โปรแกรมวิเคราะห์เสียงบนระบบวินโดวส์

Voice Analyzer run on Windows

จัดทำโดย

นาย	สมศักดิ์	สุขเทียม	รหัส	331009
Mr.	SOMSAK	SUKTHIAM	No.	331009
นาย	พงศ์ธร	รัตนกรวิทย์	รหัส	321196
Mr.	PONGTORN	RUTTANAGONRAVIT	No.	321196

นักศึกษา ภาควิชาคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

อาจารย์ที่ปรึกษา

รศ.ดร. ชม กิ้มปาน
อาจารย์ สมศักดิ์ มิตะถา

อาจารย์ประจำภาควิชาคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ADVISOR

Mr. CHOM KIMPAN
Mr. SOMSAK MITATHA

LECTURER IN DEPARTMENT OF COMPUTER ENGINEERING
FACULTY OF ENGINEERING

ปริญญานิพนธ์ปีการศึกษา 2535

ภาควิชาคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง

โปรแกรมวิเคราะห์เสียงบนระบบวินโดวส์

ผู้จัดทำ

นาย สมศักดิ์ สุขเทียม 331009

นาย พงศ์ธร รัตนกรวิทย์ 321196

(.....) อาจารย์ที่ปรึกษา

รศ.ดร. ชม กิ้มปาน

(.....) อาจารย์ที่ปรึกษา

อาจารย์ สมศักดิ์ มิตะถา

โปรแกรมวิเคราะห์เสียงบนระบบวินโดวส์

สมศักดิ์ สุขเทียม
พงศ์ธร รัตนกรวิทย์
ปีการศึกษา 2535

บทคัดย่อ

การประมวลผลสัญญาณเสียง โดยใช้คอมพิวเตอร์นี้ แบ่งออกเป็นขั้นตอนต่าง ๆ ดังนี้ ขั้นตอนแรกนั้น จะเป็นการรับเสียงจากภายนอก เพื่อนำเข้ามาเก็บ ไว้ในคอมพิวเตอร์ ก่อนเพื่อที่จะนำข้อมูล เสียงดังกล่าว เข้าสู่ขั้นตอนต่อไป คือขั้นตอนในการประมวลผล ซึ่งในขั้นตอนนี้ มีวิธีที่ใช้ได้หลายวิธี ซึ่ง ในโครงงานนี้ได้เลือกใช้ วิธีการวิเคราะห์ Spectrogram ของเสียงโดยการ นำข้อมูลเสียงที่ได้รับเข้ามา ซึ่ง อยู่ในฟังก์ชันของ เวลา(Time) และ แอมปลิจูด (Amplitude) แล้วเปลี่ยนให้อยู่ ในรูปฟังก์ชันของ เวลา (Time) , ความถี่(Frequency) และ พลังงาน(Power) วิธีที่ใช้ในการแปลง ฟังก์ชันดังกล่าวคือ ใช้สมการ ของการแปลงแบบ Fast Furier Transform หลังจากการ วิเคราะห์โดย Spectrogram แล้ว จะได้คุณสมบัติ เฉพาะ ของเสียงที่ นำมาวิเคราะห์ จากคุณสมบัติเฉพาะ ของเสียงที่วิเคราะห์ได้ จะนำมาเก็บไว้ เพื่อให้ คอมพิวเตอร์ ใช้ในการเปรียบเทียบ และ วิเคราะห์ เพื่อการรู้จำเสียงต่อไป

Voice Analyzer run on Windows

Somsak Sukthiam
Pongporn Ruttanagonravit
1992

Abstract

Voice Processing process by computer that has serveral step. Step one that recive voice to store in computer. Step two that is processing the voice. It has serveral type to process voice. In this project we use Spectrum Analusis to process voice. First the voice is displaying in Time domain. After use Spectrum Analysis the voice is displaying in Frequency domain. We use Feature extraction that is the result of Spectrum Analysis to be a pattern for voice. And we use the pattern to teach the computer to recognize voice.

สารบัญ

บทนำ

ในยุคปัจจุบันที่คอมพิวเตอร์ มีความสำคัญมากขึ้น ในทุกวงการ แต่การติดต่อกับคอมพิวเตอร์ โดยการใช้ คีย์บอร์ด หรือ เม้าส์ ก็ไม่ใช่ เรื่องง่าย ๆ สำหรับผู้ใช้โดยทั่วไป ระบบ การติดต่อกับ ผู้ใช้ ด้วยเสียง จะเป็นวิธีทางใหม่ สำหรับผู้ใช้คอมพิวเตอร์ การติดต่อด้วยภาษาพูด เป็นรูปแบบการ สื่อสาร ที่ถือได้ว่าดีที่สุด สำหรับมนุษย์ กับคอมพิวเตอร์

เทคโนโลยี ด้านการจดจำเสียงพูด จะทำให้ อุปสรรคกีดขวาง ระหว่าง คนกับ เครื่องจักร หมดไป มันจะทำให้ทุก ๆ คน สามารถใช้คอมพิวเตอร์ได้อย่าง ง่ายดาย และ ทำให้ผู้ ที่ใช้คอมพิวเตอร์ เป็นประจำ สามารถใช้ได้อย่างมีประสิทธิภาพมากขึ้น

นอกจากนี้ การใช้เสียงพูดยังมีข้อดีอีกมากมาย การป้อนข้อมูล ทำได้อย่างรวดเร็ว เพราะว่า โดยทั่วไป คนเราสามารถพูดได้ โดยเฉลี่ยถึง 200 คำต่อนาที แต่การใช้คีย์บอร์ด ทำได้อย่างดี ก็แค่ 60 คำต่อนาที และในสถานการณ์ หรือ สถานที่ที่มีอุปสรรค ในการใช้เครื่อง คอมพิวเตอร์ เช่น การทำงาน ในที่มืด การทำงานที่มือไม่ว่าง การขับรถยนต์ และ อื่น ๆ อีกมาก การสั่งงานด้วยเสียงพูดทำให้อุปสรรค เหล่านี้หมดไปอย่างง่ายดาย

บทที่ 1 ทฤษฎีการตรวจรู้เสียง

ระบบการตรวจรู้เสียง

เป็นเวลามากกว่ายี่สิบปี ที่ได้มีการค้นคว้า วิจัยด้านการจดจำเสียงพูด รูปแบบต่าง ๆ สามารถเห็นได้อย่างชัดเจนมาก ได้แก่

– การขึ้นตรงต่อผู้พูด (Specker dependence) และการไม่ขึ้นตรงต่อผู้พูด (Speaker-independence)

ระบบขึ้นตรงต่อผู้พูด จะต้องได้รับการฝึกเพื่อ ให้จดจำเสียงพูดเฉพาะบุคคลไป และระบบที่ไม่ขึ้นตรงต่อผู้พูด สามารถจดจำเสียงของใครก็ได้ แต่ความถูกต้องจะน้อยกว่า

– การพูดแบบเป็นคำ ๆ และการพูดแบบต่อเนื่อง (Continous speech)

ระบบพูดเป็นคำ ๆ จะต้องมีการหยุดพูดระหว่างคำ ส่วนระบบพูดแบบ ต่อเนื่อง สามารถรับคำพูดได้แบบต่อเนื่อง ในลักษณะที่เหมือนธรรมชาติมากกว่า แต่มีข้อเสียคือ ระบบจะมีความซับซ้อน และผิดผลาดมากกว่า

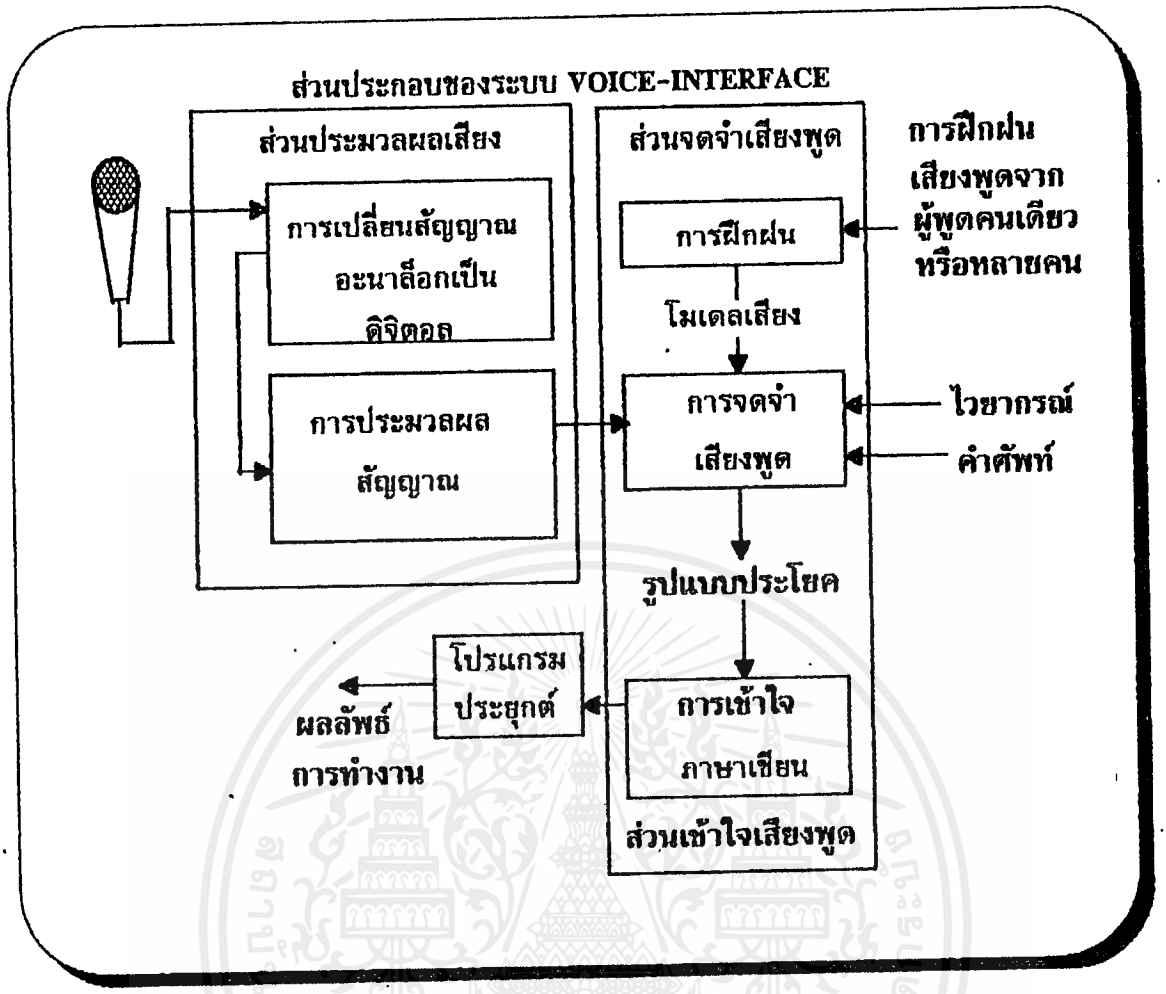
– ขนาดของคำศัพท์ และ ความซับซ้อนของไวยากรณ์

ขนาดของคำศัพท์ที่เครื่องสามารถจดจำได้ และ ไวยากรณ์ ที่กำหนดรูปประโยค ที่เครื่องสามารถแยกแยะได้ ถ้าระบบที่มีจำนวนคำศัพท์น้อย และ รูปแบบประโยคจำกัด จะซับซ้อนน้อยกว่า แต่ถ้าเป็นระบบที่มีจำนวน คำศัพท์มาก และ รูปแบบประโยคหลาย ๆ แบบ จะมีประโยชน์มากกว่า

ระบบติดต่อกับผู้ใช้ ด้วยเสียง (Voice-interface system) ทั้งที่มีอยู่ในท้องตลาด และที่วิจัยพัฒนากันอยู่นั้น ต่างก็มุ่งหวังที่จะให้ ระบบที่มีความถูกต้องสูง แต่ทุก ๆ ระบบก็ถูกจำกัดไว้ด้วยคุณสมบัติ ทั้งสามข้อ ดังที่กล่าวไปแล้ว ดังนั้น เวลาที่จะเลือกใช้งานระบบนี้ จะต้องพิจารณา และเลือกอย่างระมัดระวัง เพื่อให้เหมาะสมกับ โปรแกรมประยุกย์ที่ใช้ด้วย

ตามรูปที่ 1 แสดงให้เห็นถึงโครงสร้างโดยทั่วไป ของระบบติดต่อกับผู้ใช้ด้วยเสียง ทุก ๆ ระบบประกอบด้วย ส่วนสำคัญ สามส่วนดังนี้ ส่วนประมวลผลเสียงพูด (speech processing) ส่วนจดจำเสียงพูด (speech recognition) และ ส่วนเข้าใจเสียงพูด (speech understanding)

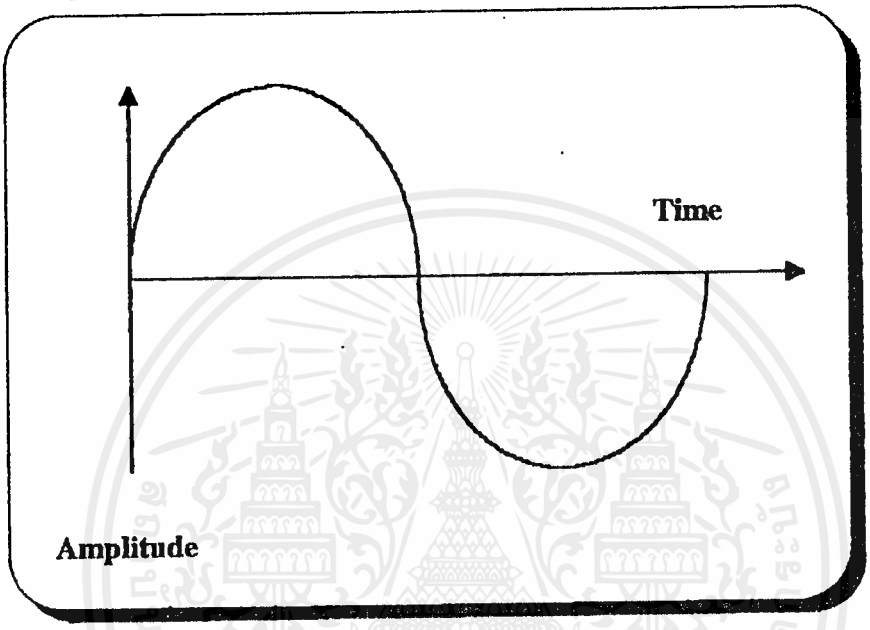
ในส่วนประมวลผลเสียง จะประกอบไปด้วย สองส่วนย่อย คือ ส่วน การเปลี่ยน สัญญาณ อะนาล็อก เป็นดิจิตอล เพื่อเปลี่ยน สัญญาณเสียง ที่อยู่ในรูปอะนาล็อก ให้เป็น สัญญาณเสียงแบบดิจิตอล แล้วนำสัญญาณเสียงแบบดิจิตอลนี้ ไปประมวลผลโดยคอมพิวเตอร์ ต่อไป และอีกส่วนหนึ่ง คือส่วนการประมวลผลสัญญาณ ขั้นตอนนี้ถือเป็นขั้นตอนหลักขั้นตอนหนึ่ง ในการรู้จำเสียง หลักการคือ นำข้อมูลเสียง ที่ได้มาในรูปโดเมนของเวลา กับ แอมปลิจูด มาเปลี่ยนให้อยู่ในรูป โดเมนของ ความถี่ เวลา และ ความเข้มของเสียง ซึ่งจากขั้นตอนนี้ จะนำไปวิเคราะห์หา เอกลักษณ์ของเสียงต่อไป



รูปที่ 1 ส่วนประกอบของระบบ VOICE-INTERFACE

การรับเสียงและการแปลงสัญญาณ

ในการรับรู้เสียง จากผู้ใช้นั้นจะเริ่มต้น จากเมื่อผู้ใช้ พูดคำลงใน ไมโครโฟน เมื่อไมโครโฟน รับเสียงแล้ว จะแปลงเสียงนั้น ให้เป็นสัญญาณไฟฟ้า ของเสียง ซึ่งจะอยู่ในรูปของสัญญาณแบบอะนาล็อก ดังรูปที่ 2

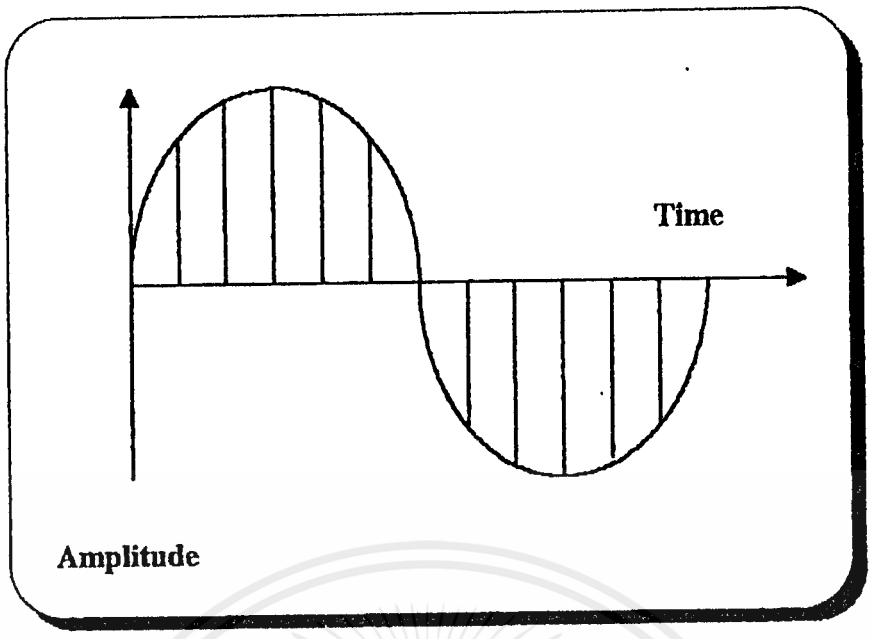


รูปที่ 2 สัญญาณอะนาล็อก

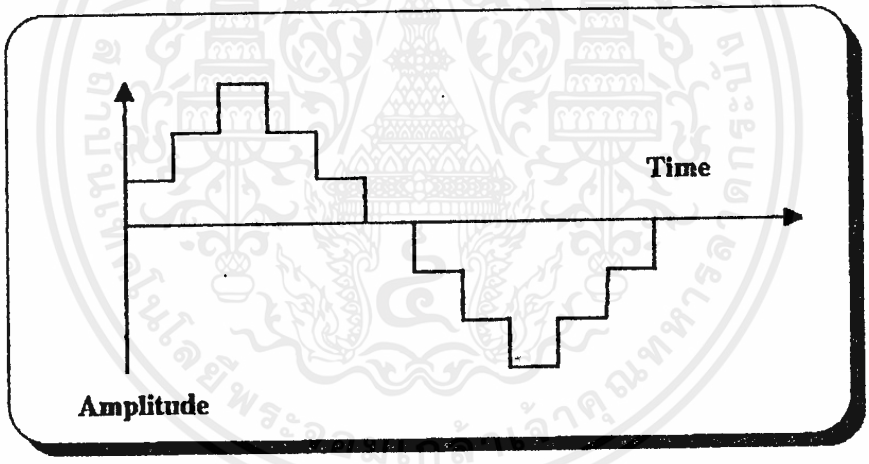
จากสัญญาณเสียงแบบอะนาล็อกที่ได้นี้ จะต้องนำมาเปลี่ยนให้เป็นสัญญาณ ดิจิตอลเสียก่อน เพื่อที่จะนำไปประมวลผลโดย คอมพิวเตอร์ ต่อไป สำหรับ ตัวที่จะทำหน้าที่แปลงสัญญาณจากอะนาล็อกเป็นดิจิตอล จะเรียกว่า Analog/Digital Converter ซึ่งในโครงงานนี้ได้ใช้ Sound Blaster Card เป็นตัวรับเสียง และ แปลงสัญญาณเสียงอะนาล็อก ให้เป็น ดิจิตอล

โดยทั่วไปแล้ว เสียงของมนุษย์ ที่ใช้พูดกัน จะมีความถี่อยู่ในช่วง ไม่เกิน 5 KHz ซึ่งโดยหลักการของการแปลงสัญญาณอะนาล็อก ให้เป็นดิจิตอล จะใช้ อัตราการแปลงข้อมูลเป็น 2 เท่าของความถี่ จึงเลือกใช้อัตราการแปลงข้อมูล ด้วยความถี่ 11.025 KHz โดยข้อมูลที่ได้ จะมีขนาด 8 bit ต่อหนึ่งสัญญาณดิจิตอล

จากสัญญาณเสียงที่เป็นอะนาล็อก เมื่อทำการแปลง เป็นสัญญาณ ดิจิตอล แล้วจะได้ข้อมูลเสียง ซึ่งอยู่ในรูป ของสัญญาณแบบ ดิจิตอล ดังรูปที่ 4

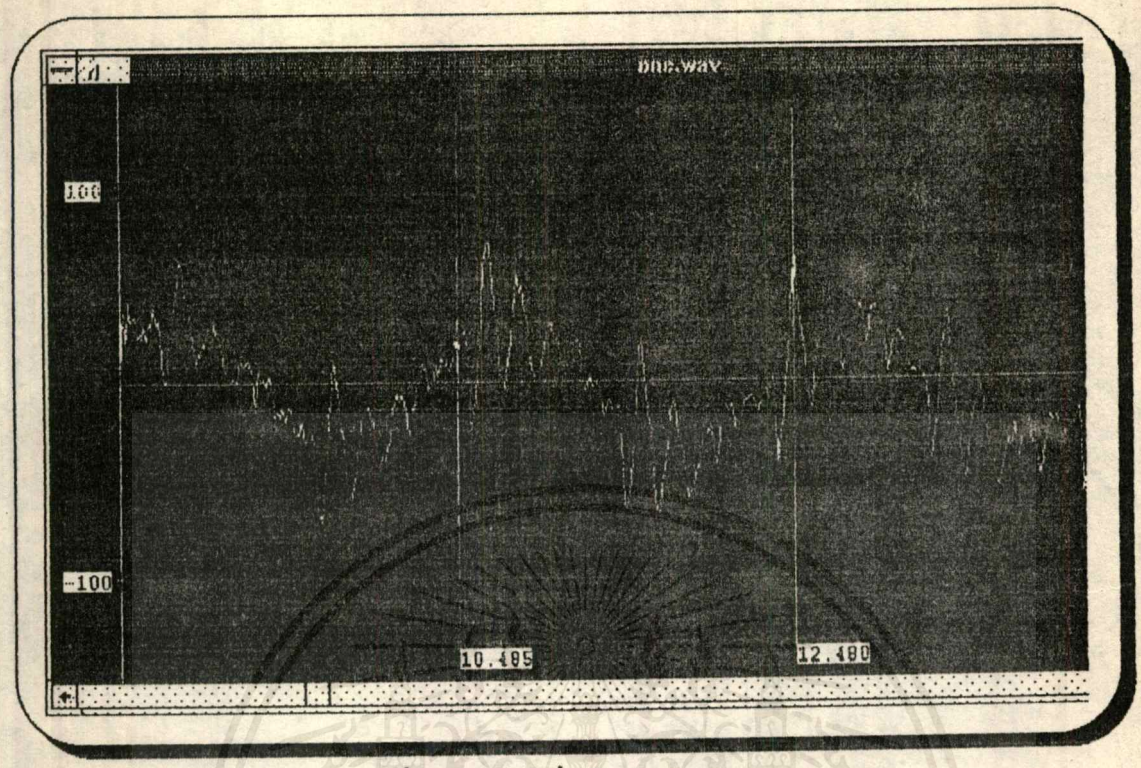


รูปที่ 3 การ Sampling สัญญาณ

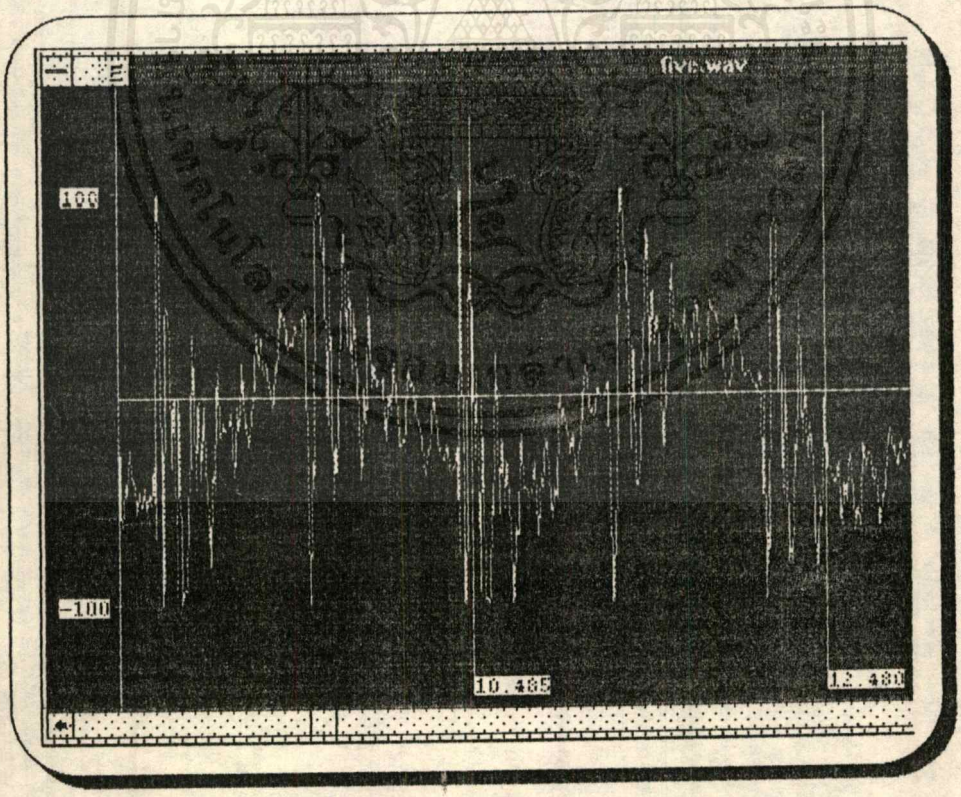


รูปที่ 4 สัญญาณดิจิตอล

ตัวอย่างของสัญญาณเสียงที่อยู่ในรูปโดเมนของเวลา



รูปที่ 5 คำว่า "หนึ่ง" ใน Time domain



รูปที่ 6 คำว่า "ห้า" ใน Time domain

การประมวลผลสัญญาณเสียง

สัญญาณเสียงที่ได้ จากวงจรแปลงสัญญาณนั้น ในตอนแรก จะได้สัญญาณ ซึ่งอยู่ในโดเมนของ เวลา กับ Amplitude เท่านั้น ซึ่งยังไม่สามารถที่จะใช้ แสดงลักษณะ และ คุณสมบัติเฉพาะของเสียงแต่ละเสียงได้ จึงต้องมีการ ปรับเปลี่ยน และหาค่า พารามิเตอร์ ของสัญญาณ ให้มีความละเอียดมากขึ้นโดย อาศัยหลักเกณฑ์ที่ว่า มนุษณ์สามารถรับฟังเสียง และจะแยกแยะความแตกต่างของเสียงที่รับฟังนั้นในรูป ความแตกต่างของ ช่วงความถี่ และยังพิจารณาถึงค่า ปริมาณความดังของเสียง ในช่วงความถี่ต่าง ๆ ที่แปรเปลี่ยนไปในช่วงแกนของเวลาอีกด้วย และ สิ่งสำคัญอีกประการหนึ่ง มนุษย์รู้จักเสียง จากค่าพลังงานของเสียง ไม่ใช่ค่าความต่างศักย์ หรือค่า Amplitude ของเสียง นั้นคือ เราอาจพิจารณาได้ว่า สมการของสัญญาณเสียงที่เหมาะสม ในการที่จะให้ คอมพิวเตอร์สามารถตรวจรู้เสียงได้นั้น ควรจะเป็นฟังก์ชัน ที่ประกอบไปด้วยค่า พลังงาน (Power) ค่าความถี่ (Frequency) และค่าเวลา (Time) ดังนี้คือ

$$\text{Voice function} = f(P,F,T) \quad \text{----------} \quad (1)$$

P คือ ค่าพลังงานของแต่ละความถี่

F คือ ค่าความถี่

T คือ ค่าเวลา

จากสัญญาณเสียงที่รับเข้ามาในรูป ของค่าปริมาณเสียงในช่วงเวลา หรือเป็นฟังก์ชันของ ปริมาณ (Amplitude) และ เวลา (Time) จะถูกเปลี่ยนให้อยู่ในรูปของความถี่เสียก่อน โดยใช้ หลักการของ Discreate furier transtorm (DFT) ซึ่ง DFT นี้จะมีสมการดังนี้

$$X(k) = (\tfrac{1}{n})\Sigma_{n=0}^{N-1} X(n)W_n^{nk} \quad k=0,1,...,N-1 \quad \text{----------} \quad (2)$$

