การควบคุมอุปกรณ์ไฟฟ้าโดยใช้เสียง

ELECTRICAL EQUIPMENT CONTROLLING BY SPEECH RECOGNITION

โดย

นายวรา คงคาวิทูร

นายวิรัช ลีนะวงศ์อนันต์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2540

การควบคุมอุปกรณ์ไฟฟ้าโดยใช้เสียง

ELECTRICAL EQUIPMENT CONTROLLING BY SPEECH RECOGNITION

โดย

นายวรา คงคาวิฑูร 37014379

นายวิรัช ลีนะวงศ์อนันต์ 37014404

อาจารย์ที่ปรึกษา

ผศ.ดร.ไกรสิน ส่งวัฒนา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2540

ปริญญานิพนธ์ปีการศึกษา 2540

ภาควิชาวิศวกรรมโทรคมนาคม

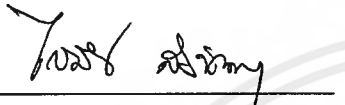
คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง **การควบคุมอุปกรณ์ไฟฟ้าโดยใช้เสียง**

ELECTRICAL EQUIPMENT CONTROLLING BY SPEECH RECOGNITION

ผู้จัดทำ

1. นายวรา คงคาวิทูร 37014379
2. นายวิรัช ลีนะวงศ์อนันต์ 37014404



อาจารย์ที่ปรึกษา

(ผศ.ดร.ไกรสิน ไกรสินสังวัฒนา)

การควบคุมอุปกรณ์ไฟฟ้าโดยใช้เสียง

Electrical Equipment Controlling By Speech Recognition

โดย นายวรา คงคาวิทูร 37014379
นายวิรัช ลีนะวงศ์อนันต์ 37014404
อาจารย์ที่ปรึกษา ผศ.ดร.ไกรสิน ส่งวัฒนา

บทคัดย่อ

ปริญญานิพนธ์เรื่องนี้ มีจุดประสงค์ ที่จะทำการศึกษา และทำการเขียนโปรแกรมเพื่อสร้างโปรแกรมประยุกต์ที่สามารถทำให้สามารถใช้เสียงมนุษย์ควบคุมอุปกรณ์ไฟฟ้าได้ โปรแกรมนี้จะมีหน้าที่การทำงานคร่าวๆ คือ สามารถทำการสร้างเสียงโมเดลเสียง (ใช้ Hidden Makov Model:HMM) จากไฟล์เสียงต้นแบบ, สามารถบันทึกเสียงแล้วนำเสียงที่ได้ไปตัดแบ่งเป็นคำ ๆ ได้ ,สามารถนำเสียงที่ตัดแบ่งเป็นคำ ๆ แล้วนั้นไปเปรียบเทียบกับโมเดลเสียงที่มีอยู่เพื่อตรวจรู้ได้ว่าตรงกับโมเดลใด และสุดท้ายนำผลที่ได้ส่งออกไปเป็นอินพุทของวงจรขับรีเลย์เพื่อควบคุมอุปกรณ์ไฟฟ้าได้

ABSTRACT

This project's objective is to leran and create the application program which can receivers commands from human in the form of human voice. Then it can translate that command and respond to the real world through data buses and relay circuits. This project can satisfy these following function. First ,it can construct the model from sample sounds (use Hidden Makov Model:HMM). Second ,it can capture voice from a sound card in the computer and then segment that voice to words. Next ,it can recognize each word by comparing sounds from the previous step with the model , and finally use that recognized model to decide what data must be send to the relay circuit that is controlling electrical equipments.

สารบัญ

สารบัญรูป

หน้า

สารบัญตาราง

หน้า

บทที่ 1
บทนำ

เนื่องจากความก้าวหน้าทางเทคโนโลยีในปัจจุบันรวมทั้ง ความต้องการความสะดวกสบายของมนุษย์ จึงเป็นผลให้เกิดการพัฒนาการทางด้านคอมพิวเตอร์อย่างรวดเร็ว ทั้งในด้านฮาร์ดแวร์ และซอฟท์แวร์ ทางด้านฮาร์ดแวร์มีการพัฒนาให้หน่วยประมวลผลมีประสิทธิภาพมากขึ้น ราคาถูกลงและหน่วยความจำก็มีขนาดใหญ่ขึ้น ความเร็วในการถ่ายโอนสูง จึงเป็นผลให้คอมพิวเตอร์เข้ามามีส่วนรวมในชีวิตมนุษย์มากขึ้นอย่างหลีกเลี่ยงไม่ได้

ในสมัยเริ่มแรกมนุษย์สามารถติดต่อสื่อสาร หรือสั่งการคอมพิวเตอร์ให้ทำงานโดย ใช้รหัสอักษรหรือเรียกว่าเทกส์โหมด (Text mode) แล้วต่อมาก็ได้พัฒนาขึ้นมาให้สามารถใช้งานง่ายขึ้นโดยผ่านทางรูปภาพหรือกราฟฟิคเรียกว่า กราฟฟิคโหมด (Graphic mode) แต่วิวัฒนาการก็ยังมิได้หยุดยั้งเพียงเท่านั้นจากที่กล่าวมาในข้างต้นนี้ว่า มีการพัฒนาทั้งฮาร์ดแวร์ และซอฟท์แวร์ของคอมพิวเตอร์ขึ้นอย่างมาก ทำให้การคำนวณที่ซับซ้อนซึ่งไม่อาจทำได้รวดเร็วในอดีตนั้นสามารถทำให้สำเร็จผลอย่างมีประสิทธิภาพ ด้วยการคำนวณที่รวดเร็วของเครื่องคอมพิวเตอร์ในปัจจุบัน

การติดต่อระหว่างคอมพิวเตอร์ และมนุษย์ซึ่งเราเคยแต่เพียงเห็นในนิยายวิทยาศาสตร์จึงสามารถจะทำให้สำเร็จได้ เป็นการใช้เสียงมนุษย์สั่งการให้คอมพิวเตอร์ทำงาน ซึ่งการที่คอมพิวเตอร์จะสามารถทำงานตามที่เราสั่งได้อย่างถูกต้องนั้น คอมพิวเตอร์จะต้องเข้าใจว่าเราพูดอะไร การที่เราจะทำให้คอมพิวเตอร์รู้จำคำพูดของเราได้นั้นสามารถทำได้โดยการใช้โปรแกรมที่ชื่อว่า โปรแกรมรู้จำเสียงพูดของมนุษย์ (Natural Speech Recognition) ซึ่งโปรแกรมนี้จะเป็นตัวที่ทำหน้าที่รับเสียงพูดของมนุษย์เข้ามา แล้วทำการแยกแยะคำพูดออกเป็นคำ ๆ แล้วจึงทำการวิเคราะห์คำแต่ละคำนั้นว่าเป็นคำว่าอะไร และจะต้องทำอย่างไรเมื่อได้รับคำสั่งนั้น

ปริญญานิพนธ์เรื่องนี้เป็นส่วนของการแยกแยะคำพูดออกเป็นคำ ๆ ก่อนที่จะทำกระบวนการวิเคราะห์หรือ รู้จักคำนั้น

เหตุที่ทำปริญญานิพนธ์เรื่องนี้เนื่องจากว่าพบว่าส่วนใหญ่แล้ว ผู้ที่ทำปริญญานิพนธ์ในรุ่นก่อนๆ นั้นมุ่งเน้นเฉพาะการออกแบบส่วนที่ทำหน้าที่ในการรู้จำคำพูดแต่เพียงอย่างเดียว ทำให้ตัวโปรแกรมที่ได้นั้นไม่สามารถนำมาใช้ได้จริงในทางปฏิบัติ

เนื่องจากว่าส่วนของการรู้จำเสียงนั้น จะต้องการอินพุทที่เป็นคำโดด ๆ และมีขอบเขตของคำที่แน่นอน ซึ่งในทางปฏิบัตินั้นแต่ก่อนจะใช้การบันทึกเสียงโดยใช้คอมพิวเตอร์ก่อน แล้วจึงใช้การสังเกตุดูจากรูปคลื่น (Wave form) แล้วใช้วิจารณญาณของเราในการตัดสินใจว่าจุดใดเป็นจุดเริ่มต้นของคำ และจุดใดเป็นจุดสิ้นสุดของคำ แล้วจึงตัดคำนั้นด้วยโปรแกรมแก้ไขรูปคลื่น (Waveform editor) ซึ่งเป็นโปรแกรมสำเร็จรูป ซึ่งการทำเช่นนี้เท่ากับว่าไม่มีประโยชน์ในการใช้งานจริง และยังไปได้ประสิทธิภาพที่เท่าเทียมกันในแต่ละครั้งด้วย

ดังนั้นปริญญานิพนธ์กลุ่มของข้าพเจ้านี้จะทำในส่วนที่รับเสียงพูดจากมนุษย์ และส่งต่อให้กับ ส่วนรู้จำเสียงพูดในลักษณะที่เป็นคำโดด สามารถประมวลผลได้ทันที โดยจะแบ่งออกเป็น 2 ส่วน

ส่วนแรกเป็นส่วนที่จะทำการเขียนโปรแกรมเพื่อควบคุมการ์ดเสียง (Sound Card) ของเครื่องคอมพิวเตอร์ให้สามารถทำงานในลักษณะที่ไม่ต้องทำการบันทึกเสียงด้วยโปรแกรมสำเร็จรูปอื่น แต่จะทำการ

รับเสียง และส่งข้อมูลเสียงนั้นเข้าสู่โปรแกรมตลอดเวลา เพื่อที่จะสามารถรับคำสั่งทางเสียง และนำข้อมูลนั้นไปประมวลผลจนถึงการที่คอมพิวเตอร์นำผลที่ได้ ไปควบคุมอุปกรณ์ไฟฟ้าที่ต่ออยู่ภายนอกได้

ส่วนที่สองจะเป็นส่วนที่นำข้อมูลเสียงนั้นที่ได้มาทำงานการคำนวณ เพื่อที่จะตัดแบ่งคำ โดยอินพุทของส่วนนี้จะเป็นเสียงที่ประกอบด้วยคำโดดหลายคำ โดยแต่ละคำจะต้องมีเสียงเงียบ (silence) กั้นระหว่างแต่ละคำเล็กน้อย และการคำนวณจะใช้ 3 วิธีร่วมกันคือ การใช้ค่าพลังงานของเสียง ค่าอัตราการตัดศูนย์เฉลี่ย และค่าความถี่พิทช์ร่วมกัน เมื่อคำนวณค่าทั้ง 3 นี้ได้ก็จะนำไปใช้ในการพิจารณาในการตัดคำต่อไป

บทที่ 2
ทฤษฎี

หลักการประมวลผลในเชิงเวลานั้นคือ การวิเคราะห์สัญญาณของเสียงพูดโดยตรงจากรูปคลื่นสัญญาณ (wave form) ซึ่งแตกต่างจากการประมวลผลเชิงความถี่ (Frequency-Domain) ตัวอย่างของการประมวลผลของสัญญาณเสียงพูดเชิงเวลา เช่น การหาค่าอัตราการตัดศูนย์เฉลี่ย (Zero-crossing Rate) ,การหาค่าพลังงาน (Energy) และค่าออโตคอรีเลชั่น (Autocorrelation) โดยการประมวลผลหาค่าเหล่านี้เป็นไปโดยง่ายมีขั้นตอนที่ไม่ซับซ้อน แต่ให้องค์ประกอบสำคัญของสัญญาณเสียง

2.1 การประมวลผลเสียงพูดโดยขึ้นกับเวลา

จากรูปที่ 2.1 ซึ่งเป็นสัญญาณเสียงพูดที่มีอัตราการสุ่มตัวอย่างของสัญญาณเป็น 8000 ค่าใน 1 วินาที จะเห็นได้ชัดว่าคุณสมบัติของสัญญาณเสียงพูดจะมีลักษณะที่เปลี่ยนแปลงไปตามเวลา ตัวอย่างเช่นการเปลี่ยนแปลงสัญญาณเสียงพูดระหว่างเสียงพูดที่เป็นเสียงก้องหรือเสียงโฆษะ(voice) และเสียงไม่ก้องหรือเสียงอโฆษะ(unvoice) โดยที่เสียงจะมีการเปลี่ยนแปลงที่เห็นได้ชัดจากขนาด(magnitude) ของสัญญาณ และยังมีการเปลี่ยนแปลงของความถี่มูลฐานของเสียง (fundamental frequency) ภายในช่วงที่เป็นโฆษะด้วย โดยการที่แสดงรูปแบบสัญญาณเป็นรูปคลื่นนี้สามารถทำให้ง่ายต่อการสังเกตุถึงองค์ประกอบและคุณลักษณะของเสียงนั้น เช่น ความเข้มของเสียง(Intensity) ,ชนิดของเสียง(เสียงก้องหรือ เสียงไม่ก้อง) หรือ excitation mode, ความถี่มูลฐาน(pitch หรือ fundamental frequency) และอาจรวมถึงสัมประสิทธิ์ในอวัยวะกำเนิดเสียง เช่น ความถี่กำทอน (resonant หรือ formant frequency)

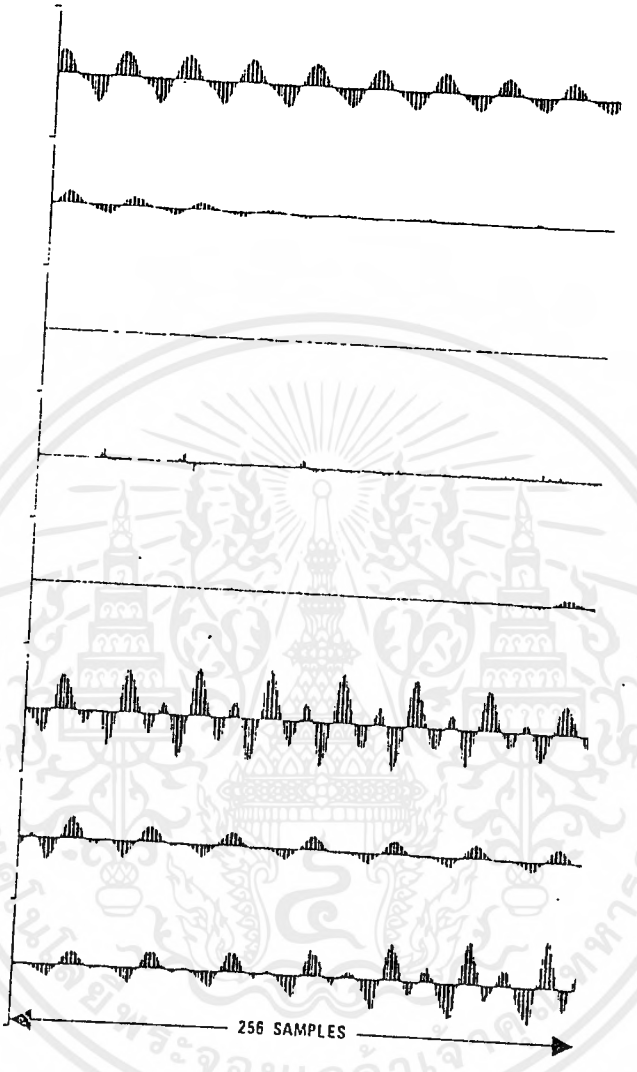
สมมติฐานที่ใช้ในการประมวลผลสัญญาณเสียงส่วนมากก็ คือการที่ถือเอาว่าคุณลักษณะของเสียงนั้นมีการเปลี่ยนแปลงช้ามากเมื่อเทียบกับเวลา และสมมติฐานนี้ทำให้เกิดการประมวลผลที่เรียกว่าการประมวลผลแบบช่วงเวลาสั้น ๆ(short-time) โดยที่ส่วนเล็ก ๆ แต่ละส่วนของเสียงพูด นั้นที่ถูกแยกออกมาและจะถูกประมวลผลโดยแต่ละส่วนของเสียงพูดนี้ถูกเรียกว่า การวิเคราะห์เฟรม(analysis frames) ซึ่งเฟรมแต่ละเฟรมจะมีส่วนที่ซ้อนทับกัน และผลลัพธ์ที่ได้ของแต่ละส่วนนั้นอาจเป็นตัวเลขตัวเดียว หรืออาจเป็นกลุ่มของตัวเลขก็ได้ ดังนั้นกระบวนการนี้ก็จะทำให้เกิดลำดับของค่าใหม่ในเชิงของเวลา ซึ่งแสดงถึงลักษณะของสัญญาณเสียง

กระบวนการวิเคราะห์เสียงแบบช่วงเวลาสั้น ๆ นั้นเกือบทั้งหมดสามารถแสดงในรูปของสมการทางคณิตศาสตร์ที่มีรูปแบบ คือ

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

สมการนี้สัญญาณเสียงจะถูกกรองเอาเฉพาะย่านความถี่ที่ต้องการ และจะถูกกระทำโดยฟังก์ชั่น $T[\ ]$ ซึ่งอาจเป็นเชิงเส้นหรือไม่ขึ้นอยู่กับสัมประสิทธิ์ที่จะทำการคำนวณ และหลังจากนั้นลำดับที่ได้จะถูกคูณด้วย

ลำดับวินโดว์ ในตำแหน่งที่สัมพันธ์กับเวลาของตัวอย่างสัญญาณที่ n และผลคูณก็จะถูกรวมกัน โดยทั่วไปแล้ว ผลลัพธ์จะเกิดจากลำดับที่จำกัด



รูปที่ 2.1 แสดงรูปสัญญาณเสียงที่มีอัตราการสุ่มเป็น 8 kHz

2.2 ค่าพลังงานของสัญญาณเฉลี่ยในช่วงเวลาสั้น ๆ(Short-time Average Energy)

จากรูปที่ 2.1 เราสามารถสังเกตได้ว่าค่าพลังงานของสัญญาณ(Amplitude) นั้นจะมีการเปลี่ยนแปลงไปตามเวลา โดยเฉพาะพลังงานของสัญญาณของเสียงไม่ก้อง มักจะมีค่าต่ำกว่าพลังงานของสัญญาณของเสียง ก้องมากซึ่งการใช้ค่าพลังงานของสัญญาณเฉลี่ยในช่วงสั้น ๆ สามารถแสดงให้เห็นถึงความแตกต่างนี้ได้อย่างชัดเจน โดยมีรูปแบบสมการของการใช้ในการคำนวณ 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 2.2 สัญญาณ $X^2(m)$ จะถูกกรองโดยตัวกรองแบบเชิงเส้น $h(n)$ ซึ่ง $h(n)$ ก็คือวินโดว์ (window) ลองมาดูว่าวินโดว์ที่แตกต่างกันจะมีผลอย่างไรกับค่าพลังงานผลลัพธ์ที่ได้ โดยถ้า $h(n)$ ให้ค่าขนาดของสัญญาณคงที่ในช่วงเวลาที่ยาวนานแล้วค่า E_n จะมีการเปลี่ยนแปลงน้อยมาก เนื่องจากเราต้องการให้ค่าขนาดกำลังสองในช่วงเวลาสั้น ๆ แสดงให้เห็นถึงการเปลี่ยนแปลงของขนาดเสียงในสัญญาณเสียง ดังนั้นจึงต้องใช้วินโดว์ ที่มีช่วงเวลาสั้นเกินไปก็จะทำให้ไม่ได้ค่าที่มีความเรียบพอเพียง โดยตัวฟังก์ชั่นวินโดว์มีรูปแบบเป็นดังนี้

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 $0 \le n \le N-1$

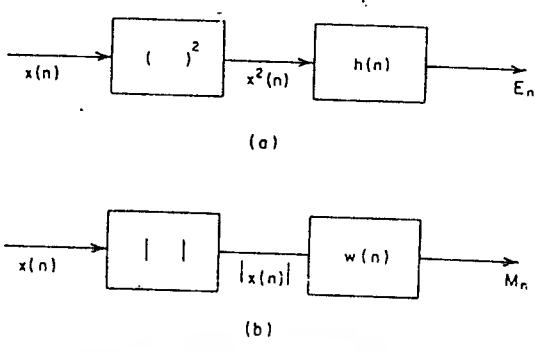
$h(n) = 0$ อื่น ๆ

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

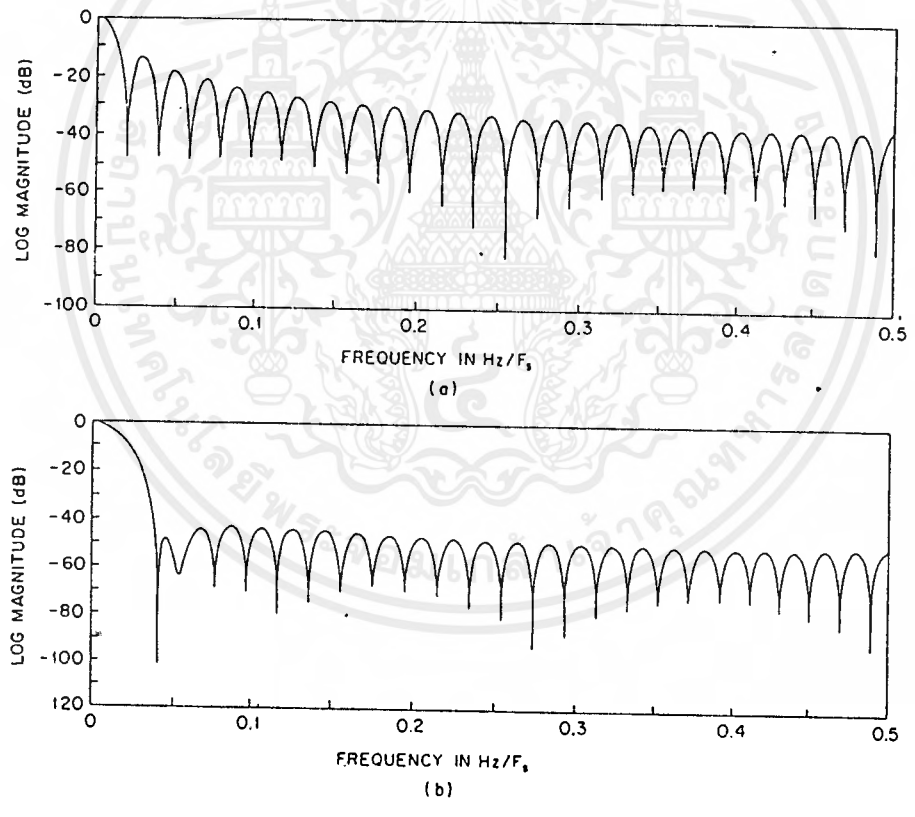
$$H\left(e^{j\Omega T}\right) = \frac{\sin\left(\Omega NT/2\right)}{\sin\left(\Omega T/2\right)}e^{-j\Omega T(N-1)/2}$$

ส่วนแฮมมิ่งวินโดว์ นั้นจะใช้ประโยชน์ได้ดีสำหรับการวิเคราะห์เชิงความถี่ เพราะให้ค่า ลดทอนภายนอกย่านมากกว่าดังรูปที่ 2.3

จากที่กล่าวมาแล้วว่าถ้าค่า N ในวินโดว์มีค่าน้อยเกินไปก็จะมีการเปลี่ยนแปลงของค่า E_n 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)

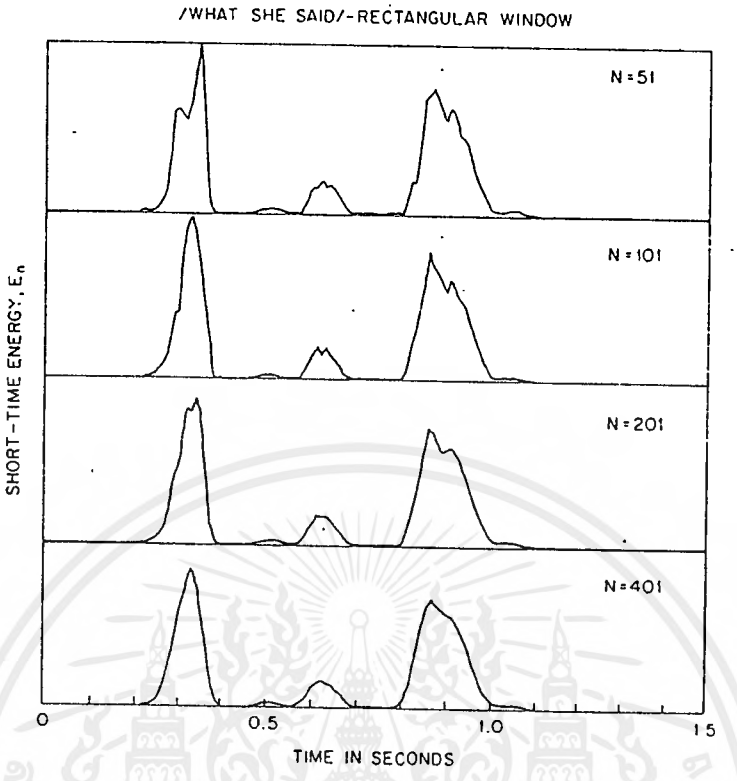


รูปที่ 2.2 แสดงบล็อกไดอะแกรม (a) พลังงานในช่วงเวลาสั้น (b) ขนาดในช่วงเวลาสั้น ๆ



รูปที่ 2.3 แสดงการแปลงฟูเรียร์ของ (a) วินโดว์สี่เหลี่ยม (b)วินโดว์แบบแฮมมิ่ง

จากรูปที่ 2.4 จะแสดงให้เห็นผลจากการเปลี่ยนแปลงค่า N ในวินโดว์ ซึ่งจะสังเกตได้ชัดเจนว่าเมื่อค่า N เพิ่มก็จะทำให้ค่าพลังงานในช่วงเวลาสั้น ๆ มีผลออกมามีความราบเรียบมากยิ่งขึ้น



รูปที่ 2.4 แสดงค่าพลังงานช่วงเวลาสั้น ๆ สำหรับวินโดว์แบบสี่เหลี่ยมเมื่อเปลี่ยนค่า N

สิ่งสำคัญที่ที่ได้จากค่า E_n นั้นคือ การที่เราสามารถแยกแยะส่วนของเสียงก้อง และเสียงไม่ก้อง ออกจากกันได้โดยส่วนของเสียงที่ไม่ก้อง นั้นจะมีค่า E_n ที่ต่ำกว่าเสียงก้องอย่างเห็นได้ชัด ซึ่งจะทำให้สามารถทำการกำหนดจุดอย่างคร่าว ๆ ได้ว่าตำแหน่งไหนเป็นจุดที่เสียงได้ทำการเปลี่ยนจากเสียงก้องเป็นเสียงไม่ก้องหรือในทางกลับกันและสำหรับสัญญาณที่มีค่า S/N สูง ๆ นั้นค่า E_n จะทำให้เราสามารถแยกเสียงพูดออกจากเสียงเงียบ(silence) ได้ด้วย

