส่วนของการต่อพอร์ตและการ์ดโปรโตไทป์

3.3.2 ส่วนของวงจรขับรีเลย์

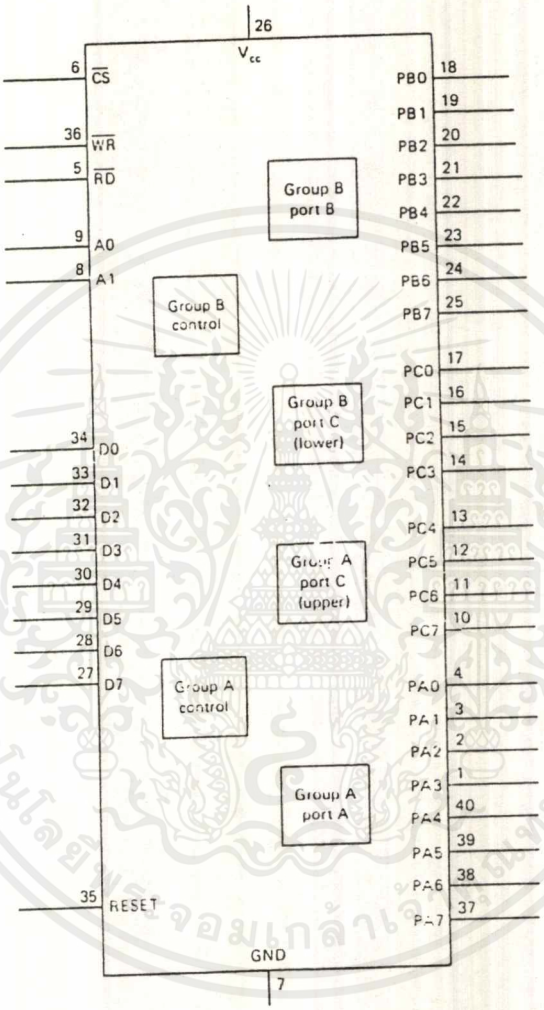
3.3.1 ส่วนของการต่อพอร์ตและการ์ดโปรโตไทป์

ในส่วนของการอินเตอร์เฟสเพื่อใช้ในการต่อพอร์ต เราสามารถทำได้โดยการใช้การ์ดโปรโตไทป์และชิปเบอร์ 8255 ซึ่งสามารถทำหน้าที่ได้ทั้งพอร์ตอินพุท หรือเอาท์พุท ซึ่งมีระดับสัญญาณเป็น TTL เช่นกัน ซึ่งการ์ดโปรโตไทป์จะเป็นการ์ดอินเตอร์เฟสอเนกประสงค์ ซึ่งสามารถที่จะใช้งานกับเครื่อง PC ได้ทุกรุ่น ซึ่งจะต้องนำสัญญาณควบคุมบัสข้อมูล แอดเดรสบัส สัญญาณรีเซต ไฟ VCC และกราวน์ ของเครื่อง PC เชื่อมต่อกับการ์ดให้ถูกต้องเสียก่อน

การต่อสายสัญญาณแอดเดรสบัส ดาต้าบัส สัญญาณควบคุม สัญญาณอินพุทและเอาท์พุท ควรที่จะใช้สายวายแล็บ โดยเราจะทำการต่อกับการ์ดแล้วค่อยใช้บอร์ดเอนกประสงค์ในการลงตัวชิป 8255 แล้วจึงค่อยใช้สายเชื่อมจากตัวการ์ดโปรโตไทป์ไปยังบอร์ดอีกที ทั้งนี้ก็เพื่อความสะดวกในการตรวจเช็ค



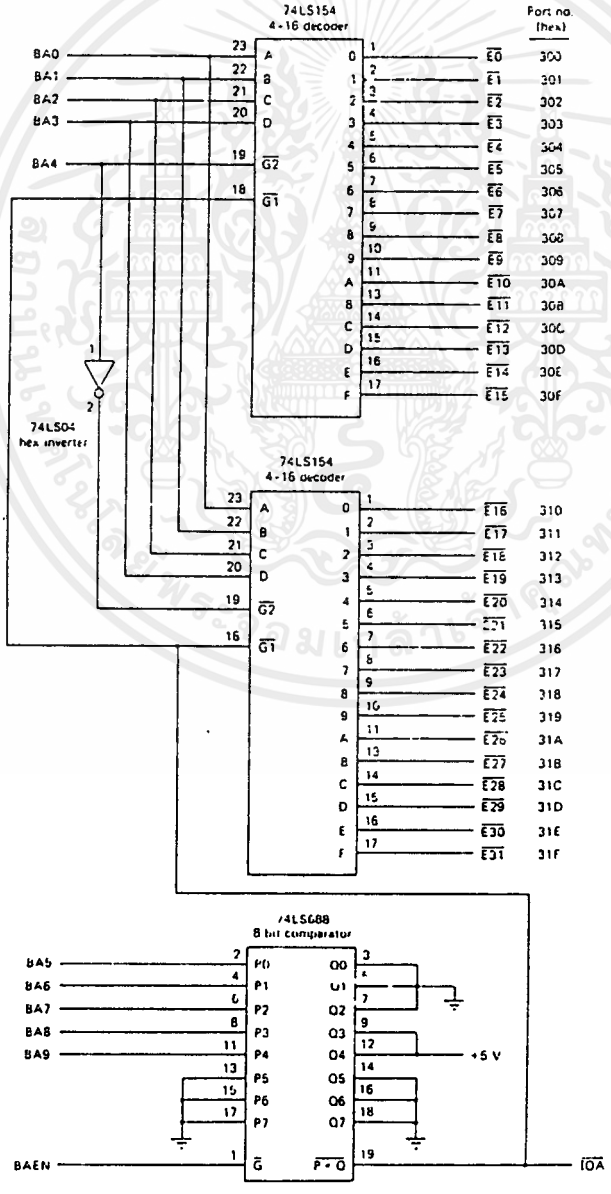
รูปที่ 3.32 แสดงการเชื่อมต่อการ์ดโปรโตไทป์กับตัวชิป 8255



รูปที่ 3.33 แสดงขาต่างๆที่ใช้ในการเชื่อมต่อของตัวชิป 8255

A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	หมายเลขพอร์ต	หมายเหตุ
G1	G2	G2	C_2	B_2	A_2	B_1	A_1				
1	1	0	0	0	0	0	0	0	0	300	พอร์ต A
	1	0	0	0	0	0	0	0	1	301	พอร์ต B
1	1	0	0	0	0	0	0	1	0	302	พอร์ต C
1	1	0	0	0	0	0	0	1	1	303	พอร์ตควบคุม

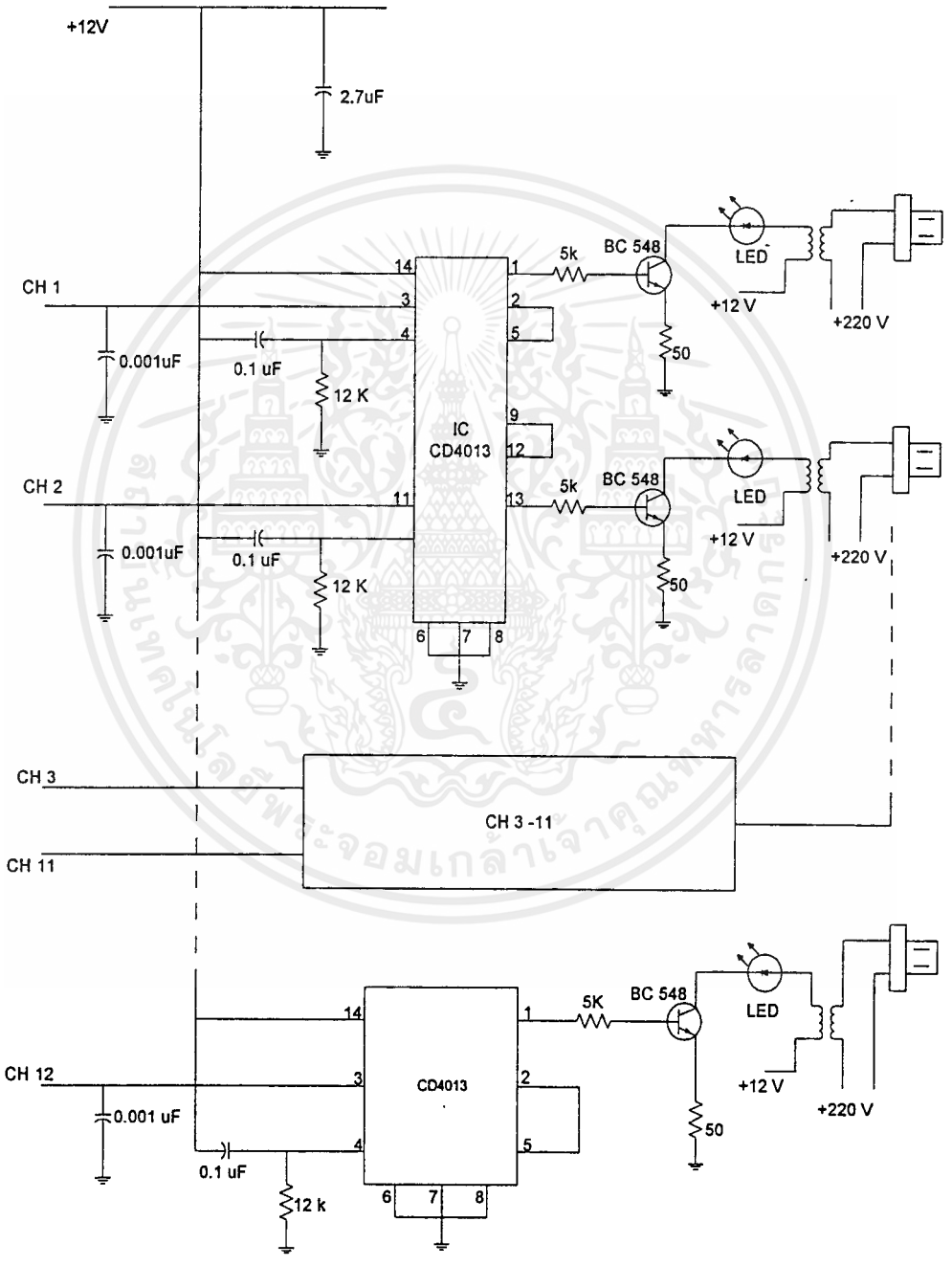
ตารางที่ 3.1 แสดงหมายเลขพอร์ตที่ใช้ในการถอดรหัส



รูปที่ 3.34 แสดงวงจรการ์ดที่ใช้อินเตอร์เฟส

3.3.2 ส่วนของวงจรขับรีเลย์

ในการออกแบบวงจรขับรีเลย์เราออกแบบว่าจะใช้ IC เบอร์ CD4013 ซึ่งเป็นไอซีฟลิปฟลอป ซึ่งไอซีหนึ่งตัวสามารถที่จะขับรีเลย์ได้ 2 ช่องสัญญาณ ในด้านการทำงานก็จะมีอยู่ว่า เมื่อมีสัญญาณอินพุท 5 Volt เข้ามาที่ขาอินพุทของฟลิปฟลอป(ขา 3 และ ขา 11) สัญญาณที่ขาเอาท์พุทของฟลิปฟลอป(ขา 1 และ ขา 13) ก็ จะทำการไบอัสทรานซิสเตอร์เบอร์ BC548 ทำให้รีเลย์ตัวที่ต่อกับขาอีมิตเตอร์ทำงาน และ โฟโต้ไดโอด(LED) ก็จะทำงาน ซึ่งการทำงานจะเป็นอย่างนี้ในทุกๆช่องของการทำงาน



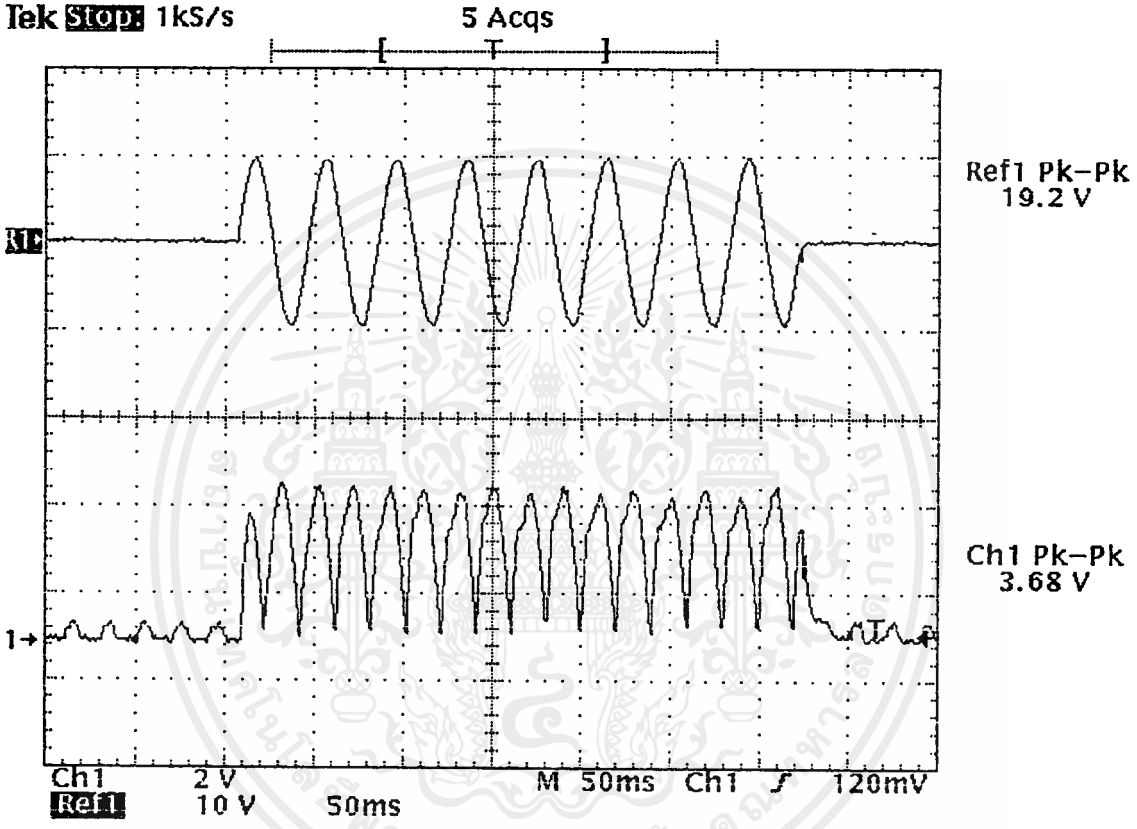
รูปที่ 3.35 แสดงรูปวงจรขับรีเลย์ที่ได้ออกแบบใช้งาน

บทที่ 4

การทดลองและผลการทดลอง

4.1 ผลการทดลองในส่วนเครื่องตอบรับโทรศัพท์

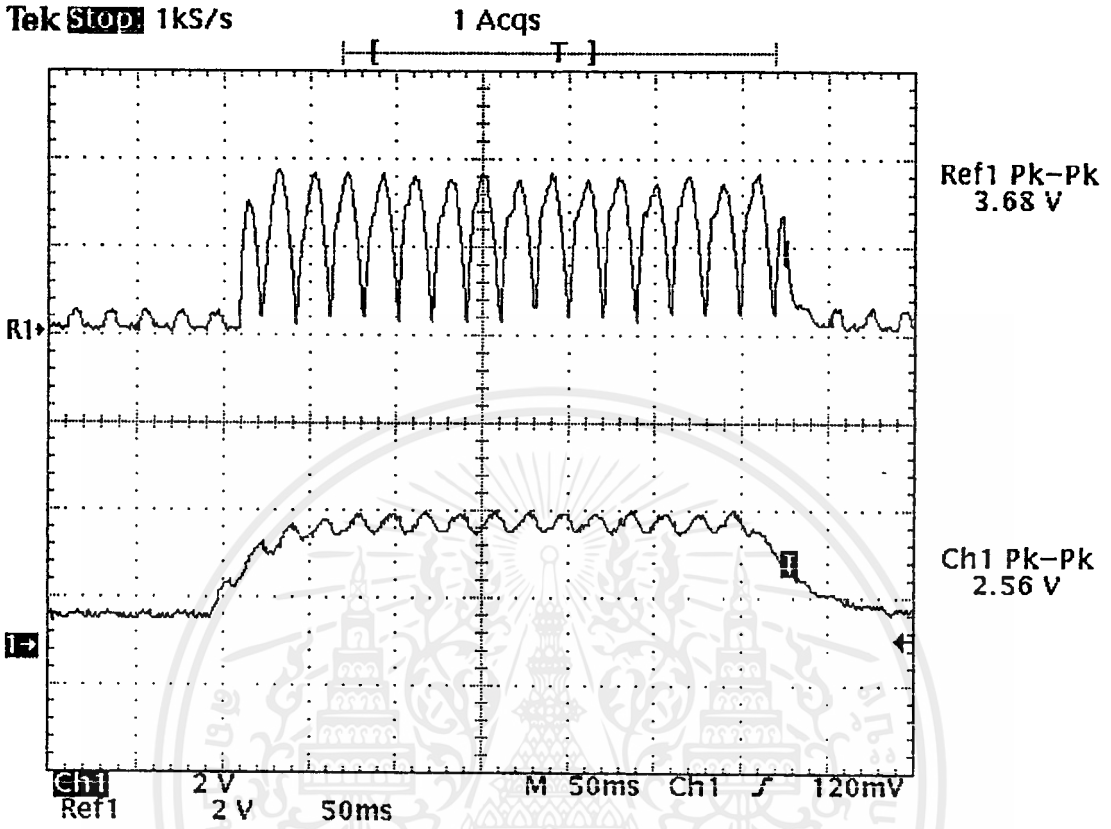
ในส่วนของเครื่องตอบรับโทรศัพท์เราสามารถทดสอบในส่วนต่างๆตามหลักการที่กล่าวมาในบทที่ 3 โดยเราจะทำการตรวจวัดสัญญาณในแต่ละส่วน โดยจะมีผลที่ได้ดังนี้



รูปที่ 4.1 รูปของสัญญาณอินพุทเทียบกับสัญญาณที่ออกจากวงจรถ่ายทอดสัญญาณกระดิ่ง
Ref 1 คือ สัญญาณ Ringing ที่เข้าไปในวงจร ฟูลเวฟเรคติไฟเออร์
Ch 1 คือ สัญญาณเอาท์พุทที่ออกจากวงจร ฟูลเวฟเรคติไฟเออร์

จากรูปที่ 4.1 จะเห็นว่าสัญญาณกระดิ่งที่มีลักษณะเป็นกระแสสลับมีขนาดประมาณ 192 Vp-p ซึ่งเป็นค่าใกล้เคียงตามทฤษฎี โดยปกติสัญญาณกระดิ่งจะมีแรงดันไฟกระแสสลับ 100 Vp หรือ 200 Vp-p เมื่อผ่านวงจรตรวจจับสัญญาณกระดิ่ง ก็จะได้ลักษณะของสัญญาณออกมาเหมือนเดิม และ เมื่อผ่านวงจรบริดจ์เรคติไฟร์แล้ว ส่วนของสัญญาณในซีกที่เป็นลบจะถูกกลับเฟสให้เป็นบวก เพราะตามทฤษฎีแล้ววงจรบริดจ์เรคติไฟเออร์จะยอมให้ไฟกระแสตรงผ่านเท่านั้น

จากรูปที่ 4.1 จะเห็นว่าจะมีการกระเพื่อมของสัญญาณ (Ripple) ซึ่งเราจำเป็นที่จะต้องนำไปผ่านวงจรกรองสัญญาณต่อไป

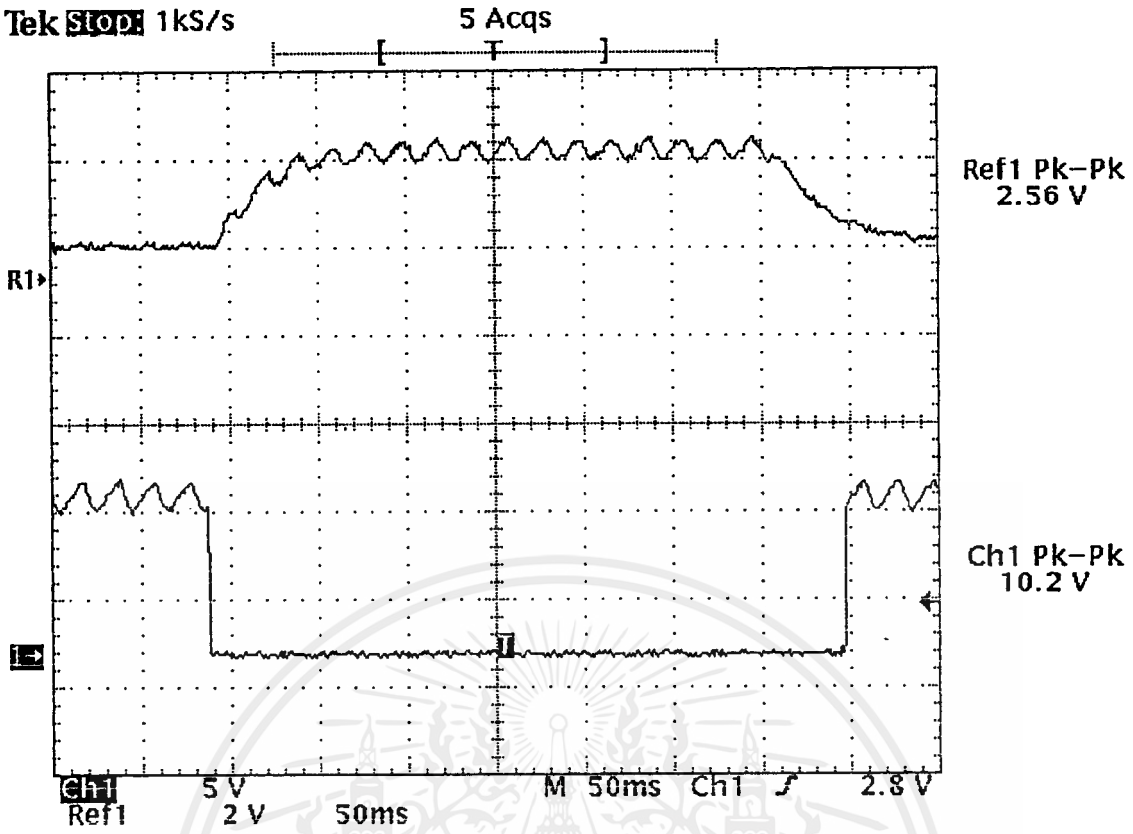


รูปที่ 4.2 แสดงการเปรียบเทียบของสัญญาณที่จะเข้าไปในวงจรกรองสัญญาณ และสัญญาณที่ออกจากวงจรกรองสัญญาณ

Ref 1 คือ สัญญาณที่ออกมาจากวงจรเรคติไฟเออร์แล้วเป็นอินพุทในวงจรกรองสัญญาณความถี่ต่ำผ่าน

Ch 1 คือ สัญญาณเอาท์พุทที่ออกมาจากวงจรกรองสัญญาณความถี่ต่ำผ่าน

จากรูปที่ 4.2 สัญญาณที่ผ่านวงจรบริดเรคติไฟเออร์มาแล้วจะเห็นได้ว่ามีการกระเพื่อมของสัญญาณ ดังนั้นเราจึงจำเป็นที่จะต้องทำการลดการกระเพื่อมของสัญญาณลงโดยใช้วงจรกรองสัญญาณ และผลที่ได้ออกมาจากวงจรกรองสัญญาณจะมีการกระเพื่อมของสัญญาณน้อยลง ซึ่งมีลักษณะใกล้เคียงกับไฟตรง และสามารถทำให้มีกระแสไหลเข้าไปในขาเบสของตัวทรานซิสเตอร์ได้ และสามารถทำให้ทรานซิสเตอร์ทำงานตามขั้นตอนต่อไป

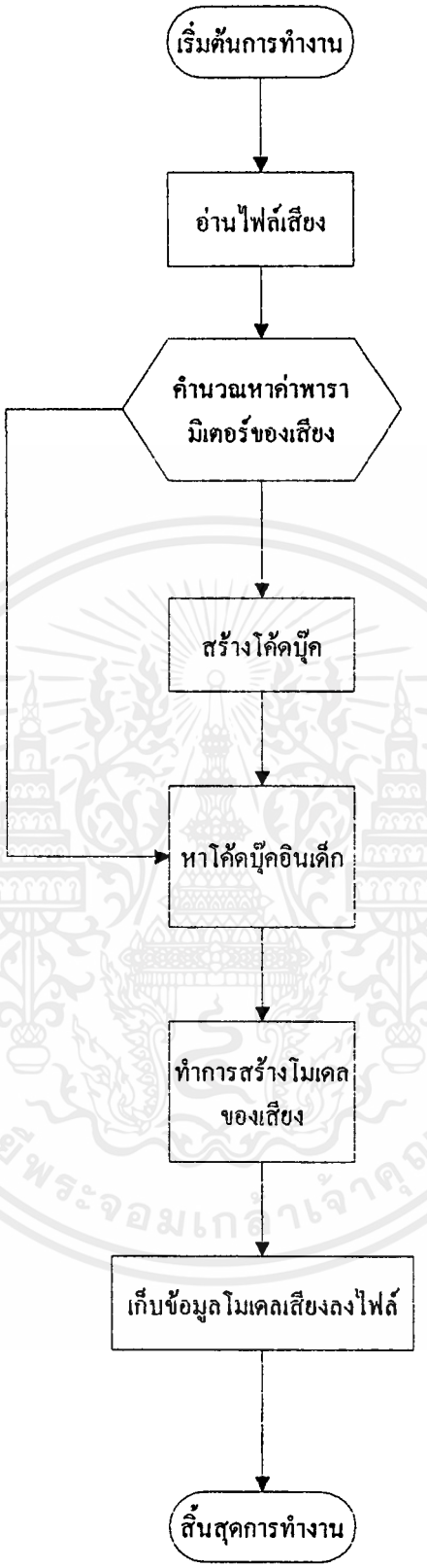


รูปที่4.3 แสดงสัญญาณที่เข้าไปที่ทรานซิสเตอร์และลักษณะของสัญญาณแรงดันที่เปลี่ยนไปที่ขาคอลเล็กเตอร์ของทราซิสเตอร์

Ref 1 คือ สัญญาณที่ออกมาจากวงจรกรองความถี่ต่ำผ่านแล้วเป็นอินพุทที่ขาเบสของสวิตช์ทรานซิสเตอร์

Ch 1 คือ แรงดันที่ขาคอลเล็กเตอร์ของทรานซิสเตอร์

จากรูปที่ 4.3 สัญญาณที่มีการกระเพื่อมเล็กน้อยที่เข้าไปยังขาเบสและจากกระแสเหนี่ยวนำนี้จะเสมือนกับว่ามีการช็อตระหว่างขาคอลเล็กเตอร์และขาอีมิเตอร์ จากรูปจะเห็นได้ว่าเมื่อมีแรงดันไฟเข้ามายังขาเบส จะทำให้โวลเตจที่ตกคร่อมขาคอลเล็กเตอร์ตกลง จาก 12 V เป็น 0 Vก็หมายความว่าได้มีการช็อตระหว่างขาคอลเล็กเตอร์กับขาอีมิตเตอร์เป็นที่เรียบร้อยแล้ว และเมื่อแรงดันไฟที่เข้ามาที่ขาเบสของทราซิสเตอร์หมดไป ก็จะไม่มีกระแสไหลเข้ามาที่ทรานซิสเตอร์ ก็จะไม่มีการช็อตกันระหว่างขาคอลเล็กเตอร์และอีมิตเตอร์ เราจะสามารถสังเกตได้จากกราฟว่าแรงดันไฟตกคร่อมที่ขาคอลเล็กเตอร์จะกลับมาเป็น 12 V อีกครั้ง



รูปที่ 3.11 แผนผังขั้นตอนการทำงานของโปรแกรมเรียนรู้เสียงและโปรแกรมสร้างโมเดลเสียง

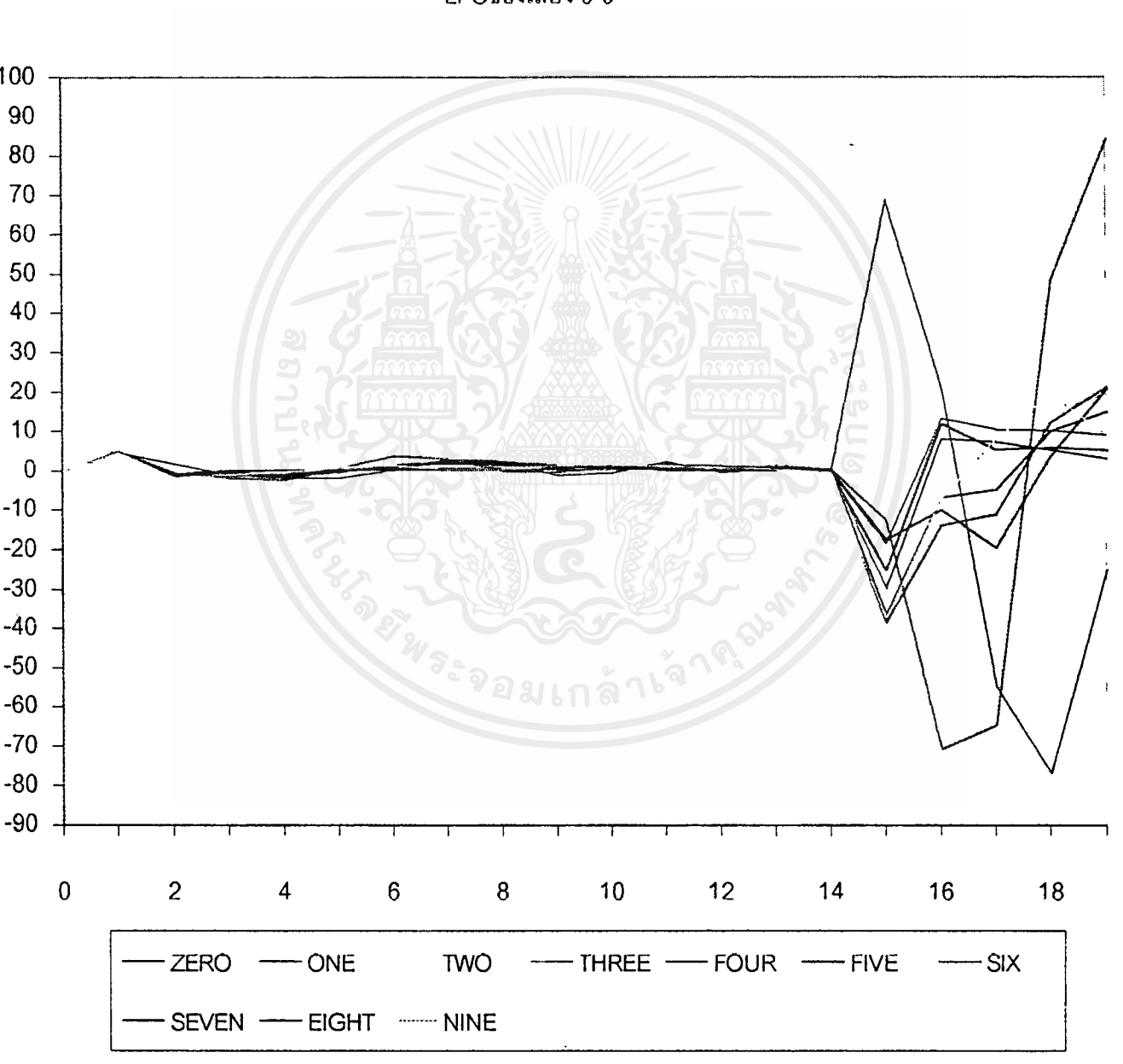
ตัวเลข	Std	Q4	Q3	Q2	Q1
1	H	0	0	0	1
2	H	0	0	1	0
3	H	0	0	1	1
4	H	0	1	0	0
5	H	0	1	0	1
6	H	0	1	1	0
7	H	0	1	1	1
8	H	1	0	0	0
9	H	1	0	0	1
0	H	1	0	1	0
*	H	1	0	1	1
#	H	1	1	0	0

ตารางที่ 4.1 แสดงผลการทดลองวัดค่าตรรกะที่ขาต่างๆในการกดแต่ละปุ่มโทรศัพท์

จากตารางที่ 4.1 เราจะเห็นได้ว่าในบทที่ 3 เราออกแบบให้ใช้ แป้น # เป็นตัวกำหนดในการตัดสายโทรศัพท์ ดังนั้นในบทที่ 3 เราจึงใช้ เอาท์พุทที่ขา Q3, Q4 และ Std เพราะค่าตรรกะที่ออกมาของทั้ง 3 ขาในการกดปุ่ม # มีค่าเป็น 1 ทั้งหมด

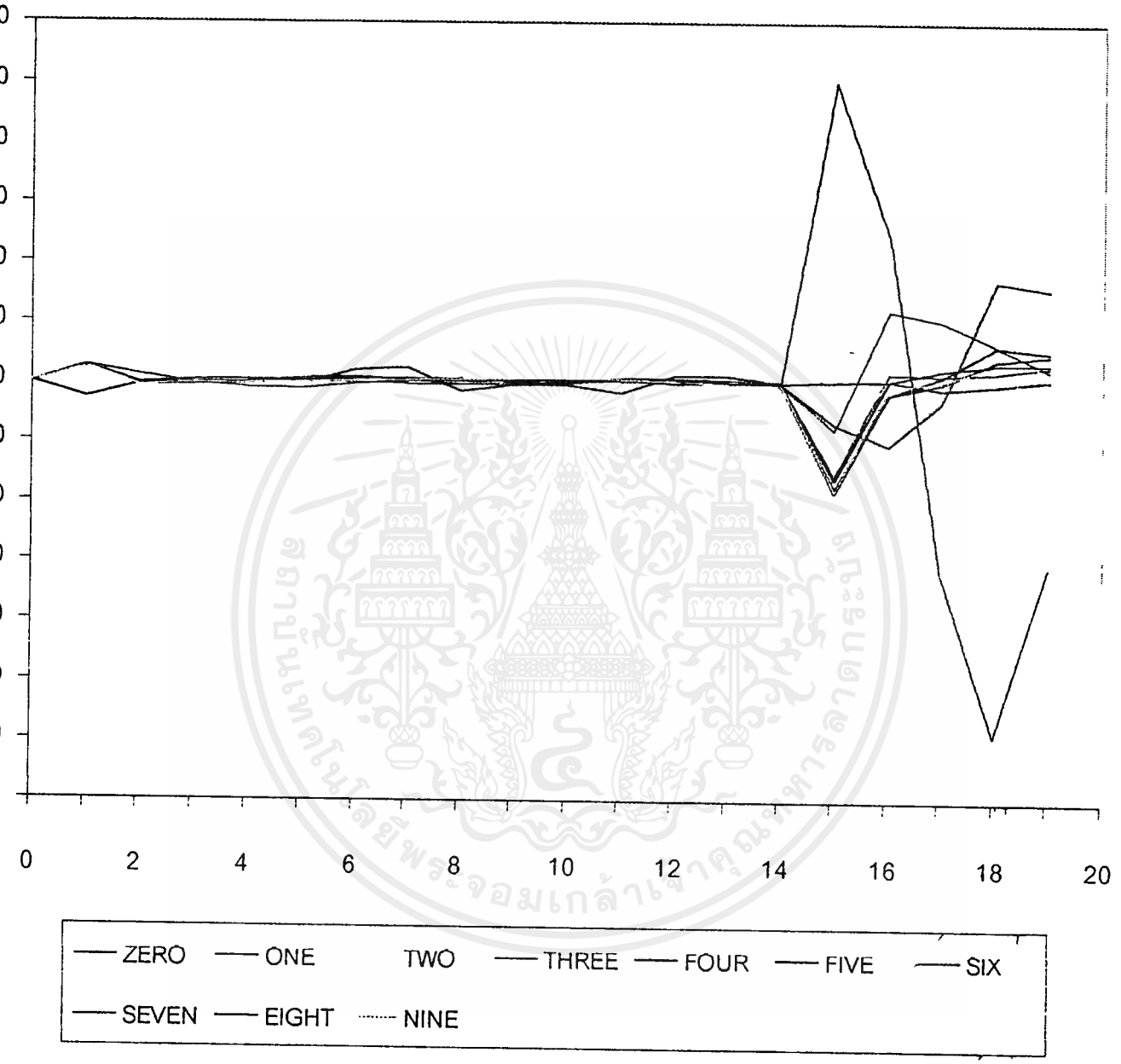
4.2 ผลการทดลองในส่วนของโปรแกรม

จากการทดลองที่ได้ให้โปรแกรมทำการคำนวณหาค่าสัมประสิทธิ์ LPC ค่าแกน ค่าCoefficient และค่าคุณลักษณะต่างๆของเสียงก็จะได้ค่าออกมาเป็นตัวเลขแล้วจึงนำมาทำการพล็อตกราฟ โดยค่าต่างๆ ของสัญญาณเสียงที่นำมาทำการพล็อตนั้น เราจะนำมาจากเฟรมแรกของทุกๆเสียง ทั้งนี้ก็เนื่องมาจากเสียงหนึ่งเสียงมีค่าต่างๆมาก เราจึงไม่สามารถที่จะนำมาทำการคำนวณได้หมดเราจึงเอามาจากแค่เฟรมแรกของทุกๆเสียง

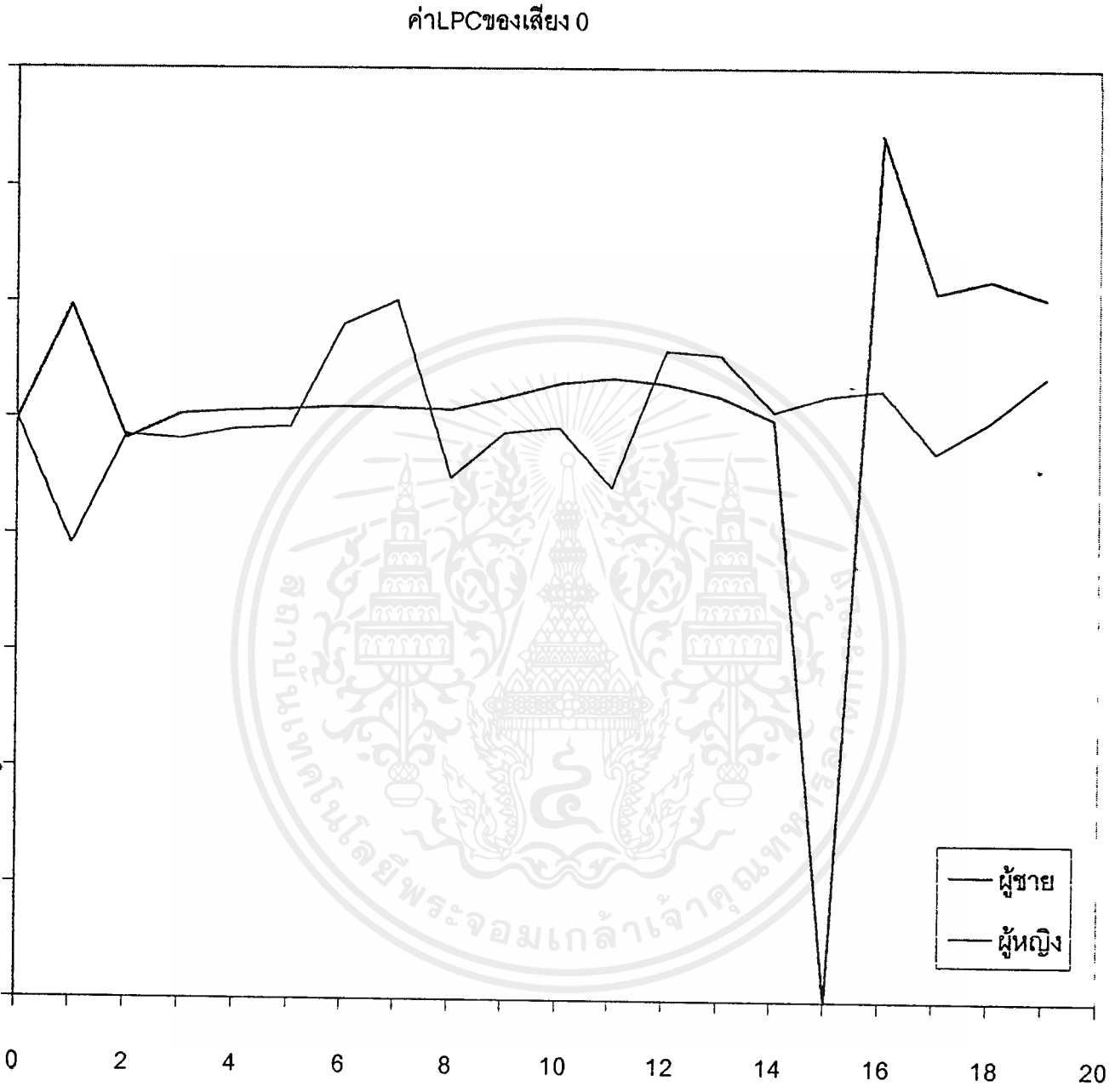


รูปที่ 4.5 แสดงค่า LPC เสียง 0-9 ของผู้ชาย

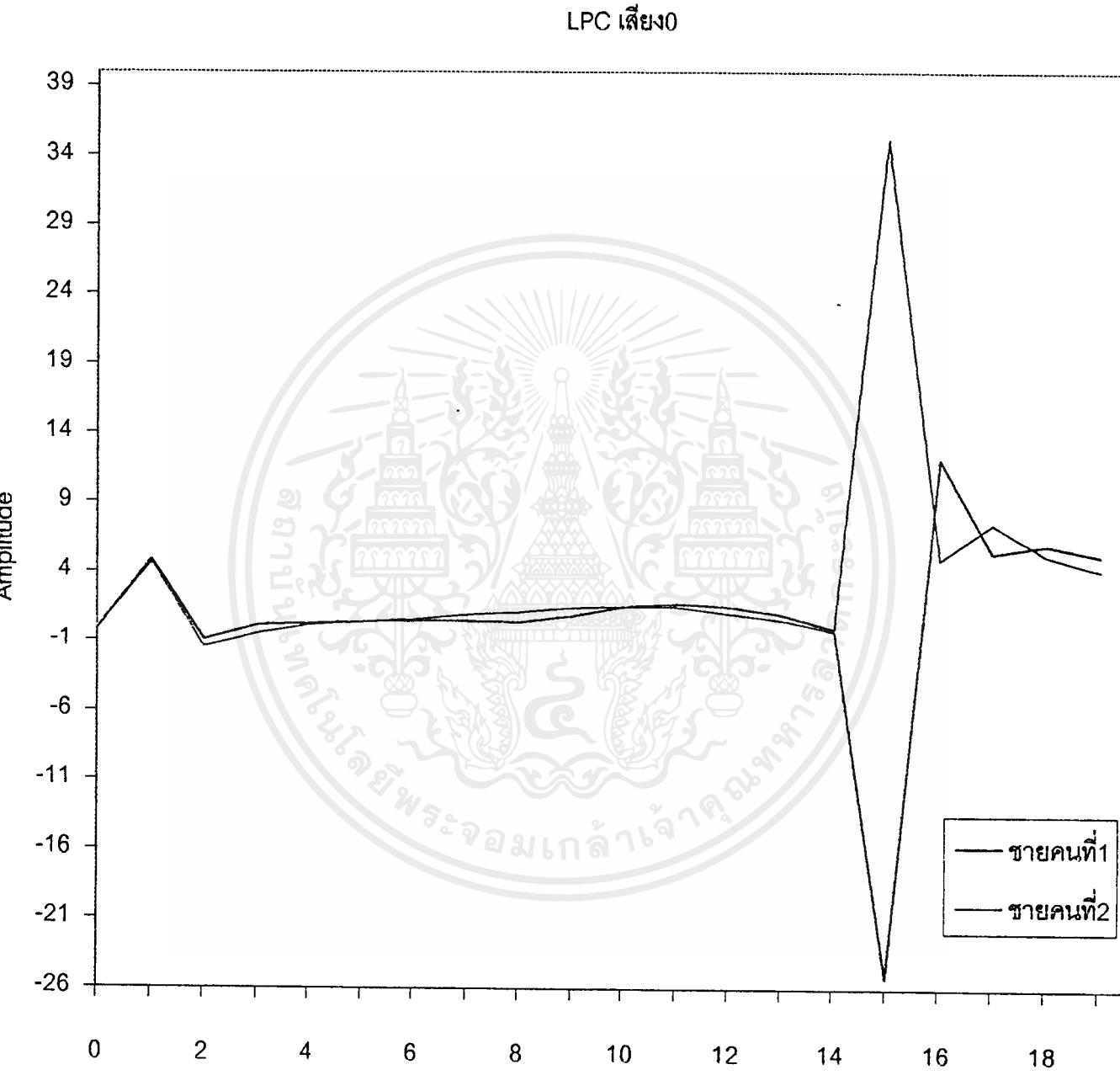
ค่า LPC เสียง 0-9



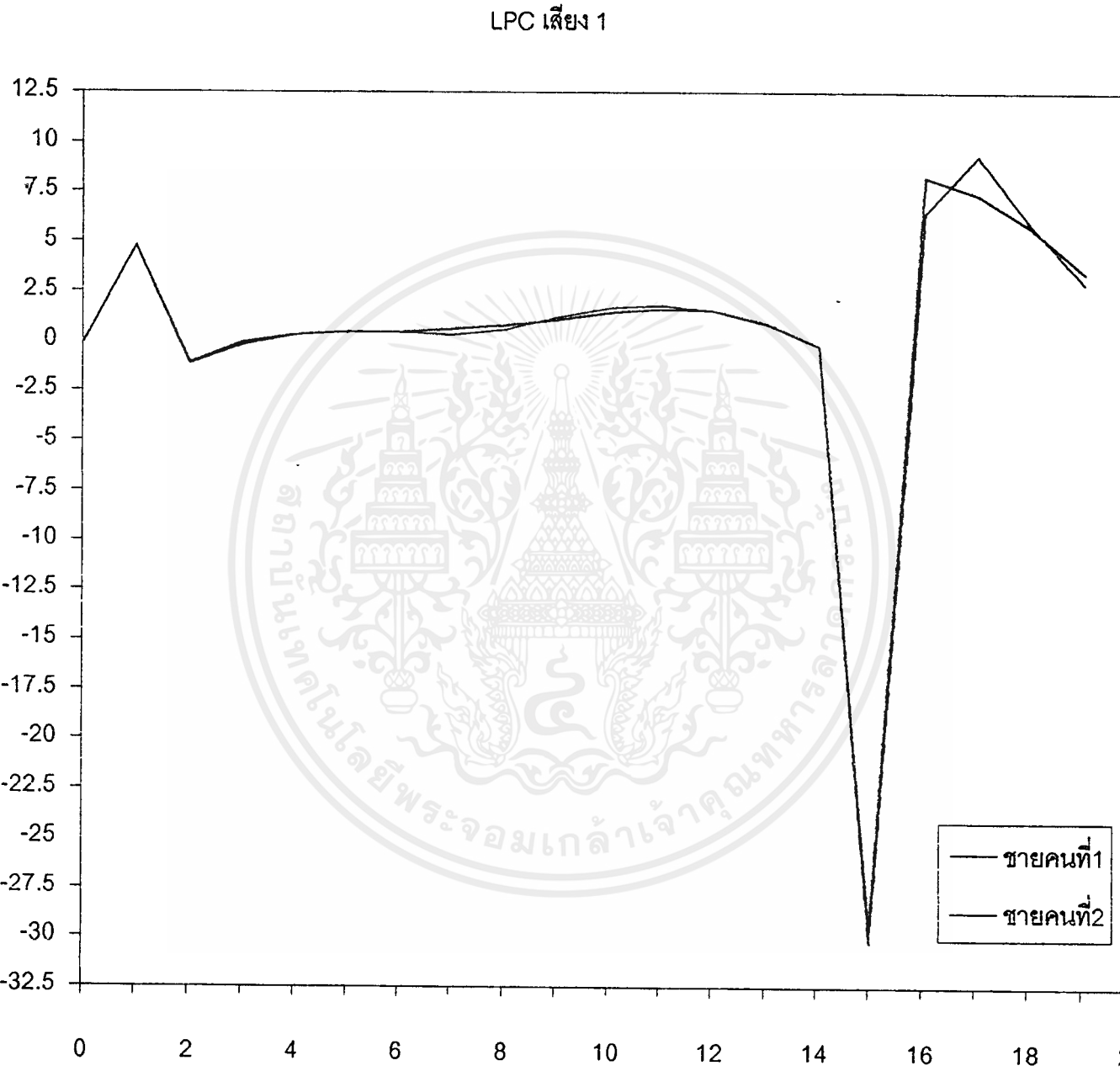
รูปที่ 4.6 แสดงค่า LPC เสียง 0-9 ของผู้หญิง



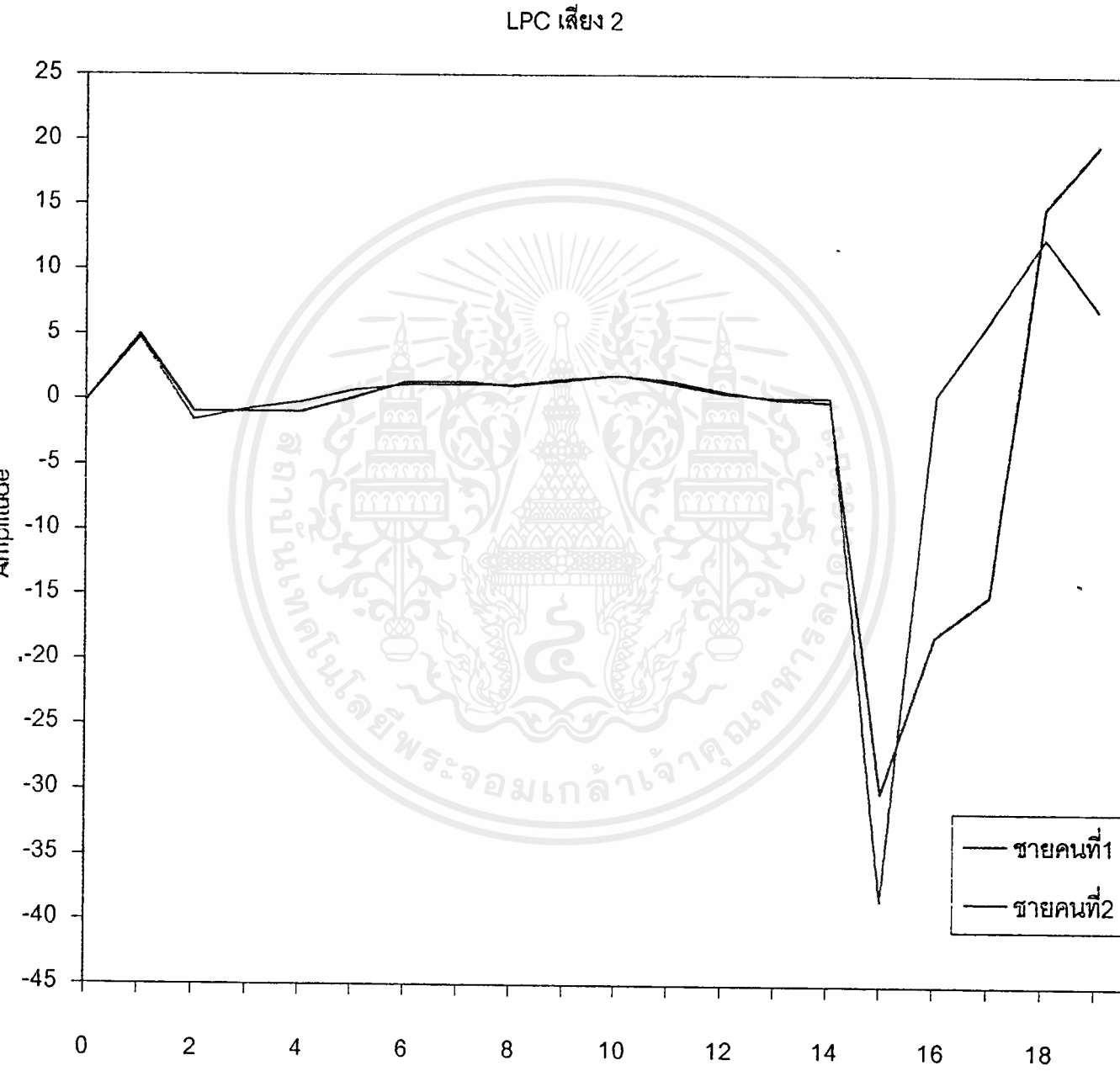
รูปที่4.7 แสดงค่า LPC เสียง 0 ของผู้ชายเทียบกับผู้หญิง