โดยที่ X(k) คือค่าปริมาณเสียงในช่วงความถี่ $\frac{k}{2N}$

N คือค่าตัวอย่างของเสียงที่ใช้ในการคำนวณ

$W_n^{nk} = e^{-j2\delta nk/N}$ คือค่าเชิงซ้อนมีสมการเป็นรูปคาบเวลา

แต่เนื่องจากการพิจารณา โดยใช้ DFT จะเป็นการพิจารณาถึงสัญญาณ ที่มีลักษณะเป็นคาบเวลา และมีค่าคงที่ต่อเนื่องกันไป จนถึงอนันต์ ซึ่งตามความเป็นจริงแล้ว สัญญาณที่อ่านเข้ามามิได้เป็นเช่นนั้น จึงต้องมีการคำนวณ ด้วยสมการที่เหมาะสม เพื่อกำหนดให้ ความสำคัญ กับสัญญาณในช่วงเวลาปัจจุบัน และใกล้เคียง สมการดังกล่าวเรียกว่า Weighting Functions ซึ่งมีอยู่หลายแบบ เช่น Hanning , Parzen , Welch แต่ในโครงงานนี้ ได้เลือกใช้ ฟังก์ชันของ Parzen ซึ่งเป็นฟังก์ชันที่ดีที่สุด

จากการใช้ DFT เราจะได้ ฟังก์ชันในรูปของค่า พลังงาน ความถี่ และ เวลา ดังนี้

$$\text{Voice function} = f(P,F,T) \quad \text{----------} \quad (3)$$

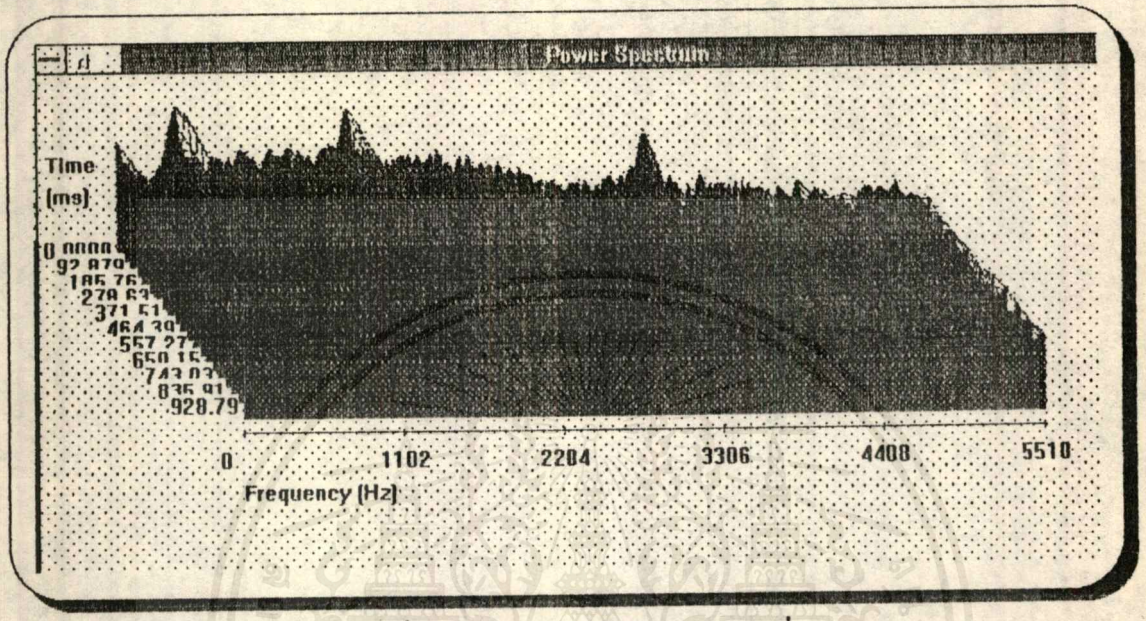
แต่จากการที่มนุษย์ ได้ยินเสียงในรูปของพลังงาน และหูของมนุษย์ รับฟังสัญญาณในรูปสเกล Log เราจึงต้องเปลี่ยน และ ปรับสัญญาณให้เหมาะสม โดยเลือกใช้สมการ

$$P = 10\text{Log}(A/A_0)$$

A คือ ค่าพลังงานของเสียงในแต่ละความถี่

A_0 มีค่าเป็น 1

เมื่อกำหนดให้ A_0 มีค่าเป็น 1 และปรับค่า สัญญาณก่อนเข้า DFT ให้มีค่าไม่เกิน 1 จะได้สัญญาณในรูป ของ Spectral dencity ที่มีความเข้มของสัญญาณไม่เกิน 0 เดซิเบล ดังตัวอย่าง ของ Spectral dencity ดังรูป



รูปที่ 7 สัญญาณเสียงในโดเมนความถี่

จากรูปจะเป็นกราฟแสดงเสียง ในรูปฟังก์ชันของ ค่าความถี่ , ค่าพลังงาน และ ค่าเวลา คือในแกนนอนจะเป็น ค่าความถี่ แกนเฉียงจะเป็นค่าของเวลา ส่วนในแกนตั้ง จะเป็นค่าพลังงาน จากรูปกราฟที่ได้นี้ จะนำไปใช้ในขั้นตอนต่อไป คือขั้นตอนของการทำ Feature Extraction คือการดึงเอาลักษณะเฉพาะของแต่ละเสียงออกมา เพื่อนำไปทำเป็น โมเดล สำหรับใช้ในการเปรียบเทียบ เพื่อการ Recognition ต่อไป

ส่วนจดจำเสียงพูด

ส่วนจดจำเสียงพูดจะทำการค้นหา และจับคู่เสียงที่พูด เข้ามากับ โมเดล หรือ ฐานข้อมูลเสียง ซึ่งได้รับการฝึก และ เรียนรู้ จากตัวอย่างผู้พูด นอกจากนี้ ยังทำหน้าที่ แยกแยะคำศัพท์ และ หลักไวยากรณ์ที่พูด

โดยปกติ ผู้ใช้จำเป็นต้องทดสอบ การพูดเพื่อสร้างโมเดลเสียง ขึ้นมา ก่อนใช้งานระบบได้ บางระบบจะทดสอบ ให้พูดแตกต่างกัน เช่น พูดเป็นคำ วลี หรือ พยางค์ บางระบบต้องพูดทุก ๆ คำ ที่จะใช้งาน เมื่อระบบมีขนาดเล็ก และบางระบบพูดเสียงบางคำเท่านั้น

สำหรับเทคนิคด้านการแยกแยะ คำศัพท์ ที่คำแต่ละคำที่เราพูดนี้หมายความถึงอะไร นอกจากวิธีที่เปรียบเทียบ คำศัพท์ตัวต่อตัวแล้ว ระบบส่วนใหญ่จะเอาการวิเคราะห์ ด้านไวยากรณ์ และ ข้อจำกัด ของแต่ละภาษามาแยกแยะ เพื่อกำหนดแนวทางที่เป็นไปไม่ได้ออก เช่น เมื่อเราพูดคำว่า "เรา" ระบบจะทำการแยกสระ ออกมาก่อนเป็น เ-า เพื่อนำไปใช้ เปรียบเทียบ เฉพาะคำในกลุ่มนี้เท่านั้น

ส่วนเข้าใจเสียงพูด

ส่วนเข้าใจเสียงพูดทำหน้าที่ เข้าใจ และ เปลี่ยนข้อความเหล่านี้ ให้เป็นคำสั่ง ที่แน่-นอน สำหรับสั่งงานเครื่องคอมพิวเตอร์ ระบบในปัจจุบัน ใช้งานได้ดีกับระบบ ขนาดเล็ก และจำกัดรูปแบบการพูด แต่ยังไม่ดีพอ สำหรับการพูดที่ ไม่มีการจำกัด รูปแบบประโยค

สำหรับการประยุกต์ ใช้งานบางงาน เช่น การป้อนข้อมูล (Data Entry) การสั่งงานด้วยคำสั่ง เป็นต้น เพียงแต่ต้องการให้ระบบ จดจำได้ถูกลำดับ ของคำที่พูด ออกมาเท่านั้น ไม่จำเป็นที่จะต้องเข้าใจ ถึงรูปประโยค และความหมายมากนัก แต่ในงานบางอย่าง ระบบต้องเข้าใจ และ สามารถแยกแยะความแตกต่าง ของแต่ละประโยค และความหมายได้ ตัวอย่างเช่น ในการถามข้อมูลว่า "จำนวนลูกค้าใหม่ในหน่วยงานมีเท่าไหร่" จำเป็นที่จะต้อง ประมวลผลที่ซับซ้อนพอดู เพื่อให้เข้าใจถึงคำว่า "ใหม่" นอกจากนี้ ระบบไม่เพียงแต่จะ พิจารณาถึงรูปประโยค ของประโยคคำถามปัจจุบันเท่านั้น แต่จะต้องต่อเนื่อง และสอดคล้องกับคำถาม และคำตอบก่อนหน้านี้ด้วย

ในปัจจุบัน การเข้าใจเสียงพูด และภาษา ไม่จำกัดรูปแบบประโยค เป็นสิ่งที่ยากมาก และยังไม่สามารถหาวิธีการ ที่เหมาะสมออกมาได้ แต่สำหรับระบบ ที่มีการจำกัดรูปแบบ ประโยค สามารถเป็นไปได้

บทที่ 2 ทฤษฎีของระบบวินโดส์

แนวคิดเบื้องต้นของ WINDOWS

Microsoft Windows เป็นระบบ Window ซึ่งสามารถรันได้หลายงานพร้อมกันแต่เป็นระบบหลายงาน ในลักษณะที่เรียกว่า Non-preemtive นั่นคือ ความสามารถในการรันหลายงานต้องได้จากความร่วมมือของทุก ๆ Application ที่รันในขณะนั้น

การเขียนโปรแกรมบนวินโดส์หรือบน GUIs อื่น ๆ ต่างจากการเขียนโปรแกรมตรงที่ดอสมีลักษณะ เป็นงานเดียว โปรแกรมที่รันอยู่จะครอบครองทรัพยากร (เช่นCPU, MEMORY, PRINTER, KEYBOARD เป็นต้น) แต่เพียงผู้เดียว เมื่อโปรแกรมบนดอสรอรับคีย์บอร์ด จะไม่สามารถไปทำอย่างอื่นได้ แต่โปรแกรม บนวินโดส์จะต้องแชร์ทรัพยากรเหล่านี้ร่วมกัน คือเมื่อโปรแกรมใช้ทรัพยากรจนเสร็จเรียบร้อยแล้ว ต้องคืน ทรัพยากรกลับคืนไป

Message

Message คือข้อมูลข่าวสารที่บอกว่าเกิดอะไรขึ้นกับโปรแกรม โปรแกรมทำเพียงแต่คอยแมสเสจ เหล่านั้น และเมื่อมีแมสเสจเข้ามา โปรแกรมก็จะตอบสนองแมสเสจเหล่านั้นตามชนิด ของแมสเสจ ที่มาของ แมสเสจ นั้นอาจจะมาจากวินโดส์หรือมาจากโปแกรมอื่น ๆ ก็ได้โดย โปรแกรมสามารถส่ง แมสเสจ ไปยัง โปรแกรมอื่นได้

ตัวอย่างแมสเสจที่เห็นได้ชัดคือแมสเสจจากอุปกรณ์อินพุต เมื่อมีการคลิกเมาส์หรือ กดคีย์บอร์ดขณะที่โปรแกรมกำลังรัน วินโดส์จะจัดการข้อมูล ที่มาจากเมาส์หรือคีย์บอร์ด และ สร้างเป็นแมสเสจส่งมา ให้โปรแกรมของเราโปรแกรมก็จะตอบสนองตามชนิด ของแมสเสจ เมื่อ ตอบสนองแล้วจะต้องกลับไปคอยแมสเสจอีก

จากการรอคอยแมสเสจนี้เอง วินโดส์จะเอาเวลาที่รอคอยแมสเสจไปทำอย่างอื่นเช่นส่งแมสเสจ ไปยังโปรแกรมอื่น เมื่อโปแกรมนั้นตอบสนองแมสเสจ เสร็จเรียบร้อยแล้วก็จะคอย แมสเสจต่อไปอีก ถึงเวลานี้อาจจะเกิดแมสเสจของโปแกรมอีก โปรแกรมก็จะคอยสนอง ต่อแมสเสจนั้นแล้วกลับไปคอยอีก เป็นอย่างนี้เรื่อยไป ซึ่งจะทำให้เป็นระบบหลายงานได้

Message Queue

แอพพลิเคชันทุกตัวจะมีสิ่งที่เรียกว่า message queue ซึ่งจะเป็นที่เก็บแมสเสจต่าง ๆ ที่ส่งมาให้ แอพพลิเคชันนั้น ตามปกติแล้วแอพพลิเคชันจะได้รับแมสเสจจาก message queue

ของตัวมันเองตามลำดับ ก่อนหลังแต่จะมีบางกรณีที่แอพพลิเคชันจะได้รับแมสเสจโดยไม่ผ่าน message queue

โครงสร้างหลักของ Windows Application

include file

Function prototypes

Variables

Main function WinMain()

Initialization function InitApplication()

Creating Window InitInstance()

Window function MainWndProc()

Paint function PaintFunc()

Include file

```
#include <windows.h>
#include "hellowin.h"
```

สำหรับไฟล์ windows.h นั้นเป็นไฟล์ที่มาพร้อมกับชุด SDK ของไมโครซอฟต์และจะต้องใส่เข้าไว้ใน ซอร์สโค้ดเสมอ ไฟล์นี้มีขนาดค่อนข้างใหญ่มาก (108,580 byte) ซึ่งประกอบไปด้วย ค่าคงที่ ,ชนิดข้อมูล แบบโครงสร้าง (Structure type) , แมคโคร,ชนิดของแมสเสจต่างๆ และโปรโตไทพ์ฟังก์ชัน ของวินโดวส์ ทั้งหลายเป็นต้น

ส่วนไฟล์ hellowin.h เป็นโปรโตไทพ์ฟังก์ชันของ windows application

Variables

เป็นส่วนตัวแปรโกลบอลของโปรแกรมเช่น

```
HANDLE hInst;                  /* Instance of application */
HWND   hWndMain                /* Main window             */
char   szAppIName[]="Hellowin"; /* Application name       */
char   szTitleName[]="Sample First Application";
```

```
char  szMessage[]="Hello, Windows";
```

Function Prototype

เป็นส่วนที่ประกาศเอาไว้ว่าฟังก์ชันทั้งหลายใน windows application มีอะไรบ้าง return ชนิดตัวแปร แบบใด และมี parameter และชนิดใดอยู่บ้าง การทำเช่นนี้มีประโยชน์มาก เพราะเป็นการลดข้อผิดผลาด ของโปรแกรมและการรายงานข้อผิดผลาด (warning) ขณะที่ทำการ compile ตัวอย่างของ Prototype

HELLOWIN.H

```
int PASCAL WinMain(HANDLE,HANDLE,HANDLE,LPSTR,int);
BOOL InitApplication(HANDLE);
BOOL InitInstance(HANDLE,int);
void PaintFunc(HWND);
long FAR PASCAL MainWinProc(HWND,unsigned,WORD,LONG);
```

Main function

ส่วนนี้คือ function WinMain() function นี้ถ้ามองกันคร่าว ๆ แล้วก็คือส่วน main()ใน program ภาษา C ทั่ว ๆ ไปนั้นเอง กล่าวคือ เป็นจุดเริ่มต้นของการทำงานที่ตัว windows จะเรียกเข้ามาเมื่อเริ่ม run application ซึ่งมีลักษณะการทำงานดังนี้

```
int PASCAL WinMain(hInstance,hPrevInstance,lpCmdLine,nCmdShow)
HANDLE hInstance;              /* current instance */
HANDLE hPrevInstance           /* previous instance */
LPSTR  lpCmdLine               /* command line */
int nCmdShow                   /* show window type (open/icon */
{
    MSG msg;                   /* message */
    if(!hPrevInstance)         /* other instance of app running */
    if(!InitApplication(hInstance)) /* Initialize shared things */
        return  (FALSE);       /* Exits if unable to initialize */
    /* Perform initialization that apply to specific instance */
    if(!InitInstance(hInstance,nCmdShow))
        return (FALSE);
    /* Acquire and dispatch message until a WM_QUIT message is
```

```
    receive.  */
  while(GetMessage(&msg,/* message structure */
    NULL,          /* handle of window receiving the
                      message */
    NULL,          /* lowest message to examine */
    NULL))         /* highest message to examine */
  {
     TranslateMessage(&msg); /* Translates virtual key codes */
     DispatchMessage(&msg);  /* Dispatch message to window */
  }
  return (msg,wParam); /* Return the value from PostQuitMessage */
```

Parameter hPrevInstance จะเป็นตัวบอกว่า application นี้ถูกเปิดมาแล้วหรือยัง ถ้ายังค่านี้จะเป็นศูนย์ ต้องทำการเรียก function InitApplication() ก่อน

Register window class

เป็นการลงทะเบียน (register)ให้กับ application เพื่อให้ windows รู้จัก classนี้โดยก่อนที่จะสร้างหน้าต่าง ใด ๆ ขึ้นมาจะต้องบอกไปด้วยว่าจะใช้ class อะไรซึ่ง class นี้จะมีส่วนหนึ่งที่ windows เองก็ ใช้และได้ ลงทะเบียนไว้ก่อนแล้วที่เรียกว่า

Predefined window class อันได้แก่ EDIT,BUTTON,LISTBOX,SCROLLBAR และ STATIC เป็นต้น หากเราสร้างหน้าต่างที่มี class ตัวที่ windows รู้จักแล้ว (จาก predefined window class หรือจาก การที่เราทำ Registering window class ไปแล้ว) เราสามารถที่จะสร้าง windows class เหล่านี้ได้ทันที โดยไม่ต้องมาลงทะเบียนซ้ำอีก

วิธีการลงทะเบียนเพื่อสร้าง class ใหม่จะเรียกผ่าน function

```
BOOL RegisterClass(LPWND CLASS lpWndClass)
```

โครงสร้างข้อมูลของ WNDCLASS เป็นดังนี้

```
typedef struct tag WNDCLASS
{
   WORD style;                  /* window class type */
   long(FAR PASCAL *lpfnWndProc); /* pointer of window function */
   int cbClsExtra;  /* extra bytes end of structure */
   int cbWndExtra;  /* extra byte end of window instance */
```

```
    HANDLE hInstance;   /* window class instance */
    HICON hIcon;        /* window class icon */
    HCURSOR hCursor;    /* window class cursor */
    HBRUSH hBrBackground; /* background color */
    LPSTR lpszMenuname    /* menu associated with window class */
    LPSTR lpszClassName   /* window class name */
}WNDCLASS;
```

lpfnWndProc จะชี้ไปยัง Window function ที่เป็นเจ้าของ class นี้ (ในที่นี้คือ MainWndProc())

hInstance เป็นค่า instance ของ application

hIcon เป็นค่า handle ของ icon ที่จะใช้เป็นสัญลักษณ์ของ class (สามารถใช้ชุดเครื่องมือของ SDK คือ SDK PAINT หรือ tool ตัวใด ๆ ก็ได้มาสร้าง) ซึ่งตอนที่ application ถูกทำให้เล็กสุด (minimize) windows จะนำรูป icon นี้มาโชว์แทน แต่ตอนนี้เราจะใช้ icon จาก GDI ของ windows ไปก่อนคือเรียก

```
LoadIcon(NULL,IDI.APPLICATION)
```

hCursor เป็นค่า handle ของ cursor และเช่นกันเราสามารถสร้างโดยใช้ชุดเครื่องมือของ SDK มาสร้าง หรือใช้ของ GDI ก็ได้ ซึ่งมีอยู่หลายตัว เช่น รูปลูกศร (ARROW), รูปนาฬิกาทราย (Hour glass) ใน application นี้จะใช้รูปลูกศร ซึ่งใช้กันทั่วไป โดยเรียก

```
LoadCursor(NULL,IDC_ARROW)
```

hBrBackground เป็น handle ของแปรง (Brush) ที่จะให้ GDI ใช้ทาลงไปบนพื้น windowในที่นี้เราจะใช้ แปรงของ GDI ไปก่อนและเป็นแปรงสีขาวโดยเรียก

```
GetStockObject(WHITE_BRUSH)
```

lpszMenuName เป็น long pointer to string ซึ่งเป็นชื่อของเมนูที่ปิดท้ายด้วยไบต์ศูนย์ ('\0') ต้องเป็นชื่อ ที่อยู่ในส่วน resource file

lpszClassName เป็น long pointer to string อันเป็นชื่อของ class ที่จะลงทะเบียนนี้

หลังจากเซตค่าทุกตัวเรียบร้อยแล้วจึงเรียก RegisterClass() ถ้าสำเร็จจะคืนค่าที่ไม่ใช่ศูนย์ แต่ถ้าไม่สำเร็จจะคืนค่าศูนย์กลับไป

Creating Window

หลังจากที่ได้ class เรียบร้อยแล้วเราก็พร้อมที่จะสร้างหน้าต่างได้แล้วรูปแบบ การ สร้าง หน้าต่าง จะเป็นดังนี้

```
HWND CreateWindow(lpClassName,  /* class ที่ได้ลงทะเบียนแล้ว */
        lpWindowName, /* ชื่อของ title bar หรือ caption bar */
        dwStyle,      /* เป็น window style */
        x,            /* x corodinate นับจากมุมซ้ายบน */
        y,            /* y corodinate นับจากมุมซ้ายบน */
        nWidth,       /* ความกว้างของหน้าต่าง */
        nWndParent,   /* ความสูงของ window */
        nMenu,        /* handle of menu */
        nInstance,    /* window instance */
        lpParam);     /* parameter ส่งพร้อมกับ WM_CREAT */
```