2.3 การหาอัตราการตัดศูนย์เฉลี่ย (Short-time Average Zerocrossing Rate)

สำหรับสัญญาณที่ไม่ต่อเนื่องในแกนเวลา(discrete-time) นั้น ค่าตัดศูนย์ คือ การเปลี่ยนของเครื่องหมายทางพีชคณิตของค่าสัญญาณ และ อัตราการเปลี่ยนเครื่องหมายหรืออัตราการตัดศูนย์นี้สามารถใช้บอกคุณสมบัติทางความถี่ของสัญญาณได้ โดยเฉพาะกับช่วงความถี่ที่แคบมาก ๆ ดังเช่น สัญญาณซายน์ที่มีความถี่ F_0 ที่ถูกแซมปลิ้งด้วยความถี่ F_s จะมี F_s/F_0 ตัวอย่าง ต่อหนึ่งคาบของสัญญาณซายน์ โดยแต่ละรอบจะมีค่าอัตราการตัดศูนย์เป็น 2 ดังนั้นสำหรับค่าเฉลี่ยในช่วงเวลายาวนานนั้นจะได้เป็น

$$Z = \frac{2F_0}{F_s} \qquad \text{Crossing/Sample}$$

ดังนั้นจะเห็นได้ว่าอัตราการตัดศูนย์เฉลี่ย นั้นมีประโยชน์มากในการคาดคะเนความถี่ของสัญญาณซาย์นนี้

จากการที่เสียงพูดมีย่านความถี่ที่กว้างพอสมควรจึงเป็นการยากที่จะตีความจากอัตราการตัดศูนย์เฉลี่ย แต่อย่างไรก็ตามการคาดคะเนอย่างหยาบ ๆ ของคุณลักษณะทางความถี่ของเสียงสามารถทำได้ โดยใช้วิธีการคำนวณหาค่าอัตราการตัดศูนย์เฉลี่ยในช่วงเวลาสั้น ๆ (Short-time average Zero-Crossing Rate) โดยสมการการคำนวณหาอัตราการตัดศูนย์เฉลี่ย เป็นดังนี้

$$Z_n = \sum_{n=-\infty}^{\infty} \left| Sgn[X(m)] - Sgn[X(m-1)] \right| W(n-m)$$

โดยที่

$$Sgn[X(n)] = 1 \qquad \text{เมื่อ} \qquad X(n) > 0$$
$$Sgn[X(n)] = -1 \qquad \text{เมื่อ} \qquad X(n) < 0$$

และ

$$W(n) = \frac{1}{2N} \qquad \text{เมื่อ} \qquad 0 < n < N-1$$
$$W(n) = 0 \qquad \text{อื่น ๆ}$$

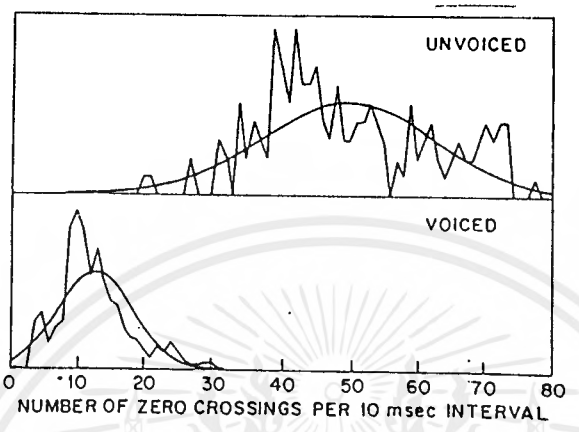
การหาค่าอัตราการตัดศูนย์เฉลี่ยนั้นสามารถแสดงเป็นบล็อกไดอะแกรมได้ตามรูปที่ 5

รูปที่ 2.5 แสดงบล็อกไดอะแกรมในการหาค่าอัตราการตัดศูนย์เฉลี่ยในช่วงเวลาสั้น

ในการนำค่าอัตราการตัดศูนย์เฉลี่ย ไปประยุกต์ใช้กับสัญญาณเสียงนั้นจากที่ว่าถ้าค่าอัตราการตัดศูนย์เฉลี่ยสูงแสดงว่าสัญญาณเสียงนั้นมีความถี่สูง แต่ถ้าได้ค่าอัตราการตัดศูนย์เฉลี่ยต่ำก็แสดงว่าสัญญาณเสียงนั้นมีความถี่ต่ำ โดยเสียงที่เป็นเสียงไม่ก้องจะมีความถี่สูงกว่าเสียงที่เป็นเสียงก้อง จึงสามารถกล่าวได้ว่าถ้าค่าอัตราการตัดศูนย์เฉลี่ยมีค่าสูงจะเป็นเสียงไม่ก้อง แต่ถ้าต่ำก็จะเป็นเสียงก้อง จากรูปที่ 2.6 จะเป็นการหาค่าอัตราการตัดศูนย์เฉลี่ยของทั้งสัญญาณเสียงก้องและ ไม่ก้องในช่วงเวลา 10 msec โดยการใช้ฟังก์ชั่นของเกาส์เซียน (Gaussian) โดยที่ค่าอัตราการตัดศูนย์เฉลี่ยของเสียงไม่ก้องนั้นจะมีค่า 49 ครั้งส่วนเสียงก้องนั้นจะมีค่าประมาณ 14 ครั้งในช่วงเวลา 10 msec อย่างไรก็ยังเป็นการยากที่ตัดสินลงไปว่าค่าอัตราการตัดศูนย์เฉลี่ยค่าใดเป็นค่าต่ำหรือ ค่าใดเป็นค่าสูง ดังนั้นในการแบ่งแยกว่าเป็นเสียงก้อง หรือเสียงไม่ก้อง จึงไม่เหมาะสมที่จะใช้ค่าอัตราการตัดศูนย์เฉลี่ยเพียงอย่างเดียวในการตัดสิน

ในการคำนวณค่าอัตราการตัดศูนย์เฉลี่ย นั้นเป็นการเพียงการเปรียบเทียบเครื่องหมายของสัญญาณเท่านั้น จุดที่ควรจะให้ความสนใจคือ กระบวนการของการแซมปลิ้ง เนื่องจากมันส่งผลกระทบต่อค่าอัตราการตัดศูนย์เฉลี่ยเป็นอย่างมาก จากการกำหนดค่า offset (ค่าที่อ้างอิงเป็นแกนศูนย์) หรือ 50 Hz Hum ที่เกิดสัญญาณ

power และสัญญาณรบกวน(noise) ที่อาจเกิดจากระบบ ดังนั้นจึงต้องให้ความสำคัญกับกระบวนการที่เป็นสัญญาณอนาล็อก เพื่อลดปัญหาต่าง ๆ เหล่านี้ อาจมีการใช้ฟิลเตอร์ที่ให้ความถี่ผ่านเป็นช่วง แทนที่จะใช้ฟิลเตอร์แบบความถี่ต่ำผ่าน เพื่อกำจัดส่วนที่เป็นสัญญาณ 50 Hz และอีกจุดหนึ่งคือ ช่วงเวลาในการแซมปลิ้ง ในการที่จะให้ได้ค่าอัตราการตัดศูนย์เฉลี่ยที่มีความแม่นยำสูง ก็จะต้องใช้อัตราการแซมปลิ้งที่สูงด้วย



รูปที่ 2.6 แสดงการแบ่งแยกของค่าอัตราการตัดศูนย์เฉลี่ยของเสียงก้องและไม่ก้อง

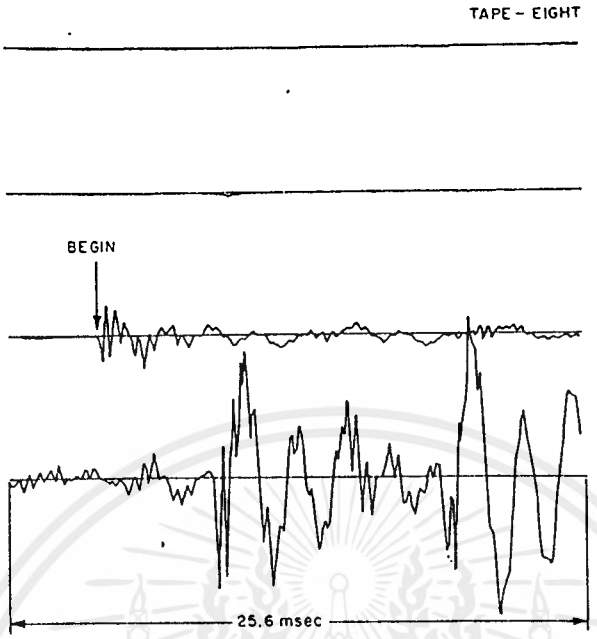
2.4 การแยกเสียงพูดออกจากเสียงเงียบ(silence) โดยใช้ค่าพลังงานและค่าอัตราการตัดศูนย์เฉลี่ย

ในระบบการประมวลผลสัญญาณเสียงพูดนั้นสิ่งสำคัญสิ่งหนึ่งคือ การแยกแยะ และการหาขอบเขตทั้งจุดเริ่มต้นและจุดสิ้นสุดของเสียง ออกจากสัญญาณรบกวน(Background Noise) โดยเฉพาะในกระบวนการรู้จำเสียงพูดชนิดคำโดด (isolated word) จำเป็นอย่างยิ่งที่จะต้องกำหนดจุดเริ่มต้นและ จุดสิ้นสุดของคำให้ได้อย่างมีประสิทธิภาพ

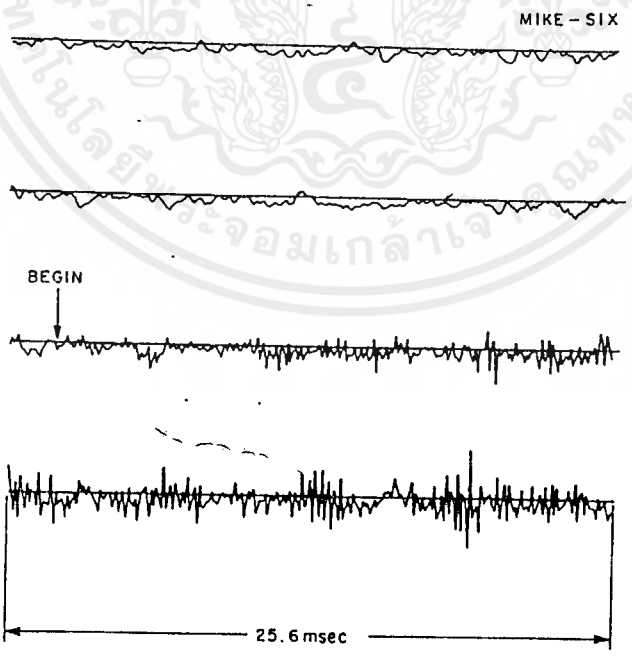
การใช้ค่าพลังงานและค่าอัตราการตัดศูนย์เฉลี่ย จากในรูปที่ 2.7 เป็นตัวอย่างแรก จะเห็นได้ว่า รูปคลื่นนี้สามารถแยกแยะเสียงออกจากสิ่งแวดล้อมได้ง่ายโดยใช้ค่าพลังงาน ส่วนอีกตัวอย่างหนึ่งในรูปที่ 2.8 เป็นเสียงที่ค่าพลังงานของเสียงใกล้เคียงกับสัญญาณรบกวนมาก ไม่สามารถแยกแยะออกได้ แต่เมื่อมาดูที่ค่าอัตราการตัดศูนย์เฉลี่ย จะมีความแตกต่างกับสัญญาณรบกวนมากพอสมควร

จากรูปที่ 2.9 เป็นตัวอย่างที่ยากมากที่จะทำการกำหนดจุดเริ่มต้นและสิ้นสุดของเสียง ซึ่งจุดเริ่มต้นของเสียง ที่มีลักษณะพ่นลมออกมา(fricative) ซึ่งจะทำให้ค่าพลังงานมีค่าต่ำมาก ซึ่งลักษณะของเสียงที่ยากต่อการกำหนดขอบเขต มีดังเช่น

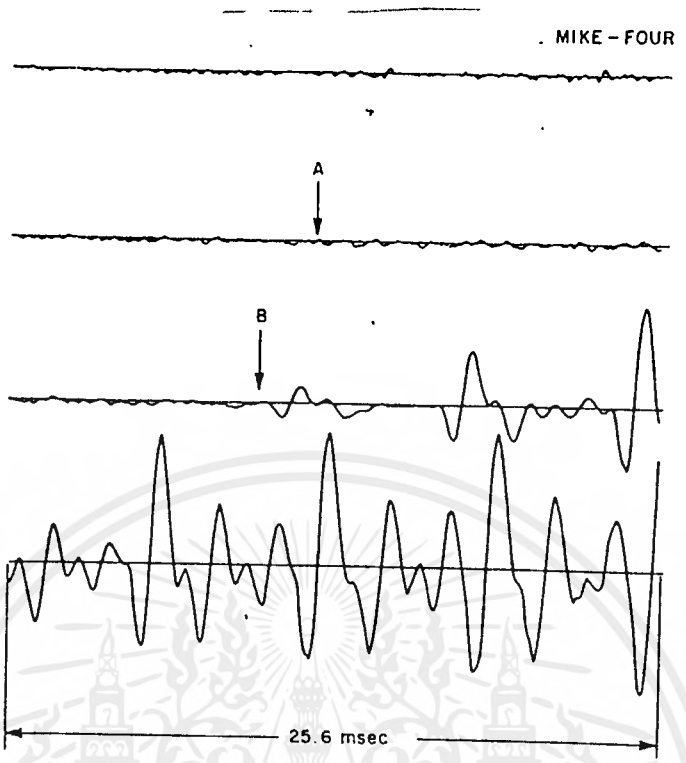
1. ลักษณะของเสียงที่พ่นลมออกมา ณ จุดเริ่มต้นหรือสิ้นสุด (Weak fricative)
2. ลักษณะของเสียงที่เป็นการระเบิดออกมา(Weak plosive bursts)
3. ลักษณะของเสียงที่เป็นเสียงนาสิก(Nasals)



รูปที่ 2.7 แสดงรูปคลื่นของจุดเริ่มต้นเสียงคำว่า /eignt/



รูปที่ 2.8 แสดงรูปคลื่นของจุดเริ่มต้นของเสียงคำว่า /six/



รูปที่ 2.9 แสดงรูปคลื่นของจุดเริ่มต้นเสียงคำว่า /four/

ในการที่จะกำหนดจุดเริ่มต้น และจุดสิ้นสุดของเสียงเหล่านี้ เราสามารถใช้กระบวนการของการหาค่าพลังงาน และอัตราการตัดศูนย์เฉลี่ย ร่วมกันได้ ตัวอย่างในระบบรู้จำเสียงพูดแบบคำโดด การบันทึกเสียงจะบันทึกส่วนที่เป็นเสียงพูดและส่วนที่ผู้พูดไม่ได้พูดไว้ด้วย แล้วใช้ระบบหาจุดเริ่มต้น และสิ้นสุดของคำเพื่อที่จะทำให้ผ่านเข้ากระบวนการรู้จำเสียงโดยส่งเฉพาะส่วนที่เป็นเสียงพูดเข้าไปประมวลผลต่อไป

การทำงานของระบบนี้จะใช้การหาค่าอัตราการตัดศูนย์เฉลี่ยที่มีขนาดเฟรมเท่ากับ 10 msec และการหาค่าพลังงาน และอัตราการตัดศูนย์เฉลี่ยร่วมกัน โดยใช้วินโดว์ขนาด 10 msec ซึ่งทั้ง 2 กระบวนการจะทำตลอดช่วงของสัญญาณด้วยอัตรา 100 ครั้งต่อวินาที โดยการสมมติให้ช่วงเวลา 100 msec แรกไม่มีเสียงพูด หมายความว่า ค่าอัตราการตัดศูนย์ และค่าพลังงานในช่วงนี้จะเป็นค่าของสัญญาณรบกวน และทำการคำนวณค่าพลังงาน และอัตราการตัดศูนย์เฉลี่ยที่ที่ใช้เป็นเกณฑ์ในการตัดสินใจค่าระดับของพลังงาน

ค่าระดับที่ใช้ส่วนมากจะให้ค่าโดยประมาณซึ่งในความจริงแล้วจุดเริ่มต้น และจุดสิ้นสุดนั้นจะอยู่ภายนอกจากค่าระดับนี้จึงมีการทำซ้ำโดยดูจากจุดที่กำหนดโดยค่าระดับในครั้งแรกและลดลงจนถึงค่า ระดับอีกค่าหนึ่งจึงถือว่าเป็นจุดเริ่มต้นและจุดสิ้นสุดของคำ

ขอบเขตที่ได้จากการใช้ค่าระดับค่าแรกนั้นเป็นที่น่าเชื่อถือว่าภายในขอบเขตนี้จะไม่มีจุดเริ่มต้นหรือสิ้นสุดของคำอยู่

กระบวนการถัดไปคือ การย้อนกลับจากจุดเริ่มต้นที่ได้จากค่าระดับค่าแรกแล้วทำการเปรียบเทียบค่าอัตราการตัดศูนย์เฉลี่ยกับค่าระดับการตัดศูนย์เฉลี่ยของมันซึ่งได้จากสัญญาณรบกวน โดยถ้าค่าอัตราการตัด

ศูนย์เฉลี่ยมีค่ามากกว่าค่าระดับเกิน 3 ครั้ง แล้วจะถือว่าจุดเริ่มต้นเป็นจุดที่ค่าอัตราการตัดศูนย์เฉลี่ยเกินค่าระดับเป็นครั้งแรก

2.5 การคำนวณหาค่าออโตคอรีเลชั่น(Auto Correlation function)

สมการทั่วไปของฟังก์ชั่นนี้คือ

$$\varphi(k) = \sum_{m=-\infty}^{\infty} X(m)X(m+k)$$

ในบางกรณีฟังก์ชั่นออโตคอรีเลชั่น สามารถอธิบายคุณลักษณะของเสียงได้อย่างดี ดังตัวอย่างเช่น สัญญาณที่เป็นคาบ มีความถี่คือ f จะสามารถแสดงได้ว่า

$$\varphi(k) = \varphi(k+f)$$

คุณลักษณะสำคัญของฟังก์ชั่นออโตคอรีเลชั่น คือ

1. เป็นฟังก์ชั่นคู่ $\varphi(k) = \varphi(-k)$
2. ฟังก์ชั่นจะให้ค่าสูงสุดที่ $k = 0; |\varphi(k)| \le \varphi(0)$ สำหรับทุกค่า k
3. ค่า $\varphi(0)$ นั้นมีค่าเท่ากับค่าพลังงานของสัญญาณจุดนั้น

จากสมการและคุณลักษณะข้างต้น เราเห็นได้ว่าสำหรับสัญญาณที่มีคาบนั้น ฟังก์ชั่นนี้จะให้ค่าสูงสุดที่ตำแหน่ง $0, \pm P, \pm 2P \ldots$ ดังนั้นเราจึงสามารถคาดคะเนคาบเวลาของสัญญาณโดยหาจากตำแหน่งของจุดสูงสุดแรกของฟังก์ชั่น ประโยชน์ข้อนี้เป็นที่น่าสนใจในการนำมาใช้หาคาบสัญญาณใด ๆ รวมถึงเสียงพูดด้วย

ฟังก์ชั่นออโตคอรีเลชั่นในช่วงเวลาสั้น ๆ (Short-time Autocorrelation function) เป็นการนำวิธีการของช่วงเวลาสั้น ๆ ที่กล่าวมาก่อนหน้านี้ มาใช้กับฟังก์ชั่นออโตคอรีเลชั่น โดยจะได้สมการเป็น

$$R_n(k) = \sum_{m=-\infty}^{\infty} X(m)W(n-m)X(m+k)W(n-k-m)$$

เนื่องจากฟังก์ชั่นออโตคอรีเลชั่นเป็นฟังก์ชั่นคู่ $R_n(k) = R_n(-k)$ สามารถเขียนสมการใหม่ได้เป็น

$$R_n(k) = \sum_{m=-\infty}^{\infty} [X(n+m)W'(m)][X(n+m+k)W'(k+m)]$$

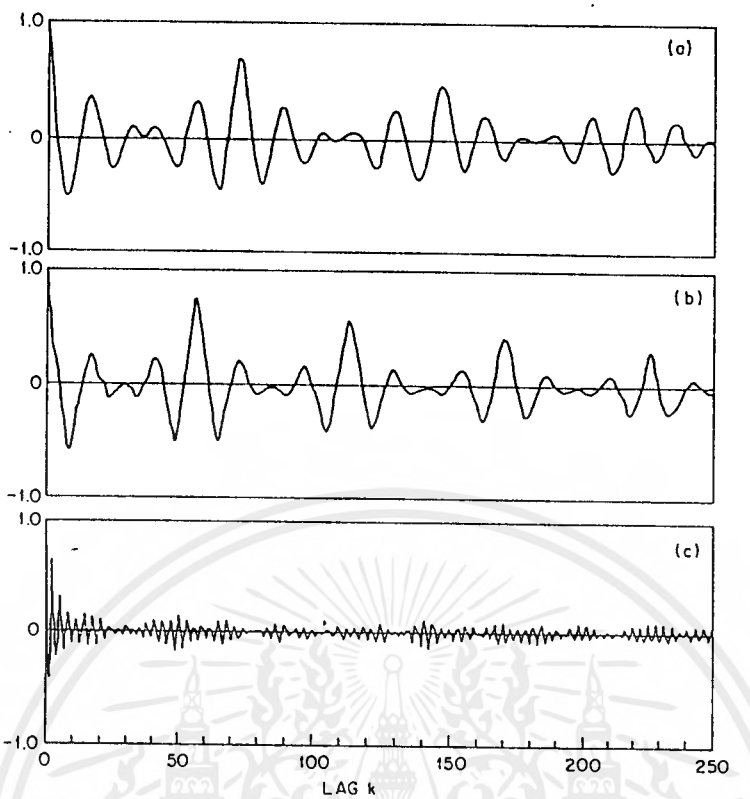
โดยที่ $W'(n) = W'(-n)$

จากสมการจะเห็นได้ว่าจุดเริ่มต้นของลำดับของสัญญาณจะถูกเลื่อนไป n ค่า แล้วถูกคูณด้วยวินโดว์ (W') เพื่อที่จะเลือกช่วงเวลาของเสียงพูดและถ้าวินโดว์ฟังก์ชั่น มีค่าจำกัดจะสามารถเขียนได้เป็น

$$R_n(k) = \sum_{m=0}^{N-1-k} [X(n+m)W'(m)][X(n+m+k)W'(m+k)]$$

การคำนวณค่าออโตคอรีเลชั่น อันดับที่ k โดยใช้สมการข้างต้นต้องมีการคูณกัน N ครั้งของ $X(n+m)W'(m)$ และมีการคูณกัน และบวกกัน $N-k$ ครั้งของ Lagged Product โดยการคำนวณ Lag มาก ๆ นั้นมีความจำเป็นต่อการคาดคะเนคาบสัญญาณ แต่ก็สามารถลดการคำนวณลงได้บ้างโดยใช้วิธีพิเศษอื่นๆ

ดังรูปตัวอย่างเป็นสัญญาณที่ผ่านฟังก์ชั่นออโตคอรีเลชั่น โดยมีค่า $N = 401$ โดยมีค่า Lag อยู่ระหว่าง $0 \le k \le 250$ และสัญญาณนั้นมีอัตราเสียงแซมปลิ้ง ที่ 10 kHz



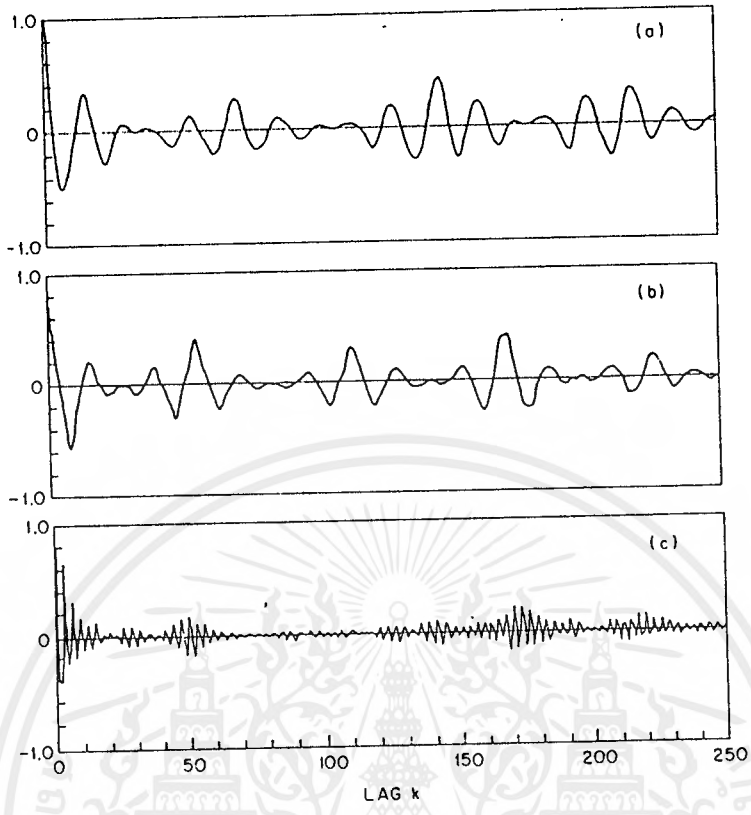
รูปที่ 2.10 แสดงค่าของฟังก์ชั่นออโตคอรีเลชั่นสำหรับ (a) และ (b) เป็นเสียงก้อง (c) เป็นเสียงไม่ก้องทั้ง 3 แบบจะใช้วินโดว์สี่เหลี่ยมและมีค่า N = 401

จารูปที่ 2.10 (a) และรูปที่ 2.10 (b) เป็นค่าฟังก์ชั่นออโตคอรีเลชั่นจะเป็นเสียงก้อง ส่วนอันสุดท้ายรูป 2.10 (c) เป็นของเสียงไม่ก้อง จากรูปจะได้ค่าสูงสุดของรูปแรกมีระยะห่างกัน 72 ค่า ดังนั้นจะมีความถี่มูลฐานประมาณ 140 Hz โดยค่าที่ได้นี้ จะมีค่าต่างจากค่าความถี่มูลฐานจริง ๆ เล็กน้อย เนื่องจากคาบเวลาของสัญญาณมีการเปลี่ยนแปลงภายในค่า 401 Sample และรูปคลื่นก็มีการเปลี่ยนแปลง และถ้าค่า Lag มีค่ามากขึ้นก็จะทำให้ค่าสูงสุด (peak) ลดลงด้วย ส่วนรูปที่สองมีระยะห่างของแต่ละ peak คือ 58 ดังนั้นจะมีคาบเวลา คือ 5.8 msec