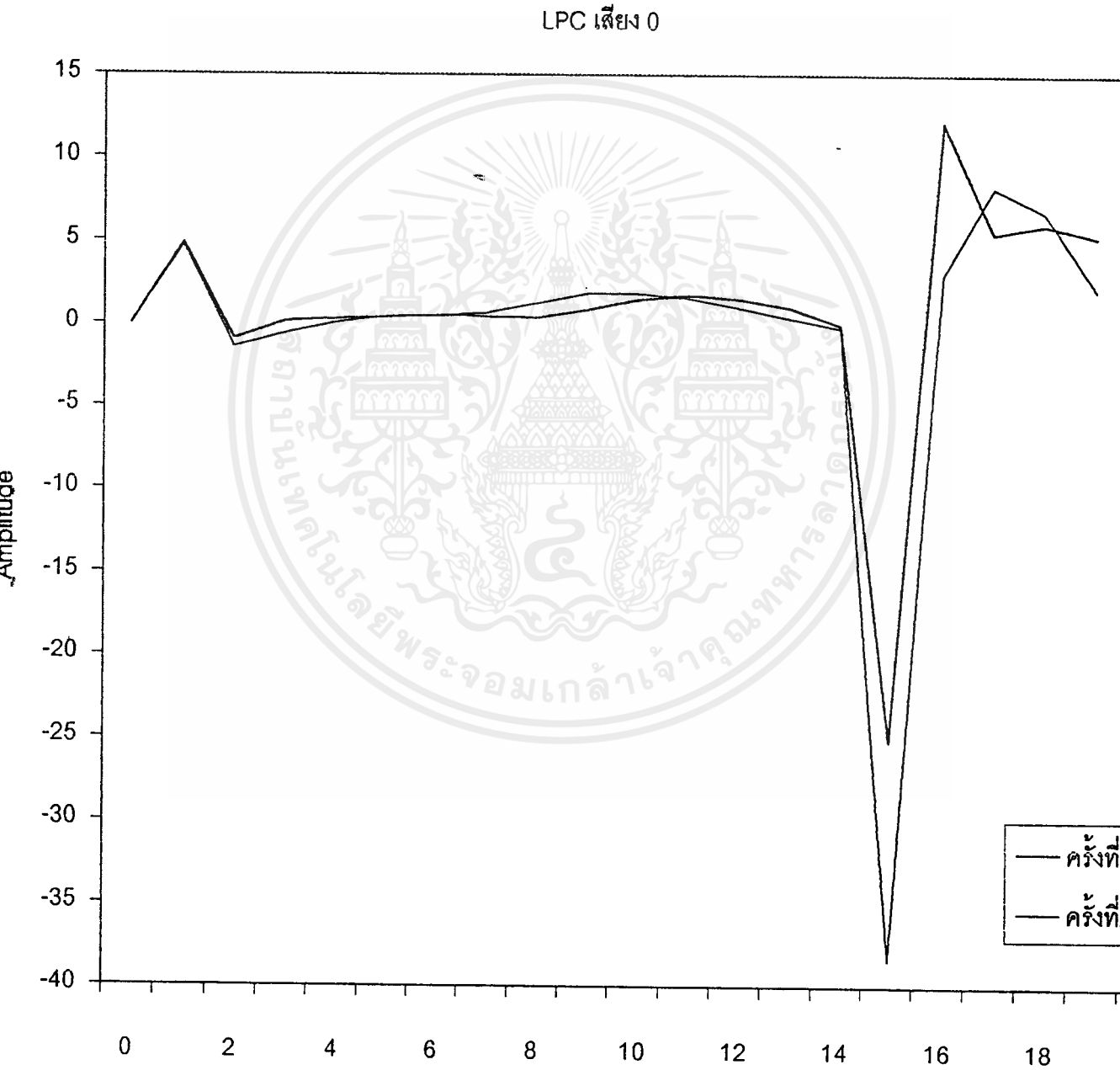
รูปที่ 4.8 แสดงค่า LPC เสียง 0 ของชายคนที่ 1 เทียบกับชายคนที่ 2



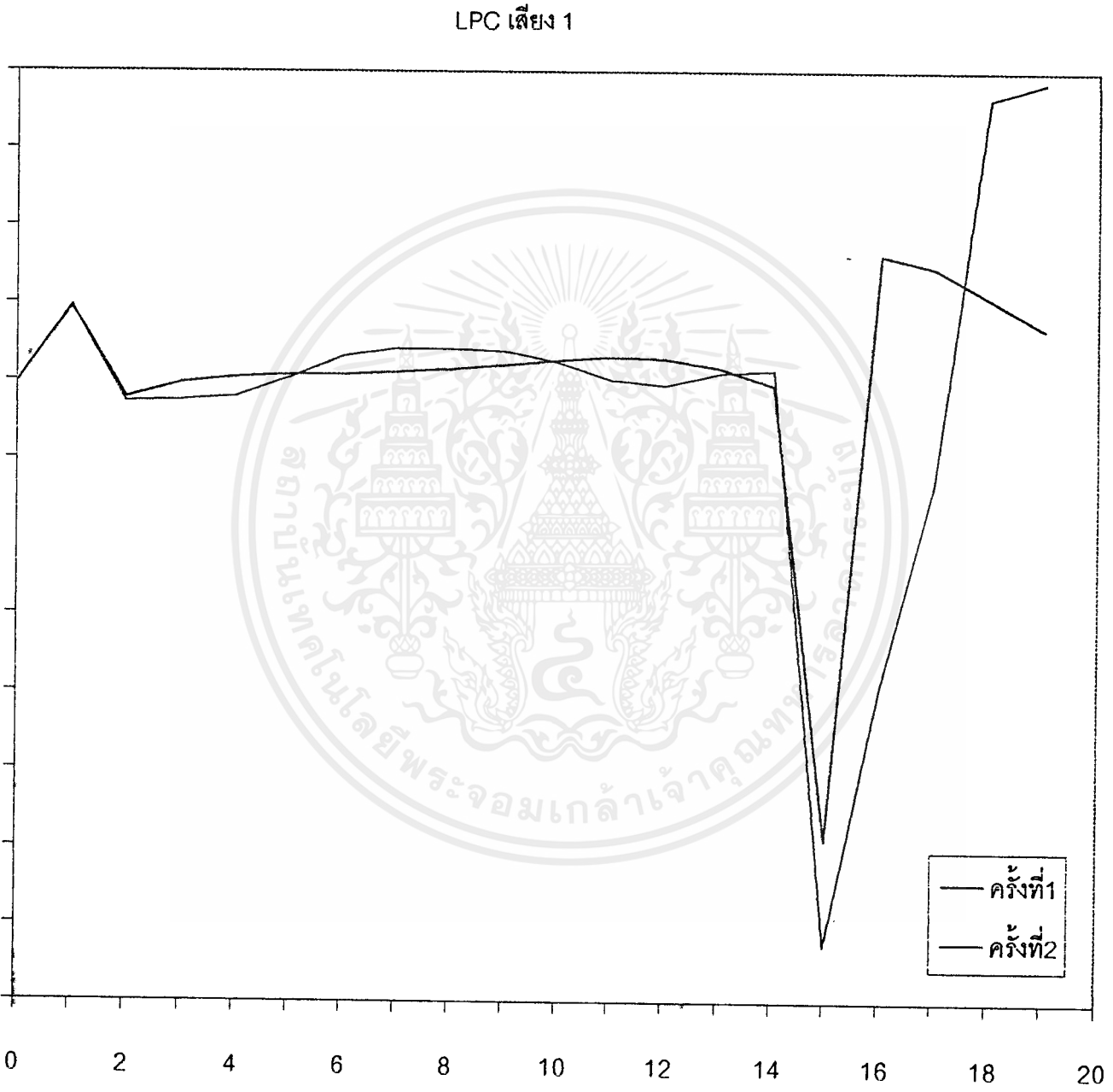
รูปที่ 4.9 แสดงค่า LPC เสียง 1 ของชายคนที่ 1 เทียบกับชายคนที่ 2



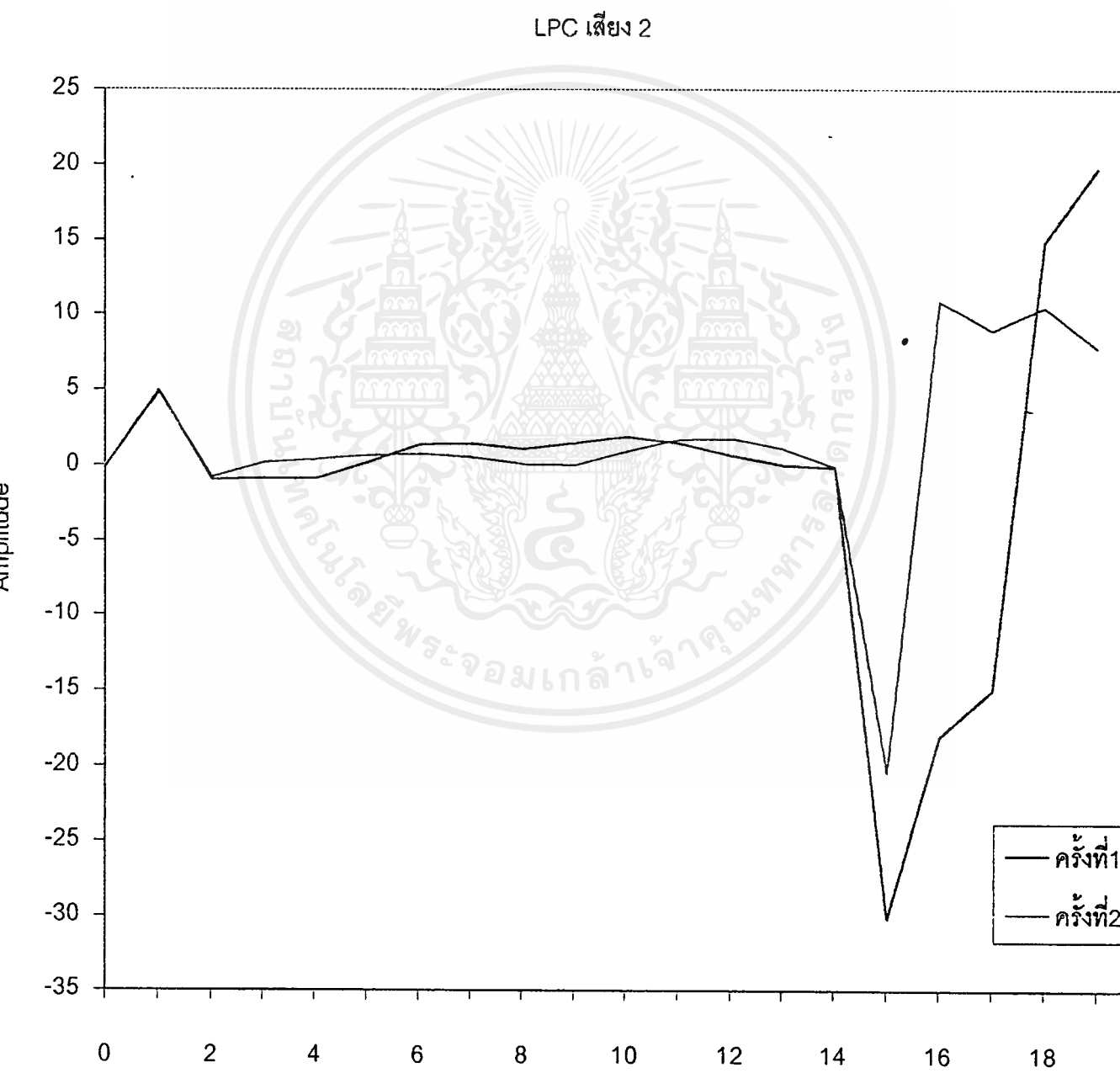
รูปที่ 4.10 แสดงค่า LPC เสียง 2 ของชายคนที่ 1 เทียบกับชายคนที่ 2



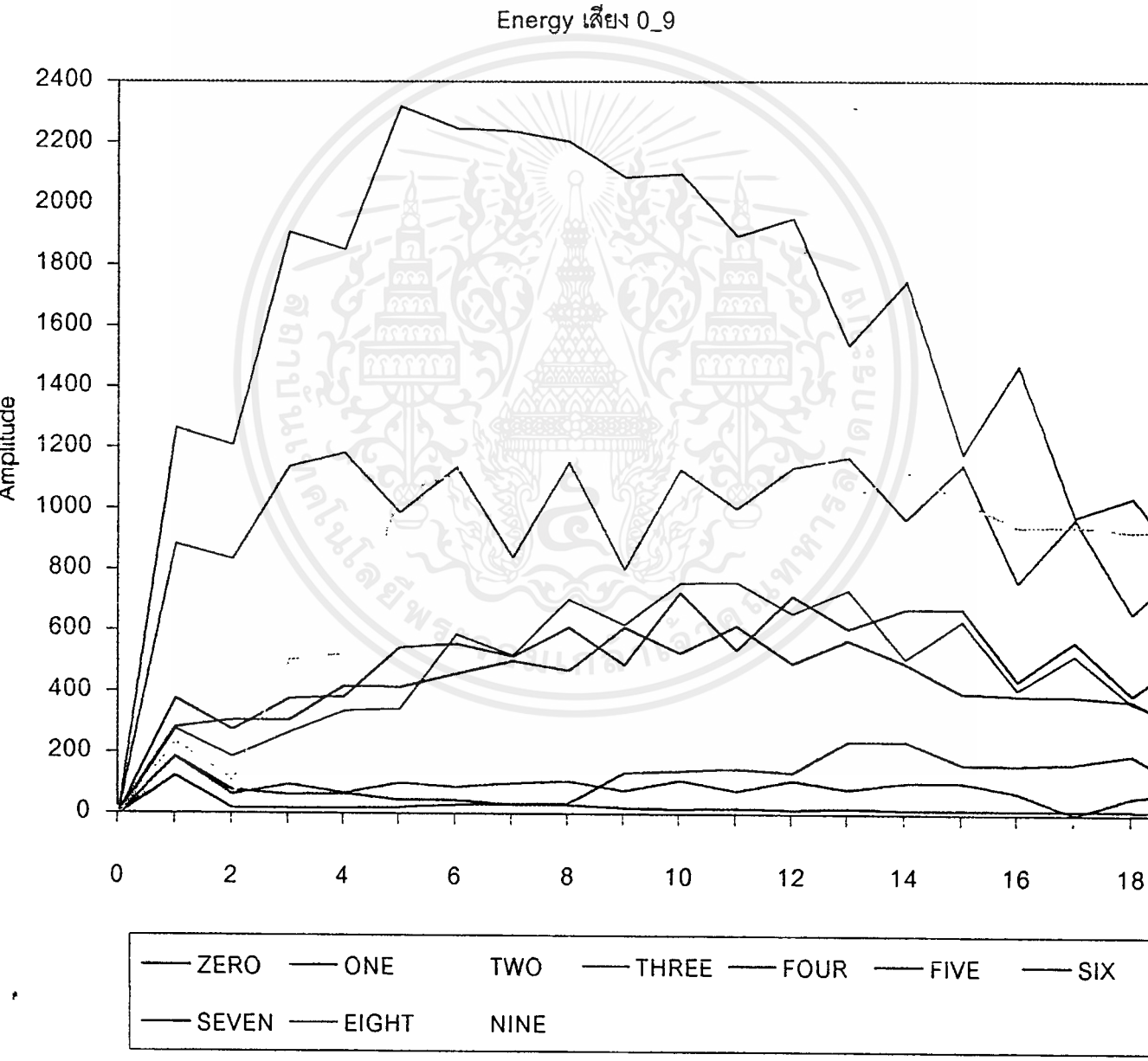
รูปที่ 4.11 แสดงค่า LPC เสียง 0 ของชายคนเดียวกันครั้งที่ 1 เทียบกับครั้งที่ 2



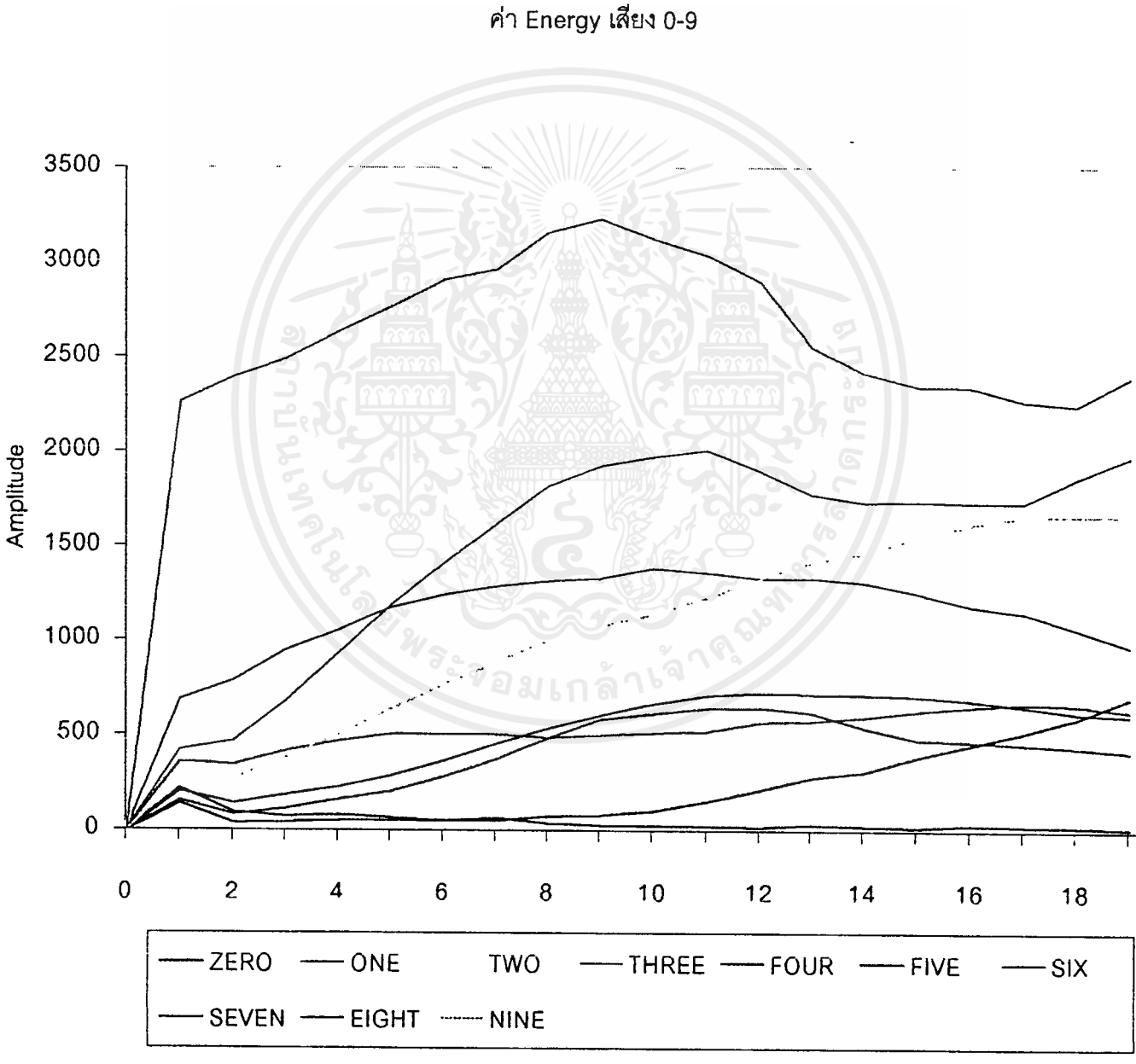
รูปที่ 4.12 แสดงค่า LPC เสียง 1 ของชายคนเดียวกันครั้งที่ 1 เทียบกับครั้งที่ 2



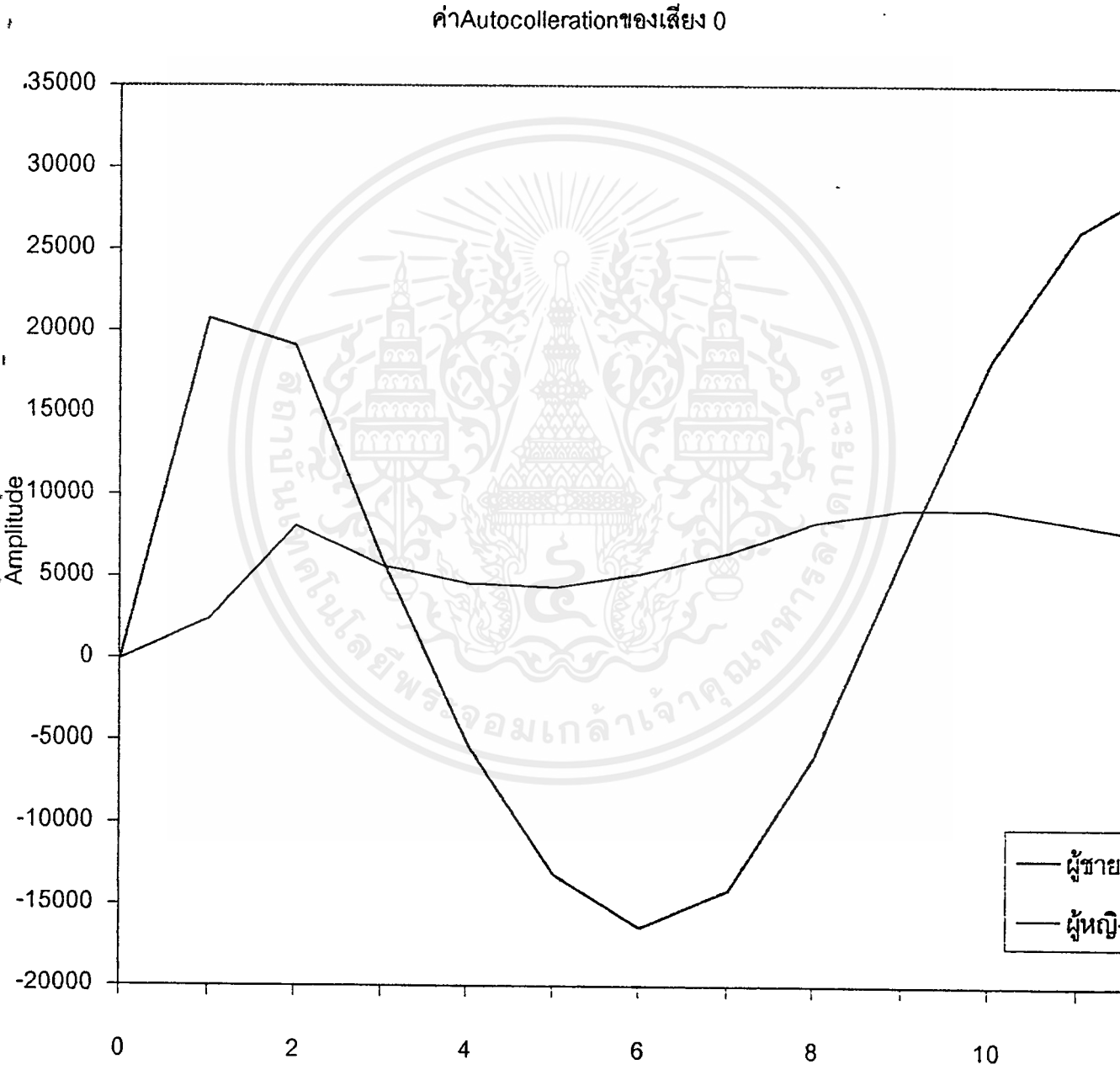
รูปที่ 4.13 แสดงค่า LPC เสียง 2 ของชายคนเดียวกันครั้งที่ 1 เทียบกับครั้งที่ 2



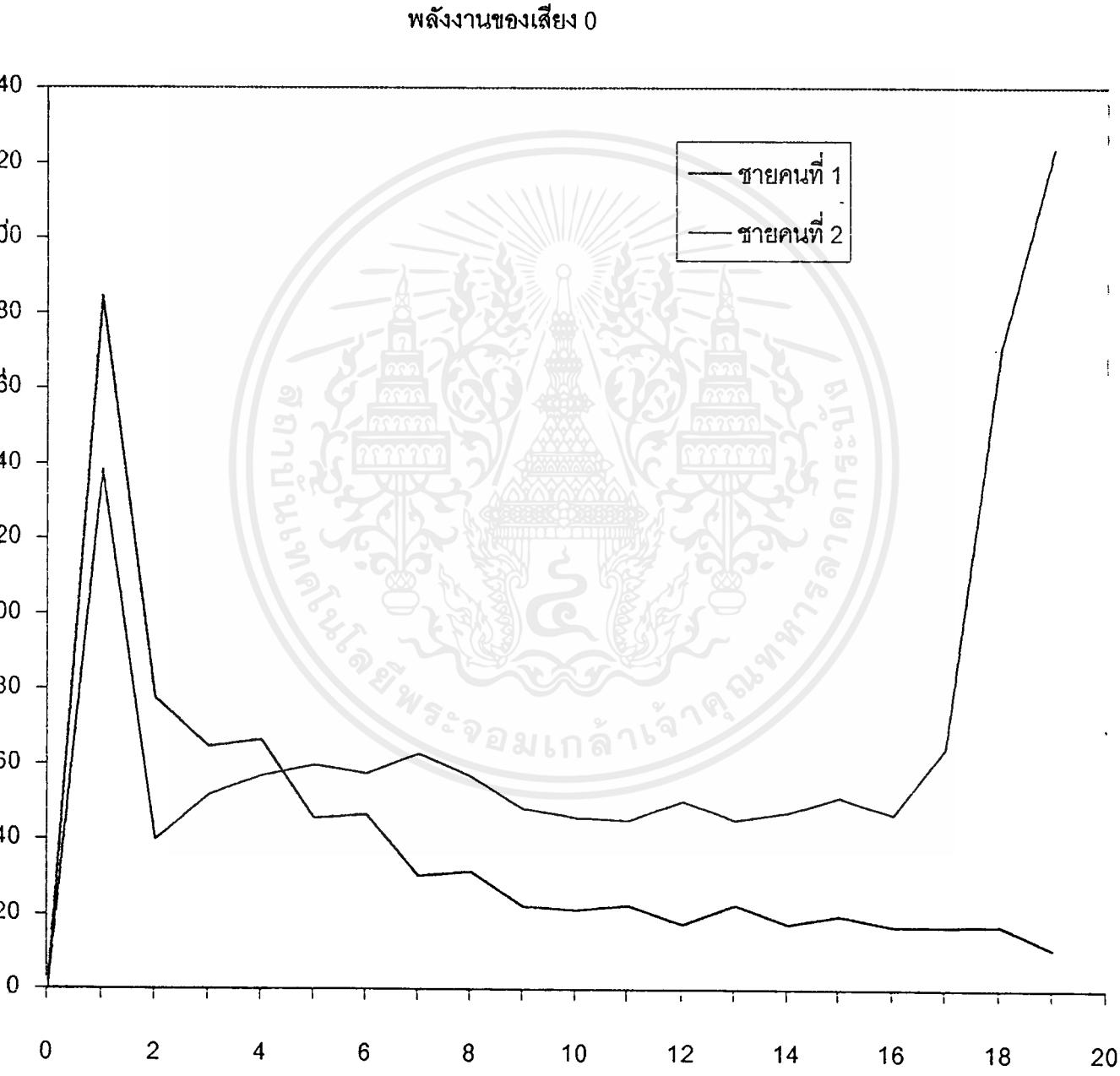
รูปที่ 4.14 แสดงค่า Energy เสียง 0-9 ของผู้ชาย



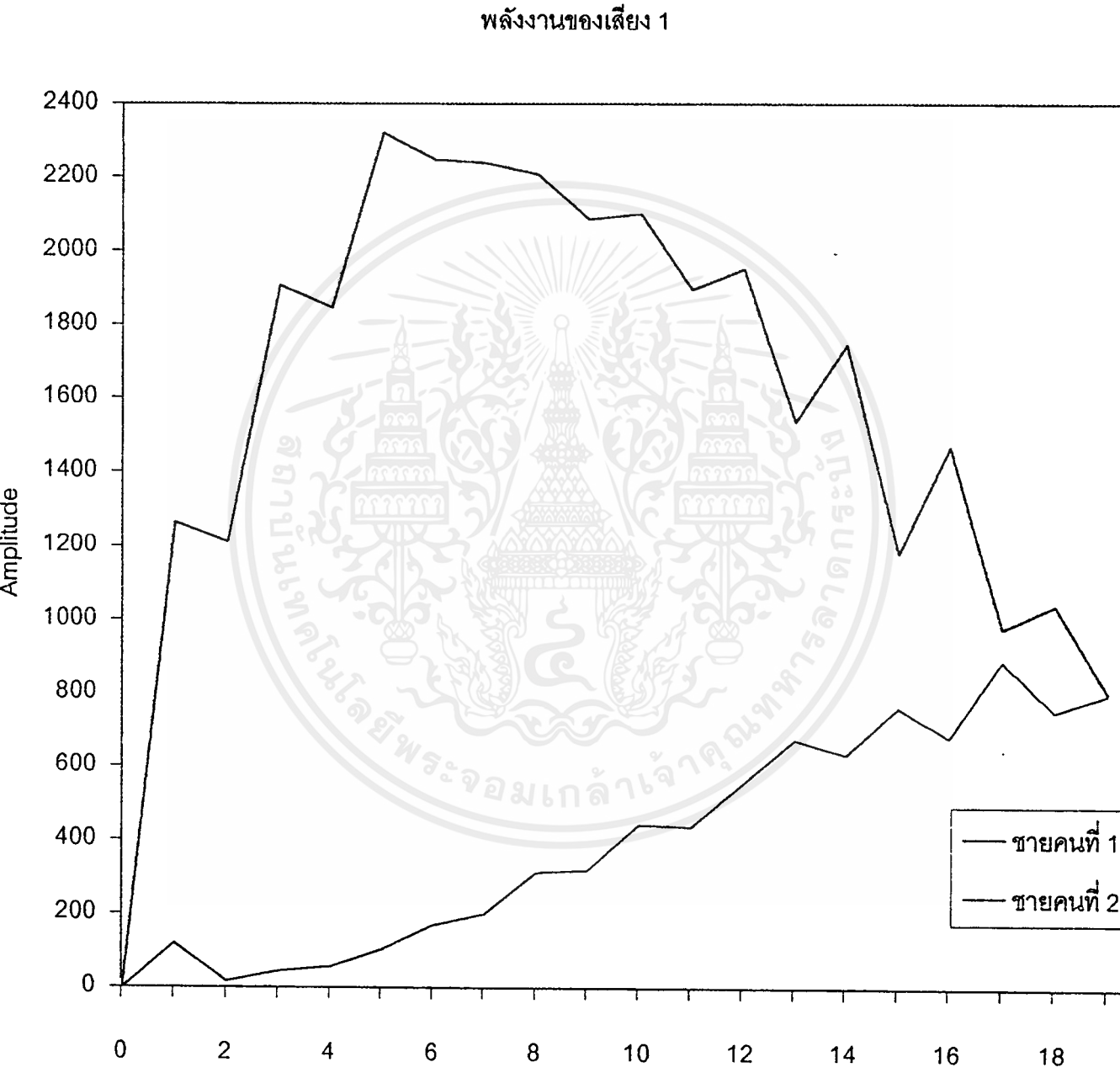
รูปที่ 4.15 แสดงค่า Energy เสียง 0-9 ของผู้หญิง



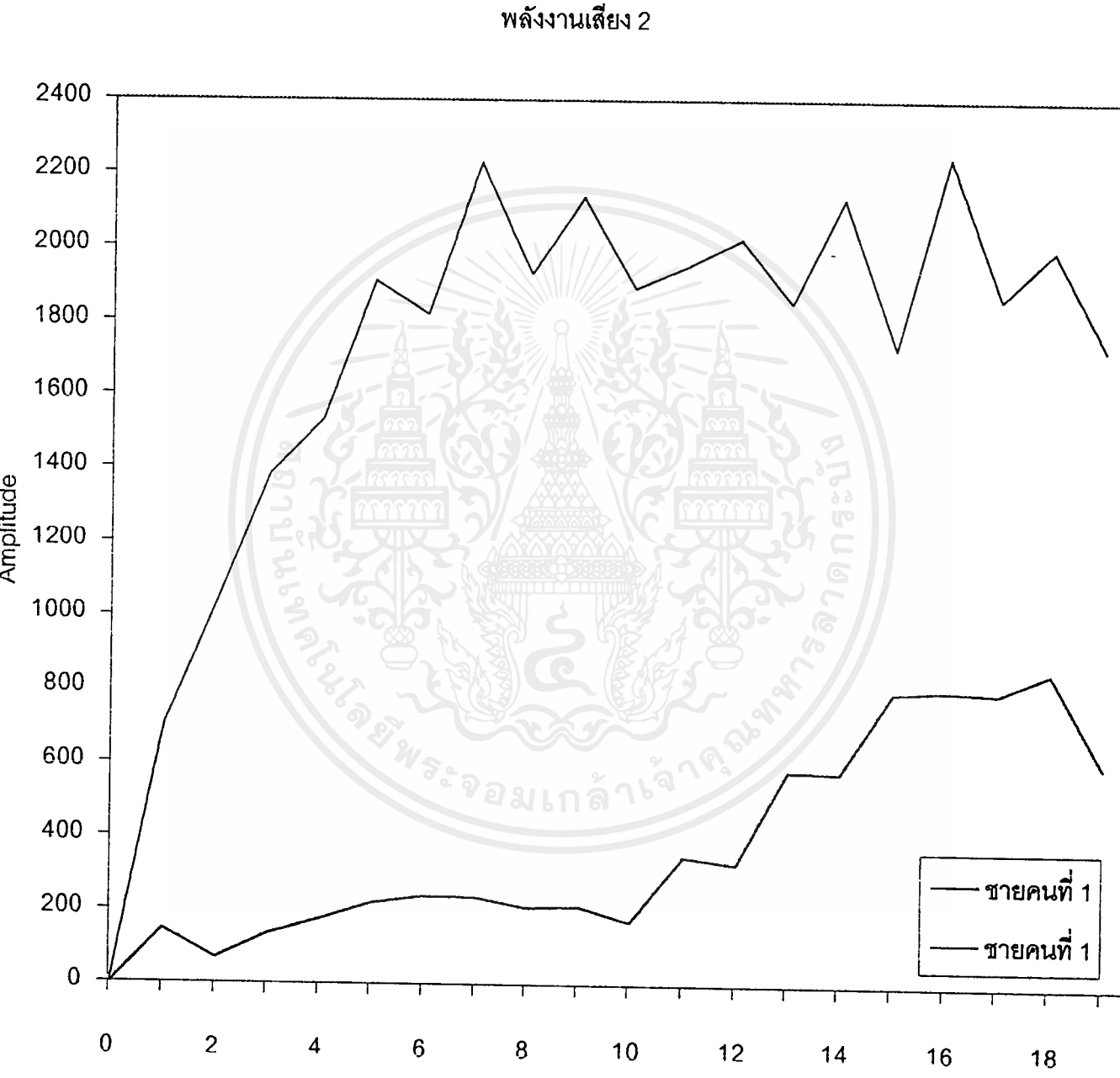
รูปที่ 4.16 แสดงค่า Energy เสียง 0 ของผู้ชาย เทียบกับ ผู้หญิง



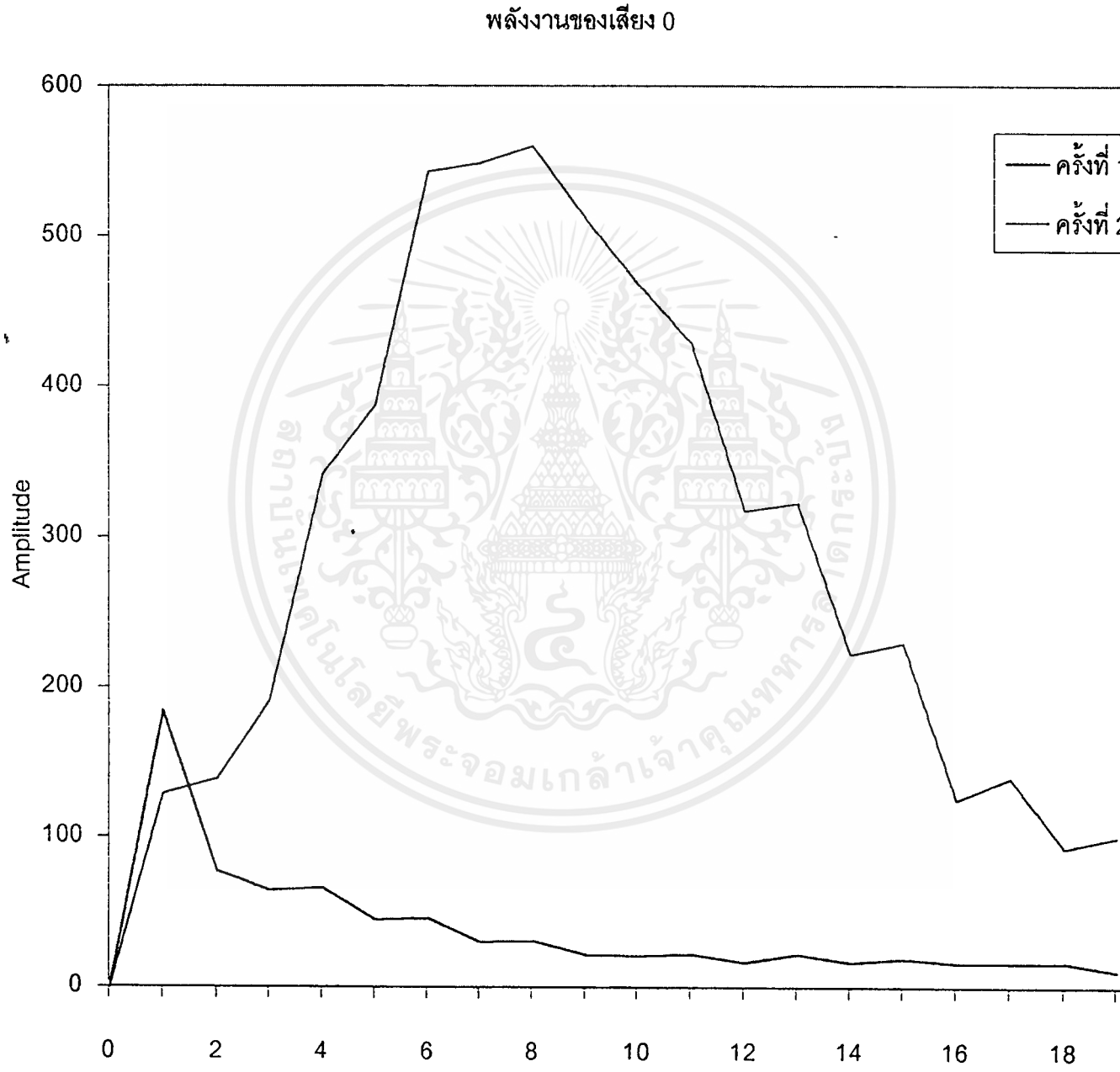
รูปที่ 4.17 แสดงค่า Energy เสียง 0 ของชายคนที่ 1 เทียบกับ ชายคนที่ 2



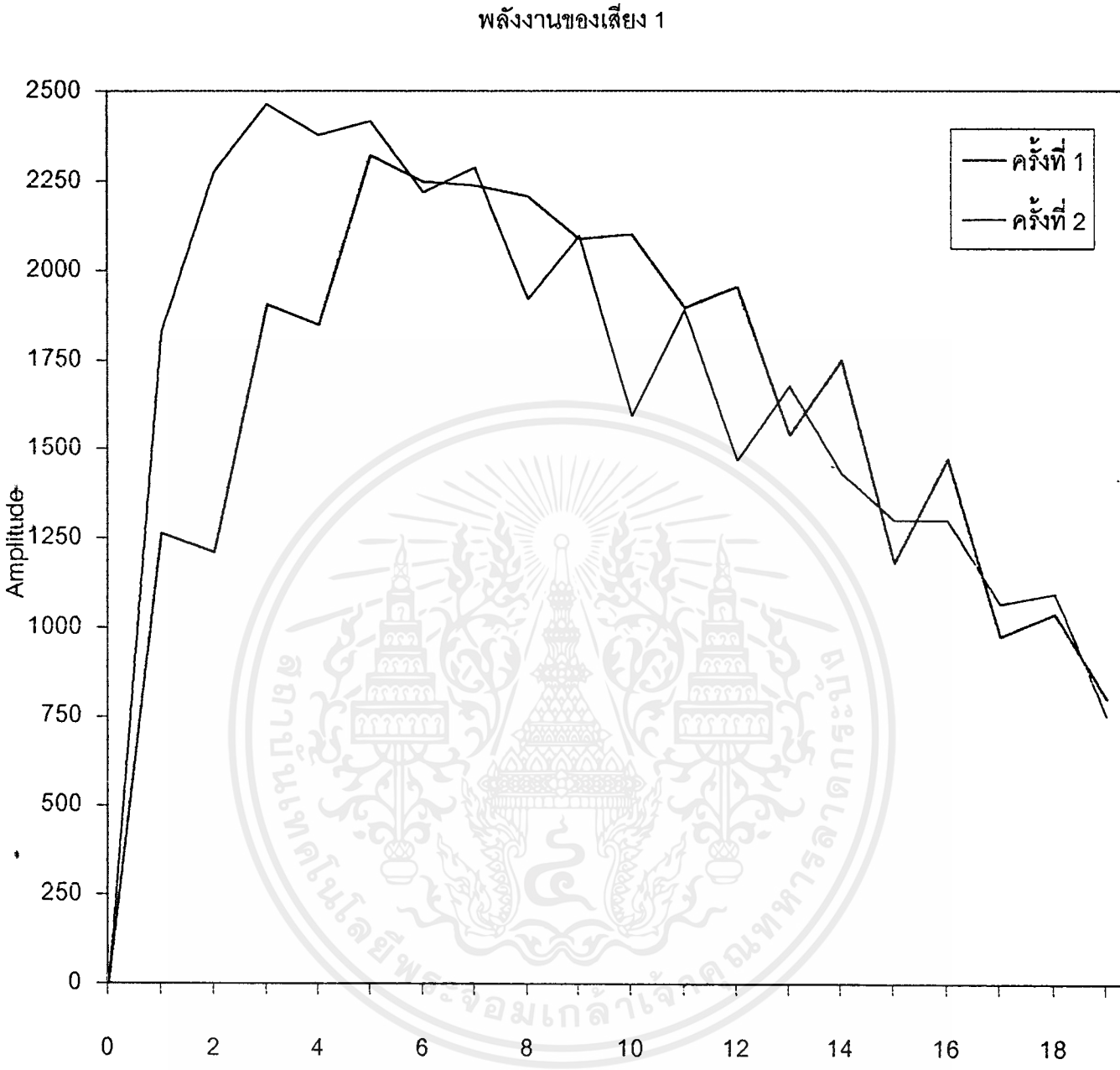
รูปที่ 4.18 แสดงค่า Energy เสียง 1 ของชายคนที่ 1 เทียบกับชายคนที่ 2



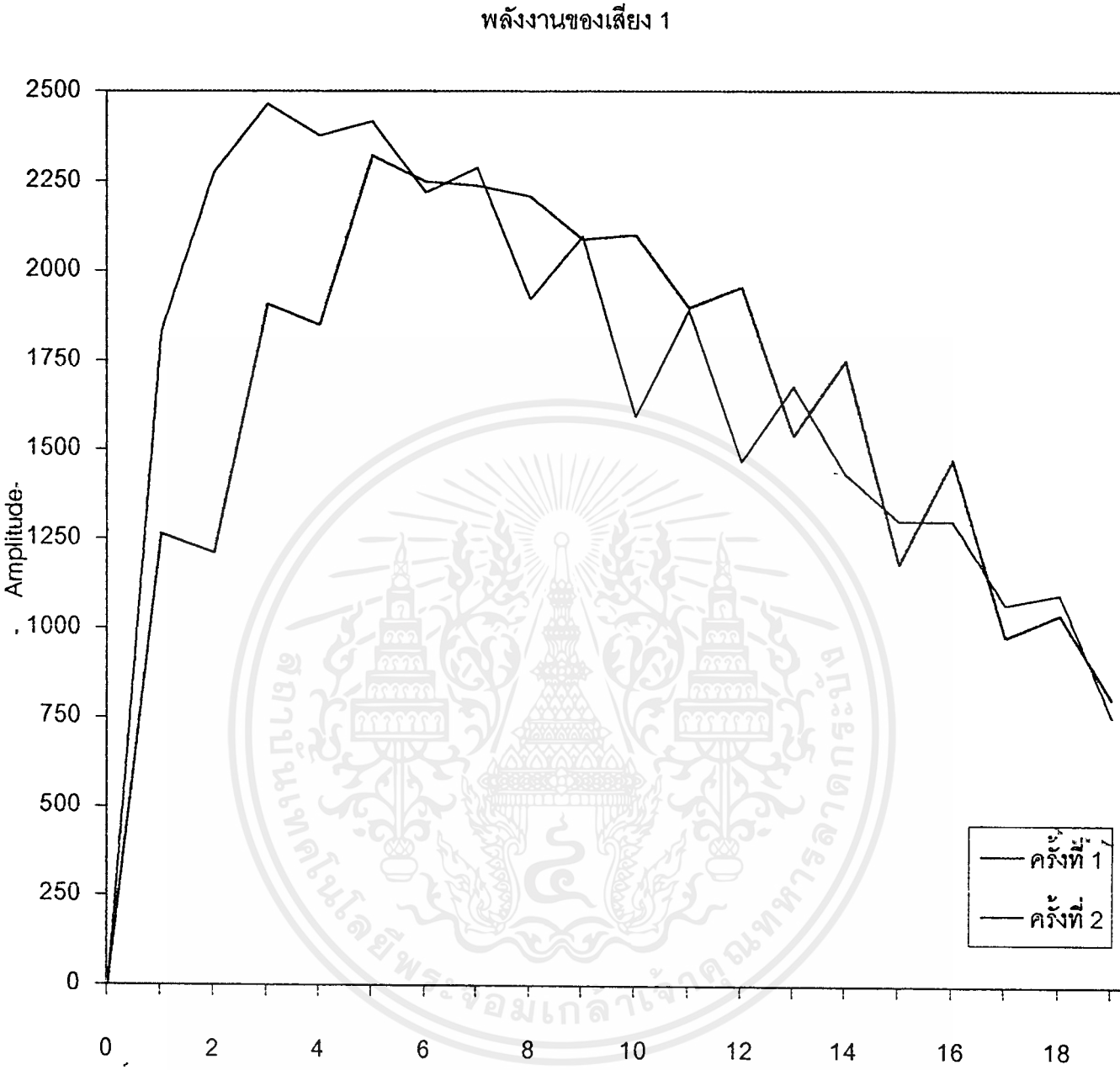
รูปที่ 4.19 แสดงค่า Energy เสียง 2 ของชายคนที่ 1 เทียบกับชายคนที่ 2