ตัวอย่าง CreateWindow

```
hWndMain = CreateWindow(
    szApplName,          /* See RegisterClass() call */
    szTitleName,         /* Text for window title bar */
    WS_OVERLAPPEDWINDOW,     /* Window style */
    CW_USEDEFAULT,           /* default horizontal position */
    CW_USEDEFAULT,           /* default vertical position */
    CW_USEDEFAULT,           /* default width */
    CW_USEDEFAULT,           /* default higth */
    NULL,                /* Overlapped windows have no parent */
    NULL,                /* Use  the window class menu */
    hInstance,           /* This hinstance owns this window */
    NULL);               /* Pointer not needed */
```

ให้สังเกต field dwStyle ซึ่งได้ดชต์ให้เป็น WS_OVERLAPPEDWINDOW เป็น style ที่ใช้กันทั่ว ๆ ไป ประกอบด้วย Minimize,Maximize box,Caption bar และSystem menu หลังจากเรียก CreateWindow() เสร็จก็จะต้องเรียก

```
/* Make the window visible: update its client area; and return "SUCCESS" */
ShowWindow(hWndMain,nCmdShow);
/* Show thw window */
UpdateWindow(hWndMain);
/* Sends  WM_PAINT message */
```

เพื่อให้ windows แสดงหน้าจอโดย hWndMain เป็นค่า handle window ที่ได้รับเมื่อตอนที่สร้างหน้าต่าง และ nCmdShow เป็น parameter ตัวที่ 4 จาก function WinMain() ซึ่ง windows เรียกเมื่อตอนที่เริ่ม run application โดยค่านี้จะเป็นตัวบอกให้ การแสดงหน้าต่าง นี้เป็นหน้าจอ window เต็มรูปแบบเลย (กรณีที่เราเรียก application จากการใช้ mouse กด double click หรือกด <Enter> เป็นต้น) หรืออีกแบบ จะแสดงเป็น icon

ส่วนการเรียก UpdateWindow() ก็เพื่อให้ windows ส่งคำสั่ง WM_PAINT เพื่อให้ตัว application ทำการวาดส่วนที่ต้องการแสดงใน Client area

โดยเมื่อ windows application ได้รับ message WM_PAINT จาก windows จะเรียก function PaintFunc(hWnd) ซึ่งมีการทำงานดังนี้

```
hDC = BeginPaint(hWnd,&ps);
/* Get display context of main Window */
```

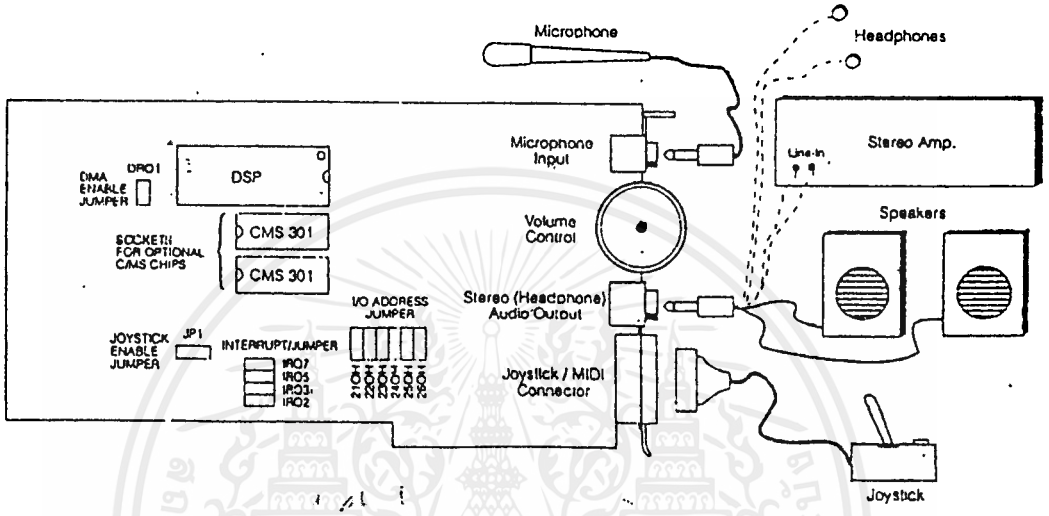
ค่า ps มีชนิดข้อมูลเป็น PAINT STRUCT ซึ่งจะเก็บส่วนสำคัญที่จำเป็นต่อการวาดนี้ เช่น ส่วนใน Client area ที่จำเป็นต้องได้รับการวาดใหม่

```
EndPaint(hWnd,&ps);
```

ซึ่งจะทำให้ windows รับรู้ว่าส่วน Client area ที่ยังไม่ได้รับก่อน update นั้นได้ update เป็นที่เรียบร้อยแล้ว windows ก็จะปลดปล่อย (free) ส่วน display context นี้ต่อไป

บทที่ 3 Sound Blaster Card

ในโครงงานนี้ได้เลือกใช้ Sound Card ที่ชื่อว่า Sound Blaster version 2.0 ซึ่งมีส่วนประกอบดังรูป



General Specifications

Sound Capabilities

- มีระบบเสียง 12 ระบบ (12 C/MS)
- มี 11 เสียง สำหรับระบบ FM
- มีตัวแปลงสัญญาณจาก ดิจิตอลเป็น อะนาล็อก 1 ช่อง มีรายละเอียดดังนี้
  - * มีอัตราการ Sampling 4 KHz to 23 KHz
  - * Compressions schemes
    - 8 bit data no compression
    - 2 to 1 data compression 4 bit
    - 3 to 1 data compression 2.6 bit
    - 4 to 1 data compression 2 bit
- มีระบบ stereo power amplifier มีอัตรากำลัง 4 watt ต่อ 1 channel ระบบ output เป็น 4 Ohm

- RCA plug
- มีปุ่ม volume control

Voice Input (Digital Sampling) Capability

- 8 bit A/D conversion
- อัตราการ Sampling อยู่ระหว่าง 4 KHz to 44.5 KHz
- ADC Transfer mode
    * Direct mode
    * DMA mode no CPU
- มีไมโครโฟน และ ระบบ amplifier พร้อมในตัว ซึ่งมีอัตราขยายแบบ auto
- มี Software สำหรับบันทึกเสียงด้วย

Joystick Port

- มี standard "Game I/O port" และมี standard IBM compatible joystick

MIDI Interface

- มี MIDI Interface ในตัว สำหรับเชื่อมต่อกับ เครื่อง MIDI หรือ Keyboards

Hardware Data

ใน Sound Blaster จะใช้ I/O port เบอร์ระหว่าง 2x0H - 2xFH ซึ่ง x คือเบอร์ของ Jumper ระหว่าง 1 - 6

I/O address คือ 210H,220H,230H,240H,250H,260H

FM music จะใช้ address ระหว่าง 388H - 389H

การใช้งาน Sound Blaster Card

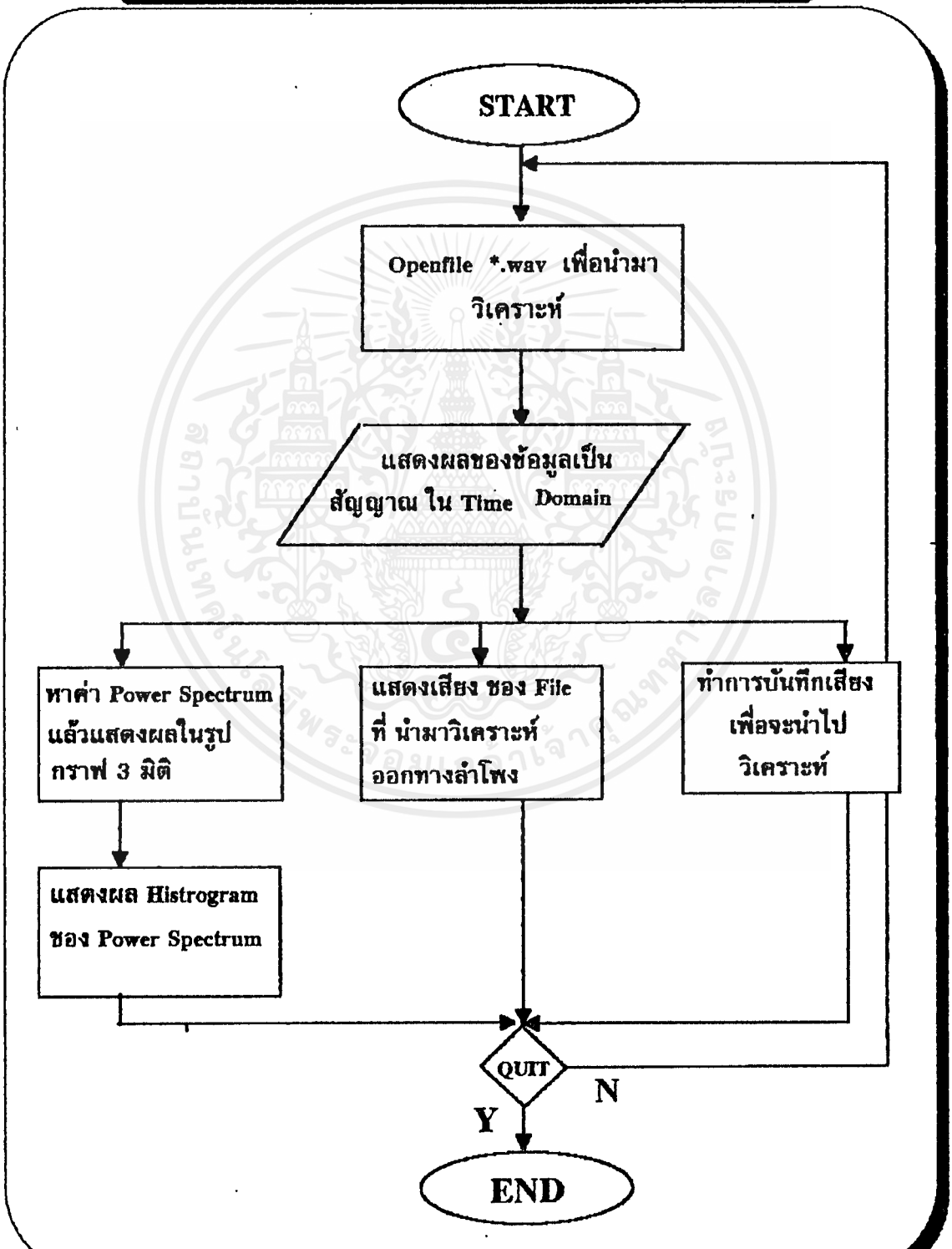
1. นำ Sound Blaster Card เสียบลงใน slot ของเครื่อง computer
2. ต่อลำโพงเข้ากับ Sound Blaster
3. ปรับ Volume Control ของ Sound Blaster ให้อยู่กึ่งกลาง
4. ใช้ File Test-SBC ที่มาพร้อมกับ Card ทดสอบ Sound Blaster โดยจะเช็ค I/O Address , IRQ และก็ Feature ต่าง ๆ ของ Sound Blaster
5. ตั้ง Config ของ Sound Blaster ลงใน file Autoexec.bat ของ DOS

    SET BLASTER = A220 I7 D1 T3

    เพื่อ Set ให้ Sound Blaster ทำงานได้ถูกต้องกับเครื่อง Computer
6. จากขั้นตอนที่ผ่านมา ต่อจากนี้เราสามารถใช้งาน Sound Blaster ได้แล้ว

บทที่ 4 โปรแกรมวิเคราะห์เสียง

ALGORITHM ของ PROGRAM



ทฤษฎี ของ อนุกรมฟูริเยอร์ (Fourier series)

ถ้าให้ x(t) เป็นสัญญาณต่อเนื่องใน โดเมนเวลา (Time Domain) ถ้า x(t) มีคุณสมบัติเป็นฟังก์ชันคาบ (period function) ที่มี ค่าคาบเป็น T_0 กล่าวคือ $x(t+T_0) = x(t)$ ต่อค่าทุกค่าของ t เราสามารถกระจายหรือเขียนแทนฟังก์ชันเป็นคาบนี้ได้ด้วย อนุกรมตรีโกณมิติ คือ

$$x(\tau) = \sum_{n=-\infty}^{\infty} C_n \exp(j2\pi t / T_0) \qquad \text{.....(1)}$$

$$= C_0 + \sum_{n=1}^{\infty} \{(C_n \exp(j2n\pi t/ T_0) + C_{-n} \exp(-j2n\pi t/T_0)\}$$

โดยที่ n เป็นจำนวนเต็ม และ สัมประสิทธิ์ หาได้จากสมการที่ 1 โดยการอินทิเกตพจน์ต่อพจน์ ดังนี้

$$C_n = (1/T_0)\int_0^{T_0} X(t) \exp\{-j(2n\pi t/T_0)\}dt$$

อนุกรม (1) นี้เรียกว่าเป็น อนุกรมฟูริเยอร์ ของฟังก์ชัน x(t)

สัญญาณในโดเมนเวลา x(t) สามารถแปลงหรือกระจายให้อยู่ในรูปของสัญญาณในโดเมนความถี่ (Frequency Domain)โดยใช้ หลักการของ Discreate furier transtorm (DFT) ซึ่ง DFT นี้จะมีสมการดังนี้

$$X(k) = (1/n)\sum_{n=0}^{N-1} X(n)W_n^{n_k} \quad ;k=0,1,...,N-1 \text{ ----------(2)}$$

โดยที่ X(k) คือค่าปริมาณเสียงในช่วงความถี่

N คือค่าตัวอย่างของเสียงที่ใช้ในการคำนวณ

$W_n^{n_k} = e^{-j2\pi n_k/N}$ คือค่าเชิงซ้อนมีสมการเป็นรูปคาบเวลา

แต่เนื่องจากการพิจารณา โดยใช้ DFT จะเป็นการพิจารณาถึงสัญญาณ ที่มีลักษณะเป็นคาบเวลา และมีค่าคงที่ต่อเนื่องกันไป จนถึงอนันต์ ซึ่งตามความเป็นจริงแล้ว สัญญาณที่อ่านเข้ามามิได้เป็นเช่นนั้น จึงต้องมีการคำนวณ ด้วยสมการที่เหมาะสม เพื่อกำหนดให้ ความสำคัญ กับสัญญาณ

ในช่วงเวลาปัจจุบัน และใกล้เคียง สมการดังกล่าวเรียกว่า Weighting Functions ซึ่งมีอยู่หลายแบบ เช่น Hanning , Parzen , Welch

ALGORITHM ของ HISTROGRAM

1. รับข้อมูลที่ได้จากการคำนวณ จาก routine Spectrum
2. ทำการอ่านข้อมูลเข้ามาทีละตัว อ่านค่าที่ได้ซึ่งมีค่าตั้งแต่ -100 ถึง 0 นับจำนวนครั้งที่อ่านได้ เก็บจำนวนนั้นไว้ใน Array อีกตัวหนึ่ง
3. นำค่าที่ได้ไปแสดงผลบนมอนิเตอร์

ALGORITHM ของ POWER SPECTRUM

1. รับค่าข้อมูลที่จะทำการวิเคราะห์ เข้ามาทีละ 1024 ข้อมูล
2. กำหนด Usefull Factor เพื่อใช้ใน routine ดังต่อไปนี้

   mm = จำนวน O/P *2;

   m43 = (mm*2)+3;

   m44 = (m43+1);

   facm = (m-0.5); m คือ จำนวน O/P

   facp = 1.0/(m+0.5);
3. หาค่าเฉลี่ยของ Amplitude ทั้งหมด แล้วยกกำลังสอง เก็บค่าไว้
4. Initial Spectrum = 0
5. นำค่าข้อมูลเก็บใน ARRAY ของ Spectrum โดยข้อมูลมีค่าอยู่ระหว่าง -1 ถึง 1
6. นำข้อมูลคูณกับ factor ที่ได้จากการคำนวณค่าเฉลี่ยของ Amplitude ในแต่ละเฟรม
7. ส่งข้อมูลเข้า FFT Routine
8. ข้อมุลที่ได้จาก FFT Routine นำมาทำการ Normalization ด้วยการหารด้วย (ค่าเฉลี่ยของ Amplitudeยกกำลังสอง * จำนวนข้อมูลที่ใช้ในขบวนการ FFT)
9. นำข้อมูลที่ได้ออกแสดงผลทางจอภาพ

10. ข้อมูลที่จะทำการวิเคราะห์หมดหรือยัง ถ้ายังไม่หมด ไป ทำข้อ 1. ใหม่ หมดแล้วทำข้อ 11
11. END

ALGORITHM ของ FFT

1. รับข้อมูลจาก Spectrum Routine ทีละ 1024 ข้อมูล
2. ทำการเปลี่ยนข้อมูลให้เป็นแบบ Complex Number โดยการสลับตำแหน่งของข้อมูล
3. ข้อมูลที่ได้นำไปคำนวนใน Routine Danieson - Lanczos
4. Return ข้อมูลที่ได้ กลับไปยัง Spectrum Routine

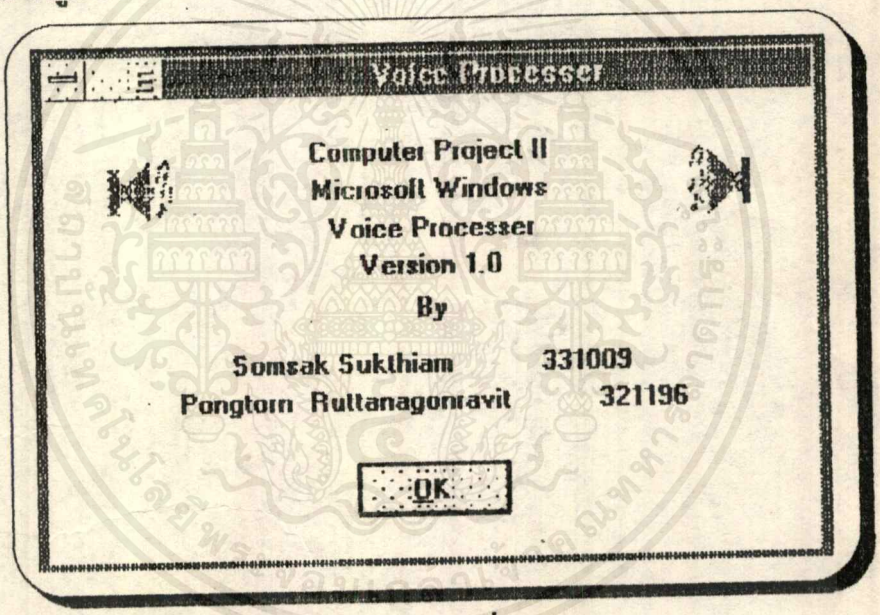
ขั้นตอนการทำงานของโปรแกรม

โปรแกรมนี้ได้พัฒนาขึ้นบนระบบวินโดว์ โดยใช้ภาษา C เป็นภาษาที่ใช้ในการเขียน และใช้ BorlandC Version 3.1 เป็นตัว Compiler และ ใช้ Sound Blaster cardเป็นอุปกรณ์ร่วมเพื่อทำหน้าที่เกี่ยวกับเสียง

ผลการทำงานของ Program มีดังนี้

1. เมื่อเริ่มการทำงาน Program ทำการสร้าง Dialog Box ขึ้นมา ซึ่งแนะนำ ตัวผู้พัฒนา Program

มีลักษณะดังรูปที่ 8

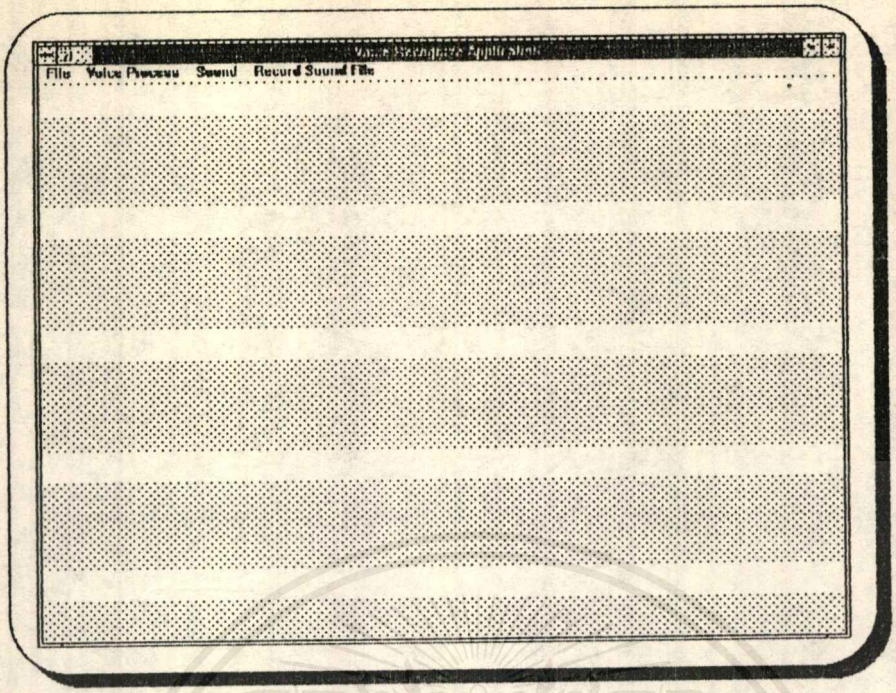


รูปที่ 8

2. เมื่อผู้ใช้กดปุ่ม OK แล้ว Dialog Box จะหายไป แต่จะปรากฏ Window หลักของ Programขึ้นมามีลักษณะดังรูปที่ 9

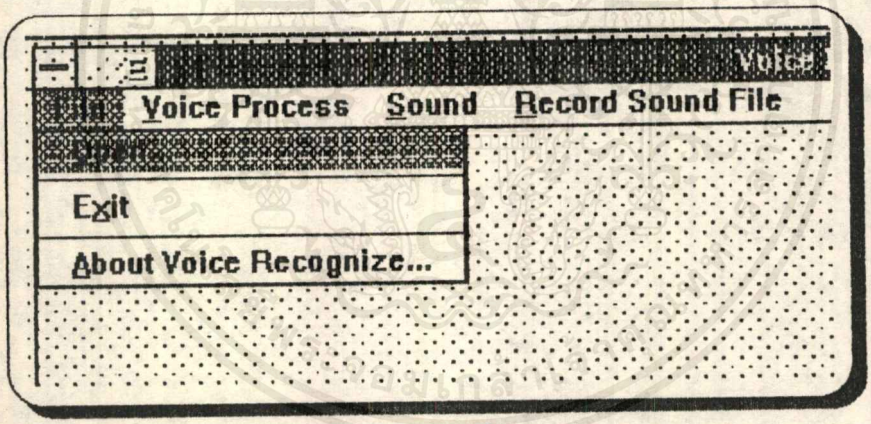
บน Window หลัก จะมี Menu ให้เลือกดังต่อไปนี้

1. File เปิดไฟล์ขึ้นมาเพื่อนำข้อมูลไปวิเคราะห์
2. Voice Process ทำการวิเคราะห์ข้อมูล
3. Sound แสดงเสียงออกทางลำโพง
4. Record Sound File บันทึกเสียงเพื่อนำไปวิเคราะห์