ส่วนค่าฟังก์ชั่นออโตคอรีเลชั่น ของส่วนเสียงไม่ก้อง นั้นจะมีรูปคลื่นลักษณะเป็นความถี่สูงคล้ายกับสัญญาณรบกวน

รูปที่ 2.11 แสดงถึงการใช้วินโดว์แบบแฮมมิ่งแทนวินโดว์สี่เหลี่ยม เราจะเห็นได้ว่าการใช้วินโดว์สี่เหลี่ยม นั้นสามารถแยกแยะจุดสูงสุดได้ดีกว่า

จากตัวอย่างที่ใช้ค่า N=401 ปัญหาสำคัญคือ จะเลือกขนาดของ N ได้อย่างไรจึงได้ประสิทธิภาพดีที่สุด ถ้าคิดจากการเปลี่ยนแปลงของเสียงนั้นก็ควรจะใช้ N มีค่าน้อยที่สุดเพื่อจะได้คุณลักษณะของเสียงอย่างครบถ้วน แต่สิ่งที่จำเป็นก็คือ ภายในวินโดว์ นั้นต้องประกอบอย่างน้อย 2 คาบสัญญาณ และในการคำนวณนั้น ถ้าค่า k เพิ่มขึ้นมากเท่าไร ก็จะทำให้ข้อมูลที่ใช้คำนวณลดลงเรื่อย ๆ ซึ่งจะทำให้จุดสูงสุด ของค่าฟังก์ชั่นออโตคอรีเล



รูปที่ 2.11 แสดงฟังก์ชั่นออโตคอรีเลชั่นสำหรับ (a) และ (b) เป็นเสียงก้อง
(c) เป็นเสียงไม่ก้อง ซึ่งทั้งสามใช้วินโดว์แฮมมิ่งและค่า N = 401

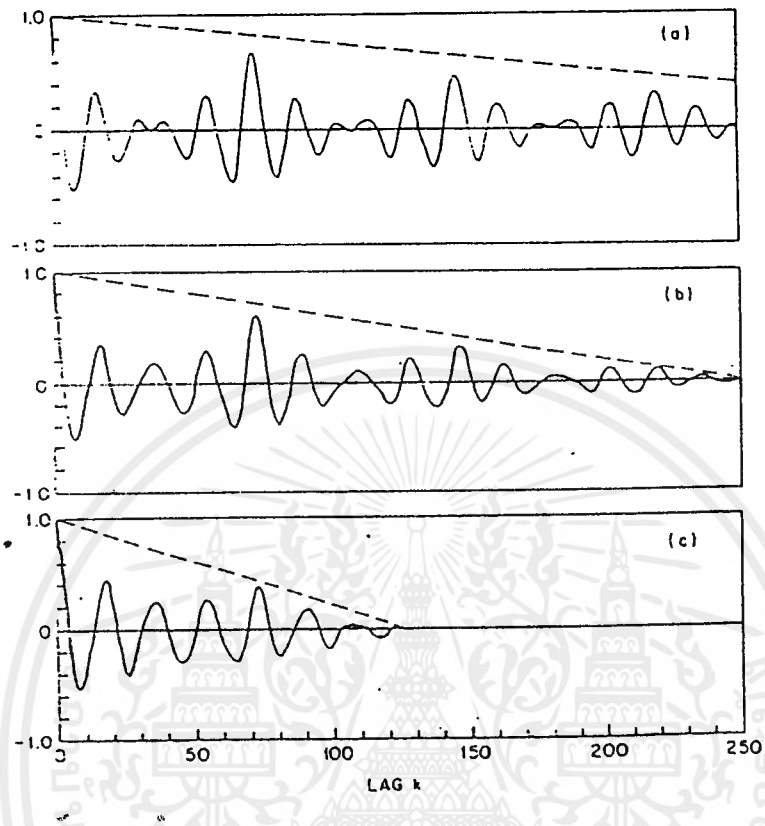
ชั่นลดลง ตัวอย่างดังรูปที่ 2.12 จะแสดงให้เห็นถึงค่าฟังก์ชั่นออโตคอรีเลชั่น เมื่อขนาดของวินโดว์ เปลี่ยนแปลงไป และส่วนของเส้นประแสดงถึงฟังก์ชั่น

$$R(k) = 1 - {}^{k}\!/_{N}\ , |k| < N$$

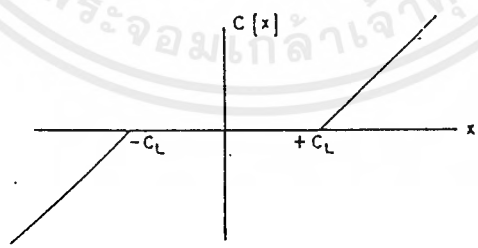
จากรูปจะเห็นว่าการที่ใช้ช่วงของวินโดว์ N=125 นั้นจะได้คาบของสัญญาณไม่ครบ 2 คาบจึงเป็นการไม่ดีนัก การที่จะหลีกเลี่ยงเหตุการเช่นนี้ อาจทำได้โดยการกำหนด N ให้มีค่ามาก ๆ พอเพียงสำหรับคาบเวลาพิทช์ที่มีค่ายาว ๆ แต่ก็จะเป็นการไม่ดีต่อสัญญาณที่มีคาบพิทช์สั้น ๆ จะเกิดการเฉลี่ยกันของค่าพิทช์มากเกินไป

2.6 การคาดคะเนคาบของพิทช์ (Pitch period) โดยใช้ฟังก์ชั่นออโตคอรีเลชั่น

การใช้เทคนิคที่เรียกว่า การคลิบตรงกลาง (Center Clipping) เป็นการแปลงแบบไม่เป็นเชิงเส้น มีรูปแบบของฟังก์ชั่นเป็น $y(n) = C[X(n)]$ โดย $C[\ ]$ มีลักษณะเป็นดังรูปที่ 13 และผลจากการกระทำ การคลิบตรงกลาง แสดงในรูปที่ 2.14



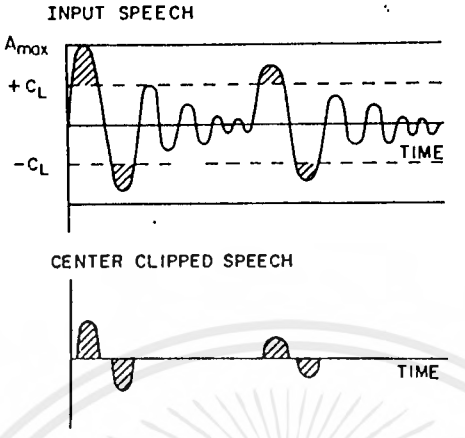
รูปที่ 2.12 แสดงฟังก์ชั่นออโตคอรีเลชั่นสำหรับเสียงก้องโดยที่ (a) N = 401 (b) N = 251 และ (c) N = 125 ซึ่งใช้วินโดว์แบบสี่เหลี่ยมหมด



รูปที่ 2.13 แสดงฟังก์ชันที่ใช้ในการคลิบตรงกลาง

จากรูปที่ 2.14 ส่วนของเสียงที่ถูกใช้ในการคำนวณถูกแสดงในรูปบนส่วนรูปล่างแสดงถึงจุดสูงสุด (peak) ที่มีค่าขนาด คือ A_{max} นั้นถูกคลิบ โดยระดับที่ใช้ในการคลิบ (C_L) โดยค่านี้ถูกกำหนดจากอัตราส่วนจากค่าสูงสุด (Sondhi ใช้ 30%) เอาท์พุทของการคลิบตรงกลาง นี้จะเท่ากับอินพุทลบกับ ระดับที่ใช้ในการคลิบ

(Clipping level) ก็ต่อเมื่อมันมีค่ามากกว่าระดับที่ใช้ในการคลิบ ส่วนที่มีค่าต่ำกว่าระดับที่ใช้ในการคลิบ นั้น เอาท์พุทของมันจะมีค่าเป็นศูนย์ โดยเอาท์พุทของรูปบนจะแสดงอยู่ในรูปล่าง



รูปที่ 2.14 รูปคลื่นที่ใช้ในการคลิบทั้งอินพุทและเอาท์พุท

ในการกำหนดค่าระดับที่ใช้ในการคลิบนั้นถ้าเรากำหนดค่าสูง ๆ ก็จะทำให้เป็นการง่ายในการคำนวณค่าฟังก์ชั่นออโตคอรีเลชั่น แต่ถ้าตั้งค่าไว้สูงเกินไป นั้นจะทำให้ค่าบางส่วนอยู่ต่ำกว่าระดับคลิบหมดทำให้สูญเสียข้อมูลไป แต่การที่จะตั้งค่าระดับคลิบให้ต่ำก็จะมีจุดยอด จำนวนมากที่ผ่านการคลิบมาได้ทำให้เพิ่มความสับส่นในการหาค่าฟังก์ชั่นออโตคอรีเลชั่น

Sondhi ได้กำหนด ระดับที่ใช้ในการคลิบไว้ที่ 30% ของค่าสูงสุด ส่วนวิธีที่ใช้ค่าเปอร์เซนต์สูงถึง 60-80% นั้นใช้สำหรับหาค่าจุดสูงสุดที่ต่ำสุดจากอันดับที่สามจากเริ่มต้น และจากสุดท้ายของวินโดว์ ปัญหาที่เกิดจากจุดสูงสุด ที่มีต้องการนั้นสามารถลดลงได้มากโดยการใช้การคลิบตรงกลางเพื่อที่จะคำนวณค่าฟังก์ชั่นออโตคอรีเลชั่น

อย่างไรก็ตามถึงแม้จะมีการใช้การคลิบร่วมด้วยในการคำนวณค่าฟังก์ชั่นออโตคอรีเลชั่น ก็ยังคงมีการคำนวณเป็นจำนวนมากอยู่ดี วิธีที่จะสามารถลดการคำนวณลง และไม่ส่งผลกระทบต่อการหาค่า พิทช์ คือ การกำหนดเอาท์พุทของกระบวนการคลิบ ใหม่ให้ง่ายขึ้น คือ ถ้า $X(n)\rangle -C_L$ แล้วจะได้เอาท์พุทเป็น +1 และถ้า $X(n)\langle -C_L$ แล้วเอาท์พุทจะเป็น -1 นอกนั้นเอาท์พุทจะเป็น 0 เรียกวิธีแบบนี้ว่า การคลิบ 3 ระดับ (3- level Center Clipping) ดังรูปที่ 2.15 แสดงเอาท์พุทของสัญญาณในรูป

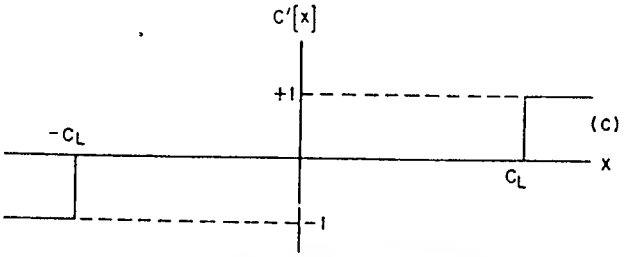
การคำนวณค่าฟังก์ชั่นออโตคอรีเลชั่น หลังจากการทำการคลิบแบบ 3 ระดับนั้นจะง่ายขึ้นดังสมการ ถ้าให้เอาท์พุทของการคลิบแบบ 3 ระดับเป็น $y(n)$ จะได้ว่า

$$R_n(k) = \sum_{m=0}^{N-k-1} y(n+m)y(n+m+k)$$

ซึ่งจะมีเพียง 3 กรณีที่เกิดขึ้นในผลคูณระหว่าง 2 พจน์เท่านั้นคือ

$$y(n+m)y(n+m+k) = 0 \text{ ถ้า } y(n+m) \text{ หรือ } y(n+m+k) = 0$$

$$y(n+m)y(n+m+k) = 1 \text{ ถ้า } y(n+m) = y(n+m+k)$$
$$y(n+m)y(n+m+k) = -1 \text{ ถ้า } y(n+m) \neq y(n+m+k) \neq 0$$

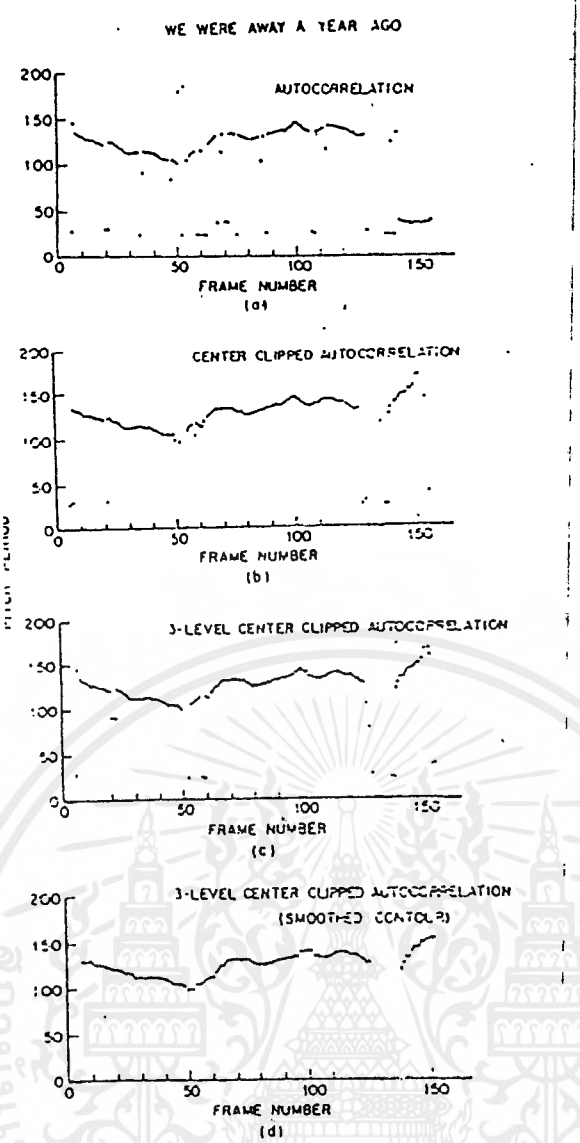


รูปที่ 2.15 แสดงฟังก์ชั่นที่ใช้ในการคลิบแบบ 3 ระดับ

ขั้นตอนในการคำนวณเพื่อที่จะหาค่าคาบเวลาของพิทช์จากสมการการคำนวณหาค่าฟังก์ชั่นออโตคอรีเลชั่นในช่วงเวลาสั้น ๆ มีดังนี้

1. สัญญาณเสียงจะนำไปผ่านตัวกรอง ซึ่งตัวกรองที่ใช้นั้นเป็นแบบความถี่ต่ำผ่านซึ่งมีความถี่คัทออฟ (cut-off) ที่ 900 Hz และนำไปทำการสุ่มตัวอย่างที่ความถี่ 10 kHz
2. กำหนดให้ขนาดของส่วนของเสียง (ขนาดwindow) เป็น 30 msec (300 ตัวอย่าง) และมีช่วงห่างแต่ละส่วนของเสียง 10 msec ดังนั้นแต่ละส่วนจะซ้อนกันอยู่ 20 msec
3. ทำการคำนวณค่าพลังงานเฉลี่ยใน 100 ตัวอย่างแรกของส่วนของเสียงแล้วนำค่าจุดสูงสุด ไปเปรียบเทียบกับค่าระดับที่กำหนดไว้โดยการวัดจากจุดสูงสุดของสัญญาณสัญญาณรบกวน ที่มีความยาว 50 msec ถ้าจุดสูงสุดของ 100 ตัวอย่างแรกมีค่าสูงกว่าก็จะถือว่าเป็นเสียงพูด แต่ถ้าไม่ก็จะไม่ถือว่าส่วนนั้นเป็นเสียงพูด
4. ระดับการคลิบจะถูกกำหนดจากค่าเปอร์เซนต์ของค่าต่ำสุดของ peak สูงสุดระหว่าง 100 ตัวอย่างแรก และ 100 ตัวอย่างหลังของส่วนของเสียง (ใช้ 68%)
5. ใช้ระดับการคลิบนี้ไปใช้กับการคลิบแบบ 3 ระดับและนำไปคำนวณค่าฟังก์ชั่นออโตคอรีเลชั่น
6. หาค่าสูงสุดของจุดสูงสุดของฟังก์ชั่นออโตคอรีเลชั่น และเปรียบเทียบกับค่าระดับ(30% ของ $R_n(0)$) ถ้าต่ำกว่าค่าระดับจะถือว่าเป็นเสียงไม่ก้อง และถ้ามากกว่าจะถือว่าจุดนั้นเป็นจุดสูงสุดที่ใหญ่ที่สุด

ในขั้นตอนที่ 4 และ 5 นั้นอาจจะใช้การคลิบแบบธรรมดา หรือการคลิบแบบ 3 ระดับ หรืออาจไม่ใช้การคลิบเลยก็ได้ ผลที่ได้จากการหาค่าพิทช์แสดงดังในรูปที่ 2.16 โดยรูปแรกเป็นการคำนวณฟังก์ชั่นออโตคอรีเลชั่น โดยไม่ใช้การคลิบแบบใด ๆ เลย จะสามารถสังเกตุ Lag เช่นได้ว่ามีการกระจายของความผิดพลาดมาก เนื่องจากความจริงที่ว่าจุดยอด ที่มีค่าสั้น ๆ นั้นจะมีขนาดใหญ่กว่าจุดสูงสุดที่คาบเวลาพิทช์ ในรูปถัดไปมีการใช้การคลิบตรงกลางและ การคลิบแบบ 3 ระดับ จะเห็นได้ว่าสามารถกำจัดความผิดพลาดออกไปได้มาก



รูปที่ 2.16 แสดงเอาท์พุทจากการหาพิทช์โดยใช้ฟังก์ชั่นออโตคอรีเลชั่น (a) ไม่มีการคลิบ (b) การคลิบแบบตรงกลาง (c) การคลิบแบบ 3 ระดับ (d) จากเอาท์พุทภาพ c ไปทำให้เรียบ

2.7 การอินเตอร์เฟซพื้นฐาน

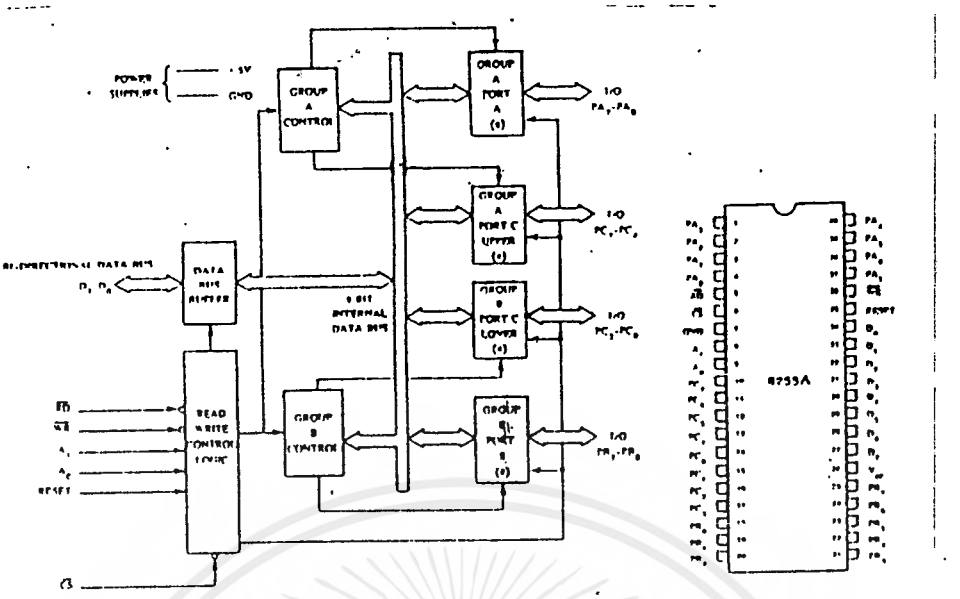
เครื่องคอมพิวเตอร์ทุกรุ่นจะมีหมายเลขพอร์ตสำหรับใช้งานต่าง ๆ ดังตารางที่ 2.2 จะเห็นว่ามีบางพอร์ตที่ไม่ได้ถูกใช้งาน (สงวนไว้) เช่น 360-36F, 3C0-3CF เราสามารถที่จะนำหมายเลขพอร์ตเหล่านี้ไปประยุกต์ใช้งานได้ หรือจะใช้งานหมายเลขพอร์ตที่ถูกกำหนดเอาไว้แล้ว แต่เครื่องไมโครคอมพิวเตอร์ของเรา

ไม่ได้ต่ออุปกรณ์ใช้งานกับพอร์ตนั้น เช่น พอร์ตหมายเลข 27F (เครื่องพิมพ์ขนานพอร์ต 2) เพราะโดยทั่วไปจะนิยมต่อเครื่องพิมพ์ที่พอร์ตหมายเลข 3BC หรือ 37F

2.8 8255 พอร์ตอินพุท/เอาท์พุทของระบบ (Programmable Peripheral Interface)

ตารางที่ 2.1 แสดงการจัดตำแหน่งพอร์ตของระบบ

หมายเลขพอร์ต	การใช้งาน
000-01F	ตัวควบคุมดีเอ็มเอ 1, 8237A-5
020-03F	ตัวควบคุมอินเตอร์รัปต์ 1, 8259 (มาสเตอร์)
040-05F	ตัวควบคุมไทเมอร์เคาน์เตอร์ 8254-2
060-06F	ตัวควบคุมพอร์ตขนานและคีย์บอร์ 8042
070-07F	Real Time Clock, NMI ของระบบ
080-09F	ดีเอ็มเอเพจรีจิสเตอร์ 74LS612
0A0-0BF	ตัวควบคุมอิมเตอร์รัปต์ 2, 8259 (สเลฟ)
0C0-0DF	ตัวควบคุมดีเอ็มเอ 2, 8237A-5
0F0	เคลียร์ไมโคโปรเซสเซอร์
0F1	รีเซตไมโคโปรเซสเซอร์
0F8-0FF	ไมโคโปรเซสเซอร์ 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	ไบต์ซิงโครไนซ์ 1
3B0-3BF	อะแดปเตอร์โมโนโครม และเครื่องพิมพ์
3C0-3CF	สงวนไว้
3D0-3DF	อะแดปเตอร์สี/กราฟฟิก
3F0-3F7	ตัวควบคุมดิสก์ไดรฟ์
3F8-3FF	พอร์ตอนุกรม 1



รูปที่ 2.17 แสดงแผนผังและการจัดขาของ 8255A-5

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

การใช้งานพอร์ตสามารถทำได้โดยการเขียนโปรแกรมควบคุมพอร์ตนั้น ๆ สามารถจะใช้ภาษาเบสิก ภาษาซี ภาษาปาสคาล และแอสเซมบลี ก่อนอื่นจะต้องทราบหมายเลขพอร์ตที่จะใช้งาน แล้วศึกษารายละเอียดของตัวอุปกรณ์ที่ทำงานอยู่ในพอร์ตนั้น ว่ามีการทำงานอย่างไร จากนั้นจึงเขียนโปรแกรมควบคุมการทำงานของพอร์ตนั้น บนเครื่องไมโครคอมพิวเตอร์จะมีพอร์ตที่ทำหน้าที่ควบคุมการรับคีย์บอร์ดสแกนโค้ดเป็นพอร์ตขนาน แล้วยังผลิตเสียงได้ด้วย คือพอร์ตหมายเลข 060-06F พอร์ตนี้จะใช้ชิป 8255 ในการทำงาน

ตารางที่ 2.2 หน้าที่และขาสัญญาณต่าง ๆ ของ 8255

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 เพื่อใช้เขียนข้อมูลภายในพอร์ตของ 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 เป็นอินพุตทั้งหมด

2.9 การโปรแกรม 8255

การใช้งาน 8255 จะต้องทำการโปรแกรมเสียก่อน โดยการส่งค่าคอนโทรลไบต์ให้แก่พอร์ตควบคุมจะเป็นคำสั่งขนาด 8 บิท หรือ 1 ไบต์ ซึ่งแต่ละบิทจะมีความหมาย และใช้งานต่างกัน คอนโทรลไบต์นี้จะเป็นคำสั่งกำหนดโหมดการทำงานของ 8255 และการกำหนดให้พอร์ตทั้ง 3 (A,B,C) เป็นอินพุท หรือเอาท์พุท ดังแสดงในรูปที่ 2.18

บิท D0 = ข้อมูลในบิทนี้ กำหนดให้พอร์ต C ล่าง (PC0 - PC3) เป็นอินพุทหรือเอาท์พุท ถ้าบิทนี้เป็น 1 จะเป็นอินพุท แต่ถ้าเป็น 0 จะเป็นเอาท์พุท

บิท D1 = ข้อมูลในบิทนี้ กำหนดให้พอร์ต B เป็นอินพุทหรือเอาท์พุท ถ้าบิทนี้เป็น 1 จะเป็นอินพุท แต่ถ้าเป็น 0 จะเป็นเอาท์พุท

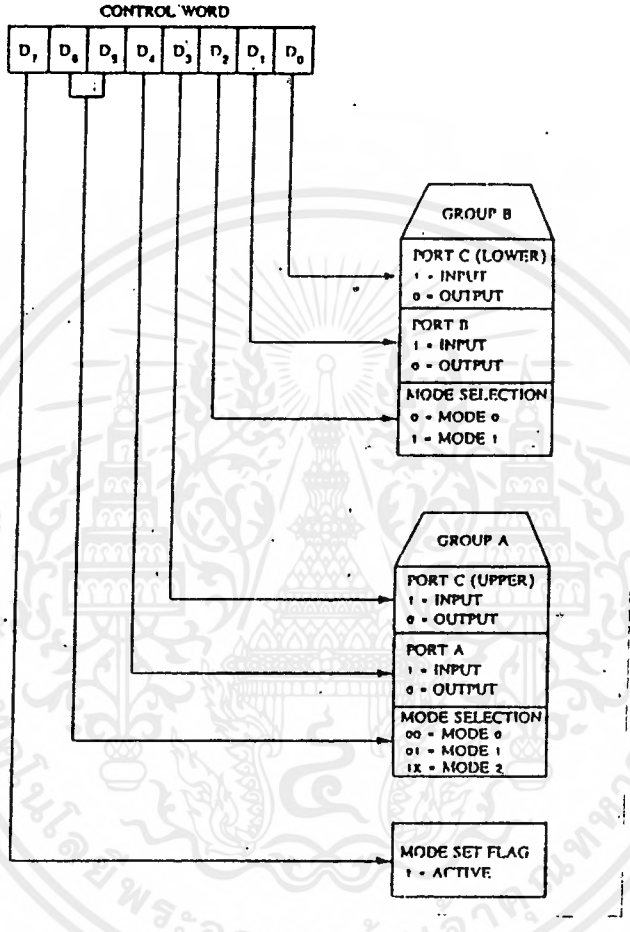
บิท D2 = ข้อมูลในบิทนี้ กำหนดการเลือกโหมดของกลุ่ม B ถ้าบิทนี้เป็น 1 จะทำงานในโหมด 1 ถ้าบิทนี้เป็น 0 จะทำงานในโหมด 0