รูปที่ 4.20 แสดงค่า Energy เสียง 0 ของชายคนเดียวกันครั้งที่ 1 เทียบกับครั้งที่ 2

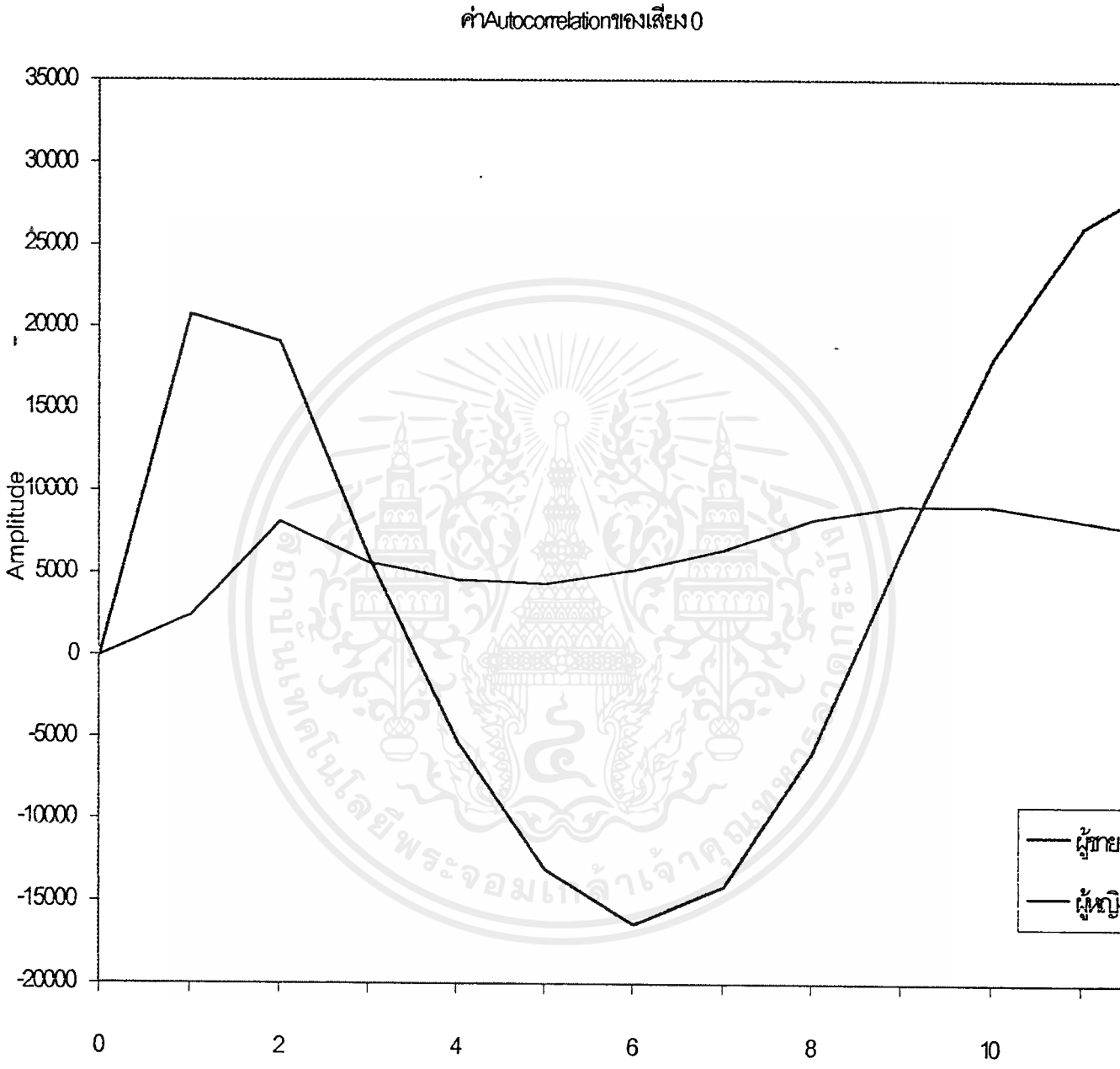


รูปที่ 4.21 แสดงค่า Energy เสียง 1 ของชายคนเดียวกันครั้งที่ 1 เทียบกับครั้งที่ 2

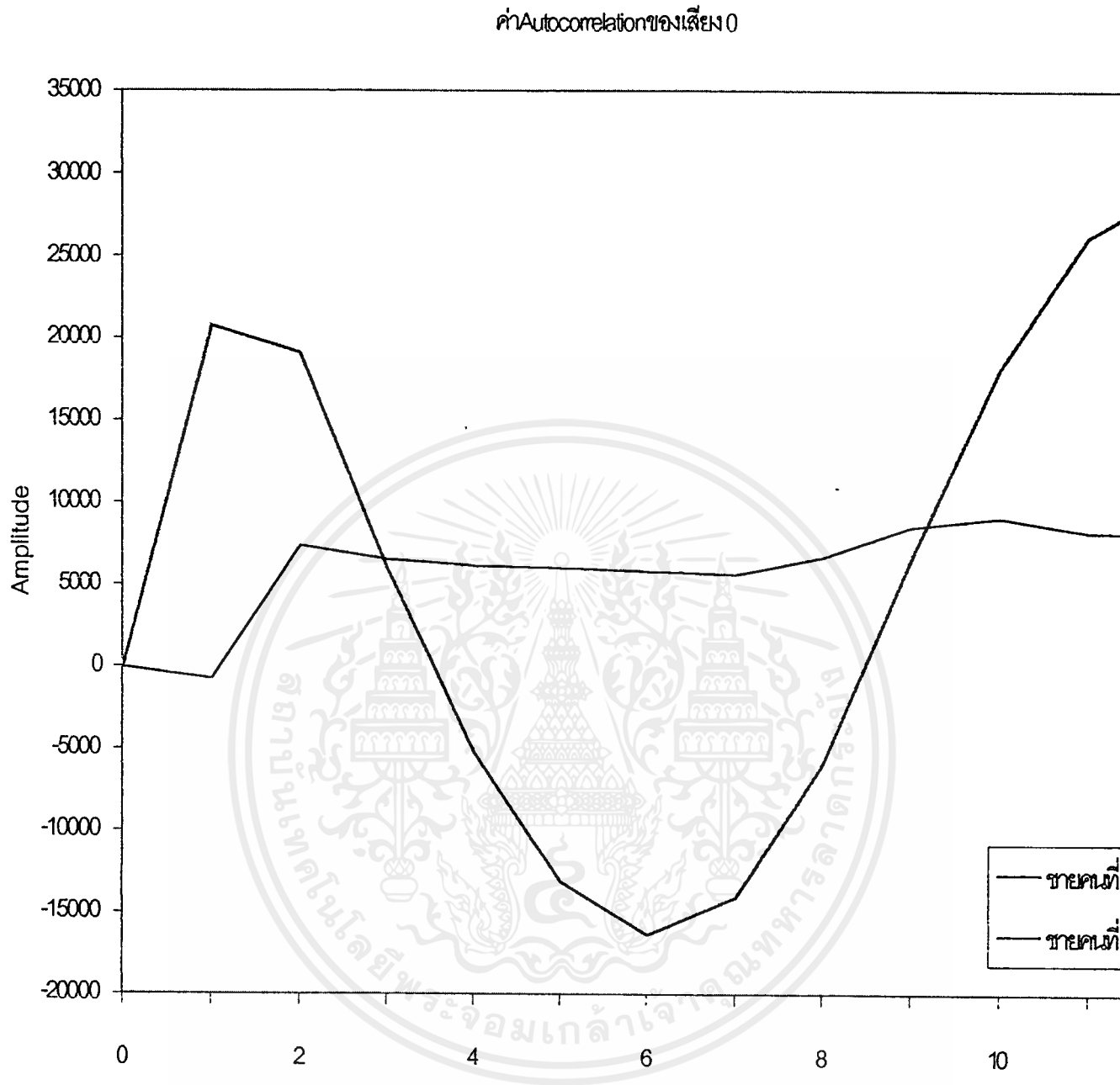


รูปที่ 4.22 แสดงค่า Energy เสียง 2 ของชายคนเดียวกันครั้งที่ 1 เทียบกับครั้งที่ 2

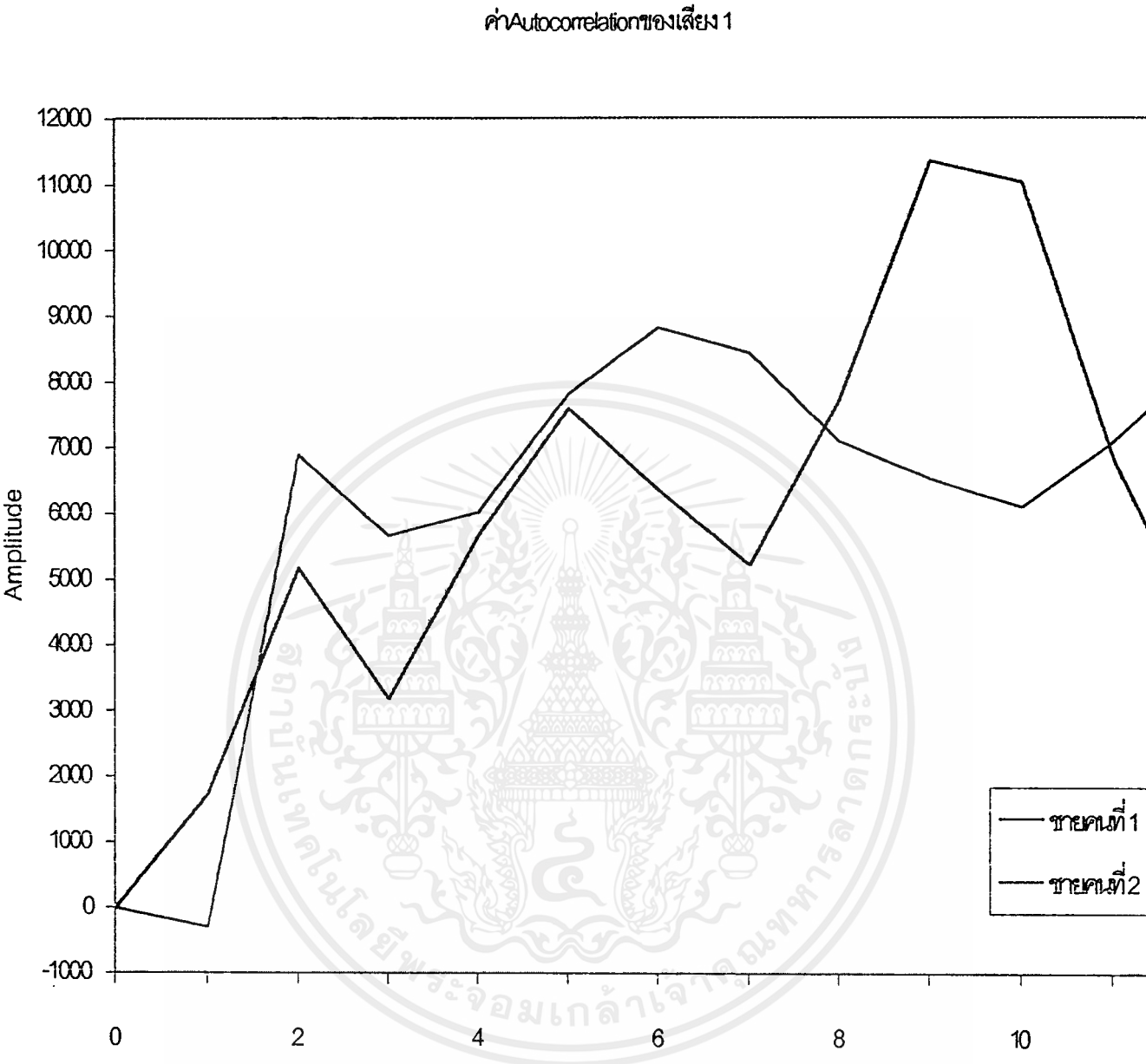
ค่า Autocorrelation



รูปที่ 4.23 แสดงค่า Autocorrelation เสียง 0 ของผู้ชายเทียบกับผู้หญิง

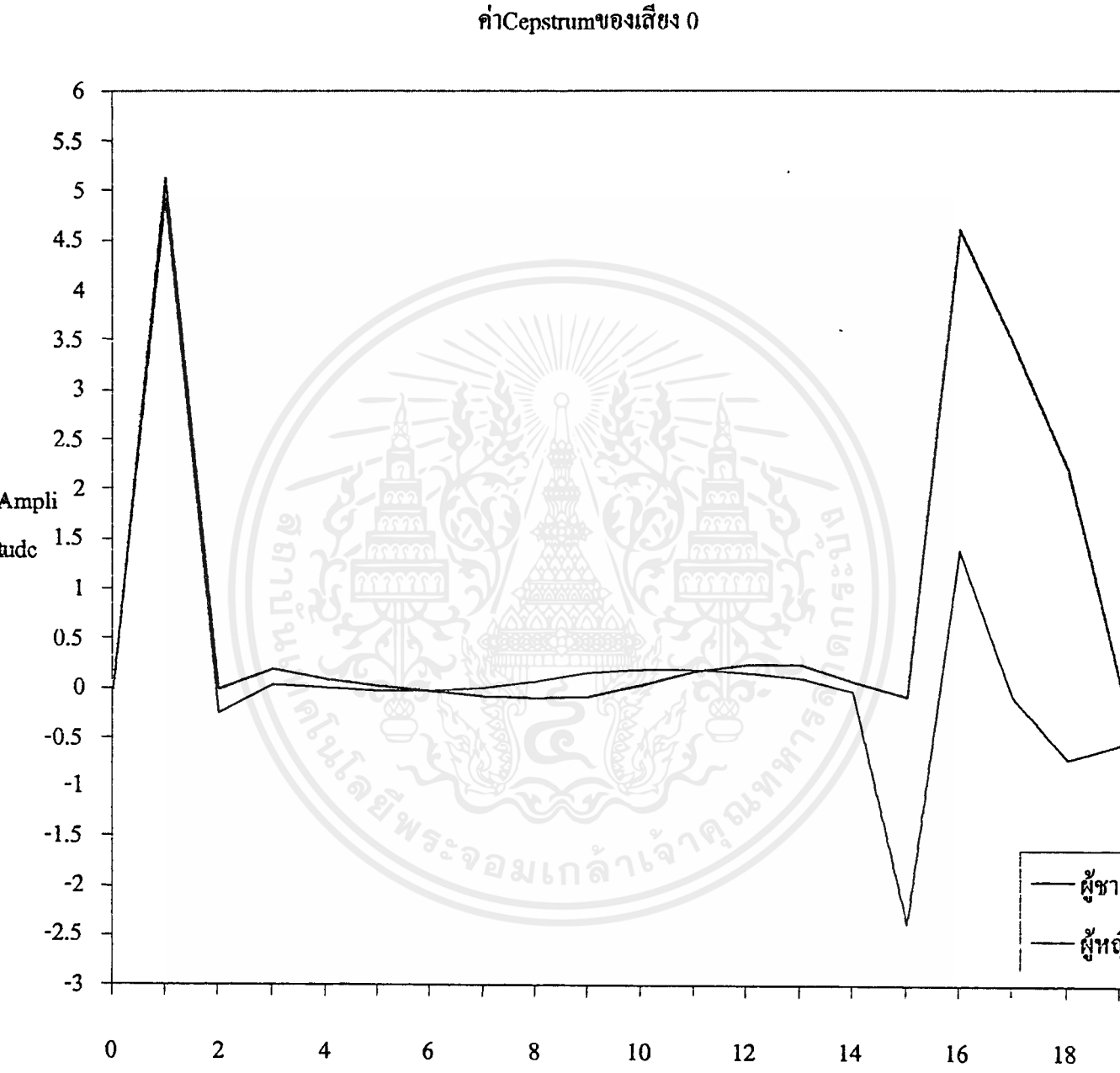


รูปที่ 4.24 แสดงค่า Autocorrelation เสียง 0 ของชายคนที่ 1 เทียบกับชายคนที่ 2

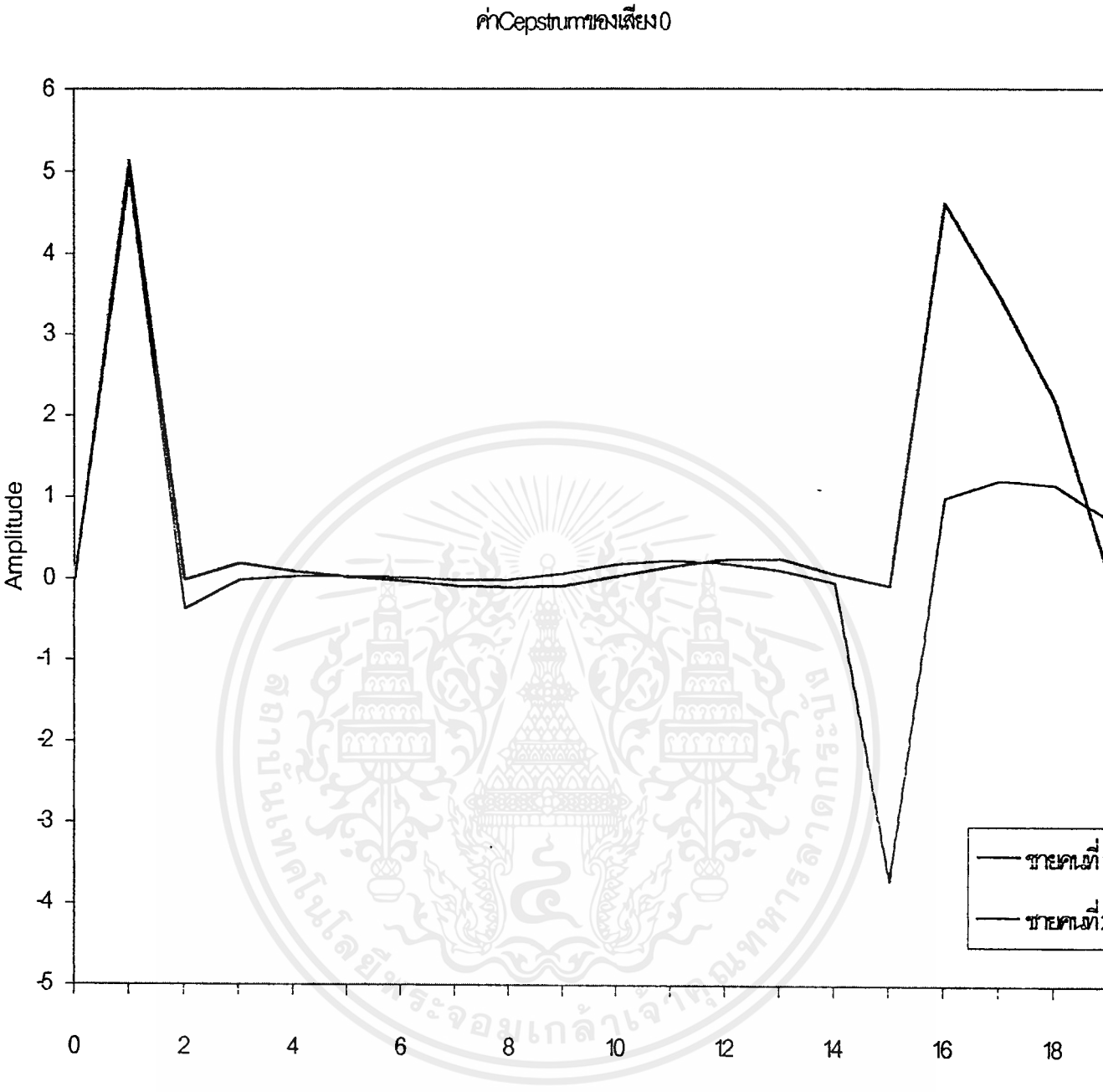


รูปที่ 4.25 แสดงค่า Autocorrelation เสียง 1 ของชายคนที่ 1 เทียบกับชายคนที่ 2

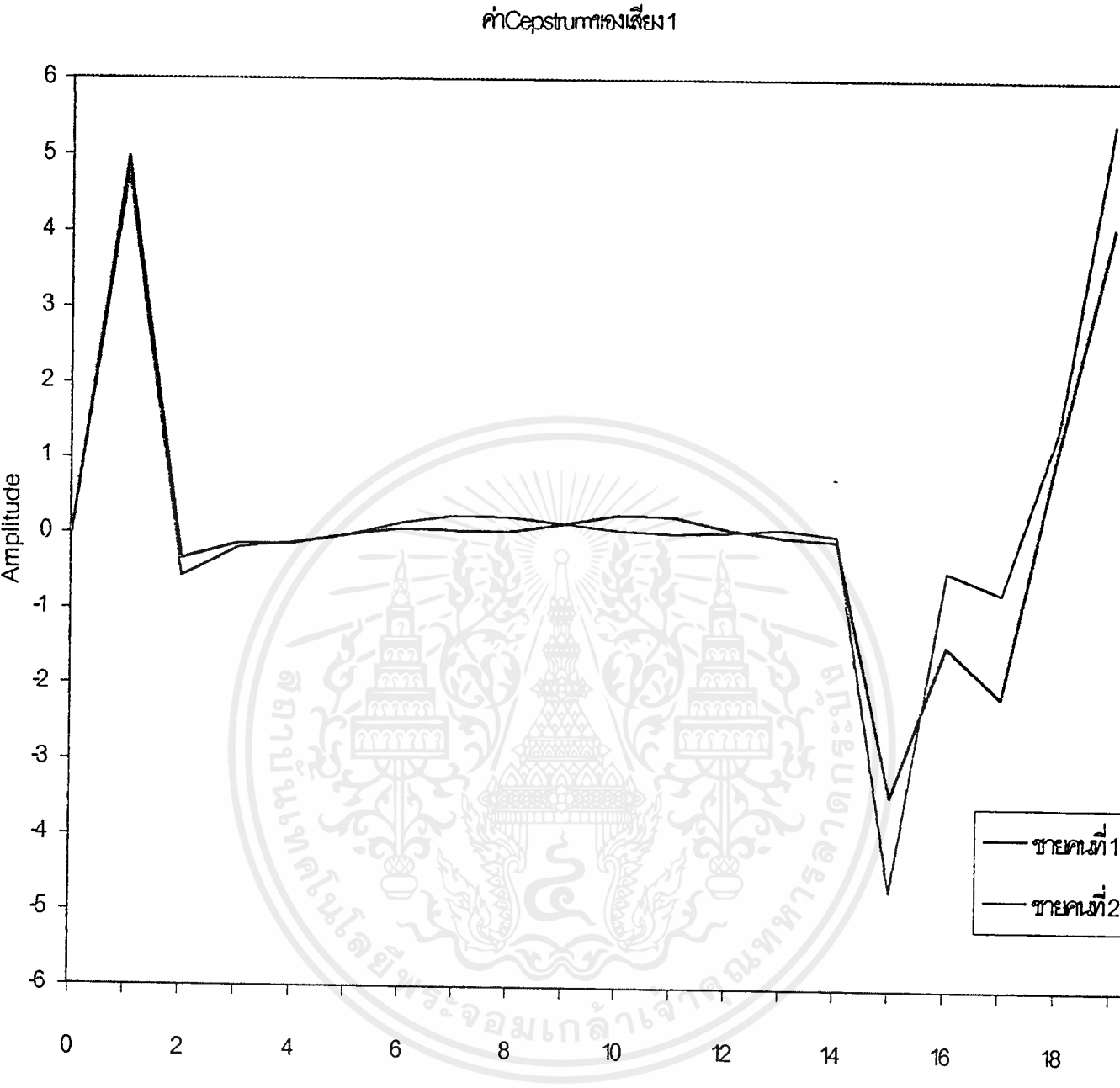
ค่าพารามิเตอร์ Cepstrum



รูปที่ 4.26 แสดงค่า Cepstrum เสียง 0 ของผู้ชายเทียบกับผู้หญิง

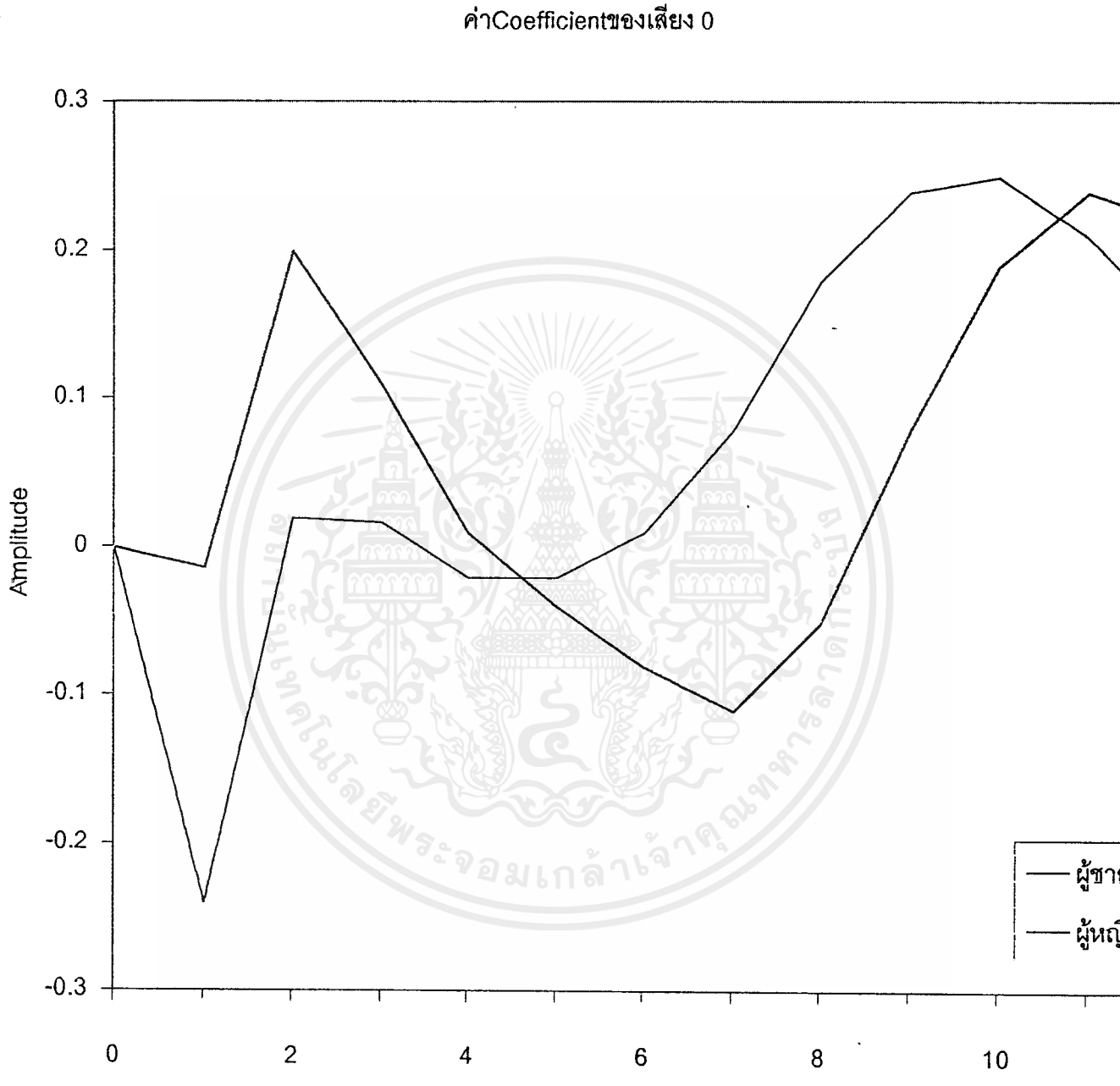


รูปที่ 4.27 แสดงค่า Cepstrum เสียง 0 ของชายคนที่ 1 เทียบกับชายคนที่ 2

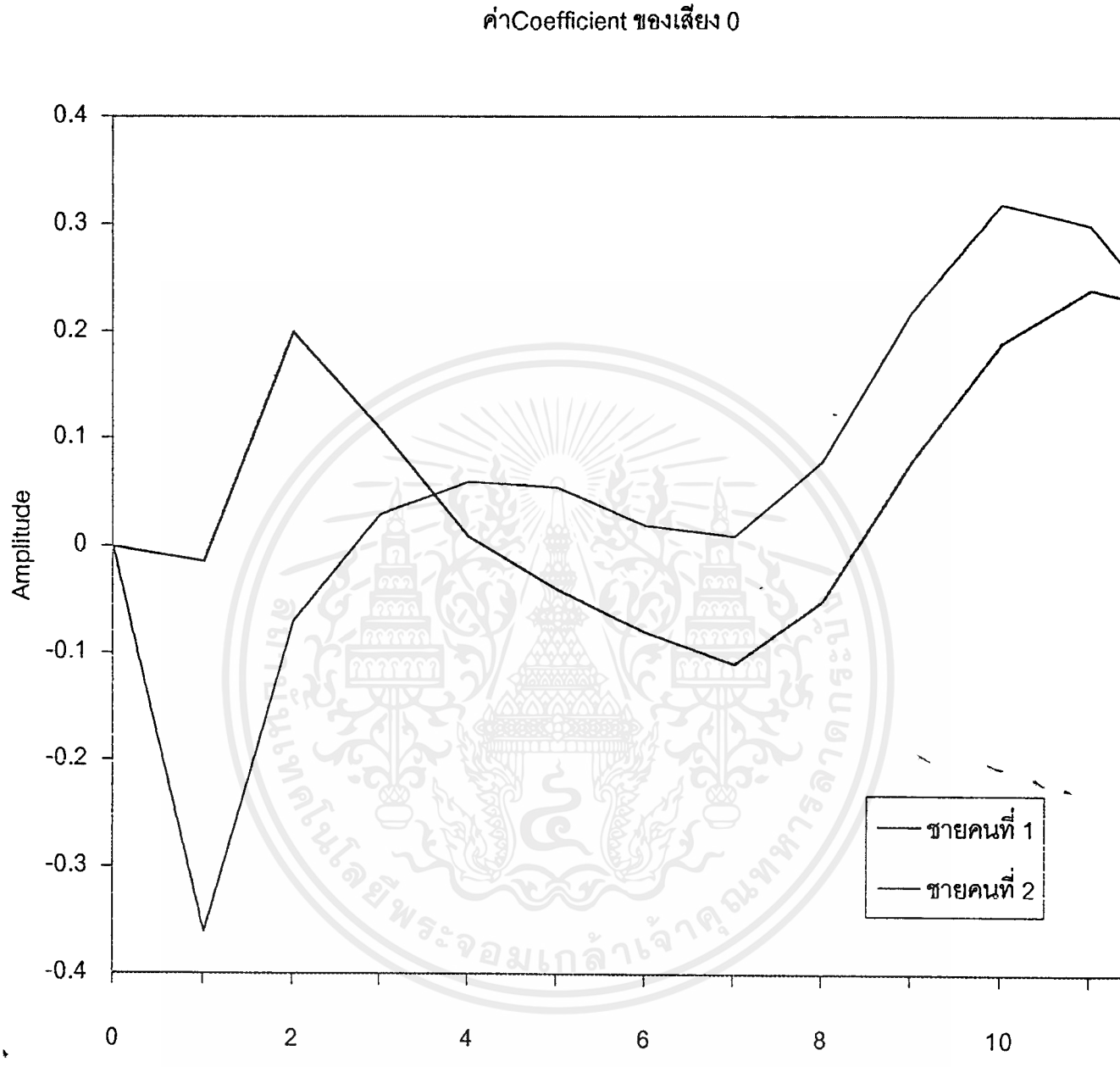


รูปที่ 4.28 แสดงค่า Cepstrum เสียง 1 ของชายคนที่ 1 เทียบกับชายคนที่ 2

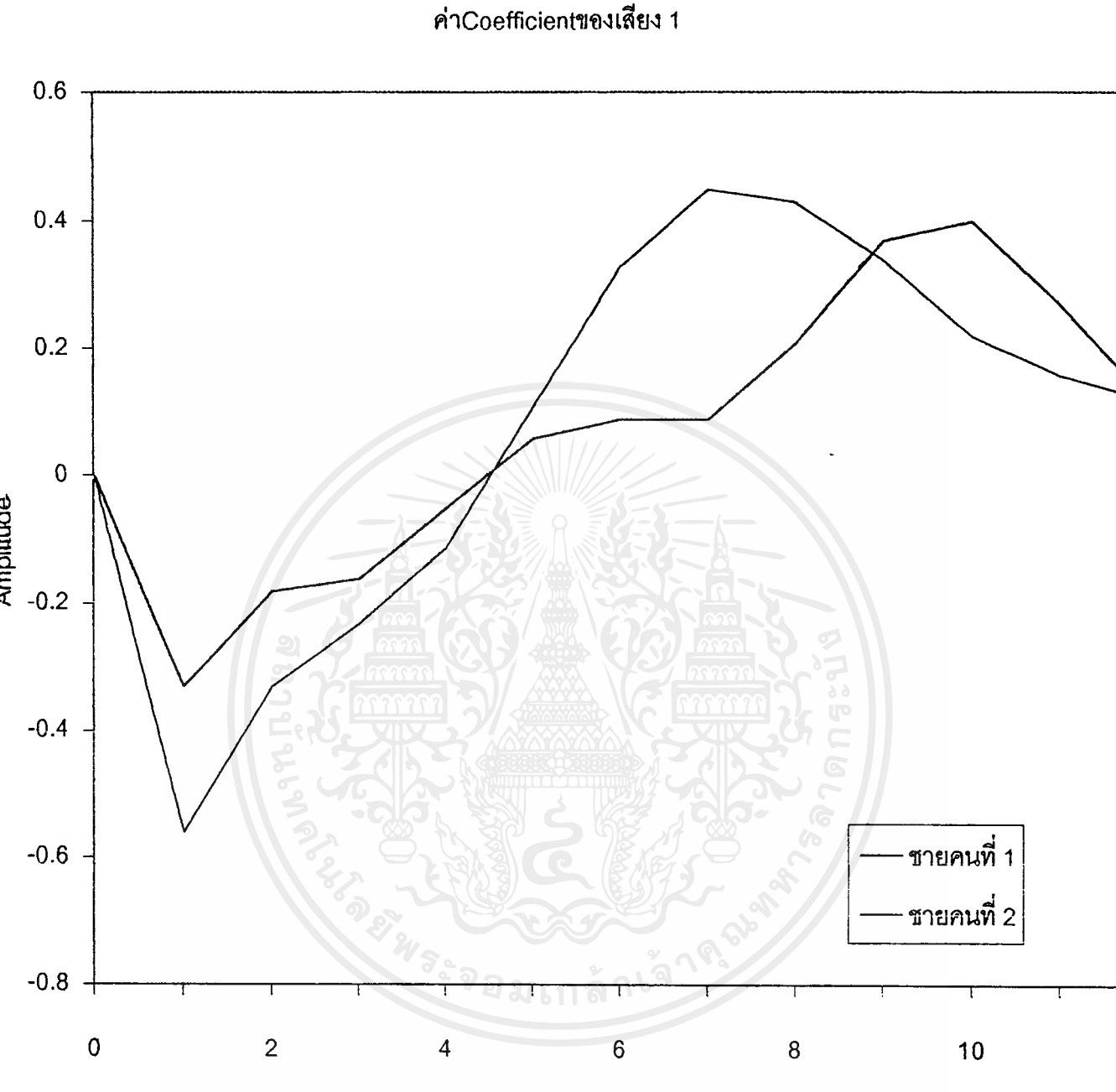
ค่าพารามิเตอร์ Coefficient



รูปที่ 4.29 แสดงค่า Coefficient เสียง 0 ของผู้ชายเทียบกับผู้หญิง

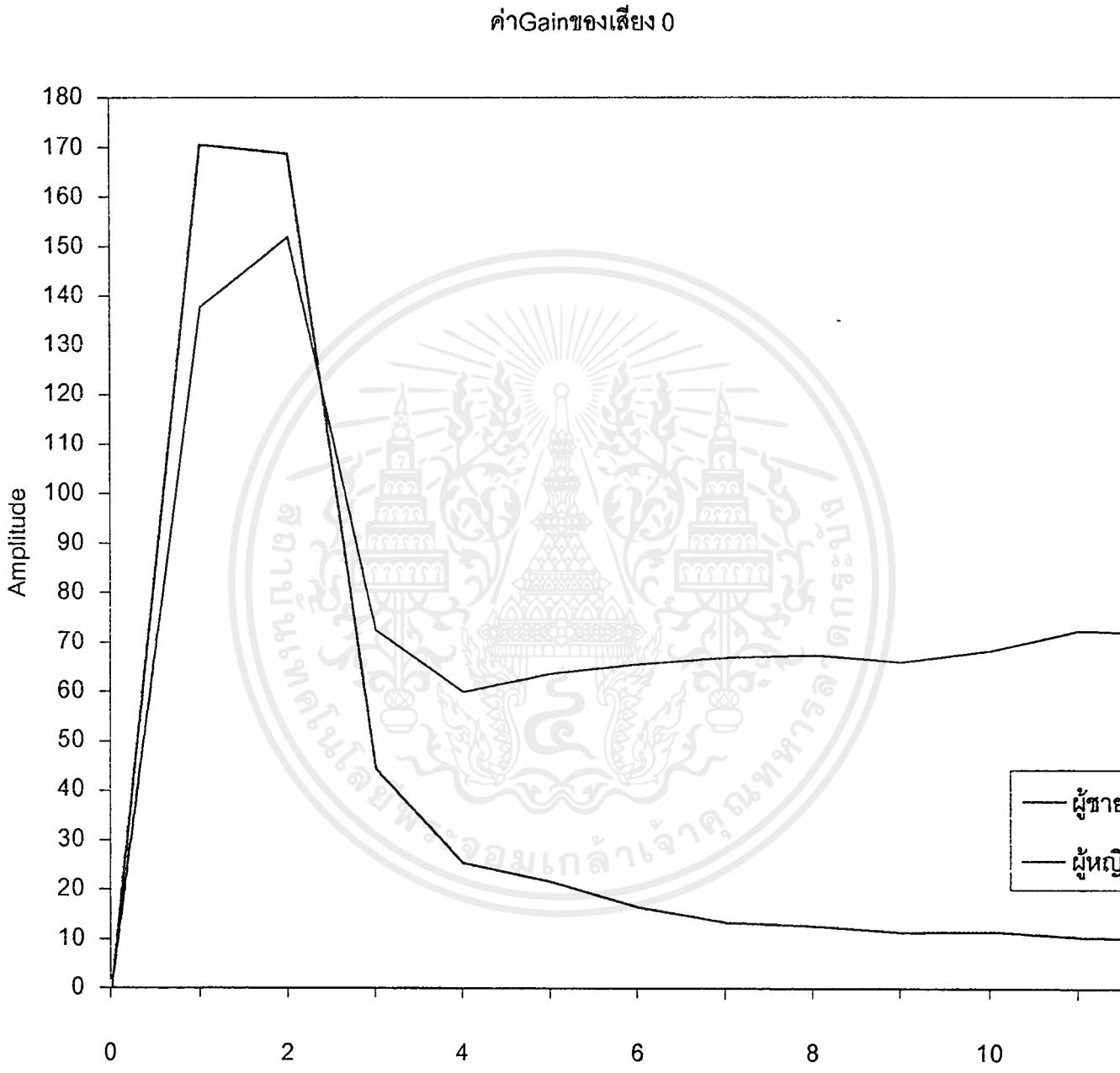


รูปที่ 4.30 แสดงค่า Coefficient เสียง 0 ของชายคนที่ 1 เทียบกับชายคนที่ 2

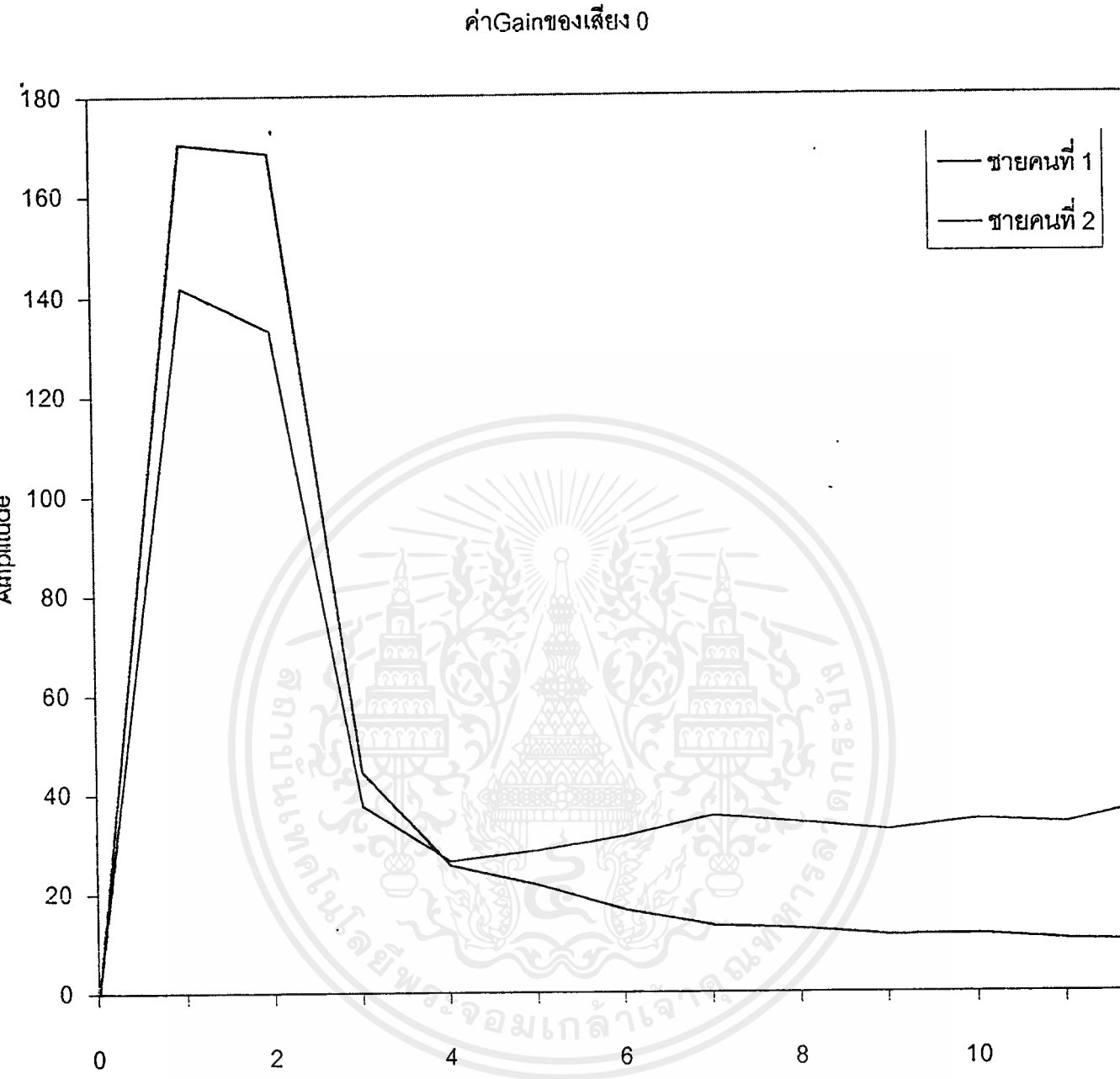


รูปที่ 4.31 แสดงค่า Coefficient เสียง 1 ของชายคนที่ 1 เทียบกับชายคนที่ 2

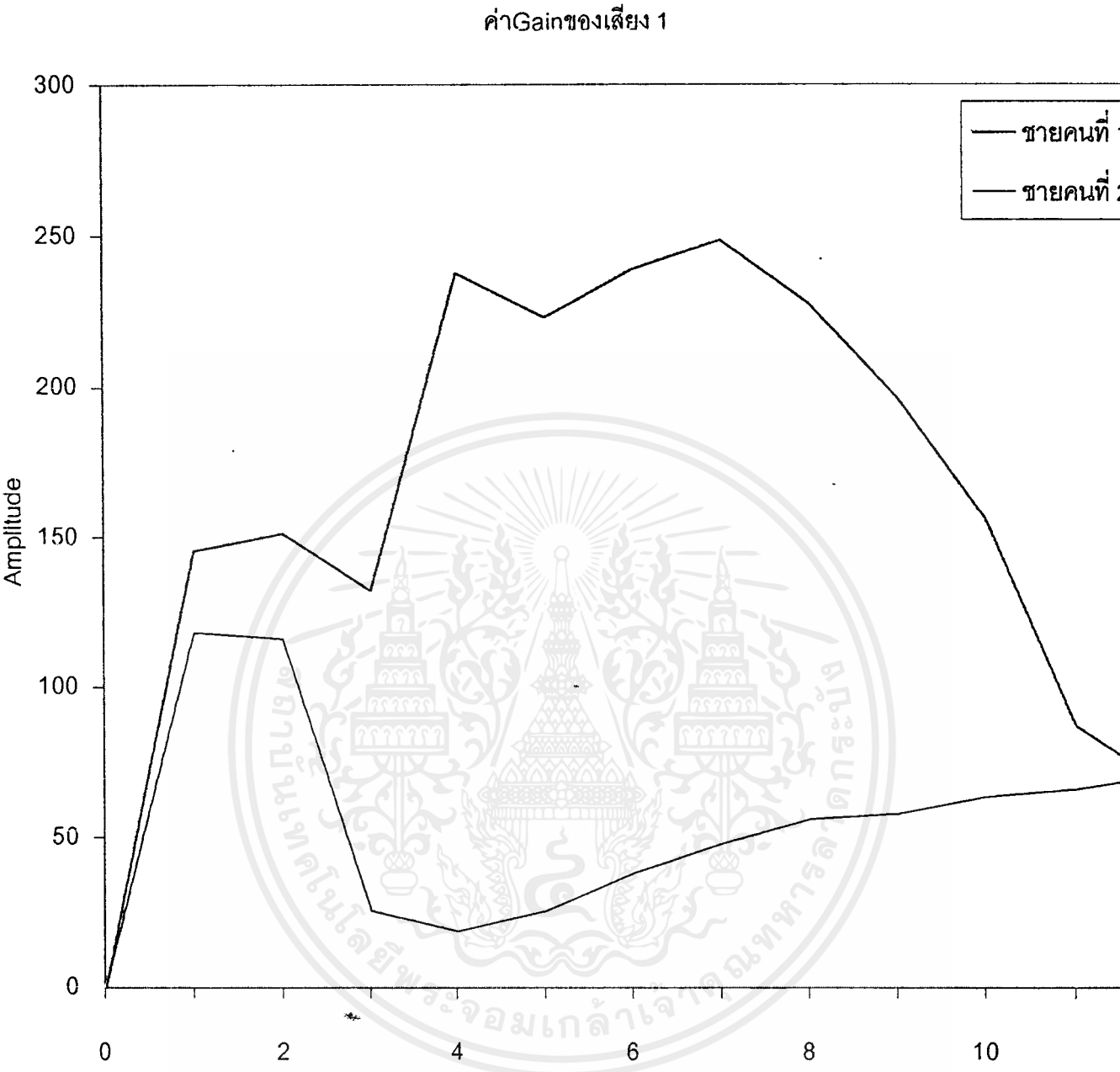
ค่าพารามิเตอร์Gain



รูปที่ 4.32 แสดงค่า Gain เสียง 0 ของผู้ชายเทียบกับผู้หญิง



รูปที่ 4.33 แสดงค่า Gain เสียง 0 ของชายคนที่ 1 เทียบกับชายคนที่ 2



รูปที่ 4.34 แสดงค่า Gain เสียง 1 ของชายคนที่ 1 เทียบกับชายคนที่ 2

ผู้ชาย					
Autocolleration	Coefficient	Gain	Cepstrum	Energy	LPC
20762.71	-0.014	170.62	5.139	184.493	4.885
19148.61	0.202	170.62	-0.014	77.693	-0.877
6038.47	0.111	170.62	0.202	64.780	0.188
-5168.54	0.016	170.62	0.108	66.737	0.326
-13032.31	-0.042	170.62	0.035	45.723	0.422
-16323.51	-0.089	170.62	-0.020	46.697	0.550
-14006.19	-0.111	170.62	-0.077	30.687	0.509
-5855.24	-0.053	170.62	-0.112	31.777	0.428
6570.29	0.085	170.62	-0.071	22.510	0.936
18240.20	0.199	170.62	0.054	21.667	1.550
26210.61	0.249	170.62	0.170	22.937	1.800
29323.08	0.223	170.62	0.251	17.640	1.565
			0.267	23.150	1.016
			0.079	17.697	-0.014
			-0.079	20.280	-25.065
			4.629	17.140	12.214
			3.492	17.250	5.513
			2.210	17.520	6.076
			0.054	11.217	5.282

ผู้หญิง					
Autocolleration	Coefficient	Gain	Cepstrum	Energy	LPC
2423.924	-0.241	138.07	4.928	221.8	-5.41
8173.333	0.021	138.07	-0.241	96.5	-0.75
5695.761	0.018	138.07	0.050	75.8	-0.88
4681.887	-0.022	138.07	0.008	81.5	-0.46
4442.553	-0.028	138.07	-0.024	69.1	-0.34
5289.335	0.015	138.07	-0.022	54.3	4.10
6544.194	0.087	138.07	0.020	67.5	5.12
8366.611	0.183	138.07	0.081	40.3	-2.51
9243.459	0.242	138.07	0.163	30.41	-0.60
9233.546	0.250	138.07	0.205	29.8	-0.35
8315.148	0.214	138.07	0.205	28.8	-2.92
7467.865	0.150	138.07	0.170	24.6	2.98
			0.111	34.8	2.80
			-0.027	27.5	0.39
			-2.374	24.1	1.05
			1.391	35.8	1.32
			-0.084	34.7	-1.36
			-0.702	28.1	0.03
			-0.558	19.3	1.87

ตารางที่ 4.2 แสดงค่าองค์ประกอบของเสียงผู้ชายเทียบกับผู้หญิง

กรณีที่เสียงต้นแบบเป็นผู้ชาย 1 คนๆละ 1 ครั้ง

เสียง	เสียงต้นแบบ	เสียงเดิม			เสียงของ ชายคนอื่น		เสียงของ หญิงคนอื่น	
0	0	6	6	6	6	6	2	6
1	1	7	1	1	ปิด	8	8	7
2	2	2	เปิด	2	6	6	8	2
3	3	8	3	9	9	3	9	9
4	4	4	4	4	4	4	4	0
5	5	5	8	8	9	8	9	9
6	6	6	0	6	6	6	2	6
7	7	5	1	1	4	7	0	4
8	8	8	8	9	9	8	9	9
9	9	9	9	9	9	9	9	9
เปิด	เปิด	8	8	5	5	เปิด	เปิด	2
ปิด	ปิด	7	7	7	6	4	4	0
		50%	41.67%	41.67%	16.67%.	58.33%	25%	25%

ตารางที่ 4.3 แสดงการจดจำเสียงในกรณีต่างๆจากเสียงผู้ชายเป็นต้นแบบ

กรณีที่เสียงต้นแบบเป็นผู้หญิง 1 คนๆละ 1 ครั้ง

เสียง	เสียงต้นแบบ	เสียงของผู้หญิงคนเดิม			เสียง ผู้ชาย		เสียง ผู้หญิง	
0	0	0	0	0	4	0	0	0
1	1	1	1	1	1	4	1	1
2	2	2	2	2	9	0	0	2
3	3	3	3	5	5	3	3.	5
4	4	4	4	4	4	4	4	ปิด
5	5	5	5	5	8	8	8	8
6	6	6	6	6	0	9	0	6
7	7	7	1	7	1	4	ปิด	7
8	8	8	8	5	8	8	8	8
9	9	9	9	9	9	9	9	9
เปิด	เปิด	เปิด	เปิด	เปิด	5	เปิด	เปิด	8
ปิด	ปิด	ปิด	ปิด	ปิด	1	4	4	4
		100%	91.67%	83.33%	33.33%	41.67%	58.33%	33%

ตารางที่ 4.4 แสดงการจดจำเสียงในกรณีต่างๆจากเสียงผู้หญิงเป็นต้นแบบ

กรณีที่เสียงต้นแบบเป็นผู้ชาย 5 คนๆละ 1 ครั้ง

เสียงต้นแบบ	คนในกลุ่ม ครั้งต่อไป		ผู้หญิง คนอื่น	
0	0	0	2	2
1	7	1	1	5
2	2	2	9	0
3	3	9	เปิด	3
4	4	4	4	4
5	1	8	9	8
6	0	0	2	2
7	7	1	0	ปิด
8	8	8	9	5
9	9	9	9	9
เปิด	1	3	9	8
ปิด	ปิด	1	4	4
	66.67%	41.67%	25%	25%

ตารางที่ 4.5 แสดงการจดจำเสียงในกรณีต่างๆจากเสียงผู้ชายเป็นต้นแบบ

กรณีที่เสียงต้นแบบเป็นผู้หญิง 5 คนๆละ 1 ครั้ง

คนในกลุ่ม ครั้งต่อไป		ชายคน อื่น	
0	0	9	0
1	1	1	8
2	2	เปิด	0
3	3	3	3
4	4	4	4
5	9	5	8
6	6	0	6
7	1	ปิด	1
8	5	5	8
9	9	9	3
เปิด	เปิด	8	เปิด
ปิด	ปิด	ปิด	1
	75%	50%	50%

ตารางที่ 4.6 แสดงการจดจำเสียงในกรณีต่างๆจากเสียงผู้หญิงเป็นต้นแบบ

กรณีที่เสียงต้นแบบเป็นผู้ชาย 1 คนๆละ 5 ครั้ง

เสียงต้นแบบ	ผู้ชาย คนอื่น		ผู้หญิง คนอื่น	
0	6	6	2	0
1	8	7	7	ปิด
2	2	0	2	0
3	3	5	9	3
4	4	4	4	4
5	8	1	9	8
6	6	0	2	2
7	7	ปิด	0	ปิด
8	5	5	9	8
9	9	9	9	9
เปิด	เปิด	5	เปิด	8
ปิด	4	4	4	4
	53%	16.67%	25%	41.67%

ตารางที่ 4.7 แสดงการจดจำเสียงในกรณีต่างๆจากเสียงผู้ชายเป็นต้นแบบ

กรณีที่เสียงต้นแบบเป็นผู้หญิง 5 คนๆละ 1 ครั้ง

เสียงต้นแบบ	หญิง คนอื่น		ชาย คนอื่น	
0	0	0	9	9
1	1	4	4	1
2	2	0	9	9
3	3	9	5	3
4	ปิด	4	4	4
5	8	8	1	8
6	6	6	0	0
7	7	ปิด	4	7
8	8	8	8	8
9	9	9	3	9
เปิด	8	เปิด	8	5
ปิด	ปิด	4	ปิด	1
	75%	41.67%	25%	50%

ตารางที่ 4.8 แสดงการจดจำเสียงในกรณีต่างๆจากเสียงผู้หญิงเป็นต้นแบบ

ตารางที่ 4.9 แสดงผลการทดลองในกรณีที่ใช้เสียงผู้ชาย 1 คนๆละ 1 ครั้ง เป็นต้นแบบทำโมเดลเสียง

Model	ชายคน	เดิม	ชายคนอื่น			หญิงคนอื่น		
0	6	6	6	0	6	2	6	6
1	1	1	7	5	8	1	1	7
2	2	6	6	2	6	8	2	2
3	9	3	9	9	8	9	9	9
4	4	4	4	4	4	4	4	4
5	5	5	8	0	9	9	9	5
6	6	0	6	0	6	2	0	2
7	1	1	4	7	7	0	4	7
8	9	8	9	9	8	9	9	8
9	9	9	9	9	9	9	9	8
	60%	60%	30%	50%	50%	30%	40%	50%

ตารางที่ 4.10 แสดงผลการทดลองในกรณีที่ใช้เสียงผู้หญิง 1 คนๆละ 1 ครั้ง เป็นต้นแบบทำโมเดลเสียง

Model	หญิงคน	เดิม	หญิงคนอื่น			ชายคนอื่น		
0	0	0	0	0	0	4	0	0
1	1	1	1	4	8	8	1	4
2	2	2	0	0	2	0	0	0
3	3	5	3	3	3	2	3	5
4	4	4	4	4	4	4	4	4
5	5	5	8	9	8	5	3	8
6	6	6	6	0	9	0	0	0
7	7	7	7	4	7	4	4	4
8	8	5	8	8	1	2	1	8
9	9	9	9	9	9	8	8	9
	100%	80%	80%	50%	60%	20%	40%	40%

ตารางที่ 4.11 แสดงผลการทดลองในกรณีที่ใช้เสียงผู้ชาย 1 คนๆละ 5 ครั้ง เป็นต้นแบบทำโมเดลเสียง

Model	คน เดิม		ชายคนอื่น			หญิงคนอื่น		
0	2	0	0	0	2	2	0	2
1	1	5	9	8	1	7	7	8
2	2	2	2	2	2	0	2	2
3	3	3	8	8	3	9	9	8
4	4	4	4	4	4	4	4	4
5	7	1	8	8	9	8	8	8
6	6	6	6	6	6	2	2	2
7	7	7	7	7	7	7	4	7
8	8	9	8	8	8	8	8	8
9	9	9	9	9	9	9	9	9
	80%	80%	60%	70%	80%	40%	50%	50%

ตารางที่ 4.12 แสดงผลการทดลองในกรณีที่ใช้เสียงผู้หญิง 1 คนๆละ 5 ครั้ง เป็นต้นแบบทำโมเดลเสียง

Model	คน เดิม		หญิงคนอื่น			ผู้ ชาย	
0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	4
2	2	2	2	2	2	0	0
3	3	3	3	3	3	8	3
4	4	4	4	4	4	4	4
5	5	5	8	5	8	5	9
6	6	6	6	0	0	6	6
7	7	7	7	7	7	4	4
8	8	8	8	1	8	9	8
9	9	9	9	9	9	9	9
	100%	100%	90%	80%	80%	60%	60%

ตารางที่ 4.13 แสดงผลการทดลองในกรณีที่ใช้เสียงผู้ชาย 5 คนๆละ 1 ครั้ง เป็นต้นแบบทำโมเดลเสียง

Model	ชายกลุ่มเดิม			ชายคน อื่น		หญิงคน อื่น	
0	0	0	0	2	0	0	2
1	1	7	1	4	1	1	1
2	0	2	0	0	2	9	2
3	3	3	3	9	9	9	9
4	4	4	4	4	4	4	4
5	1	8	5	5	5	5	5
6	0	6	0	0	0	2	6
7	1	7	1	1	4	2	4
8	8	8	8	5	5	5	5
9	9	9	9	9	9	9	9
	60%	80%	60%	30%	60%	50%	50%

ตารางที่ 4.14 แสดงผลการทดลองในกรณีที่ใช้เสียงผู้หญิง 5 คนๆละ 1 ครั้ง เป็นต้นแบบทำโมเดลเสียง

Model	หญิงกลุ่มเดิม			หญิงคน อื่น		ผู้ ชาย	
0	0	0	0	0	0	6	0
1	1	1	1	7	9	8	1
2	2	2	2	2	0	0	0
3	3	5	3	8	5	5	3
4	4	4	4	4	4	7	4
5	5	5	5	8	9	5	8
6	6	6	0	6	6	6	0
7	1	1	4	1	7	7	1
8	5	9	8	8	8	8	8
9	9	9	9	8	9	9	9
	80%	70%	80%	50%	60%	50%	60%

ตารางที่ 4.15 แสดงผลการทดลองในกรณีที่ใช้เสียงผู้ชาย 10 คนๆละ 1 ครั้ง เป็นต้นแบบทำโมเดลเสียง

Model	ชายในกลุ่ม			ชายคนอื่น			หญิงคนอื่น		
0	0	0	0	7	0	0	0	2	2
1	7	1	1	0	8	1	1	1	4
2	2	2	2	2	2	2	9	2	2
3	3	3	3	3	9	3	9	3	5
4	4	4	4	4	4	4	4	4	4
5	8	8	5	9	9	3	9	5	5
6	2	0	0	6	6	6	2	6	6
7	4	9	1	7	1	4	0	4	7
8	8	8	3	3	3	3	5	5	8
9	9	9	9	9	9	9	9	9	9
	60%	70%	70%	60%	50%	70%	40%	70%	70%

ตารางที่ 4.16 แสดงผลการทดลองในกรณีที่ใช้เสียงหญิง 10 คนๆละ 1 ครั้ง เป็นต้นแบบทำโมเดลเสียง

Model	หญิงในกลุ่ม			หญิงคนอื่น			ชายคนอื่น		
0	0	0	0	0	0	0	9	0	6
1	1	1	8	1	1	1	5	1	5
2	2	2	2	2	2	2	0	0	2
3	3	5	3	3	5	3	3	5	3
4	4	4	4	4	4	4	4	4	1
5	5	5	5	5	1	5	8	8	2
6	6	6	6	0	6	0	6	6	6
7	1	7	7	7	7	7	1	1	8
8	3	5	8	8	8	5	8	8	9
9	9	9	9	9	3	3	3	9	2
	80%	80%	80%	90%	70%	70%	40%	60%	30%

ตารางที่ 4.17 แสดงผลการทดลองในกรณีที่ใช้เสียงผู้ชาย 5 คนๆละ 1 ครั้ง และผู้หญิง 5 คนๆละ 1 ครั้ง เป็นต้นแบบทำโมเดลเสียง

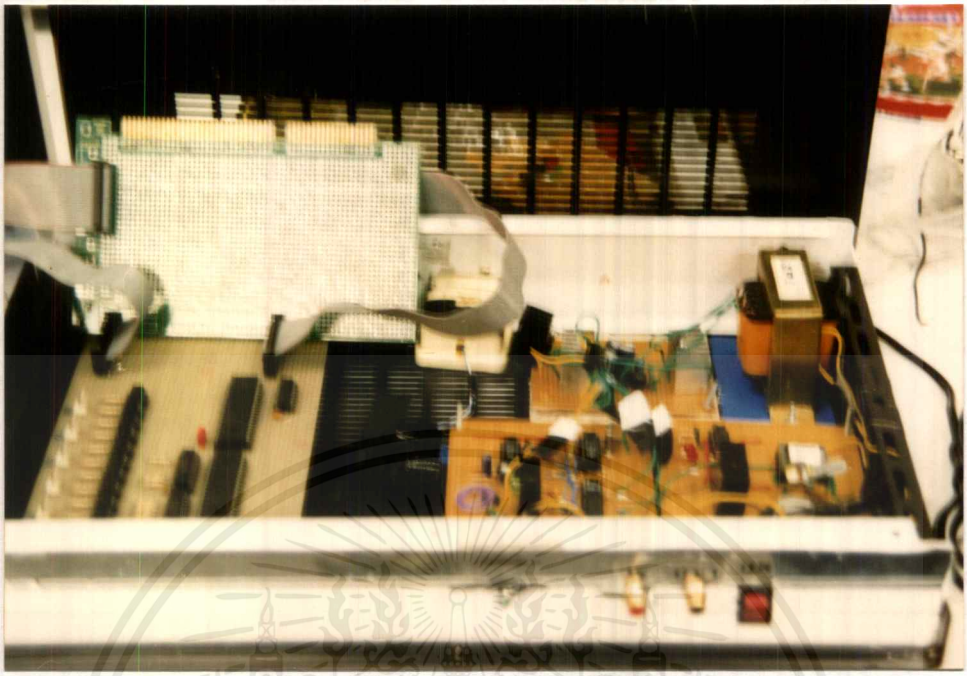
Model	ชายใน กลุ่ม		หญิงใน กลุ่ม		ชายคน อื่น		หญิงคน อื่น	
0	0	0	2	0	0	0	0	0
1	1	1	1	1	1	4	1	9
2	2	2	2	2	2	2	2	2
3	3	3	9	3	9	9	8	9
4	4	4	4	4	4	4	4	0
5	8	8	9	8	1	8	1	5
6	0	0	6	6	0	6	6	0
7	1	1	7	7	1	1	1	7
8	8	8	9	3	3	8	1	9
9	9	9	9	9	9	9	9	9
	70%	70%	60%	80%	50%	60%	60%	50%