รูปที่ 9

3. เมื่อผู้ใช้เลือก Menu File จะปรากฏ Popup Menu ดังรูปที่ 10

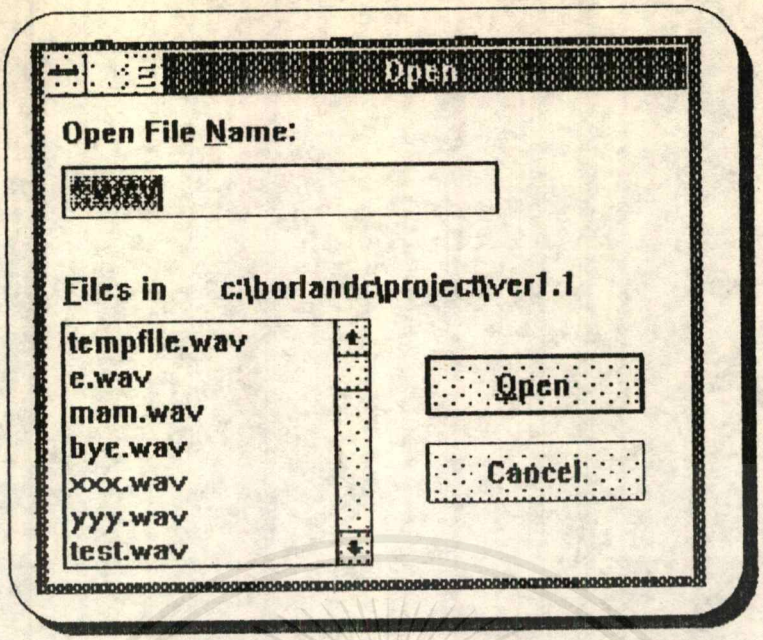


รูปที่ 10

4. เมื่อผู้ใช้ เลือกคำสั่ง ต่อไปนี้

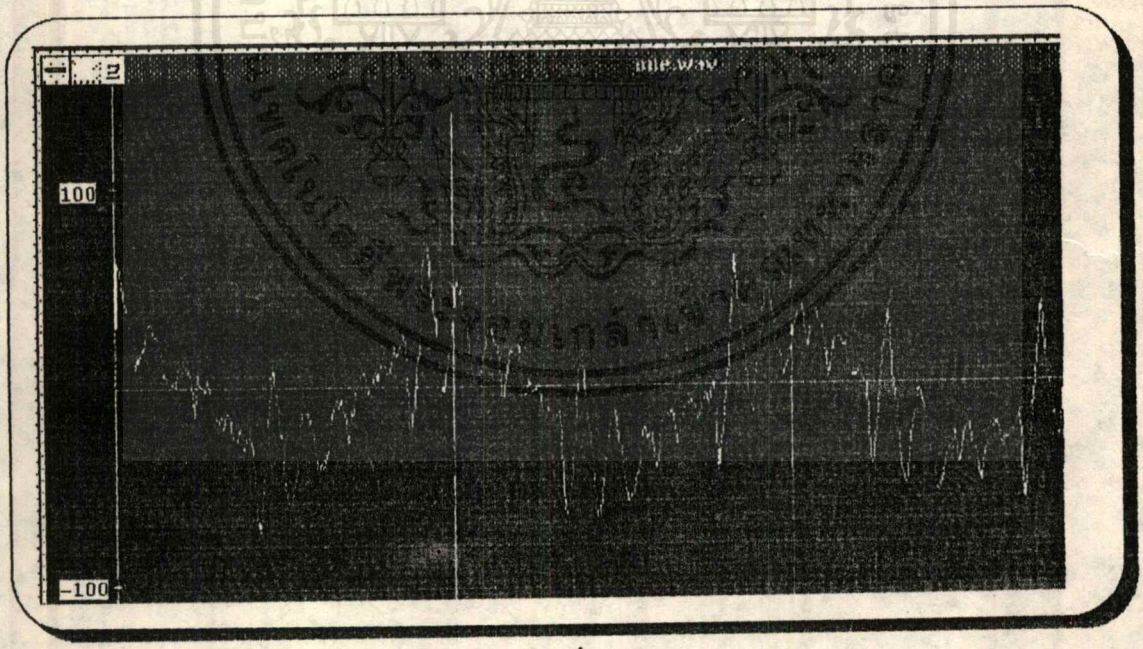
- Open จะมี Dialog Box รายชื่อของแฟ้มข้อมูลที่จะเปิดเพื่อวิเคราะห์
- Exit เลิกโปรแกรม
- About Voice Recognize.. แสดง Dialog Box เริ่มต้นขึ้นมา

Dialog ของ Open มีลักษณะดังรูปที่ 11



รูปที่ 11

5. เมื่อผู้ใช้เปิดแฟ้มข้อมูลแล้ว Program จะแสดงผลในแกน Time Domain ดังแสดงใน รูปที่ 12

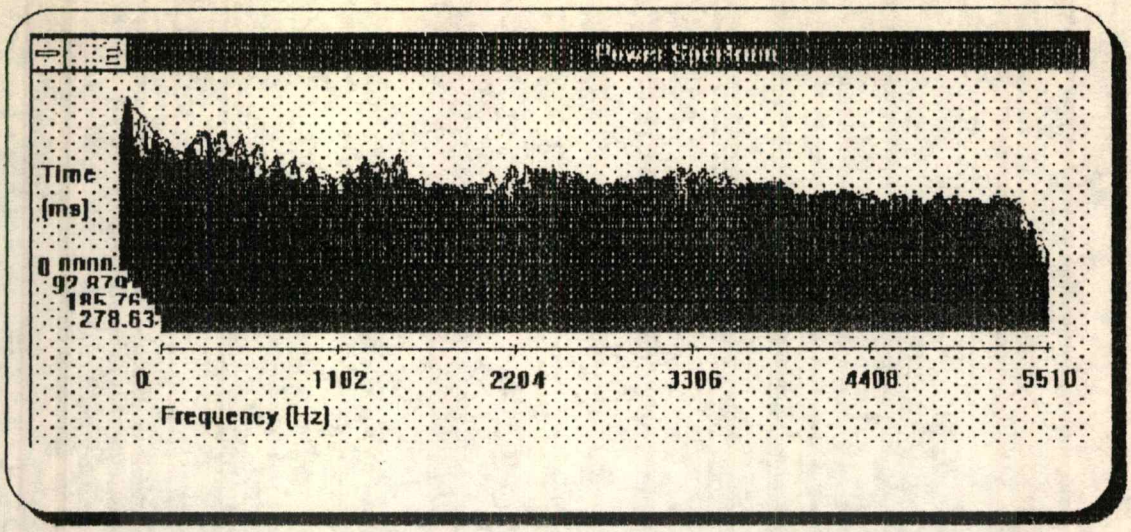


รูปที่ 12

6.เมื่อเลือกแฟ้มข้อมูล เพื่อนำมาวิเคราะห์แล้ว เมื่อผู้ใช้เลือก Menu Voice Process จะขึ้น Popup Menu ดังต่อไปนี้

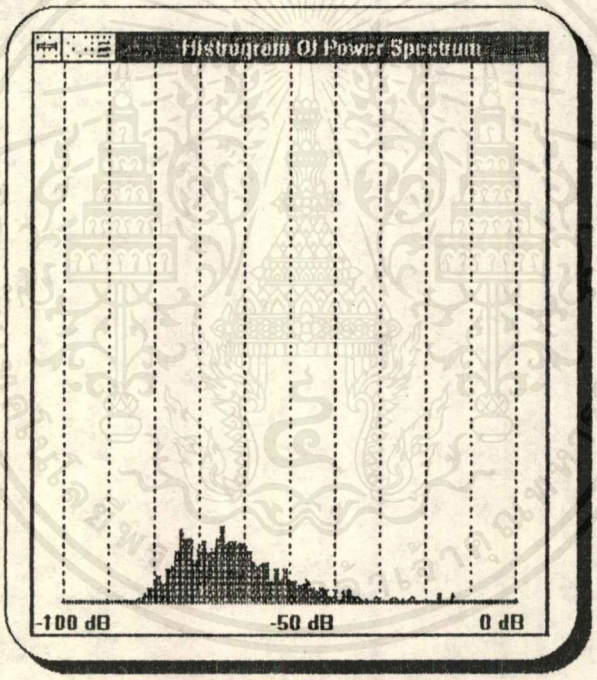
- Spectrum ใช้หาค่า Power Spectrum ของเสียง
- Histrogram แสดง Histrogram ของ Power Spectrum

เมื่อเลือกคำสั่ง Spectrum จะแสดงดังในรูปที่ 13



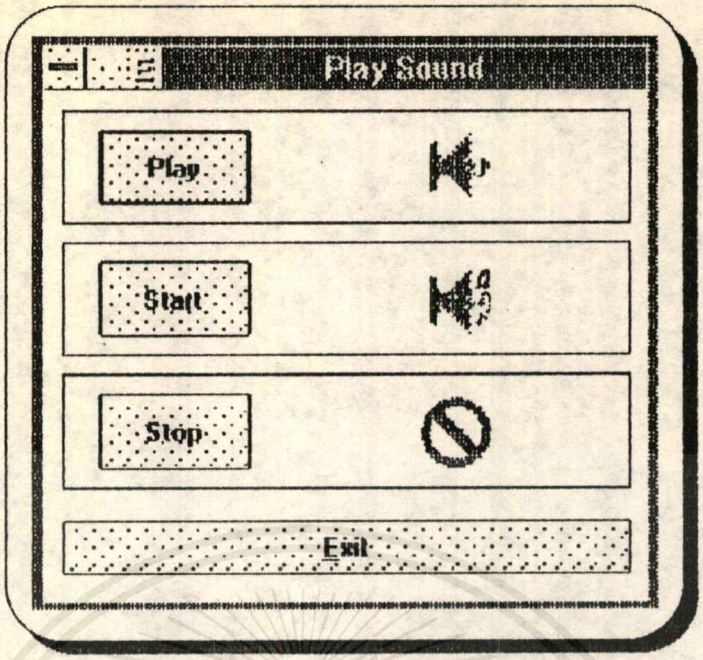
รูปที่ 13

เมื่อหาค่า Power Spectrum แล้ว เมื่อเลือกคำสั่ง Histrogram จะได้ผลดังรูปที่ 14



รูปที่ 14

8. จาก Menu บน Window หลัก เมื่อผู้ใช้เลือกคำสั่ง Sound จะปรากฏ Dialog Box ขึ้นมาดังรูปที่ 15

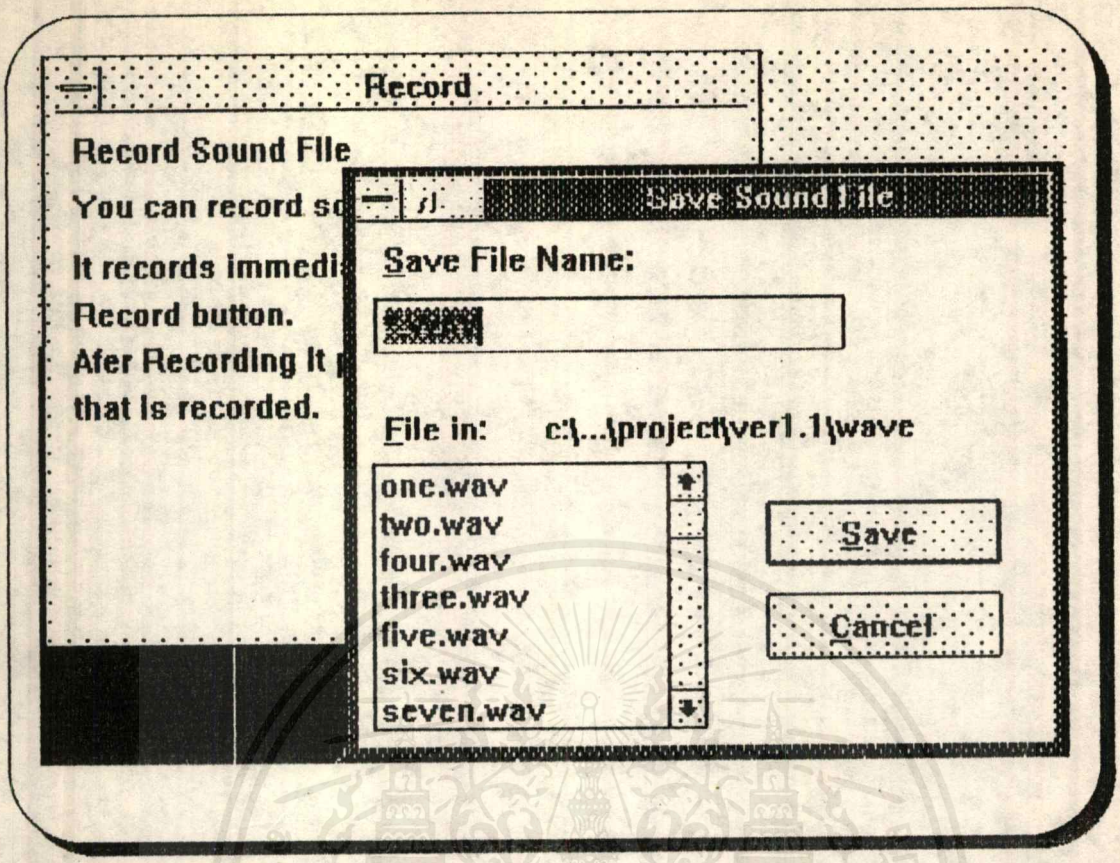


รูปที่ 15

9. จาก Menu บน Window หลัก เมื่อผู้ใช้เลือกคำสั่ง Record Sound File จะปรากฏ Dialog Box ขึ้นมาดังรูปที่ 16

รูปที่ 16

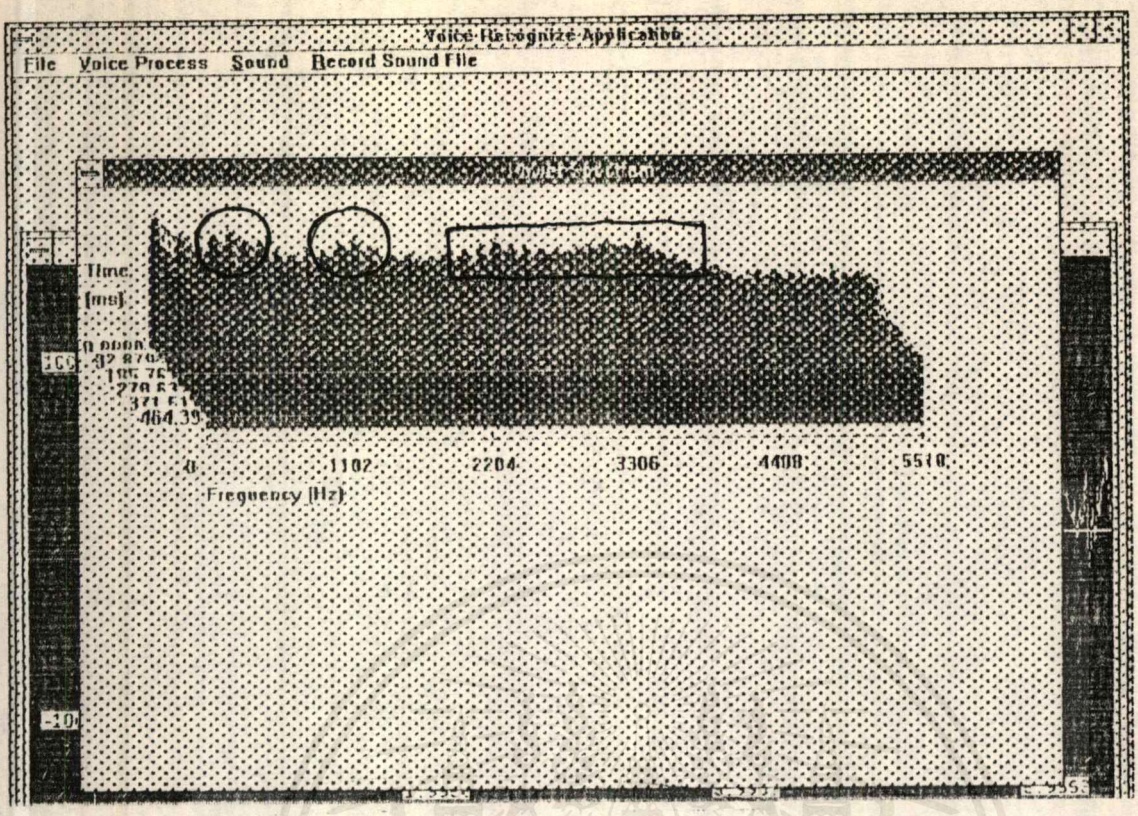
10. จากการ Record เมื่อผู้ใช้ต้องการจะเก็บข้อมูลเสียงที่ทำการ Record ไว้ในชื่อใดชื่อหนึ่ง Program จะแสดง Dialog Box ของการ Save ข้อมูล ดังรูปที่ 17



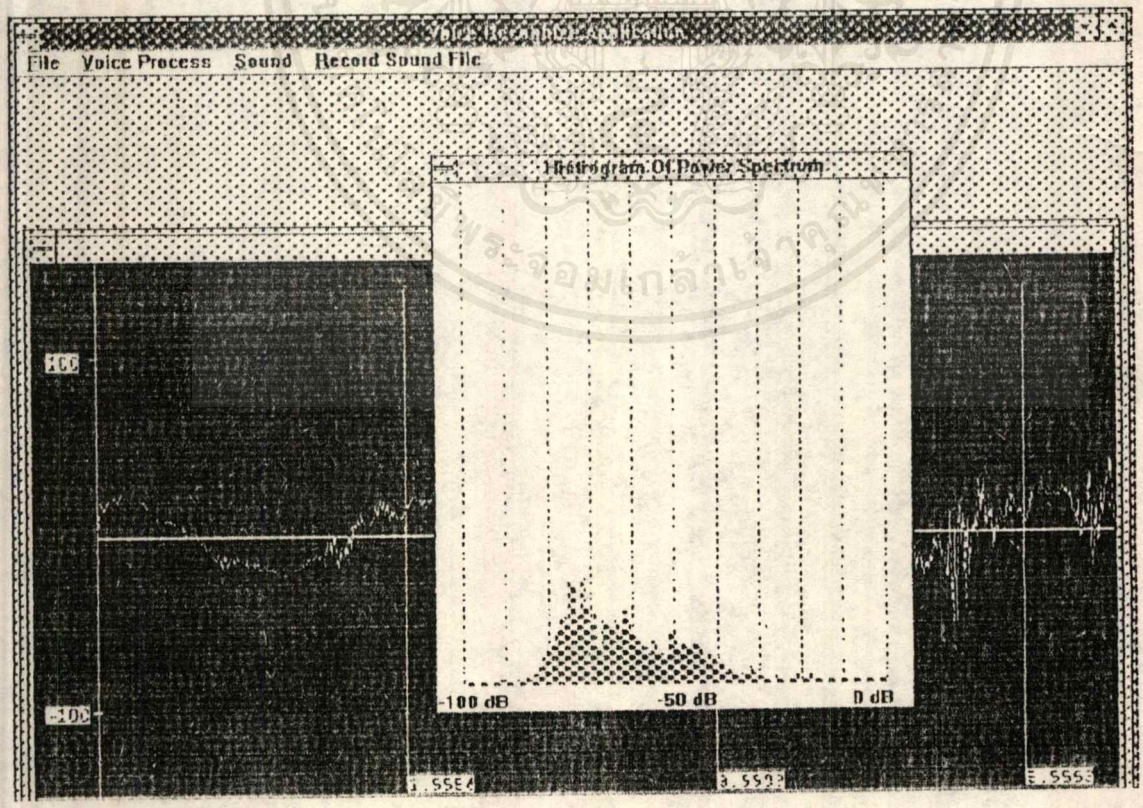
รูปที่ 17

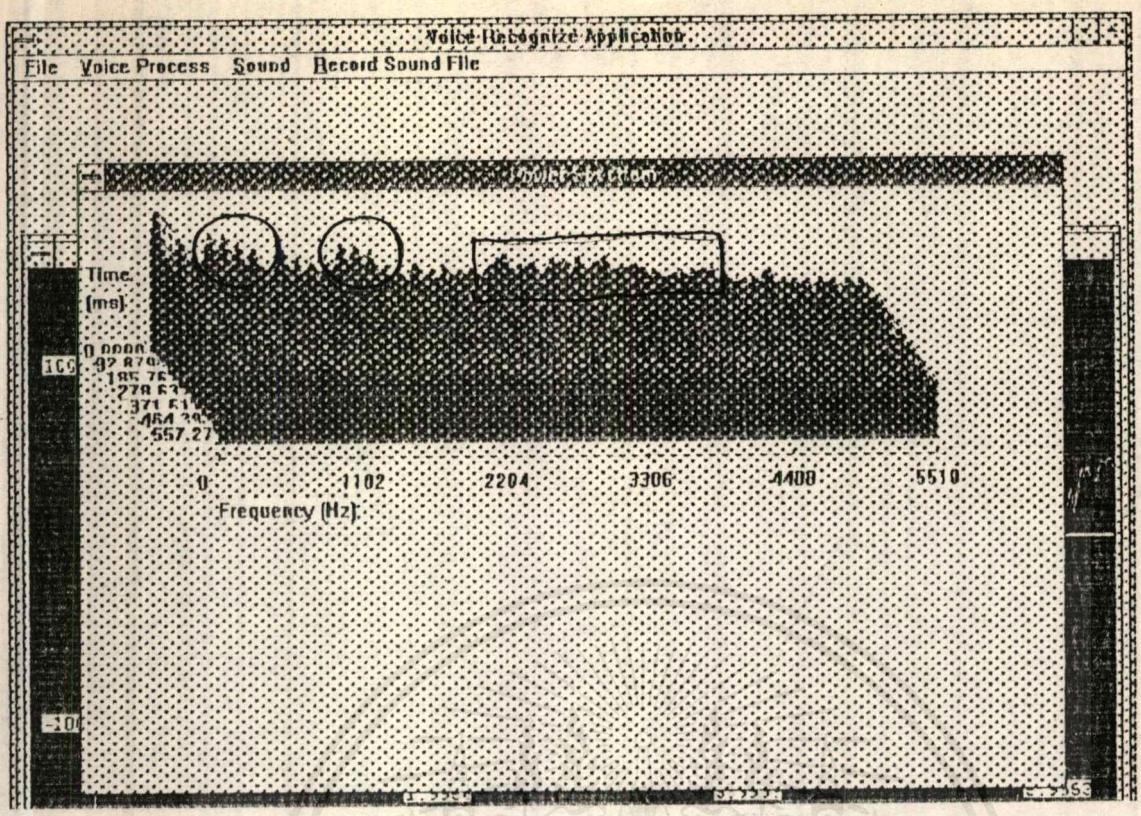
บทที่ 5 ผลการทดลองและสรุป

เสียง	ลักษณะของเสียง
หนึ่ง	จะมีความถี่เด่นชัดที่ 500 Hz - 1102 Hz,2 KHz - 2.5 KHz และจะมีความถี่เด่นเป็นช่วงยาว 2.7 KHz - 4 KHz
สอง	มีความถี่เด่นชัดที่ 500 Hz-1.8 KHz,2.5 KHz - 3 KHz,3.3 KHz - 4KHz
สาม	ความถี่เด่นชัดที่ 1 KHz ซึ่งมีช่วงเริ่มประมาณ 500 Hz-2 KHz และ 1.5 Hz และอีกความถี่หนึ่งเป็นช่วง 2.5 KHz - 3 KHz
สี่	ความถี่เด่นมากที่ 500 Hz และมีความถี่เด่นชัดรองลงมาที่ 2.2 KHz, 3 KHz และ 4 KHz
ห้า	ความถี่จะเด่นชัดเป็นช่วงตั้งแต่ 500 Hz - 2.2 KHz และมีความถี่เด่นอีกคือ 3 KHz , 3.7 KHz
หก	ความถี่จะเด่นชัดเป็นช่วงซึ่งเด่นมากในช่วง 500 Hz - 1.1 KHz และมีความถี่เด่นอีกที่ 3.3 KHz และ 3.5 KHz
เจ็ด	จะมีความถี่เด่นที่ 1 KHzและ1.1 KHz ซึ่งอยู่บนช่วงความถี่ 500 Hz-1.3KHz และมีความถี่เด่นที่ 2.2 KHz และมีความถี่ช่วงยาวตั้งแต่ 2.5 KHz-4.4 KHz
แปด	มีความถี่เด่น ๆ ที่ 1.3 KHz, 2.2 KHz และ 3 KHz
เก้า	มีช่วงความถี่เด่นตั้งแต่ 500 Hz-2 KHz ซึ่งมีความถี่เด่นที่ 1.1 KHz และ 1.8 KHz และมีความถี่เด่นอีกที่ 2.5 KHz และมีช่วงความถี่เด่นอีกตั้งแต่ 3.3 KHz - 5 KHz
สิบ	ความถี่เด่นชัดมากที่ 500 Hz และ 2.2 KHz และมีช่วงความถี่ช่วงยาวตั้งแต่ 2.3 KHz - 5.5 KHz