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

บิท D4 = ข้อมูลในบิทนี้กำหนดให้พอร์ต 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



รูปที่ 2.18 แสดงรูปแบบการกำหนดค่าของคอนโทรลไบต์

2.10 โหมดการทำงานของ 8255

8255 มีโหมดการทำงานอยู่ 3 โหมดด้วยกัน คือ

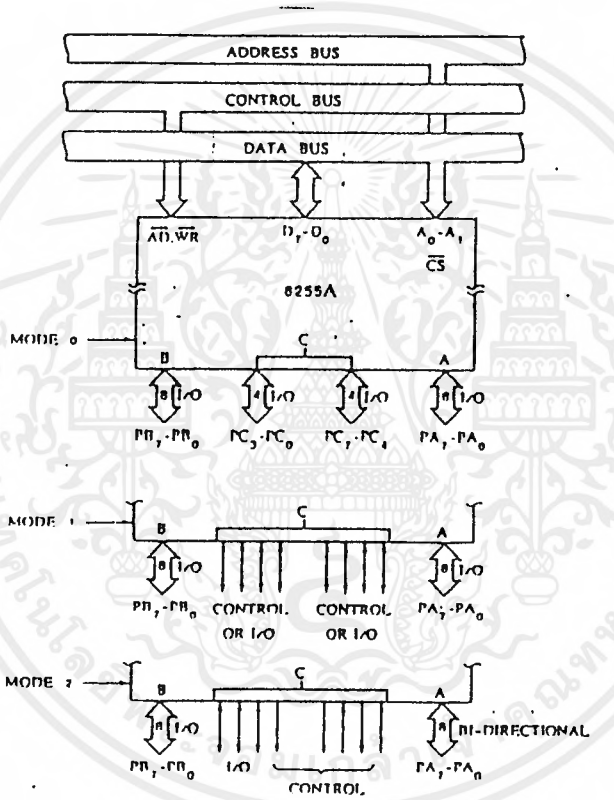
1. โหมด 0 (อินพุท/เอาท์พุทพื้นฐาน)
2. โหมด 1 (อินพุท/เอาท์พุทสโตรบ)
3. โหมด 2 (บัสแบบสองทิศทาง)

โหมด 0 (โหมดอินพุท/เอาท์พุทพื้นฐาน)

ในโหมด 0 นี้จะกำหนดให้พอร์ต A, B และ C เป็นอินพุท/เอาท์พุทก็ได้ ไม่มีสัญญาณแฮนด์เชกกิ้ง (Handshaking) เอาท์พุทที่ออกจากพอร์ต A, B และ C จะแลตช์ (Latched) ค่าไว้ด้วย สามารถจัดรูปแบบของอินพุท/เอาท์พุตได้ 16 แบบด้วยกันดังรูปที่ 2.19

โหมด 1 (โหมดอินพุท/เอาท์พุทสโตรบ)

การทำงานในโหมด 1 มีการตรวจสอบสัญญาณแฮนด์เชกกิ้ง คือเป็นการทำงานระหว่างอุปกรณ์ภายนอกซีพียู และชิป 8255 โดยให้พอร์ต 8255 เป็นตัวกลางในการทำงาน สามารถใช้งานได้ 2 พอร์ต คือพอร์ต A และพอร์ต B เป็นได้ทั้งอินพุทและเอาท์พุท โดยใช้พอร์ต C บน (PC4 - PC7) ทำการตรวจสอบสัญญาณแฮนด์เชกกิ้งของพอร์ต A และใช้พอร์ต C ล่าง (PC0 - PC3) คอยตรวจสอบสัญญาณแฮนด์เชกกิ้งของพอร์ต B



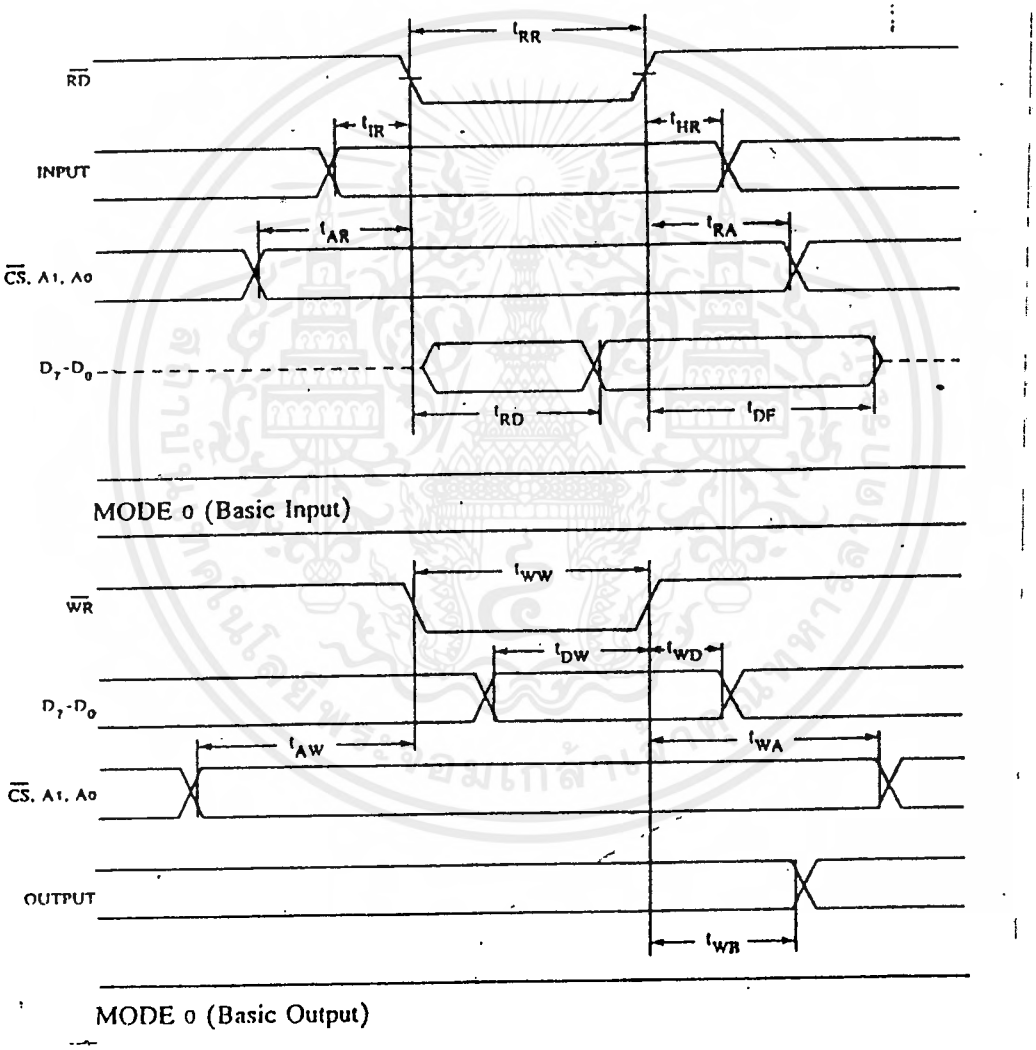
รูปที่ 2.19 แสดงโหมดการทำงานทั้ง 3 พอร์ต และการเชื่อมต่อระบบบัส

การใช้งาน 8255 ในโหมด 1 นี้ จะมีประโยชน์ในกรณีที่อุปกรณ์ภายนอกไม่สามารถจะทำงานได้เร็วทันการทำงานของซีพียู และเพื่อให้การรับส่งของข้อมูลเป็นไปอย่างถูกต้อง จากรูปที่ 2.21 จะเห็นว่า 8255 ส่งสัญญาณอินเตอร์รัปต์ (INTR) ไปให้ซีพียู เพื่อให้ซีพียูทำการอ่าน หรือเขียนข้อมูลกับอุปกรณ์ภายนอก

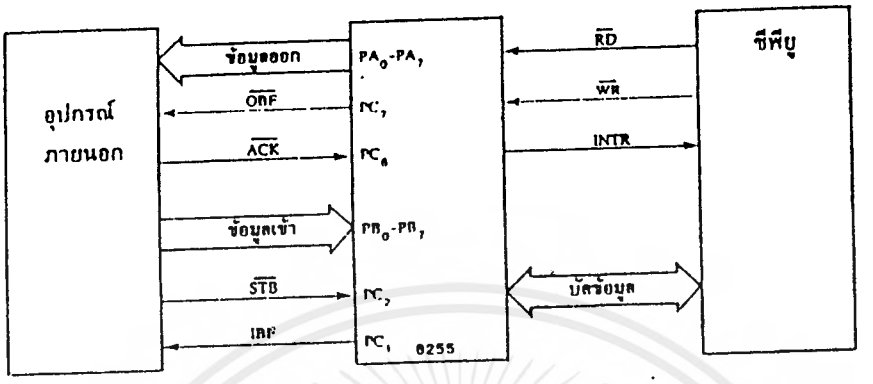
เมื่ออุปกรณ์ภายนอกต้องการจะส่งข้อมูลให้ซีพียูส่งสัญญาณ STB (Strobe Input) มาให้ชิป 8255 (รูปที่ 2.22) พร้อมทั้งส่งข้อมูลเข้าไปเก็บไว้ในพอร์ต จากนั้นชิป 8255 จะส่งสัญญาณ IBF (Input Buffer Full) ไปยังอุปกรณ์ภายนอกเพื่อไม่ให้ส่งข้อมูลเข้าไปอีก และ 8255 ก็จะส่งสัญญาณ $INTR_B$ ไปให้ซีพียู จากนั้นซีพียูจะส่ง

สัญญาณ RD เพื่อทำการอ่านข้อมูลจากพอร์ต B เมื่ออ่านเสร็จแล้วสัญญาณ $INTR_B$ จะไม่ทำงาน และสัญญาณ IBF จะไม่ทำงานด้วย เป็นการบอกให้อุปกรณ์ภายนอกส่งข้อมูลชุดใหม่มาได้

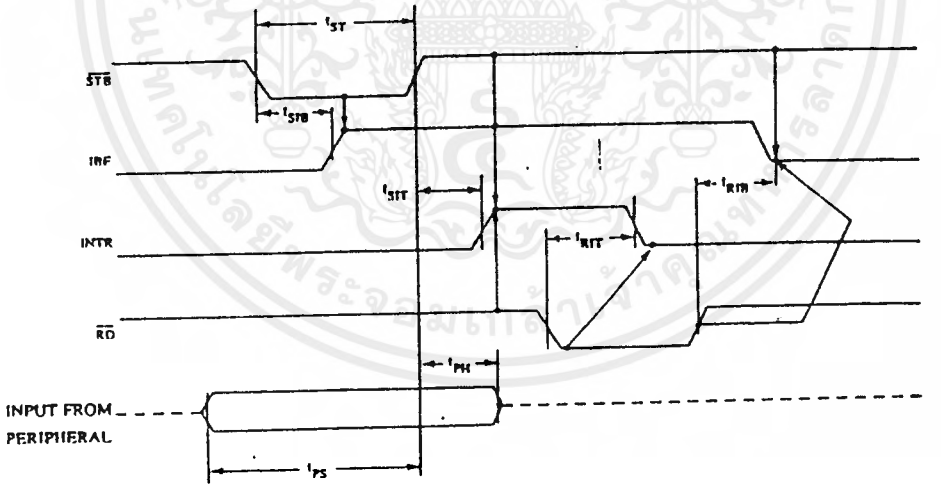
ถ้าซีพียูต้องการจะส่งข้อมูลออกไปให้อุปกรณ์ภายนอกสามารถทำได้โดยส่งสัญญาณ WR ออกมา (รูปที่ 2.23) พร้อมทั้งข้อมูล 8255 จะรับข้อมูลจากซีพียูไว้ในพอร์ต แล้วส่งสัญญาณ OBF (Output Buffer Full) ให้อุปกรณ์ภายนอกรับข้อมูลไปได้ เมื่ออุปกรณ์ภายนอกรับข้อมูลไปแล้วจะส่งสัญญาณ ACK (Acknowledge) ไปให้ 8255 ว่ารับข้อมูลเรียบร้อยแล้ว ทำให้สัญญาณ $INTR_A$ ทำงาน เพื่อให้ซีพียูส่งข้อมูลชุดใหม่ออกไป



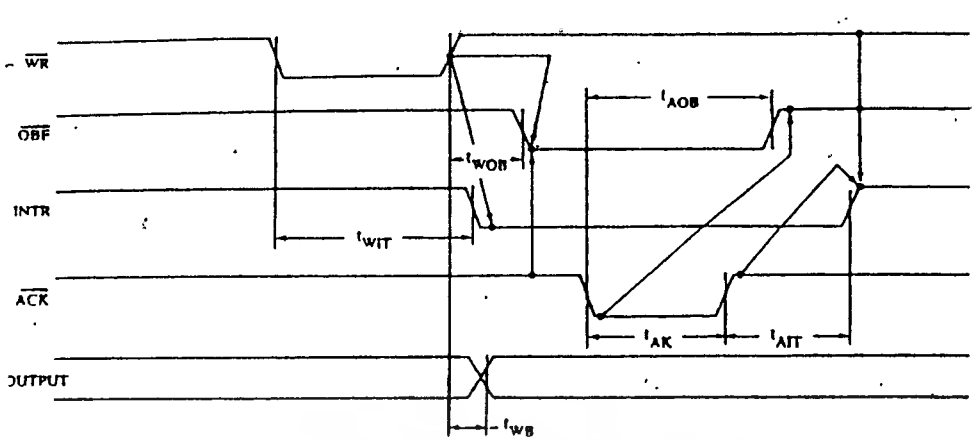
รูปที่ 2.20 แสดงบัสไซเคิลในโหมด 0 ของ 8255



รูปที่ 2.21 แสดงการใช้งานในโหมด 1 ของชิป 8255



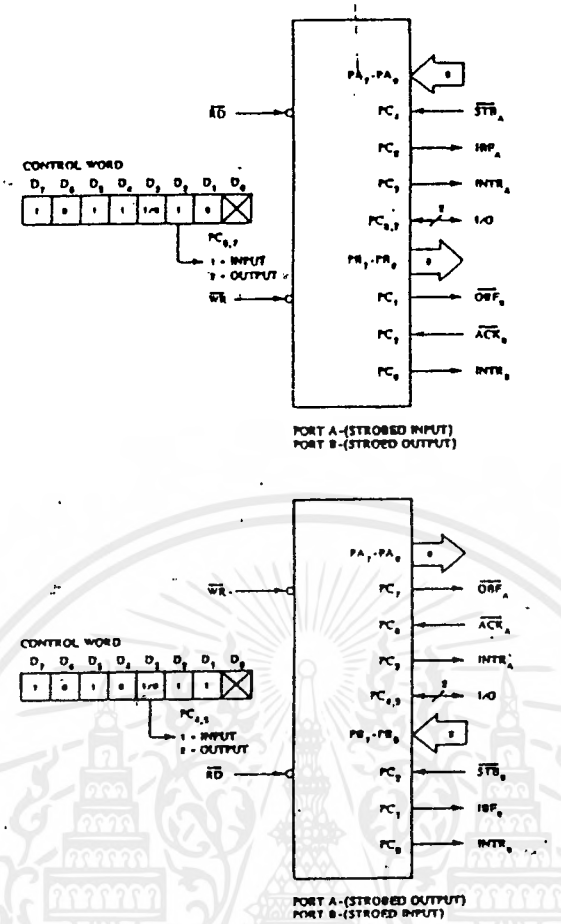
รูปที่ 2.22 แสดงบัสไซเคิลของการรูปที่แฮนด์เชกกิ้งขณะเป็นอินพุทพอร์ต



รูปที่ 2.23 แสดงบัสไซเคิลของการแฮนด์เชกกิ้งขณะเป็นเอาท์พุทพอร์ต

ตารางที่ 2.3 แสดงหน้าที่สัญญาณของพอร์ต C ในโหมด 1

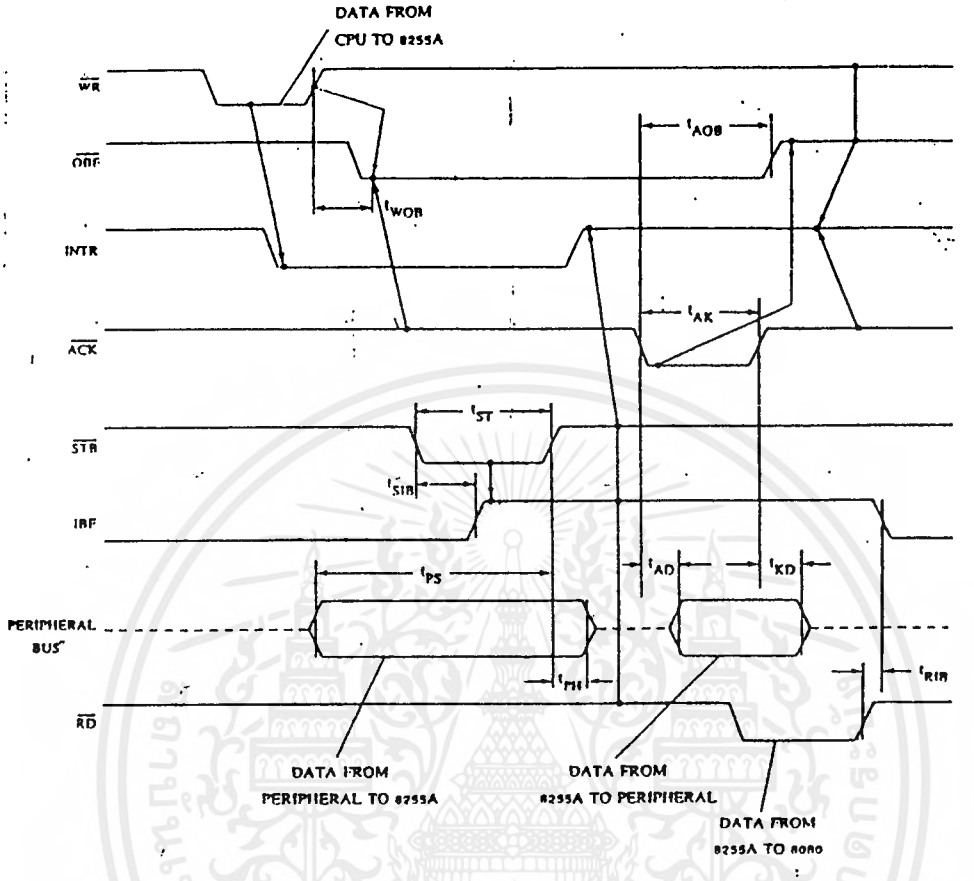
ขา	อินพุท	เอาท์พุท
Pc0	INTRB	INTRB
Pc1	IBFB	OBFB
Pc2	STBB	ACKB
Pc3	INTRA	INTRA
Pc4	STBA	I/O
Pc5	IBFA	I/O
Pc6	I/O	ACKA
Pc7	I/O	OBFA



รูปที่ 2.24 แสดงการจัดให้พอร์ต A และพอร์ต B เป็นอินพุท/เอาท์พุทพอร์ตในโหมด 1

โหมด 2 (บัสแบบสองทิศทาง)

โหมดสองนี้ยังคงใช้สัญญาณการตรวจสอบแฮนด์เชกกิ้งอยู่ เป็นบัสแบบสองทิศทาง จัดได้เฉพาะพอร์ต A โดยให้พอร์ต A เป็นทั้งอินพุทและเอาท์พุท พอร์ตจะใช้งานเกี่ยวกับการสื่อสารระหว่างอุปกรณ์ 2 ตัวที่สลับกันรับ/ส่งข้อมูล และใช้พอร์ต C ขนาด 5 บิท ในการตรวจสอบสัญญาณ สัญญาณที่เข้า/ออกจากพอร์ต A จะแลชต์ค่าไว้ด้วย การทำงานจะคล้าย ๆ กับโหมด 1 ของชิป 8255



รูปที่ 2.25 แสดงบัสไซเคิลของการแฮนด์เชกกิ้งในโหมด 2

บทที่ 3

การคำนวณ และการทำงานของโปรแกรมตัดเสียงพูด

3.1 โปรแกรมสั่งงานด้วยเสียง

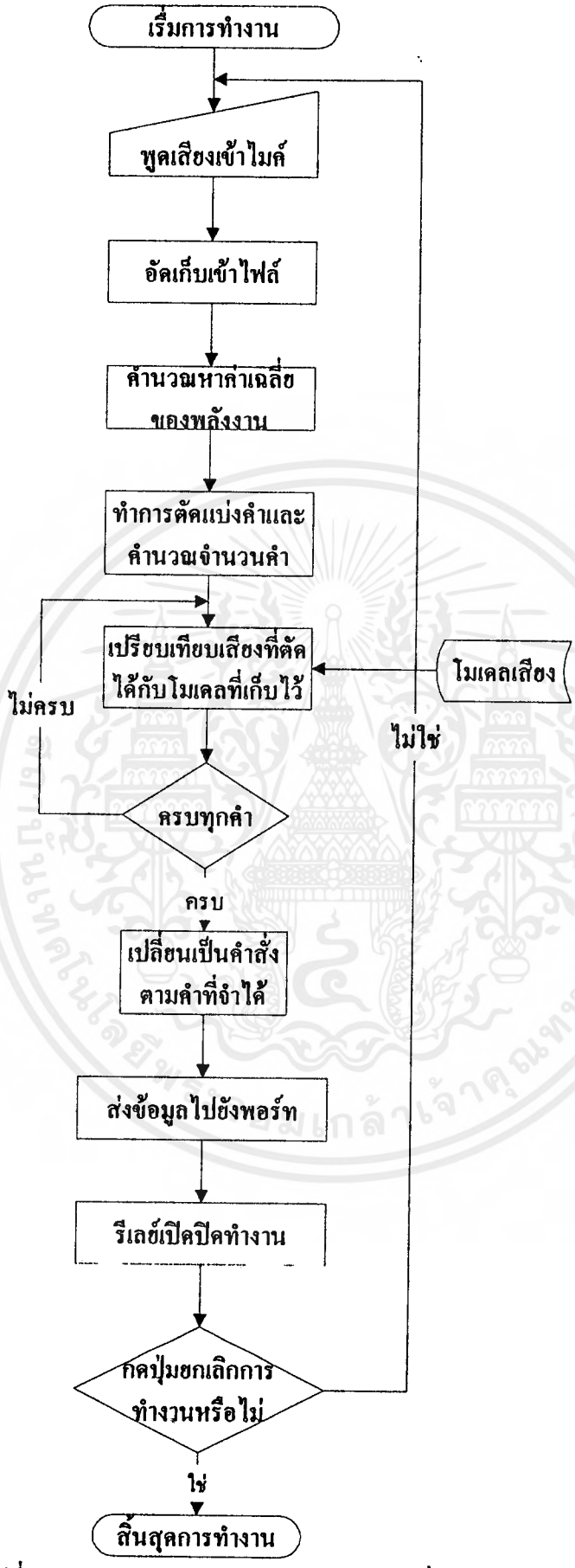
โปรแกรมนี้จะเป็นโปรแกรมที่ใช้งานจริง ในการพูดสัญญาณเสียงผ่านไมโครโฟน ที่ต่ออยู่กับคอมพิวเตอร์ เพื่อที่จะสั่งงานให้ออกพอร์ทต่าง ๆ ซึ่งแสดงแผนผังขั้นตอนการทำงานเอาไว้ตามรูปที่ 3.1 โดยที่จะสามารถอธิบายขั้นตอนการทำงานของแต่ละขั้นได้ดังนี้

1. เมื่อโปรแกรมนี้อยู่ในสภาวะที่ถูกเรียกขึ้นมาใช้งานแล้ว ก็จะต้องทำการพูดเสียงผ่านไมโครโฟนที่ต่ออยู่กับคอมพิวเตอร์
2. เสียงพูดที่พูดผ่านไมโครโฟนนั้น ก็จะถูกบันทึกเก็บเอาไว้เพื่อที่จะสามารถนำสัญญาณเสียงพูดที่พูดนั้นสามารถไปทำกระบวนการต่าง ๆ ต่อไปได้
3. เสียงพูดที่ถูกบันทึกเก็บเอาไว้นั้น จะถูกนำไปหาค่าพลังงานในช่วงเวลาสั้น ๆ (short time average Energy)
4. เมื่อได้ค่าพลังงานของสัญญาณเสียงที่พูดเข้าไปแล้ว ก็จะนำค่าพลังงานที่หามาได้ นำไปหาขอบเขตของคำ เพื่อใช้ในการตัดแบ่งคำออกมา
5. เมื่อสามารถหาขอบเขตของคำได้แล้ว ก็จะนำคำพูดที่บันทึกไว้ไปผ่านกระบวนการในการหาโค้ดบุ๊คอินเด็ก เพื่อที่จะนำไปเปรียบเทียบกับโมเดลเสียงที่ถูกสร้างเอาไว้อยู่ก่อน ว่าตรงกับโมเดลเสียงของเสียงไหน
6. ทำการเปรียบเทียบเสียงสัญญาณที่บันทึกไว้ จนหมดทุกคำที่แบ่ง
7. สัญญาณเสียงเมื่อผ่านกระบวนการเปรียบเทียบโมเดลเสียงแล้ว ก็จะถูกแปลงเป็นคำสั่ง ตามผลการเปรียบเทียบของสัญญาณเสียงกับโมเดลเสียง
8. คำสั่งที่ได้ จะถูกส่งผ่านไปยังการ์ด ออกมายังพอร์ท เพื่อใช้ในการควบคุมให้รีเลย์ทำงานเปิดปิด
9. ถ้าไม่กดปุ่มยกเลิกการทำงาน ก็จะกลับไปเริ่มขั้นตอนรอรับสัญญาณเสียงพูด
10. เมื่อกดปุ่มยกเลิกการทำงานก็เป็นอันว่าจบโปรแกรม

3.2 โปรแกรมหาค่าพลังงานเพื่อใช้ในการกำหนดขอบเขตคำ

โปรแกรมนี้จะใช้เพื่อหาค่าพลังงาน เพื่อใช้ในการกำหนดขอบเขตคำ โดยมีขั้นตอนซึ่งแสดงตามแผนผังขั้นตอนตามรูปที่ 3.2 โดยมีขั้นตอนดังนี้