ตารางที่ 4.18 แสดงผลการทดลองในกรณีที่ใช้เสียงผู้ชาย 5 คนๆละ 2 ครั้ง และผู้หญิง 5 คนๆละ 2 ครั้งเป็นต้นแบบทำโมเดลเสียง

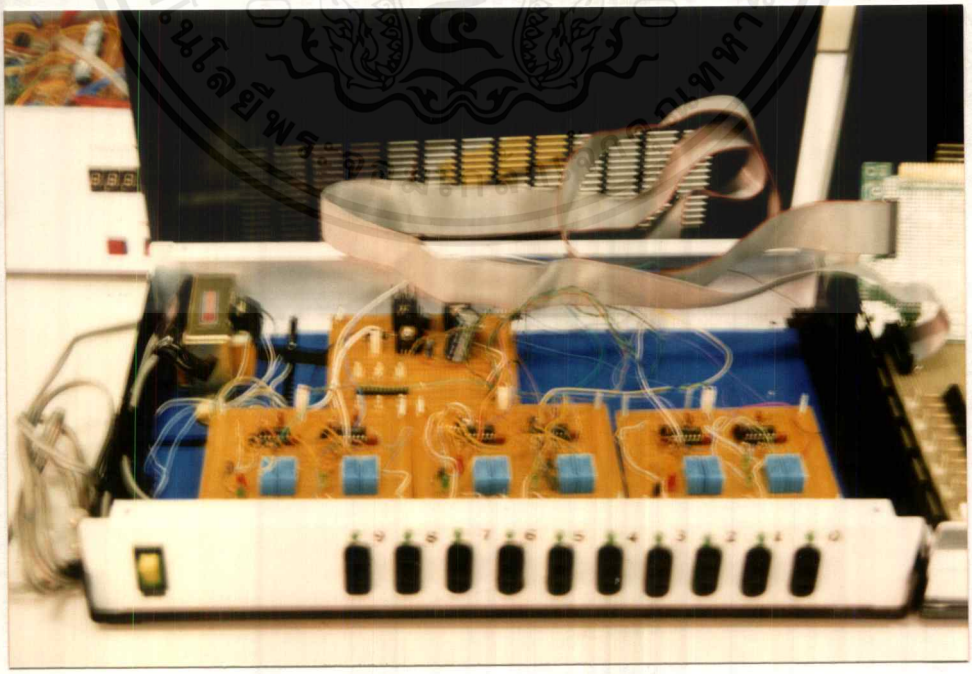
Model	ชายในกลุ่ม			หญิงในกลุ่ม			ชายคนอื่น			หญิงคนอื่น		
0	0	2	0	0	0	0	2	0	0	0	0	2
1	1	1	1	1	1	8	1	1	1	1	1	5
2	0	2	0	2	2	2	2	2	2	2	2	2
3	3	3	5	9	9	3	3	3	3	5	9	9
4	4	4	4	4	7	4	4	4	4	4	4	4
5	5	8	8	5	9	9	8	3	9	5	5	9
6	0	6	6	6	6	6	6	6	6	6	6	2
7	1	1	7	7	7	7	1	7	7	4	4	7
8	8	9	8	8	5	8	8	8	1	8	1	9
9	9	9	9	9	9	9	9	9	9	9	5	9
เปอร์เซ็นต์ความถูกต้อง	70%	60%	70%	90%	60%	80%	70%	90%	80%	80%	60%	40%

ตารางที่ 4.19 แสดงผลการทดลองในกรณีที่ใช้เสียงผู้ชาย 10 คนๆละ 1 ครั้ง และผู้หญิง 10 คนๆละ 1 ครั้งเป็นต้นแบบทำโมเดลเสียง

Model	ชายในกลุ่ม			หญิงในกลุ่ม			ชายคนอื่น			หญิงคนอื่น		
0	0	0	0	0	0	0	1	0	0	0	0	0
1	7	1	1	1	1	9	0	1	7	1	1	1
2	0	2	2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	5	3	3	3	3
4	4	4	4	4	4	4	7	1	4	4	4	4
5	8	5	5	5	5	5	5	5	3	8	8	5
6	6	6	6	6	6	6	6	6	6	6	6	0
7	7	1	1	1	7	7	7	1	7	1	1	1
8	8	8	9	9	3	9	3	3	3	8	1	3
9	9	9	9	9	9	9	9	9	9	9	3	3
	70%	90%	80%	80%	90%	80%	60%	60%	70%	80%	60%	60%



รูปที่ 4.35 แสดงรูปของชิ้นงานในส่วนของเครื่องตอบรับโทรศัพท์



รูปที่ 4.36 แสดงรูปชิ้นงานในส่วนของวงจรขับรีเลย์

บทที่ 5
บทวิจารณ์และบทสรุป

5.1 สรุปผลการทดลอง

จากการทดลองในส่วนของเครื่องตอบรับโทรศัพท์อัตโนมัติ เครื่องตอบรับโทรศัพท์สามารถตอบรับโทรศัพท์ได้ไม่มีปัญหา โดยการทำงานในแต่ละส่วนสามารถทำงานได้ถูกต้องตามทฤษฎี แต่ในส่วนของการเชื่อมต่อกับคอมพิวเตอร์ยังมีปัญหาอยู่ตรงที่ถ้านำไปเชื่อมต่อกับเครื่องคอมพิวเตอร์คนละเครื่องจะทำให้ความแรงของสัญญาณเสียงต่างกันไปด้วย ทั้งนี้เกิดขึ้นมาจากซาวน์การ์ด ถ้าซาวน์การ์ดคนละชนิดและยี่ห้อนั้นก็มีผลต่อสัญญาณเสียงที่เข้ามาที่คอมพิวเตอร์ โดยถ้าซาวน์การ์ดมีคุณภาพที่ดีก็จะสามารถให้สัญญาณเสียงที่ใส ดัง และคมชัด แต่ถ้าซาวน์การ์ดมีคุณภาพไม่ดีก็จะให้เสียงออกมาแย่จนไม่สามารถนำสัญญาณเสียงนั้นมาวิเคราะห์ได้ ดังนั้นจึงจำเป็นที่จะต้องใช้คอมพิวเตอร์ตัวเดียวกันตลอดการทดลองและใช้งาน

ส่วนผลการทดลองในส่วนของการรู้จำเสียงสามารถสรุปผลการทดลองได้เป็นข้อๆดังนี้

5.1.1 ความถูกต้องในการรับรู้เสียงของ Set ที่เป็นบุคคลเดียวกับต้นแบบจะถูกต้องมากกว่าบุคคลที่ไม่ใช่เป็นต้นแบบ

5.1.2 ความถูกต้องในการจำเสียงจากผลการทดลองจะเห็นได้ว่าเสียงของผู้หญิงจะสามารถจำได้ดีกว่าเสียงของผู้ชาย

5.1.3 ความถูกต้องของการรู้จำกับจำนวน Set ของต้นแบบ เมื่อเพิ่มจำนวนบุคคลที่ใช้เป็นต้นแบบขึ้นเรื่อยๆ ผลความถูกต้องในการทดสอบการรู้จำจะดีขึ้น ในช่วงแรกการเพิ่มจำนวนบุคคลเพียงเล็กน้อยก็มีผลทำให้ความถูกต้องในการรู้จำเพิ่มขึ้นอย่างมาก จนเพิ่มจำนวนบุคคลต้นแบบถึงระยะหนึ่ง อัตราความถูกต้องค่อยๆ ลดลง สรุปได้ว่า HMM ต้องการต้นแบบในการรู้จำจำนวน มาก ให้หลากหลายเพื่อครอบคลุมความเป็นไปได้ของเสียงนั้นๆ ได้มากขึ้น

5.1.4 จากผลการทดลอง ทำให้เห็นว่าวิธีการ HMM เป็นวิธีที่ใช้ในการหาต้นแบบที่เหมาะสมในรูปแบบของการจำลองของความน่าจะเป็นในการเปลี่ยน state และความน่าจะเป็นในการเกิดเหตุการณ์ จึงทำให้ผลการทดสอบกับบุคคลที่ไม่บุคคลต้นแบบมีความถูกต้องค่อนข้างมาก เมื่อจำนวนต้นแบบมีความหลากหลายเพียงพอ

5.1.5 ในการทดสอบความถูกต้องของการรู้จำกับจำนวน set ต้นแบบ โดยการค่อยๆ เพิ่มจำนวนบุคคลที่ใช้เป็นต้นแบบขึ้นเรื่อยๆ ผลความถูกต้องในการทดสอบการรู้จำก็จะดีขึ้น ในช่วงแรกการเพิ่มจำนวนบุคคลเพียงเล็กน้อยก็มีผลทำให้ความถูกต้องในการรู้จำเพิ่มขึ้นอย่างมาก จึงสรุปได้ว่า HMM ต้องการต้นแบบในการรับรู้จำนวนมากให้มีความหลากหลายเพื่อครอบคลุมความเป็นไปได้ของเสียงนั้นๆ ได้มากขึ้นเช่นกัน

5.1.6 สัมประสิทธิ์ LPC ที่คำนวณได้จากโปรแกรม 'LPC' โดยใช้ภาษา C++ มีค่าใกล้เคียงกับสัมประสิทธิ์ LPC ที่ใช้โปรแกรม MATLAB จากที่เคยมีการทดลองไว้แล้ว

5.1.7 ค่าสัมประสิทธิ์ของเสียงตัวเลขใด ๆ ที่วิเคราะห์ได้ใน 1 เฟรม ประกอบไปด้วย

- อัตราขยาย (Gain) 1 ค่า
- สัมประสิทธิ์ LPC 12 ค่า
- สัมประสิทธิ์เซปสตรัม 19 ค่า

5.1.8 สัมประสิทธิ์เซปสตรัมนั้นใช้ในการปรับปรุงสัมประสิทธิ์ LPC ให้คงลักษณะของเสียงได้มากขึ้น และจำนวนสัมประสิทธิ์ที่ได้จากการวิเคราะห์จะมีมากกว่าสัมประสิทธิ์ LPC

5.1.9 การเวทค่าพารามิเตอร์จะช่วยลดความผิดพลาดอันเกิดจากรอยต่อของเฟรมซึ่งทำให้ได้ค่าสัมประสิทธิ์ที่ถูกต้องยิ่งขึ้น

5.1.10 การพูดเสียงเดียวกัน ในครั้งใหม่จะได้สัมประสิทธิ์ชุดใหม่ที่มีลักษณะใกล้เคียงกับชุดเดิม แต่เปลี่ยนตำแหน่งไป เนื่องจากการเริ่มต้นวิเคราะห์เสียงในแต่ละครั้งไม่ได้เริ่มที่ตำแหน่งเดียวกัน

5.1.11 เวลาที่ใช้พูดของแต่ละเสียงในการทดลองมีผลต่อสัมประสิทธิ์ที่วิเคราะห์ เพราะฉะนั้นเวลานำเสียงมาวิเคราะห์ ควรพูดให้เป็นเสียงตามธรรมชาติมากที่สุด ซึ่งจะทำให้ได้ค่าสัมประสิทธิ์ที่ถูกต้อง

5.1.12 เดิมได้มีการออกแบบเสียงคำสั่งเช่น "เปิด-1" หรือ "ปิด-1" จากผลการทดลองที่ได้เก็บผลลงมาจะเห็นได้ว่าต้องมีการทำโมเดลเสียงถึง 12 เสียง คือ 0-9 ,เปิด ,ปิด ซึ่งจากการเพิ่มโมเดลนี้เองเป็นผลทำให้การรู้จำของเสียงมีค่าลดลงทำให้การใช้งานจริงมีข้อผิดพลาดเยอะ ดังนั้นจึงออกแบบรหัสคำสั่งให้เป็น "0-6" หมายความว่า เปิดสวิตช์เบอร์ 6 "1-9" หมายความว่าปิดสวิตช์เบอร์ 9 ซึ่งรหัสคำสั่งจะเป็น 2 คำ ถ้าเกิดมีเสียงอื่นเข้ามาเครื่องก็จะไม่มีการทำงาน ซึ่งจะได้ผลดีเมื่อผู้เป็นต้นแบบเสียงใช้งานเองถ้าเกิดเป็นผู้อื่นใช้งานก็อาจจะมีข้อผิดพลาดจนไม่สามารถสั่งงานได้เลย

5.1.13 จากการที่เราตั้งว่าให้ผู้เป็นต้นแบบเสียงจะเป็นผู้ที่สามารถใช้งานได้เท่านั้นเราก็สามารถที่จะทำโมเดลไว้อีก 1 ชุดในกรณีที่ต้องการให้ใครใช้งานก็ได้โดยที่โมเดลนั้นต้องเป็นการรวบรวมเสียงที่หลากหลายจึงสามารถให้คนอื่นใช้งานได้

5.1.14 การสั่งงานเครื่องใช้ไฟฟ้าจะสามารถสั่งงานได้ดีเมื่อไม่มีการพูดในจังหวะที่เป็นการคำนวณของโปรแกรม และการสั่งงานไม่สามารถที่จะสั่งงานในช่วงของการคำนวณแรกได้เพราะเมื่อมีการยกหูรับสายโทรศัพท์อัตโนมัติจะมีเสียงกระตุกเข้าไปในซาวน์การ์ด

5.1.15 การสั่งงานสามารถทำได้ดี สามารถที่จะใช้งานในระบบที่เป็น real-time ได้จริง คือหมายความว่าเราสามารถที่เปิดเครื่องทิ้งไว้แล้วสามารถที่จะโทรมาสั่งงานเมื่อไรก็ได้

5.1.16 การใช้งานในขั้นตอนการทำโมเดลเสียงได้มีการพัฒนาในด้านการใช้งานโดยที่เราไม่จำเป็นที่จะต้องพิมพ์ชื่อไฟล์ หรือมีการนับไฟล์เป็นขั้นๆ หรือต้องมีรูปแบบการตั้งชื่อไฟล์คือหมายความว่าสามารถที่จะเรียนรู้การใช้งานได้ง่าย ส่วนใหญ่สามารถทำงานด้วย mouse ไม่จำเป็นต้องป้อนค่าให้ยุ่งยาก

5.1.17 ช่วงในการวิเคราะห์ในการทำงานระบบ real-time เราสามารถที่จะออกแบบช่วงนั้นได้เองจากการทดลอง เราจะพบว่าช่วงในการวิเคราะห์มีค่าระหว่าง 6-10 วินาทีเป็นช่วงที่เหมาะสม ทั้งนี้จะ

ตั้งช่วงในการวิเคราห์ยาวหรือสั้นก็ขึ้นอยู่กับการออกแบบคำสั่งนั้นๆ ในที่นี้ออกแบบไว้ 2 คำ 6-10 วินาที ก็เลยเป็นช่วงที่เหมาะสม

5.1.18 การเขียนโปรแกรมด้วยโปรแกรม VISUAL C++ ทำให้มีความสะดวกในการแก้ไขโปรแกรมโดยที่โปรแกรมจะสามารถที่จะแยกออกเป็นส่วนๆได้ และการใช้งานจะมีความสวยงามและการจัดรูปแบบการทำงานที่ดีกว่า และขนาดของโปรเจ็ค และไฟล์ที่ใช้ในการสร้างจะมีขนาดเล็กกว่าจึงทำให้กินเนื้อที่ใน Hard Disk น้อย

5.2 ข้อแนะนำและปัญหาที่พบ

5.2.1 เสียงที่อัดมาจากโทรศัพท์มีสัญญาณรบกวนจากภายนอก ทำให้เสียงมีลักษณะที่เปลี่ยนไปบ้าง ควรมีขั้นตอนการลดสัญญาณรบกวนเป็น pre-processing ก่อนจะทำให้ได้เสียงที่มีความสมบูรณ์เพียงพอที่จะนำไปวิเคราะห์ต่อไป

5.2.2 Algorithm ที่ใช้ในการตัดหัวท้ายของเสียงยังไม่สมบูรณ์เพียงพอ ยังมีการตัดคำบางคำเกินหรือขาด

5.2.3 ความผิดพลาดจากการทดลองอาจเกิดได้จาก 2 สาเหตุ คือความผิดปกติของเสียงเอง หรือจากขั้นตอนการ Quantize มีผลทำให้การตัดสินใจผิดพลาด

5.2.4 การวัดความคลาดเคลื่อนของการวิจัยนี้ใช้วิธี Square error ซึ่งง่ายและรวดเร็ว แต่ยังไม่เหมาะเพียงพอสำหรับ Linear Predictive ควรใช้วิธี Itakura distance แต่วิธีนี้ซับซ้อนมากและยุ่งยากในการคำนวณ

5.2.5 การใช้วิธี HMM จะรู้จำได้อย่างมีประสิทธิภาพเมื่อมีตัวแบบมาก จึงควรศึกษาวิธีอื่น ๆ ที่ลดจุดด้อยนี้

5.2.6 จากขั้นตอนการควอนไตซ์ เนื่องจาก Codebook ไม่ได้มาตรฐานพอ ทำให้จัดระดับออกมาผิดพลาด โดยการควอนไตซ์ที่ผิดพลาดนั้นไม่จำเป็นต้องผิดพลาดทั้งหมด อาจผิด พลาดบางตัวที่มีความน่าจะเป็นมาก ก็สามารถทำให้ความน่าจะเป็นรวมเปลี่ยน ซึ่งมีผลทำให้การตัดสินใจผิดพลาดด้วย

5.2.7 อาจเกิดในขั้นตอนการสร้างแบบจำลอง เนื่องจากแบบจำลองที่ได้ไม่ดีพอ คือ พารามิเตอร์ของแบบจำลองที่คำนวณได้ไม่สอดคล้องกับเสียงนั้นๆ ซึ่งอาจเกิดจาก set ลำดับเหตุการณ์ที่ใช้ในการเทรนแบบจำลองน้อยเกินไป, ค่าเริ่มต้นไม่ดี, รอบของการเทรนไม่เหมาะสม เป็นต้น

5.2.8 Codebook ขนาด 64 สำหรับต้นแบบที่ใช้ทดสอบในปริญญานิพนธ์นี้ เพียงพอกับการรู้จำเสียง แต่ถ้าต้นแบบมีจำนวนมากกว่านี้ ขนาดของ Codebook ก็ควรมากกว่านี้

5.2.9 ขั้นตอนในการยกหูรับโทรศัพท์อัตโนมัติจะทำให้เกิดเสียงกระตุกขึ้นมาจึงทำให้เราไม่สามารถที่จะพูดสั่งงานในช่วงแรกที่รับสายเข้ามาได้ จึงต้องรออีกช่วงของการวิเคราะห์ ถ้าเราสามารถที่จะแก้ปัญหาเสียงกระตุกได้ก็จะสามารถสั่งงานที่ช่วงการคำนวณแรกได้

5.2.10 เราไม่สามารถที่จะรู้ได้ว่าต้นแบบเสียงมากจำนวนเท่าไรจึงจะเป็นค่าที่เหมาะสมและทำให้จดจำเสียงได้ดีที่สุด ทั้งนี้ต้องขึ้นอยู่กับสภาวะ และค่าต่างๆจึงไม่สามารถที่จะบอกตายตัวว่าต้นแบบเสียงจำนวนเท่าไรจึงจะให้ผลดีที่สุด ดังนั้นทั้งหมดนี้ก็ต้องขึ้นอยู่กับการทดลอง

5.2.11 ในการทำโมเดลเสียงในกรณีที่มีเสียงต้นแบบมากๆจะต้องใช้เวลาในการคำนวณนานมาก ซึ่งกว่าเราจะได้โมเดลเสียง 1 เสียงจะทำให้เสียเวลา แต่ในขั้นตอนการจดจำใช้เวลาน้อยดีอยู่แล้ว แต่เวลาในการจดจำก็ต้องขึ้นอยู่กับขนาดของ Codebook ซึ่งโดยปกติจะใช้เวลาในการจำไม่เกิน 5 วินาที ซึ่งจากการศึกษาการจำเสียงด้วยวิธีอื่นๆ เมื่อเปรียบเทียบกันแล้ววิธีนี้เร็วมากจนสามารถที่จะทำให้เป็น real-time ได้ ควรจะแก้ไขในการทำโมเดลเสียงมากกว่า

5.3 แนวทางพัฒนา

ควรที่จะมีการพัฒนาทำ Codebook ที่มากกว่าขนาด 64 แต่ทั้งนี้ต้องใช้เวลาในการคำนวณนานมาก ยิ่งถ้าเรามีการทำต้นแบบเสียงมาก ก็จะต้องใช้เวลาในการทำงานมากซึ่งบางครั้งอาจจะเป็นหลายชั่วโมง แต่ถ้าให้ผลในการจำเสียงได้ดีก็คุ้มค่าแก่เวลาที่เสียไป ควรลองหา software ในการเขียนโปแกรมเผื่อว่าจะมีโปรแกรมที่สามารถทำงานได้เร็วกว่า ในส่วนของวิธีการคำนวณ ขั้นตอนต่างๆก็ดีอยู่แล้ว ส่วนเสียงในการเก็บเป็นต้นแบบก็ควรเก็บจากสถานะการณ์ใช้งานจริงจึงทำให้ได้ผลดีที่สุด คือ ถ้าเราคิดว่าจะสั่งงานด้วยการใช้โทรศัพท์มือถือ เสียงที่เอามาเป็นแบบก็ควรที่จะมาจากเสียงของโทรศัพท์มือถือนั้นๆ ในการพัฒนาควรที่จะลองหาวิธีการที่จะกรองสัญญาณเสียงที่เข้ามาให้ดี อย่างเช่นในกรณีที่โทรศัพท์เสียงซ่ามากๆ ในส่วนของเครื่องตอบรับควรจะมีการกองสัญญาณรบกวนออกไป ทั้งนี้ก็มีแนวทางการพัฒนาต่างๆมากมายซึ่งก็อยู่ที่การใช้งานของเรา

```
#if !defined(AFX_VQLNDIA_H__E03C8B82_26EB_11D2_AE2D_444553540001__INCLUDED_)
#define AFX_VQLNDIA_H__E03C8B82_26EB_11D2_AE2D_444553540001__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
// Vqlndia.h : header file
//

/////////////////////////////////////////////////////////////////////////////
// Vqlndia dialog

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

// Dialog Data
	//{{AFX_DATA(Vqlndia)
	enum { IDD = IDD_DIALOG1 };
	CString m_input;
	CString m_out;
	CString m_outtext;
	//}}AFX_DATA


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

```
// Implementation
protected:

    // Generated message map functions
    //{{AFX_MSG(Vqlndia)
    afx_msg void OnRUN();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

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

#endif // !defined(AFX_VQLNDIA_H__E03C8B82_26EB_11D2_AE2D_444553540001__INCLUDED_)
```

```
#if !defined(AFX_SEGMENT_H__C572A8A5_2792_11D2_AE2D_444553540001__INCLUDED_)
#define AFX_SEGMENT_H__C572A8A5_2792_11D2_AE2D_444553540001__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
// Segment.h : header file
//

/////////////////////////////////////////////////////////////////////////////
// Segment view

class Segment : public CView
{
//protected:
public:
    Segment();          // protected constructor used by dynamic creation
//  DECLARE_DYNCREATE(Segment)

bool   segmentEnergy(CString energyname
                     ,CString segmentname
                     ,CString segmenttextname
                     ,CString summary
                     ,int lowpercent
                     ,int mininterval
                     ,int minnumber
                     ,int setno);

    bool   energy(CString namewav
                  ,CString nameeng
                  ,CString nameengtxt
                  ,CString namewavtxt
                  ,int sampleshift
```

```
                    segmentdia.txt


                    ,int samplewindow);


    bool    editwave(CString inputfilename,CString segmentfilename);
    bool    editrecog(CString inputfilename,CString segmentfilename,CString resultfilename);

// Attributes
public:
#define MAXSEGMENT                  30
// Operations
public:


// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(Segment)
    protected:
    virtual void OnDraw(CDC* pDC);    // overridden to draw this view
    //}}AFX_VIRTUAL


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


    // Generated message map functions
protected:
    //{{AFX_MSG(Segment)
        // NOTE - the ClassWizard will add and remove member functions here.
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
```

};

//

//{{AFX_INSERT_LOCATION}}

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

#endif // !defined(AFX_SEGMENT_H__C572A8A5_2792_11D2_AE2D_444553540001__INCLUDED_)

```
// 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__E24D24F4_263E_11D2_AE2D_444553540001__INCLUDED_)
#define AFX_STDAFX_H__E24D24F4_263E_11D2_AE2D_444553540001__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 <afxsock.h>            // MFC socket extensions

// This macro is the same as IMPLEMENT_OLECREATE, except it passes TRUE
//  for the bMultiInstance parameter to the COleObjectFactory constructor.
//  We want a separate instance of this application to be launched for
//  each OLE automation proxy object requested by automation controllers.
#ifndef IMPLEMENT_OLECREATE2
#define IMPLEMENT_OLECREATE2(class_name, external_name, l, w1, w2, b1, b2, b3, b4, b5, b6, b7, b8) \
    AFX_DATADEF COleObjectFactory class_name::factory(class_name::guid, \
        RUNTIME_CLASS(class_name), TRUE, _T(external_name)); \
    const AFX_DATADEF GUID class_name::guid = \
        { l, w1, w2, { b1, b2, b3, b4, b5, b6, b7, b8 } };
```

```
#endif // IMPLEMENT_OLECREATE2

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

#endif // !defined(AFX_STDAFX_H__E24D24F4_263E_11D2_AE2D_444553540001__INCLUDED_)
```

```
#if !defined(AFX_WAVE_H__50877B45_D6D5_11D2_AE2D_444553540001__INCLUDED_)
#define AFX_WAVE_H__50877B45_D6D5_11D2_AE2D_444553540001__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
// Wave.h : header file
//

/////////////////////////////////////////////////////////////////////////////
// Wave view

class Wave : public CView
{
public:
    Wave();         // protected constructor used by dynamic creation

        bool openwav(CString name);
        void recordwav(void);
        bool savewav(CString name);
        void stopwav(void);
        void closewav(void);
        void playwav(void);
        void easyplay(CString filename,int delaytime);
        void cal_wav_play(char input[30],int    wait_time);


// Attributes
public:

// Operations
public:
```

```
// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(Wave)
    protected:
    virtual void OnDraw(CDC* pDC);    // overridden to draw this view
    //}}AFX_VIRTUAL

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

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

/////////////////////////////////////////////////////////////////////////////

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

#endif // !defined(AFX_WAVE_H__50877B45_D6D5_11D2_AE2D_444553540001__INCLUDED_)
```

```
// Record.h : header file
//

#ifndef _RECORDThread_
#define _RECORDThread_

#if !defined(AFX_RECORD_H__93A99C64_E1F1_11D2_AE2D_444553540001__INCLUDED_)
#define AFX_RECORD_H__93A99C64_E1F1_11D2_AE2D_444553540001__INCLUDED_

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

#include "art3dlg.h"
#include "wave.h"
#define MAXSEGMENT 30

/////////////////////////////////////////////////////////////////////////////
// CRecord thread

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

 // Attributes
 public:
  CArt3Dlg *m_pOwner;

 // Operations
 public:
  void SetOwner(CArt3Dlg* pOwner) {m_pOwner=pOwner;};
```

```
// BOOL Energy(CString namewav);
    bool Auto(CString namewav);
    bool editrecog(CString inputfilename,CString segmentfilename,CString resultfilename);

 void KillThread();
 void ClearUpThread();

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

// Implementation
protected:
 virtual ~CRecord();

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

DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////

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

#endif // !defined(AFX_RECORD_H__93A99C64_E1F1_11D2_AE2D_444553540001__INCLUDED_)
```

record.txt

#endif // _RECORDThread_

```
#if !defined(AFX_VQLN_H__E03C8B83_26EB_11D2_AE2D_444553540001__INCLUDED_)
#define AFX_VQLN_H__E03C8B83_26EB_11D2_AE2D_444553540001__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
// Vqln.h : header file
//
#define vector_dimension     19
#define number_of_codebook   64
#define REJECT                0
#define ACCEPT                1
#define REJECT_VALUE          0.01
/////////////////////////////////////////////////////////////////////////
// Vqln view

class Vqln : public CView
{
public:
//protected:
    Vqln();         // protected constructor used by dynamic creation
    //DECLARE_DYNCREATE(Vqln)

    bool open_all_file_vq(CString      inputfilename
                          ,CString     outputfilename
                          ,CString     outputtextfilename
                          );
    int         find_number_of_frame_vq(void);
    bool   allocate_data_memory(void);
    void   read_training_set(void);
    void   find_codebook(void);
    void   random_start_centroid_from_training_set(void);
    int         check_distance(void);
```

```
    void   copy_final_centroid_to_previous_centroid(void);
    void   write_codebook_file(void);
    void   close_all_file_vq(void);
    void   RunVqln(CString       inputfilename
                   ,CString      outputfilename
                   ,CString      outputtextfilename
                   );


// Attributes
public:

    FILE   *training_set_file;
    FILE   *codebook_file;
    FILE   *codebook_text_file;
    FILE   *random_text_file;
    FILE   *distortion_text_file;
    float  *training_set;
    float  *previous_centroid;
    float  *final_centroid;
    char   *min_index;
    double  vector_distance[number_of_codebook];

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(Vqln)
    protected:
    virtual void OnDraw(CDC* pDC);     // overridden to draw this view
    //}}AFX_VIRTUAL
```

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

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

/////////////////////////////////////////////////////////////////////////////

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

#endif // !defined(AFX_VQLN_H__E03C8B83_26EB_11D2_AE2D_444553540001__INCLUDED_)
```

```
#if !defined(AFX_VQCOMDIA_H__D86B65C1_2736_11D2_AE2D_444553540001__INCLUDED_)
#define AFX_VQCOMDIA_H__D86B65C1_2736_11D2_AE2D_444553540001__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
// Vqcomdia.h : header file
//

/////////////////////////////////////////////////////////////////////////////
// Vqcomdia dialog

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

// Dialog Data
	//{{AFX_DATA(Vqcomdia)
	enum { IDD = IDD_VQCOM };
	CListBox	m_list1;
	CListBox	m_list;
	CString m_input;
	CString m_number;
	int number_of_file;
	CString m_text;
	CString m_output;
	//}}AFX_DATA


// Overrides
	// ClassWizard generated virtual function overrides
```

```
    //{{AFX_VIRTUAL(Vqcomdia)
    protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
    //}}AFX_VIRTUAL

// Implementation
protected:

    // Generated message map functions
    //{{AFX_MSG(Vqcomdia)
    afx_msg void OnShowList();
    afx_msg void OnRUN();
    afx_msg void OnSaveAs();
    afx_msg void OnShowSave();
    afx_msg void OnShowsave2();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

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

#endif // !defined(AFX_VQCOMDIA_H__D86B65C1_2736_11D2_AE2D_444553540001__INCLUDED_)
```

```
#if !defined(AFX_VQCOM_H__F970D301_2730_11D2_AE2D_444553540001__INCLUDED_)
#define AFX_VQCOM_H__F970D301_2730_11D2_AE2D_444553540001__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
// Vqcom.h : header file
//

/////////////////////////////////////////////////////////////////////////////
// Vqcom view
#define  vector_dimension     19
#define  number_of_codebook    64


class Vqcom : public CView
{
public:
//protected:
    Vqcom();        // protected constructor used by dynamic creation
//    DECLARE_DYNCREATE(Vqcom)

bool   open_codebook_and_output_file(CString   codebookfilename
                                     ,CString    codebook_i
ndex_file_name
                                     ,CString    codebookin
dextextfilename);
    bool   allocate_codebook_memory(void);
    void   read_codebook_file(void);
    void   get_number_of_input_file_and_set_codebook_index_file_pointer(int number_of_input_file);
    bool   open_input_vector_file(CString input_vector_file_name);
    int        find_number_of_frame(void);
    bool   allocate_input_vector_and_min_index_memory(void);
```

```
    void   read_input_vector(void);
    void   find_codebook_index(void);
    void   write_codebook_index_file(int file_number);
    void   free_input_vector_and_min_index_memory(void);
    void   close_input_vector_file(void);
    bool   write_codebook_index_text_file(CString codebook_index_file_name);
    void   free_codebook_memory(void);
    void   close_all_vq_file(void);


// Attributes
public:

    FILE   *codebook_file;
    FILE   *codebook_index_file;
    FILE   *codebook_index_text_file;
    FILE   *input_vector_file;
    float  *codebook;
    float  *input_vector;
    char   *min_index;
    double  vector_distance[number_of_codebook];

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(Vqcom)
    protected:
    virtual void OnDraw(CDC* pDC);     // overridden to draw this view
    //}}AFX_VIRTUAL

// Implementation
```

```
//protected:
public:
    virtual ~Vqcom();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

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

/////////////////////////////////////////////////////////////////////////////

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

#endif // !defined(AFX_VQCOM_H__F970D301_2730_11D2_AE2D_444553540001__INCLUDED_)
```

```
#if !defined(AFX_SEGMENT_H__C572A8A5_2792_11D2_AE2D_444553540001__INCLUDED_)
#define AFX_SEGMENT_H__C572A8A5_2792_11D2_AE2D_444553540001__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
// Segment.h : header file
//

/////////////////////////////////////////////////////////////////////////////
// Segment view

class Segment : public CView
{
//protected:
public:
    Segment();          // protected constructor used by dynamic creation
//  DECLARE_DYNCREATE(Segment)

bool   segmentEnergy(CString energyname
                     ,CString segmentname
                     ,CString segmenttextname
                     ,CString summary
                     ,int lowpercent
                     ,int mininterval
                     ,int minnumber
                     ,int setno);

    bool   energy(CString namewav
                  ,CString nameeng
                  ,CString nameengtxt
                  ,CString namewavtxt
                  ,int sampleshift
```

```
                segment.txt

                ,int samplewindow);

    bool    editwave(CString inputfilename,CString segmentfilename);
    bool    editrecog(CString inputfilename,CString segmentfilename,CString resultfilename);

// Attributes
public:
#define MAXSEGMENT                  30
// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(Segment)
    protected:
    virtual void OnDraw(CDC* pDC);    // overridden to draw this view
    //}}AFX_VIRTUAL

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

    // Generated message map functions
protected:
    //{{AFX_MSG(Segment)
        // NOTE - the ClassWizard will add and remove member functions here.
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
```

segment.txt

};

//

//{{AFX_INSERT_LOCATION}}

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

#endif // !defined(AFX_SEGMENT_H__C572A8A5_2792_11D2_AE2D_444553540001__INCLUDED_)

```
#if !defined(AFX_SEGANDRECOG_H__CDAD94A8_27B3_11D2_AE2D_444553540001__INCLUDED_)
#define AFX_SEGANDRECOG_H__CDAD94A8_27B3_11D2_AE2D_444553540001__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
// Segandrecog.h : header file
//

/////////////////////////////////////////////////////////////////////////////
// Segandrecog dialog

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

// Dialog Data
	//{{AFX_DATA(Segandrecog)
	enum { IDD = IDD_SEG_RECOG };
	CListBox	m_list;
	CString m_input;
	//}}AFX_DATA


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

```
// Implementation
protected:

    // Generated message map functions
    //{{AFX_MSG(Segandrecog)
    afx_msg void OnShowList();
    afx_msg void OnRun();
    afx_msg void OnAddinput();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

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

#endif // !defined(AFX_SEGANDRECOG_H__CDAD94A8_27B3_11D2_AE2D_444553540001__INCLUDED_)
```

```
#if !defined(AFX_RECOGDIA_H__C572A8A4_2792_11D2_AE2D_444553540001__INCLUDED_)
#define AFX_RECOGDIA_H__C572A8A4_2792_11D2_AE2D_444553540001__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
// Recogdia.h : header file
//

/////////////////////////////////////////////////////////////////////////////
// Recogdia dialog

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

// Dialog Data
	//{{AFX_DATA(Recogdia)
	enum { IDD = IDD_RECOG };
	CListBox	m_list;
	CString m_input;
	//}}AFX_DATA


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

```
// Implementation
protected:

    // Generated message map functions
    //{{AFX_MSG(Recogdia)
    afx_msg void OnShow();
    afx_msg void OnRUN();
    afx_msg void OnInput();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

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

#endif // !defined(AFX_RECOGDIA_H__C572A8A4_2792_11D2_AE2D_444553540001__INCLUDED_)
```

```
#if !defined(AFX_RECOG_H__C572A8A3_2792_11D2_AE2D_444553540001__INCLUDED_)
#define AFX_RECOG_H__C572A8A3_2792_11D2_AE2D_444553540001__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
// Recog.h : header file
//
#define numberofmodel   10
#define N1              6
#define K               64
/////////////////////////////////////////////////////////////////////////////
// Recog view

class Recog : public CView
{
//protected:
public:
    Recog();        // protected constructor used by dynamic creation
//  DECLARE_DYNCREATE(Recog)
    bool   load_unknown_word_file(void);
    bool   allocate_memory(void);
    bool   load_model(int model_name);
    void   viterbi(int model);
    int        display_recognized_word(void);
    void   free_mem(void);
    int        auto_recog(CString input);

// Attributes
public:
    FILE   *input_wav_file;
    FILE   *parameter_file;
    FILE   *unknown_word_file;
```

```
    FILE   *result_file;
    char   T;
    char   *O;
    double *dt;
    char   *ar;
    char   *q_st;
    double *p_st;
    float *Buf_aprime;
    float *Buf_bprime;
    FILE   *model_file;
    CString *model_file_name;
    float  a[N1][N1];
    float  b[N1][K];
    double d1[N1];
    double dmax;
    int          max_index;
    char   max_index_text[5];

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(Recog)
    protected:
    virtual void OnDraw(CDC* pDC);     // overridden to draw this view
    //}}AFX_VIRTUAL

// Implementation
//protected:
    public:
    virtual ~Recog();
#ifdef _DEBUG
```

```
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

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

/////////////////////////////////////////////////////////////////////////////

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

#endif // !defined(AFX_RECOG_H__C572A8A3_2792_11D2_AE2D_444553540001__INCLUDED_)
```

```
#if !defined(AFX_OMBINEDIA_H__E03C8B81_26EB_11D2_AE2D_444553540001__INCLUDED_)
#define AFX_OMBINEDIA_H__E03C8B81_26EB_11D2_AE2D_444553540001__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
// ombinedia.h : header file
//

/////////////////////////////////////////////////////////////////////////////
// Combinedia dialog

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

// Dialog Data
	//{{AFX_DATA(Combinedia)
	enum { IDD = IDD_COMBINE };
	CListBox	m_list;
	CString m_list1;
	CString m_list2;
	//}}AFX_DATA


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

```
// Implementation
protected:

    // Generated message map functions
    //{{AFX_MSG(Combinedia)
    afx_msg void OnOpen();
    afx_msg void OnRun();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

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

#endif // !defined(AFX_OMBINEDIA_H__E03C8B81_26EB_11D2_AE2D_444553540001__INCLUDED_)
```

```
#if !defined(AFX_LPC_H__E24D24FC_263E_11D2_AE2D_444553540001__INCLUDED_)
#define AFX_LPC_H__E24D24FC_263E_11D2_AE2D_444553540001__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
// LPC.h : header file
//
#define FRAME_LENGTH    300
#define SHIFT_LENGTH    100
#define ADAPTIVE            0.9375
#define PI          3.141592654
#define FIRST_CALL          1
#define NEXT_CALL           2
#define P               12
#define N               19

//**************************************************************************

/////////////////////////////////////////////////////////////////////////////
// LPC view

class LPC : public CView
{
//protected:
public:
    LPC();      // protected constructor used by dynamic creation
    //DECLARE_DYNCREATE(LPC)

    bool open_all_file(CString filename,CString lpcfilename,CString textfilename);
    int    prepare_data(void);
    void find_preemphasis_and_window(void);
    void find_autocorrelation(void);
```

```
    void find_coefficient(void);
    void find_gain(void);
    void find_cepstrum(void);
    void find_weight(void);
    void write_parameter_file(void);
    void close_all_file(void);

    bool lpc_dialog(CString filename,CString lpcfilename,CString textfilename);
    bool Combine(CString inputfilename,CString outputfilename);

// Attributes
public:
    FILE   *input_wav_file;
    FILE   *parameter_file;
    FILE   *text_file;
    int          signal[FRAME_LENGTH+1];
    float  preemphasis[FRAME_LENGTH+1],windowed[FRAME_LENGTH+1];
    float  R[P+1],A[P][P],X[P][P];
    float  phi[P+1];
    float  alpha[P+1];
    float  gain;
    float  cepstrum[N],weight[N];

    FILE   *outputfile;
    FILE   *inputfile;

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(LPC)
    protected:
```

```
    virtual void OnDraw(CDC* pDC);    // overridden to draw this view
    //}}AFX_VIRTUAL

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

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

/////////////////////////////////////////////////////////////////////////////

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

#endif // !defined(AFX_LPC_H__E24D24FC_263E_11D2_AE2D_444553540001__INCLUDED_)
```

```
#if !defined(AFX_LIST_H__E24D24FD_263E_11D2_AE2D_444553540001__INCLUDED_)
#define AFX_LIST_H__E24D24FD_263E_11D2_AE2D_444553540001__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
// List.h : header file
//

/////////////////////////////////////////////////////////////////////////////
// List dialog

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

// Dialog Data
	//{{AFX_DATA(List)
	enum { IDD = IDD_LIST };
	CListBox	m_list;
	CString m_list1;
	CString m_list2;
	//}}AFX_DATA


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

```
// Implementation
protected:

    // Generated message map functions
    //{{AFX_MSG(List)
    afx_msg void OnOPEN();
    afx_msg void OnRUN();
    afx_msg void OnLIST();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

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

#endif // !defined(AFX_LIST_H__E24D24FD_263E_11D2_AE2D_444553540001__INCLUDED_)
```

```
#if !defined(AFX_HMM_H__C572A8A1_2792_11D2_AE2D_444553540001__INCLUDED_)
#define AFX_HMM_H__C572A8A1_2792_11D2_AE2D_444553540001__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
// HMM.h : header file
//
#define N1          6
#define K           64
#define ROUND       50
#define MIN_B       0.00001
/////////////////////////////////////////////////////////////////////////////
// HMM view

class HMM : public CView
{
//protected:
public:
    HMM();          // protected constructor used by dynamic creation
    //DECLARE_DYNCREATE(HMM)
bool    open_file(CString observation_file_name
                  ,CString model_file_name_hmm
                  ,CString model_text_file_name_hmm);
    int         find_number_of_observation(void);
    bool    allocate_memory(void);
    void    random_abvalue(void);
    void    copy_abprime_to_ab(int w);
    void    find_lf_bt_value(int w,int v);
    void    scaling_lf_bt(int w,int v);
    void    find_logP(int w,int v);
    void    find_A_AM_B_BM_prime(int w,int v);
    void    find_new_abvalue(int w);
```

```
    void   save_model_file(void);
    void   RunHmm(CString codebookindexfilename,CString modelfilename,CString modeltextfilename);

// Attributes
public:
FILE        *observation_file;
    FILE        *model_file;
    FILE        *model_text_file;
    int             v,maximum;
    char        *T1;
    char        *O;
    long double    *lf_test,*lfps,*lfs,*Bt,*sc,*c1,*plog;
    long double *cB_test,*Bts;
    long double    aprime[N1][N1];
    long double bprime[N1][K];
    long double a[N1][N1];
    long double b[N1][K];
    float       Pi[N1];
    long double    Plog[ROUND];
    int             w;
    long double x[N1][N1];
    long double Aprime[N1][N1], AMprime[N1];
    long double Bprime[N1][K], BMprime[N1], 11[N1];
    long double y[N1][K], z[N1];
    long double imb[N1];
    float       *Buf_aprime,*Buf_bprime,*Buf_Pi;

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(HMM)
```

```
    protected:
    virtual void OnDraw(CDC* pDC);    // overridden to draw this view
    //}}AFX_VIRTUAL

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

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

/////////////////////////////////////////////////////////////////////////////

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

#endif // !defined(AFX_HMM_H__C572A8A1_2792_11D2_AE2D_444553540001__INCLUDED_)
```

```
#if !defined(AFX_HMM_H__C572A8A1_2792_11D2_AE2D_444553540001__INCLUDED_)
#define AFX_HMM_H__C572A8A1_2792_11D2_AE2D_444553540001__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
// HMM.h : header file
//
#define N1          6
#define K           64
#define ROUND   50
#define MIN_B   0.00001
/////////////////////////////////////////////////////////////////////////////
// HMM view

class HMM : public CView
{
//protected:
public:
    HMM();          // protected constructor used by dynamic creation
    //DECLARE_DYNCREATE(HMM)
bool   open_file(CString observation_file_name
                        ,CString model_file_name_hmm
                        ,CString model_text_file_name_hmm);
    int         find_number_of_observation(void);
    bool    allocate_memory(void);
    void    random_abvalue(void);
    void    copy_abprime_to_ab(int w);
    void    find_lf_bt_value(int w,int v);
    void    scaling_lf_bt(int w,int v);
    void    find_logP(int w,int v);
    void    find_A_AM_B_BM_prime(int w,int v);
    void    find_new_abvalue(int w);
```

```
    void   save_model_file(void);
    void   RunHmm(CString codebookindexfilename,CString modelfilename,CString modeltextfilename);

// Attributes
public:
FILE        *observation_file;
    FILE        *model_file;
    FILE        *model_text_file;
    int             v,maximum;
    char        *T1;
    char        *O;
    long double    *lf_test,*lfps,*lfs,*Bt,*sc,*c1,*plog;
    long double *cB_test,*Bts;
    long double    aprime[N1][N1];
    long double bprime[N1][K];
    long double a[N1][N1];
    long double b[N1][K];
    float       Pi[N1];
    long double    Plog[ROUND];
    int             w;
    long double x[N1][N1];
    long double Aprime[N1][N1], AMprime[N1];
    long double Bprime[N1][K], BMprime[N1], l1[N1];
    long double y[N1][K], z[N1];
    long double imb[N1];
    float       *Buf_aprime,*Buf_bprime,*Buf_Pi;

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(HMM)
```

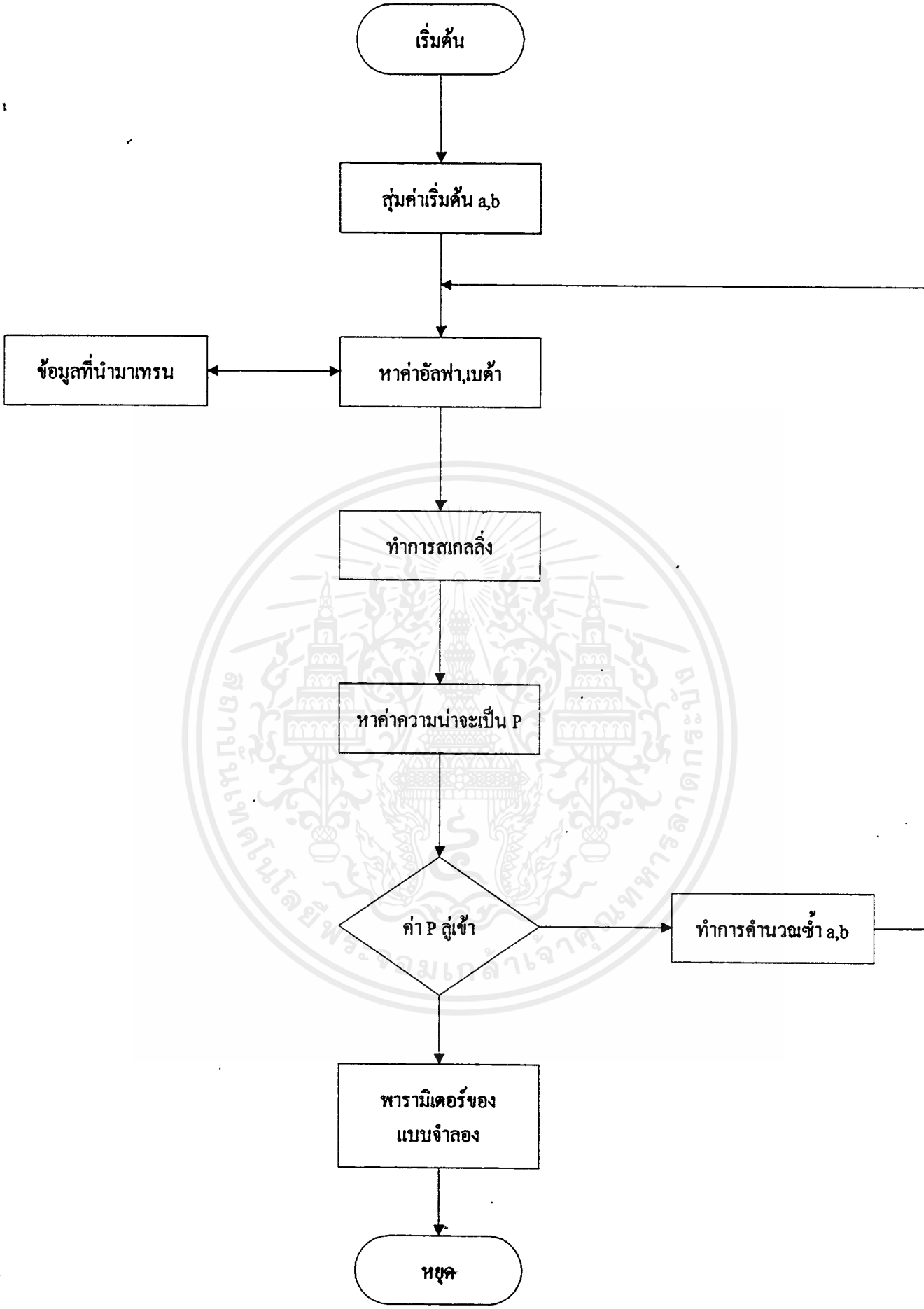
```
    protected:
    virtual void OnDraw(CDC* pDC);    // overridden to draw this view
    //}}AFX_VIRTUAL

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

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

/////////////////////////////////////////////////////////////////////////////

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

#endif // !defined(AFX_HMM_H__C572A8A1_2792_11D2_AE2D_444553540001__INCLUDED_)
```