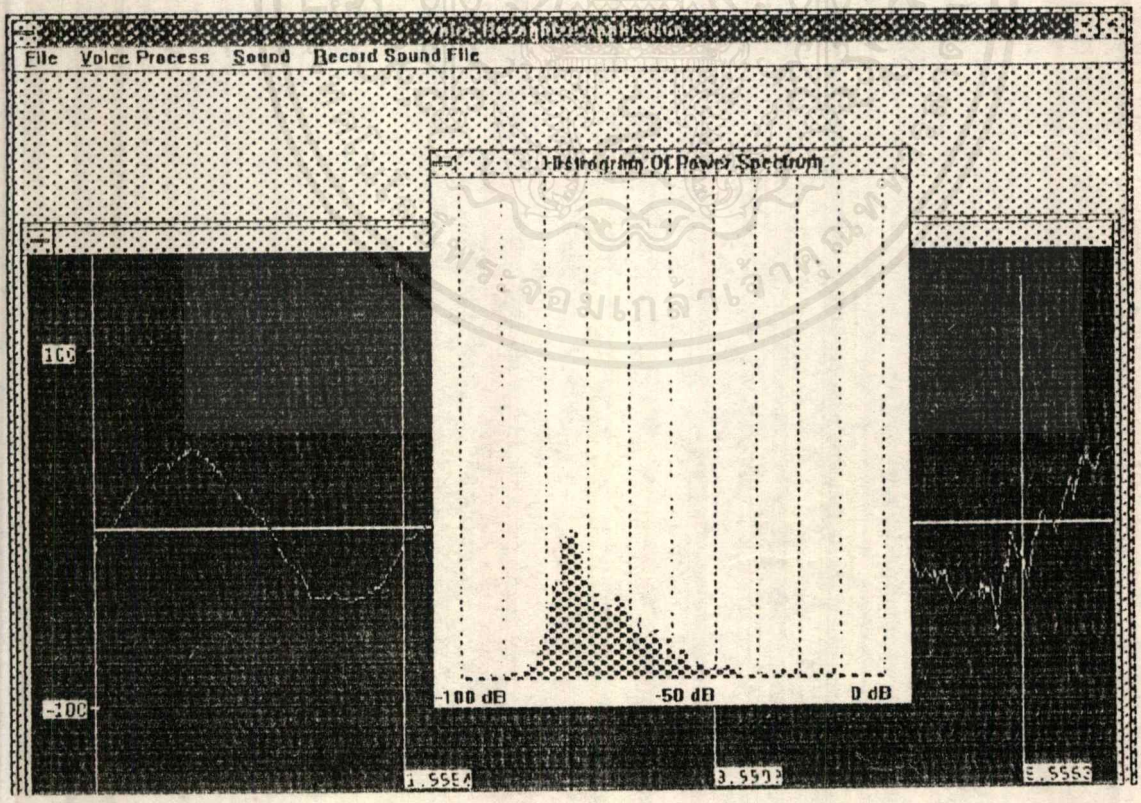


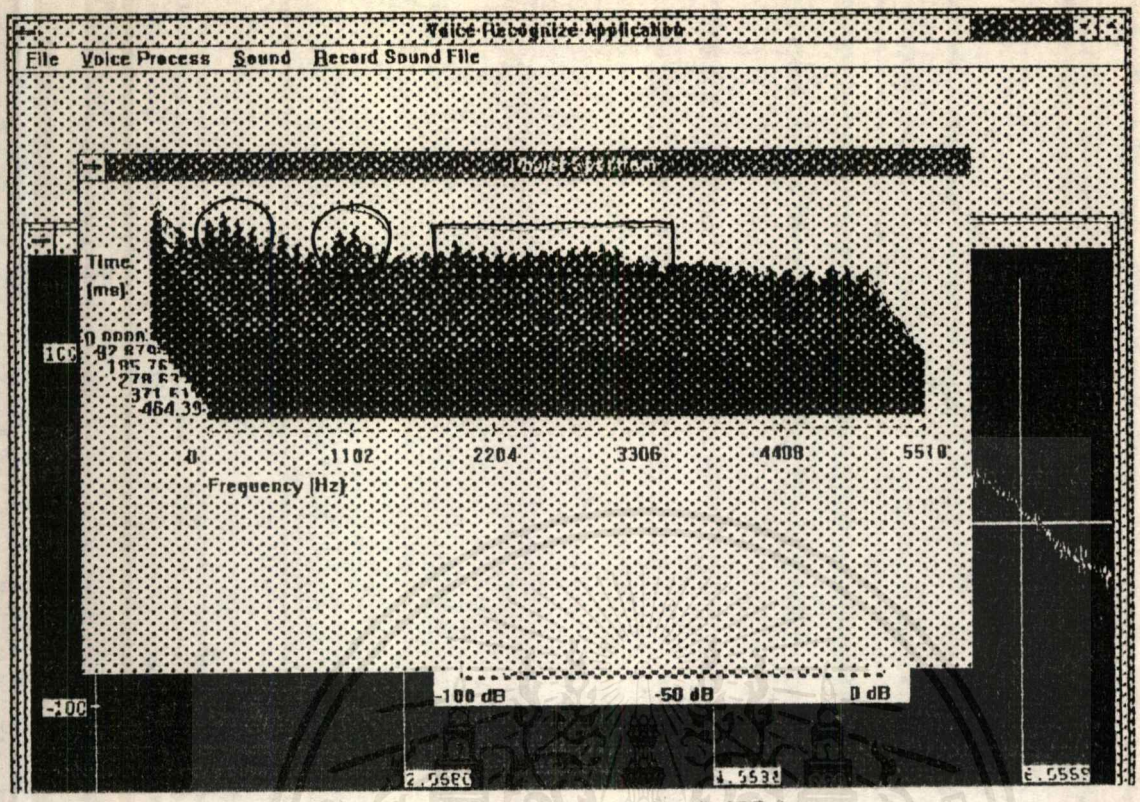
คำว่า "หนึ่ง" พูดโดยบุคคลที่ 1 ครั้งที่ 1



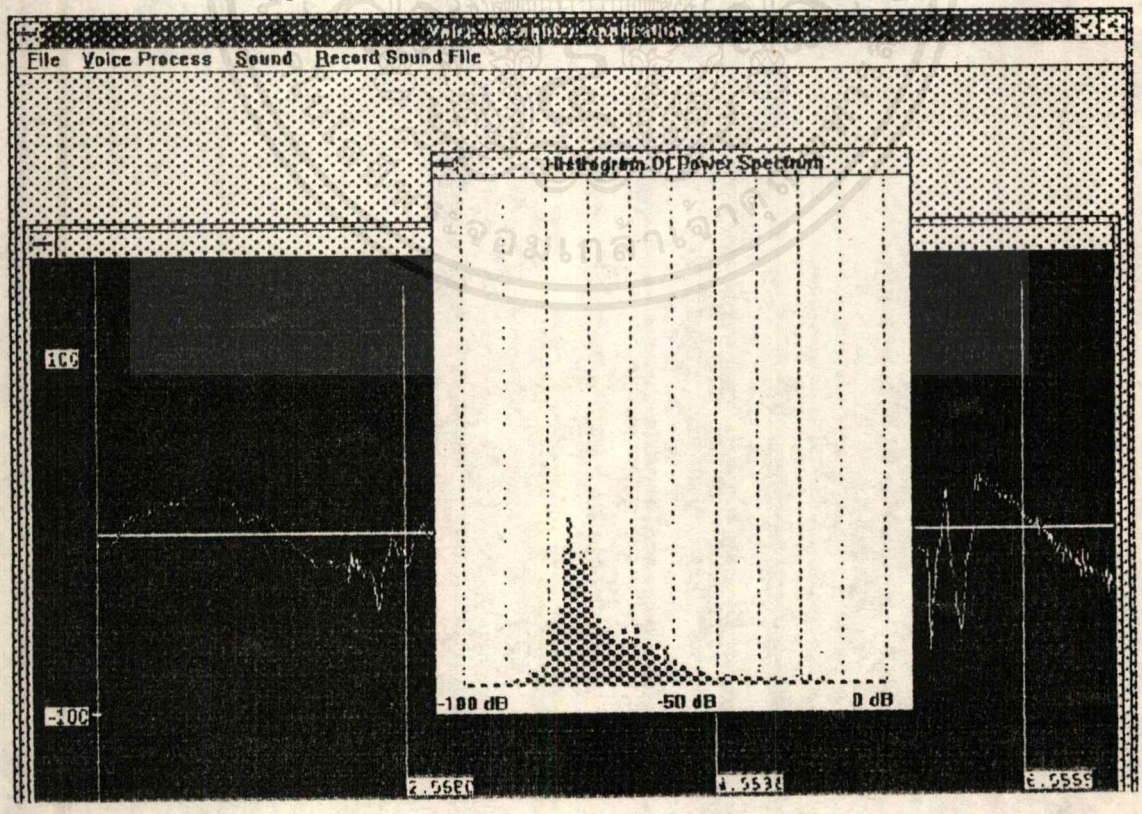


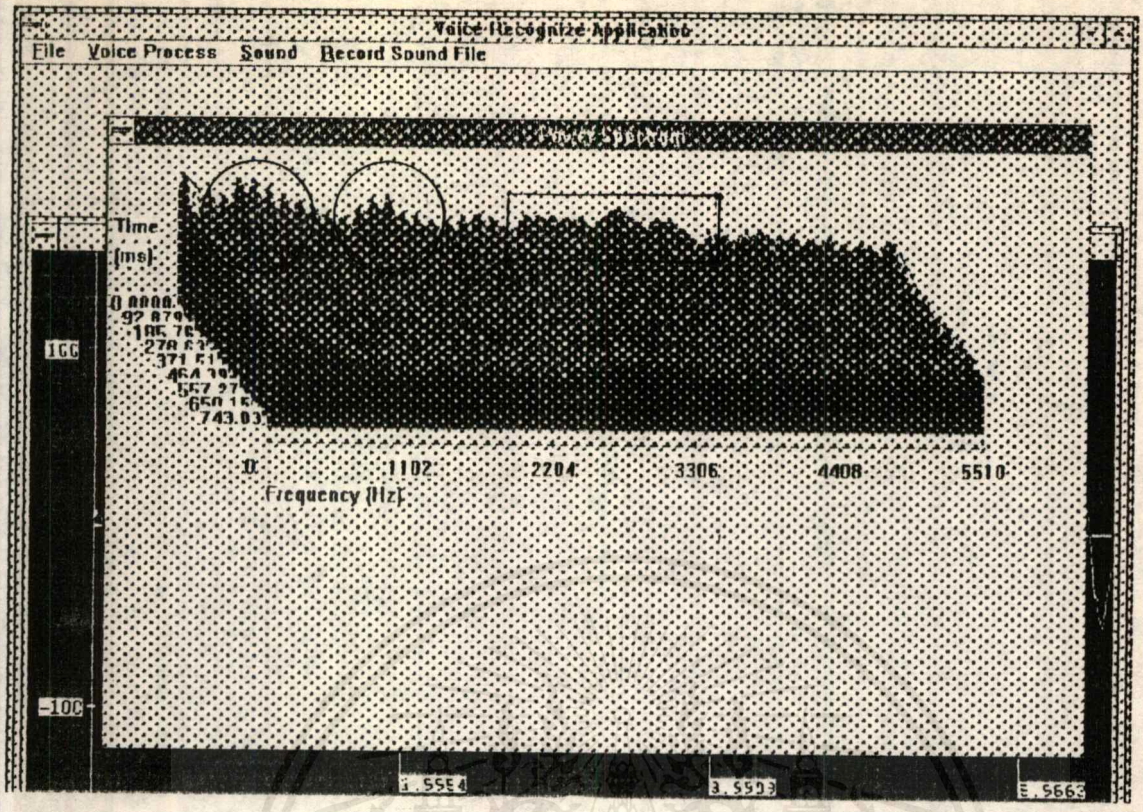
คำว่า "หนึ่ง" พูดโดยบุคคลที่ 1 ครั้งที่ 2



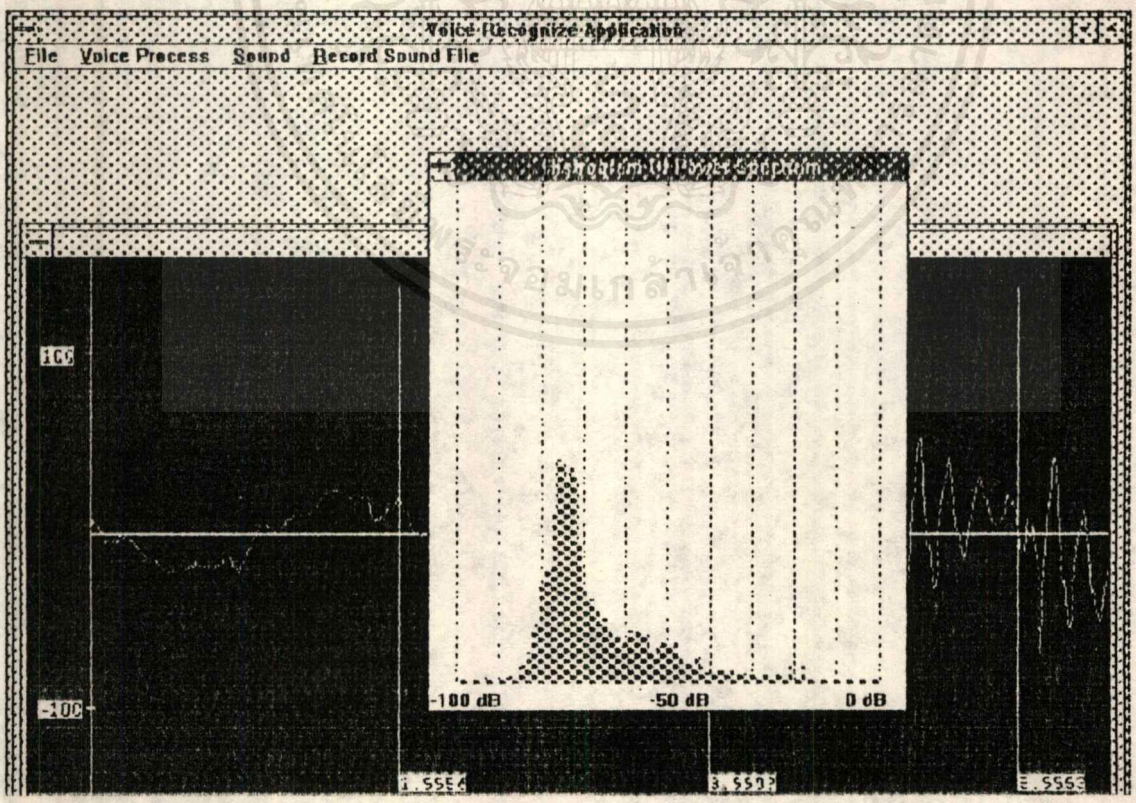


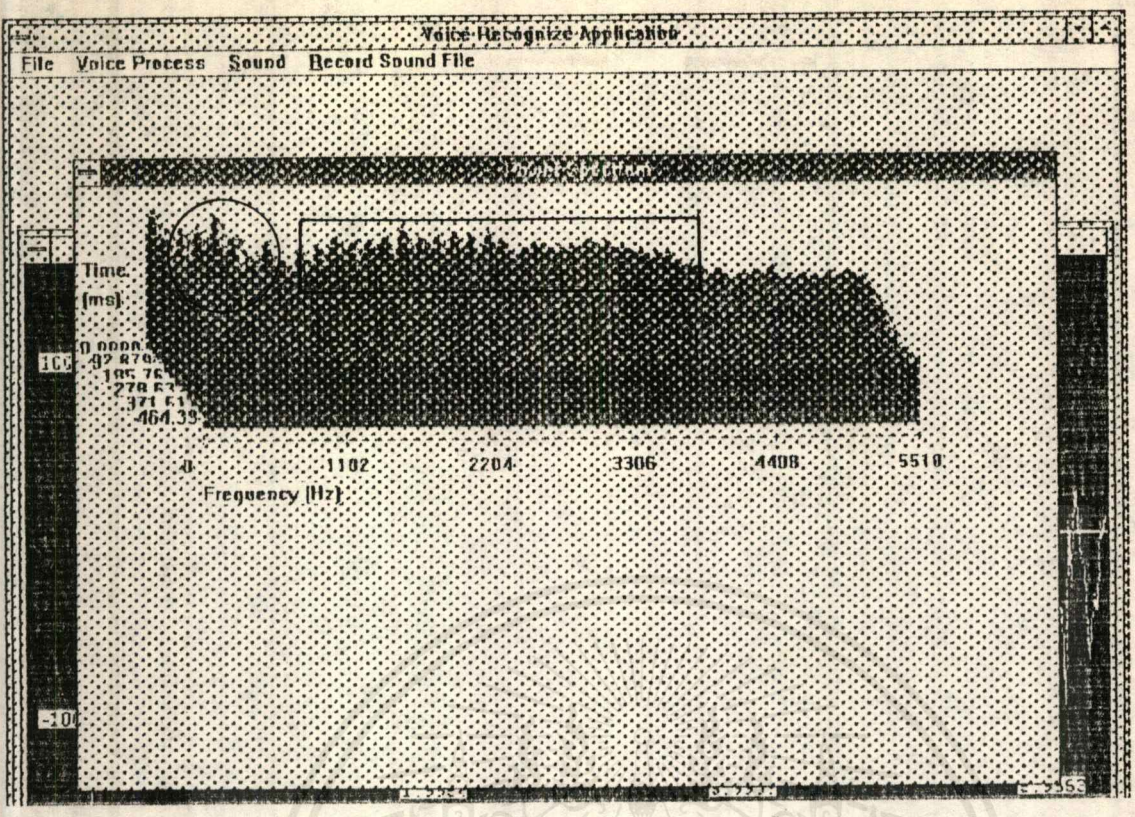
คำว่า "หนึ่ง" พูดโดยบุคคลที่ 1 ครั้งที่ 3



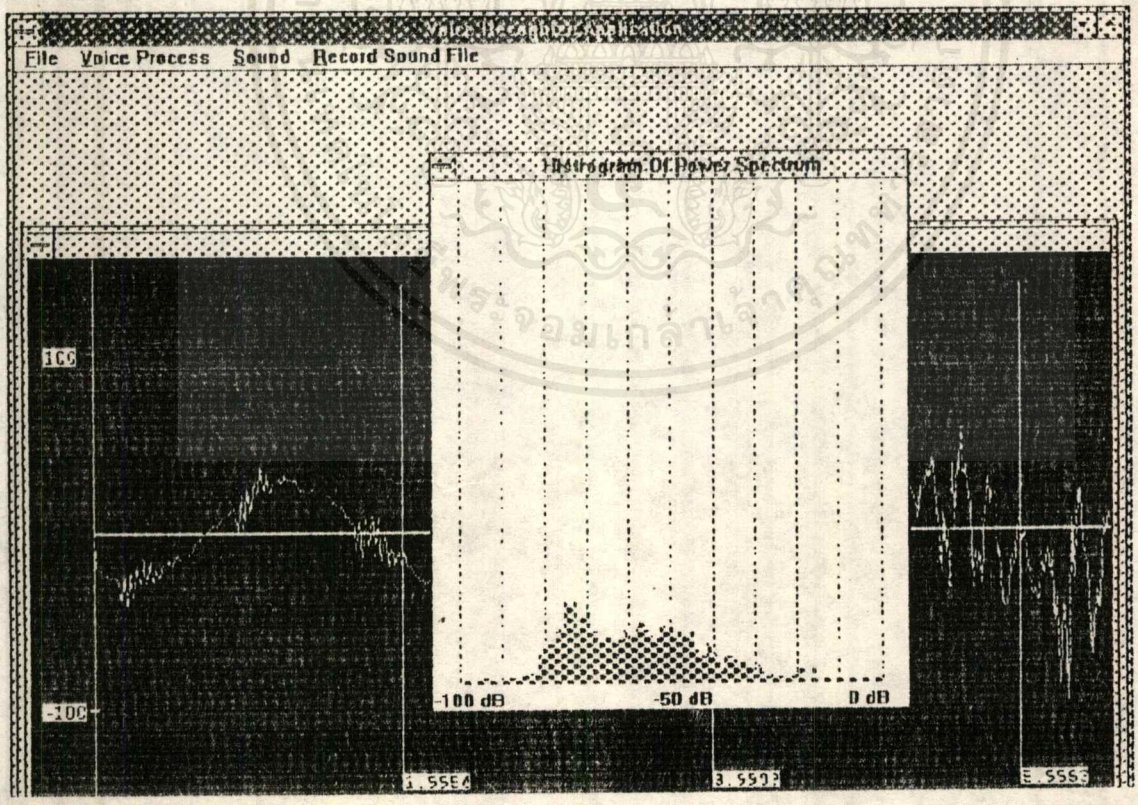


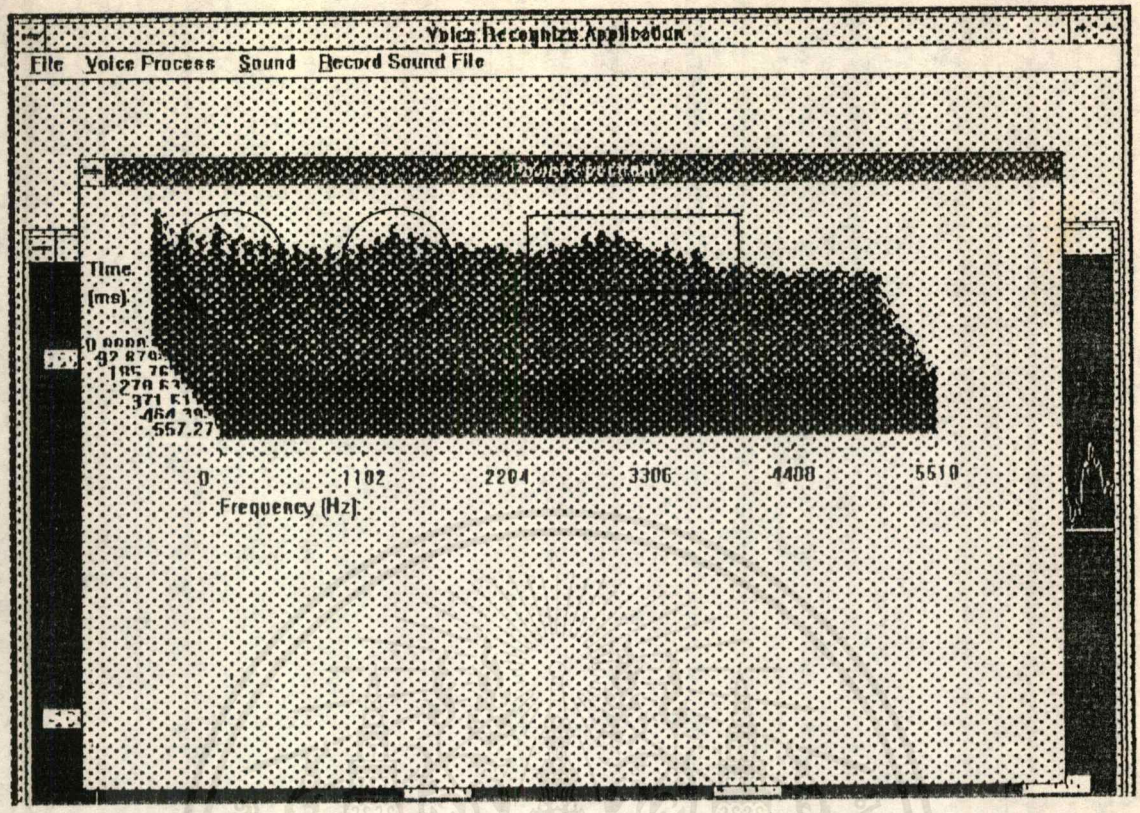
คำว่า "หนึ่ง" พูดโดยบุคคลที่ 2 ครั้งที่ 1



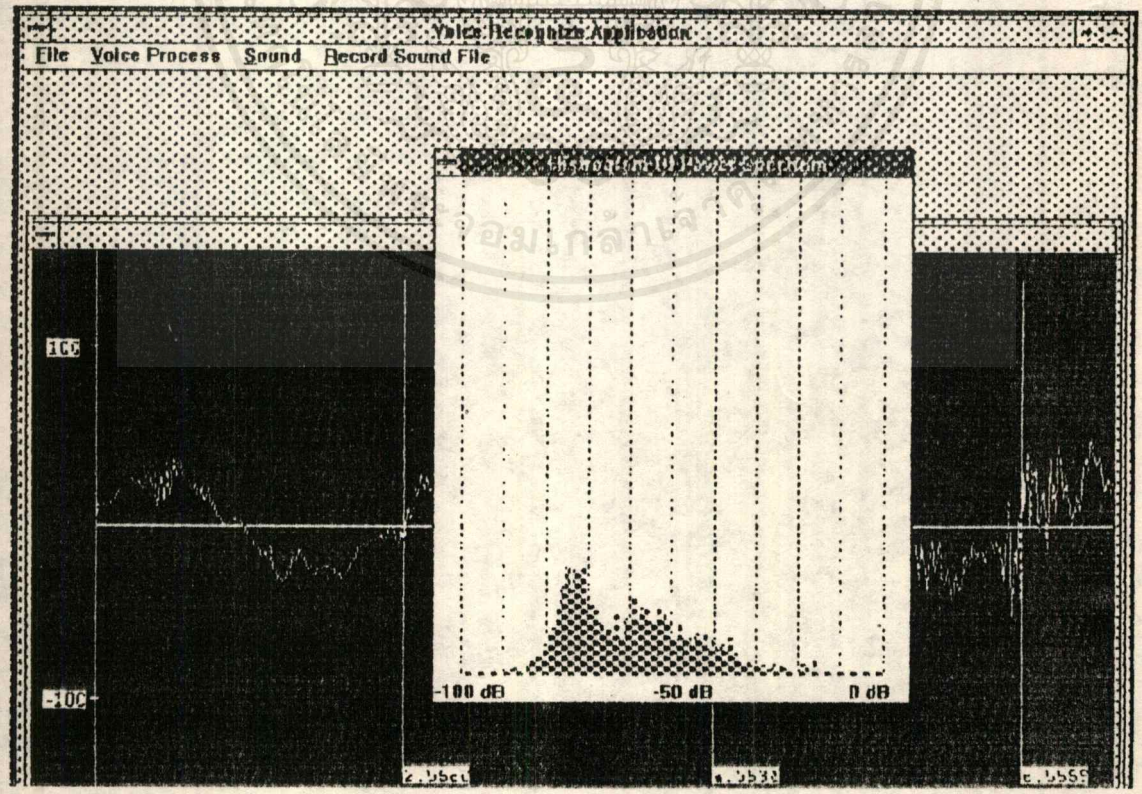


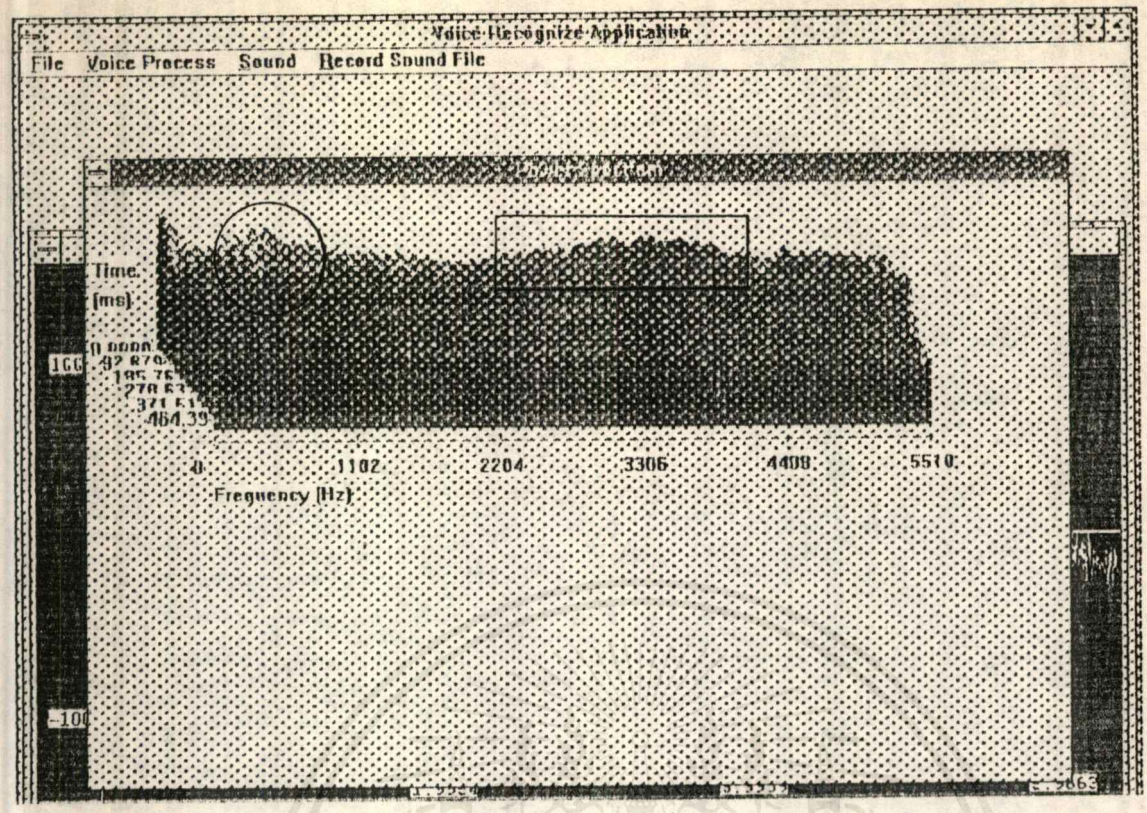
คำว่า "หนึ่ง" พูดโดยบุคคลที่ 2 ครั้งที่ 2