1. กำหนดค่าพารามิเตอร์ต่าง ๆ เช่น ขนาดเฟรม ขนาดความกว้างที่ใช้ในการเลื่อนเฟรม โดยค่าพารามิเตอร์จะถูกนำไปใช้ในการหาค่าพลังงานของสัญญาณเสียงที่ป้อนเข้าไป
2. อ่านแฟ้มข้อมูลสัญญาณเสียง
3. คำนวณหาจำนวนเฟรม ว่าในสัญญาณเสียงที่ป้อนเข้ามา ว่ามีทั้งหมดกี่เฟรม
4. คำนวณหาค่าพลังงานในแต่ละเฟรม ซึ่งจะต้องทำจนครบทุกเฟรม
5. ทำการหาค่าพลังงานเฉลี่ยจากทั้งไฟล์เสียงเพื่อที่จะนำค่าพลังงานเฉลี่ยไปช่วยในการกำหนดขอบเขตคำอย่างคร่าว ๆ
6. ทำการกำหนดขอบเขตของคำโดยใช้ค่าพลังงานเฉลี่ยซึ่งการใช้ค่าพลังงานเฉลี่ยในการกำหนดขอบเขตของคำนั้น จะไม่สามารถครอบคลุมขอบเขตคำได้ทั้งหมด
7. ทำการหาค่าพลังงานเฉลี่ยระดับต่ำ เพื่อช่วยในการขยายขอบเขตคำ ให้สามารถครอบคลุมขอบเขตของคำได้จนเกือบหมด
8. ทำการขยายขอบเขตของคำโดยใช้ค่าพลังงานเฉลี่ยระดับต่ำ
9. นำขอบเขตของคำที่ได้ มาใช้ในการตัดแบ่งคำ และบันทึกเก็บไว้
10. สิ้นสุดการทำงาน



รูปที่ 3.1 แสดงแผนผังการทำงานของโปรแกรมสั่งงานด้วยเสียง

รูปที่ 3.2 แผนผังแสดงขั้นตอนการทำงานของโปรแกรมหาค่าพลังงานเฉลี่ย

3.3 โปรแกรมคำนวณหาค่าพิทช์

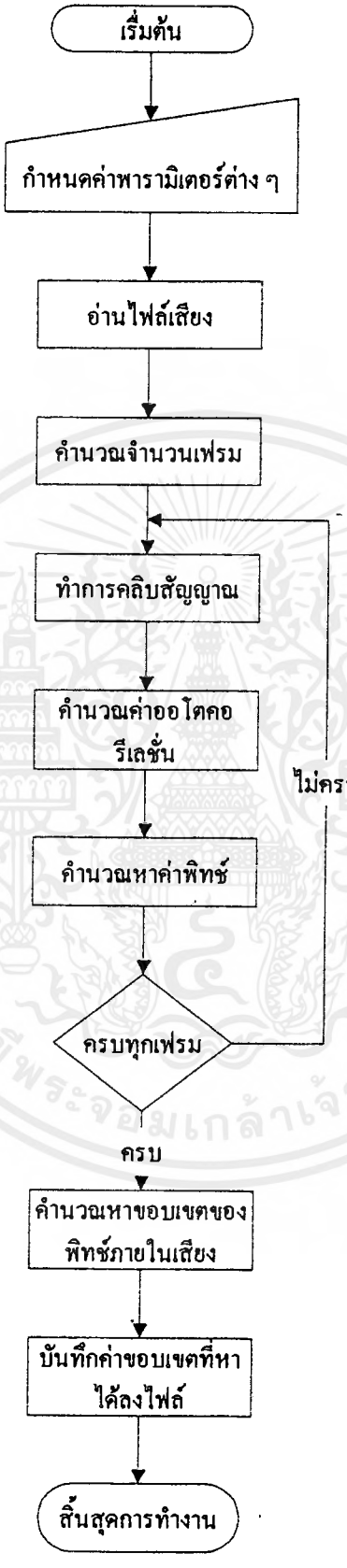
โปรแกรมคำนวณหาค่าพิทช์ เพื่อที่ใช้ในการกำหนดขอบเขตของคำ จะมีแผนผังแสดงการทำงานของโปรแกรมตามรูปที่ 3.3 ซึ่งสามารถอธิบายขั้นตอนต่าง ๆ ได้ดังนี้

1. กำหนดค่าพารามิเตอร์ต่าง ๆ เช่น ค่าระดับสัญญาณที่ใช้ในการคลิบ
2. ทำการอ่านแฟ้มข้อมูลสัญญาณเสียง
3. คำนวณหาจำนวนเฟรมทั้งหมด ที่ต้องทำการคำนวณ
4. ทำการคลิบสัญญาณ โดยขั้นตอนนี้จะใช้วิธีการคลิบแบบ 3 ระดับเพราะว่าค่าที่ใช้ในการคำนวณในขั้นตอนต่อไป จะสามารถทำการคำนวณได้ง่าย และรวดเร็วกว่าและผลที่ออกมาก็ชัดเจนดีด้วย ซึ่งจะต้องทำการคลิบจนครบทุกเฟรม
5. ทำการหาค่าออโตคอรีเลชั่นทุกเฟรม
6. ทำการหาค่าพิทช์ โดยอาศัยวิธีการตรวจดูจุดสูงสุด โดยจุดสูงสุดถัดไปจะต้องมีค่ามากกว่าค่า 30 % ของพีคก่อนหน้านี้ ทุกเฟรม
7. นำค่าพิทช์ที่ได้มาตรวจดูเพื่อหาขอบเขตของสัญญาณเสียง
8. เมื่อได้ขอบเขตของสัญญาณเสียงจากค่าพิทช์ แล้วก็จะนำค่าขอบเขตที่ได้มาทำการบันทึกเก็บลงไฟล์ต่อไป
9. สิ้นสุดการทำงาน

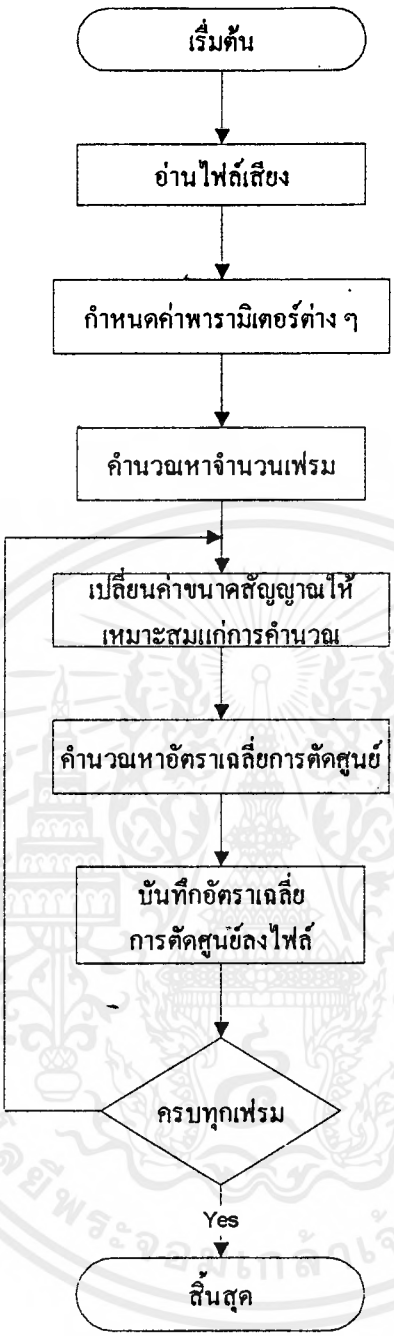
3.4 โปรแกรมหาอัตราเฉลี่ยการตัดศูนย์

โปรแกรมนี้จะใช้เพื่อหาอัตราเฉลี่ยการตัดศูนย์ โดยมีขั้นตอนตามรูปที่ 3.4 โดยอธิบายได้ตามขั้นตอนดังนี้

1. ทำการอ่านแฟ้มข้อมูลสัญญาณเสียง
2. กำหนดขนาดความกว้างของเฟรมที่ใช้สำหรับแบ่งสัญญาณเสียงออกเป็นส่วน ๆ
3. คำนวณหาจำนวนเฟรมที่เกิดขึ้นได้ทั้งหมด จากจำนวนตัวอย่างที่มีอยู่ในสัญญาณเสียง
4. ทำการคำนวณหาอัตราเฉลี่ยการตัดศูนย์
5. ทำการบันทึกเก็บอัตราเฉลี่ยการตัดศูนย์



รูปที่ 3.3 แผนผังแสดงขั้นตอนการทำงานของโปรแกรมคำนวณหาพิทช์



รูปที่ 3.4 แผนผังแสดงขั้นตอนการทำงานของโปรแกรมอัตราเฉลี่ยการตัดศูนย์

3.5 โปรแกรมเรียนรู้เสียง หรือสร้างโมเดลเสียง

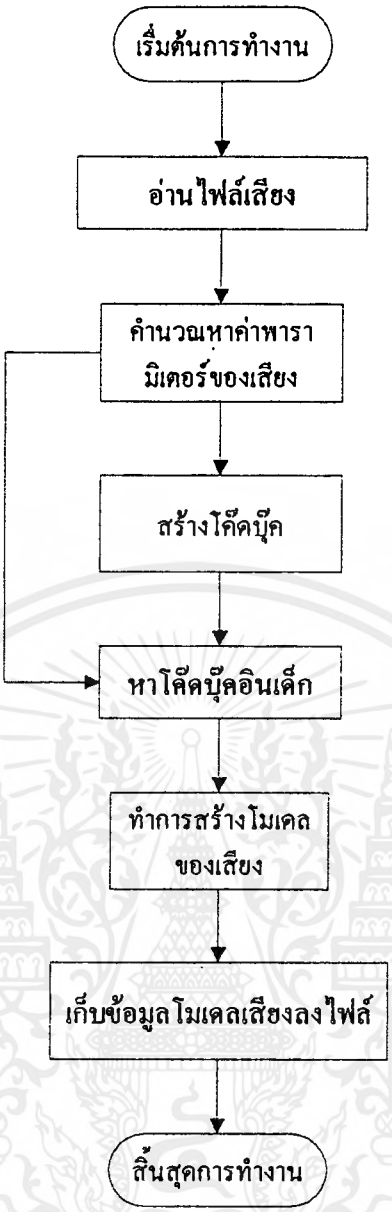
โปรแกรมเรียนรู้เสียง หรือสร้างโมเดลเสียง จะใช้เพื่อในการให้คอมพิวเตอร์เรียนรู้เสียงรู้จักคำ หรือพูดง่าย ๆ ก็คือ เอาไว้สร้างโมเดลของเสียง ซึ่งแผนผังแสดงการทำงานของโปรแกรมสร้างโมเดลเสียงจะแสดงอยู่ในรูปที่ 3.5 ซึ่งสามารถอธิบายเป็นขั้นตอนต่าง ๆ ได้ดังนี้

1. ทำการเก็บรวบรวมสัญญาณเสียงของคำทุกคำ ที่ต้องทำการเรียนรู้มาเก็บไว้
2. นำข้อมูลเสียงทั้งหมดที่เก็บไว้ในแต่ละกรณี มาทำการหาค่าพารามิเตอร์ ซึ่งจะถูกใช้เป็นตัวแทนของเสียงโดยวิธี Linear Predictive Coding (LPC)
3. นำพารามิเตอร์เสียง ทั้งหมดจากทุกกลุ่มเสียงมาทำการลดจำนวนข้อมูล โดยการใช้วิธีเวคเตอร์ควอนไตเซชั่น (Vector Quantization) แล้วจะได้เวคเตอร์ตัวแทนของเสียงที่ระบบสามารถรับรู้ได้ทั้งหมดออกมา เรียกว่าโค้ดบุค (Code Book)
4. ทำการหาโค้ดบุคอินเด็ก (Codebook Index) ของแต่ละกลุ่มเสียง โดยการนำพารามิเตอร์ของเสียงของกลุ่มนั้นๆ มาผ่านขั้นตอนอีกส่วนหนึ่งของการลดข้อมูล (VQ) จะได้ตัวเลขมากลุ่มหนึ่งที่บอกถึงความสัมพันธ์ของเสียงกลุ่มนั้นๆกับโค้ดบุค
5. ทำการสร้างโมเดลของเสียง โดยการนำโค้ดบุค ของแต่ละกลุ่มเสียงมาผ่านขั้นตอนการสร้างโมเดลด้วย วิธีฮิดเด็นมาร์คอฟโมเดล (Hidden Markov Model (HMM)) เมื่อทำจนครบทุกกลุ่มเสียง ก็จะได้โมเดลครบทุกเสียง
6. สิ้นสุดการทำงาน

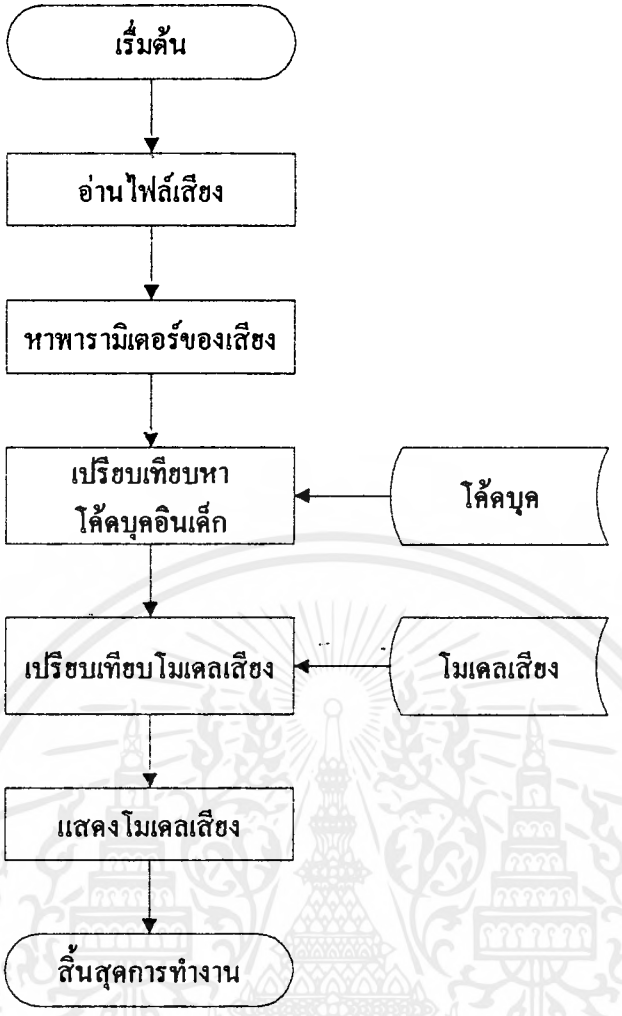
3.6 โปรแกรมวิเคราะห์เสียง

โปรแกรมวิเคราะห์เสียง เป็นโปรแกรมที่ใช้สำหรับในการทดสอบสัญญาณเสียงที่ทำการป้อนเข้าไปว่า เสียงที่ป้อนเข้าไปนั้นน่าจะเป็นเสียงใดในโมเดลมากที่สุด ซึ่งแผนผังแสดงขั้นตอนการทำงานของโปรแกรมนี้จะถูกแสดงอยู่ในรูปที่ 3.6 ซึ่งสามารถอธิบายได้ดังนี้

1. ทำการอ่านสัญญาณเสียงที่จะนำมาทดสอบ
2. นำเสียงที่ต้องการทดสอบ มาผ่านขั้นกระบวนการหาค่าพารามิเตอร์ของเสียง (LPC) จะได้พารามิเตอร์ของเสียงนั้น
3. นำพารามิเตอร์เสียงที่ได้มาเปรียบเทียบกับโค้ดบุคที่มีอยู่ก่อนแล้ว จะได้โค้ดบุคอินเด็ก
4. นำโค้ดบุคอินเด็ก มาเปรียบเทียบกับโมเดลที่เก็บไว้ทั้งหมด ก็จะได้ผลของการเปรียบเทียบของแต่ละคนออกมา ซึ่งถ้าเปรียบเทียบกับโมเดลใดแล้วได้ความน่าจะเป็นสูงสุด ก็จะถือว่าเป็นเสียงที่นำมาทดสอบนั้นตรงกับโมเดลนั้น
5. สิ้นสุดการทำงาน



รูปที่ 3.5 แผนผังแสดงขั้นตอนของโปรแกรมเรียนรู้เสียง หรือสร้างโมเดลเสียง



รูปที่ 3.6 แผนผังแสดงขั้นตอนการทำงานของโปรแกรมวิเคราะห์เสียง

3.7 การสร้างพอร์ตเพื่ออินเตอร์เฟสโดยใช้การ์ด

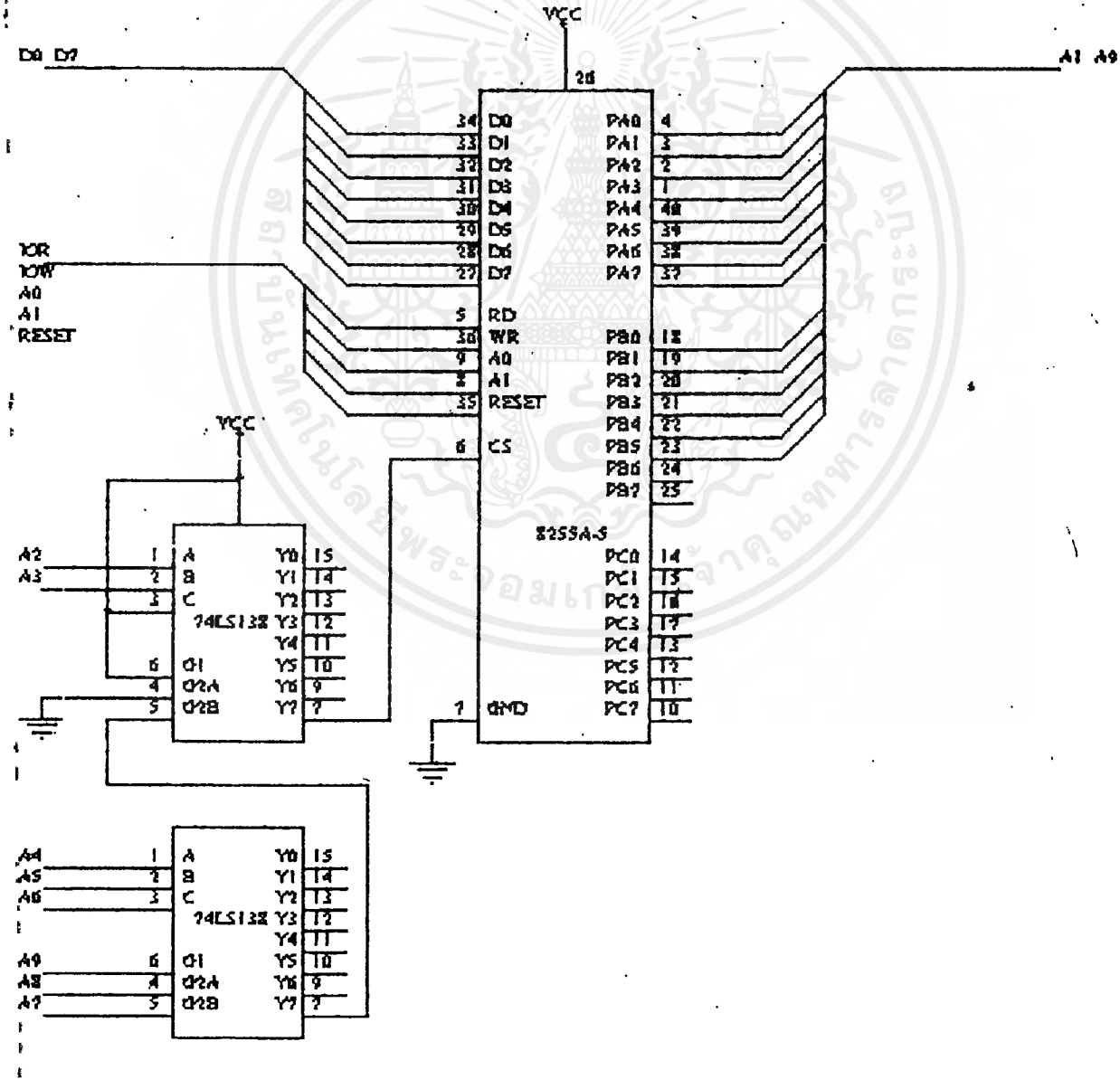
การอินเตอร์เฟส หรือการเชื่อมต่อระหว่างไมโครคอมพิวเตอร์ กับอุปกรณภายนอก สามารถสร้างขึ้นได้ โดยใช้ชิป 8255 ทำหน้าที่เป็นพอร์ตอินพุต หรือเอาต์พุต ซึ่งมีระดับ สัญญาณเป็น TTL เช่นกัน จะเป็นการ์ดอินเตอร์เฟสอเนกประสงค์ ซึ่งใช้งานกับไมโครคอมพิวเตอร์ PC/XT/AT ได้ทุกรุ่น ซึ่งจะต้องนำสัญญาณควบคุมบัสข้อมูล แอดเดรสบัส สัญญาณรีเซตไฟ VCC กราวด์ของไมโครคอมพิวเตอร์เชื่อมต่อกับการ์ดนี้ให้ถูกต้องเสียก่อน

การต่อสายสัญญาณ แอดเดรสบัส คาต้าบัส สัญญาณควบคุม และสัญญาณ I/O ควรใช้สายวายแล็ปอย่างละสี เพื่อสะดวกในการตรวจเช็ค เมื่อต่อสายสัญญาณต่าง ๆ เสร็จเรียบร้อยแล้ว ควรจะใช้มัลติมิเตอร์วัดสัญญาณต่าง ๆ ที่ต่อไว้ว่าถูกต้องตามวงจรหรือไม่ และที่สำคัญระวังอย่าให้ไฟ VCC กับกราวด์ช็อต หรือลัดวงจรเป็นอันขาด เพราะจะทำให้ตัวอุปกรณ์ และเครื่องไมโครคอมพิวเตอร์เสียหาย

จากวงจรรูปที่ 3.7 จะเห็นว่าใช้ 74LS138 ในการถอดรหัส 27X และเมื่อรวม A0-A19 จะได้เป็นหมายเลข 27C-27F ดังตารางที่ 3.1

ตารางที่ 3.1 แสดงหมายเลขพอร์ตที่ใช้ 74 LS138 ในการถอดรหัส

A9	A8	A7	A6	A5	A4	A3	Â2	A1	A0	หมายเลขพอร์ต	หมายเหตุ
G1	$G2_A$	$G2_B$	C_2	B_2	A_2	B_1	A_1				
1	0	0	1	1	1	1	1	0	0	27C	พอร์ต A
1	0	0	1	1	1	1	1	0	1	27D	พอร์ต B
1	0	0	1	1	1	1	1	1	0	27E	พอร์ต C
1	0	0	1	1	1	1	1	1	1	27F	พอร์ตควบคุม

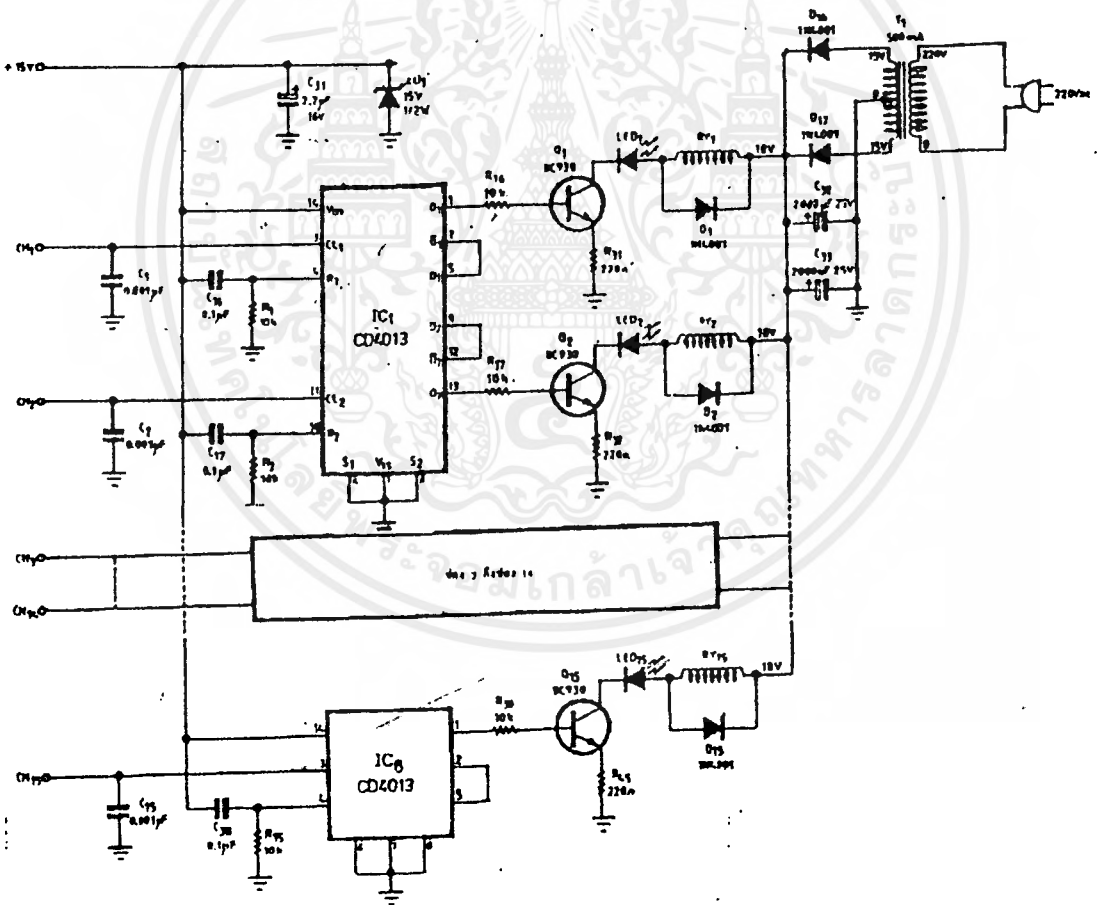


รูปที่3.7 วงจรการ์ดที่ใช้อินเตอร์เฟส

3.8 วงจรขับรีเลย์

วงจรในรูปที่ 3.8 ใช้ CD4013 ซึ่งเป็นไอซีฟลิปฟลอป สามารถขับรีเลย์ได้ 2 ตัว ช่องสัญญาณ 14 ช่อง ใช้ CD4013 ทั้งหมด 8 ตัว เมื่อสัญญาณอินพุทช่อง 1 ถูกส่งเข้ามา (ขา 3 ของ CD4013) สัญญาณเอาท์พุทของช่องที่ 1 (ขา 1 ของ CD4013) ก็จะให้ไบแอสแก่ Q_1 เบอร์ 2SC930 ทำให้รีเลย์ 1 ทำงาน LED_1 ก็จะสว่างเพื่อแสดงว่ารีเลย์ 1 ทำงานอยู่ ช่องอื่นๆ ก็เช่นเดียวกัน C_1, C_2 ใส่เพื่อป้องกันสัญญาณรบกวน