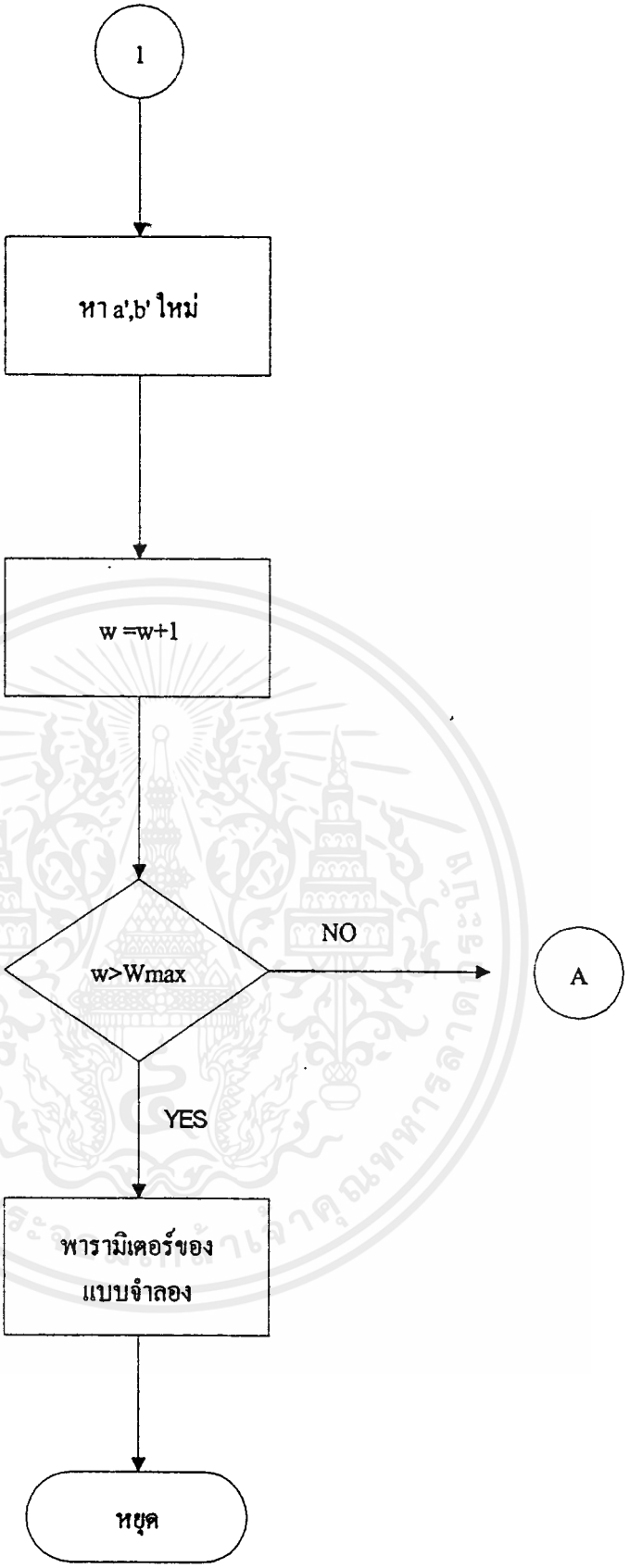
```
#if !defined(AFX_EDITWAV_H__CDAD94A3_27B3_11D2_AE2D_444553540001__INCLUDED_)
#define AFX_EDITWAV_H__CDAD94A3_27B3_11D2_AE2D_444553540001__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
// Editwav.h : header file
//

/////////////////////////////////////////////////////////////////////////////
// Editwav dialog

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

// Dialog Data
	//{{AFX_DATA(Editwav)
	enum { IDD = IDD_EDITWAV };
	CListBox	m_list;
	CString m_input1;
	CString m_input2;
	//}}AFX_DATA


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

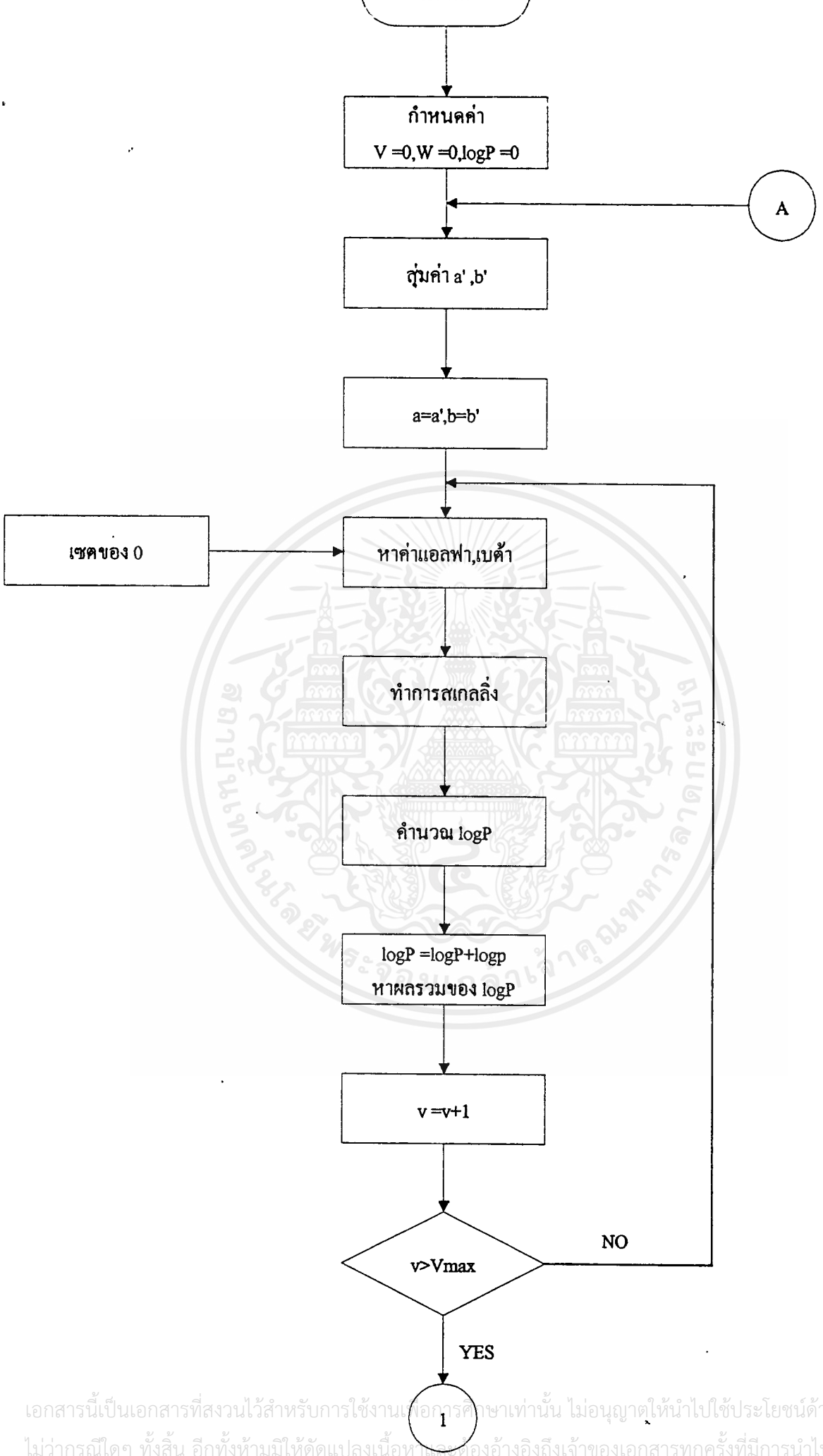
```
// Implementation
protected:

    // Generated message map functions
    //{{AFX_MSG(Editwav)
    afx_msg void OnShowList();
    afx_msg void OnRUN();
    afx_msg void OnInput();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

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

#endif // !defined(AFX_EDITWAV_H__CDAD94A3_27B3_11D2_AE2D_444553540001__INCLUDED_)
```

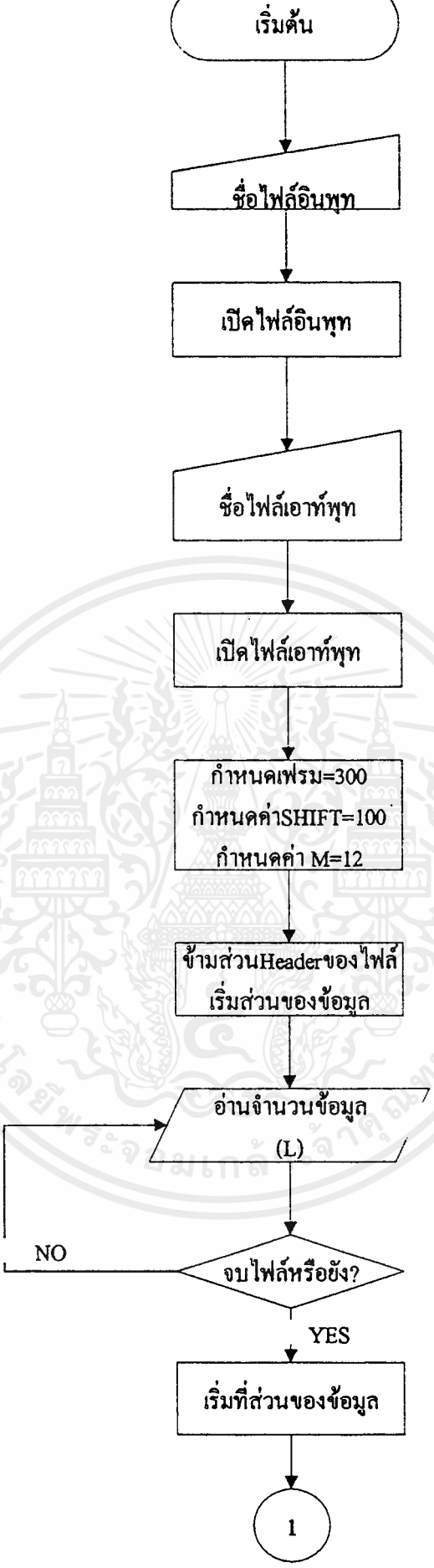
```
// art3Dlg.h : header file
//

#if !defined(AFX_ART3DLG_H__E24D24EF_263E_11D2_AE2D_444553540001__INCLUDED_)
#define AFX_ART3DLG_H__E24D24EF_263E_11D2_AE2D_444553540001__INCLUDED_

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

class CRecord;
class CArt3DlgAutoProxy;

/////////////////////////////////////////////////////////////////////////////
// CArt3Dlg dialog

class CArt3Dlg : public CDialog
{
    DECLARE_DYNAMIC(CArt3Dlg);
    friend class CArt3DlgAutoProxy;

// Construction
public:
    CArt3Dlg(CWnd* pParent = NULL); // standard constructor
    virtual ~CArt3Dlg();

// Dialog Data
    //{{AFX_DATA(CArt3Dlg)
    enum { IDD = IDD_ART3_DIALOG };
        // NOTE: the ClassWizard will add data members here
    //}}AFX_DATA

    // ClassWizard generated virtual function overrides
```

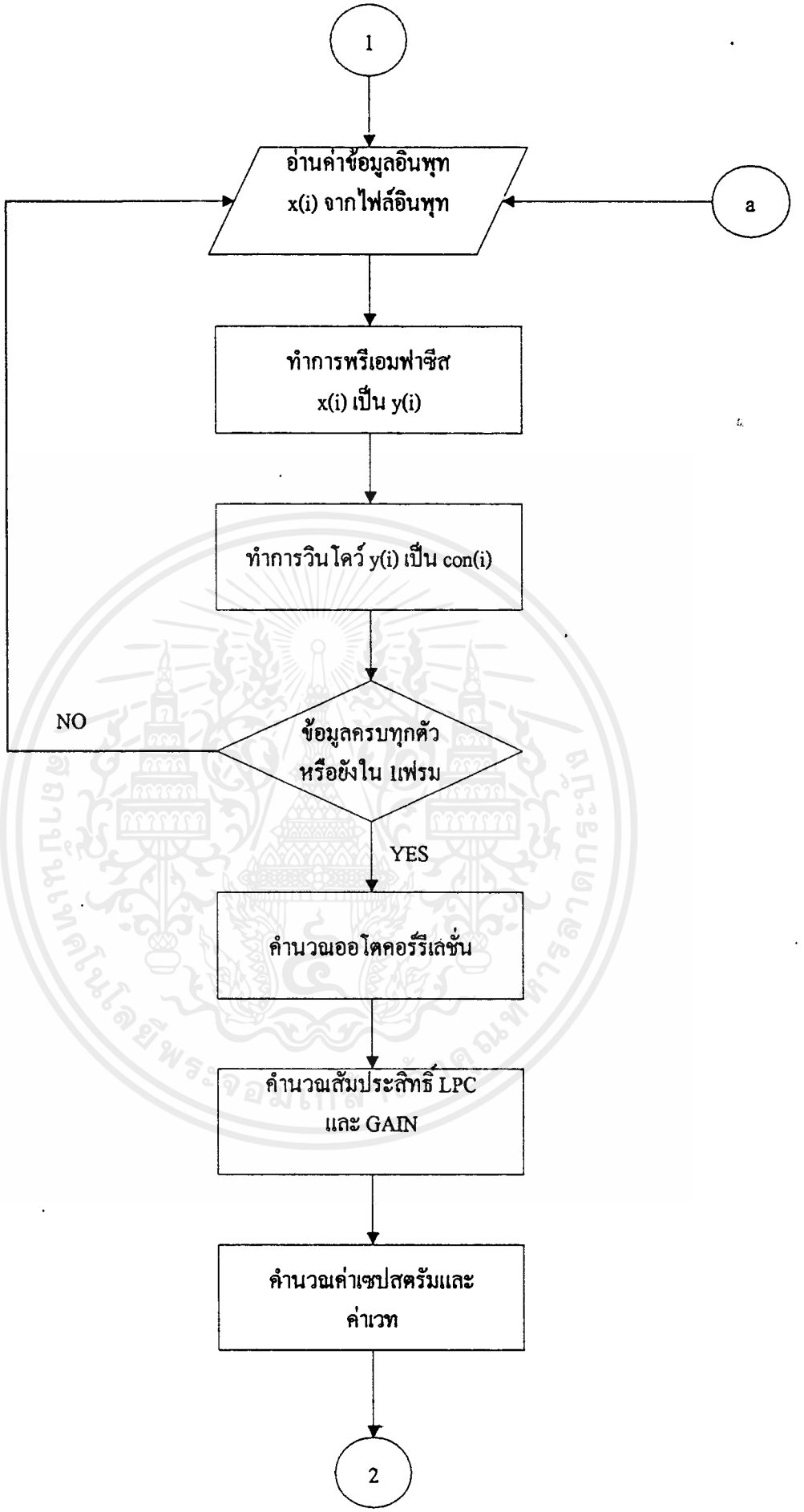
```
    afx_msg void OnCommandAuto();
    afx_msg void OnCommandAutoPort();
    afx_msg void OnEdit();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