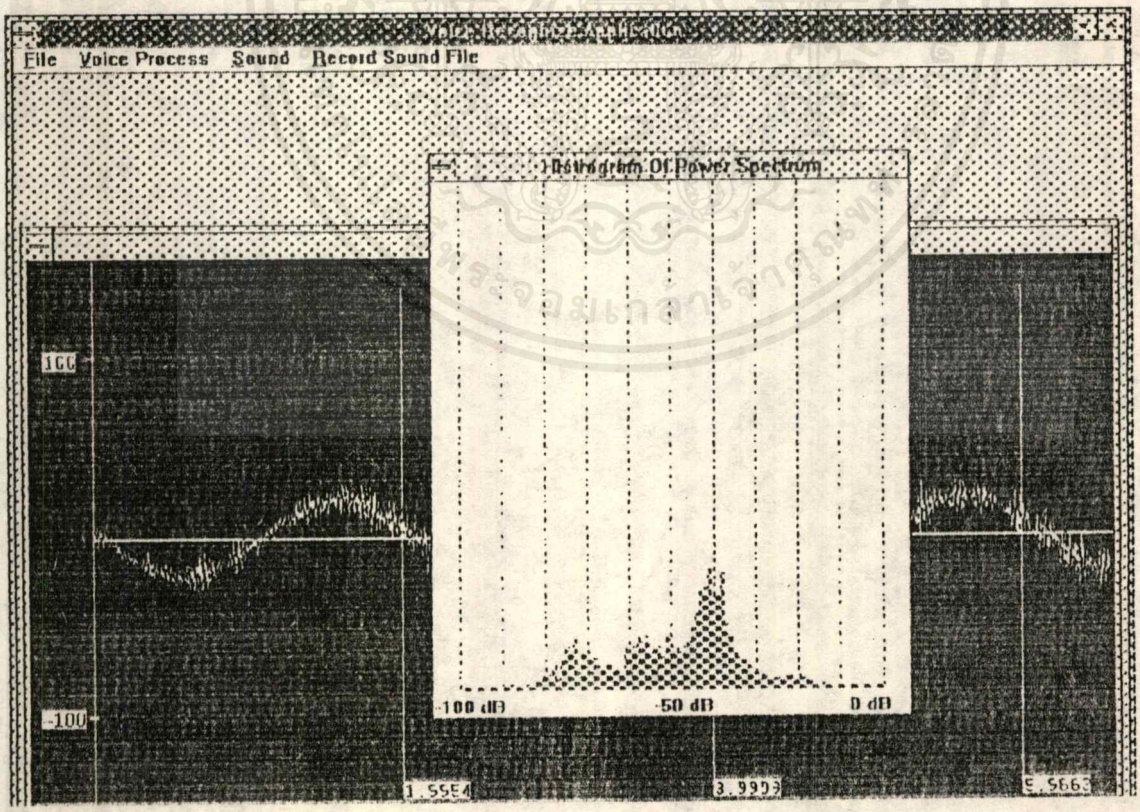


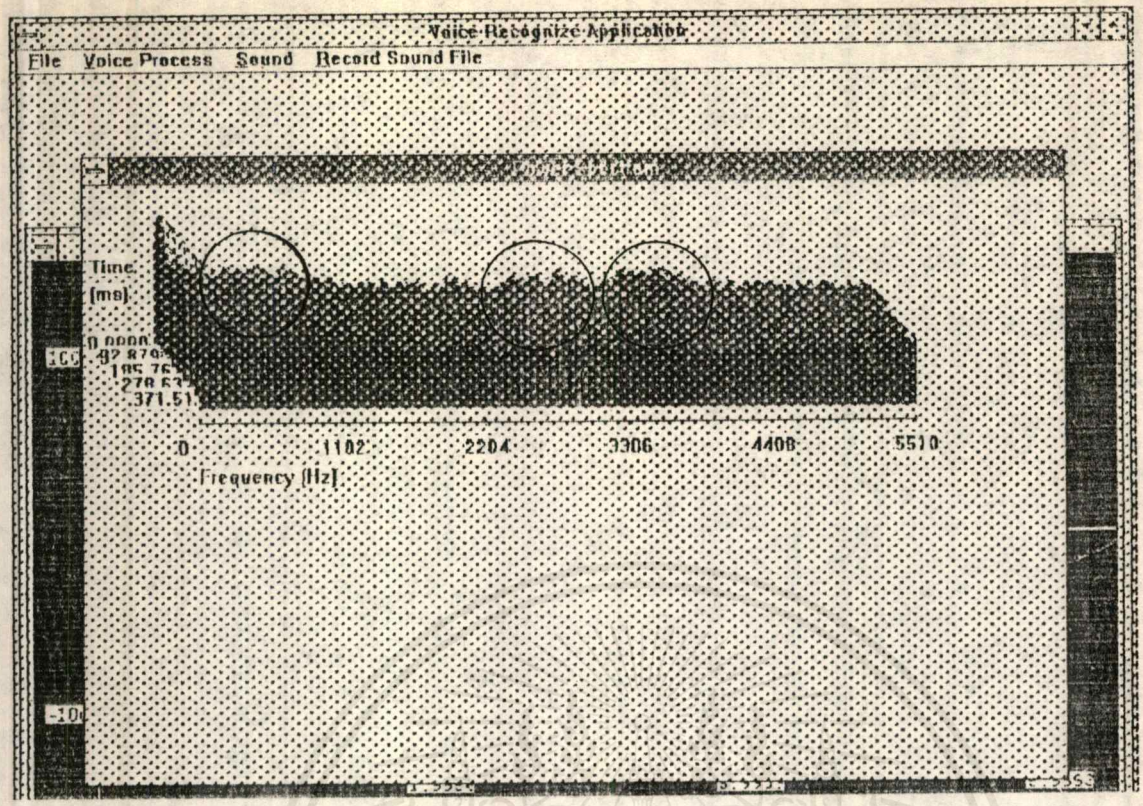
คำว่า "หนึ่ง" พูดโดยบุคคลที่ 2 ครั้งที่ 3