ส่วน T_1, D_{16}, C_{32}, C_{33}, C_{31}, R_{d6} และ ZD_1 เป็นส่วนของวงจรแหล่งจ่ายไฟตรงสำหรับภาครับสัญญาณอินฟราเรด



รูปที่ 3.8 แสดงวงจรภาคขับรีเลย์ขับรีเลย์จำนวน 14 ช่อง

บทที่ 4

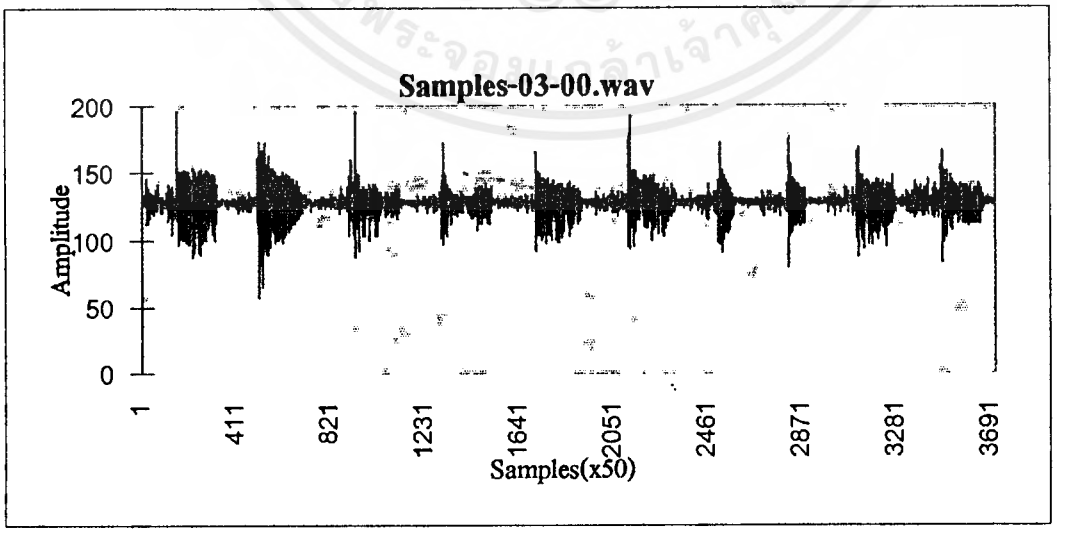
การทดลองและผลการทดลอง

4.1 การทดลองและผลการทดลอง

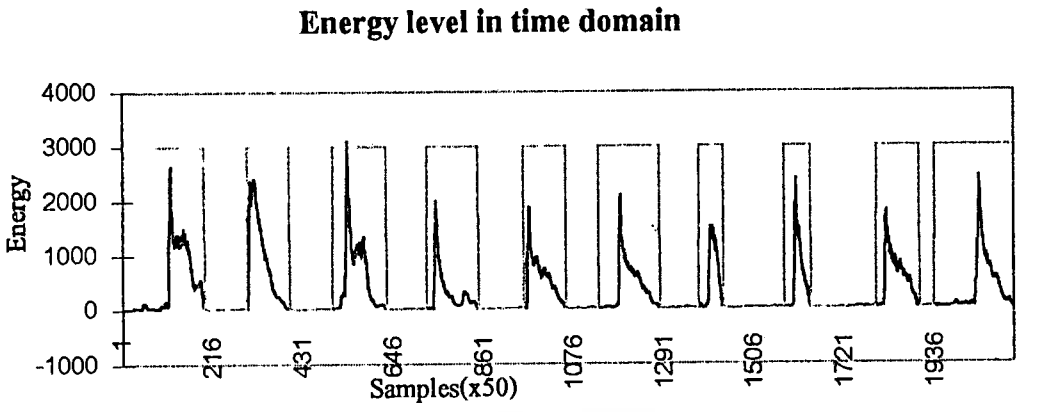
4.1.1 การทดลองตัดเสียงพูดตัวเลข 0-9

ในการทดลองการตัดคำนี้ จะมีขั้นตอนดังต่อไปนี้

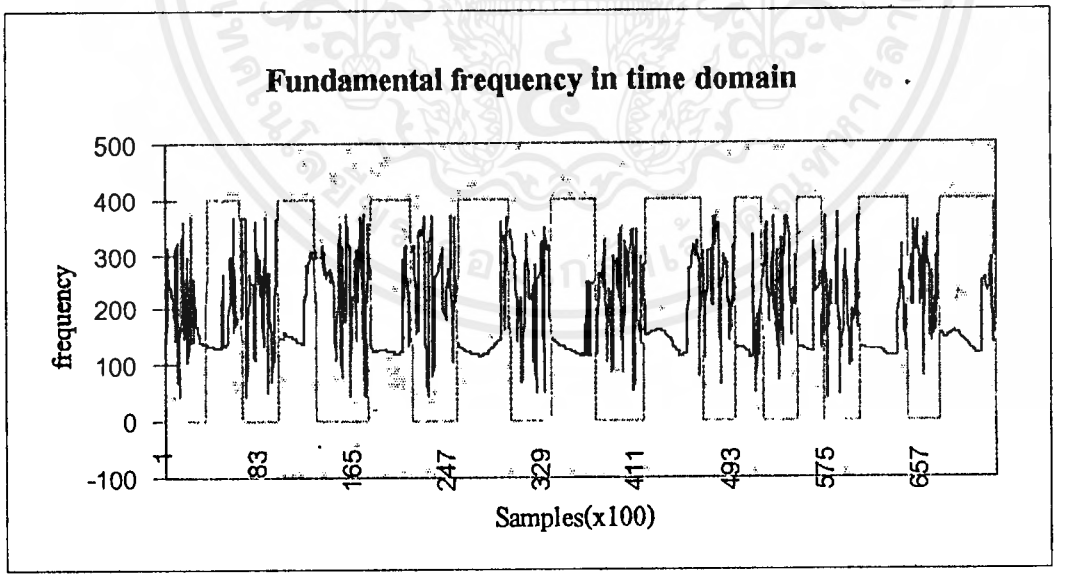
1. ทำการเก็บตัวอย่างเสียงพูด โดยให้แต่ละคนพูดตัวเลข 0-9 คนละ 5 ครั้ง โดยเก็บมาทั้งหมดเป็นจำนวน 10 คน ซึ่งจะทำให้ได้ไฟล์เสียงทั้งหมด 50 ไฟล์ ซึ่งจะมีลักษณะรูปคลื่นตามรูปที่ 4.1
2. นำไฟล์เสียงพูดที่บันทึกเก็บไว้ มาคำนวณหาค่าพลังงานเฉลี่ย โดยใช้โปรแกรมหาค่าพลังงานเฉลี่ยเพื่อหาขอบเขตคำ แล้วนำค่าที่ได้ไปพล็อตเป็นกราฟ ซึ่งลักษณะของกราฟที่เป็นค่าพลังงานเฉลี่ยของเสียงจะเป็นไปตามรูปที่ 4.2
3. นำไฟล์เสียงพูดที่บันทึกเก็บไว้ มาคำนวณหาค่าพิทช์ แล้วนำค่าที่ได้ไปพล็อตเป็นกราฟ ซึ่งลักษณะของกราฟที่แสดงค่าพิทช์ของสัญญาณเสียงพูด 0-9 นั้นจะเป็นไปตามรูปที่ 4.3
4. นำไฟล์เสียงพูดที่บันทึกเก็บไว้ มาคำนวณหาค่าอัตราเฉลี่ยการตัดศูนย์ แล้วนำค่าที่ได้ไปพล็อตเป็นกราฟ จะได้ตามรูปที่ 4.4
5. นำไฟล์เสียงที่ใช้ทำโมเดลต้นแบบมาทดสอบประสิทธิภาพการรู้จำ แล้วบันทึกเป็นตาราง ได้ผลแสดงดังตารางที่ 4.1
6. นำไฟล์เสียงที่ยังไม่ได้ผ่านการตัดคำ มาทดสอบการตัดคำว่าจะมีประสิทธิภาพในการตัดคำเท่าไร บันทึกเป็นตาราง ได้ผลแสดงดังตารางที่ 4.2



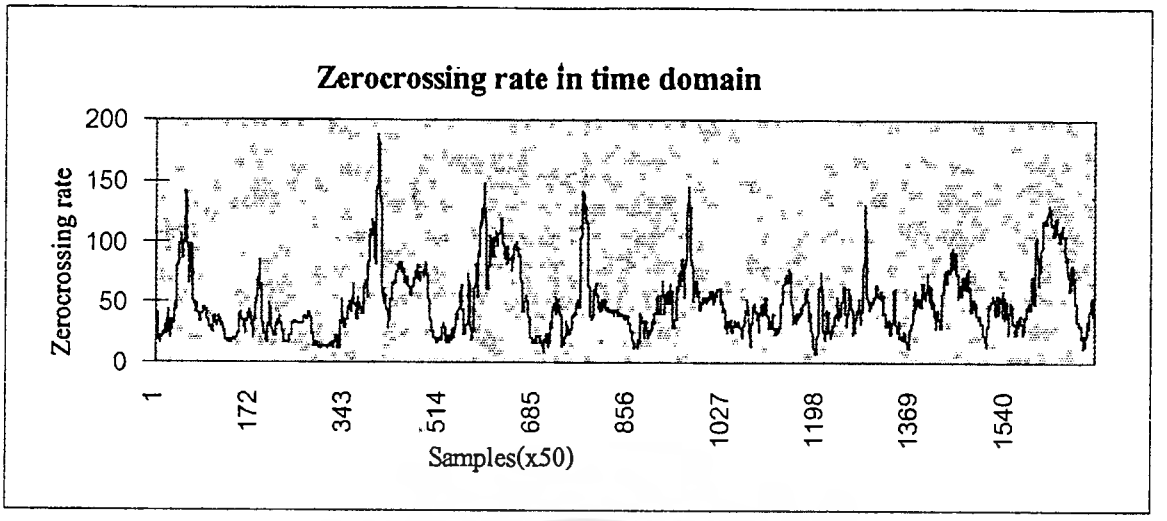
รูปที่ 4.1 แสดงรูปคลื่นของเสียงพูด 0-9



รูปที่ 4.2 แสดงรูปค่าพลังงานเฉลี่ยของเสียงพูด 0-9 ของผู้ชาย



รูปที่ 4.3 แสดงค่าความถี่พิทช์ของเสียงพูด 0-9



รูปที่ 4.4 แสดงค่าอัตราเฉลี่ยการตัดศูนย์ของเสียงพูด 0-9 ของผู้ชาย

ตารางที่ 4.1 แสดงประสิทธิภาพของเสียงต้นแบบในการวิเคราะห์เสียง

	เสียงที่จำได้											
		1	2	3	4	5	6	7	8	9	0	%
เสียงที่ป้อน	1	46	1	0	0	0	2	0	1	0	0	92
	2	0	39	1	0	0	3	0	5	0	2	78
	3	2	1	34	0	2	0	0	6	5	0	68
	4	0	1	0	47	0	0	0	0	0	2	94
	5	3	0	5	0	35	0	0	6	1	0	70
	6	0	4	0	0	0	43	0	0	0	3	86
	7	8	1	0	3	1	1	34	1	1	1	68
	8	0	0	0	0	1	0	1	47	1	0	94
	9	3	1	3	0	2	0	0	3	38	0	76
	0	0	8	0	1	0	2	3	0	0	36	72

ตารางที่ 4.2 แสดงประสิทธิภาพในการตัดตัวเลข 0-9

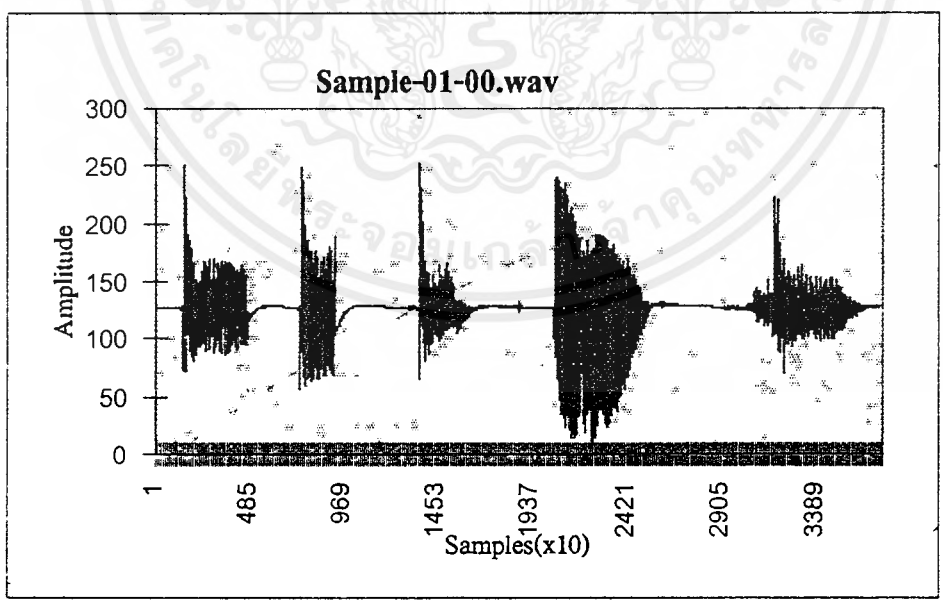
ชื่อไฟล์	จำนวนคำจริง	จำนวนคำที่ตัดได้	ความผิดพลาดในการตัด
sample-00-00.wav	10	11	1
sample-00-01.wav	10	14	4
sample-00-02.wav	10	14	4
sample-00-03.wav	10	10	0
sample-00-04.wav	10	11	1
sample-01-00.wav	10	10	0
sample-01-01.wav	10	10	0
sample-01-02.wav	10	10	0
sample-01-03.wav	10	10	0
sample-01-04.wav	10	10	0
sample-02-00.wav	10	11	1
sample-02-01.wav	10	12	2
sample-02-02.wav	10	10	0
sample-02-03.wav	10	10	0
sample-02-04.wav	10	12	2
sample-03-00.wav	10	10	0
sample-03-01.wav	10	12	2
sample-03-02.wav	10	12	2
sample-03-03.wav	10	10	0
sample-03-04.wav	10	10	0
sample-04-00.wav	10	13	3
sample-04-01.wav	10	10	0
sample-04-02.wav	10	10	0
sample-04-03.wav	10	11	1
sample-04-04.wav	10	14	4
sample-05-00.wav	10	10	0
sample-05-01.wav	10	15	5
sample-05-02.wav	10	13	3
sample-05-03.wav	10	13	3
sample-05-04.wav	10	14	4

ชื่อไฟล์	จำนวนคำจริง	จำนวนคำที่ตัดได้	ความผิดพลาดในการตัด
sample-06-00.wav	10	11	1
sample-06-01.wav	10	11	1
sample-06-02.wav	10	14	4
sample-06-03.wav	10	13	3
sample-06-04.wav	10	10	0
sample-07-00.wav	10	12	2
sample-07-01.wav	10	14	4
sample-07-02.wav	10	11	1
sample-07-03.wav	10	10	0
sample-07-04.wav	10	12	2
sample-08-00.wav	10	11	1
sample-08-01.wav	10	10	0
sample-08-02.wav	10	11	1
sample-08-03.wav	10	10	0
sample-08-04.wav	10	11	1
sample-09-00.wav	10	10	0
sample-09-01.wav	10	10	0
sample-09-02.wav	10	10	0
sample-09-03.wav	10	11	1
sample-09-04.wav	10	11	1
รวมทุกไฟล์	500	565	65
เปอร์เซนต์ความผิดพลาดในการตัด		13%	

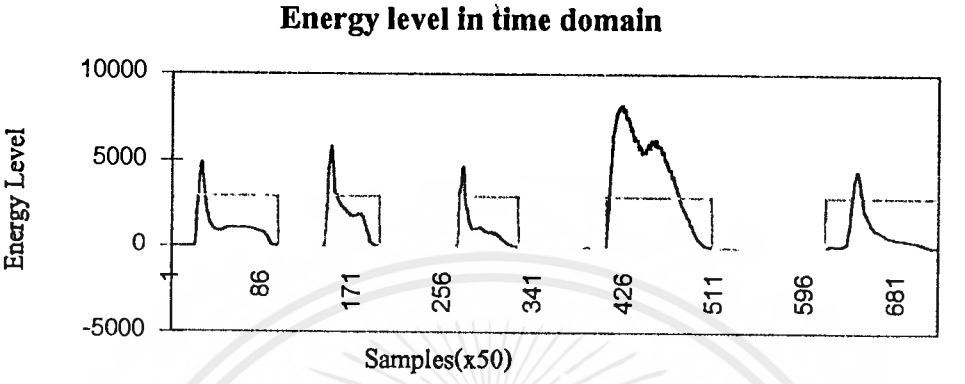
4.1.2การทดลองตัดเสียงพูดคำทั่วไป

ในการทดลองการตัดคำนี้ จะมีขั้นตอนดังต่อไปนี้

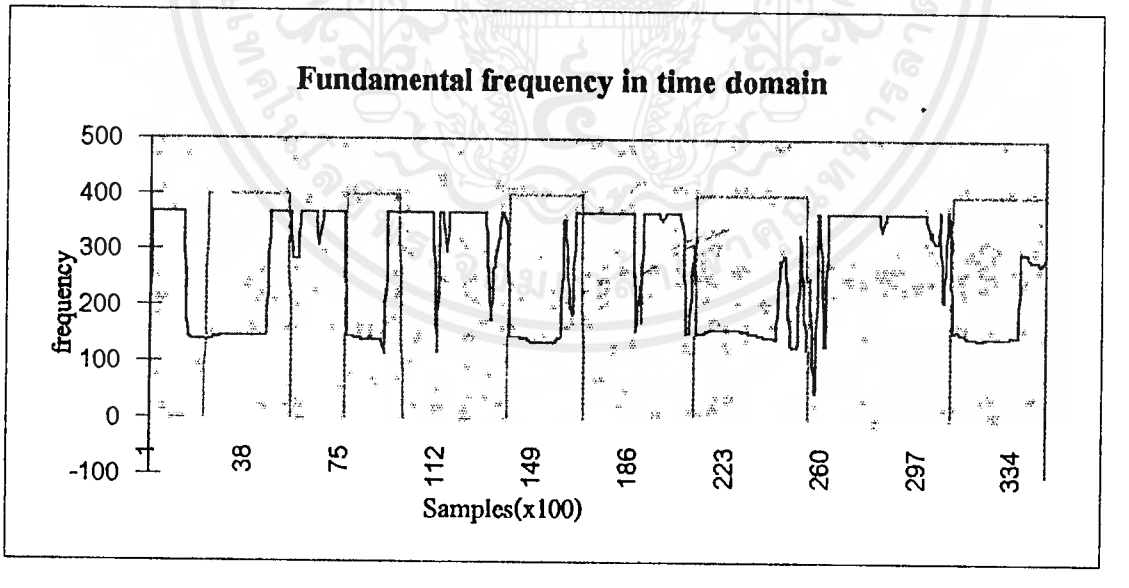
1. ทำการเก็บตัวอย่างเสียงพูด โดยให้แต่ละคนพูดคำว่า "เปิด" "ปิด" "ประ" "ดู" "ไฟ" คนละ 5 ครั้ง โดยเก็บมาทั้งหมดเป็นจำนวน 10 คน ซึ่งจะทำให้ได้ไฟล์เสียงทั้งหมด 50 ไฟล์ ซึ่งจะมีลักษณะรูปคลื่นตามรูปที่ 4.5
2. นำไฟล์เสียงพูดที่บันทึกเก็บไว้ มาคำนวณหาค่าพลังงาน โดยใช้โปรแกรมหาค่าพลังงานเพื่อหาขอบเขตคำ แล้วนำค่าที่ได้ไปพล็อตเป็นกราฟ ซึ่งลักษณะของกราฟที่เป็นค่าพลังงานของเสียงจะเป็นไปตามรูปที่ 4.6
3. นำไฟล์เสียงพูดที่บันทึกเก็บไว้ มาคำนวณหาค่าพิทช์ แล้วนำค่าที่ได้ไปพล็อตเป็นกราฟ ซึ่งลักษณะของกราฟ ที่แสดงค่าพิทช์ของสัญญาณเสียงพูดนั้นจะเป็นไปตามรูปที่ 4.7
4. นำไฟล์เสียงพูดที่บันทึกเก็บไว้ มาคำนวณหาค่าอัตราเฉลี่ยการตัดศูนย์ แล้วนำค่าที่ได้ไปพล็อตเป็นกราฟ จะได้ตามรูปที่ 4.8
5. นำไฟล์เสียงที่ใช้ทำโมเดลต้นแบบมาทดสอบประสิทธิภาพการรู้จำ แล้วบันทึกเป็นตาราง ได้ผลแสดงดังตารางที่ 4.3
6. นำไฟล์เสียงที่ยังไม่ได้ผ่านการตัดคำ มาทดสอบการตัดคำว่าจะมีประสิทธิภาพในการตัดคำเท่าไร บันทึกเป็นตาราง ได้ผลแสดงดังตารางที่ 4.4



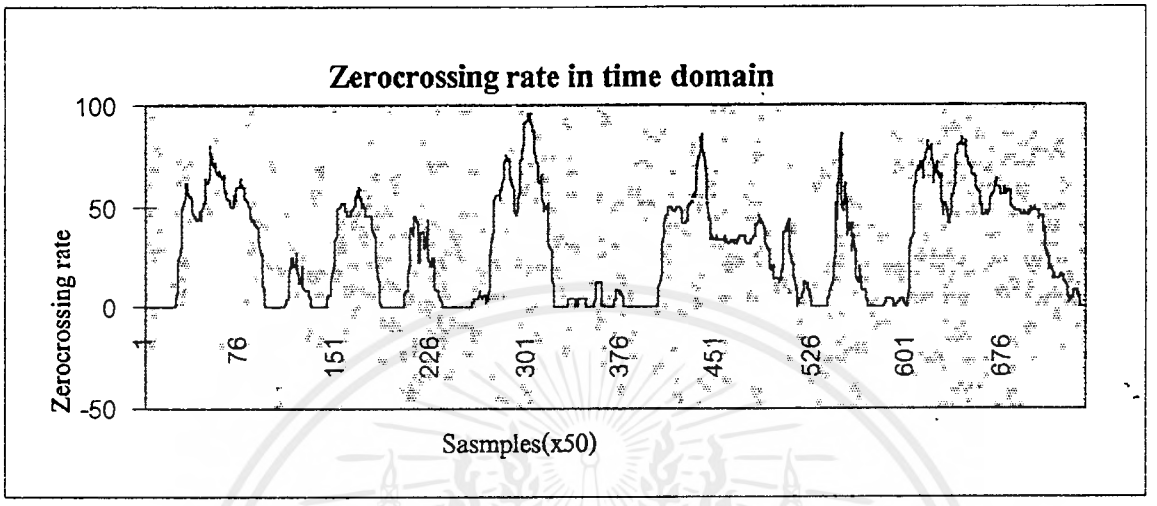
รูปที่ 4.5 แสดงรูปคลื่นสัญญาณเสียงพูดคำทั่วไป



รูปที่ 4.6 แสดงรูปค่าพลังงานเฉลี่ยของคำพูดทั่วไป



รูปที่ 4.7 แสดงค่าพิทช์ของคำพูดทั่วไป



รูปที่ 4.8 แสดงอัตราเฉลี่ยการตัดศูนย์ของคำทั่วไป

ตารางที่ 4.3 แสดงประสิทธิภาพของการวิเคราะห์เสียงของโมเดลเสียงต้นแบบ

	เสียงที่จำได้						
		เปิด	ปิด	ประ	ตู	ไฟ	%
เสียงที่ป้อน	เปิด	45	2	0	3	0	90
	ปิด	0	50	0	0	0	100
	ประ	3	0	46	0	1	92
	ตู	0	0	1	49	0	98
	ไฟ	0	0	0	1	49	98

ตารางที่ 4.4 แสดงประสิทธิภาพการตัดเสียงของคำทั่วไป

ชื่อไฟล์	จำนวนคำจริง	จำนวนคำที่ตัดได้	ความผิดพลาดในการตัด
sample-00-00.wav	5	4	2
sample-00-01.wav	5	5	0
sample-00-02.wav	5	5	0
sample-00-03.wav	5	5	0
sample-00-04.wav	5	4	2
sample-01-00.wav	5	5	0
sample-01-01.wav	5	5	0
sample-01-02.wav	5	5	0
sample-01-03.wav	5	5	0
sample-01-04.wav	5	5	0
sample-02-00.wav	5	4	2
sample-02-01.wav	5	5	0
sample-02-02.wav	5	5	0
sample-02-03.wav	5	5	0
sample-02-04.wav	5	5	0
sample-03-00.wav	5	5	0
sample-03-01.wav	5	5	0
sample-03-02.wav	5	5	0
sample-03-03.wav	5	6	1
sample-03-04.wav	5	5	0
sample-04-00.wav	5	5	0
sample-04-01.wav	5	5	0
sample-04-02.wav	5	5	0
sample-04-03.wav	5	5	0
sample-04-04.wav	5	5	0
sample-05-00.wav	5	5	0
sample-05-01.wav	5	5	0
sample-05-02.wav	5	5	0
sample-05-03.wav	5	5	0
sample-05-04.wav	5	5	0

ชื่อไฟล์	จำนวนคำจริง	จำนวนคำที่ตัดได้	ความผิดพลาดในการตัด
sample-06-00.wav	5	5	0
sample-06-01.wav	5	5	0
sample-06-02.wav	5	5	0
sample-06-03.wav	5	5	0
sample-06-04.wav	5	5	0
sample-07-00.wav	5	4	2
sample-07-01.wav	5	5	0
sample-07-02.wav	5	4	2
sample-07-03.wav	5	6	1
sample-07-04.wav	5	4	2
sample-08-00.wav	5	4	2
sample-08-01.wav	5	6	1
sample-08-02.wav	5	5	0
sample-08-03.wav	5	4	2
sample-08-04.wav	5	4	2
sample-09-00.wav	5	4	2
sample-09-01.wav	5	4	2
sample-09-02.wav	5	4	2
sample-09-03.wav	5	5	0
sample-09-04.wav	5	4	2
รวมทุกไฟล์	250	240	29
เปอร์เซนต์ความผิดพลาดในการตัด		11.6%	
ตัดขาด 1 คำเท่ากับ หายไป 2 คำ			