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

#endif // !defined(AFX_ART3DLG_H__E24D24EF_263E_11D2_AE2D_444553540001__INCLUDED_)
```

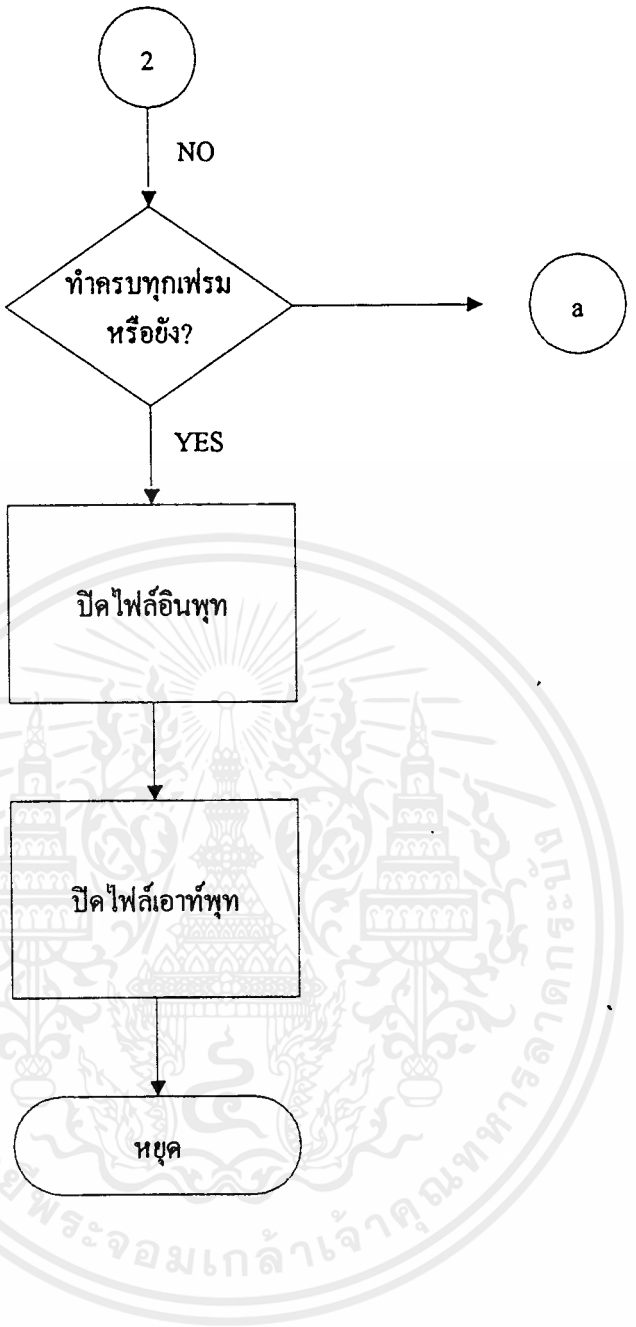


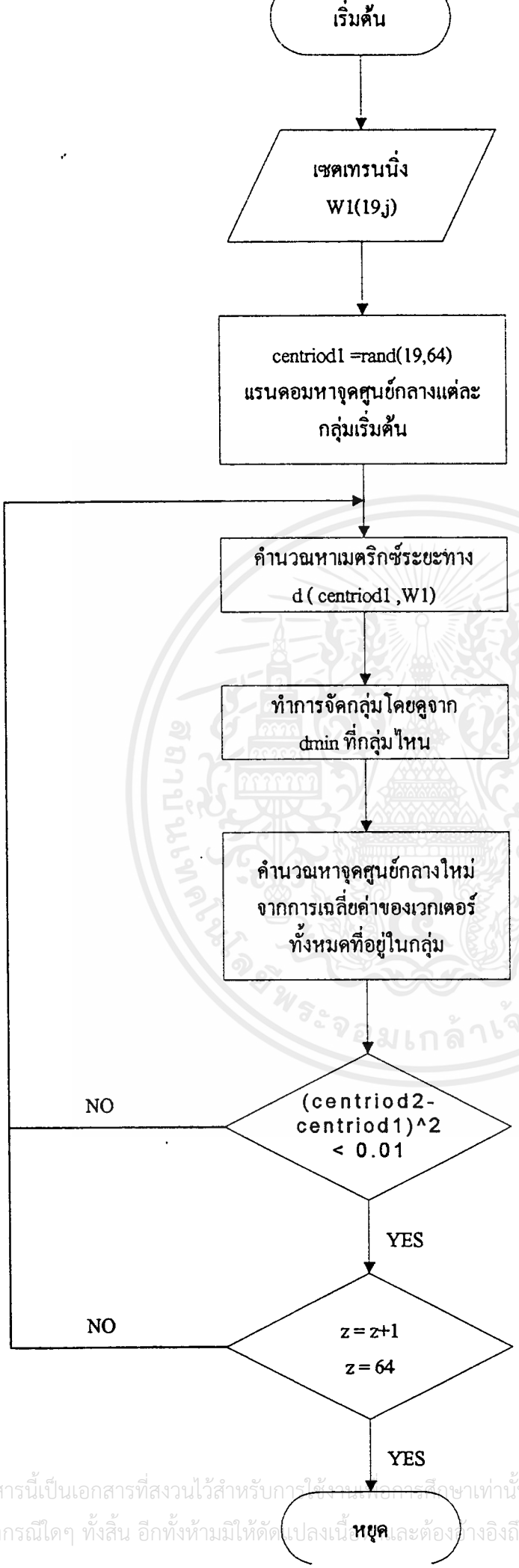


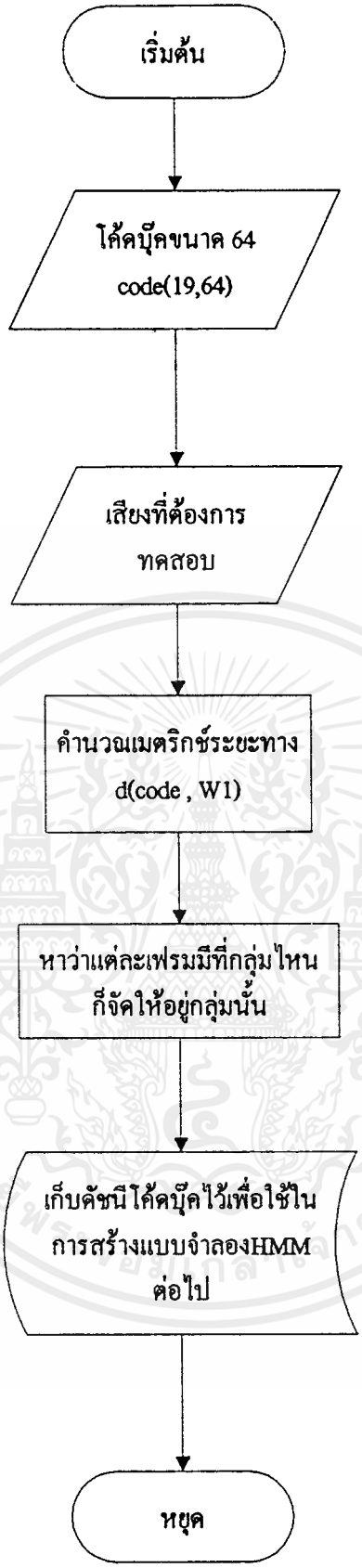












ZERO	4.88	-0.87	0.18	0.32	0.42	0.55	0.51	0.42	0.93	1.55	1.8	1.56	1.01	-0.01	-25.06	12.21	5.51	6.07	5.28
ONE	4.84	-1.06	-0.04	0.33	0.47	0.48	0.65	0.85	1.12	1.47	1.67	1.57	0.96	-0.14	-29.4	8.29	7.45	5.48	3.52
TWO	5.01	-0.93	-0.81	-0.89	0.24	1.41	1.47	1.14	1.56	1.97	1.57	0.73	0.13	-0.09	-30.08	-17.99	-14.88	14.95	19.83
THREE	4.96	-0.58	0.32	0.49	0.6	0.79	0.98	0.98	0.58	0.42	1	1.49	1.14	0.12	-18.23	13.6	10.8	10.45	9.42
FOUR	4.84	-1.41	-1.1	-1.14	-0.09	1.7	2.88	2.62	1.58	0.78	0.36	0.11	0.46	0.97	-38.37	-13.75	-10.94	12.39	21.67
FIVE	4.98	-0.55	-0.39	-0.82	-0.03	1.49	2.13	1.56	1.18	1.4	1.01	0.41	0.52	0.33	-17.14	-9.7	-19.39	4.03	21.1
SIX	5.16	-0.32	-1.8	-2.23	0.94	4.04	3.21	0.24	-0.13	1.16	0.75	0.47	1.66	0.64	-12.2	-70.4	-64.26	48.5	84.6
SEVEN	4.79	-1.3	-0.8	-0.6	0.15	1.46	2.29	2.07	1.5	1.14	0.6	0.5	0.8	0.36	-36.3	-6.4	-4.6	10.25	15.3
EIGHT	5	2.11	-0.82	-1.58	-1.68	0.32	3.26	2.75	-1.08	-0.26	2.59	-0.13	0.65	0.56	68.96	21.62	-54.31	-76.37	-24.71
NINE	4.8	-1.36	-0.8	-0.11	1.05	1.48	0.69	0.55	1.5	1.91	1.43	1.07	0.76	-0.2	-37.08	-6.3	8.17	19.76	9.25

แสดงค่า LPC ที่ใช้อ้างอิงในกราฟรูปที่ 4.5

ชายคนที่1	4.88	-0.87	0.18	0.32	0.42	0.55	0.51	0.42	0.93	1.55	1.8	1.56	1.01	-0.01	-25.06	12.21	5.51	6.07	5.28
ชายคนที่2	4.75	-1.4	-0.4	0.17	0.46	0.67	0.98	1.2	1.47	1.65	1.57	1.13	0.65	-0.13	35.23	5.02	7.61	5.37	4.26

แสดงค่า LPC ที่ใช้อ้างอิงในกราฟรูปที่ 4.8

ชายคนที่1	4.84	-1.06	-0.04	0.33	0.47	0.48	0.65	0.85	1.12	1.47	1.67	1.57	0.96	-0.14	-29.4	8.3	7.45	5.78	3.52
ชายคนที่2	4.82	-1.12	-0.14	0.37	0.53	0.48	0.36	0.63	1.24	1.72	1.86	1.58	0.9	-0.17	-30.18	6.5	9.4	5.9	2.95

แสดงค่า LPC ที่ใช้อ้างอิงในกราฟรูปที่ 4.9

ชายคนที่1	5.01	-0.9	-0.8	-0.89	0.24	1.4	1.47	1.14	1.55	1.96	1.56	0.72	0.12	-0.08	-30.08	-17.98	-14.87	14.95	19.8
ชายคนที่2	4.74	-1.54	-0.71	-0.11	0.75	1.21	1.19	1.21	1.69	1.95	1.31	0.56	0.31	0.28	-38.3	0.5	6.36	12.67	7.07

แสดงค่า LPC ที่ใช้อ้างอิงในกราฟรูปที่ 4.10

ครั้งที่1	4.88	-0.87	0.18	0.32	0.42	0.55	0.51	0.42	0.93	1.55	1.8	1.56	1.01	-0.01	-25.06	12.21	5.51	6.07	5.28
ครั้งที่2	4.77	-1.4	-0.56	0.08	0.5	0.53	0.69	1.31	1.92	1.91	1.65	1.03	0.41	-0.19	-38.28	3.02	8.29	6.82	2.13

แสดงค่า LPC ที่ใช้อ้างอิงในกราฟรูปที่ 4.11

ครั้งที่1	4.84	-1.06	-0.04	0.33	0.47	0.48	0.65	0.85	1.12	1.47	1.67	1.57	0.96	-0.14	-29.4	8.29	7.45	5.48	3.52
ครั้งที่2	4.89	-1.26	-1.15	-0.93	0.33	1.66	2.2	2.16	2.04	1.33	0.21	-0.16	0.65	0.86	-36.25	-19.58	-6.52	18.36	19.37

แสดงค่า LPC ที่ใช้อ้างอิงในกราฟรูปที่ 4.12

ครั้งที่1	5.01	-0.93	-0.81	-0.89	0.24	1.41	1.47	1.14	1.56	1.97	1.57	0.73	0.13	-0.09	-30.08	-17.99	-14.88	14.95	19.83
ครั้งที่2	4.88	-0.72	0.22	0.45	0.68	0.81	0.61	0.17	0.13	1.04	1.82	1.86	1.22	-0.02	-20.32	11.01	9.02	10.54	7.9

แสดงค่า LPC ที่ใช้อ้างอิงในกราฟรูปที่ 4.13

ZERO	184.5	77.7	64.7	66.7	45.7	46.7	30.6	31.7	22.5	21.6	22.9	17.6	23.14	17.6	20.2	17.1	17.25	17.52	11.21
ONE	1266	1211	1910	1851	2323	2249	2239	2209	2089	2103	1900	1957	1542	1751	1185	1475	979.9	1043	807
TWO	151.1	71.47	136.8	177.2	222.5	242.3	238.2	211.8	217.7	175.6	353.2	331.7	585.6	581.4	800.7	807.1	800.7	857.7	604.1
THREE	276.9	187.8	265.7	338.8	346.2	592	521.5	708.1	624.6	761.8	764.7	661.1	740.1	512.1	637.5	414.5	524.5	371.8	288.7
FOUR	125.7	19.37	19.15	21.62	22.04	30.61	34.11	37.33	137.6	144.7	149.4	139.5	240.2	243.3	167.1	165.3	170.9	201.1	95.84
FIVE	284.9	305.9	308.5	420.3	418.8	460.1	503.6	473.8	614.3	531.2	620.4	498.7	574.9	497.9	400.2	394.7	392.6	378.1	294.1
SIX	185.9	62.87	94.7	68.3	100.8	89.14	101.8	108.7	78.86	113.2	78.04	111.8	83.5	107.7	107.8	76.26	5.51	62	81.5
SEVEN	379.8	274.1	380.9	385.4	547.7	559.4	517.6	613.9	492.1	731.8	543.3	719.2	611	674.6	677.8	443.9	571.9	395.7	531.5
EIGHT	886	839.1	1141	1185	991.5	1138	843.9	1155	806.7	1134	1006	1140	1174	971.6	1148	765.1	974.1	663.9	847.8
NINE	237.4	111.3	506.5	524.7	1055	1118	1396	1630	1373	1696	1199	1349	1041	1128	1033	944.9	949.9	930.1	942.5

แสดงค่า Energy ที่ใช้อ้างอิงในกราฟรูปที่ 4.14

ชายคนที่ 1	184.5	77.7	64.7	66.7	45.7	46.7	30.6	31.7	22.5	21.6	22.9	17.6	23.14	17.6	20.2	17.1	17.25	17.52	11.21
ชายคนที่ 2	138.5	39.9	52.14	57.16	59.8	57.65	63.1	56.9	48.4	46.01	45.31	50.72	45.68	47.59	51.72	46.98	64.73	171.1	223.7

แสดงค่า Energy ที่ใช้อ้างอิงในกราฟรูปที่ 4.17

ชายคนที่ 1	1265.8	1211.2	1910.1	1850.9	2322.97	2248.8	2239.4	2209.4	2089.2	2103.1	1899.9	1956.5	1541.5	1750.9	1184.9	1474.5	979.9	1042.9
ชายคนที่ 2	123.18	19.87	48.07	59.8	104.9	172.5	202.2	315.6	323.3	446.4	442.4	556.5	678.3	635.3	764.8	682.6	893.6	753.7

แสดงค่า Energy ที่ใช้อ้างอิงในกราฟรูปที่ 4.18

ชายคนที่1	5.01	-0.9	-0.8	-0.89	0.24	1.4	1.47	1.14	1.55	1.96	1.56	0.72	0.12	-0.08	-30.08	-17.98	-14.87	14.95	19.8
ชายคนที่2	4.74	-1.54	-0.71	-0.11	0.75	1.21	1.19	1.21	1.69	1.95	1.31	0.56	0.31	0.28	-38.3	0.5	6.36	12.67	7.07

แสดงค่า Energy ที่ใช้อ้างอิงในกราฟรูปที่ 4.19

ครั้งที่ 1	184.5	77.7	64.7	66.7	45.7	46.7	30.6	31.7	22.5	21.6	22.9	17.6	23.14	17.6	20.2	17.1	17.25	17.52	11.21
ครั้งที่ 2	129.2	139.1	191.3	342	388.2	543.5	549.1	560.6	512.5	469.1	429.7	317.8	322.7	222.7	229.8	125.3	140.1	93.15	100.6

แสดงค่า Energy ที่ใช้อ้างอิงในกราฟรูปที่ 4.20

ครั้งที่ 1	1266	1211	1910	1851	2323	2249	2239	2209	2089	2103	1900	1957	1542	1751	1185	1475	979.9	1043	807
ครั้งที่ 2	1831	2277	2466	2377	2419	2221	2289	1924	2100	1597	1893	1471	1682	1436	1304	1304	1070	1098	759

แสดงค่า Energy ที่ใช้อ้างอิงในกราฟรูปที่ 4.21

ครั้งที่ 1	151.1	71.47	136.8	177.2	222.5	242.3	238.2	211.8	217.7	175.6	353.2	331.7	585.6	581.4	800.7	807.1	800.7	857.7	604.1
ครั้งที่ 2	201.7	199.8	220.7	443.5	444.1	778.1	790.4	1013	1039	959.2	1126	783.1	1054	720.9	922.6	773.5	706.9	812.4	521

แสดงค่า Energy ที่ใช้อ้างอิงในกราฟรูปที่ 4.22

ชายคนที่ 1	20762.7	19148.6	6038.4	-5168.5	-13032.3	-16323.5	-14006.1	-5855.2	6570.3	18240.2	26210.6	29323.1
ชายคนที่ 2	-728.2	7414.7	6572.3	6218.1	6120.7	5910.3	5719.1	6750.7	8598.2	9165.5	8295.3	8088.2

แสดงค่า Autocorrelation ที่ใช้อ้างอิงในกราฟรูปที่ 4.24

ชายคนที่ 1	1746.47	5178.76	3188.7	5688.05	7627.26	6380.78	5225.8	7754.6	11375.1	11058	6875.58	4072.1
ชายคนที่ 2	-291.12	6908.45	5678.5	6044.2	7847.8	8845.9	8466.5	7113.1	6554.8	6126.5	7102.7	8410.1

แสดงค่า Autocorrelation ที่ใช้อ้างอิงในกราฟรูปที่ 4.25

ชายคนที่ 1	5.13	-0.01	0.2	0.1	0.03	-0.02	-0.07	-0.1	-0.07	0.05	0.17	0.25	0.26	0.07	-0.07	4.62	3.49	2.2	0.05
ชายคนที่ 2	4.95	-0.36	-0.01	0.04	0.05	0.03	0	0.01	0.08	0.19	0.24	0.2	0.12	-0.03	-3.7	1	1.22	1.16	0.75

แสดงค่า Ceptrum ที่ใช้อ้างอิงในกราฟรูปที่ 4.27

ชายคนที่ 1	4.98	-0.33	-0.12	-0.11	0	0.09	0.07	0.06	0.16	0.28	0.26	0.1	0	-0.05	-3.43	-1.42	-2.1	1.2	4.1
ชายคนที่ 2	4.77	-0.56	-0.18	-0.1	0	0.16	0.26	0.25	0.16	0.08	0.04	0.07	0.11	0.03	-4.68	-0.44	-0.72	1.44	5.46

แสดงค่า Ceptrum ที่ใช้อ้างอิงในกราฟรูปที่ 4.28

ชายคนที่ 1	-0.014	0.2	0.11	0.01	-0.04	-0.08	-0.11	-0.05	0.08	0.19	0.24	0.22
ชายคนที่ 2	-0.36	-0.07	0.03	0.06	0.055	0.02	0.01	0.08	0.22	0.32	0.3	0.2

แสดงค่า Coefficient ที่ใช้อ้างอิงในกราฟรูปที่ 4.30

ชายคนที่ 1	-0.33	-0.18	-0.16	-0.05	0.06	0.09	0.09	0.21	0.37	0.4	0.27	0.12
ชายคนที่ 2	-0.56	-0.33	-0.23	-0.11	0.11	0.33	0.45	0.43	0.34	0.22	0.16	0.12

แสดงค่า Coefficient ที่ใช้อ้างอิงในกราฟรูปที่ 4.31

ชายคนที่ 1	170.6	168.9	44.52	25.74	21.76	16.65	13.57	12.77	11.47	11.74	10.66	10.02
ชายคนที่ 2	142	133.3	37.84	26.77	28.62	31.6	35.57	34.18	32.69	34.74	33.91	38

แสดงค่า Gainที่ใช้อ้างอิงในกราฟรูปที่ 4.33

ชายคนที่ 1	145.5	151.5	132.5	237.8	223	239.1	248.7	228	196.5	156.2	87.5	67.9
ชายคนที่ 2	118.5	116.5	25.91	18.94	25.83	38.34	48.14	56.52	58.15	63.94	66.47	71.3

แสดงค่า Gainที่ใช้อ้างอิงในกราฟรูปที่ 4.34

ISSUE 5 M[illegible]1997

Features

- Complete DTMF Receiver
- Low power consumption
- Internal gain setting amplifier
- Adjustable guard time
- Central office quality
- Power-down mode
- Inhibit mode
- Backward compatible with MT8870C/MT8870C-1

Applications

- Receiver system for British Telecom (BT) or CEPT Spec (MT8870D-1)
- Paging systems
- Repeater systems/mobile radio
- Credit card systems
- Remote control
- Personal computers
- Telephone answering machine

Ordering Information

MT8870DE/DE-1	18 Pin Plastic DIP
MT8870DS/DS-1	18 Pin SOIC
MT8870DN/DN-1	20 Pin SSOP

-40 °C to +85 °C

Description

The MT8870D/MT8870D-1 is a complete DTMF receiver integrating both the bandsplit filter and digital decoder functions. The filter section uses switched capacitor techniques for high and low group filters; the decoder uses digital counting techniques to detect and decode all 16 DTMF tone-pairs into a 4-bit code. External component count is minimized by on chip provision of a differential input amplifier, clock oscillator and latched three-state bus interface.

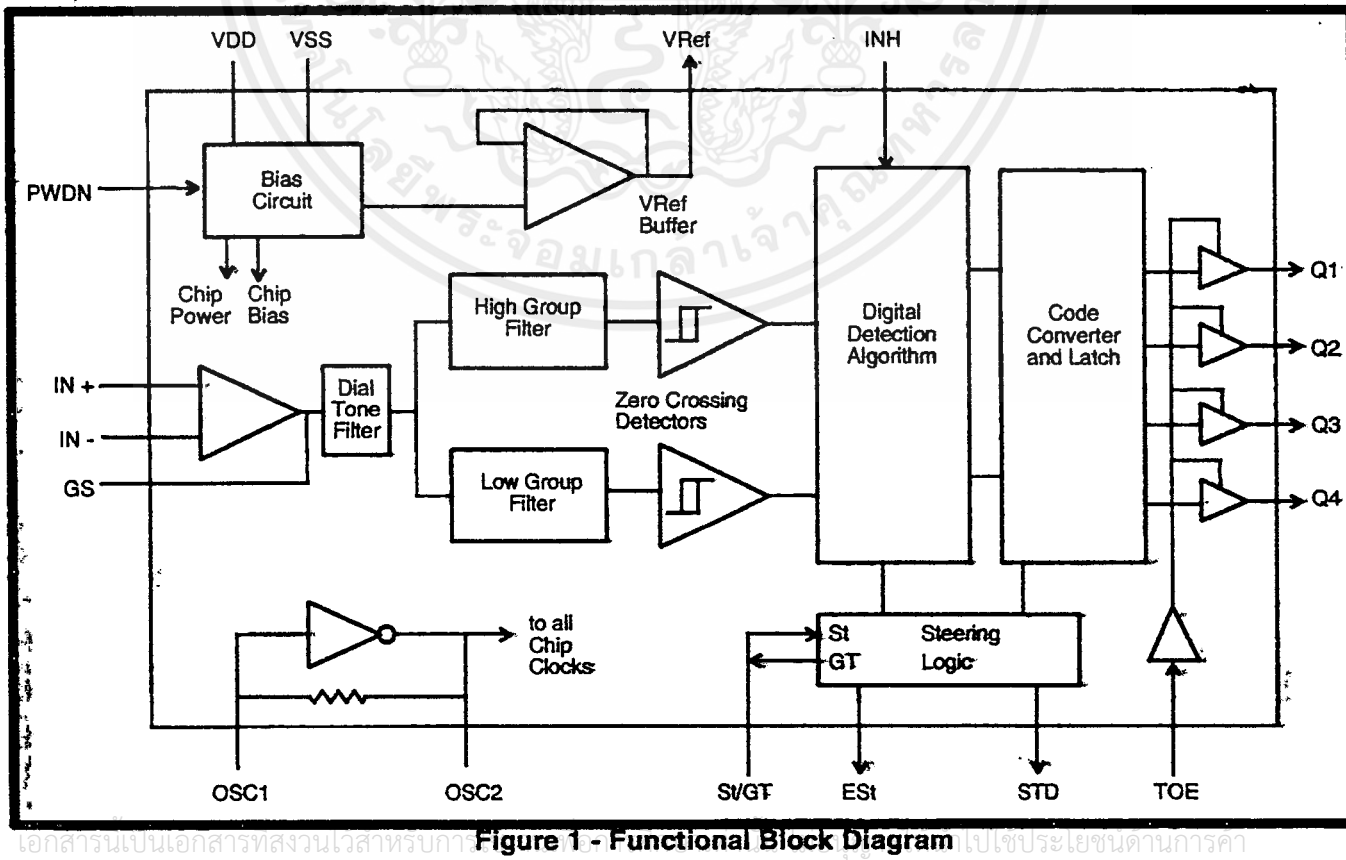


Figure 1 - Functional Block Diagram

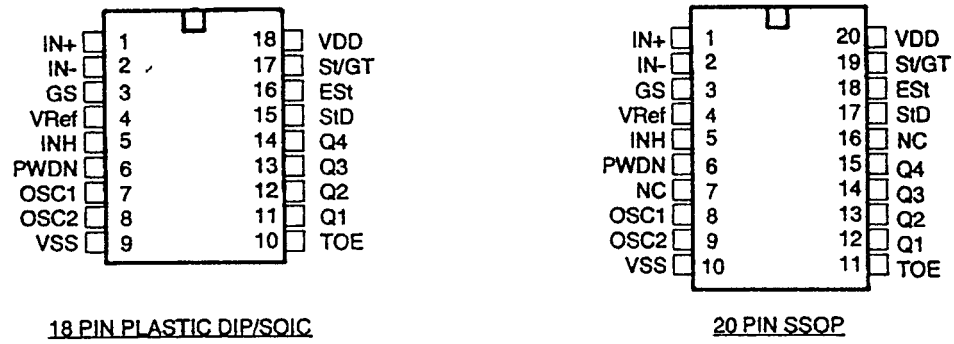


Figure 2 - Pin Connections

Pin Description

Pin #		Name	Description
18	20		
1	1	IN+	**Non-Inverting Op-Amp (Input).**
2	2	IN-	**Inverting Op-Amp (Input).**
3	3	GS	**Gain Select.** Gives access to output of front end differential amplifier for connection of feedback resistor.
4	4	V_{Ref}	**Reference Voltage (Output).** Nominally $V_{DD}/2$ is used to bias inputs at mid-rail (see Fig. 6 and Fig. 10).
5	5	INH	**Inhibit (Input).** Logic high inhibits the detection of tones representing characters A, B, C and D. This pin input is internally pulled down.
6	6	PWDN	**Power Down (Input).** Active high. Powers down the device and inhibits the oscillator. This pin input is internally pulled down.
7	8	OSC1	**Clock (Input).**
8	9	OSC2	**Clock (Output).** A 3.579545 MHz crystal connected between pins OSC1 and OSC2 completes the internal oscillator circuit.
9	10	V_{SS}	**Ground (Input).** 0V typical.
10	11	TOE	**Three State Output Enable (Input).** Logic high enables the outputs Q1-Q4. This pin is pulled up internally.
11-14	12-15	Q1-Q4	**Three State Data (Output).** When enabled by TOE, provide the code corresponding to the last valid tone-pair received (see Table 1). When TOE is logic low, the data outputs are high impedance.
15	17	StD	**Delayed Steering (Output).** Presents a logic high when a received tone-pair has been registered and the output latch updated; returns to logic low when the voltage on St/GT falls below V_{TSt}.
16	18	ESt	**Early Steering (Output).** Presents a logic high once the digital algorithm has detected a valid tone pair (signal condition). Any momentary loss of signal condition will cause ESt to return to a logic low.
17	19	St/GT	**Steering Input/Guard time (Output) Bidirectional.** A voltage greater than V_{TSt} detected at St causes the device to register the detected tone pair and update the output latch. A voltage less than V_{TSt} frees the device to accept a new tone pair. The GT output acts to reset the external steering time-constant; its state is a function of ESt and the voltage on St.
18	20	V_{DD}	**Positive power supply (Input).** +5V typical.
	7, 16	NC	No Connection.

Functional Description

The MT8870D/MT8870D-1 monolithic DTMF receiver offers small size, low power consumption and high performance. Its architecture consists of a bandsplit filter section, which separates the high and low group tones, followed by a digital counting section which verifies the frequency and duration of the received tones before passing the corresponding code to the output bus.

Filter Section

Separation of the low-group and high group tones is achieved by applying the DTMF signal to the inputs of two sixth-order switched capacitor bandpass filters, the bandwidths of which correspond to the low and high group frequencies. The filter section also incorporates notches at 350 and 440 Hz for exceptional dial tone rejection (see Figure 3). Each filter output is followed by a single order switched capacitor filter section which smooths the signals prior to limiting. Limiting is performed by high-gain comparators which are provided with hysteresis to prevent detection of unwanted low-level signals. The outputs of the comparators provide full rail logic swings at the frequencies of the incoming DTMF signals.

Decoder Section

Following the filter section is a decoder employing digital counting techniques to determine the frequencies of the incoming tones and to verify that they correspond to standard DTMF frequencies. A complex averaging algorithm protects against tone simulation by extraneous signals such as voice while

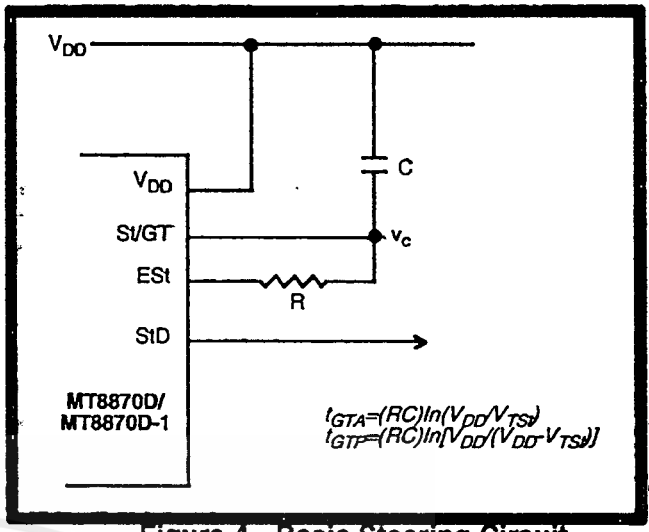


Figure 4 - Basic Steering Circuit

providing tolerance to small frequency deviations and variations. This averaging algorithm has been developed to ensure an optimum combination of immunity to talk-off and tolerance to the presence of interfering frequencies (third tones) and noise. When the detector recognizes the presence of two valid tones (this is referred to as the "signal condition" in some industry specifications) the "Early Steering" (ESt) output will go to an active state. Any subsequent loss of signal condition will cause ESt to assume an inactive state (see "Steering Circuit").

Steering Circuit

Before registration of a decoded tone pair, the receiver checks for a valid signal duration (referred to as character recognition condition). This check is performed by an external RC time constant driven by ESt. A logic high on ESt causes v_c (see Figure 4) to rise as the capacitor discharges. Provided signal

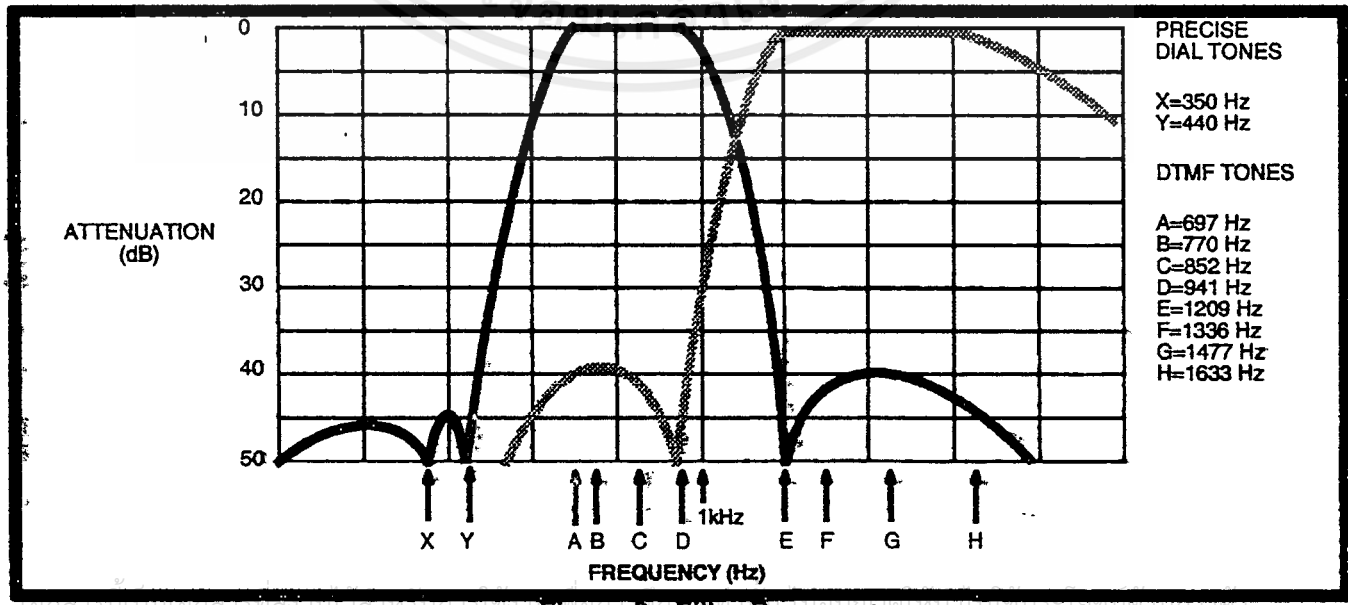


Figure 3 - Filter Response

condition is maintained (ESt remains high) for the validation period (t_{GTP}), v_c reaches the threshold (V_{TSt}) of the steering logic to register the tone pair, latching its corresponding 4-bit code (see Table 1) into the output latch. At this point the GT output is activated and drives v_c to V_{DD}. GT continues to drive high as long as ESt remains high. Finally, after a short delay to allow the output latch to settle, the delayed steering output flag (StD) goes high, signalling that a received tone pair has been registered. The contents of the output latch are made available on the 4-bit output bus by raising the three state control input (TOE) to a logic high. The steering circuit works in reverse to validate the interdigit pause between signals. Thus, as well as rejecting signals too short to be considered valid, the receiver will tolerate signal interruptions (dropout) too short to be considered a valid pause. This facility, together with the capability of selecting the steering time constants externally, allows the designer to tailor performance to meet a wide variety of system requirements.

Digit	TOE	INH	ESt	Q_4	Q_3	Q_2	Q_1
ANY	L	X	H	Z	Z	Z	Z
1	H	X	H	0	0	0	1
2	H	X	H	0	0	1	0
3	H	X	H	0	0	1	1
4	H	X	H	0	1	0	0
5	H	X	H	0	1	0	1
6	H	X	H	0	1	1	0
7	H	X	H	0	1	1	1
8	H	X	H	1	0	0	0
9	H	X	H	1	0	0	1
0	H	X	H	1	0	1	0
*	H	X	H	1	0	1	1
#	H	X	H	1	1	0	0
A	H	L	H	1	1	0	1
B	H	L	H	1	1	1	0
C	H	L	H	1	1	1	1
D	H	L	H	0	0	0	0
A	H	H	L	undetected, the output code will remain the same as the previous detected code			
B	H	H	L				
C	H	H	L				
D	H	H	L				

Table 1. Functional Decode Table

L=LOGIC LOW, H=LOGIC HIGH, Z=HIGH IMPEDANCE
X = DON'T CARE

Guard Time Adjustment

In many situations not requiring selection of tone duration and interdigital pause, the simple steering circuit shown in Figure 4 is applicable. Component values are chosen according to the formula:

$$t_{REC}=t_{DP}+t_{GTP}$$
$$t_{ID}=t_{DA}+t_{GTA}$$

The value of t_{DP} is a device parameter (see Figure 11) and t_{REC} is the minimum signal duration to be recognized by the receiver. A value for C of 0.1 µF is recommended for most applications, leaving R to be selected by the designer.

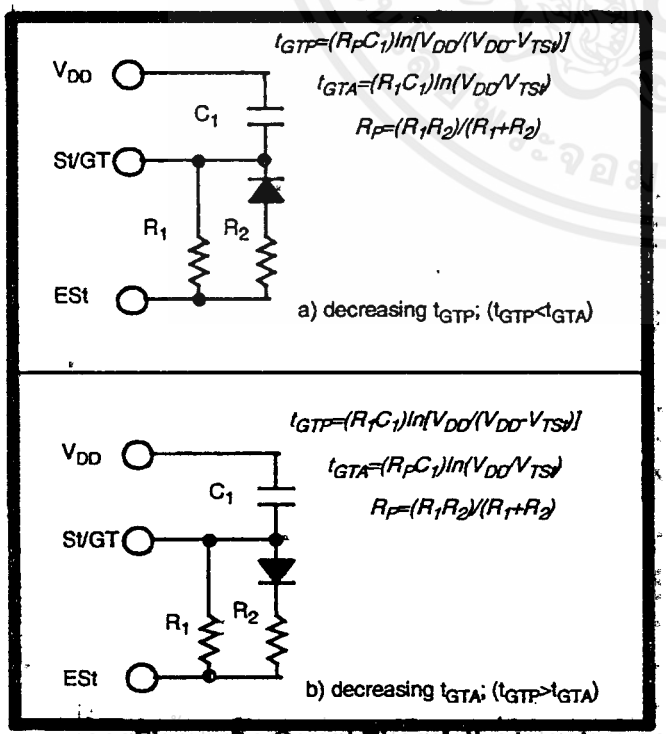


Figure 5 - Guard Time Adjustment

Different steering arrangements may be used to select independently the guard times for tone present (t_{GTP}) and tone absent (t_{GTA}). This may be necessary to meet system specifications which place both accept and reject limits on both tone duration and interdigital pause. Guard time adjustment also allows the designer to tailor system parameters such as talk off and noise immunity. Increasing t_{REC} improves talk-off performance since it reduces the probability that tones simulated by speech will maintain signal condition long enough to be registered. Alternatively, a relatively short t_{REC} with a long t_{DO} would be appropriate for extremely noisy environments where fast acquisition time and immunity to tone drop-outs are required. Design information for guard time adjustment is shown in Figure 5.

Power-down and Inhibit Mode

A logic high applied to pin 6 (PWDN) will power down the device to minimize the power consumption in a standby mode. It stops the oscillator and the functions of the filters.

Inhibit mode is enabled by a logic high input to the pin 5 (INH). It inhibits the detection of tones representing characters A, B, C, and D. The output code will remain the same as the previous detected code (see Table 1).

Differential Input Configuration

The input arrangement of the MT8870D/MT8870D-1 provides a differential-input operational amplifier as well as a bias source (V_{Ref}) which is used to bias the inputs at mid-rail. Provision is made for connection of a feedback resistor to the op-amp output (GS) for adjustment of gain. In a single-ended configuration, the input pins are connected as shown in Figure 10 with the op-amp connected for unity gain and V_{Ref} biasing the input at $^1/_2V_{DD}$. Figure 6 shows the differential configuration, which permits the adjustment of gain with the feedback resistor R_5.

Crystal Oscillator

The internal clock circuit is completed with the addition of an external 3.579545 MHz crystal and is normally connected as shown in Figure 10 (Single-Ended Input Configuration). However, it is possible to configure several MT8870D/MT8870D-1 devices employing only a single oscillator crystal. The oscillator output of the first device in the chain is coupled through a 30 pF capacitor to the oscillator input (OSC1) of the next device. Subsequent devices are connected in a similar fashion. Refer to Figure 7 for details. The problems associated with unbalanced loading are not a concern with the arrangement shown, i.e., precision balancing capacitors are not required.

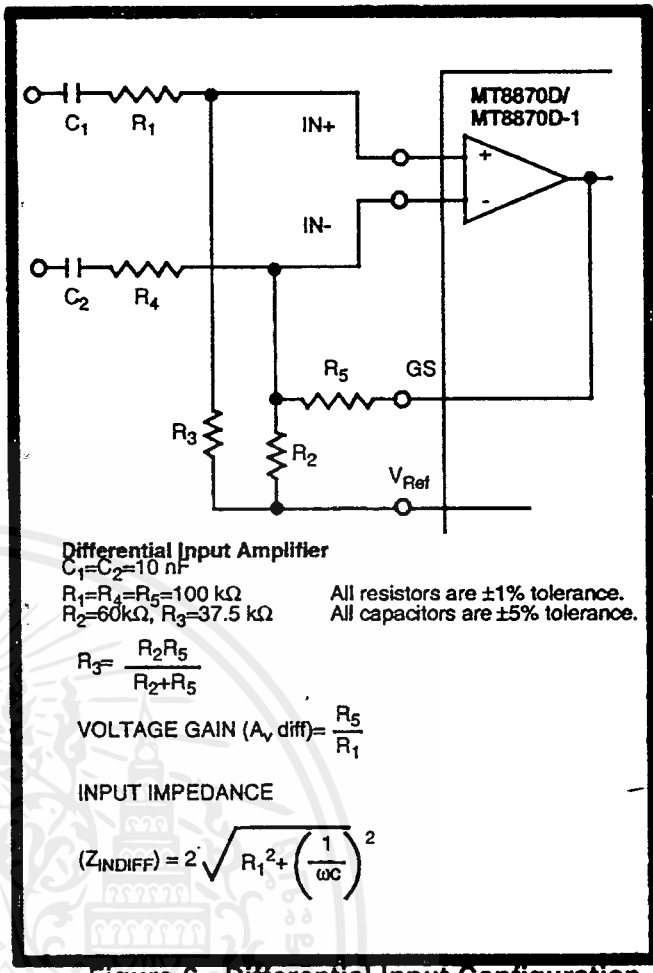


Figure 6 - Differential Input Configuration

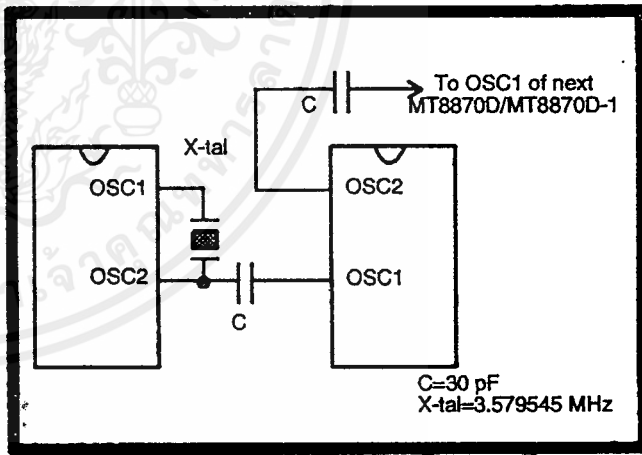


Figure 7 - Oscillator Connection

Parameter	Unit	Resonator
R1	Ohms	10.752
L1	mH	.432
C1	pF	4.984
C0	pF	37.915
Qm	-	896.37
Δf	%	±0.2%

Table 2. Recommended Resonator Specifications

Note: Qm=quality factor of RLC model, i.e., $1/2\Pi fR1C1$.

Applications

RECEIVER SYSTEM FOR BRITISH TELECOM SPEC POR 1151

The circuit shown in Fig. 9 illustrates the use of MT8870D-1 device in a typical receiver system. BT Spec defines the input signals less than -34 dBm as the non-operate level. This condition can be attained by choosing a suitable values of R_1 and R_2 to provide 3 dB attenuation, such that -34 dBm input signal will correspond to -37 dBm at the gain setting pin GS of MT8870D-1. As shown in the diagram, the component values of R_3 and C_2 are the guard time requirements when the total component tolerance is 6%. For better performance, it is recommended to use the non-symmetric guard time circuit in Fig. 8.

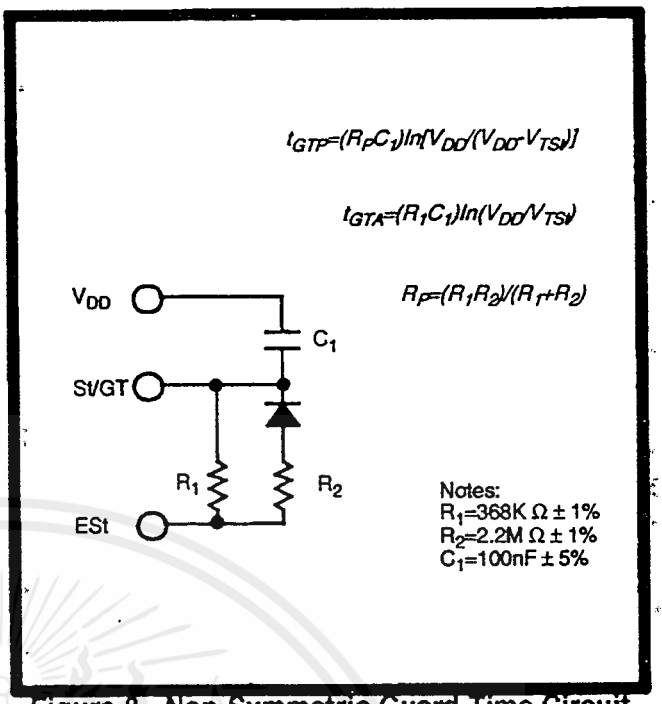


Figure 8 - Non-Symmetric Guard Time Circuit

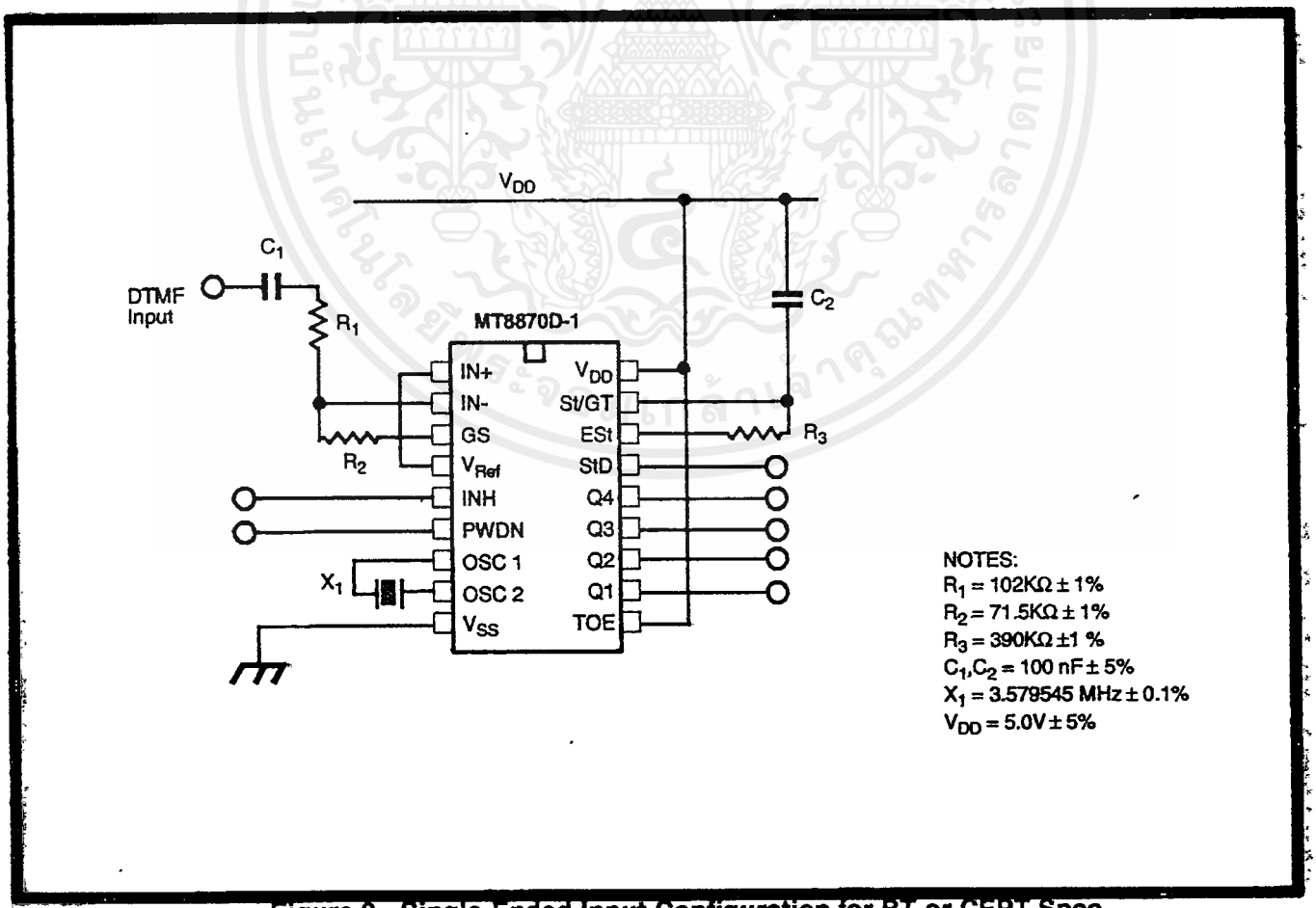


Figure 9 - Single-Ended Input Configuration for BT or CEPT Spec

Absolute Maximum Ratings†

	Parameter	Symbol	Min	Max	Units
1	DC Power Supply Voltage	V_{DD}		7	V
2	Voltage on any pin	V_I	V_{SS}-0.3	V_{DD}+0.3	V
3	Current at any pin (other than supply)	I_I		10	mA
4	Storage temperature	T_{STG}	-65	+150	°C
5	Package power dissipation	P_D		500	mW

† Exceeding these values may cause permanent damage. Functional operation under these conditions is not implied.
Derate above 75 °C at 16 mW / °C. All leads soldered to board.

Recommended Operating Conditions - Voltages are with respect to ground (V_{SS}) unless otherwise stated.

	Parameter	Sym	Min	Typ‡	Max	Units	Test Conditions
1	DC Power Supply Voltage	V_{DD}	4.75	5.0	5.25	V	
2	Operating Temperature	T_O	-40		+85	°C	
3	Crystal/Clock Frequency	fc		3.579545		MHz	
4	Crystal/Clock Freq.Tolerance	Δfc		±0.1		%	

‡ Typical figures are at 25°C and are for design aid only: not guaranteed and not subject to production testing.

DC Electrical Characteristics - V_{DD}=5.0V± 5%, V_{SS}=0V, -40°C ≤ T_O ≤ +85°C, unless otherwise stated.

		Characteristics	Sym	Min	Typ‡	Max	Units	Test Conditions
1	SUPPLY	Standby supply current	I_{DDQ}		10	25	µA	PWDN=V_{DD}
2		Operating supply current	I_{DD}		3.0	9.0	mA	
3		Power consumption	P_O		15		mW	fc=3.579545 MHz
4	INPUTS	High level input	V_{IH}	3.5			V	V_{DD}=5.0V
5		Low level input voltage	V_{IL}			1.5	V	V_{DD}=5.0V
6		Input leakage current	I_{IH}/I_{IL}		0.1		µA	V_{IN}=V_{SS} or V_{DD}
7		Pull up (source) current	I_{SO}		7.5	20	µA	TOE (pin 10)=0, V_{DD}=5.0V
8		Pull down (sink) current	I_{SI}		15	45	µA	INH=5.0V, PWDN=5.0V, V_{DD}=5.0V
9		Input impedance (IN+, IN-)	R_{IN}		10		MΩ	@ 1 kHz
10		Steering threshold voltage	V_{TSt}	2.2	2.4	2.5	V	V_{DD} = 5.0V
11	OUTPUTS	Low level output voltage	V_{OL}			V_{SS}+0.03	V	No load
12		High level output voltage	V_{OH}	V_{DD}-0.03			V	No load
13		Output low (sink) current	I_{OL}	1.0	2.5		mA	V_{OUT}=0.4 V
14		Output high (source) current	I_{OH}	0.4	0.8		mA	V_{OUT}=4.6 V
15		V_{Ref} output voltage	V_{Ref}	2.3	2.5	2.7	V	No load, V_{DD} = 5.0V
16		V_{Ref} output resistance	R_{OR}		1		kΩ	

‡ Typical figures are at 25°C and are for design aid only: not guaranteed and not subject to production testing.

Operating Characteristics - V_{DD}=5.0V±5%, V_{SS}=0V, -40°C ≤ T_O ≤ +85°C ,unless otherwise stated.

Gain Setting Amplifier

	Characteristics	Sym	Min	Typ‡	Max	Units	Test Conditions
1	Input leakage current	I_{IN}			100	nA	$V_{SS} \le V_{IN} \le V_{DD}$
2	Input resistance	R_{IN}	10			MΩ	
3	Input offset voltage	V_{OS}			25	mV	
4	Power supply rejection	PSRR	50			dB	1 kHz
5	Common mode rejection	CMRR	40			dB	0.75 V ≤ V_{IN} ≤ 4.25 V biased at V_{Ref} =2.5 V
6	DC open loop voltage gain	A_{VOL}	32			dB	
7	Unity gain bandwidth	f_C	0.30			MHz	
8	Output voltage swing	V_O	4.0			V_{pp}	Load ≥ 100 kΩ to V_{SS} @ GS
9	Maximum capacitive load (GS)	C_L			100	pF	
10	Resistive load (GS)	R_L			50	kΩ	
11	Common mode range	V_{CM}	2.5			V_{pp}	No Load

MT8870D AC Electrical Characteristics - V_{DD}=5.0V ±5%, V_{SS}=0V, -40°C ≤ T_O ≤ +85°C , using Test Circuit shown in Figure 10.

	Characteristics	Sym	Min	Typ‡	Max	Units	Notes*
1	Valid input signal levels (each tone of composite signal)		-29		+1	dBm	1,2,3,5,6,9
			27.5		869	mV_{RMS}	1,2,3,5,6,9
2	Negative twist accept				8	dB	2,3,6,9,12
3	Positive twist accept				8	dB	2,3,6,9,12
4	Frequency deviation accept		±1.5% ± 2 Hz				2,3,5,9
5	Frequency deviation reject		±3.5%				2,3,5,9
6	Third tone tolerance			-16		dB	2,3,4,5,9,10
7	Noise tolerance			-12		dB	2,3,4,5,7,9,10
8	Dial tone tolerance			+22		dB	2,3,4,5,8,9,11

‡ Typical figures are at 25 °C and are for design aid only: not guaranteed and not subject to production testing.

*NOTES:
1. dBm= decibels above or below a reference power of 1 mW into a 600 ohm load.
2. Digit sequence consists of all DTMF tones.
3. Tone duration= 40 ms, tone pause= 40 ms.
4. Signal condition consists of nominal DTMF frequencies.
5. Both tones in composite signal have an equal amplitude.
6. Tone pair is deviated by ±1.5 %± 2 Hz.
7. Bandwidth limited (3 kHz) Gaussian noise.
8. The precise dial tone frequencies are (350 Hz and 440 Hz) ± 2 %.
9. For an error rate of better than 1 in 10,000.
10. Referenced to lowest level frequency component in DTMF signal.
11. Referenced to the minimum valid accept level.
12. Guaranteed by design and characterization.

MT8870D-1 AC Electrical Characteristics - V_{DD}=5.0V±5%, V_{SS}=0V, -40°C ≤ T_O ≤ +85°C , using Test Circuit shown in Figure 10.

	Characteristics	Sym	Min	Typ‡	Max	Units	Notes*
1	Valid input signal levels (each tone of composite signal)		-31		+1	dBm	Tested at V_{DD}=5.0V 1,2,3,5,6,9
			21.8		869	mV_{RMS}	
2	Input Signal Level Reject		-37			dBm	Tested at V_{DD}=5.0V 1,2,3,5,6,9
			10.9			mV_{RMS}	
3	Negative twist accept				8	dB	2,3,6,9,13
4	Positive twist accept				8	dB	2,3,6,9,13
5	Frequency deviation accept		±1.5%± 2 Hz				2,3,5,9
6	Frequency deviation reject		±3.5%				2,3,5,9
7	Third zone tolerance			-18.5		dB	2,3,4,5,9,12
8	Noise tolerance			-12		dB	2,3,4,5,7,9,10
9	Dial tone tolerance			+22		dB	2,3,4,5,8,9,11

‡ Typical figures are at 25 °C and are for design aid only: not guaranteed and not subject to production testing.

*NOTES
1. dBm= decibels above or below a reference power of 1 mW into a 600 ohm load.
2. Digit sequence consists of all DTMF tones.
3. Tone duration= 40 ms, tone pause= 40 ms.
4. Signal condition consists of nominal DTMF frequencies.
5. Both tones in composite signal have an equal amplitude.
6. Tone pair is deviated by ±1.5 %± 2 Hz.
7. Bandwidth limited (3 kHz) Gaussian noise.
8. The precise dial tone frequencies are (350 Hz and 440 Hz) ± 2 %.
9. For an error rate of better than 1 in 10,000.
10. Referenced to lowest level frequency component in DTMF signal.
11. Referenced to the minimum valid accept level.
12. Referenced to Fig. 10 input DTMF tone level at -25dBm (-28dBm at GS Pin) interference frequency range between 480-3400Hz.
13. Guaranteed by design and characterization.

AC Electrical Characteristics - V_{DD}=5.0V±5%, V_{SS}=0V, -40°C ≤ To ≤ +85°C , using Test Circuit shown in Figure 10.

		Characteristics	Sym	Min	Typ‡	Max	Units	Conditions
1	TIMING	Tone present detect time	t_{DP}	5	11	14	ms	Note 1
2		Tone absent detect time	t_{DA}	0.5	4	8.5	ms	Note 1
3		Tone duration accept	t_{REC}			40	ms	Note 2
4		Tone duration reject	$t_{\overline{REC}}$	20			ms	Note 2
5		Interdigit pause accept	t_{ID}			40	ms	Note 2
6		Interdigit pause reject	t_{DO}	20			ms	Note 2
7	OUTPUTS	Propagation delay (St to Q)	t_{PQ}		8	11	µs	TOE=V_{DD}
8		Propagation delay (St to StD)	t_{PStD}		12	16	µs	TOE=V_{DD}
9		Output data set up (Q to StD)	t_{QStD}		3.4		µs	TOE=V_{DD}
10		Propagation delay (TOE to Q ENABLE)	t_{PTE}		50		ns	load of 10 kΩ, 50 pF
11		Propagation delay (TOE to Q DISABLE)	t_{PTD}		300		ns	load of 10 kΩ, 50 pF
12	PDWN	Power-up time	t_{PU}		30		ms	Note 3
13		Power-down time	t_{PD}		20		ms	
14	CLOCK	Crystal/clock frequency	f_C	3.5759	3.5795	3.5831	MHz	
15		Clock input rise time	t_{LHCL}			110	ns	Ext. clock
16		Clock input fall time	t_{HLCL}			110	ns	Ext. clock
17		Clock input duty cycle	DC_{CL}	40	50	60	%	Ext. clock
18		Capacitive load (OSC2)	C_{LO}			30	pF	

‡ Typical figures are at 25°C and are for design aid only: not guaranteed and not subject to production testing.

*NOTES:
1. Used for guard-time calculation purposes only.
2. These, user adjustable parameters, are not device specifications. The adjustable settings of these minimums and maximums are recommendations based upon network requirements.
3. With valid tone present at input, t_{PU} equals time from PDWN going low until ESt going high.

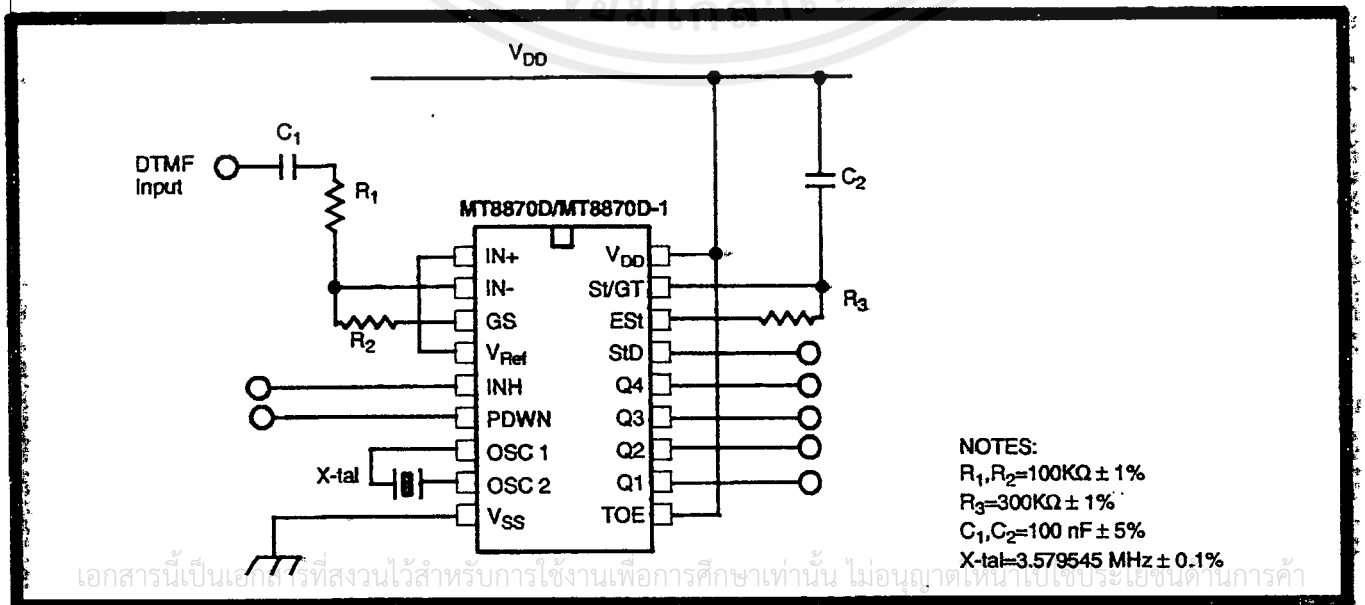
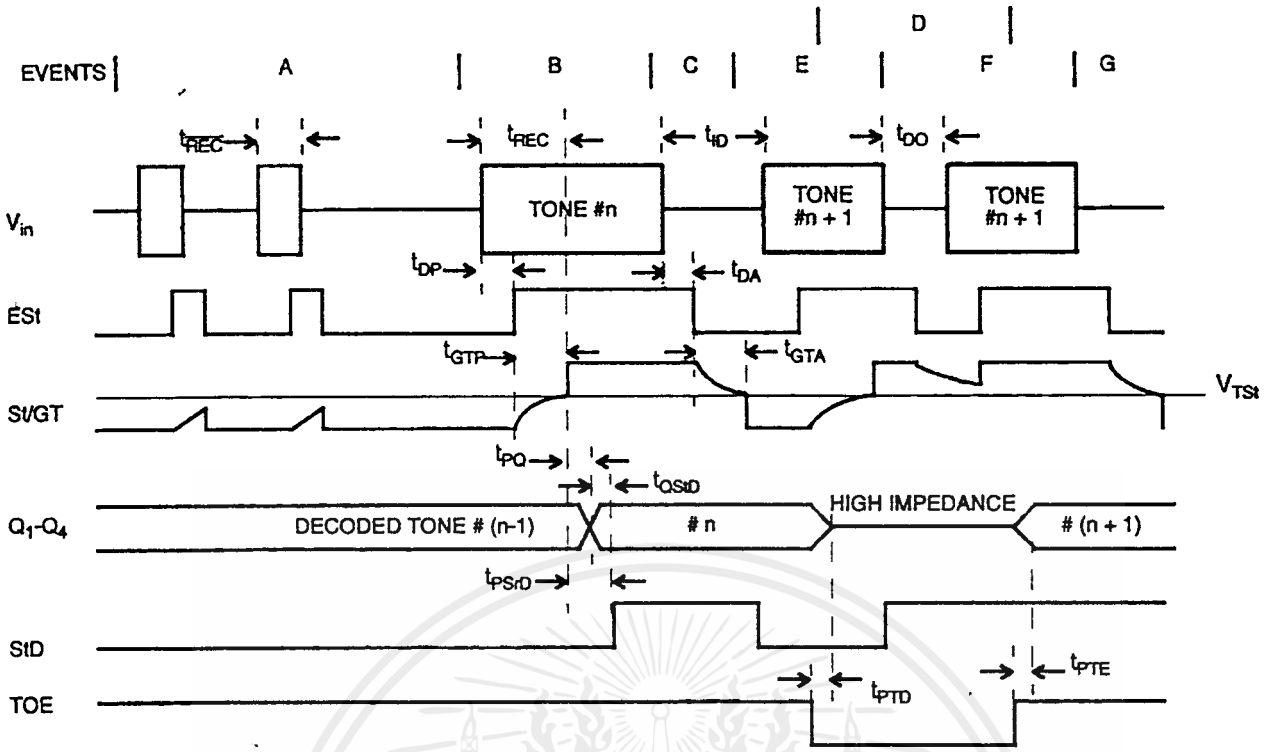


Figure 10 - Single-Ended Input Configuration



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 11 - Timing Diagram

National Semiconductor

LF353 Wide Bandwidth Dual JFET Input Operational Amplifier

General Description

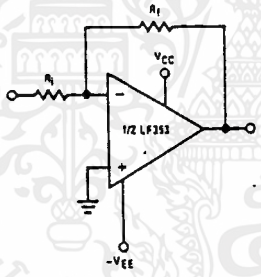
These devices are low cost, high speed, dual JFET input operational amplifiers with an internally trimmed input offset voltage (BI-FET II™ technology). They require low supply current yet maintain a large gain bandwidth product and fast slew rate. In addition, well matched high voltage JFET input devices provide very low input bias and offset currents. The LF353 is pin compatible with the standard LM1558 allowing designers to immediately upgrade the overall performance of existing LM1558 and LM358 designs.

These amplifiers may be used in applications such as high speed integrators, fast D/A converters, sample and hold circuits and many other circuits requiring low input offset voltage, low input bias current, high input impedance, high slew rate and wide bandwidth. The devices also exhibit low noise and offset voltage drift.

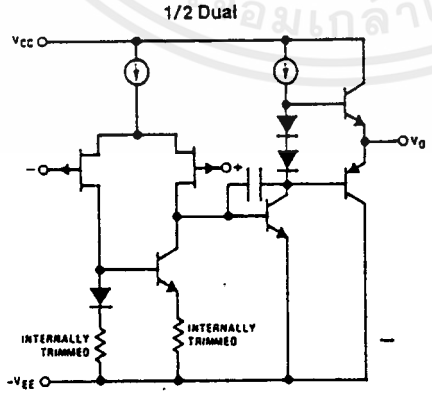
Features

■ Internally trimmed offset voltage	10 mV
■ Low input bias current	50pA
■ Low input noise voltage	25 nV/$\sqrt{Hz}$
■ Low input noise current	0.01 pA/$\sqrt{Hz}$
■ Wide gain bandwidth	4 MHz
■ High slew rate	13 V/μs
■ Low supply current	3.6 mA
■ High input impedance	$10^{12}\Omega$
■ Low total harmonic distortion $A_V = 10$, $R_L = 10k$, $V_O = 20V_{p-p}$, BW = 20 Hz-20 kHz	<0.02%
■ Low 1/f noise corner	50 Hz
■ Fast settling time to 0.01%	2 μs

Typical Connection

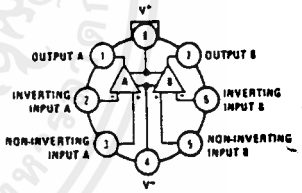


Simplified Schematic



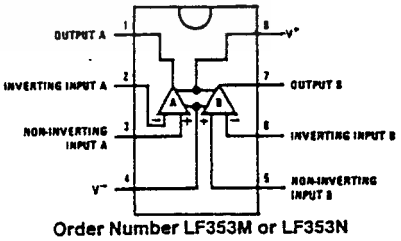
Connection Diagrams

Metal Can Package (Top View)



Order Number LF353H
See NS Package Number H08A

Dual-In-Line Package (Top View)



Order Number LF353M or LF353N
See NS Package Number M08A or N08E

TL/H/5849–1

Absolute Maximum Ratings

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

Supply Voltage	±18V
Power Dissipation	(Note 1)
Operating Temperature Range	0°C to +70°C
T_j(MAX)	150°C
Differential Input Voltage	±30V
Input Voltage Range (Note 2)	±15V
Output Short Circuit Duration	Continuous
Storage Temperature Range	−65°C to +150°C
Lead Temp. (Soldering, 10 sec.)	260°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.

ESD Tolerance (Note 7)	1700V
θ_{JA} M Package	TBD

DC Electrical Characteristics (Note 4)

Symbol	Parameter	Conditions	LF353			Units
			Min	Typ	Max	
V_{OS}	Input Offset Voltage	$R_S = 10k\Omega$, $T_A = 25°C$ Over Temperature		5	10 13	mV mV
$\Delta V_{OS}/\Delta T$	Average TC of Input Offset Voltage	$R_S = 10\ k\Omega$		10		µV/°C
I_{OS}	Input Offset Current	$T_j = 25°C$, (Notes 4, 5) $T_j \leq 70°C$		25	100 4	pA nA
I_B	Input Bias Current	$T_j = 25°C$, (Notes 4, 5) $T_j \leq 70°C$		50	200 8	pA nA
R_{IN}	Input Resistance	$T_j = 25°C$		10^{12}		Ω
A_{VOL}	Large Signal Voltage Gain	$V_S = \pm 15V$, $T_A = 25°C$ $V_O = \pm 10V$, $R_L = 2\ k\Omega$ Over Temperature	25 15	100		V/mV V/mV
V_O	Output Voltage Swing	$V_S = \pm 15V$, $R_L = 10k\Omega$	±12	±13.5		V
V_{CM}	Input Common-Mode Voltage Range	$V_S = \pm 15V$	±11	+15 −12		V V
CMRR	Common-Mode Rejection Ratio	$R_S \leq 10k\Omega$	70	100		dB
PSRR	Supply Voltage Rejection Ratio	(Note 6)	70	100		dB
I_S	Supply Current			3.6	6.5	mA

AC Electrical Characteristics (Note 4)

Symbol	Parameter	Conditions	LF353			Units
			Min	Typ	Max	
	Amplifier to Amplifier Coupling	$T_A = 25°C$, f = 1 Hz − 20 kHz (Input Referred)		−120		dB
SR	Slew Rate	$V_S = \pm 15V$, $T_A = 25°C$	8.0	13		V/µs
GBW	Gain Bandwidth Product	$V_S = \pm 15V$, $T_A = 25°C$	2.7	4		MHz
e_n	Equivalent Input Noise Voltage	$T_A = 25°C$, $R_S = 100\Omega$, f = 1000 Hz		16		$nV/\sqrt{Hz}$
i_n	Equivalent Input Noise Current	$T_j = 25°C$, f = 1000 Hz		0.01		$pA/\sqrt{Hz}$

Note 1: For operating at elevated temperatures, the device must be derated based on a thermal resistance of 115°C/W typ junction to ambient for the N package, and 158°C/W typ junction to ambient for the H package.

Note 2: Unless otherwise specified the absolute maximum negative input voltage is equal to the negative power supply voltage.

Note 3: The power dissipation limit, however, cannot be exceeded.

Note 4: These specifications apply for $V_S = \pm 15V$ and $0°C \leq T_A \leq +70°C$. V_{OS}, I_B and I_{OS} are measured at $V_{CM} = 0$.

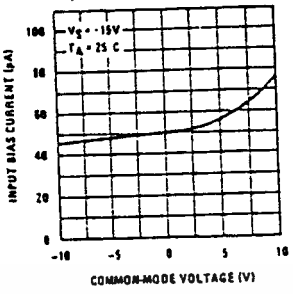
Note 5: The input bias currents are junction leakage currents which approximately double for every 10°C increase in the junction temperature, T_j. Due to the limited production test time, the input bias currents measured are correlated to junction temperature. In normal operation the junction temperature rises above the ambient temperature as a result of internal power dissipation, P_D. $T_j = T_A + \theta_{jA} P_D$ where θ_{jA} is the thermal resistance from junction to ambient. Use of a heat sink is recommended if input bias current is to be kept to a minimum.

Note 6: Supply voltage rejection ratio is measured for both supply magnitudes increasing or decreasing simultaneously in accordance with common practice. $V_S = \pm 6V$ to $\pm 15V$.

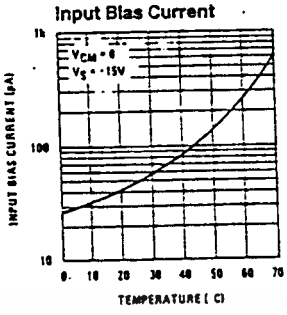
Note 7: Human body model, 1.5 kΩ in series with 100 pF.

Typical Performance Characteristics

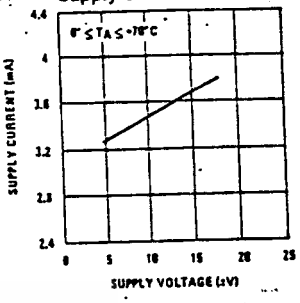
Input Bias Current



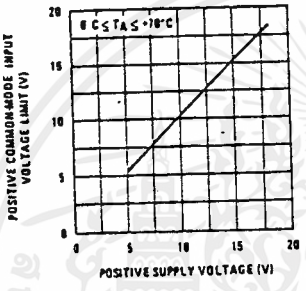
Input Bias Current



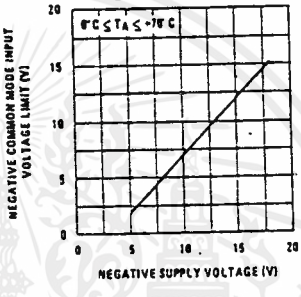
Supply Current



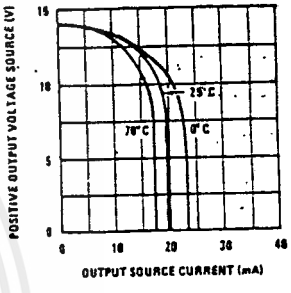
Positive Common-Mode Input Voltage Limit



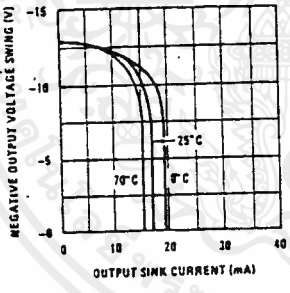
Negative Common-Mode Input Voltage Limit



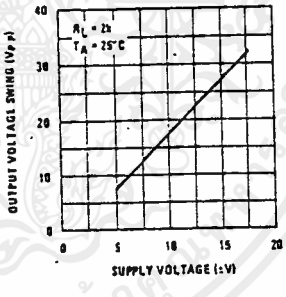
Positive Current Limit



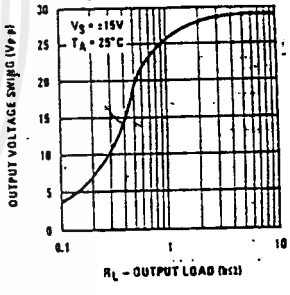
Negative Current Limit



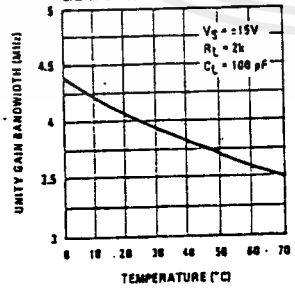
Voltage Swing



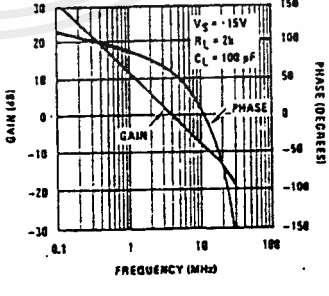
Output Voltage Swing



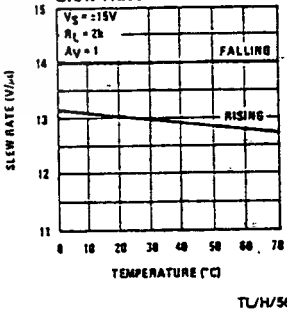
Gain Bandwidth



Bode Plot



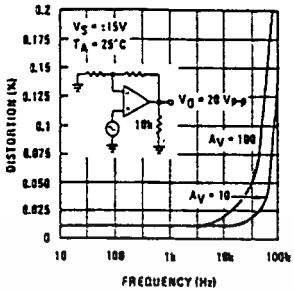
Slew Rate



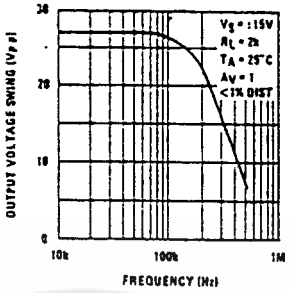
TL/H/5

Typical Performance Characteristics (Continued)

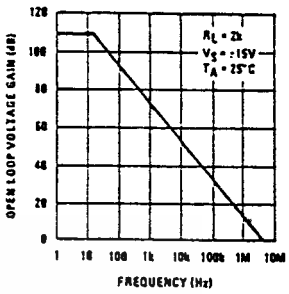
Distortion vs Frequency



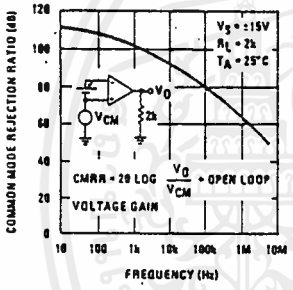
Undistorted Output Voltage Swing



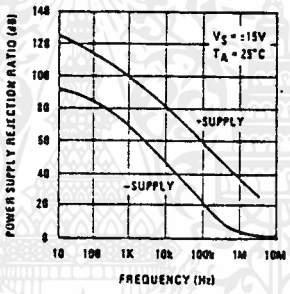
Open Loop Frequency Response



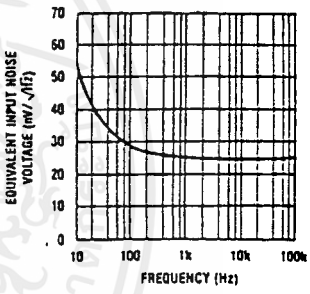
Common-Mode Rejection Ratio



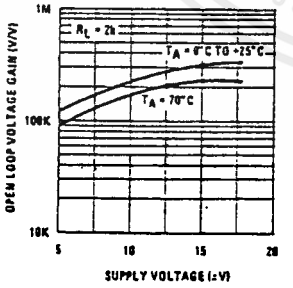
Power Supply Rejection Ratio



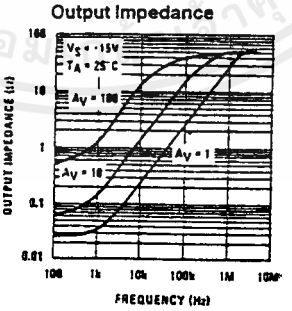
Equivalent Input Noise Voltage



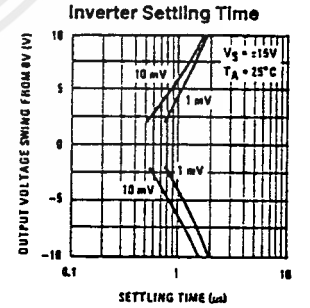
Open Loop Voltage Gain (V/V)



Output Impedance



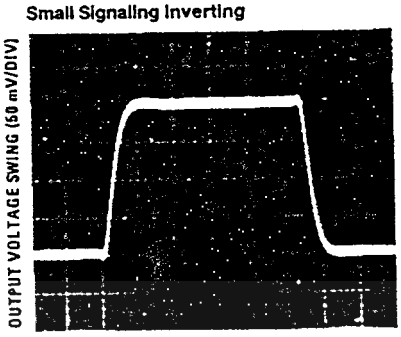
Inverter Settling Time



TL/H/5649–3

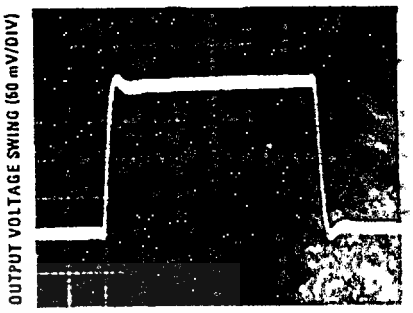
Pulse Response

Small Signaling Inverting



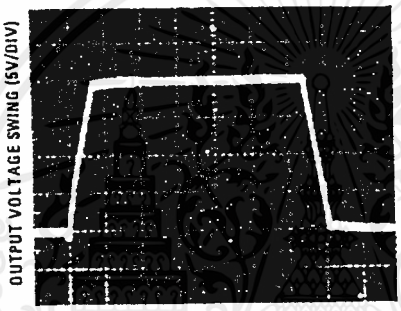
TL/H/5649–4

Small Signal Non-Inverting



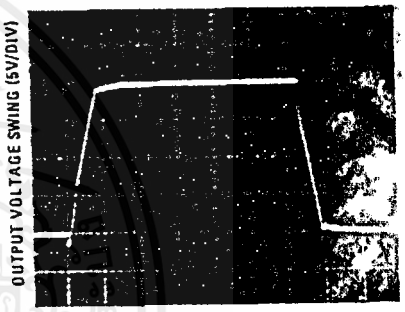
TL/H/5649

Large Signal Inverting



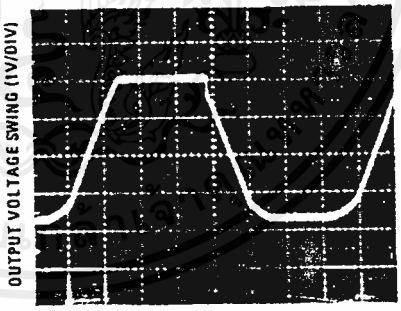
TL/H/5649–6

Large Signal Non-Inverting



TL/H/564

Current Limit (R_L = 100Ω)



TL/H/5649–8

Application Hints

These devices are op amps with an internally trimmed input offset voltage and JFET input devices (BI-FET II). These JFETs have large reverse breakdown voltages from gate to source and drain eliminating the need for clamps across the inputs. Therefore, large differential input voltages can easily be accommodated without a large increase in input current. The maximum differential input voltage is independent of the supply voltages. However, neither of the input voltages should be allowed to exceed the negative supply as this will cause large currents to flow which can result in a destroyed unit.

Exceeding the negative common-mode limit on either i will force the output to a high state, potentially causi reversal of phase to the output. Exceeding the nega common-mode limit on both inputs will force the amp output to a high state. In neither case does a latch o since raising the input back within the common-mode ra again puts the input stage and thus the amplifier in a no operating mode.

Application Hints (Continued)

Exceeding the positive common-mode limit on a single input will not change the phase of the output; however, if both inputs exceed the limit, the output of the amplifier will be forced to a high state.

The amplifiers will operate with a common-mode input voltage equal to the positive supply; however, the gain bandwidth and slew rate may be decreased in this condition. When the negative common-mode voltage swings to within 3V of the negative supply, an increase in input offset voltage may occur.

Each amplifier is individually biased by a zener reference which allows normal circuit operation on ±6V power supplies. Supply voltages less than these may result in lower gain bandwidth and slew rate.

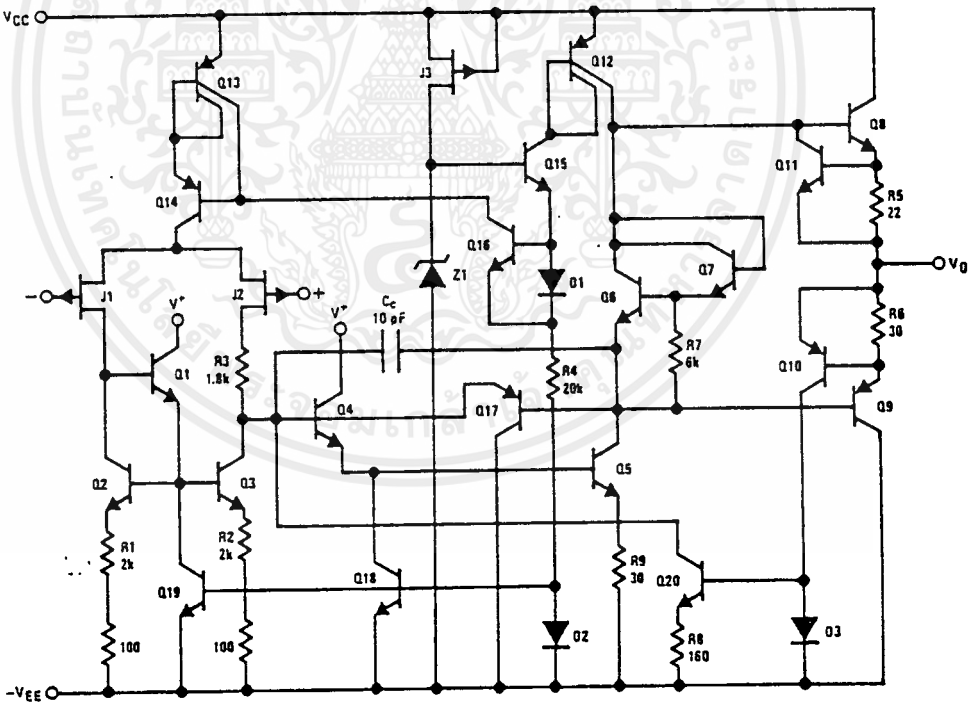
The amplifiers will drive a 2 kΩ load resistance to ±10V over the full temperature range of 0°C to +70°C. If the amplifier is forced to drive heavier load currents, however, an increase in input offset voltage may occur on the negative voltage swing and finally reach an active current limit on both positive and negative swings.

Precautions should be taken to ensure that the power supply for the integrated circuit never becomes reversed in polarity or that the unit is not inadvertently installed backwards in a socket as an unlimited current surge through the resulting forward diode within the IC could cause fusing of the internal conductors and result in a destroyed unit.

As with most amplifiers, care should be taken with lead dress, component placement and supply decoupling in order to ensure stability. For example, resistors from the output to an input should be placed with the body close to the input to minimize "pick-up" and maximize the frequency of the feedback pole by minimizing the capacitance from the input to ground.

A feedback pole is created when the feedback around any amplifier is resistive. The parallel resistance and capacitance from the input of the device (usually the inverting input) to AC ground set the frequency of the pole. In many instances the frequency of this pole is much greater than the expected 3 dB frequency of the closed loop gain and consequently there is negligible effect on stability margin. However, if the feedback pole is less than approximately 6 times the expected 3 dB frequency a lead capacitor should be placed from the output to the input of the op amp. The value of the added capacitor should be such that the RC time constant of this capacitor and the resistance it parallels is greater than or equal to the original feedback pole time constant.

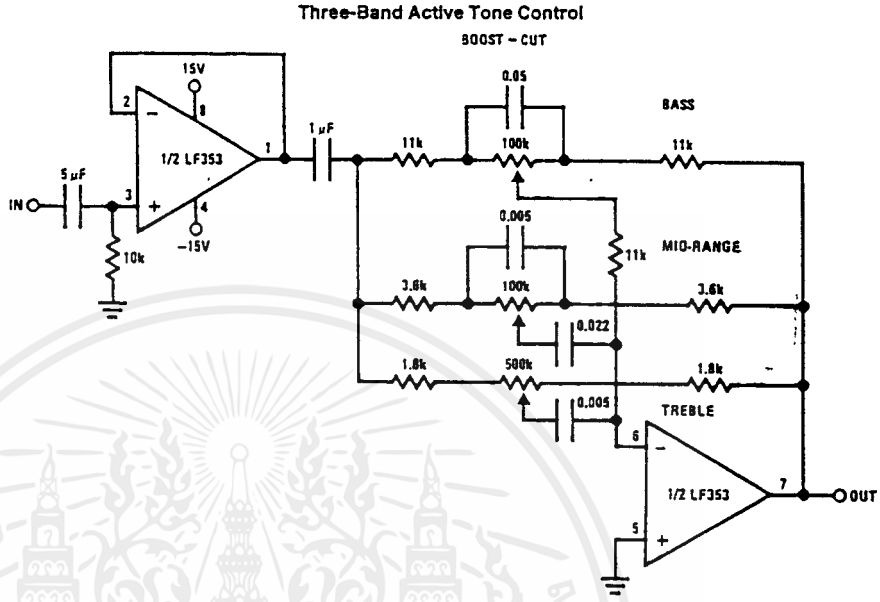
Detailed Schematic

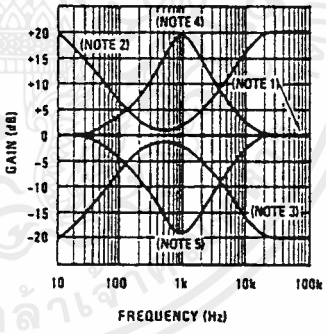


TL/H/5849–9

Typical Applications

Three-Band Active Tone Control



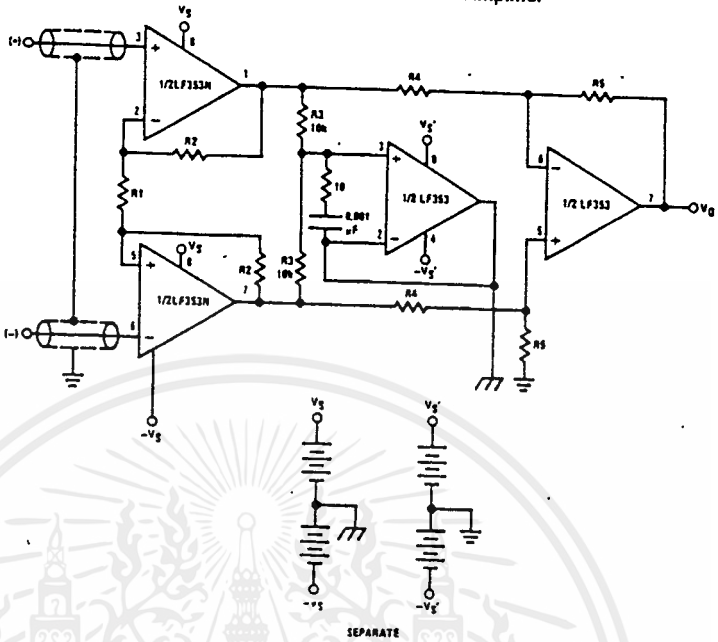


Note 1: All controls flat.
Note 2: Bass and treble boost, mid flat.
Note 3: Bass and treble cut, mid flat.
Note 4: Mid boost, bass and treble flat.
Note 5: Mid cut, bass and treble flat.

- All potentiometers are linear taper
- Use the LF347 Quad for stereo applications

Typical Applications (Continued)

Improved CMRR Instrumentation Amplifier



$$A_V = \left(\frac{2R2}{R1} + 1\right) \frac{R5}{R4}$$

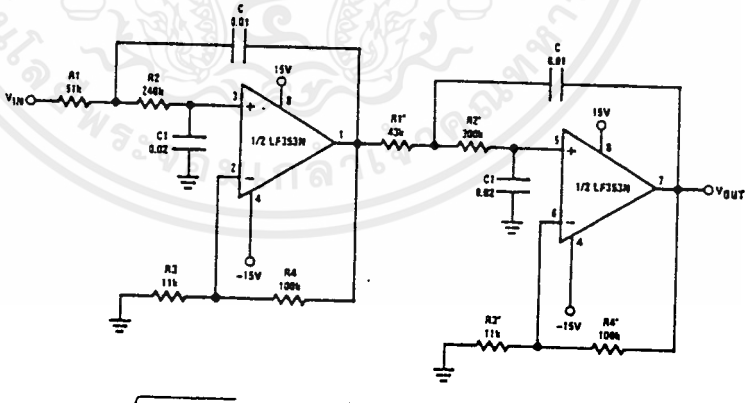
and are separate isolated grounds

Matching of R2's, R4's and R5's control CMRR

With A_{V_T} = 1400, resistor matching = 0.01%: CMRR = 136 dB

- Very high input impedance
- Super high CMRR

Fourth Order Low Pass Butterworth Filter

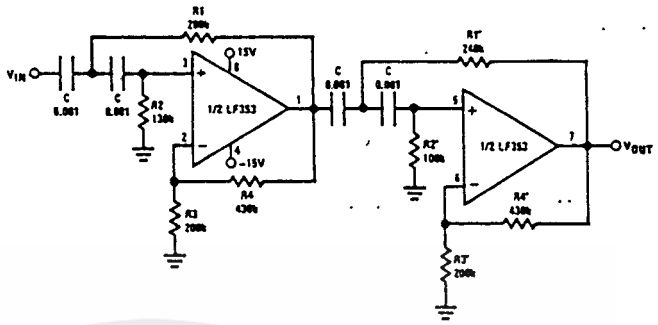


- Corner frequency (f_c) = $\sqrt{\frac{1}{R1R2CC1}} \cdot \frac{1}{2\pi} = \sqrt{\frac{1}{R1'R2'CC1}} \cdot \frac{1}{2\pi}$
- Passband gain (H_0) = (1 + R4/R3) (1 + R4'/R3')
- First stage Q = 1.31
- Second stage Q = 0.541
- Circuit shown uses nearest 5% tolerance resistor values for a filter with a corner frequency of 100 Hz and a passband gain of 100
- Offset nulling necessary for accurate DC performance

TL/H/5649–11

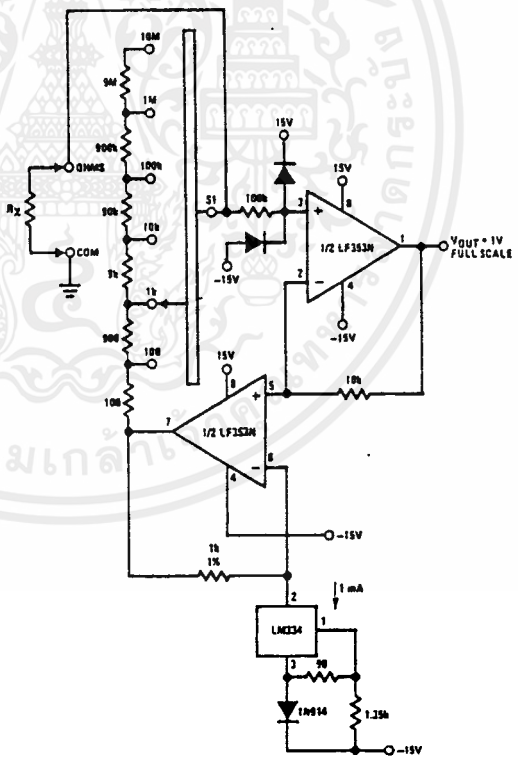
Typical Applications (Continued)

Fourth Order High Pass Butterworth Filter



- Corner frequency (f_c) = $\sqrt{\frac{1}{R1R2C^2}} \cdot \frac{1}{2\pi} = \sqrt{\frac{1}{R1'R2'C^2}} \cdot \frac{1}{2\pi}$
- Passband gain (H_O) = (1 + R4/R3) (1 + R4'/R3')
- First stage Q = 1.31
- Second stage Q = 0.541
- Circuit shown uses closest 5% tolerance resistor values for a filter with a corner frequency of 1 kHz and a passband gain of 10.

Ohms to Volts Converter



TL/H/5649–13

$$V_O = \frac{1V}{R_{LADDER}} \times R_X$$

Where R_{LADDER} is the resistance from switch S1 pole to pin 7 of the LF353.

กิตติกรรมประกาศ

ปริญญานิพนธ์นี้จะไม่สามารถสำเร็จลุล่วงไปได้ด้วยดีถ้าไม่มีผู้เกี่ยวข้องต่อไปนี้

- รศ. ดร. กอบชัย เดชหาญ ที่คอยให้คำปรึกษาด้านต่างๆ ที่เป็นประโยชน์จนทำให้โครงงานสำเร็จลุล่วงไปได้ด้วยดี
- อ. สมเกียรติ ฤกษ์วรัญญู ที่ช่วยให้คำปรึกษา
- นายทิษณุ งามเธียรธนา และนายชวลิต ชั้นไพบูลย์ ที่ช่วยให้ความช่วยเหลือและให้ความสะดวกด้านอุปกรณ์
- นายเทอดศักดิ์ ธนากิจประภา ที่ให้ความช่วยเหลือและคำปรึกษาในส่วนของการเขียนโปรแกรม Visual C++
- นายณฤทธิ์ ฉัังฉิศรา ที่ให้ความช่วยเหลือในส่วนของวงจร
- และขอขอบคุณเพื่อนๆและพี่ๆทุกคนที่ให้ความช่วยเหลือในการอัดเสียงทำโมเดลเสียง

จึงขอแสดงความขอบคุณมา ณ ที่นี้ด้วย

คณะผู้จัดทำ

เอกสารอ้างอิง

1. LAWRENCE RABINER,BIING-HWANG JUANG:FUNDAMENTALS OF SPEECH RECONITION

2. น.ส กรุณา แก้วสมศรี และคณะ, การต่อหมายเลขโทรศัพท์โดยใช้เสียง, ปริญญานิพนธ์ : สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, พ.ศ.2539

3. ชวดี อิศรปรีดา และคณะ, การรู้จำเสียงพูด, ปริญญานิพนธ์ : สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, พ.ศ.2538

4. สุนทร อรอินทร์ และ อัฐ เครือฟัก, การประมวลเสียงพูดโดยการประมาณเชิงเส้น, ปริญญานิพนธ์ : สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, พ.ศ.2537

5. วิทยา เรืองพรวิสุทธ์, การเขียนโปรแกรมภาษา C สำหรับผู้เริ่มต้น, กรุงเทพ : บริษัท ซีเอ็ดยูเคชั่น จำกัด, พ.ศ. 2537

6. วรา คงคาวิทูร และ วิรัช ลีนะวงศ์อนันต์, การควบคุมอุปกรณ์ไฟฟ้าโดยใช้เสียง, ปริญญานิพนธ์ : สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, พ.ศ.2540

ภาคผนวก

```
  //{{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)
    // No message handlers
  //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CArt3Dlg dialog

IMPLEMENT_DYNAMIC(CArt3Dlg, CDialog);

CArt3Dlg::CArt3Dlg(CWnd* pParent):CDialog(CArt3Dlg::IDD, pParent)
 {
 //{{AFX_DATA_INIT(CArt3Dlg)
 //}}AFX_DATA_INIT
 // Note that LoadIcon does not require a subsequent DestroyIcon in Win32
 m_hIcon=AfxGetApp()->LoadIcon(IDR_MAINFRAME);
 m_pAutoProxy=NULL;
 m_pRECORDThread=NULL;

 hEventDone=CreateEvent(NULL,FALSE,FALSE,NULL);
 hEventDead=CreateEvent(NULL,FALSE,FALSE,NULL);
 hEventKill=CreateEvent(NULL,FALSE,FALSE,NULL);
 }

CArt3Dlg::~CArt3Dlg()
 {
 // If there is an automation proxy for this dialog, set
 //  its back pointer to this dialog to NULL, so it knows
 //  the dialog has been deleted.
 //if(m_pAutoProxy!=NULL) m_pAutoProxy->m_pDialog=NULL;
 if(m_pRECORDThread!=NULL) m_pRECORDThread->KillThread();

 CloseHandle(hEventDone);
 CloseHandle(hEventDead);
 CloseHandle(hEventKill);
 }

void CArt3Dlg::DoDataExchange(CDataExchange* pDX)
 {
 CDialog::DoDataExchange(pDX);
 //{{AFX_DATA_MAP(CArt3Dlg)
 //}}AFX_DATA_MAP
 }

BEGIN_MESSAGE_MAP(CArt3Dlg, CDialog)
 //{{AFX_MSG_MAP(CArt3Dlg)
 ON_WM_SYSCOMMAND()
 ON_WM_PAINT()
 ON_WM_QUERYDRAGICON()
 ON_WM_CLOSE()
 ON_BN_CLICKED(IDC_BUTTON1, OnRunIpc)
```

```
 ON_BN_CLICKED(IDC_BUTTON2, OnRunCombine)
 ON_BN_CLICKED(IDC_BUTTON3, OnRunVqln)
 ON_BN_CLICKED(IDC_BUTTON4, OnRunVqcmp)
 ON_BN_CLICKED(IDC_BUTTON5, OnRunHMM)
 ON_BN_CLICKED(IDC_BUTTON6, OnRecogmanual)
 ON_BN_CLICKED(IDC_BUTTON10, OnSegmenting)
 ON_BN_CLICKED(IDC_BUTTON8, OnEnergy)
 ON_BN_CLICKED(IDC_BUTTON7, OnSegRecog)
 ON_BN_CLICKED(IDC_BUTTON9, OnPORT)
 ON_BN_CLICKED(IDC_BUTTON12, OnCommandManual)
 ON_BN_CLICKED(IDC_BUTTON13, OnCommandAuto)
 ON_BN_CLICKED(IDC_BUTTON14, OnCommandAutoPort)
 ON_BN_CLICKED(IDC_BUTTON11, OnEdit)
 //}}AFX_MSG_MAP
END_MESSAGE_MAP()

////////////////////////////////////////////////////////////////////////////
// CArt3Dlg message handlers

BOOL CArt3Dlg::OnInitDialog()
 {
 CDialog::OnInitDialog();

 // Add "About..." menu item to system menu.