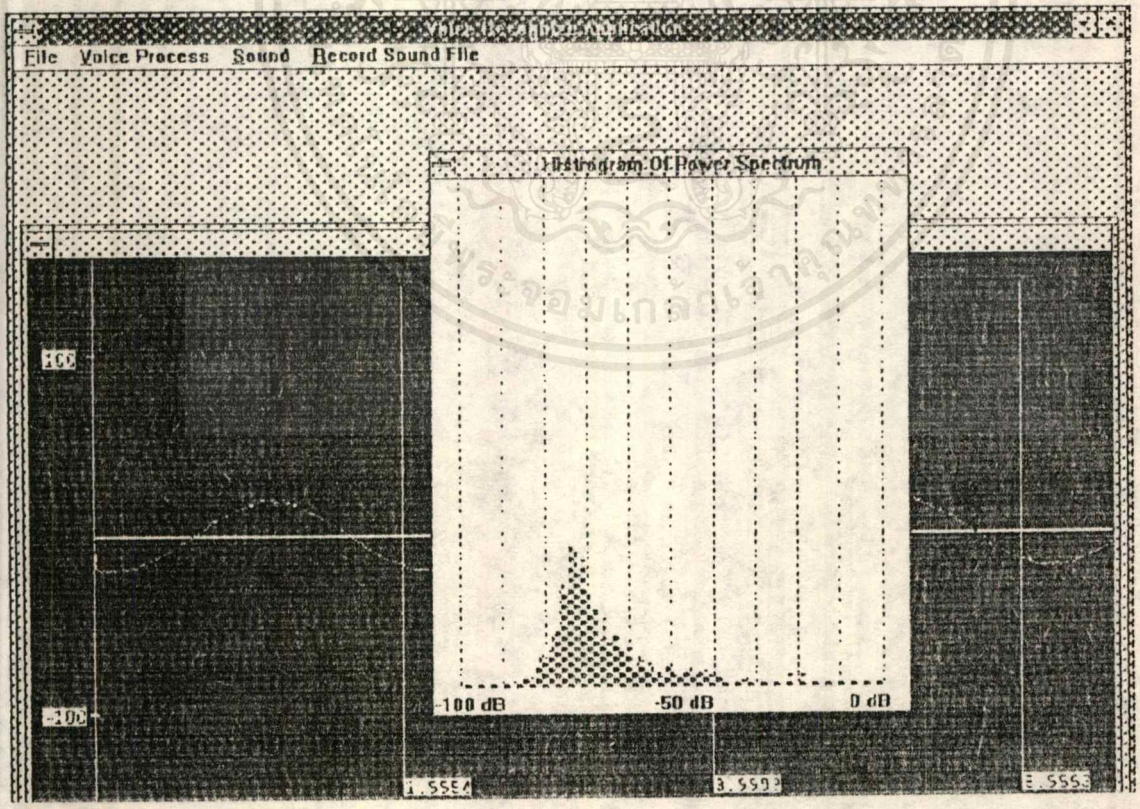


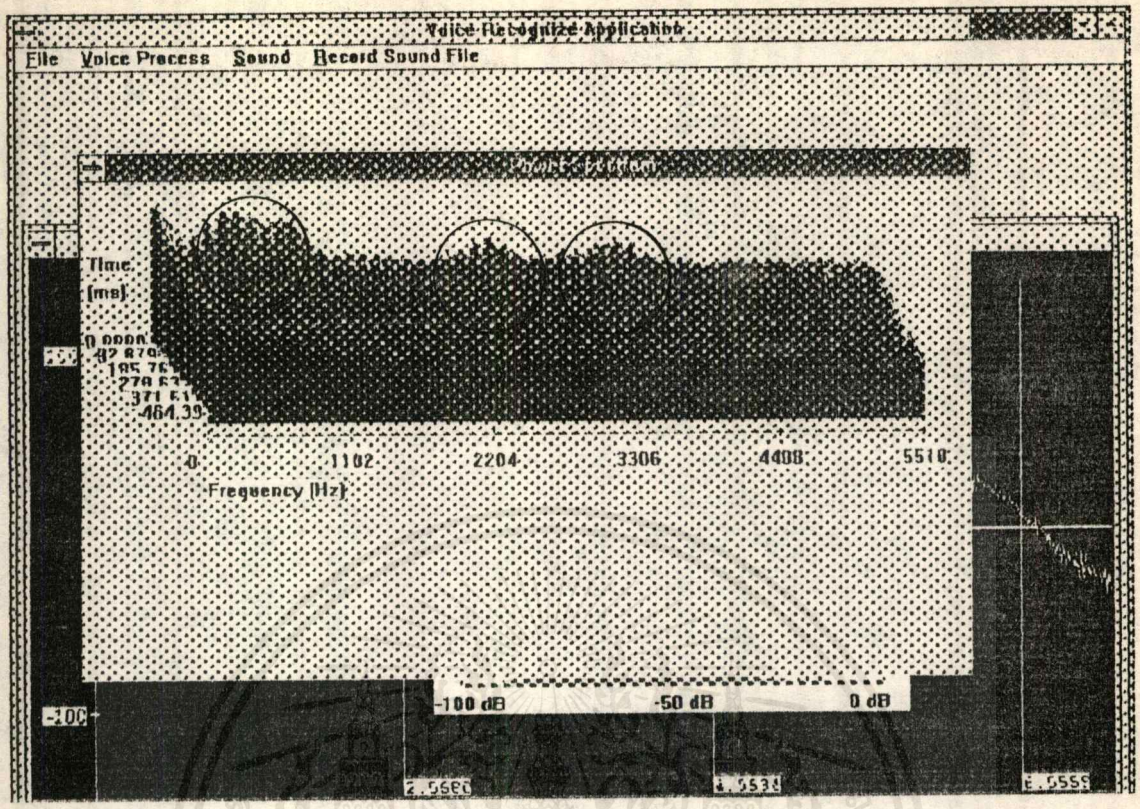
คำว่า "สอง" พูดโดยบุคคลที่ 1 ครั้งที่ 1



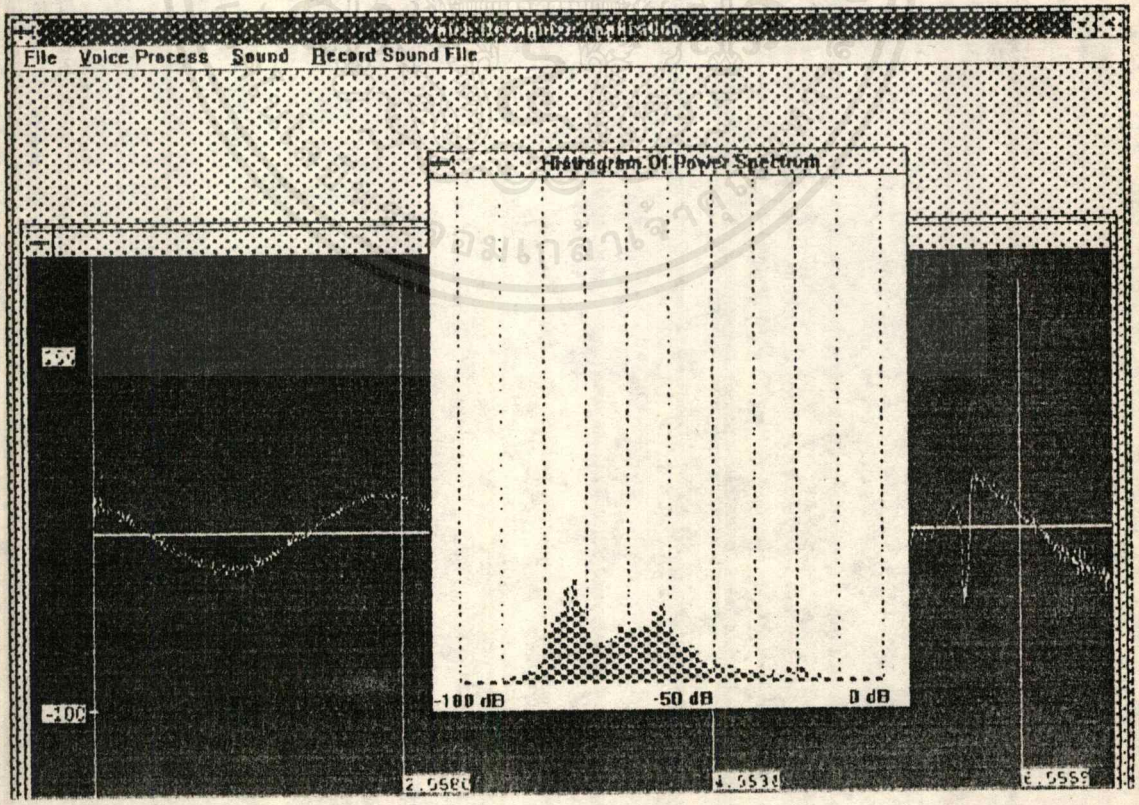


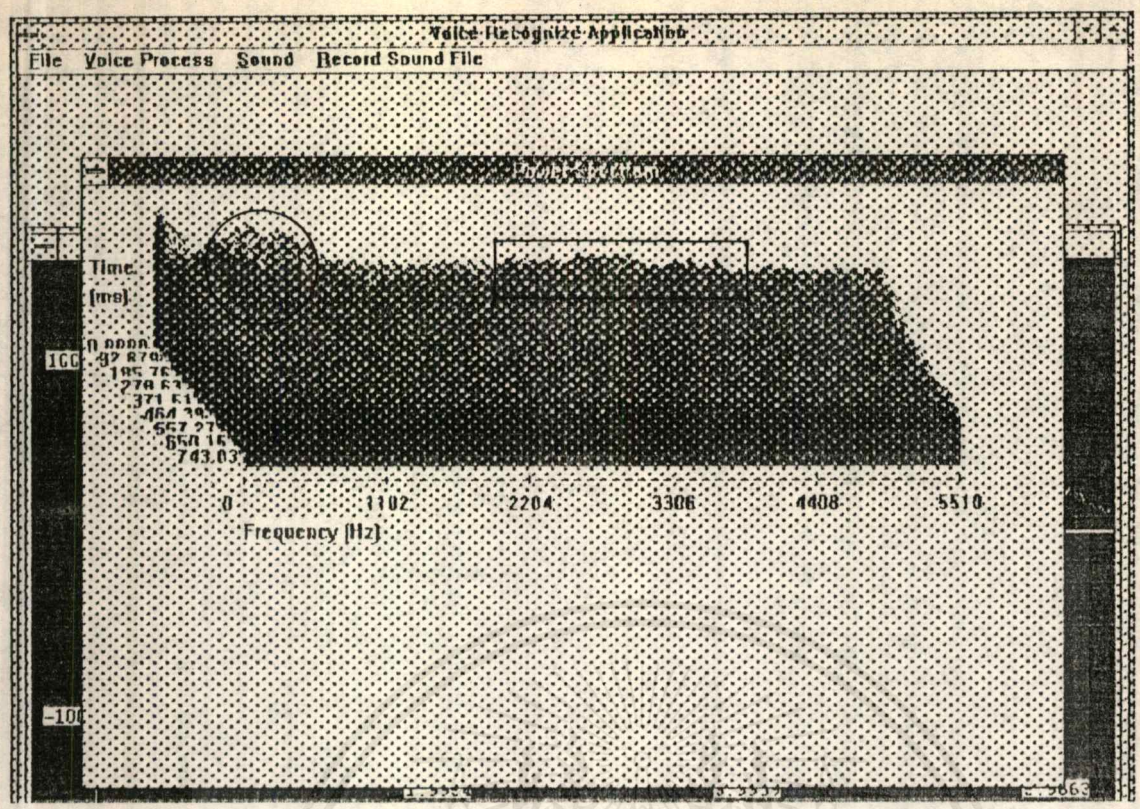
คำว่า "สอง" พูดโดยบุคคลที่ 1 ครั้งที่ 2



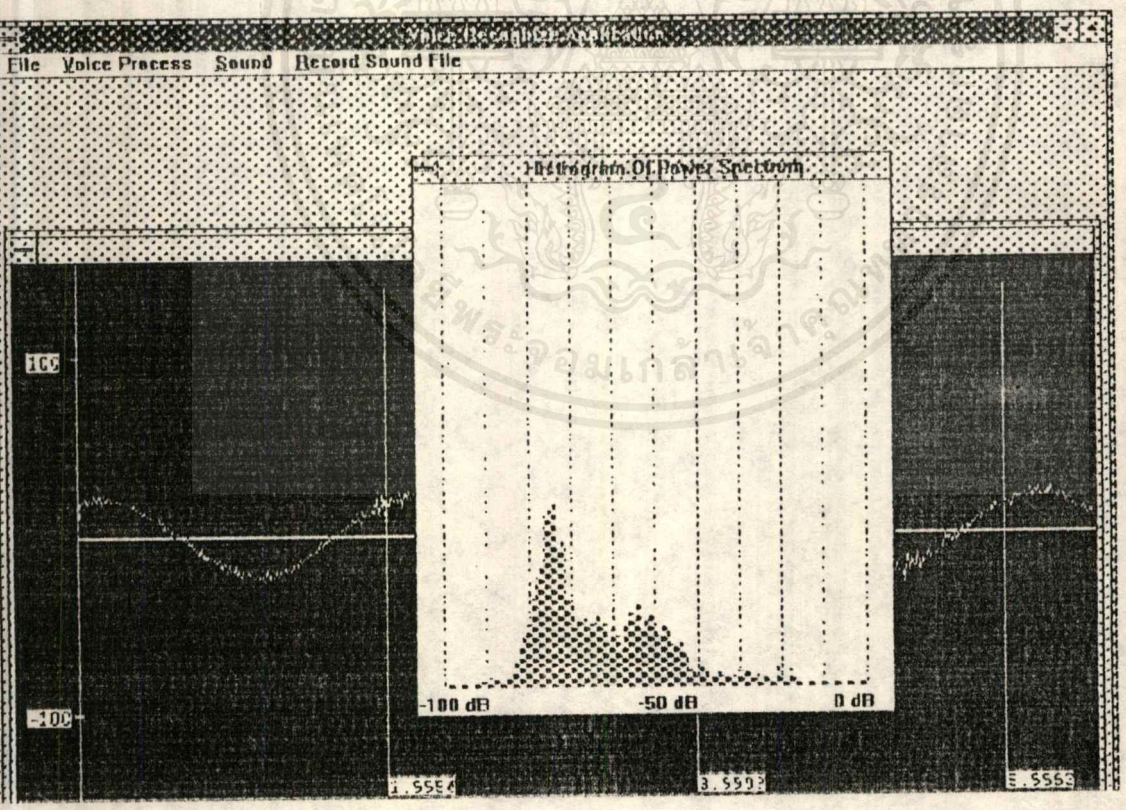


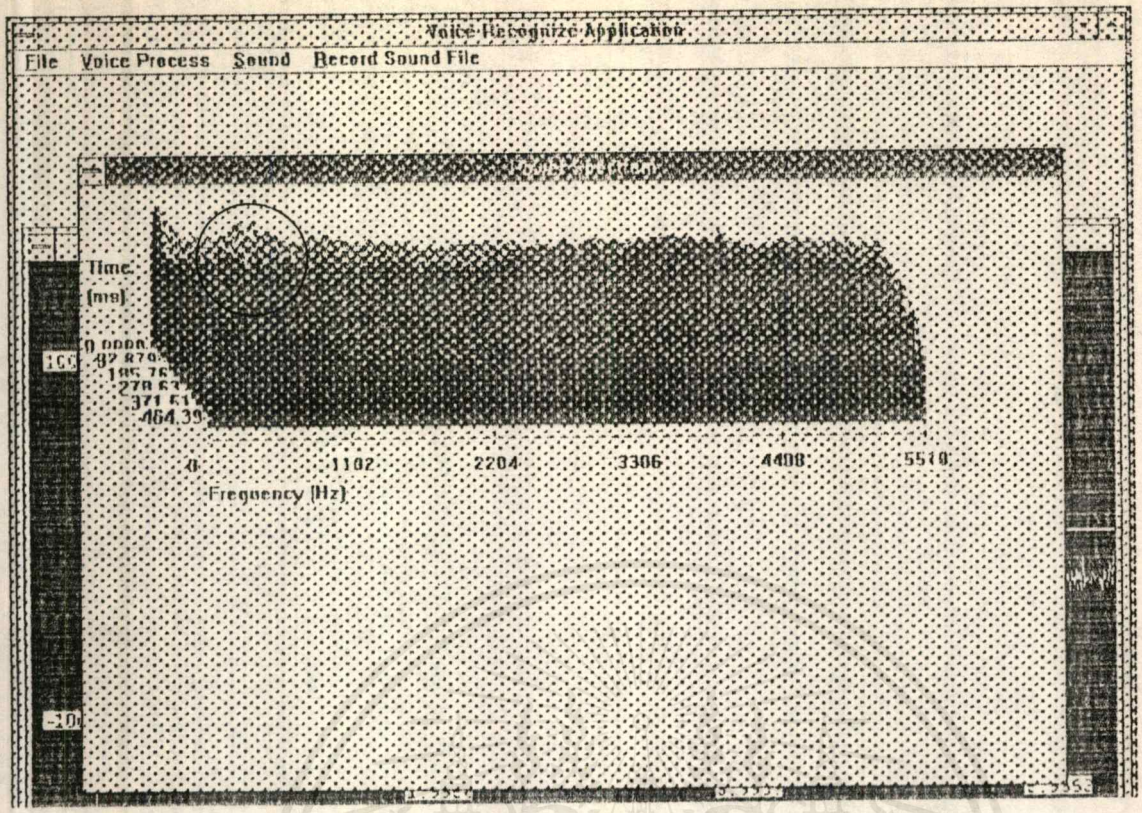
คำว่า "สอง" พูดโดยบุคคลที่ 1 ครั้งที่ 3



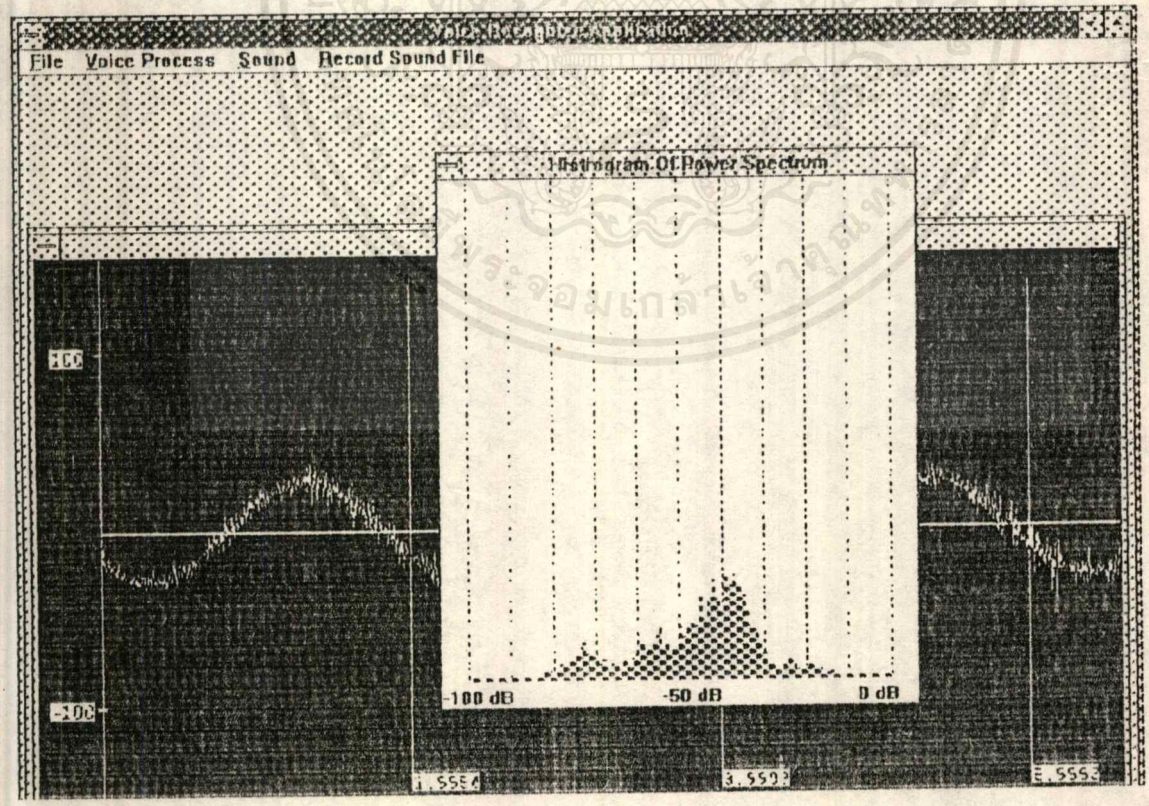


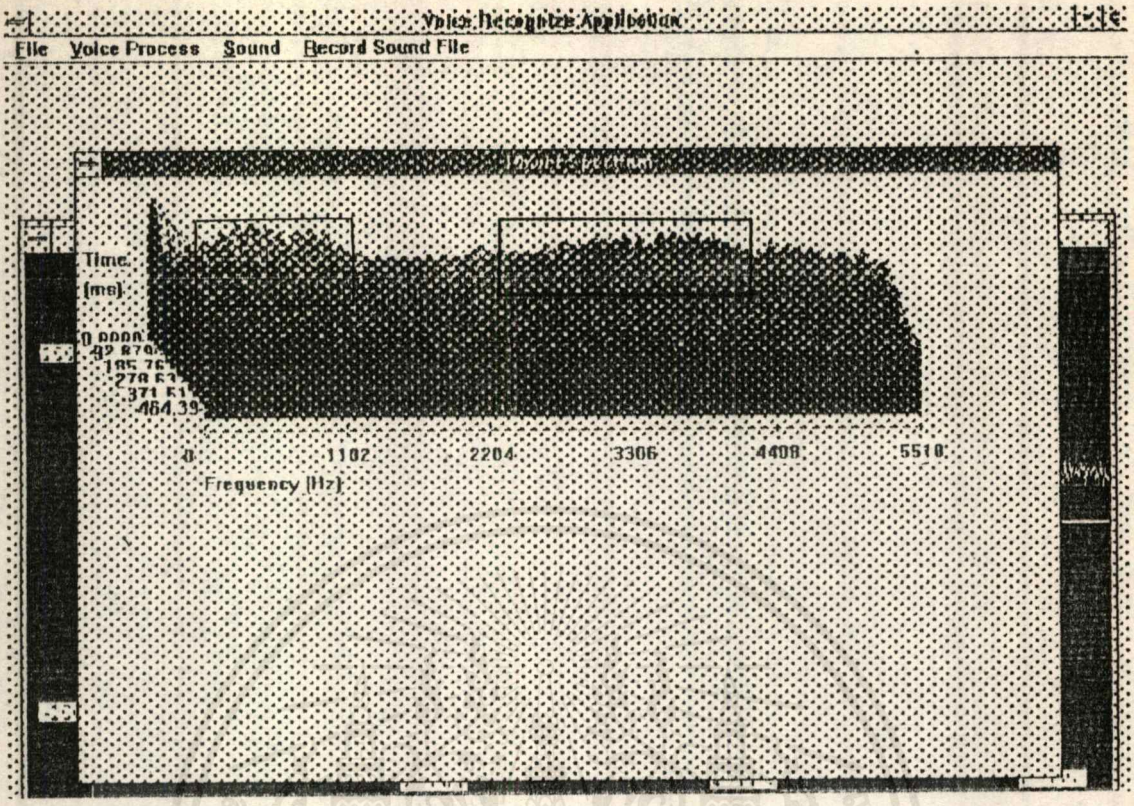
คำว่า "สอง" พูดโดยบุคคลที่ 2 ครั้งที่ 1



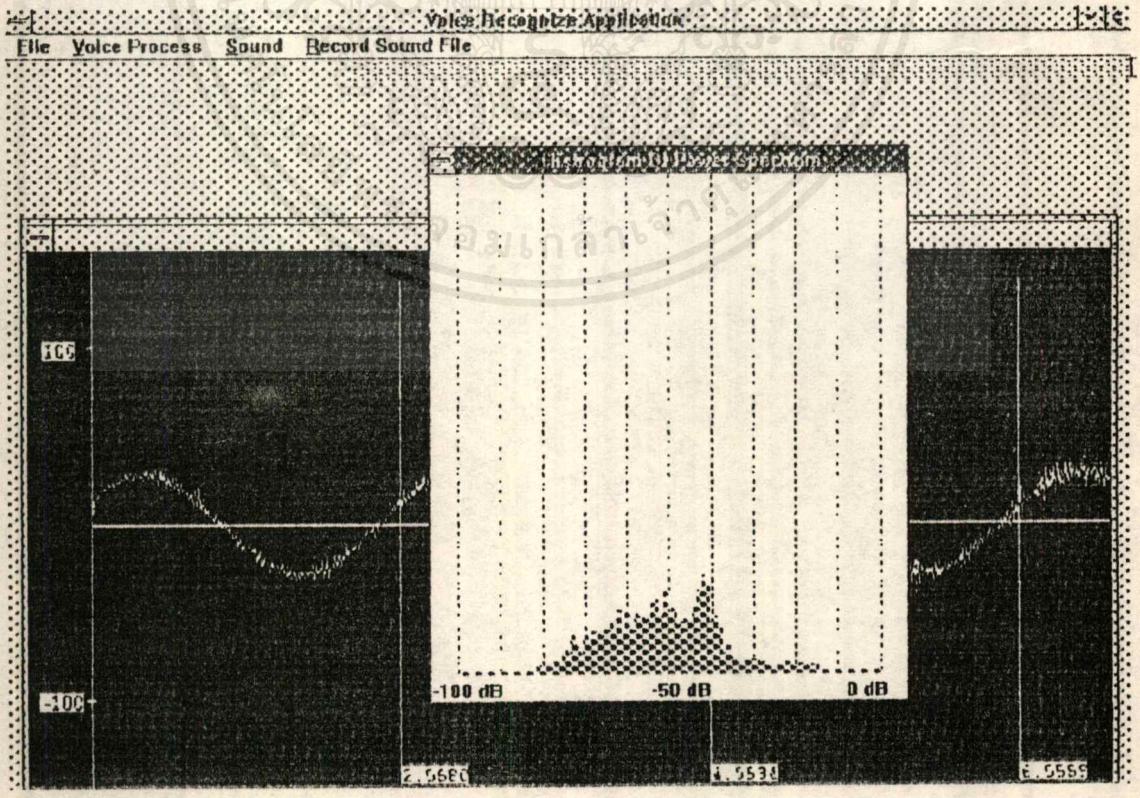


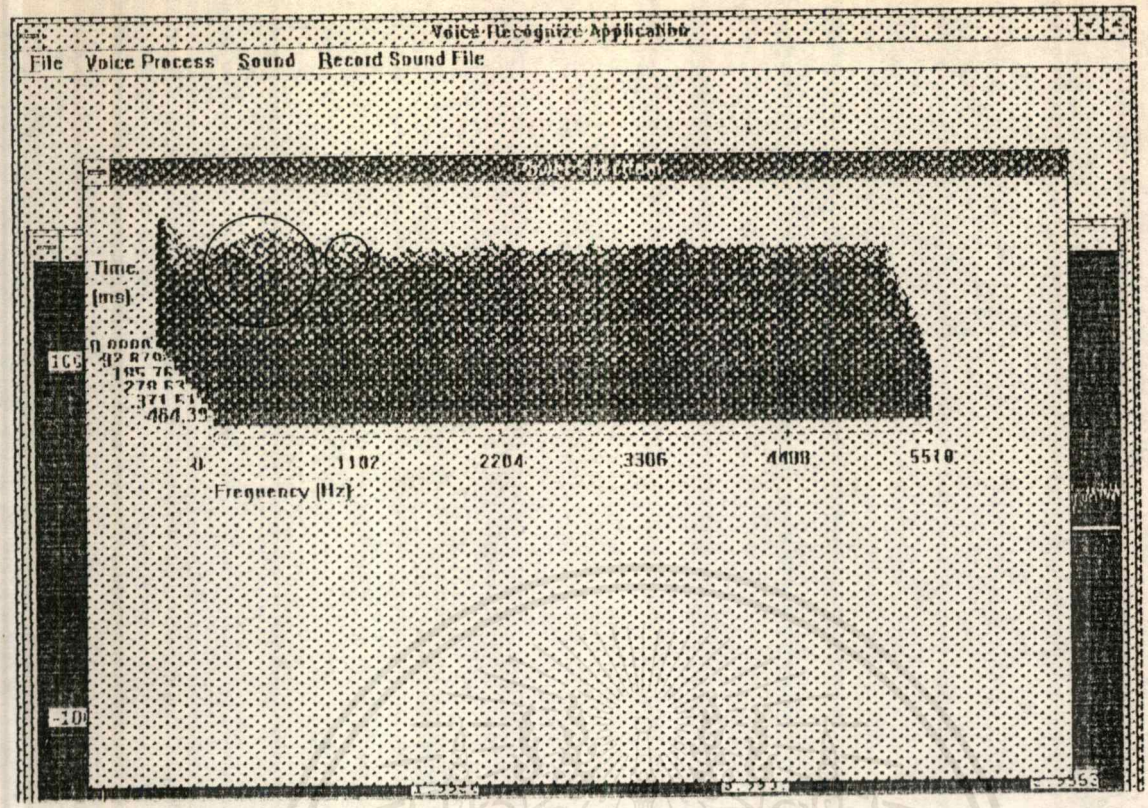
คำว่า "สอง" พูดโดยบุคคลที่ 2 ครั้งที่ 2



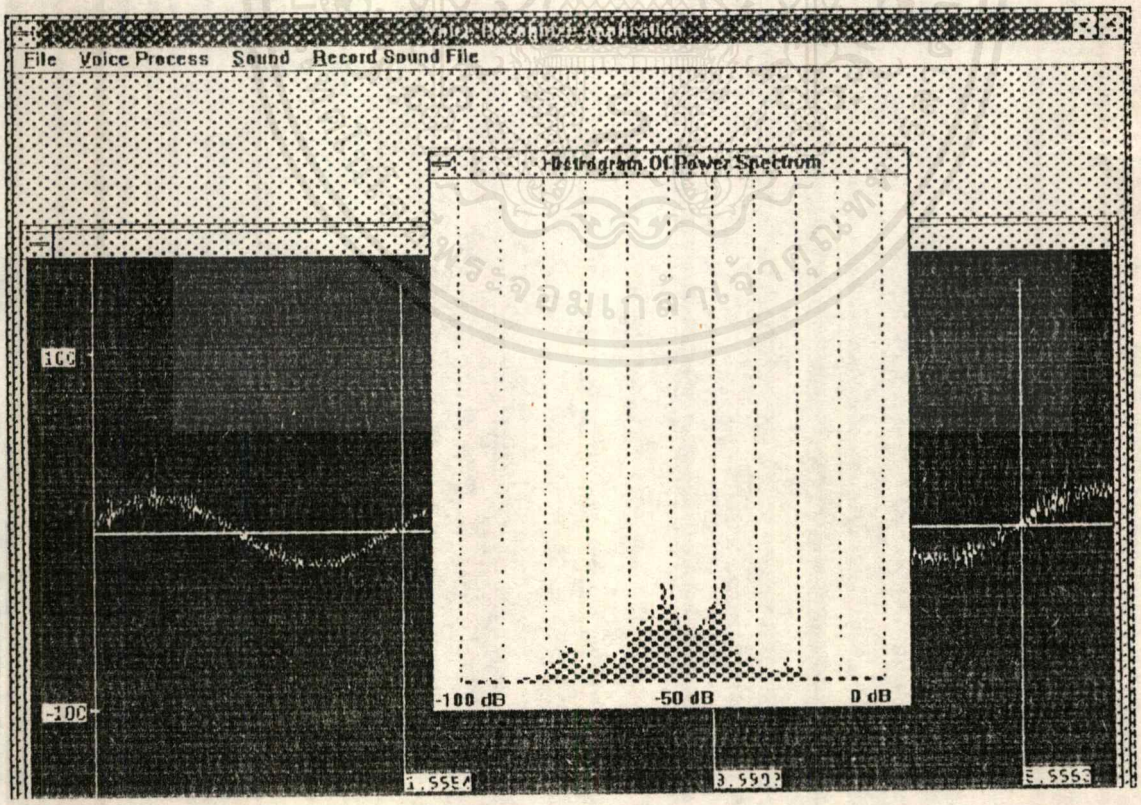


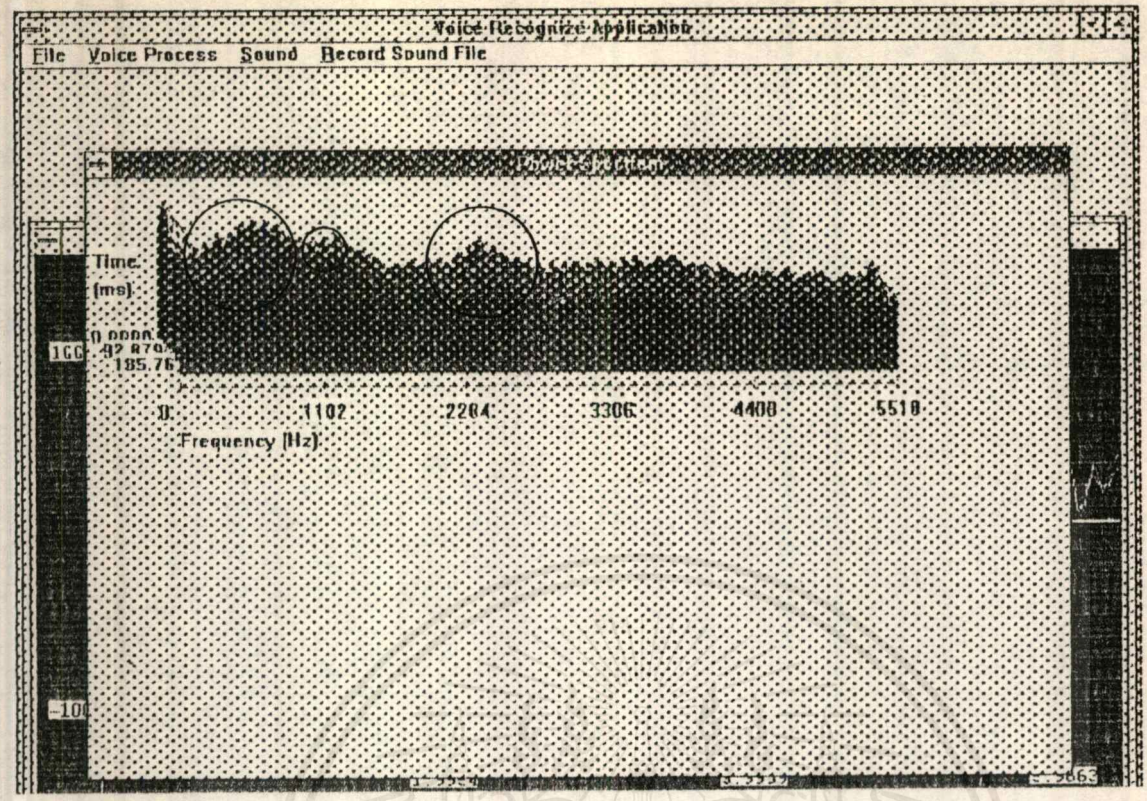