บทที่ 5

สรุปผลการทดลองและข้อเสนอแนะ

5.1 สรุปผลการทดลอง และข้อเสนอแนะ

จุดสำคัญในการวิเคราะห์เสียง เพื่อใช้ในการรู้จำเสียงนั้น ประการหนึ่งคือ การสร้างโมเดลเสียง ถ้าเสียงที่ใช้ในการสร้างโมเดลเสียงมีจำนวนมาก และแต่ละเสียงเป็นเสียงที่ถูกเก็บมาจากสภาพแวดล้อมที่ดีแล้ว จะทำให้อัตราการรู้จำของเสียงมีความถูกต้องมากยิ่งขึ้น อีกประการหนึ่งที่สำคัญมากไม่แพ้กันคือ การตัดแบ่งประโยคออกเป็นคำให้ได้อย่างถูกต้อง เนื่องจากถึงแม้ว่าตัวรู้จำเสียง และโมเดลเสียงจะมีคุณสมบัติดีเพียงใด แต่ถ้าเสียงที่เป็นสัญญาณป้อนเข้าของตัวรู้จำเสียงไม่ได้ตัดแบ่งคำอย่างถูกต้องแล้ว ผลของการรู้จำเสียง ก็จะไม่มีทางที่จะมีความถูกต้องได้เลย

การทำปริญญานิพนธ์ฉบับนี้ ส่วนมากจะเกี่ยวข้องกับการเขียนโปรแกรม ซึ่งโปรแกรมจะประกอบด้วย 3 ส่วนคือ

1. ส่วนคำนวณค่าพารามิเตอร์ที่ใช้ในการตัดคำ คือ ค่าพลังงานเฉลี่ย, ค่าความถี่มูลฐาน และค่าอัตราเฉลี่ยการตัดศูนย์
2. การสร้างขอบเขตของคำ โดยนำค่าพารามิเตอร์ที่ได้จากส่วนแรกมาใช้
3. นำขอบเขตของคำที่ได้ ไปสร้างเป็นไฟล์เสียงย่อย ๆ จากไฟล์เสียงเดิม

แต่ในการกำหนดขอบเขตของคำ มีพารามิเตอร์ 2 ตัว คือ ค่าความถี่มูลฐาน และค่าอัตราเฉลี่ยการตัดศูนย์ ไม่สามารถใช้ได้ผลดีเหมือนกับทางทฤษฎี เนื่องจากพารามิเตอร์ทั้งสองตัวนี้แสดงค่าเชิงความถี่ ซึ่งจะได้รับผลกระทบจากสิ่งแวดล้อมในขณะบันทึกเสียงอย่างมาก แต่อย่างไรก็ตามค่าความถี่มูลฐานสามารถที่จะใช้ในการตรวจสอบ ส่วนของเสียงที่ตัดได้นั้นเป็นเสียงที่เป็นคำพูดหรือไม่ แต่เนื่องจากการคำนวณการหาค่าความถี่มูลฐานนั้นใช้เวลาในการคำนวณมาก ซึ่งไม่คุ้มกับผลที่ได้รับ จึงไม่ได้นำมาใช้เป็นส่วนหนึ่งในการตัดคำจริง ๆ ส่วนค่าอัตราการตัดศูนย์นั้น พบว่าไม่สามารถใช้ได้จริงในทางปฏิบัติ ส่วนค่าพลังงานเฉลี่ยที่ใช้เป็นหลักในการตัดแบ่งคำนั้นสามารถใช้ได้ผลดี

ส่วนแนวทางในการพัฒนานั้น เนื่องจากการรู้จำเสียงซึ่งใช้ในการสร้างโมเดลเสียงแบบ HMM นั้นมีประสิทธิภาพสูงอยู่แล้ว ขึ้นอยู่กับว่าอินพุทของส่วนรู้จำเสียงนั้น จะเป็นเสียงที่มีขอบเขตของคำถูกต้องมากน้อยเพียงใด ดังนั้นสิ่งสำคัญจึงอยู่ที่การค้นหาวิธี ที่จะกำหนดขอบเขตคำ เพื่อที่จะตัดแบ่งประโยคออกเป็นคำให้ได้ถูกต้องที่สุด อีกจุดหนึ่งที่น่าสนใจในการพัฒนาคือ การที่ทำให้การทำงานทั้งหมดเป็นระบบเวลาจริง (Real time system) ซึ่งจะทำให้ระบบที่ได้มีประโยชน์ในการใช้งานเพิ่มขึ้น แต่การคำนวณทั้งหมดก็จะเปลี่ยนแปลงด้วย