 // IDM_ABOUTBOX must be in the system command range.
 ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
 ASSERT(IDM_ABOUTBOX < 0xF000);

 CMenu* pSysMenu=GetSystemMenu(FALSE);
 if(pSysMenu!=NULL)
  {
  CString strAboutMenu;
  strAboutMenu.LoadString(IDS_ABOUTBOX);
  if(!strAboutMenu.IsEmpty())
   {
   pSysMenu->AppendMenu(MF_SEPARATOR);
   pSysMenu->AppendMenu(MF_STRING,IDM_ABOUTBOX,strAboutMenu);
   }
  }
 // Set the icon for this dialog.  The framework does this automatically
 //  when the application's main window is not a dialog
 SetIcon(m_hIcon,TRUE ); // Set big icon
 SetIcon(m_hIcon,FALSE); // Set small icon

 return TRUE;
 }

void CArt3Dlg::OnSysCommand(UINT nID, LPARAM lParam)
{
  if ((nID & 0xFFF0) == IDM_ABOUTBOX)
  {
    CAboutDlg dlgAbout;
    dlgAbout.DoModal();
  }
  else
  {
    CDialog::OnSysCommand(nID, lParam);
  }
}

// If you add a minimize button to your dialog, you will need the code below
//  to draw the icon.  For MFC applications using the document/view model,
```

```
// this is automatically done for you by the framework.

void CArt3Dlg::OnPaint()
{
  if (IsIconic())
  {
    CPaintDC dc(this); // device context for painting

    SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(), 0);

    // Center icon in client rectangle
    int cxIcon = GetSystemMetrics(SM_CXICON);
    int cyIcon = GetSystemMetrics(SM_CYICON);
    CRect rect;
    GetClientRect(&rect);
    int x = (rect.Width() - cxIcon + 1) / 2;
    int y = (rect.Height() - cyIcon + 1) / 2;

    // Draw the icon
    dc.DrawIcon(x, y, m_hIcon);
  }
  else
  {
    CDialog::OnPaint();
  }
}

// The system calls this to obtain the cursor to display while the user drags
// the minimized window.
HCURSOR CArt3Dlg::OnQueryDragIcon()
{
  return (HCURSOR) m_hIcon;
}

// Automation servers should not exit when a user closes the UI
// if a controller still holds on to one of its objects.  These
// message handlers make sure that if the proxy is still in use,
// then the UI is hidden but the dialog remains around if it
// is dismissed.

void CArt3Dlg::OnClose()
{
  if (CanExit())
    CDialog::OnClose();
}

void CArt3Dlg::OnOK()
{
  if(m_pRECORDThread!=NULL) {m_pRECORDThread->KillThread(); Sleep(5000);}
  if (CanExit())
    CDialog::OnOK();
}

void CArt3Dlg::OnCancel()
{
  if (CanExit())
    CDialog::OnCancel();
}

BOOL CArt3Dlg::CanExit()
{
  // If the proxy object is still around, then the automation
  // controller is still holding on to this application.  Leave
```

```
    // the dialog around, but hide its UI.
    if (m_pAutoProxy != NULL)
    {
        ShowWindow(SW_HIDE);
        return FALSE;
    }

    return TRUE;
}

void CArt3Dlg::OnRunlpc()
{
    // TODO: Add your control notification handler code here
    //LPC lpc;
    List list;

    if( list.DoModal()==IDOK )
    {
        
        
        UpdateData(false);
    }

}



void CArt3Dlg::OnRunCombine()
{
    Combinedia com;

    if( com.DoModal()==IDOK )
    {

        UpdateData(false);
    }
}

void CArt3Dlg::OnRunVqln()
{
    Vqlndia vqlndia;
    vqlndia.m_input="AllSample.lpc";
    vqlndia.m_out="Codebook.vq";
    vqlndia.m_outtext="Codebook.txt";

    UpdateData(false);
    if (vqlndia.DoModal()==IDOK)
    {

    UpdateData(false);
    }
}

void CArt3Dlg::OnRunVqcmp()
{
    Vqcomdia   vqcomdia;

    vqcomdia.m_input="Codebook.vq";

    if( vqcomdia.DoModal()==IDOK )
    {

        UpdateData(false);
    }
}
```

```
void CArt3Dlg::OnRunHMM()
{
  Hmmdia hmmdia;

  if( hmmdia.DoModal()==IDOK )
  {

    UpdateData(false);
  }
}

void CArt3Dlg::OnRecogmanual()
{

  Recogdia recogdia;

  if( recogdia.DoModal()==IDOK )
  {

    UpdateData(false);
  }
}

void CArt3Dlg::OnSegmenting()
{
  Segmentdia segmentdia;

  segmentdia.m_percent="100";
  segmentdia.m_min="50";
  segmentdia.m_max="20";
  if (segmentdia.DoModal()==IDOK)
  {
    UpdateData(true);
  }
  UpdateData(false);
}

void CArt3Dlg::OnEnergy()
{
  energydia  energydia;

  energydia.m_shift="50";
  energydia.m_window="300";

  if (energydia.DoModal()==IDOK)
  {
    UpdateData(true);
  }
  UpdateData(false);
}


void CArt3Dlg::OnSegRecog()
{
    Segandrecog segandrecog;
  if (segandrecog.DoModal()==IDOK)
  {
    UpdateData(true);
  }
  UpdateData(false);
}
```

```
void CArt3Dlg::OnEdit()
{
  Editwav editwav;
  if (editwav.DoModal()==IDOK)
  {
    UpdateData(true);
  }
  UpdateData(false);
}

void CArt3Dlg::OnPORT()
{
  Port port;
  if (port.DoModal()==IDOK)
  {
    UpdateData(true);
  }
  UpdateData(false);
}

void CArt3Dlg::OnCommandManual()
{
  commanual comm;

  if( comm.DoModal()==IDOK )
  {

    UpdateData(false);
  }
}

void CArt3Dlg::OnCommandAuto()
{
/*  CString m_input="art.wav";
  Wave wave;

   int check=MessageBox("Click","Begin click OK", MB_OKCANCEL);
   if (check==IDOK)
   {
    wave.openwav(m_input);
    Sleep(50);
    wave.recordwav();
    Sleep(50);
    wave.stopwav();
    wave.savewav(m_input);
    wave.closewav();
   }
  //MessageBox("Click","Wait", MB_OK);

/*  MessageBox("Segmenting&Recognizing","Processing", MB_OK);
    Segment segment;
    segment.energy(m_input,"temp.eng"
          ,"temp_eng.txt"
          ,"temp_wav.txt"
          ,50
          ,300);

    segment.segmentEnergy("temp.eng"
          ,"temp.seg"
          ,"temp_seg.txt"
          ,"summary.txt"
          ,100
          ,50
```

```
            ,20);
  segment.editrecog(m_input,"temp.scg","result.txt");*/
  MessageBox("Commandauto","Complete!");
  UpdateData(false);
}

void CArt3Dlg::OnCommandAutoPort()
 {
 if(m_pRECORDThread==NULL)
  {
  m_pRECORDThread=(CRecord*)AfxBeginThread(RUNTIME_CLASS(CRecord),THREAD_PRIORITY_NORMAL,0,CREATE_SUSPENDED);
  m_pRECORDThread->SetOwner(this);
  m_pRECORDThread->ResumeThread();
  }
 else m_pRECORDThread->KillThread();

 //Command a;
 //a.Commandautoport();
 }
```

```
// Editwav.cpp : implementation file
//

#include "stdafx.h"
#include "art3.h"
#include "Editwav.h"
#include "Segment.h"

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

/////////////////////////////////////////////////////////////////////////////
// Editwav dialog


Editwav::Editwav(CWnd* pParent /*=NULL*/)
  : CDialog(Editwav::IDD, pParent)
{
  //{{AFX_DATA_INIT(Editwav)
  m_input1 = _T("");
  m_input2 = _T("");
  //}}AFX_DATA_INIT
}


void Editwav::DoDataExchange(CDataExchange* pDX)
{
  CDialog::DoDataExchange(pDX);
  //{{AFX_DATA_MAP(Editwav)
  DDX_Control(pDX, IDC_LIST1, m_list);
  DDX_Text(pDX, IDC_EDIT1, m_input1);
  DDX_Text(pDX, IDC_EDIT2, m_input2);
  //}}AFX_DATA_MAP
}


BEGIN_MESSAGE_MAP(Editwav, CDialog)
  //{{AFX_MSG_MAP(Editwav)
  ON_BN_CLICKED(IDC_BUTTON1, OnShowList)
  ON_BN_CLICKED(IDC_BUTTON2, OnRUN)
  ON_LBN_DBLCLK(IDC_LIST1, OnInput)
  //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// Editwav message handlers

void Editwav::OnShowList()
{
  CFileFind  filefind;
  char    notfound;

  m_list.ResetContent();
  filefind.FindFile("*.wav",0);
  for (;notfound!=0;)
  {
    notfound=filefind.FindNextFile();
    m_list.AddString(filefind.GetFileName());
  }
  UpdateData(false);
```

```
}

void Editwav::OnInput()
{

  UpdateData(true);
  m_list.GetText(m_list.GetCurSel(),m_input1);
  int pos=m_input1.Find('.');
    m_input2=m_input1;
    m_input2.SetAt(pos+1,'s');
    m_input2.SetAt(pos+2,'e');
    m_input2.SetAt(pos+3,'g');
  UpdateData(false);
}

void Editwav::OnRUN()
{
  Segment segment;
  UpdateData(true);

  segment.editwave(m_input1,m_input2);

  MessageBox(m_input1,"Splitting..Complete!");
  UpdateData(false);

}
```

```
// energydia.cpp : implementation file
//

#include "stdafx.h"
#include "art3.h"
#include "energydia.h"
#include "Segment.h"

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

CString m_outwavtext;
/////////////////////////////////////////////////////////////////////////////
// energydia dialog


energydia::energydia(CWnd* pParent /*=NULL*/)
  : CDialog(energydia::IDD, pParent)
{
  //{{AFX_DATA_INIT(energydia)
  m_input = _T("");
  m_output = _T("");
  m_outtext = _T("");
  m_shift = _T("");
  m_window = _T("");
  //}}AFX_DATA_INIT
}


void energydia::DoDataExchange(CDataExchange* pDX)
{
  CDialog::DoDataExchange(pDX);
  //{{AFX_DATA_MAP(energydia)
  DDX_Control(pDX, IDC_LIST1, m_list);
  DDX_Text(pDX, IDC_EDIT1, m_input);
  DDX_Text(pDX, IDC_EDIT2, m_output);
  DDX_Text(pDX, IDC_EDIT3, m_outtext);
  DDX_Text(pDX, IDC_EDIT4, m_shift);
  DDX_Text(pDX, IDC_EDIT5, m_window);
  //}}AFX_DATA_MAP
}


BEGIN_MESSAGE_MAP(energydia, CDialog)
  //{{AFX_MSG_MAP(energydia)
  ON_BN_CLICKED(IDC_BUTTON1, OnShowlist)
  ON_BN_CLICKED(IDC_BUTTON2, OnRUN)
  ON_LBN_DBLCLK(IDC_LIST1, OnCopyinput)
  //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// energydia message handlers

void energydia::OnShowlist()
{

  CFileFind  filefind;
  char     notfound;
```

```
// art3Dlg.cpp : implementation file
//

#include "stdafx.h"
#include "art3.h"
#include "art3Dlg.h"
#include "DlgProxy.h"
#include "list.h"
#include "ombinedia.h"
#include "Vqlndia.h"
#include "Vqcondia.h"
#include "Hmmdia.h"
#include "Recogdia.h"
#include "energydia.h"
#include "Segmentdia.h"
#include "Editwav.h"
#include "Segandrecog.h"
#include "Port.h"
#include "commanual.h"
#include "Wave.h"
#include "Segment.h"
#include "Command.h"
#include "Record.h"

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

HANDLE hEvent[2];
HANDLE hEventDone;
HANDLE hEventDead;
HANDLE hEventKill;

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

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)
{
```

```
    m_list.ResetContent();
    filefind.FindFile("*.wav",0);
    for (;notfound!=0;)
    {
      notfound=filefind.FindNextFile();
      m_list.AddString(filefind.GetFileName());
    }
    UpdateData(false);
}

void energydia::OnCopyinput()
{
    UpdateData(true);

    m_list.GetText(m_list.GetCurSel(),m_input);
      int pos=m_input.Find('.');
      m_output=m_input;
      m_output.SetAt(pos+1,'e');
      m_output.SetAt(pos+2,'n');
      m_output.SetAt(pos+3,'g');
      m_outtext=m_input;
      m_outtext.SetAt(pos,'-');
      m_outtext.SetAt(pos+1,'E');
      m_outtext.SetAt(pos+2,'N');
      m_outtext.SetAt(pos+3,'G');
      m_outtext+=".txt";
      m_outwavtext=m_input;
      m_outwavtext.SetAt(pos,'-');
      m_outwavtext.SetAt(pos+1,'W');
      m_outwavtext.SetAt(pos+2,'A');
      m_outwavtext.SetAt(pos+3,'V');
      m_outwavtext+=".txt";
    UpdateData(false);
}

void energydia::OnRUN()
{
    Segment segment;
    UpdateData(true);
    for (int index=0;index<m_list.GetCount();index++)
    {
      if (m_list.GetSel(index)!=0)
      {
        m_list.GetText(index,m_input);
      int pos=m_input.Find('.');
      m_output=m_input;
      m_output.SetAt(pos+1,'e');
      m_output.SetAt(pos+2,'n');
      m_output.SetAt(pos+3,'g');
      m_outtext=m_input;
      m_outtext.SetAt(pos,'-');
      m_outtext.SetAt(pos+1,'E');
      m_outtext.SetAt(pos+2,'N');
      m_outtext.SetAt(pos+3,'G');
      m_outtext+=".txt";
      m_outwavtext=m_input;
      m_outwavtext.SetAt(pos,'-');
      m_outwavtext.SetAt(pos+1,'W');
      m_outwavtext.SetAt(pos+2,'A');
      m_outwavtext.SetAt(pos+3,'V');
      m_outwavtext+=".txt";
      segment.energy(m_input,m_output
            ,m_outtext
```

```
            ,m_outwavtext
            ,atoi(m_shift)
            ,atoi(m_window));
    }
  }
  MessageBox(m_input,"Calculating Energy..Complete!");
  UpdateData(false);
}
```

```
// HMM.cpp : implementation file
//

#include "stdafx.h"
#include "art3.h"
#include "HMM.h"
#include "math.h"
#include "time.h"

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

/////////////////////////////////////////////////////////////////////////////
// HMM

//IMPLEMENT_DYNCREATE(HMM, CView)

HMM::HMM()
{
}

HMM::~HMM()
{
}

//************************** PROCEDURE HMM *********************************
bool HMM::open_file(CString observation_file_name,CString model_file_name_hmm,CString model_text_file_name_hmm)
{
    if ((observation_file = fopen(observation_file_name,"rb")) == NULL)
    {
        MessageBox(observation_file_name,"Can't open file", MB_OK);
        return false;
    }
    if ((model_file = fopen(model_file_name_hmm,"wb")) == NULL)
    {
        MessageBox(model_file_name_hmm, "Can't create file", MB_OK);
        return false;
    }

    if ((model_text_file = fopen(model_text_file_name_hmm,"wt")) == NULL)
    {
        MessageBox(model_text_file_name_hmm, "Can't create file", MB_OK);
        return false;
    }
    return true;
}
int HMM::find_number_of_observation(void)
{
    char    temp;
    int     number_of_observation=0;
    rewind(observation_file);
    for (;;)
    {
        fread(&temp,1,1,observation_file);
        if (temp != -1 )  number_of_observation++; else break;
        if (feof(observation_file))
        {
            MessageBox("Invalid input file type", "ERROR!", MB_OK);
            return 0;
        }
    }
    rewind(observation_file);
    return number_of_observation;
}
bool HMM::allocate_memory(void)
{
    char    temp;
    int number_of_observation=find_number_of_observation();
    T1 = (char *) calloc(number_of_observation,sizeof(char));
    if (T1 == NULL)
    {
        MessageBox("Can't allocate memory for T1", "ERROR!", MB_OK);
        return false;
    }
    fread(T1,1,number_of_observation,observation_file);
    fread(&temp,1,1,observation_file);
    for (int i=0;i<number_of_observation;i++)
        if (T1[i]<0) T1[i]=256+T1[i];
            maximum = T1[0];
    for (i=1;i<number_of_observation;i++)
        if (maximum < T1[i])
            maximum = T1[i];
    O = (char *) calloc(number_of_observation*maximum,sizeof(char));
    if (O == NULL)
    {
        MessageBox("Can't allocate memory for O", "ERROR!", MB_OK);
        return false;
    }

    lf_test=(long double*)calloc(maximum*N1,sizeof(long double));
    cs_test=(long double*)calloc(maximum,sizeof(long double));

    for (i=0;i<number_of_observation;i++)
    {
        fread(&O[i*maximum],1,T1[i],observation_file);
    }

    lfps = (long double*)calloc(maximum*N1,sizeof(long double));
    lfs  = (long double*)calloc(maximum*N1,sizeof(long double));
    Bt   = (long double*)calloc(maximum*N1,sizeof(long double));
    Bts  = (long double*)calloc(maximum*N1,sizeof(long double));
    sc   = (long double*)calloc(maximum,  sizeof(long double));
```

```
    c1   = (long double*)calloc(maximum,  sizeof(long double));
    plog = (long double*)calloc(number_of_observation,sizeof(long double));

    if ((lf_test||lfps||lfs||Bt||CB_test||Bts||sc||c1||plog) == NULL)
    {
        MessageBox("Can't allocate memory for variable", "ERROR!", MB_OK);
        return false;
    }
    return true;
}

void HMM::random_abvalue(void)
{
    long double     sum_a[N1],sum_b[N1];

    a[0][0]=(1.00/3.00);    a[0][1]=(1.00/3.00);    a[0][2]=(1.00/3.00);    a[0][3]=0;              a[0][4]=0;
  a[0][5]=0;
    a[1][0]=0;              a[1][1]=(1.00/3.00);    a[1][2]=(1.00/3.00);    a[1][3]=(1.00/3.00);    a[1][4]=0;
  a[1][5]=0;
    a[2][0]=0;              a[2][1]=0;              a[2][2]=(1.00/3.00);    a[2][3]=(1.00/3.00);    a[2][4]=(1.00/3.00);
  a[2][5]=0;
    a[3][0]=0;              a[3][1]=0;              a[3][2]=0;              a[3][3]=(1.00/3.00);    a[3][4]=(1.00/3.00);
  a[3][5]=(1.00/3.00);
    a[4][0]=0;              a[4][1]=0;              a[4][2]=0;              a[4][3]=0;              a[4][4]=(1.00/2.00);
  a[4][5]=(1.00/2.00);
    a[5][0]=0;              a[5][1]=0;              a[5][2]=0;              a[5][3]=0;              a[5][4]=0;
  a[5][5]=1;
    for(int i=0;i<N1;i++)
        for(int k=0;k<K;k++)
        {
/*          int random_number=100;
            for (;(random_number<0)||(random_number>99);)
                random_number=rand();
            b[i][k]=(float)random_number;
*/
            b[i][k]=k+1;
        }
    for(i=0;i<N1;i++)
    {
        sum_a[i] = 0;
        sum_b[i] = 0;
        for(int j=0;j<N1;j++)
            sum_a[i] += a[i][j];
        for(int k=0;k<K;k++)
            sum_b[i] += b[i][k];
    }
    for(i=0;i<N1;i++)
    {
        for(int j=0;j<N1;j++)
        {
//          VERIFY(sum_a[i]!=0);
            if(sum_a[i]!=0)
                aprime[i][j] = a[i][j]/sum_a[i];
        }
        for(int k=0;k<K;k++)
        {
//          VERIFY(sum_b[i]!=0);
            if(sum_b[i]!=0)
                bprime[i][k] = b[i][k]/sum_b[i];
        }
    }
    for(i=0;i<N1;i++)
        for(int j=0;j<N1;j++)
        {
            Aprime[i][j] = 0;
            AMprime[i] = 0;
        }
    for(i=0;i<N1;i++)
        for(int k=0;k<K;k++)
        {
            Bprime[i][k] = 0;
            BMprime[i] = 0;
        }
}

void HMM::copy_abprime_to_ab(int w)
{
    for(int i=0;i<N1;i++)
        for(int j=0;j<N1;j++)
        {
            a[i][j] = aprime[i][j];
        }
    for(i=0;i<N1;i++)
        for(int k=0;k<K;k++)
        {
            b[i][k] = bprime[i][k];
        }


}

void HMM::find_lf_bt_value(int w,int v)
{
    CString temp;
    long double slf;
    Pi[0]=1;Pi[1]=0;Pi[2]=0;Pi[3]=0;Pi[4]=0;Pi[5]=0;
    for(int i=0;i<N1;i++)
    {
//      VERIFY(i>=0||i<maximum*N1);
        if(i>=0||i<maximum*N1)
        lf_test[i] = Pi[i] * b[i][O[v*maximum+0]];
    }

    for(int t=0;t<T1[v]-1;t++)
        for(int j=0;j<N1;j++)
```

```
        {
            slf = 0;
            for(i=0;i<N1;i++)
            {
//              VERIFY(t*N1+i>=0||t*N1+i<maximum*N1);
                if(t*N1+i>=0||t*N1+i<maximum*N1)
                slf += lf_test[t*N1+i] * a[i][j];
            }
//          VERIFY((t+1)*N1+j>=0||(t+1)*N1+j<maximum*N1);
            if((t+1)*N1+j>=0||(t+1)*N1+j<maximum*N1)
            lf_test[(t+1)*N1+j] = slf * b[j][O[v*maximum+(t+1)]];
        }
    for(i=0;i<N1;i++)
    {
//      VERIFY((T1[v]-1)*N1+i>=0||(T1[v]-1)*N1+i<maximum*N1);
        if((T1[v]-1)*N1+i>=0||(T1[v]-1)*N1+i<maximum*N1)
        Bt[(T1[v]-1)*N1+i] = 1;
    }
    for(t=T1[v]-2;t>=0;t--)
        for(int i=0;i<N1;i++)
        {
//          VERIFY(t*N1+i>=0||t*N1+i<maximum*N1);
            if(t*N1+i>=0||t*N1+i<maximum*N1)
            Bt[t*N1+i] = 0;
            for(int j=0;j<N1;j++)
            {
//              VERIFY(t*N1+i>=0||t*N1+i<maximum*N1);
//              VERIFY((t+1)*N1+j>=0||(t+1)*N1+j<maximum*N1);
                if(t*N1+i>=0||t*N1+i<maximum*N1)
                    if((t+1)*N1+j>=0||(t+1)*N1+j<maximum*N1)
                    Bt[t*N1+i] += a[i][j]*b[j][O[v*maximum+(t+1)]]*Bt[(t+1)*N1+j];
//              fprintf(errordetect,"Bt[%2d]=%10.7f w=%2d   v=%2d   i=%2d   j=%2d   t=%3d\n",t*N1+i,Bt[t*N1+i],w,v,i,j,t);
            }
        }

}

void HMM::scaling_lf_bt(int w,int v)
{
    for (int i=0;i<(maximum*N1);i++)
        lfps[i] = 0;
    for (i=0;i<maximum;i++)
        cB_test[i] = 0;
    for(i=0;i<T1[v];i++)
    {
        sc[i] = 0;
        for(int j=0;j<N1;j++)
        {
//          VERIFY(i*N1+j>=0||i*N1+j<maximum*N1);
            if(i*N1+j>=0||i*N1+j<maximum*N1)
            sc[i] += lf_test[i*N1+j];
        }
    }
//  VERIFY(sc[0]!=0);
    if(sc[0]!=0)
    c1[0] = 1/sc[0];
    for(i=0;i<N1;i++)
    {
//      VERIFY(i>=0||i<maximum*N1);
        if(i>=0||i<maximum*N1)
        lfps[i] = lf_test[i];
        lfs [i] = (c1 [0] * lf_test[i]);
//      fprintf(errordetect,"lfs[%2d]=%10.7f    v=%2d   w=%2d   i=%2d\n",i,lfs[i],v,w,i);
    }
    for(int t=1;t<T1[v];t++)
    {
        c1[t] = 0;
        for(i=0;i<N1;i++)
        {
            lfps[t*N1+i] = 0;
            for(int j=0;j<N1;j++)
            {
//              VERIFY((t-1)*N1+j>=0||(t-1)*N1+j<maximum*N1);
                if((t-1)*N1+j>=0||(t-1)*N1+j<maximum*N1)
                lfps[t*N1+i] += (lfs[(t-1)*N1+j]*a[j][i]*b[i][O[v*maximum+t]]);
//              fprintf(errordetect,"lfs[%2d]=%10.7e\n",(t-1)*N1+j,lfs[(t-1)*N1+j]);
            }
            c1[t] += lfps[t*N1+i];
        }
//      VERIFY(c1[t]!=0);
        if(c1[t]!=0)
        c1[t] = 1/c1[t];
        for(i=0;i<N1;i++)
        {
//          VERIFY(t*N1+i>=0||t*N1+i<maximum*N1);
            if(t*N1+i>=0||t*N1+i<maximum*N1)
            lfs[t*N1+i] = (c1[t] * lfps[t*N1+i]);
//          fprintf(errordetect,"lfs[%2d]=%10.7f    w=%2d   v=%2d   i=%2d   t=%2d\n",t*N1+i,lfs[t*N1+i],w,v,i,t);
        }
    }
//  VERIFY(T1[v]-1>=0||T1[v]-1<maximum);
    if(T1[v]-1>=0||T1[v]-1<maximum)
    cB_test[T1[v]-1] = c1[T1[v]-1];
    for (t=T1[v]-2;t>=0;t--)
    {
//      VERIFY(t>=0||t<maximum);
//      VERIFY(t+1>=0||t+1<maximum);
        if(t>=0||t<maximum)
            if(t+1>=0||t+1<maximum)

//  ERROR>>     cB_test[ 9] =1.0379005e+305 w= 1    v= 0    t= 9
//              c1[ 8]      =1.0006109e+005 w= 1    v= 0    t= 8
//              cB_test[ 8] =1.#INF000e+000 w= 1    v= 0    t= 8
//  *******  Because more than e+308.....But long double should allow more e+4900!!!!!!!!!
    if ((w==1)&&(v==0)&&(t==9))
```

```
        {
            int o=0;
        }


        cB_test[t]=(cB_test[t+1]*c1[t]);

    }
    for(t=0;t<T1[v];t++)
        for(i=0;i<N1;i++)
        {
//          VERIFY(t>=0||t<maximum);
//          VERIFY(t*N1+i>=0||t*N1+i<maximum*N1);
            if(t>=0||t<maximum)
                if(t*N1+i>=0||t*N1+i<maximum*N1)
                Bts[t*N1+i] = (cB_test[t] * Bt[t*N1+i]);
        }
}

void HMM::find_logP(int w,int v)
{
    plog[v] = 0;
    for(int t=0;t<T1[v];t++)
        plog[v] += log10(c1[t]);
    plog[v]  = -plog[v];
    Plog[w] +=  plog[v];
}

void HMM::find_A_AM_B_BM_prime(int w,int v)
{
    long double q,G;
    for(int i=0;i<N1;i++)
        for(int j=0;j<N1;j++)
        {
            x[i][j] = 0;
            for(int t=0; t<T1[v]-1;t++)
            {
//              VERIFY((t+1)*N1+j>=0||(t+1)*N1+j<maximum*N1);
//              VERIFY(t*N1+i>=0||t*N1+i<maximum*N1);
                if((t+1)*N1+j>=0||(t+1)*N1+j<maximum*N1)
                    if(t*N1+i>=0||t*N1+i<maximum*N1)
                    x[i][j] += lfs[t*N1+i] * a[i][j] * b[j][O[v*maximum+(t+1)]] * Bts[(t+1)*N1+j];
            }
            Aprime[i][j] += x[i][j];
        }
    for(i=0;i<N1;i++)
    {
        z[i] = 0;
        for(int t=0;t<T1[v]-1;t++)
        {
            q = 0;
            for(int j=0;j<N1;j++)
            {
//              VERIFY((t+1)*N1+j>=0||(t+1)*N1+j<maximum*N1);
//              VERIFY(t*N1+i>=0||t*N1+i<maximum*N1);
                if((t+1)*N1+j>=0||(t+1)*N1+j<maximum*N1)
                    if(t*N1+i>=0||t*N1+i<maximum*N1)
                    q += lfs[t*N1+i] * a[i][j] * b[j][O[v*maximum+(t+1)]] * Bts[(t+1)*N1+j];
            }
            z[i] += q;
        }
        AMprime[i] += z[i];
    }
    for(int j=0;j<N1;j++)
        for(int k=0;k<K;k++)
        {
            y[j][k] = 0;
            l1[j]   = 0;
            for(int t=0;t<T1[v]-1;t++)
            {
                G = 0;
                for(i=0;i<N1;i++)
                {
//                  VERIFY((t+1)*N1+i>=0||(t+1)*N1+i<maximum*N1);
//                  VERIFY(t*N1+j>=0||t*N1+j<maximum*N1);
                    if((t+1)*N1+i>=0||(t+1)*N1+i<maximum*N1)
                        if(t*N1+j>=0||t*N1+j<maximum*N1)
                        G += lfs[t*N1+j]*a[j][i]*b[i][O[v*maximum+(t+1)]]*Bts[(t+1)*N1+i];
                }
                if(O[v*maximum+t] == k+1)
                    y[j][k] += G;
                l1[j] += G;
            }
            if(O[v*maximum+(T1[v]-1)] == k+1)
            {
//              VERIFY((T1[v]-1)*N1+j>=0||(T1[v]-1)*N1+j<maximum*N1);
                if((T1[v]-1)*N1+j>=0||(T1[v]-1)*N1+j<maximum*N1)
                y[j][k] += lfs[(T1[v]-1)*N1+j];
            }
//          VERIFY((T1[v]-1)*N1+j>=0||(T1[v]-1)*N1+j<maximum*N1);
            if((T1[v]-1)*N1+j>=0||(T1[v]-1)*N1+j<maximum*N1)
            l1[j] += lfs[(T1[v]-1)*N1+j];
        }
    for(i=0;i<N1;i++)
        for(j=0;j<K;j++)
            Bprime[i][j] += y[i][j];
    for(i=0;i<N1;i++)
        BMprime[i] += l1[i];
}

void HMM::find_new_abvalue(int w)
{
    long double bcomplex;
    for(int i=0;i<N1;i++)
        for(int j=0;j<N1;j++)
```

```
        {
//          VERIFY(AMprime[i]!=0);
            if(AMprime[i]!=0)
            aprime[i][j] = (long double)( Aprime[i][j]/AMprime[i] );
        }
    for(int j=0;j<N1;j++)
        for(int k=0;k<K;k++)
        {
//          VERIFY(BMprime[j]!=0);
            if(BMprime[j]!=0)
            bprime[j][k] = Bprime[j][k]/BMprime[j];
            if(bprime[j][k] < MIN_B)
                bprime[j][k] = MIN_B;
        }
    for (i=0;i<N1;i++)
    {
        bcomplex = 0;
        for (j=0;j<K;j++)
            bcomplex += bprime[i][j];
    }
    for(i=0;i<N1;i++)
    {
        imb[i] = 0;
        for(int k=0;k<K;k++)
            imb[i] += bprime[i][k];
    }
    for(i=0;i<N1;i++)
        for(int k=0;k<K;k++)
        {
//          VERIFY(imb[i]!=0);
            if(imb[i]!=0)
            bprime[i][k] /= imb[i];
        }
    for (i=0;i<N1;i++)
    {
        bcomplex = 0;
        for (j=0;j<K;j++)
            bcomplex += bprime[i][j];
    }
}

void HMM::save_model_file(void)
{
    Buf_aprime  =(float*)calloc(N1*N1,sizeof(float));
    Buf_bprime  =(float*)calloc(N1*K,sizeof(float));
    Buf_Pi      =(float*)calloc(N1,sizeof(float));
    for (int i=0;i<N1;i++)
        for (int j=0;j<N1;j++)
        {
            fprintf(model_text_file,"A [%3d][%3d] = %.40f\n",i,j,aprime[i][j]);
            Buf_aprime[(i*N1)+j]=(float)aprime[i][j];
        }
    fprintf(model_text_file,"\n");
    for (i=0;i<N1;i++)
        for (int j=0;j<K;j++)
        {
            fprintf(model_text_file,"B [%3d][%3d] = %.40f\n",i,j,bprime[i][j]);
            Buf_bprime[(i*K)+j]=(float)bprime[i][j];
        }
    fprintf(model_text_file,"\n");
    for (i=0;i<N1;i++)
    {
        fprintf(model_text_file,"Pi[%d]=%.2f   ",i+1,Pi[i]);
        Buf_Pi[i]=Pi[i];
    }
    fprintf(model_text_file,"\n");
    fwrite(Buf_aprime,4,N1*N1,model_file);
    fwrite(Buf_bprime,4,N1*K,model_file);
    fwrite(Buf_Pi,4,N1,model_file);
    fclose(observation_file);
    fclose(model_file);
    fclose(model_text_file);
    free(Buf_aprime);
    free(Buf_bprime);
    free(Buf_Pi);
}

void HMM::RunHmm(CString codebookindexfilename,CString modelfilename,CString modeltextfilename)
{
    open_file(codebookindexfilename,modelfilename,modeltextfilename);
    allocate_memory();
    random_abvalue();
    int number_of_observation=find_number_of_observation();
    for (int w=0;w<ROUND;w++)
    {
        copy_abprime_to_ab(w);
        for (int v=0;v<number_of_observation;v++)
        {
            find_lf_bt_value(w,v);
            scaling_lf_bt(w,v);
            find_logP(w,v);
            find_A_AM_B_BM_prime(w,v);
        }
        find_new_abvalue(w);
    }
    save_model_file();
}


BEGIN_MESSAGE_MAP(HMM, CView)
    //{{AFX_MSG_MAP(HMM)
        // NOTE - the ClassWizard will add and remove mapping macros here.
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()
```

```
/////////////////////////////////////////////////////////////////////////////
// HMM drawing

void HMM::OnDraw(CDC* pDC)
{
    CDocument* pDoc = GetDocument();
    // TODO: add draw code here
}

/////////////////////////////////////////////////////////////////////////////
// HMM diagnostics

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

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

/////////////////////////////////////////////////////////////////////////////
// HMM message handlers
```

```
// Hmmdia.cpp : implementation file
//

#include "stdafx.h"
#include "art3.h"
#include "Hmmdia.h"
#include "HMM.h"

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

/////////////////////////////////////////////////////////////////////////////
// Hmmdia dialog


Hmmdia::Hmmdia(CWnd* pParent /*=NULL*/)
    : CDialog(Hmmdia::IDD, pParent)
{
    //{{AFX_DATA_INIT(Hmmdia)
    m_output = _T("");
    m_outtext = _T("");
    m_input = _T("");
    //}}AFX_DATA_INIT
}


void Hmmdia::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(Hmmdia)
    DDX_Control(pDX, IDC_LIST1, m_list);
    DDX_Text(pDX, IDC_EDIT1, m_output);
    DDX_Text(pDX, IDC_EDIT2, m_outtext);
    DDX_Text(pDX, IDC_EDIT3, m_input);
    //}}AFX_DATA_MAP
}


BEGIN_MESSAGE_MAP(Hmmdia, CDialog)
    //{{AFX_MSG_MAP(Hmmdia)
    ON_BN_CLICKED(IDC_BUTTON1, OnOPEN)
    ON_LBN_DBLCLK(IDC_LIST1, OnOutmodel)
    ON_BN_CLICKED(IDC_BUTTON2, OnRUN)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// Hmmdia message handlers

void Hmmdia::OnOPEN()
{
    // TODO: Add your control notification handler code here
    CFileFind   filefind;
    char        notfound;

    UpdateData(true);
    m_list.ResetContent();
    filefind.FindFile("*.idx",0);
    for (;notfound!=0;)
    {
        notfound=filefind.FindNextFile();
        m_list.AddString(filefind.GetFileName());
    }
    UpdateData(false);
}

void Hmmdia::OnOutmodel()
{
    UpdateData(true);
    m_list.GetText(m_list.GetCurSel(),m_input);
/*  int pos=m_input.Find('.');
    m_output=m_input;
    //m_outtext.SetAt(pos,'-');
    //m_outtext.SetAt(pos+1,'h');
    //m_outtext.SetAt(pos+2,'m');
    //m_outtext.SetAt(pos+3,'m');
    //m_outtext.SetAt(pos+4,'.');
    m_output.SetAt(pos+1,'h');
    m_output.SetAt(pos+2,'m');
    m_output.SetAt(pos+3,'m');*/
    UpdateData(false);
}

void Hmmdia::OnRUN()
{
    UpdateData(true);
    HMM hmm;
    int pos=m_output.Find('.');
    m_outtext=m_output;
    m_outtext.SetAt(pos+1,'t');
    m_outtext.SetAt(pos+2,'x');
    m_outtext.SetAt(pos+3,'t');

    hmm.RunHmm(m_input,m_output,m_outtext);
    MessageBox(m_output,"HMM Model File..Complete!");
    UpdateData(false);

}
```

```
// List.cpp : implementation file
//

#include "stdafx.h"
#include "art3.h"
#include "List.h"
#include "LPC.h"


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

/////////////////////////////////////////////////////////////////////////////
// List dialog


List::List(CWnd* pParent /*=NULL*/)
    : CDialog(List::IDD, pParent)
{
    //{{AFX_DATA_INIT(List)
    m_list1 = _T("");
    m_list2 = _T("");
    //}}AFX_DATA_INIT
}


void List::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(List)
    DDX_Control(pDX, IDC_LIST1, m_list);
    DDX_Text(pDX, IDC_EDIT1, m_list1);
    DDX_Text(pDX, IDC_EDIT2, m_list2);
    //}}AFX_DATA_MAP
}


BEGIN_MESSAGE_MAP(List, CDialog)
    //{{AFX_MSG_MAP(List)
    ON_BN_CLICKED(IDC_BUTTON1, OnOPEN)
    ON_BN_CLICKED(IDC_BUTTON2, OnRUN)
    ON_LBN_DBLCLK(IDC_LIST1, OnLIST)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// List message handlers
void List::OnLIST()
{
    LPC lpc;
    // TODO: Add your control notification handler code here
    m_list.GetText(m_list.GetCurSel(),m_list1);
    int pos=m_list1.Find('.');
    m_list2=m_list1;
    m_list2.SetAt(pos+1,'l');
    m_list2.SetAt(pos+2,'p');
    m_list2.SetAt(pos+3,'c');
    //lpc.lpc_dialog(m_list1,m_list2);

    //m_list2+="...Complete!";

    UpdateData(false);
}

void List::OnOPEN()
{
    // TODO: Add your control notification handler code here
    CFileFind   filefind;
    char        notfound;

    m_list.ResetContent();
    filefind.FindFile("*.wav",0);
    for (;notfound!=0;)
    {
        notfound=filefind.FindNextFile();
        m_list.AddString(filefind.GetFileName());
    }
    UpdateData(false);
}

void List::OnRUN()
{
    CString m_text;
    LPC lpc;

    for (int index=0;index<m_list.GetCount();index++)
    {
        if (m_list.GetSel(index)!=0)
        {
            m_list.GetText(index,m_list1);
            int pos=m_list1.Find('.');
            m_list2=m_list1;
            m_list2.SetAt(pos+1,'l');
            m_list2.SetAt(pos+2,'p');
            m_list2.SetAt(pos+3,'c');
            m_text=m_list1;
            m_text.SetAt(pos,'-');
            m_text.SetAt(pos+1,'L');
            m_text.SetAt(pos+2,'P');
            m_text.SetAt(pos+3,'C');
            m_text+=".txt";
            lpc.lpc_dialog(m_list1,m_list2,m_text);
```

```
//          m_list2+="...Complete!";
            UpdateData(false);
        }
    }
AfxMessageBox("LPC File Complete...!");
}
```

```
// LPC.cpp : implementation file
//

#include "stdafx.h"
#include "art3.h"
#include "LPC.h"
#include "math.h"

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


/////////////////////////////////////////////////////////////////////////////
// LPC

//IMPLEMENT_DYNCREATE(LPC, CView)

LPC::LPC()
{
    input_wav_file=NULL;
    parameter_file=NULL;
}

LPC::~LPC()
{
}


//****************************** PROCEDURE LPC ******************************
bool LPC::open_all_file(CString filename,CString lpcfilename,CString textfilename)
{
    if ((input_wav_file = fopen(filename,"rb")) == NULL)
    {
        MessageBox(filename,"Can't open file", MB_OK);
        return false;
    }
    if ((parameter_file = fopen(lpcfilename,"wb")) == NULL)
    {
         MessageBox(lpcfilename, "Can't create file", MB_OK);
         return false;
    }

    if ((text_file = fopen(textfilename,"wt")) == NULL)
    {
        MessageBox(lpcfilename, "Can't create file", MB_OK);
        return false;
    }
    return true;
}


int LPC::prepare_data(void)
{
   long length = -1;
   while (!feof(input_wav_file))
   {
      int c = fgetc(input_wav_file);
      length++;
   }
   rewind(input_wav_file);
   length -= 44L;
   fseek(input_wav_file,44L,SEEK_CUR);
   int number_of_frame = (length-FRAME_LENGTH+SHIFT_LENGTH)/SHIFT_LENGTH;
   signal[0] = (int)fgetc(input_wav_file);
   return number_of_frame;
}

void LPC::find_preemphasis_and_window(void)
{
    static int called = FIRST_CALL;
    if (called == FIRST_CALL)
    {
        for ( int fl=1;fl<=FRAME_LENGTH;fl++ )
        {
            signal     [fl] = (int)fgetc(input_wav_file);
            preemphasis[fl] = (float)((signal[fl] -  ADAPTIVE *(float)signal[fl-1]));
            windowed   [fl] = (float)(preemphasis[fl]*(0.54-0.46*cos(2*PI*(fl-1)/(FRAME_LENGTH-1))));
        }
        called = NEXT_CALL;
    }
    else
    {
        for (int fl=1;fl<=(FRAME_LENGTH - SHIFT_LENGTH);fl++)
        {
            signal     [fl] = signal[fl+SHIFT_LENGTH];
            preemphasis[fl] = (float)(preemphasis[fl+SHIFT_LENGTH]);
            windowed   [fl] = (float)(preemphasis[fl]*(0.54-0.46*cos(2*PI*(fl-1)/(FRAME_LENGTH-1))));
        }
        for (fl=(FRAME_LENGTH - SHIFT_LENGTH+1);fl<=FRAME_LENGTH;fl++)
        {
            signal     [fl] = (int)fgetc(input_wav_file);
            preemphasis[fl] = (float)(signal[fl] - ADAPTIVE * (float)signal[fl-1]);
            windowed   [fl] = (float)(preemphasis[fl]*(0.54-0.46*cos(2*PI*(fl-1)/(FRAME_LENGTH-1))));
        }
    }
}

void LPC::find_autocorrelation(void)
{
    float  matrix_temp[P][P];
```

```
    for (int k=0;k<=P;k++)
    {
        R[k] = 0;
        for (int m=0;m<=FRAME_LENGTH-1-k;m++)
            R[k] += windowed[m+1] * windowed[m+k+1];
    }
    for (int i=0;i<P;i++)
    {
        int m = 0;
        for (k=i;k<P;k++)
        {
            A[i][k] = R[m++];
            A[k][i] = A[i][k];
        }
    }
    for (i=0;i<P;i++)
        for (int j=0;j<P;j++)
            matrix_temp[i][j] = A[i][j];

    for (i=0;i<P;i++)
        for (int j=0;j<P;j++)
            if (i==j) X[i][j] = 1;
            else X[i][j]=0;

    for (k=0;k<P;k++)
    {
        int maxrow = k;
        float maxvalue = (float)fabs(matrix_temp[k][k]);

        for (i=k+1;i<P;i++)
        {
            if (maxvalue < fabs(matrix_temp[i][k]))
            {
                maxvalue = (float)fabs(matrix_temp[i][k]);
                maxrow   = i;
            }
        }
        if (maxrow != k)
        {
            for (int j=0;j<P;j++)
            {
                float temp              =  matrix_temp[k][j];
                matrix_temp[k][j]       =  matrix_temp[maxrow][j];
                matrix_temp[maxrow][j] =  temp;
            }
            for (j=0;j<P;j++)
            {
                float temp          =  X[k][j];
                X[k][j]       =  X[maxrow][j];
                X[maxrow][j] =  temp;
            }
        }
        float akk = matrix_temp[k][k];
        for (int j=k;j<P;j++)
            matrix_temp[k][j] /= akk;
        for (j=0;j<P;j++)
            X[k][j] /= akk;
        akk = matrix_temp[k][k];
        for (int l=0;l<P;l++)
        {
            float alk = matrix_temp[l][k];
            if (k != l)
            {
                for (int m=k;m<P;m++)
                    matrix_temp[l][m]   -= matrix_temp[k][m]/akk*alk;
                for (m=0;m<P;m++)
                    X[l][m] -= X[k][m]/akk*alk;
            }
        }
    }
    for (i=1;i<=P;i++)
        phi[i] = R[i];

}

void LPC::find_coefficient(void)
{
    for (int i=0;i<P;i++)
    {
        alpha[i+1] = 0;
        for (int k=0;k<P;k++)
        {
            alpha[i+1] += X[i][k]*phi[k+1];
        }
    }
}

void LPC::find_gain(void)
{
    float q = (float)R[0];
    for (int i=1;i<=P;i++)
    {
        q -= (float)((float)alpha[i]*(float)R[i]);
    }
    gain = (float)sqrt(q);
}

void LPC::find_cepstrum(void)
{
    cepstrum[0] = (float)log(gain);
    for (int m=1;m<N;m++)
    {
        if (m<=P)   cepstrum[m] = alpha[m];
        else        cepstrum[m] = 0;
```

```
        for (int k=1;k<=m-1;k++)
            cepstrum[m] += ((float)k/(float)m) * cepstrum[k] * alpha[m-k];

    }
}

void LPC::find_weight(void)
{
    for (int m=0;m<N;m++)
        weight[m] = (float)(cepstrum[m] * (1+((N)/2)*sin(PI*(float)m/(N))));
}


void LPC::write_parameter_file(void)
{
   for (int i=0;i<N;i++)
   {
    fwrite(&weight[i],sizeof(weight[i]),1,parameter_file);
    fprintf(text_file,"%13.10f\n",weight[i]);
    }
}

void LPC::close_all_file(void)
    {
    fclose(input_wav_file);
    fclose(parameter_file);
    fclose(text_file);
    }
//*****************************RUNLPC***************
bool LPC::lpc_dialog(CString inputfilename,CString lpcfilename,CString textfilename)
{
    LPC lpc;
    lpc.open_all_file(inputfilename,lpcfilename,textfilename);
    int number_of_frame=lpc.prepare_data();
    for (int nf=0;nf<number_of_frame;nf++)
    {
        lpc.find_preemphasis_and_window();
        lpc.find_autocorrelation();
        lpc.find_coefficient();
        lpc.find_gain();
        lpc.find_cepstrum();
        lpc.find_weight();
        lpc.write_parameter_file();
    }
    lpc.close_all_file();

    return true;
}

//*************************COMBINE***********
bool LPC::Combine(CString inputfilename,CString outputfilename)
{
    float   *tmp;
    float   *buf;
    int     item=0;

    if ((outputfile = fopen(outputfilename,"ab+")) == NULL)
    {
        MessageBox(outputfilename,"Can't append file", MB_OK);
        return false;
    }
    if ((inputfile=fopen(inputfilename,"rb"))==NULL)
    {
        MessageBox(inputfilename, "Can't open file!", MB_OK);
        return false;
    }
    tmp=(float *)calloc(1,sizeof(float));
    if (tmp == NULL)
    {
        MessageBox("Can't allocate memory for Temp!", "ERROR!..", MB_OK);
        return false;
    }
    rewind(inputfile);
    fseek(inputfile,1,SEEK_CUR);
    while (!feof(inputfile))
    {
        fread(tmp,sizeof(float),1,inputfile);
        item++;
    }
    rewind(inputfile);
    buf=(float *)calloc(item,sizeof(float));
    if (buf == NULL)
    {
        MessageBox("Can't allocate memory for Buf!", "ERROR!..", MB_OK);
        return false;
    }

    fread(buf,sizeof(float),item,inputfile);
    fwrite(buf,sizeof(float),item,outputfile);
    free(buf);
    free(tmp);
    fclose(inputfile);
    fclose(outputfile);

    return true;
}

//************************************message*********

BEGIN_MESSAGE_MAP(LPC, CView)
    //{{AFX_MSG_MAP(LPC)
        // NOTE - the ClassWizard will add and remove mapping macros here.
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()
```

```
////////////////////////////////////////////////////////////////////////////
// LPC drawing

void LPC::OnDraw(CDC* pDC)
{
    CDocument* pDoc = GetDocument();
    // TODO: add draw code here
}

////////////////////////////////////////////////////////////////////////////
// LPC diagnostics

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

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

////////////////////////////////////////////////////////////////////////////
// LPC message handlers
```

```
// ombinedia.cpp : implementation file
//

#include "stdafx.h"
#include "art3.h"
#include "ombinedia.h"
#include "LPC.h"

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

/////////////////////////////////////////////////////////////////////////////
// Combinedia dialog