ภาคผนวก

```
//*********************************************************************
//This procedure used for calculate Zerocrossing Rate
//*********************************************************************
void zerocrossing(char name[30])
      {
      FILE  *digitalfile;
      char  digitalfilename[30]="digital.tmp";
      int   number_sample,b1;

      WINDOW=atoi(transBuf3.window);
      SHIFT=atoi(transBuf3.shift);
      MISS=atoi(transBuf3.shift);
      for (int j=0;j<20;j++)
            infile[j]=name[j];
//Open input wave file
      if ((inputfile = fopen(infile,"rb")) == NULL )
            {
            MessageBox(0,"Can't open file.", "ERROR!", MB_OK);
            open_result=1;
            }
      ENDZERO:
      if (open_result==0)
            {
            for (int j=0;j<20;j++)
                  outfile[j]=name[j];
            outfile[dash2+3]='.';
            outfile[dash2+4]='z';
            outfile[dash2+5]='c';
            outfile[dash2+6]='r';
//Open output binary file
            if ((outputfile = fopen(outfile,"wb")) == NULL)
                  {
                  MessageBox(0,"Can't create output text file.",
"ERROR!", MB_OK);
                  open_result=1;
                  goto ENDZERO;
                  }
            filenametxt[0]='z';
            filenametxt[1]='c';
            filenametxt[2]='r';
            filenametxt[3]='a';
            filenametxt[4]='t';
            filenametxt[5]='e';
            filenametxt[6]='-';
            filenametxt[7]=name[dash1+1];
            filenametxt[8]=name[dash1+2];
            filenametxt[9]='-';
            filenametxt[10]=name[dash2+1];
            filenametxt[11]=name[dash2+2];
            filenametxt[12]='.';
            filenametxt[13]='t';
            filenametxt[14]='x';
            filenametxt[15]='t';
//Open output text file
            if ((outputtextfile = fopen(filenametxt,"wt")) == NULL)
                  {
                  MessageBox(0,"Can't create output text file.",
"ERROR!", MB_OK);
                  open_result=1;
                  }
            if (open_result==0)
                  {
//Set file pointer to first sound sample
                  fseek(inputfile,44L,SEEK_CUR);
```

```
                    number_sample = 0;
//Find number of sample in wav file
                    while (!feof(inputfile))
                          {
                          j = (int) fgetc(inputfile);
                          number_sample = number_sample+1;
                          }
                    number_sample = number_sample -1;
                    rewind(inputfile);
                    fseek(inputfile,44L,SEEK_CUR);
//Open output file of digital filter
                    if ((digitalfile = fopen(digitalfilename,"wb")) == NULL
)
                          {
                          MessageBox(0,"Can't create digital file.",
"ERROR!", MB_OK);
                          open_result=1;
                          goto ENDZERO;
                          }
                    int round = (number_sample/SHIFT)-(WINDOW/SHIFT);
//This following routine is Digital filter
                    for (int i = 1;i <= number_sample;i++)
                          {
                          int   x_n,x_n1,x_n2,x_n3;
                          float y_n,y_n1,y_n2,y_n3;
                          if (i == 1)
                                {
                                y_n  = 0;
                                y_n1 = 128;
                                y_n2 = 128;
                                y_n3 = 128;
                                x_n3 = (int) fgetc(inputfile);
                                x_n2 = (int) fgetc(inputfile);
                                x_n1 = (int) fgetc(inputfile);
                                x_n  = (int) fgetc(inputfile);
                                }
                          else
                                {
                                y_n3 = y_n2;
                                y_n2 = y_n1;
                                y_n1 = y_n;
                                x_n3 = x_n2;
                                x_n2 = x_n1;
                                x_n1 = x_n;
                                x_n = (int) fgetc(inputfile);
                                }
//Calculate output digital filter
                          y_n = -(2.13806*y_n1)-
      (1.802663*(y_n2))+(0.545353*(y_n3))+(0.085292*(x_n))-
      (0.0257159*(x_n1))-(0.0257159*(x_n2))+(0.085292*(x_n3));
                          fwrite(&y_n,sizeof(float),1,digitalfile);
                          }
                    fclose(inputfile);
                    fclose(digitalfile);
//End of Digital filter
                    if ((digitalfile = fopen(digitalfilename,"rb")) == NULL
)
                          {
                          MessageBox(0,"Can't create digital file.",
"ERROR!", MB_OK);
                          open_result=1;
                          goto ENDZERO;
                          }
//Start caluculate zerocrossing rate
```

```
                    for (int i=1;i<=round;i++)
                        {
                        int zerorate = 0;
                        for (b1=1;b1<=(WINDOW-1);b1++)
                        {
                        float x_m,x_m1;
                        int   sgnx_m,sgnx_m1;
                        int temp1 = 0;
                        if (b1 == 1)
                        {
                        fread(&x_m1,sizeof(float),1,digitalfile);
                        fread(&x_m,sizeof(float),1,digitalfile);
                        }
                        else
                        {
                        x_m1 = x_m;
                        fread(&x_m,sizeof(float),1,digitalfile);
                        }
                        if (x_m1 == 128)
                            x_m1 = 0;
                        if (x_m == 128)
                            x_m = 0;
                        if (x_m > 128)
                            sgnx_m = 1;
                        if (x_m < 128)
                            sgnx_m = -1;
                        if (x_m1 > 128)
                            sgnx_m1 = 1;
                        if (x_m1 < 128)
                            sgnx_m1 = -1;
                        int temp = sgnx_m - sgnx_m1;
                        temp1 = abs(temp);
                        if (temp1 == 1)
                            temp1 = temp1 - 1;
                        zerorate = zerorate + temp1;
                        }
  //Write an element output to binary file
                    putw(zerorate,outputfile);
  //Write output to be a line of text file
                    fprintf(outputtextfile,"\n %d ",zerorate);
                    }
                fclose(digitalfile);
                fclose(outputfile);
                fclose(outputtextfile);
            }
        }
        }
  //End Zerocrossing
  //*********************************************************************
  //This procedure used for estimate Pitch period
  //*********************************************************************
  void  Pitch(char name[30])
        {
        SHIFT=atoi(transBuf3.shift);
        FILE  *clipfile,*autocorfile,*pitchfile;
        FILE  *digitalfile,*pitchtempfile;
        FILE  *pitchtextfile;
        char  pitchtextfilename[20];
        char  pitchfilename[20];
        char  clipfilename[20]="clip.tmp";
        char  autocorfilename[20]="auto.tmp";
        char  digitalfilename[20]="digital.tmp";
        char  pitchtempfilename[20]="pitch.tmp";
        int   *pitch,sample1,round,percentpitch = 30;
```

```
        int   count,n,round2,number_sample;
        int   count_segment=0;
        for (int j=0;j<30;j++)
              infile[j]=name[j];
  //Open input wav file
        if ((inputfile = fopen(infile,"rb")) == NULL )
              {
              MessageBox(0,"Can't open wave file.", "ERROR!", MB_OK);
              open_result=1;
              }
  //Opem output file for clipping process
        if ((clipfile = fopen(clipfilename,"wb")) == NULL )
              {
              MessageBox(0,"Can't create temp file.", "ERROR!", MB_OK);
              open_result=1;
              }
  //Open output file for digital filter process
        if ((digitalfile = fopen(digitalfilename,"wb")) == NULL )
              {
              MessageBox(0,"Can't create digital file.", "ERROR!", MB_OK);
              open_result=1;
              }
  //Open output file for pitch period estimation process
        if ((pitchtempfile = fopen(pitchtempfilename,"wb")) == NULL )
              {
              MessageBox(0,"Can't create pitchtemp file.", "ERROR!",
  MB_OK);
              open_result=1;
              }
        ENDPITCH:
        if (open_result==0)
              {
              number_sample = 0;
              rewind(inputfile);
              fseek(inputfile,44L,SEEK_CUR);
              while (!feof(inputfile))
                    {
                    sample1 = (int) getc(inputfile);
                    if (sample1 == -1)
                          goto ENDOFFILE1;
                    number_sample = number_sample + 1;
                    }
              ENDOFFILE1:
              round = (number_sample/(SHIFT))-((FRAME_LENGTH-SHIFT)/SHIFT);
              rewind(inputfile);
              fseek(inputfile,44L,SEEK_CUR);
  //Start digital filter process...
              for (int i = 1;i <= number_sample;i++)
                    {
                    int   x_n,x_n1,x_n2,x_n3;
                    float y_n,y_n1,y_n2,y_n3;
                    if (i == 1)
                          {
                          y_n  = 0;
                          y_n1 = 128;
                          y_n2 = 128;
                          y_n3 = 128;
                          x_n3 = (int) fgetc(inputfile);
                          x_n2 = (int) fgetc(inputfile);
                          x_n1 = (int) fgetc(inputfile);
                          x_n  = (int) fgetc(inputfile);
                          }
                    else
                          {
```

```
                        y_n3 = y_n2;
                        y_n2 = y_n1;
                        y_n1 = y_n;
                        x_n3 = x_n2;
                        x_n2 = x_n1;
                        x_n1 = x_n;
                        x_n = (int) fgetc(inputfile);
                        }
    //calcultate output digital filter
                        y_n = (2.13806*y_n1)-
    (1.802663*(y_n2))+(0.545353*(y_n3))+(0.085292*(x_n))-(0.0257159*(x_n1))-
    (0.0257159*(x_n2))+(0.085292*(x_n3));
    //Write to file
                        fwrite(&y_n,sizeof(float),1,digitalfile);
                        }
                fclose(inputfile);
                fclose(digitalfile);
                if ((digitalfile = fopen(digitalfilename,"rb")) == NULL )
                        {
                        MessageBox(0,"Can't create digital file.", "ERROR!",
    MB_OK);
                        open_result=1;
                        goto ENDPITCH;
                        }
    //Start 3-level clip center process...
                for (int d=0;d<round;d++)
                        {
                        char  i1,outputclip[FRAME_LENGTH];
                        float wave[FRAME_LENGTH],temp2[FRAME_LENGTH];
                        float firstmax,lastmax,minofmax;
                        int   threshold_clip,upperthreshold,lowerthreshold;
                        if (d==0)
                        {
                        for (int j=0;j<FRAME_LENGTH;j++)
                                {
                                fread(&temp2[j],sizeof(float),1,digitalfile);
                                wave[j] = temp2[j];
                                }
                        }
                        else
                        {
                        for (int j=0;j<FRAME_LENGTH-SHIFT;j++)
                                {
                                temp2[j] = wave[j+SHIFT];
                                wave[j] = temp2[j];
                                }
                        for (int j=0;j<SHIFT;j++)
                                {
                                fread(&wave[j+(FRAME_LENGTH-
    SHIFT)],sizeof(float),1,digitalfile);
                                temp2[j+(FRAME_LENGTH-SHIFT)] =
    wave[j+(FRAME_LENGTH-SHIFT)];
                                }
                        }
                        for (int j=0;j<FRAME_LENGTH;j++)
                                {
                                temp2[j] = abs(temp2[j] - 128);
                                }
    //Find firstmax and lastmax in 100 samples of each frame..
                        firstmax = 0;
                        lastmax = 0;
                        for (int j=0;j<SHIFT_LENGTH;j++)
                                {
                                if (firstmax < temp2[j])
```

```
                                    firstmax = temp2[j];
                                if (lastmax < temp2[j+(FRAME_LENGTH-SHIFT)])
                                lastmax = temp2[j+(FRAME_LENGTH-SHIFT)];
                                }
                        if (firstmax < lastmax)
                              minofmax = firstmax;
                        if (lastmax < firstmax)
                              minofmax = lastmax;
                        if (firstmax == lastmax)
                              minofmax = firstmax;
    //find threshold clip level..
                        threshold_clip = (minofmax * PERCENTCLIP)/100;
                        upperthreshold = 128 + threshold_clip;
                        lowerthreshold = 128 - threshold_clip;
    //clip process
                        for (int c=0;c<FRAME_LENGTH;c++)
                              {
                              if (wave[c] >= upperthreshold)
                              i1 = 1;
                              if (wave[c] <= lowerthreshold)
                              i1 =-1;
                              if ((wave[c] > lowerthreshold) && (wave[c] <
          upperthreshold))
                                    i1 = 0;
                              outputclip[c] = i1;
                              }
                  fwrite(outputclip,sizeof(char),FRAME_LENGTH,clipfile);
                  }
          fclose(digitalfile);
          fclose(clipfile);
          if ((clipfile=fopen(clipfilename,"rb"))==NULL)
                  {
                  MessageBox(0,"Can't open temp file.", "ERROR!", MB_OK);
                  open_result=1;
                  }
          if ((autocorfile=fopen(autocorfilename,"wb"))==NULL)
                  {
                  MessageBox(0,"Can't create temp file.", "ERROR!", MB_OK);
                  open_result=1;
                  }
          if (open_result==1)
                  goto ENDPITCH;
          count = -1;
          while (!feof(clipfile))
                  {
                  e=fgetc(clipfile);
                  count = count++;
                  }
          char  x1[FRAME],flag;
          int   y1,y2,autotemp,Autocor,autocor[LAG];
          int   Autok[LAG],Nextpitch;
          int   round1 = count/FRAME_LENGTH;
          rewind(clipfile);
    //Start autocorrelation process
          for (int j=1;j<=round1;j++)
                  {
    //Read data from clip file..
                  fread(x1,sizeof(char),FRAME_LENGTH,clipfile);
    //Calculate autocorrelation process..
                  for (int k=0;k<LAG;k++)
                        {
                        Autocor = 0;
                        for (int m=0;m<FRAME_LENGTH-k-1;m++)
                              {
```

```
                        y1 = x1[m];
                        y2 = x1[m+k];
                        autotemp = y1*y2;
                        Autocor = Autocor+autotemp;
                        }
                    autocor[k]=Autocor;
                    }
//write autocorrelation to file...
                    fwrite(autocor,sizeof(int),LAG,autocorfile);
                    }
      fclose(clipfile);
      fclose(autocorfile);
      if ((autocorfile = fopen(autocorfilename,"rb")) == NULL )
            {
            MessageBox(0,"Can't open temp file.", "ERROR!", MB_OK);
            open_result=1;
            goto ENDPITCH;
            }
      for (int j=0;j<20;j++)
            pitchfilename[j]=name[j];
      pitchfilename[dash2+3]='.';
      pitchfilename[dash2+4]='p';
      pitchfilename[dash2+5]='t';
      pitchfilename[dash2+6]='h';
      if ((pitchfile = fopen(pitchfilename,"wb")) == NULL )
            {
            MessageBox(0,"Can't create output file.", "ERROR!", MB_OK);
            open_result=1;
            goto ENDPITCH;
            }
      for (int j=0;j<20;j++)
            pitchtextfilename[j]=name[j];
      pitchtextfilename[0]='p';
      pitchtextfilename[1]='i';
      pitchtextfilename[2]='t';
      pitchtextfilename[3]='c';
      pitchtextfilename[4]='h';
      pitchtextfilename[5]='-';
      pitchtextfilename[6]=name[dash1+1];
      pitchtextfilename[7]=name[dash1+2];
      pitchtextfilename[8]='-';
      pitchtextfilename[9]=name[dash2+1];
      pitchtextfilename[10]=name[dash2+2];
      pitchtextfilename[11]='.';
      pitchtextfilename[12]='t';
      pitchtextfilename[13]='x';
      pitchtextfilename[14]='t';
      pitchtextfilename[15]=pitchtextfilename[16];
      if ((pitchtextfile = fopen(pitchtextfilename,"wt")) == NULL )
            {
            MessageBox(0,"Can't create pitch text file.", "ERROR!",
MB_OK);
            open_result=1;
            goto ENDPITCH;
            }
      count = -1;
      while (!feof(autocorfile))
            {
            e = fgetc(autocorfile);
            count = count++;
            }
      count /= 4;
      round2 = (count/LAG);
      rewind(autocorfile);
```

```
        pitch=(int*)calloc(sizeof(int),round2);
        int lastk,examinepitch,pitchfrequency;
  //Start find pitch frequency process..
        for (int n=1;n<=round2;n++)
              {
              PITCH:
              for (int j=0;j<LAG;j++)
                    fread(&Autok[j],sizeof(int),1,autocorfile);
              Nextpitch = (Autok[0] * percentpitch)/100;
              for (int k=0;k<LAG;k++)
                    {
                    int flag = 0;
                    if (k == 0)
                          {
                          lastk = 0;
                          examinepitch = 0;
                          k = 30;
                          }
                    if (abs(Autok[k]) >= Nextpitch)
                    {
                          if (k < 220)
                          for (int c=1;c<=30;c++)
                          if (abs(Autok[k]) < abs(Autok[k+c]))
                                flag = 1;
                          if (flag == 0)
                          {
                          Nextpitch = (Autok[k]*percentpitch)/100;
                          if (Nextpitch == 0)
                                Nextpitch = 1;
                          examinepitch = k - lastk;
                          lastk = k;
                          pitchfrequency = 11025/examinepitch;
                          k = LAG;
                          }
                    }
              }
        if (examinepitch != 0)
              pitchfrequency = 11025/examinepitch;
        if (examinepitch == 0)
              {
              percentpitch = percentpitch - 5;
              goto PITCH;
              }
        percentpitch = 30;
        pitch[n]=pitchfrequency;
        }
        fclose(autocorfile);
        int   findstart_sound = 0;
        int   possibleend_sound,possible_sound,start_sound=0;
        int   end_sound;
        bool  check_discontinue = false;
  //find boundary of sound by use pitch..
        for (int n=1;n<=round2;n++)
              {
              int   one,two,three,four,five,six,seven;
              START:
              if (n==0)
                    {
                    one = abs(pitch[n]-pitch[n+1]);
                    two = abs(pitch[n+1]-pitch[n+2]);
                    three = abs(pitch[n+2]-pitch[n+3]);
                    four = abs(pitch[n+3]-pitch[n+4]);
                    five = abs(pitch[n+4]-pitch[n+5]);
                    six = abs(pitch[n+5]-pitch[n+6]);
```

```
                        seven = abs(pitch[n+6]-pitch[n+7]);
                        }
                  else
                        {
                  seven = six;
                  six = five;
                  five = four;
                  four = three;
                  three = two;
                  two = one;
                  one = abs(pitch[n])-abs(pitch[n+1]);
                  }
            if
(((one<=7)&&(two<=7)&&(three<=7)&&(four<=7)&&(five<=7)&&(six<=7)&&(seven<
=7))&&(pitch[n]<300))
                  {
                  if (findstart_sound == 0)
                        {
                        start_sound = n;
                        findstart_sound = 1;
                        possible_sound = n;
                        possibleend_sound = n+7;
                        check_discontinue = true;
                        }
                  n++;
                  possible_sound++;
                  possibleend_sound++;
                  goto START;
                  }
                  if (check_discontinue == true)
                        {
                        end_sound = possibleend_sound;
                        fwrite(&start_sound,sizeof(int),1,pitchtempfile);
                        fwrite(&end_sound,sizeof(int),1,pitchtempfile);
                        check_discontinue = false;
                        findstart_sound = 0;
                        }
                  }
            fclose(pitchtempfile);
            if ((pitchtempfile = fopen(pitchtempfilename,"rb")) == NULL )
                  {
                  MessageBox(0,"Can't open temp file.", "ERROR!", MB_OK);
                  open_result=1;
                  goto ENDPITCH;
                  }
            c = 0;
            rewind(pitchtempfile);
            while (!feof(pitchtempfile))
                  {
                  n = (int) getc(inputfile);  // impression
                  c++;
                  }
            c = c/2;
            rewind(pitchtempfile);
            int check_start,check_end;
            fread(&check_start,sizeof(int),1,pitchtempfile);
            fwrite(&check_start,sizeof(int),1,pitchfile);
            for (int n=1;n<c;n++)
                  {
                  fread(&check_end,sizeof(int),1,pitchtempfile);
                  fread(&check_start,sizeof(int),1,pitchtempfile);
                  int x;
                  x = check_start - check_end;
                  check_start = check_start-7;
```

```
                if (x>6)
                      {
                      fwrite(&check_end,sizeof(int),1,pitchfile);
                      fwrite(&check_start,sizeof(int),1,pitchfile);
                      }
                }
                fread(&check_end,sizeof(int),1,pitchtempfile);
                fwrite(&check_end,sizeof(int),1,pitchfile);
                fclose(pitchtempfile);
                fclose(pitchfile);
    //End find boundary of sound by using pitch...
          if ((pitchfile = fopen(pitchfilename,"rb")) == NULL )
                {
                MessageBox(0,"Can't open pitch file for read.", "ERROR!",
    MB_OK);
                open_result=1;
                goto ENDPITCH;
                }
          if ((pitchtextfile=fopen(pitchtextfilename,"wt"))==NULL)
                {
                MessageBox(0,"Can't crreate frequencytext.", "ERROR!",
    MB_OK);
                open_result=1;
                goto ENDPITCH;
                }
          rewind(pitchfile);
          while (!feof(pitchfile))
                {
                c=getw(pitchfile);
                count_segment++;
                }
          rewind(pitchfile);
          count_segment = count_segment/2;
          int oldpointstop = 0;
          int pointstart,pointstop;
    //Write boundary of pitch to text file..
          for (n=0;n<count_segment;n++)
                {
                pointstart = getw(pitchfile);
                pointstop = getw(pitchfile);
                for (i=oldpointstop;i<=pointstart;i++)
                      fprintf(pitchtextfile,"\n %d  %d",pitch[i],0);
                for (i=pointstart+1;i<=pointstop;i++)
                      fprintf(pitchtextfile,"\n %d  %d",pitch[i],400);
                oldpointstop = pointstop;
                }
          free(pitch);
          fclose(pitchtextfile);
          fclose(pitchfile);
          }
    }
    //End Pitch procedure
```

```
//*********************************************************************
//These following procedures or functions are called from auto_recog
//*********************************************************************
//This function use for find absolute value of float value
float Abs(float value)
      {
      if (value<0) return(-value); else return(value);
      }
//Open and create file for LPC input and output
void open_all_file(char filename[50])
      {
      open_result=0;
      for (j=0;j<20;j++)
      input_wav_file_name[j]=filename[j];
      if ((input_wav_file = fopen(input_wav_file_name,"rb")) == NULL)
      {
      MessageBox(0,input_wav_file_name, "Can't open file", MB_OK);
      open_result=1;
      }
//This flag use to check where is this procedure called from
      if (autorecog==0)
      {
            if ((parameter_file = fopen("para.tmp","wb")) == NULL)
            {
            MessageBox(0,"para.tmp", "Can't create file", MB_OK);
            open_result=1;
            }
      }
   else
      {
      for(j=0;j<20;j++)
      parameter_file_name[j]=filename[j];
      for(j=0;j<20;j++)
            {
            if (parameter_file_name[j]=='.')
                  {
                  parameter_file_name[j+1]='l';
                  parameter_file_name[j+2]='p';
                  parameter_file_name[j+3]='c';
                  break;
                  }
            }
      if ((parameter_file = fopen(parameter_file_name,"wb")) == NULL)
      {
      MessageBox(0,parameter_file_name, "Can't create file", MB_OK);
      open_result=1;
      }
      }
      }
void prepare_data(void)
      {
//Find size of sound file
      length = -1;
      while (!feof(input_wav_file))
            {
            c = fgetc(input_wav_file);
            length++;
            }
      rewind(input_wav_file);
//Bypass the header of wave file
      length -= 44L;
//Set pointer to first data
      fseek(input_wav_file,44L,SEEK_CUR);
//Calculate number of frame
```

```
        number_of_frame = (length-
       FRAME_LENGTH+SHIFT_LENGTH)/SHIFT_LENGTH;
        signal[0] = (int)fgetc(input_wav_file);
        }
   void find_preemphasis_and_window(void)
   {
   static int called = FIRST_CALL;
   if (called == FIRST_CALL)
   //Calculate fist window must use full window data
   {
   for ( fl=1;fl<=FRAME_LENGTH;fl++ )
   {
   signal     [fl] = (int)fgetc(input_wav_file);
   preemphasis[fl] = (signal[fl] -  ADAPTIVE *(float)signal[fl-1]);
   windowed   [fl] = preemphasis[fl]*(0.54-0.46*cos(2*PI*(fl-
   1)/(FRAME_LENGTH-1)));
   }
   called = NEXT_CALL;
   }
   else
   {
   //Calculate other window use data less than first window
   for (fl=1;fl<=(FRAME_LENGTH - SHIFT_LENGTH);fl++)
   {
   signal     [fl] = signal[fl+SHIFT_LENGTH];
   preemphasis[fl] = preemphasis[fl+SHIFT_LENGTH];
   windowed   [fl] = preemphasis[fl]*(0.54-0.46*cos(2*PI*(fl-
   1)/(FRAME_LENGTH-1)));
   }
   //Calculate last window
   for (fl=(FRAME_LENGTH - SHIFT_LENGTH+1);fl<=FRAME_LENGTH;fl++)
   {
   signal     [fl] = (int)fgetc(input_wav_file);
   preemphasis[fl] = (signal[fl] - ADAPTIVE * (float)signal[fl-1]);
   windowed   [fl] = preemphasis[fl]*(0.54-0.46*cos(2*PI*(fl-
   1)/(FRAME_LENGTH-1)));
   }
   }
   }
   void find_autocorrelation(void)
         {
         float alk,akk,maxvalue,temp;
         int    m,i,j,k,l,maxrow;
         float  matrix_temp[P][P];
         for (k=0;k<=P;k++)
         {
               R[k] = 0;
               for (m=0;m<=FRAME_LENGTH-1-k;m++)
                     {
                     R[k] += windowed[m+1] * windowed[m+k+1];
                     }
         }
         for (i=0;i<P;i++)
         {
               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 (j=0;j<P;j++)
               matrix_temp[i][j] = A[i][j];
```

```
        for (i=0;i<P;i++)
        for (j=0;j<P;j++)
              {
              if (i==j) X[i][j] = 1; else X[i][j]=0;
              }
        for (k=0;k<P;k++)
              {
              maxrow = k;
              maxvalue = Abs(matrix_temp[k][k]);
              for (i=k+1;i<P;i++)
                  {
                  if (maxvalue < Abs(matrix_temp[i][k]))
                        {
                        maxvalue = Abs(matrix_temp[i][k]);
                        maxrow   = i;
                        }
                  }
              if (maxrow != k)
                  {
                  for (j=0;j<P;j++)
                  {
                  temp                  =  matrix_temp[k][j];
                  matrix_temp[k][j]     =  matrix_temp[maxrow][j];
                  matrix_temp[maxrow][j] =  temp;
                  }
                  for (j=0;j<P;j++)
                  {
                        temp          =  X[k][j];
                        X[k][j]       =  X[maxrow][j];
                        X[maxrow][j]  =  temp;
                  }
                  }
              akk = matrix_temp[k][k];
              for (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 (l=0;l<P;l++)
                  {
                  alk = matrix_temp[l][k];
                  if (k != l)
                  {
              for (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 find_coefficient(void)
        {
        for (i=0;i<P;i++)
              {
              alpha[i+1] = 0;
              for (k=0;k<P;k++)
                  {
                  alpha[i+1] += X[i][k]*phi[k+1];
                  }
```

```
            }
      }
void find_gain(void)
      {
      q = (float)R[0];
      for (i=1;i<=P;i++)
            {
            q -= ((float)alpha[i]*(float)R[i]);
            }
      gain = sqrt(q);
      }
void find_cepstrum(void)
      {
      cepstrum[0] = log(gain);
      for (m=1;m<N;m++)
      {
      if (m<=P)    cepstrum[m] = alpha[m];
      else     cepstrum[m] = 0;
      for (k=1;k<=m-1;k++)
      cepstrum[m] += ((float)k/(float)m) * cepstrum[k] * alpha[m-k];
      }
      }
void find_weight(void)
      {
      for (m=0;m<N;m++)
      weight[m] = cepstrum[m] * (1+((N)/2)*sin(PI*(float)m/(N)));
      }
//Write LPC parameter to file
void write_parameter_file(void)
      {
      for (i=0;i<N;i++)
            fwrite(&weight[i],sizeof(weight[i]),1,parameter_file);
      }
void close_all_file(void)
      {
      fclose(input_wav_file);
      fclose(parameter_file);
      }
void open_codebook_and_output_file(void)
{
open_result=0;
if ((codebook_file = fopen(codebook_file_name,"rb")) == NULL)
      {
      MessageBox(0,"Codebook.vq", "Can't open file", MB_OK);
      open_result=1;
      };
if((codebook_index_file=fopen(codebook_index_file_name,"wb"))== NULL)
  {
  MessageBox(0,codebook_index_file_name, "Can't create file", MB_OK);
  open_result=1;
 };
if((codebook_index_text_file=fopen(codebook_index_text_file_name,"wt"))==
NULL)
{
MessageBox(0,codebook_index_text_file_name, "Can't create file", MB_OK);
open_result=1;
}
}
void allocate_codebook_memory(void)
{
codebook=(float*)calloc(number_of_codebook*vector_dimension,sizeof(float)
);
if (codebook == NULL)
{
```

```
MessageBox(0,"Can't allocate memory for codebook", "ERROR", MB_OK);
open_result=1;
}
}
void read_codebook_file(void)
{
fread(codebook,sizeof(float),number_of_codebook*vector_dimension,
codebook_file);
}
void get_number_of_input_file_and_set_codebook_index_file_pointer(void)
      {
      int  flag=-1;
      item=atoi(transBuf.person)*atoi(transBuf.sound);
      if (autorecog==0) item=1;
            number_of_input_file = item;
      fseek(codebook_index_file,number_of_input_file,SEEK_SET);
      fwrite(&flag,1,1,codebook_index_file);
      }
void open_input_vector_file(void)
{
if ((input_vector_file = fopen(input_vector_file_name,"rb")) == NULL)
      {
      MessageBox(0,input_vector_file_name, "Can't open file", MB_OK);
      open_result=1;
      };
}
void find_number_of_frame(void)
{
int file_size;
file_size = -1;
while (!feof(input_vector_file))
      {
      c = fgetc(input_vector_file);
      file_size++;
      }
rewind(input_vector_file);
file_size -= file_size % (sizeof(float)*vector_dimension);
number_of_frame=(int)(file_size/ (sizeof(float)*vector_dimension));
}
void allocate_input_vector_and_min_index_memory(void)
{
input_vector=float*)calloc(number_of_frame*vector_dimension,sizeof(float)
);
if (input_vector == NULL)
{
MessageBox(0,"Can't allocate for input_vector","ERROR", MB_OK);
open_result=1;
}
min_index = (char *) calloc(number_of_frame, sizeof(char));
if (min_index == NULL)
{
MessageBox(0,"Can't allocate memory for min_index","ERROR", MB_OK);
open_result=1;
}
}
void read_input_vector(void)
{
fread(input_vector,sizeof(float),number_of_frame*vector_dimension,
input_vector_file);
}
void find_codebook_index(void)
{
for (nf=0;nf<number_of_frame;nf++)
{
```

```
for (nc=0;nc<number_of_codebook;nc++)
{
total_distance = 0;
distance       = 0;
for (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;
}
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 write_codebook_index_file(void)
      {
      int  index_file_pointer;
      index_file_pointer = (int)ftell(codebook_index_file);
      if (autorecog==0) file_number=0;
            fseek(codebook_index_file,file_number,SEEK_SET);
      fwrite(&number_of_frame,1,1,codebook_index_file);
      fseek(codebook_index_file,index_file_pointer,SEEK_SET);
      for (nf=0;nf<number_of_frame;nf++)
// adjust index value in range 1 - 64
      min_index[nf] += 1;
//Write codebook index to file
      fwrite(min_index,1,number_of_frame,codebook_index_file);
      }
void free_input_vector_and_min_index_memory(void)
      {
      free(input_vector);
      free(min_index);
      }
void close_input_vector_file(void)
      {
      fclose(input_vector_file);
      }
void write_codebook_index_text_file(void)
{
char item;
fclose(codebook_index_file);
if ((codebook_index_file = fopen(codebook_index_file_name,"rb")) == NULL)
{
MessageBox(0,codebook_index_file_name, "Can't create file", MB_OK);
open_result=1;
}
while  (!feof(codebook_index_file))
{
//Write codebook index to text file
fread(&item,1,1,codebook_index_file);
fprintf(codebook_index_text_file,"%2d\n",item);
}
}
```

```
void free_codebook_memory(void)
      {
      free(codebook);
      }
void close_all_vq_file(void)
      {
      fclose(codebook_file);
      fclose(codebook_index_file);
      fclose(codebook_index_text_file);
      }
//Read codebook index file
void load_unknown_word_file(void)
{
char temp;
if ((unknown_word_file = fopen(unknown_word_file_name,"rb")) == NULL)
      {
      MessageBox(0,unknown_word_file_name, "Can't open file", MB_OK);
      open_result=1;
      };
      fread(&T,1,1,unknown_word_file);
      fread(&temp,1,1,unknown_word_file);  /* for bypass FLAG (-1) */
      if (temp != -1)
      {
      MessageBox(0,"Invalid input file!", "ERROR", MB_OK);
      open_result=1;
      };
      O = (char *) calloc((int)T,sizeof(char));
      if (O == NULL)
      {
      MessageBox(0,"Can't allocate memory for O", "ERROR", MB_OK);
      open_result=1;
      };
fread(O,1,(int)T,unknown_word_file);
fclose(unknown_word_file);
}
void allocate_memory(void)
{
dt   = (double *) calloc((int)T*N1,sizeof(double));
ar   = (char   *) calloc((int)T*N1,sizeof(char));
q_st = (char   *) calloc((int)T  ,sizeof(char));
if ( (dt||ar||q_st) == NULL)
{
MessageBox(0,"Can't allocate memory for dt,ar,q_st", "ERROR", MB_OK);
open_result=1;
}
}
void free_mem(void)
      {
      free(dt);
      free(ar);
      free(q_st);
      }
//Read model parameter from model file
void load_model(int model_name)
{
if ((model_file = fopen(model_file_name[model_name],"rb")) == NULL)
{
MessageBox(0,model_file_name[model_name], "Can't open file", MB_OK);
open_result=1;
}
Buf_aprime=(float*)calloc(N1*N1,sizeof(float));
Buf_bprime=(float*)calloc(N1*K,sizeof(float));
fread(Buf_aprime,4,N1*N1,model_file);
fread(Buf_bprime,4,N1*K,model_file);
```

```
fclose(model_file);
for (i=0;i<N1;i++)
      for (j=0;j<N1;j++)
            a[i][j]=Buf_aprime[(i*N1)+j];
for (i=0;i<N1;i++)
      for (j=0;j<K;j++)
            b[i][j]=Buf_bprime[(i*K)+j];
}
//Calculate propability for each model
void viterbi(void)
      {
      for (i=0;i<N1;i++)
      {
      dt[0*N1+i] = (float)Pi[i] * (float)b[i][O[0]-1];
      ar[0*N1+i] = 0;
      }
      for (t=1;t<(int)T;t++)
      {
      for (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 (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 (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])];
      }
      }
//Find the maximum propability
void display_recognized_word(void)
      {
      dmax      = 0;
      max_index = 0;
      for (k=0;k<STORED_MODEL;k++)
            {
            if (dmax < p_st[k])
                  {
                  dmax = p_st[k];
                  max_index = k;
```

```
                }
            }
        }
//*************************************************************************
//This procedure is called from Commandautoport window procedure
//*************************************************************************
void auto_recog(char input[30])
{
autorecog=0;
//Call to procedure that use to calculate LPC
open_all_file(input);
if (open_result==0)
{
prepare_data();
for (nf=0;nf<number_of_frame;nf++)
        {
        find_preemphasis_and_window();
        find_autocorrelation();
        find_coefficient();
        find_gain();
        find_cepstrum();
        find_weight();
        write_parameter_file();
        }
close_all_file();
//Call to procedure that use to calculate VQ
open_codebook_and_output_file();
if (open_result==0)
{
allocate_codebook_memory();
if (open_result==0)
{
read_codebook_file();
get_number_of_input_file_and_set_codebook_index_file_pointer();
open_input_vector_file();
if (open_result==0)
{
find_number_of_frame();
allocate_input_vector_and_min_index_memory();
if (open_result==0)
{
read_input_vector();
find_codebook_index();
write_codebook_index_file();
free_input_vector_and_min_index_memory();
close_input_vector_file();
write_codebook_index_text_file();
if (open_result==0)
{
free_codebook_memory();
close_all_vq_file();
//Call to procedure that recognize model
load_unknown_word_file();
if (open_result==0)
{
allocate_memory();
if (open_result==0)
{
for (model=0;model<STORED_MODEL;model++)
        {
        load_model(model);
        if (open_result==0)
        viterbi();
        }
```

```
display_recognized_word();
itoa(max_index,max_index_text,10);
}
free_mem();
}
}
}
}
}
}
}
}
//*********************************************************************
//This procedure is procedure that respond to the respond table
//*********************************************************************
void TMyWin::Commandautoport()
      {
      //Assign Variable
      char  *sample;
      int   sumnoise,segment,tempstart,tempstop,no,L;
      int   samplewindow=300;
      int   sampleshift=50;
      long  *sqr;
      float *sum_sqr;
      div_t frame;
      int   lowpercent,mininterval,minnumber,oldtempstop;
      int   *segmentBuf,sumlength,averagenoise,*resegment;
      float averageenergy,segmentlevel;
      div_t segmentnumber;
      int   *a;
      int   wordnumber,beginword,endword,j;
      FILE  *eachwordfile;
      char  e;
      char  input[]="Input.wav";
      int   command[MAXSEGMENT];
      int   control;
      char  controltext[5];
      double      pop;
      int   oper,exist,pos;
      int   tmp,num1,num2;
      float answer;
      char  num1_text[10]="";
      char  oper_text[10]="";
      char  num2_text[10]="";
      char  answer_text[10]="";
      char  result_text[10]="";
      int   door=1,light=1;

//Begin of procedure
      int  check=MessageBox("Voice  drives  relay!","Click  ok  to  Start",
                            MB_OKCANCEL);
      if (check==IDOK)//10
      {
      for (;;)//9
      {
      open_result=0;
      g->SetRange(0,10);
//Record sound from soundcard
      openwav(input);
      if (open_result==1) break;
      if (open_result == 0)//8
      {
      recordwav();
      Sleep(500);
```

```
g->DeltaValue(1);
Sleep(500);
g->DeltaValue(1);
Sleep(500);
g->DeltaValue(1);
Sleep(500);
g->DeltaValue(1);
Sleep(500);
g->DeltaValue(1);
Sleep(500);
g->DeltaValue(1);
Sleep(500);
g->DeltaValue(1);
Sleep(500);
g->DeltaValue(1);
Sleep(500);
g->DeltaValue(1);
Sleep(500);
g->DeltaValue(1);
stopwav();
savewav(input);
closewav();
if (open_result==1) break;
if (open_result==0)//7
{
MessageBox("Wait", "Click", MB_OK);
//Caluculate Energy
if ((inputfile=fopen(input,"rb"))== NULL)
      {
      MessageBox("Can't open input wave file.", "ERROR!", MB_OK);
      open_result=1;
      }
if (open_result==1) break;
if (open_result==0)//6
{
L=0;
fseek(inputfile,44L,SEEK_CUR);
while (!feof(inputfile))
      {
      d = fgetc(inputfile);
      L++;
      }
sample=(char*)calloc(L,sizeof(int));
sqr=(long*)calloc(L,sizeof(long));
if ((sample==NULL)||(sqr==NULL))
      {
      MessageBox("Can't allocate memory for sqr or sample.",
                  "ERROR!", MB_OK);
      open_result=1;
      fclose(inputfile);
      }
if (open_result==0)//5
{
rewind(inputfile);
fseek(inputfile,44L,SEEK_CUR);
fread(sample,L,sizeof(char),inputfile);
for (i=0;i<L;i++)
      sample[i]=sample[i]-128;
for (i=0;i<L;i++)
      sqr[i]=sample[i]*sample[i];
free(sample);
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);
            open_result=1;
            fclose(inputfile);
            }
      if (open_result==0)//4
      {
      for (i=0;i<frame.quot-(samplewindow/sampleshift);i++)
            {
            sum_sqr[i]=0;
            for (no=0;no<samplewindow;no++)
                  sum_sqr[i]+=sqr[(i*sampleshift)+no];
            sum_sqr[i]/=samplewindow;
            }
      for (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);
      free(sqr);
//Find the boundary of words from recorded sound
      a=(int*)calloc(30,4);
      lowpercent=100;
      mininterval=10;
      innumber=10;
      segmentBuf=(int*)calloc(MAXSEGMENT,sizeof(int));
      if (segmentBuf==NULL)
            {
            MessageBox("Can't allocate memory.", "ERROR!", MB_OK);
            open_result=1;
            fclose(inputfile);
            }
      if (open_result==0)//3
      {
      averageenergy=0;
      for (i=0;i<frame.quot;i++)
            averageenergy+=sum_sqr[i];
      averageenergy/=frame.quot;
      segmentlevel=averageenergy*lowpercent*0.01;
      tempstart=-1;
      tempstop=-1;
      oldtempstop=0-mininterval-1;
      sum_sqr[-1]=0;
      sum_sqr[-2]=0;
      sum_sqr[(frame.quot)+1]=0;
      sum_sqr[(frame.quot)+2]=0;
      segment=0;
      for (i=0;i<frame.quot;i++)
            {
            if ((sum_sqr[i-1]<segmentlevel)&&(sum_sqr[i]>segmentlevel)&&
            (sum_sqr[i+1]>segmentlevel)&&(sum_sqr[i+2]>segmentlevel))
                  tempstart=i;
            if ((sum_sqr[i-2]>segmentlevel)&&(sum_sqr[i-1]>
            segmentlevel)&&(sum_sqr[i]>segmentlevel)
            &&(sum_sqr[i+1]<segmentlevel))
                  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;
                }
          }
    sumnoise=0;
    sumlength=0;
    for (i=0;i<frame.quot;i++)
          {
          if (sum_sqr[i]<segmentlevel)
                {
                sumnoise+=sum_sqr[i];
                sumlength++;
                }
          }
    if (sumlength==0)
          {
          MessageBox("Wave form is too bad!", "ERROR!", MB_OK);
          fclose(inputfile);
          open_result=1;
          }
    if (open_result==0)//2
    {
    averagenoise=sumnoise/sumlength;
    averagenoise=averagenoise*100*0.01;
    segmentBuf[-1]=-1;
    sum_sqr[frame.quot]=averagenoise-1;
    resegment=(int*)calloc(segment*2,sizeof(int));
    if (resegment==NULL)
          {
          MessageBox("Can't allocate memory.", "ERROR!", MB_OK);
          open_result=1;
          fclose(inputfile);
          }
    if (open_result==0) //1
    {
    resegment[-1]=-1;
    for (j=0;j<segment*2;j++)
          {
          if (segmentBuf[j]>resegment[j-1])
                {
                segmentnumber=div(j,2);
                if (segmentnumber.rem==0)
                      for (k=segmentBuf[j];sum_sqr[k]>averagenoise;k--)
                            resegment[j]=k;
                if (segmentnumber.rem!=0)
                for (k=segmentBuf[j];sum_sqr[k]>averagenoise;k++)
                            resegment[j]=k;
                }
          else
                resegment[j]=segmentBuf[j];
          }
//Create sound file from calculate boundary
    for (j=0;j<segment*2;j++)
          resegment[j]*=50;
    for (wordnumber=0;wordnumber<segment;wordnumber++)
          {
          rewind(inputfile);
          if ((eachwordfile=fopen("temp.wav","wb"))==NULL)
                {
                MessageBox("Can't create file.", "ERROR!", MB_OK);
                open_result=1;
                fclose(inputfile);
                }
```

```
            if (open_result==1) break;
            if (open_result==0)
                  {
                  beginword=resegment[wordnumber*2];
                  endword=resegment[(wordnumber*2)+1];
                  for (j=0;j<44;j++)
                        {
                        e=fgetc(inputfile);
                        putc(e,eachwordfile);
                        }
                  for (j=0;j<L;j++)
                        {
                        e=fgetc(inputfile);
                        if ((j>beginword)&&(j<endword))
                              putc(e,eachwordfile);
                        }
                  fclose(eachwordfile);
//Bring created file to be input of the recognizer
                  auto_recog("temp.wav");
//Use output from the recognizer to control relay
                  if (open_result==1) break;
                  if (open_result==0)
                        {
                        if (wordnumber==0)
                              if (p_st[0]>p_st[1])
                                    command[0]=0;
                              else
                                    command[0]=1;
                        else
                              if (wordnumber==1)
                                    if (p_st[2]>p_st[4])
                                          command[1]=2;
                                    else
                                          command[1]=4;
                              else
                                    command[wordnumber]=max_index;
                        }
                  }
            }
      fclose(inputfile);
      if ((command[0]==0)&&(segment==3)&&(door==0))
            MessageBox("The Door is already opened!", "OPEN THE DOOR",
                  MB_OK);
      if ((command[0]==0)&&(segment==3)&&(door==1))
            {
            door=0;
            outputport(13);
            MessageBox("The Door is openning..", "OPEN THE DOOR", MB_OK);
            }
      if ((command[0]==1)&&(segment==3)&&(door==1))
            MessageBox("The Door is already closed!", "CLOSE THE DOOR",
                  MB_OK);
      if ((command[0]==1)&&(segment==3)&&(door==0))
            {
            door=1;
            outputport(13);
            MessageBox("The Door is closing..", "CLOSE THE DOOR", MB_OK);
            }
      if ((command[0]==0)&&(segment==2)&&(light==0))
            MessageBox("The Light is already ON!", "TURN ON THE LIGHT",
                  MB_OK);
      if ((command[0]==0)&&(segment==2)&&(light==1))
            {
            light=0;
```

```
        outputport(14);
        MessageBox("The light is turning ON..", "TURN ON THE LIGHT",
                   MB_OK);
        }
if ((command[0]==1)&&(segment==2)&&(light==1))
        MessageBox("The Light is already OFF!", "TURN OFF THE LIGHT",
                   MB_OK);
if ((command[0]==1)&&(segment==2)&&(light==0))
        {
        light=1;
        outputport(14);
        MessageBox("The  Light  is  turning  OFF..",  "TURN  OFF  THE
                   LIGHT", MB_OK);
        }
}//1
}//2
}//3
}//4
}//5
}//6
}//7
}//8
}//9
}
}
```

กิตติกรรมประกาศ

รายงานฉบับนี้ สามารถสำเร็จลุล่วงไปได้ด้วยดี ก็ด้วยคำแนะนำทางด้านความรู้ และชี้แนะแนวทางในการศึกษา จากอาจารย์ที่ปรึกษา จึงขอกราบขอบพระคุณอาจารย์ ดร. ไกรสิน ส่งวัฒนา ไว้ ณ ที่นี้ด้วย รวมทั้งอาจารย์ท่านอื่น ๆ ที่ไม่ได้กล่าวนามไว้ ณ ที่นี้ด้วย

นอกจากนี้ยังต้องขอขอบคุณเพื่อน ๆ ทั้งหลายที่ได้ช่วยเหลือให้เสียง เพื่อนำมาใช้ในการทำปริญญานิพนธ์ฉบับนี้ และเพื่อนที่ได้ให้คำแนะนำช่วยเหลือทั้งทางด้านฮาร์ดแวร์ และทางโปรแกรมซึ่งได้แก่

1. นายวิศิษฐ์ รัตนชัยฤทธิ์
2. นายวชิรพล บุญญะโรคล
3. นายวุฒิชัย วุฒิอุดมเลิศ

เป็นต้นรวมทั้งท่านผู้ที่ไม่ได้เอ่ยนามไว้ด้วย

และทั้งนี้รายงานฉบับนี้คงไม่สามารถทำจนเสร็จได้ ถ้าไม่ได้รับการช่วยเหลือ กำลังใจทั้งหลายจากพ่อ แม่ และ เพื่อน ๆทุกคนที่ทั้งได้ให้คำปรึกษา และให้คำแนะนำต่าง ๆ เสมอมาจึงขอขอบคุณทุก ๆ ท่านที่กล่าวถึง และไม่ได้เอ่ยนามไว้ ณ ที่นี้ด้วย

บรรณานุกรม

[1] ทวี ประทุมทาน. "การตรวจรู้เสียงพูดภาษาไทยโดยใช้หน่วยพยางค์." วิทยานิพนธ์ปริญญามหาบัณฑิต ภาควิชาวิศวกรรมคอมพิวเตอร์ บัณฑิตวิทยาลัย จุฬาลงกรณ์มหาวิทยาลัย,2533.

[2] Lawrence R.Rabiner and Rolnald W.Schafer. "Digital Processing of Speech Signals." Prentice-Hall,Inc.,Engle Wood Cliffs,New Jersey,1978

[3] Lawrence R.Rabiner and Biing-Hwang Juang. "Fundamentals of Speech Recognition." PTR Prentice Hall Englewood Cliffs,New Jersey,1993.

[4] Lori F.Lamel, Lawrence R.Rabiner,Aaron E.Rosenberg and Jay G.Wilpon. "An Improved Endpoint Detector for Isolated Word Recognition". IEEE Transactions on Acoustics,Speech and Signal Processing.

[5] ศ.ดร.วัลลภ สุระกำพลธร. "การประมวลผลสัญญาณเชิงเลข." บริษัท ไดนาพริ้นท์ จำกัด กรุงเทพมหานคร,2533

[6] Lonnie C.Ludeman. "Fundamental of Digital Signal Processing."