Combinedia::Combinedia(CWnd* pParent /*=NULL*/)
    : CDialog(Combinedia::IDD, pParent)
{
    //{{AFX_DATA_INIT(Combinedia)
    m_list1 = _T("");
    m_list2 = _T("");
    //}}AFX_DATA_INIT
}


void Combinedia::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(Combinedia)
    DDX_Control(pDX, IDC_LIST1, m_list);
    DDX_Text(pDX, IDC_EDIT1, m_list1);
    DDX_Text(pDX, IDC_EDIT2, m_list2);
    //}}AFX_DATA_MAP
}


BEGIN_MESSAGE_MAP(Combinedia, CDialog)
    //{{AFX_MSG_MAP(Combinedia)
    ON_BN_CLICKED(IDC_BUTTON1, OnOpen)
    ON_BN_CLICKED(IDC_BUTTON2, OnRun)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// Combinedia message handlers

void Combinedia::OnOpen()
{
    // TODO: Add your control notification handler code here
    CFileFind   filefind;
    char        notfound;

    m_list.ResetContent();
    filefind.FindFile("*.lpc",0);
    for (;notfound!=0;)
    {
        notfound=filefind.FindNextFile();
        m_list.AddString(filefind.GetFileName());
    }
    UpdateData(false);
}

void Combinedia::OnRun()
{
    // TODO: Add your control notification handler code here
    LPC lpc;
    UpdateData(true);

    for (int index=0;index<m_list.GetCount();index++)
    {
        if (m_list.GetSel(index)!=0)
        {
            m_list.GetText(index,m_list1);
            m_list2="AllSample.lpc";
            lpc.Combine(m_list1,m_list2);
        //  m_list1="Combine all wav file...Complete";
        //  m_list2+="...Complete!";

            UpdateData(false);
        }
    }
AfxMessageBox("AllSample.lpc...Complete!");
}
```

```
// Port.cpp : implementation file
//

#include "stdafx.h"
#include "art3.h"
#include "Port.h"
#include <conio.h>
//#include "Ports.h"
#include <math.h>

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

#define  controlport            0x303
#define  port_a                 0x300
#define  port_b                 0x301
/////////////////////////////////////////////////////////////////////////////
// Port dialog


Port::Port(CWnd* pParent /*=NULL*/)
    : CDialog(Port::IDD, pParent)
{
    //{{AFX_DATA_INIT(Port)
        // NOTE: the ClassWizard will add member initialization here
    //}}AFX_DATA_INIT
}


void Port::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(Port)
        // NOTE: the ClassWizard will add DDX and DDV calls here
    //}}AFX_DATA_MAP
}


BEGIN_MESSAGE_MAP(Port, CDialog)
    //{{AFX_MSG_MAP(Port)
    ON_BN_CLICKED(IDC_BUTTON1, OnRun0)
    ON_BN_CLICKED(IDC_BUTTON2, OnRun1)
    ON_BN_CLICKED(IDC_BUTTON10, OnRun9)
    ON_BN_CLICKED(IDC_BUTTON3, OnRun2)
    ON_BN_CLICKED(IDC_BUTTON4, OnRun3)
    ON_BN_CLICKED(IDC_BUTTON5, OnRun4)
    ON_BN_CLICKED(IDC_BUTTON6, OnRun5)
    ON_BN_CLICKED(IDC_BUTTON7, OnRun6)
    ON_BN_CLICKED(IDC_BUTTON8, OnRun7)
    ON_BN_CLICKED(IDC_BUTTON9, OnRun8)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// Port message handlers

void outputport(int portno)
    {
    int value;
      double    value1=2,value2;

      if (portno<=8)
        {
        value2=portno-1;
        value=(int)pow(value1,value2);
    //  _outp(controlport, 128);
    //  _outp(port_a,value);
        }
      else
        {
         value2=portno-9;
        value=(int)pow(value1,value2);
    //  _outp(controlport, 128);
    //  _outp(port_b,value);
        }
   }

void Port::OnRun0()
{
    //outportb(controlport, 0x80);
    //outportb(port_a,0x01);
    _outp(controlport, 128);
    _outp(port_a,1);
    //MessageBox("Port1 is swicthed","PORT");
}

void Port::OnRun1()
{
    //_outp(controlport, 128);
    //_outp(port_a,2);
    MessageBox("Port2 is swicthed","PORT");
}


void Port::OnRun2()
{
    //_outp(controlport, 128);
    //_outp(port_a,4);
    MessageBox("Port3 is swicthed","PORT");
}
void Port::OnRun3()
```

```
{
    //_outp(controlport, 128);
    //_outp(port_a,8);
    MessageBox("Port4 is swicthed","PORT");
}

void Port::OnRun4()
{
    //_outp(controlport, 128);
    //_outp(port_a,16);
    MessageBox("Port5 is swicthed","PORT");
}

void Port::OnRun5()
{
    //_outp(controlport, 128);
    //_outp(port_a,32);
    MessageBox("Port6 is swicthed","PORT");
}

void Port::OnRun6()
{
    //_outp(controlport, 128);
    //_outp(port_a,64);
    MessageBox("Port7 is swicthed","PORT");
}

void Port::OnRun7()
{
    //_outp(controlport, 128);
    //_outp(port_a,128);
    MessageBox("Port8 is swicthed","PORT");
}

void Port::OnRun8()
{
    //_outp(controlport, 128);
    //_outp(port_b,1);
    MessageBox("Port9 is swicthed","PORT");
}

void Port::OnRun9()
{
    //_outp(controlport, 128);
    //_outp(port_b,2);
    MessageBox("Port10 is swicthed","PORT");
}
```

```
// Recog.cpp : implementation file
//

#include "stdafx.h"
#include "art3.h"
#include "Recog.h"
#include "LPC.h"
#include "Vqcom.h"

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

/////////////////////////////////////////////////////////////////////////////
// Recog

//IMPLEMENT_DYNCREATE(Recog, CView)

Recog::Recog()
{
    O==NULL;
    dt==NULL;
    ar==NULL;
    q_st==NULL;
    p_st==NULL;
    Buf_aprime==NULL;
    Buf_bprime==NULL;
}

Recog::~Recog()
{
}

//************************************Recog******

bool Recog::load_unknown_word_file(void)
{


    char temp;
    if ((unknown_word_file = fopen("Temp.idx","rb")) == NULL)
    {
      MessageBox("Temp.idx", "Can't open file", MB_OK);
      return false;
    }
    fread(&T,1,1,unknown_word_file);
    fread(&temp,1,1,unknown_word_file);
    if (temp != -1)
    {
      MessageBox("Invalid input file!", "ERROR", MB_OK);
      return false;
    }
// PROBLEM WARNING Can't allocate memmory//
    O = (char *) calloc((int)T,sizeof(char));
    if (O == NULL)
    {
      MessageBox("Can't allocate memory for O", "ERROR", MB_OK);
      return false;
    }
    fread(O,1,(int)T,unknown_word_file);
    fclose(unknown_word_file);
    return true;
}
bool Recog::allocate_memory(void)
{
    int num_of_model=numberofmodel;
    model_file_name = new CString[num_of_model];
    model_file_name[0]="0.hmm";
    model_file_name[1]="1.hmm";
    model_file_name[2]="2.hmm";
    model_file_name[3]="3.hmm";
    model_file_name[4]="4.hmm";
    model_file_name[5]="5.hmm";
    model_file_name[6]="6.hmm";
    model_file_name[7]="7.hmm";
    model_file_name[8]="8.hmm";
    model_file_name[9]="9.hmm";
    //model_file_name[10]="open.hmm";
    //model_file_name[11]="close.hmm";


    dt   = (double *) calloc((int)T*N1      ,sizeof(double));
    ar   = (char   *) calloc((int)T*N1      ,sizeof(char));
    q_st = (char   *) calloc((int)T         ,sizeof(char));
    p_st = (double *) calloc(num_of_model   ,sizeof(double));

    if ( (dt||ar||q_st||p_st) == NULL)
        {
        MessageBox("Can't allocate memory for dt,ar,q_st,p_st", "ERROR", MB_OK);
        return false;
        }
    return true;
}
bool Recog::load_model(int model_name)
{
    //float *Buf_aprime;
    //float *Buf_bprime;
    //Buf_aprime==NULL;
    //Buf_bprime==NULL;
    if ((model_file = fopen(model_file_name[model_name],"rb")) == NULL)
    {
        MessageBox(model_file_name[model_name], "Can't open file", MB_OK);
```

```
        return false;
    }
    Buf_aprime=(float*)calloc(N1*N1,sizeof(float));
    Buf_bprime=(float*)calloc(N1*K,sizeof(float));
    if ( (Buf_aprime) == NULL)
        {
        MessageBox("Can't allocate memory for Buf_aprime", "ERROR", MB_OK);
        return false;
        }
    if ( (Buf_bprime) == NULL)
        {
        MessageBox("Can't allocate memory for Buf_bprime", "ERROR", MB_OK);
        return false;
        }
    fread(Buf_aprime,4,N1*N1,model_file);
    fread(Buf_bprime,4,N1*K,model_file);
    fclose(model_file);
    for (int i=0;i<N1;i++)
        for (int j=0;j<N1;j++)
            a[i][j]=Buf_aprime[(i*N1)+j];
    for (i=0;i<N1;i++)
        for (int j=0;j<K;j++)
            b[i][j]=Buf_bprime[(i*K)+j];
    //  free(Buf_aprime);
    //  free(Buf_bprime);
    //if(Buf_aprime!=NULL) free(Buf_aprime); Buf_aprime==NULL;
    //if(Buf_bprime!=NULL) free(Buf_bprime); Buf_bprime==NULL;

    return true;
}
void Recog::viterbi(int model)
{
    int Pi[N1];
    Pi[0]=1;Pi[1]=0;Pi[2]=0;Pi[3]=0;Pi[4]=0;Pi[5]=0;
    for (int i=0;i<N1;i++)
    {
        dt[0*N1+i] = (float)Pi[i] * (float)b[i][O[0]-1];
        ar[0*N1+i] = 0;
    }
    for (int t=1;t<(int)T;t++)
    {
        for (int j=0;j<N1;j++)
        {
            for (i=0;i<N1;i++)  d1[i] = dt[(t-1)*N1+i] * (float)a[i][j];
                dmax = 0;
            max_index = 0;
            for (int k=0;k<N1;k++)
            {
                if (dmax < d1[k])
                {
                    dmax = d1[k];
                    max_index = k;
                }
            }
            dt[t*N1+j] = dmax * (float)b[j][O[t]-1];
            ar[t*N1+j] = max_index;
        }
    }
    dmax      = 0;
    max_index = 0;
    for (int k=0;k<N1;k++)
        if (dmax < dt[(int)(T-1)*N1+k])
        {
            dmax = dt[(int)(T-1)*N1+k];
            max_index = k;
        }
    p_st[model] = dmax;
    q_st[(int)T-1]   = max_index;
    for (t=(int)T-2;t>=0;t--)
      q_st[t] = ar[(t+1)*N1+(q_st[t+1])];
}
int Recog::display_recognized_word(void)
{
    dmax      = 0;
    max_index = 0;
    int num_of_model=numberofmodel;
    for (int k=0;k<num_of_model;k++)
        if (dmax < p_st[k])
        {
            dmax = p_st[k];
            max_index = k;
        }
    return max_index;
}

void Recog::free_mem(void)
{
    /*free(dt);
    free(ar);
    free(q_st);
    free(p_st);
    free(O);*/
    if(dt!=NULL) free(dt); dt==NULL;
    if(O!=NULL) free(O); O==NULL;
    if(ar!=NULL) free(ar); ar==NULL;
    if(q_st!=NULL); free(q_st); q_st==NULL;
    if(p_st!=NULL); free(p_st); p_st==NULL;
    if(Buf_aprime!=NULL) free(Buf_aprime); Buf_aprime==NULL;
    if(Buf_bprime!=NULL) free(Buf_bprime); Buf_bprime==NULL;

    delete []model_file_name;
}

int Recog::auto_recog(CString input)
```

```
{
//********************************** LPC ***********************************
    LPC lpc;
    Recog recog;

    lpc.open_all_file(input,"Temp.lpc","Temp.txt");
    int number_of_frame=lpc.prepare_data();
    for (int nf=0;nf<number_of_frame;nf++)
    {
        lpc.find_preemphasis_and_window();
        lpc.find_autocorrelation();
        lpc.find_coefficient();
        lpc.find_gain();
        lpc.find_cepstrum();
        lpc.find_weight();
        lpc.write_parameter_file();
    }
    lpc.close_all_file();
//************************************ VQ **********************************
    Vqcom   vqcmp;

    vqcmp.open_codebook_and_output_file("Codebook.vq","Temp.idx","Temp-IDX.txt");
    vqcmp.allocate_codebook_memory();
    vqcmp.read_codebook_file();
    vqcmp.get_number_of_input_file_and_set_codebook_index_file_pointer(1);
    vqcmp.open_input_vector_file("Temp.lpc");
    vqcmp.allocate_input_vector_and_min_index_memory();
    vqcmp.read_input_vector();
    vqcmp.find_codebook_index();
    vqcmp.write_codebook_index_file(0);
    vqcmp.free_input_vector_and_min_index_memory();
    vqcmp.close_input_vector_file();
    vqcmp.write_codebook_index_text_file("Temp.idx");
    vqcmp.free_codebook_memory();
    vqcmp.close_all_vq_file();
//********************************** RECOG *********************************
    recog.load_unknown_word_file();
    recog.allocate_memory();
    int num_of_model=numberofmodel;
    for (int model=0;model<num_of_model;model++)
    {
        recog.load_model(model);
        recog.viterbi(model);
    }
    int result=recog.display_recognized_word();
    recog.free_mem();
    return result;
}


BEGIN_MESSAGE_MAP(Recog, CView)
    //{{AFX_MSG_MAP(Recog)
        // NOTE - the ClassWizard will add and remove mapping macros here.
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// Recog drawing

void Recog::OnDraw(CDC* pDC)
{
    CDocument* pDoc = GetDocument();
    // TODO: add draw code here
}

/////////////////////////////////////////////////////////////////////////////
// Recog diagnostics

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

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

/////////////////////////////////////////////////////////////////////////////
// Recog message handlers
```

```
// Recogdia.cpp : implementation file
//

#include "stdafx.h"
#include "art3.h"
#include "Recogdia.h"
#include "Recog.h"

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

/////////////////////////////////////////////////////////////////////////////
// Recogdia dialog


Recogdia::Recogdia(CWnd* pParent /*=NULL*/)
    : CDialog(Recogdia::IDD, pParent)
{
    //{{AFX_DATA_INIT(Recogdia)
    m_input = _T("");
    //}}AFX_DATA_INIT
}


void Recogdia::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(Recogdia)
    DDX_Control(pDX, IDC_LIST1, m_list);
    DDX_Text(pDX, IDC_EDIT1, m_input);
    //}}AFX_DATA_MAP
}


BEGIN_MESSAGE_MAP(Recogdia, CDialog)
    //{{AFX_MSG_MAP(Recogdia)
    ON_BN_CLICKED(IDC_BUTTON1, OnShow)
    ON_BN_CLICKED(IDC_BUTTON2, OnRUN)
    ON_LBN_DBLCLK(IDC_LIST1, OnInput)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// Recogdia message handlers

void Recogdia::OnShow()
{
    // TODO: Add your control notification handler code here
    CFileFind   filefind;
    char        notfound;

    m_list.ResetContent();
    filefind.FindFile("*.wav",0);
    for (;notfound!=0;)
    {
        notfound=filefind.FindNextFile();
        m_list.AddString(filefind.GetFileName());
    }
    UpdateData(false);
}

void Recogdia::OnInput()
{
    UpdateData(true);
    m_list.GetText(m_list.GetCurSel(),m_input);
    UpdateData(false);

}

void Recogdia::OnRUN()
{
    UpdateData(true);
    Recog recog;
    int result=recog.auto_recog(m_input);
        CString temp;
        temp.Format("%3d",result);
        MessageBox(temp,"Recognized as Model");

    UpdateData(false);

}
```

```
// Segandrecog.cpp : implementation file
//

#include "stdafx.h"
#include "art3.h"
#include "Segandrecog.h"
#include "Segment.h"

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

/////////////////////////////////////////////////////////////////////////////
// Segandrecog dialog


Segandrecog::Segandrecog(CWnd* pParent /*=NULL*/)
    : CDialog(Segandrecog::IDD, pParent)
{
    //{{AFX_DATA_INIT(Segandrecog)
    m_input = _T("");
    //}}AFX_DATA_INIT
}


void Segandrecog::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(Segandrecog)
    DDX_Control(pDX, IDC_LIST1, m_list);
    DDX_Text(pDX, IDC_EDIT1, m_input);
    //}}AFX_DATA_MAP
}


BEGIN_MESSAGE_MAP(Segandrecog, CDialog)
    //{{AFX_MSG_MAP(Segandrecog)
    ON_BN_CLICKED(IDC_BUTTON1, OnShowList)
    ON_BN_CLICKED(IDC_BUTTON2, OnRun)
    ON_LBN_DBLCLK(IDC_LIST1, OnAddinput)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// Segandrecog message handlers

void Segandrecog::OnShowList()
{
    CFileFind   filefind;
    char        notfound;

    m_list.ResetContent();
    filefind.FindFile("*.wav",0);
    for (;notfound!=0;)
    {
        notfound=filefind.FindNextFile();
        m_list.AddString(filefind.GetFileName());
    }
    UpdateData(false);
}

void Segandrecog::OnAddinput()
{
    UpdateData(true);
    m_list.GetText(m_list.GetCurSel(),m_input);
    UpdateData(false);
}

void Segandrecog::OnRun()
{
    Segment segment;
    segment.energy(m_input,"temp.eng"
                        ,"temp_eng.txt"
                        ,"temp_wav.txt"
                        ,50
                        ,300);

    segment.segmentEnergy("temp.eng"
                          ,"temp.seg"
                          ,"temp_seg.txt"
                          ,"summary.txt"
                          ,50
                          ,50
                          ,20
                          ,1);
    segment.editrecog(m_input,"temp.seg","result.txt");
    MessageBox("Segment&Recognize","Complete!");
    UpdateData(false);
}
```

```
// Segment.cpp : implementation file
//

#include "stdafx.h"
#include "art3.h"
#include "Segment.h"
#include "Recog.h"


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

/////////////////////////////////////////////////////////////////////////////
// Segment

//IMPLEMENT_DYNCREATE(Segment, CView)

Segment::Segment()
{

}

Segment::~Segment()
{
}


//*************************** PROCEDURE ENERGY  ******************************
bool Segment::energy(CString namewav
                    ,CString nameeng
                    ,CString nameengtxt
                    ,CString namewavtxt
                    ,int sampleshift
                    ,int samplewindow)
{
    FILE        *inputfile;
    FILE        *textfile2;
    FILE        *energyfile;
    FILE        *wavtextfile;
    char        *sample;
    long        *sqr;
    float       *sum_sqr;
    div_t       frame;
    sample==NULL;

    if ((inputfile=fopen(namewav,"rb"))== NULL)
    {
        MessageBox(namewav,"Can't open input wave file.", MB_OK);
        return false;
    }
    if ((energyfile=fopen(nameeng,"wb"))== NULL)
    {
        MessageBox(nameeng,"Can't create output energy file.", MB_OK);
        return false;
    }

        if ((textfile2=fopen(nameengtxt,"wt"))== NULL)
        {
            MessageBox(nameengtxt,"Can't create output text file.", MB_OK);
            return false;
        }

        if ((wavtextfile=fopen(namewavtxt,"wt"))== NULL)
        {
            MessageBox(namewavtxt,"Can't create output text file.", MB_OK);
            return false;
        }
    int L=0;
    fseek(inputfile,44L,SEEK_CUR);
    while (!feof(inputfile))
    {
        char d = fgetc(inputfile);
        L++;
    }

        L-=120;
    sample=(char*)calloc(L,sizeof(int));
    if (sample==NULL)
    {
        MessageBox("Can't allocate memory for sample.", "ERROR!", MB_OK);
        return false;
    }
    sqr=(long*)calloc(L,sizeof(long));
    if (sqr==NULL)
    {
        MessageBox("Can't allocate memory for sqr.", "ERROR!", MB_OK);
        return false;
    }
    rewind(inputfile);
    fseek(inputfile,44L,SEEK_CUR);
    fread(sample,L,sizeof(char),inputfile);
    for (int i=0;i<L;i++)
        {
        sample[i]=sample[i]-128;
        fprintf(wavtextfile,"%3d\n",sample[i]);
        }
    for (i=0;i<L;i++)
        sqr[i]=sample[i]*sample[i];
    frame=div(L,sampleshift);
    sum_sqr=(float*)calloc(frame.quot,sizeof(float));
    if (sum_sqr==NULL)
    {
```

```
        MessageBox("Can't allocate memory for sum_sqr.", "ERROR!", MB_OK);
        return false;
    }
    for (i=0;i<frame.quot-(samplewindow/sampleshift);i++)
    {
        sum_sqr[i]=0;
        for (int no=0;no<samplewindow;no++)
            sum_sqr[i]+=sqr[(i*sampleshift)+no];
        sum_sqr[i]/=samplewindow;
    }
    for (int no=0;no<samplewindow-sampleshift+frame.rem;no++)
        sum_sqr[frame.quot-(samplewindow/sampleshift)]+=sqr[((frame.quot-(samplewindow/sampleshift))*sampleshift)+no];
    sum_sqr[frame.quot-(samplewindow/sampleshift)]/=((samplewindow-sampleshift)+frame.rem);
    fwrite(sum_sqr,frame.quot-(samplewindow/sampleshift)+1,sizeof(float),energyfile);

        for (i=0;i<frame.quot-(samplewindow/sampleshift)+1;i++)
            fprintf(textfile2,"\n%13.10f",sum_sqr[i]);

    fclose(inputfile);
    fclose(textfile2);
    fclose(energyfile);
    fclose(wavtextfile);
    free(sample);
    free(sqr);
    free(sum_sqr);
    return true;
}


//************************ PROCEDURE SEGMENTENERGY *************************
bool Segment:: segmentEnergy(CString energyname
                            ,CString segmentname
                            ,CString segmenttextname
                            ,CString summary
                            ,int lowpercent
                            ,int mininterval
                            ,int minnumber
                            ,int setno)
{
    FILE    *energyfile;
    FILE    *segmentfile;
    FILE    *segmenttextfile;
    FILE    *sumsegmentfile;

    int     *segmentBuf;
    int     *resegment;
    float   *energy1;
    div_t   segmentnumber;

    if ((energyfile=fopen(energyname,"rb"))==NULL)
    {
        MessageBox(energyname,"Can't open file.", MB_OK);
        return false;
    }
    if ((segmentfile=fopen(segmentname,"wb"))==NULL)
    {
        MessageBox(segmentname,"Can't create file.", MB_OK);
        return false;
    }

        if ((segmenttextfile=fopen(segmenttextname,"wt"))==NULL)
        {
            MessageBox(segmenttextname,"Can't open file.", MB_OK);
            return false;
        }

        if ((sumsegmentfile=fopen(summary,"wt"))==NULL)
        {
            MessageBox(summary,"Can't create file.", MB_OK);
            return false;
        }

    int frame1=-1;
    while (!feof(energyfile))
    {
        char c=fgetc(energyfile);
        frame1++;
    }
    energy1=(float*)calloc((frame1/4)+4,sizeof(float));
    if (energy1==NULL)
    {
        MessageBox("Can't allocate memory for energy1.", "ERROR!", MB_OK);
        return false;
    }
    segmentBuf=(int*)calloc(MAXSEGMENT,sizeof(int));
    if (segmentBuf==NULL)
    {
        MessageBox("Can't allocate memory for segmentBuf.", "ERROR!", MB_OK);
        return false;
    }
    rewind(energyfile);
    fread(energy1,sizeof(float),frame1/4,energyfile);
    int averageenergy=0;
    for (int i=0;i<frame1/4;i++)
        averageenergy+=(int)energy1[i];
    if(frame1!=0)
        {
        averageenergy=averageenergy/(frame1/4);
        }
    int segmentlevel=(int)(averageenergy*lowpercent*0.01);
    int tempstart=-1;
    int tempstop=-1;
    int oldtempstop=0-mininterval-1;
    energy1[(frame1/4)+1]=0;
    energy1[(frame1/4)+2]=0;
```

```
int segment=0;
for (i=0;i<frame1/4;i++)
{
    if (i==0)
    {
        if ((energy1[i]>segmentlevel)&&(energy1[i+1]>segmentlevel)&&(energy1[i+2]>segmentlevel))
            tempstart=i;
        if ((tempstart!=-1)&&(tempstop!=-1)&&((tempstop-tempstart)>minnumber)&&((tempstart-oldtempstop)>mininterval))
        {
            segment++;
            oldtempstop=tempstop;
            segmentBuf[(segment-1)*2]=tempstart;
            segmentBuf[((segment-1)*2)+1]=tempstop;
            tempstart=-1;
            tempstop=-1;
        }
    }
    else
    {
        if ((energy1[i-1]<segmentlevel)&&(energy1[i]>segmentlevel)&&(energy1[i+1]>segmentlevel)&&(energy1[i+2]>segment
level))
            tempstart=i;
        if ((energy1[i-2]>segmentlevel)&&(energy1[i-1]>segmentlevel)&&(energy1[i]>segmentlevel)&&(energy1[i+1]<segment
level))
            tempstop=i;
        if ((tempstart!=-1)&&(tempstop!=-1)&&((tempstop-tempstart)>minnumber)&&((tempstart-oldtempstop)>mininterval))
        {
            segment++;
            oldtempstop=tempstop;
            segmentBuf[(segment-1)*2]=tempstart;
            segmentBuf[((segment-1)*2)+1]=tempstop;
            tempstart=-1;
            tempstop=-1;
        }
    }
}
if(setno=2)
    {
    if((segment==0)||(segmentlevel<50))
        {
        fclose(segmentfile);
        return false;
        }
    }
int sumnoise=0;
int sumlength=0;
for (i=0;i<frame1/4;i++)
{
    if (energy1[i]<segmentlevel)
    {
        sumnoise+=(int)energy1[i];
        sumlength++;
    }
}
if (sumlength==0)
{
    MessageBox("Wave form is too bad!", "ERROR!", MB_OK);
    return false;
}
if(setno=1)
    {
    if (segment==0) return false;
    }
int averagenoise=sumnoise/sumlength;
    resegment=(int*)calloc(segment*2,sizeof(int));
if (resegment==NULL)
{
    MessageBox("Can't allocate memory for resegment.", "ERROR!", MB_OK);
    return false;
}
energy1[frame1/4]=(float)averagenoise-1;
if(setno=2)
    {
    if((averagenoise/averageenergy)>0.5)
        {
        fclose(segmentfile);
        return false;
        }
    }
for (int j=0;j<segment*2;j++)
{
    if (j==0)
    {
        segmentnumber=div(j,2);
        if (segmentnumber.rem==0)
            for (int k=segmentBuf[j];energy1[k]>averagenoise;k--)
                resegment[j]=k;
        if (segmentnumber.rem!=0)
            for (int k=segmentBuf[j];energy1[k]>averagenoise;k++)
                resegment[j]=k;
    }
    else
        if (segmentBuf[j]>resegment[j-1])
        {
            segmentnumber=div(j,2);
            if (segmentnumber.rem==0)
                for (int k=segmentBuf[j];energy1[k]>averagenoise;k--)
                    resegment[j]=k;
            if (segmentnumber.rem!=0)
                for (int k=segmentBuf[j];energy1[k]>averagenoise;k++)
                    resegment[j]=k;
        }
        else
            resegment[j]=segmentBuf[j];
}
```

```
        fprintf(sumsegmentfile," \n%s has %2d segment(s)",energyname,segment);
        fwrite(resegment,sizeof(int),segment*2,segmentfile);
            for (j=0;j<segment;j++)
            fprintf(segmenttextfile,"\nSegment#%2d %4d-%4d",j,resegment[j*2],resegment[(j*2)+1]);

    fclose(energyfile);
    fclose(segmentfile);
    fclose(segmenttextfile);
    fclose(sumsegmentfile);

    free(segmentBuf);
    free(resegment);
    free(energy1);

    return true;
}

//************************** PROCEDURE EDITWAVE ****************************
bool    Segment::editwave(CString inputfilename,CString segmentfilename)
{
    FILE    *inputfile;
    FILE    *segmentfile;
    FILE    *eachwordfile;
    int     *boundary;
    int     beginword;
    int     endword;
    CString eachwordfilename;
    CString numtext;

    if ((inputfile=fopen(inputfilename,"rb"))==NULL)
    {
        MessageBox(inputfilename,"Can't open file.", MB_OK);
        return false;
    }
    if ((segmentfile=fopen(segmentfilename,"rb"))==NULL)
    {
        MessageBox(segmentfilename,"Can't open file.", MB_OK);
        return false;
    }
    int segment=-1;
    while (!feof(segmentfile))
    {
        char e=fgetc(segmentfile);
        segment++;
    }
    segment/=4;
    boundary=(int*)calloc(sizeof(int),segment);
    if (boundary==NULL)
    {
        MessageBox("Can't allocate memory for boundary.", MB_OK);
        return false;
    }
    rewind(segmentfile);
    fread(boundary,sizeof(int),segment,segmentfile);
    for (int i=0;i<segment;i++)
        boundary[i]=boundary[i]*50;
    int length=-1;
    while (!feof(inputfile))
    {
       char e=fgetc(inputfile);
       length++;
    }
    int numberofword=segment/2;
    for (int wordnumber=0;wordnumber<numberofword;wordnumber++)
    {
        eachwordfilename=inputfilename;
        numtext.Format("%d",wordnumber);
        if (wordnumber<10)
            numtext='0'+numtext;
        int pos = eachwordfilename.Find('.');
        eachwordfilename.SetAt(pos  ,' ');
        eachwordfilename.SetAt(pos+1,' ');
        eachwordfilename.SetAt(pos+2,' ');
        eachwordfilename.SetAt(pos+3,' ');
        eachwordfilename.TrimRight();
        eachwordfilename=eachwordfilename+"-"+numtext+".wav";

        rewind(inputfile);
        if ((eachwordfile=fopen(eachwordfilename,"wb"))==NULL)
        {
            MessageBox(eachwordfilename,"Can't create file.", MB_OK);
            return false;
        }
        beginword=boundary[wordnumber*2];
        endword=boundary[(wordnumber*2)+1];
        char e;
        for (int j=0;j<44;j++)
        {
            e=fgetc(inputfile);
            putc(e,eachwordfile);
        }
        for (j=0;j<length;j++)
        {
            e=fgetc(inputfile);
            if ((j>beginword)&&(j<endword))
                putc(e,eachwordfile);
        }
        fclose(eachwordfile);
    }
    fclose(segmentfile);
    fclose(inputfile);
    free(boundary);
    return true;
}
```

```
//************************** PROCEDURE EDITRECOG ***************************
bool    Segment::editrecog(CString inputfilename,CString segmentfilename,CString resultfilename)
{
    FILE    *inputfile;
    FILE    *segmentfile;
    FILE    *eachwordfile;
    FILE    *resultfile;
    int     *boundary;
    int     beginword;
    int     endword;
    CString eachwordfilename;
    CString numtext;

    if ((inputfile=fopen(inputfilename,"rb"))==NULL)
    {
        MessageBox(inputfilename,"Can't open file.", MB_OK);
        return false;
    }
    if ((segmentfile=fopen(segmentfilename,"rb"))==NULL)
    {
        MessageBox(segmentfilename,"Can't open file.", MB_OK);
        return false;
    }
    if ((resultfile=fopen(resultfilename,"wt"))==NULL)
    {
        MessageBox(resultfilename,"Can't create file.", MB_OK);
        return false;
    }
    int segment=-1;
    while (!feof(segmentfile))
    {
        char e=fgetc(segmentfile);
        segment++;
    }
    segment/=4;
    boundary=(int*)calloc(sizeof(int),segment);
    if (boundary==NULL)
    {
        MessageBox("Can't allocate memory for boundary.", MB_OK);
        return false;
    }
    rewind(segmentfile);
    fread(boundary,sizeof(int),segment,segmentfile);
    for (int i=0;i<segment;i++)
        boundary[i]=boundary[i]*50;
    int length=-1;
    while (!feof(inputfile))
    {
       char e=fgetc(inputfile);
       length++;

    }
    int numberofword=segment/2;
    for (int wordnumber=0;wordnumber<numberofword;wordnumber++)
    {

        rewind(inputfile);
        if ((eachwordfile=fopen("temp.wav","wb"))==NULL)
        {
            MessageBox(eachwordfilename,"Can't create file.", MB_OK);
            return false;
        }
        beginword=boundary[wordnumber*2];
        endword=boundary[(wordnumber*2)+1];
        char e;
        for (int j=0;j<44;j++)
            {
            e=fgetc(inputfile);
            putc(e,eachwordfile);
            }
        for (j=0;j<length;j++)
            {
            e=fgetc(inputfile);
            if ((j>beginword)&&(j<endword))
                putc(e,eachwordfile);
            }
        fclose(eachwordfile);
        Recog recog;
    int result=recog.auto_recog("temp.wav");
        fprintf(resultfile,"\n%s  has  segment#%d   is model  #%d",inputfilename,wordnumber,result);

        }
    fclose(segmentfile);
    fclose(inputfile);
    fclose(resultfile);
    free(boundary);
    return true;
}


BEGIN_MESSAGE_MAP(Segment, CView)
    //{{AFX_MSG_MAP(Segment)
        // NOTE - the ClassWizard will add and remove mapping macros here.
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// Segment drawing

void Segment::OnDraw(CDC* pDC)
{
    CDocument* pDoc = GetDocument();
```

```
    // TODO: add draw code here
}

/////////////////////////////////////////////////////////////////////////////
// Segment diagnostics

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

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

/////////////////////////////////////////////////////////////////////////////
// Segment message handlers
```

```
// Vqcom.cpp : implementation file
//

#include "stdafx.h"
#include "art3.h"
#include "Vqcom.h"
#include "io.h"

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

/////////////////////////////////////////////////////////////////////////////
// Vqcom

//IMPLEMENT_DYNCREATE(Vqcom, CView)

Vqcom::Vqcom()
{
    codebook_file==NULL;
    codebook_index_file==NULL;
    codebook_index_text_file==NULL;
    input_vector_file==NULL;
    codebook==NULL;
    input_vector==NULL;
    min_index==NULL;
}

Vqcom::~Vqcom()
{
}



//*****************************VQCMP*********************
bool Vqcom::open_codebook_and_output_file(CString codebookfilename,CString codebook_index_file_name,CString codebookindext
extfilename)
{
    if ((codebook_file = fopen(codebookfilename,"rb")) == NULL)
    {
        MessageBox(codebookfilename, "Can't open file", MB_OK);
        return false;
    }
    if ((codebook_index_file = fopen(codebook_index_file_name,"wb")) == NULL)
    {
        MessageBox(codebook_index_file_name, "Can't create file", MB_OK);
        return false;
    }

    if ((codebook_index_text_file = fopen(codebookindextextfilename,"wt")) == NULL)
    {
        MessageBox(codebookindextextfilename, "Can't create file", MB_OK);
        return false;
    }
    return true;
}

bool Vqcom::allocate_codebook_memory(void)
{
    codebook = (float *) calloc(number_of_codebook*vector_dimension,sizeof(float));
    if (codebook == NULL)
    {
        MessageBox("Can't allocate memory for codebook", "ERROR", MB_OK);
        return false;
    }
    return true;
}
void Vqcom::read_codebook_file(void)
{
    fread(codebook,sizeof(float),number_of_codebook*vector_dimension,codebook_file);
}

void Vqcom::get_number_of_input_file_and_set_codebook_index_file_pointer(int number_of_input_file)
{
    int  flag=-1;
    fseek(codebook_index_file,number_of_input_file,SEEK_SET);
    fwrite(&flag,1,1,codebook_index_file);
}


bool Vqcom::open_input_vector_file(CString input_vector_file_name)
{
    if ((input_vector_file = fopen(input_vector_file_name,"rb")) == NULL)
    {
        MessageBox(input_vector_file_name, "Can't open file", MB_OK);
        return false;
    }
    return true;
}
int Vqcom::find_number_of_frame(void)
{
    int file_size = -1;
    rewind(input_vector_file);
    while (!feof(input_vector_file))
    {
        char c = fgetc(input_vector_file);
        file_size++;
    }
    rewind(input_vector_file);
    file_size -= file_size % (sizeof(float)*vector_dimension);
    int number_of_frame = (int)(file_size / (sizeof(float)*vector_dimension));
    return number_of_frame;
```

```
}

bool Vqcom::allocate_input_vector_and_min_index_memory(void)
{
    int number_of_frame=find_number_of_frame();
    input_vector =(float *) calloc(number_of_frame*vector_dimension,sizeof(float));
    if (input_vector == NULL)
    {
        MessageBox("Can't allocate memory for input_vector","ERROR", MB_OK);
        return false;
    }

    min_index = (char *) calloc(number_of_frame, sizeof(char));
    if (min_index == NULL)
    {
        MessageBox("Can't allocate memory for min_index","ERROR", MB_OK);
        return false;
    }
    return true;
}
void Vqcom::read_input_vector(void)
{
    int number_of_frame=find_number_of_frame();
    fread(input_vector,sizeof(float),number_of_frame*vector_dimension,input_vector_file);
}

void Vqcom::find_codebook_index(void)
{
    int number_of_frame=find_number_of_frame();
    for (int nf=0;nf<number_of_frame;nf++)
    {
        for (int nc=0;nc<number_of_codebook;nc++)
        {
            double total_distance = 0;
            double distance       = 0;
            for (int vd=0;vd<vector_dimension;vd++)
            {
                distance = input_vector[nf*vector_dimension + vd] -codebook[nc*vector_dimension + vd];
                distance *= distance;
                total_distance += distance;
            }
            vector_distance[nc] = total_distance;
        }
        double min_distance  = vector_distance[0];
        min_index[nf] = 0;
        for (nc=0;nc<number_of_codebook;nc++)
        {
            if ( min_distance > vector_distance[nc] )
            {
                min_distance  = vector_distance[nc];
                min_index[nf] = (char)nc;
            }
        }
    }
}

void Vqcom::write_codebook_index_file(int file_number)
{
    int index_file_pointer = (int)ftell(codebook_index_file);
    fseek(codebook_index_file,file_number,SEEK_SET);
    int number_of_frame=find_number_of_frame();
    fwrite(&number_of_frame,1,1,codebook_index_file);

    fseek(codebook_index_file,index_file_pointer,SEEK_SET);
    for (int nf=0;nf<number_of_frame;nf++)
        min_index[nf] += 1;
    fwrite(min_index,1,number_of_frame,codebook_index_file);
}

void Vqcom::free_input_vector_and_min_index_memory(void)
{
//  free(codebook);
    free(input_vector);
    free(min_index);
}

void Vqcom::close_input_vector_file(void)
{
    fclose(input_vector_file);
}

bool Vqcom::write_codebook_index_text_file(CString codebook_index_file_name)
    {
    char item;


    {
        fclose(codebook_index_file);
        if ((codebook_index_file = fopen(codebook_index_file_name,"rb")) == NULL)
        {
            MessageBox(codebook_index_file_name, "Can't create file", MB_OK);
            return false;
        }
        while  (!feof(codebook_index_file))
        {
            fread(&item,1,1,codebook_index_file);
            fprintf(codebook_index_text_file,"%2d\n",item);
        }
    }
    return true;
}

void Vqcom::free_codebook_memory(void)
{
```

```
    free(codebook);
}

void Vqcom::close_all_vq_file(void)
{
    fclose(codebook_file);
    fclose(codebook_index_file);
    fclose(codebook_index_text_file);
}

//*****************************************************

BEGIN_MESSAGE_MAP(Vqcom, CView)
    //{{AFX_MSG_MAP(Vqcom)
        // NOTE - the ClassWizard will add and remove mapping macros here.
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// Vqcom drawing

void Vqcom::OnDraw(CDC* pDC)
{
    CDocument* pDoc = GetDocument();
    // TODO: add draw code here
}

/////////////////////////////////////////////////////////////////////////////
// Vqcom diagnostics

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

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

/////////////////////////////////////////////////////////////////////////////
// Vqcom message handlers
```

```
// Vqcomdia.cpp : implementation file
//

#include "stdafx.h"
#include "art3.h"
#include "Vqcomdia.h"
#include "Vqcom.h"
#include "stdlib.h"

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

/////////////////////////////////////////////////////////////////////////////
// Vqcomdia dialog


Vqcomdia::Vqcomdia(CWnd* pParent /*=NULL*/)
    : CDialog(Vqcomdia::IDD, pParent)
{
    //{{AFX_DATA_INIT(Vqcomdia)
    m_input = _T("");
    m_number = _T("");
    m_text = _T("");
    m_output = _T("");
    //}}AFX_DATA_INIT
}


void Vqcomdia::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(Vqcomdia)
    DDX_Control(pDX, IDC_LIST2, m_list1);
    DDX_Control(pDX, IDC_LIST1, m_list);
    DDX_Text(pDX, IDC_EDIT1, m_input);
    DDX_Text(pDX, IDC_EDIT3, m_number);
    DDX_Text(pDX, IDC_EDIT4, m_text);
    DDX_Text(pDX, IDC_EDIT2, m_output);
    //}}AFX_DATA_MAP
}


BEGIN_MESSAGE_MAP(Vqcomdia, CDialog)
    //{{AFX_MSG_MAP(Vqcomdia)
    ON_BN_CLICKED(IDC_BUTTON1, OnShowList)
    ON_BN_CLICKED(IDC_BUTTON2, OnRUN)
    ON_BN_CLICKED(IDC_BUTTON3, OnSaveAs)
    ON_LBN_DBLCLK(IDC_LIST1, OnShowSave)
    ON_LBN_DBLCLK(IDC_LIST2, OnShowsave2)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// Vqcomdia message handlers

void Vqcomdia::OnShowList()
{
    CFileFind   filefind;
    char        notfound;

    UpdateData(true);
    number_of_file=0;
    m_list.ResetContent();
    m_list1.ResetContent();
    filefind.FindFile("*.lpc",0);
    for (;notfound!=0;)
    {
        notfound=filefind.FindNextFile();
        m_list.AddString(filefind.GetFileName());
    }
    UpdateData(false);

}

void Vqcomdia::OnShowSave()
{
    UpdateData(true);
    CString temp;
    m_list.GetText(m_list.GetCurSel(),temp);
    m_list1.AddString(temp);
    m_list.DeleteString(m_list.GetCurSel());
    number_of_file++;
    m_number.Format("%3d",number_of_file);
    UpdateData(false);
}

void Vqcomdia::OnSaveAs()
{

    UpdateData(true);

    CString m_output1;
    m_list1.GetText(m_list1.GetCurSel(),m_output1);

    int pos=m_output1.FindOneOf("-");
    //int pos=m_output1.Find('.');
    m_output=m_output1;
    m_output.SetAt(pos,'.');
    m_output.SetAt(pos+1,'i');
    m_output.SetAt(pos+2,'d');
    m_output.SetAt(pos+3,'x');
    m_output.SetAt(pos+4,' ');
```

```
    m_output.SetAt(pos+5,' ');
    m_output.SetAt(pos+6,' ');
    m_output.SetAt(pos+7,' ');
    m_output.SetAt(pos+8,' ');
    m_output.SetAt(pos+9,' ');

    int pos1=m_output.Find('.');
    m_text=m_output;
    m_text.SetAt(pos1+1,'t');
    m_text.SetAt(pos1+2,'x');
    m_text.SetAt(pos1+3,'t');
    UpdateData(false);
}


void Vqcomdia::OnRUN()
{


    Vqcom       vqcom;

    vqcom.open_codebook_and_output_file(m_input,m_output,m_text);
    vqcom.allocate_codebook_memory();
    vqcom.read_codebook_file();
    vqcom.get_number_of_input_file_and_set_codebook_index_file_pointer(number_of_file);
    for (int file_number=0;file_number<number_of_file;file_number++)
    {
        CString inputfilename;
        m_list1.GetText(file_number,inputfilename);
        vqcom.open_input_vector_file(inputfilename);
        vqcom.allocate_input_vector_and_min_index_memory();
        vqcom.read_input_vector();
        vqcom.find_codebook_index();
        vqcom.write_codebook_index_file(file_number);
        vqcom.free_input_vector_and_min_index_memory();
        vqcom.close_input_vector_file();
    }
    vqcom.write_codebook_index_text_file(m_output);
    vqcom.free_codebook_memory();
    vqcom.close_all_vq_file();
    MessageBox("Codebook Index File..Complete!",m_output);

        UpdateData(false);


}



void Vqcomdia::OnShowsave2()
{
    // TODO: Add your control notification handler code here
        UpdateData(true);
    CString temp;
    m_list1.GetText(m_list1.GetCurSel(),temp);
    m_list.AddString(temp);
    m_list1.DeleteString(m_list1.GetCurSel());
    number_of_file--;
    m_number.Format("%3d",number_of_file);
    UpdateData(false);
}
```

```
4// vqln.cpp : implementation file
//

#include "stdafx.h"
#include "art3.h"
#include "Vqln.h"
#include "io.h"
#include "stdlib.h"
#include "malloc.h"


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

int round_number=1;
double total_distance;
double distance;
/////////////////////////////////////////////////////////////////////////////
// Vqln

//IMPLEMENT_DYNCREATE(Vqln, CView)

Vqln::Vqln()
{
}

Vqln::~Vqln()
{
}

//************************** PROCEDURE VQLN    ****************************
bool Vqln::open_all_file_vq(CString inputfilename,CString outputfilename,CString outputtextfilename)
{
    if ((training_set_file = fopen(inputfilename,"rb")) == NULL)
    {
        MessageBox(inputfilename, "Can't open file", MB_OK);
        return false;
    }
    if ((codebook_file = fopen(outputfilename,"wb")) == NULL)
    {
        MessageBox(outputfilename, "Can't create file", MB_OK);
        return false;
    }

    if ((codebook_text_file = fopen(outputtextfilename,"wt")) == NULL)
    {
        MessageBox(outputtextfilename, "Can't create file", MB_OK);
        return false;
    }
    if ((distortion_text_file = fopen("distortion.txt","wt")) == NULL)
    {
        MessageBox("distortion.txt", "Can't create file", MB_OK);
        return false;
    }
    if ((random_text_file = fopen("random.txt","wt")) == NULL)
    {
        MessageBox("random.txt", "Can't create file", MB_OK);
        return false;
    }
    return true;
}


int Vqln::find_number_of_frame_vq(void)
    {
    int file_size;
    file_size = filelength(open("allsample.lpc",0));
    file_size -= file_size % (sizeof(float)*vector_dimension);
    int number_of_frame = file_size / (sizeof(float)*vector_dimension);
    return number_of_frame;
    }

bool Vqln::allocate_data_memory(void)
    {
    int number_of_frame=find_number_of_frame_vq();
    training_set        =(float*) calloc(number_of_frame*vector_dimension,sizeof(float));
    if (training_set == NULL)
    {
        MessageBox("Can't allocate memory for training set", "ERROR!", MB_OK);
        return false;
    }
    previous_centroid   =(float *) calloc(number_of_codebook*vector_dimension,sizeof(float));
    if (previous_centroid == NULL)
    {
        MessageBox("Can't allocate memory for previous_centroid", "ERROR!", MB_OK);
        return false;
    }
    final_centroid      =(float *) calloc(number_of_codebook*vector_dimension,sizeof(float));
    if (final_centroid == NULL)
    {
        MessageBox("Can't allocate memory for final_centroid", "ERROR!", MB_OK);
        return false;
    }
    min_index           = (char*) calloc(number_of_frame, sizeof(char));
    if (min_index == NULL)
    {
        MessageBox("Can't allocate memory for min_index", "ERROR!", MB_OK);
        return false;
    }
    return true;
}
```

```
void Vqln::read_training_set(void)
{
    int number_of_frame=find_number_of_frame_vq();
    fread(training_set,sizeof(float),number_of_frame*vector_dimension,training_set_file);
}

void Vqln::random_start_centroid_from_training_set(void)
{
    int number_of_frame=find_number_of_frame_vq();
    for(int nc=0;nc<number_of_codebook;nc++)
    {
        int random_frame_number=number_of_frame;
        for (;(random_frame_number<0)||(random_frame_number>number_of_frame-1);)
            random_frame_number = rand();
        for (int vd=0;vd<vector_dimension;vd++)
        {
            previous_centroid[nc*vector_dimension + vd] =
            training_set[vector_dimension*random_frame_number + vd];
        }
    }

}

void Vqln::find_codebook(void)
{

    int number_of_frame=find_number_of_frame_vq();
    for (int nf=0;nf<number_of_frame;nf++)
    {
        for (int nc=0;nc<number_of_codebook;nc++)
        {
            total_distance = 0;
            distance       = 0;
            for (int vd=0;vd<vector_dimension;vd++)
            {
                distance = training_set[nf*vector_dimension + vd] - previous_centroid[nc*vector_dimension + vd];
                distance *= distance;
                total_distance += distance;
            }
            vector_distance[nc] = total_distance;
        }
        double min_distance = vector_distance[0];
        min_index[nf] = 0;
        for (nc=0;nc<number_of_codebook;nc++)
        {
            if ( min_distance > vector_distance[nc] )
            {
                min_distance = vector_distance[nc];
                min_index[nf] = nc;
            }
        }
    } /*end all frame*/

    fprintf(distortion_text_file,"%3d   ",round_number);
    round_number++;
    for (int nc=0;nc<number_of_codebook;nc++)
    {
        int codebook_member = 0;
        for(int vd=0;vd<vector_dimension;vd++)
            final_centroid[nc*vector_dimension + vd] = 0;
        for (nf=0;nf<number_of_frame;nf++)
        {
            if ( nc == min_index[nf] )
            {
                codebook_member += 1;
                for(int vd=0;vd<vector_dimension;vd++)
                {
                    final_centroid[nc*vector_dimension + vd] +=
                    training_set[nf*vector_dimension + vd];
                }
            }
        }
        if (codebook_member != 0 )
        {
            for(int vd=0;vd<vector_dimension;vd++)
                final_centroid[nc*vector_dimension + vd] /= codebook_member;
        }
        else
        {
            for(int vd=0;vd<vector_dimension;vd++)
                final_centroid[nc*vector_dimension + vd] = previous_centroid[nc*vector_dimension + vd];
        }
    }
}

int Vqln::check_distance(void)
{
    /*float total_distance;*/
    for (int nc=0;nc<number_of_codebook;nc++)
    {
        total_distance = 0;
        /*float*/ distance = 0;
        for (int vd=0;vd<vector_dimension;vd++)
        {
            distance = (final_centroid[nc*vector_dimension + vd]-previous_centroid[nc*vector_dimension + vd]);
            distance *= distance;
            total_distance += distance;
        }
        if ( total_distance > REJECT_VALUE )
        {
            printf(" total distortion = %13.10f\n",total_distance);
            fprintf(distortion_text_file,"%13.10f\n",total_distance);
            return(REJECT);
        }
    }
```

```
    fprintf(distortion_text_file,"%13.10f\n",total_distance);
    return(ACCEPT);
}

void Vqln::copy_final_centroid_to_previous_centroid(void)
{
    float *temp;

    temp              = previous_centroid;
    previous_centroid = final_centroid;
    final_centroid    = temp;
}

void Vqln::write_codebook_file(void)
{
    fwrite(final_centroid,4,number_of_codebook*vector_dimension,codebook_file);

    for (int nc=0;nc<number_of_codebook;nc++)
    {
        for (int vd=0;vd<vector_dimension;vd++)
        {
            fprintf(codebook_text_file,"%13.10f\n",final_centroid[nc*vector_dimension + vd]);
        }
        fprintf(codebook_text_file,"\n");
    }
}

void Vqln::close_all_file_vq(void)
{
    fclose(training_set_file);
    fclose(codebook_file);
    fclose(codebook_text_file);
    fclose(random_text_file);
    fclose(distortion_text_file);
}

void Vqln::RunVqln(CString inputfilename,CString outputfilename,CString outputtextfilename)
{
    Vqln vqln;

    vqln.open_all_file_vq(inputfilename,outputfilename,outputtextfilename);
    vqln.allocate_data_memory();
    vqln.read_training_set();
    vqln.random_start_centroid_from_training_set();
    for(;;)
    {
        vqln.find_codebook();
        if ( vqln.check_distance() == REJECT )
        {
            vqln.copy_final_centroid_to_previous_centroid();
        }
        else break;
    }
    vqln.write_codebook_file();
    vqln.close_all_file_vq();
}



BEGIN_MESSAGE_MAP(Vqln, CView)
    //{{AFX_MSG_MAP(Vqln)
        // NOTE - the ClassWizard will add and remove mapping macros here.
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// Vqln drawing

void Vqln::OnDraw(CDC* pDC)
{
    CDocument* pDoc = GetDocument();
    // TODO: add draw code here
}

/////////////////////////////////////////////////////////////////////////////
// Vqln diagnostics

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

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

/////////////////////////////////////////////////////////////////////////////
// Vqln message handlers
```

```
// Vqlndia.cpp : implementation file
//

#include "stdafx.h"
#include "art3.h"
#include "Vqlndia.h"
#include "Vqln.h"

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

/////////////////////////////////////////////////////////////////////////////
// Vqlndia dialog


Vqlndia::Vqlndia(CWnd* pParent /*=NULL*/)
    : CDialog(Vqlndia::IDD, pParent)
{
    //{{AFX_DATA_INIT(Vqlndia)
    m_input = _T("");
    m_out = _T("");
    m_outtext = _T("");
    //}}AFX_DATA_INIT
}


void Vqlndia::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(Vqlndia)
    DDX_Text(pDX, IDC_EDIT1, m_input);
    DDX_Text(pDX, IDC_EDIT2, m_out);
    DDX_Text(pDX, IDC_EDIT3, m_outtext);
    //}}AFX_DATA_MAP
}


BEGIN_MESSAGE_MAP(Vqlndia, CDialog)
    //{{AFX_MSG_MAP(Vqlndia)
    ON_BN_CLICKED(IDC_BUTTON3, OnRUN)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// Vqlndia message handlers

void Vqlndia::OnRUN()
{
    Vqln vqln;
    // TODO: Add your control notification handler code here
    vqln.RunVqln(m_input,m_out,m_outtext);
    m_outtext+="...Complete";
    AfxMessageBox("Codebook File Complete..!");
    UpdateData(false);
}
```