คำว่า "สอง" พูดโดยบุคคลที่ 2 ครั้งที่ 3



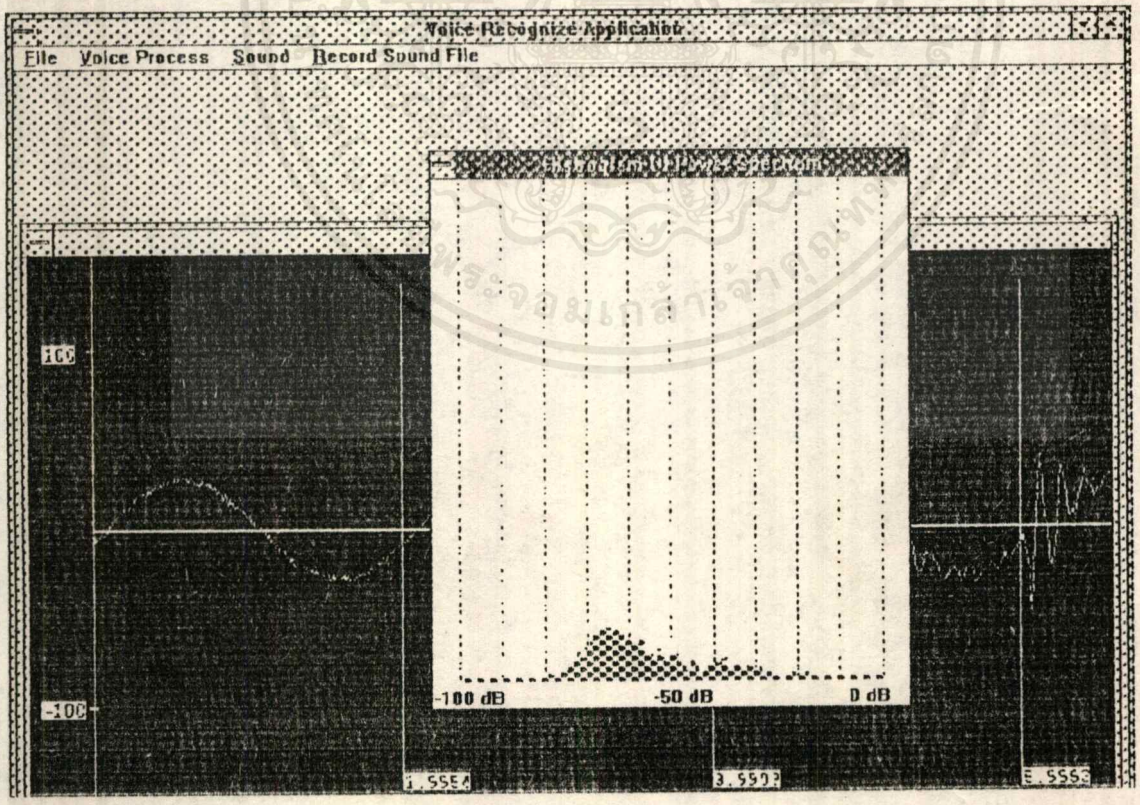


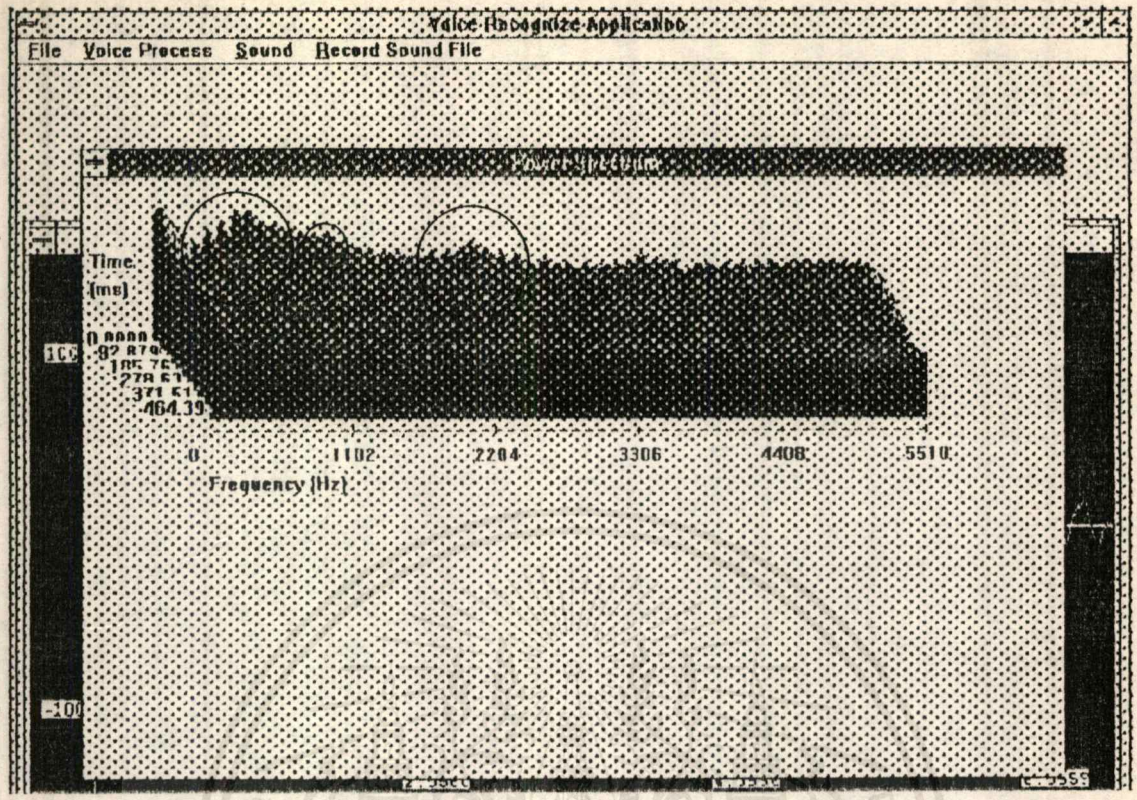
คำว่า "สาม" พูดโดยบุคคลที่ 1 ครั้งที่ 1



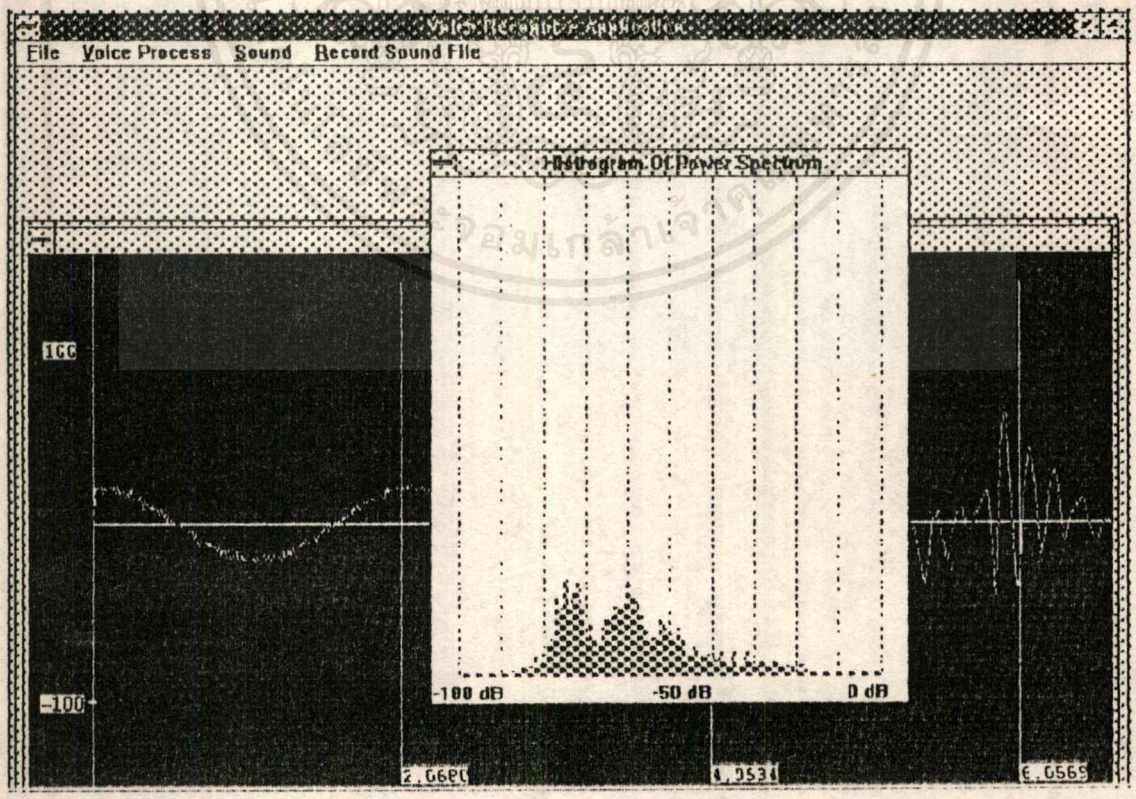


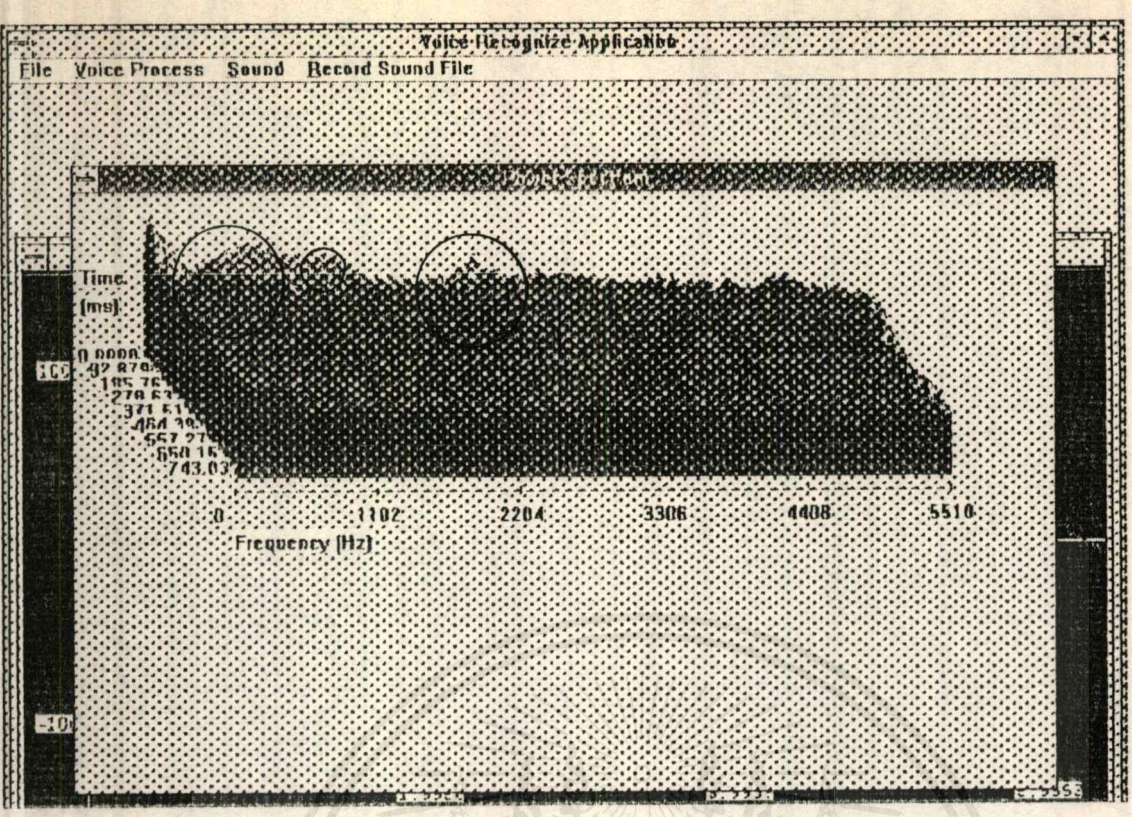
คำว่า "สาม" พูดโดยบุคคลที่ 1 ครั้งที่ 2



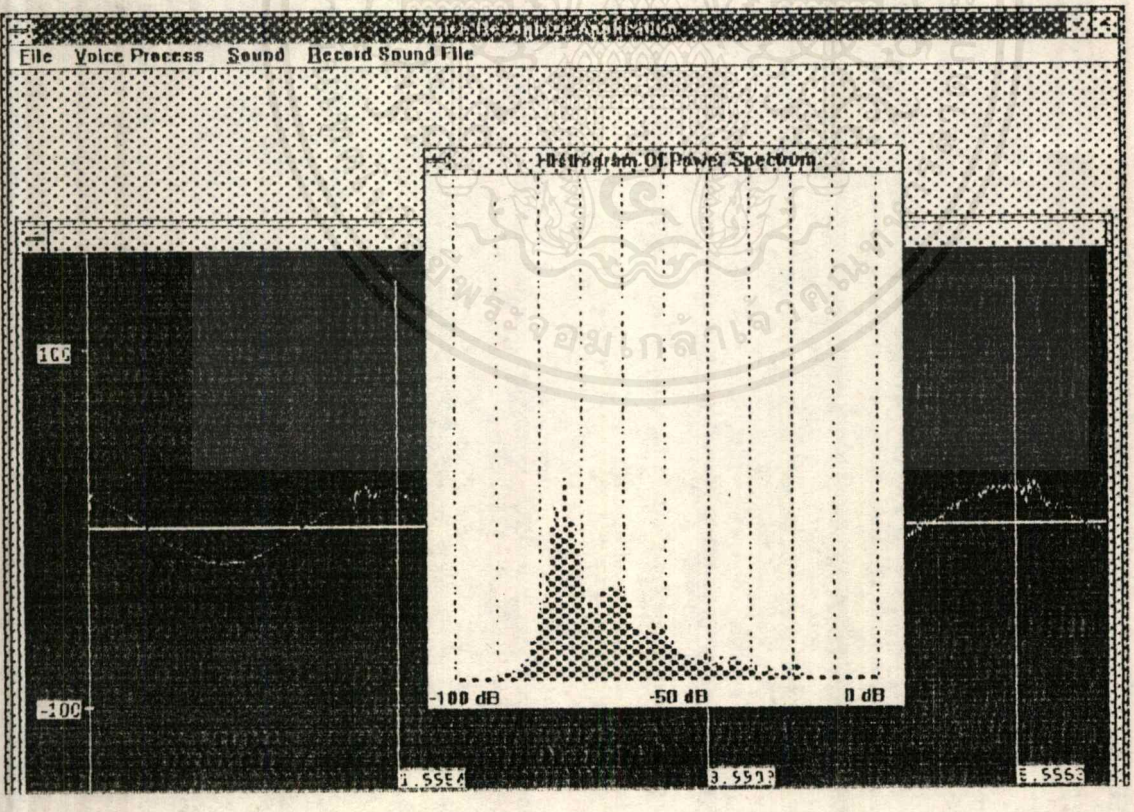


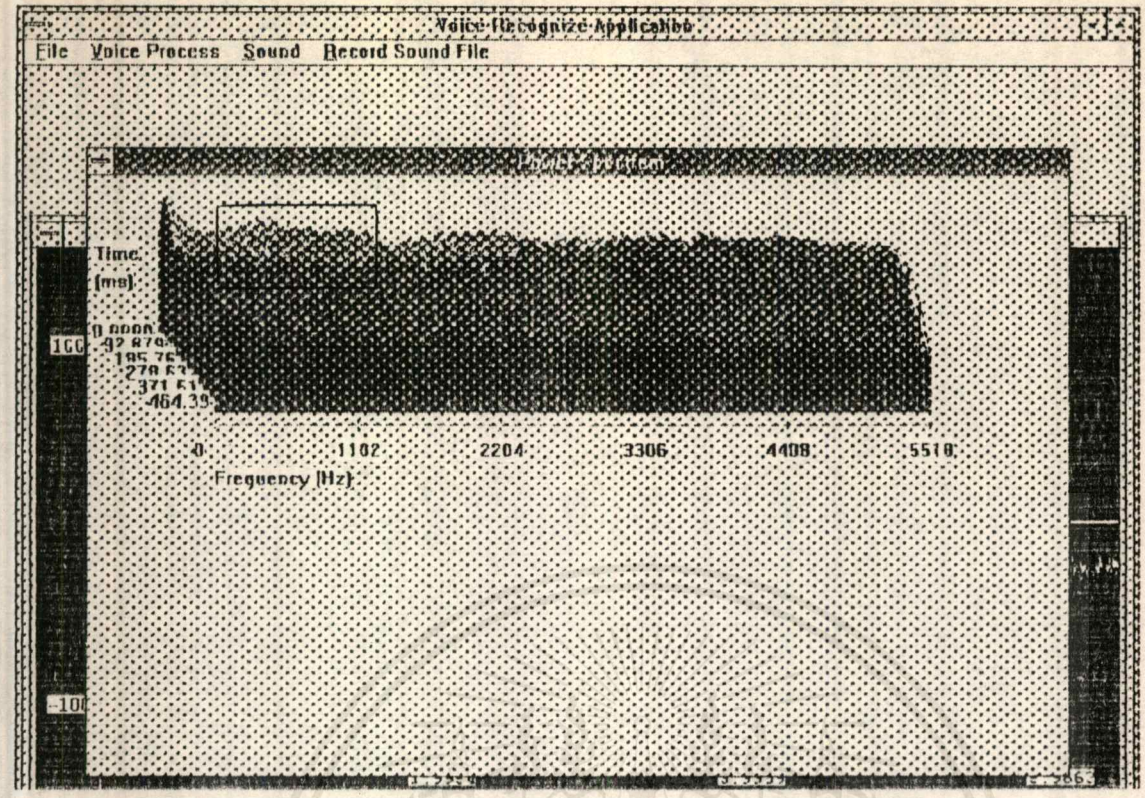
คำว่า "สาม" พูดโดยบุคคลที่ 1 ครั้งที่ 3



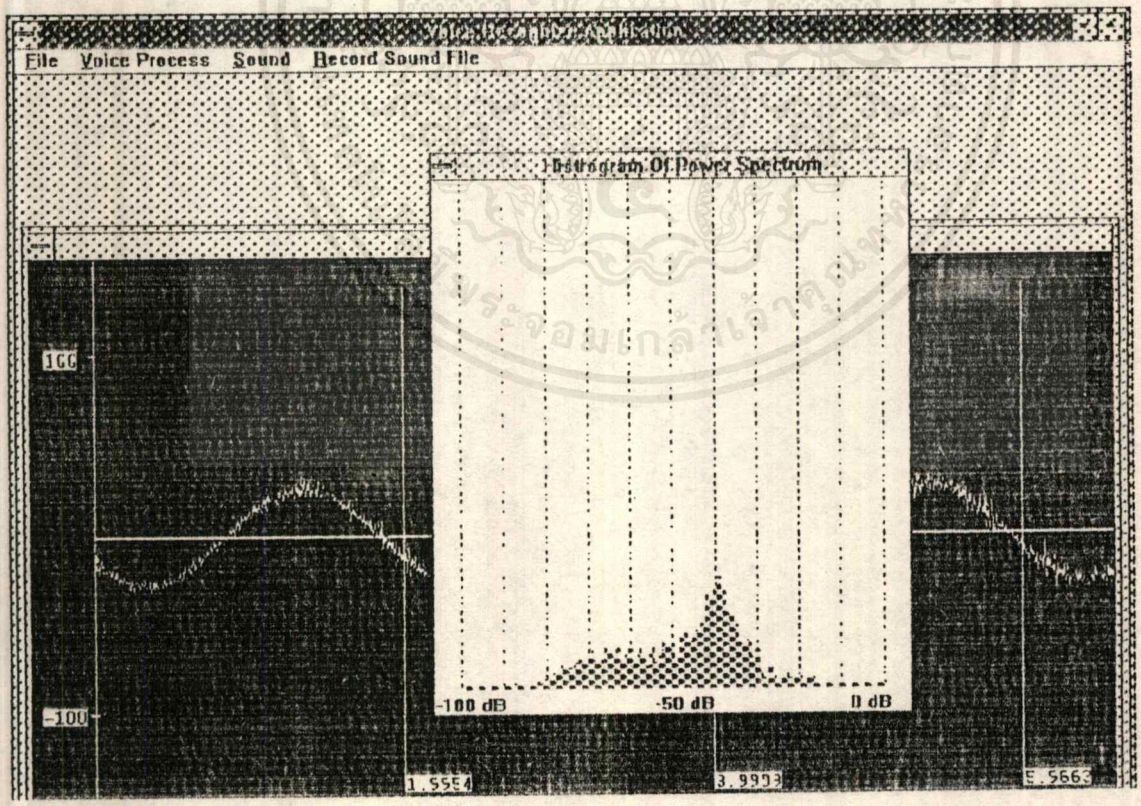


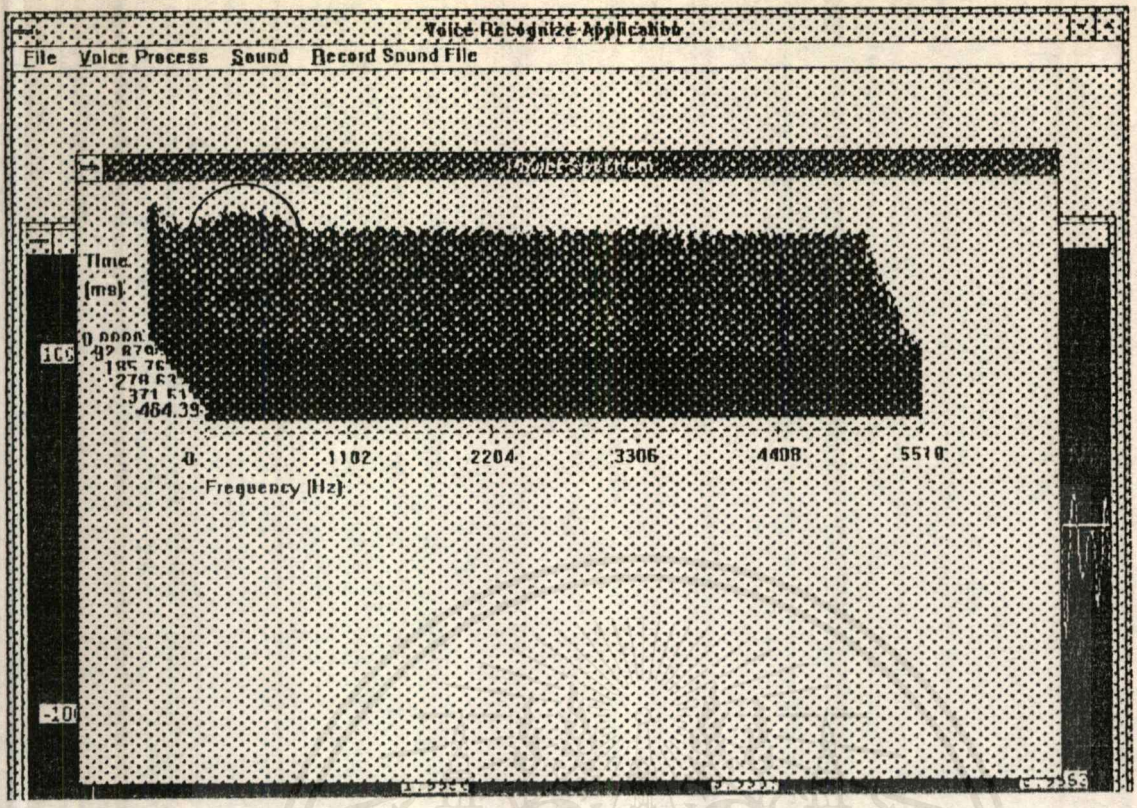
คำว่า "สาม" พูดโดยบุคคลที่ 2 ครั้งที่ 1



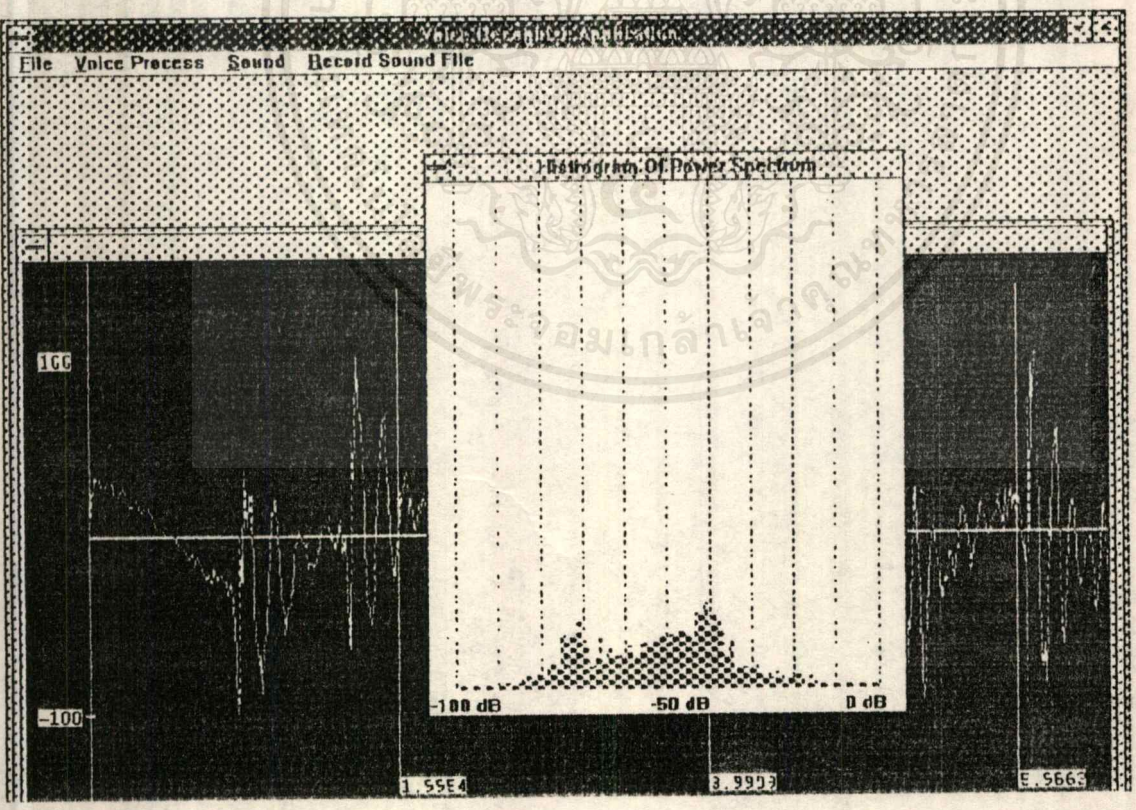


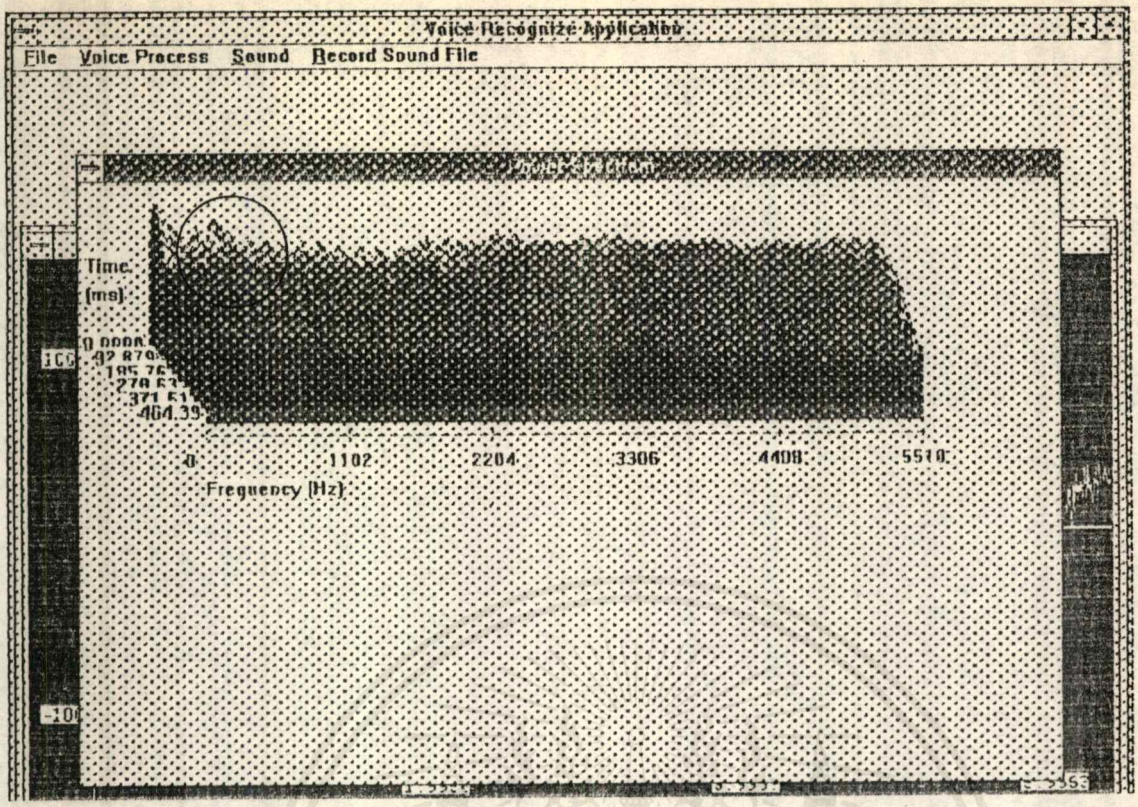
คำว่า "สาม" พูดโดยบุคคลที่ 2 ครั้งที่ 2



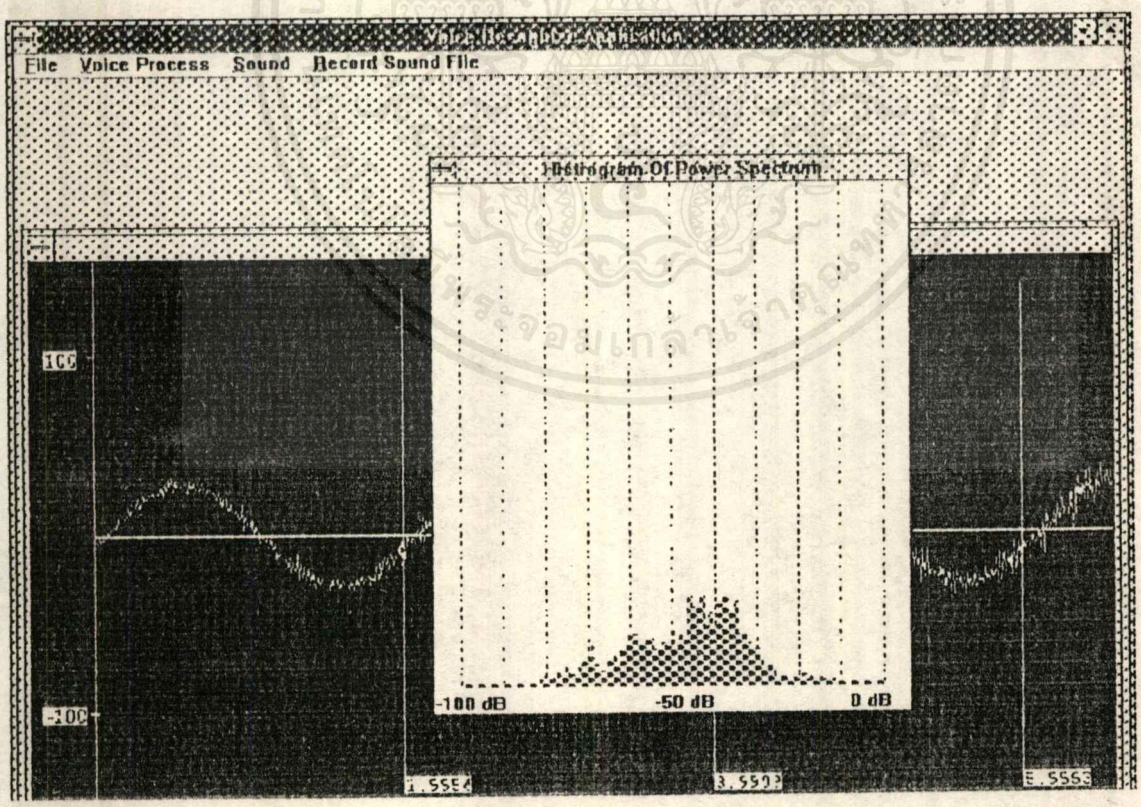


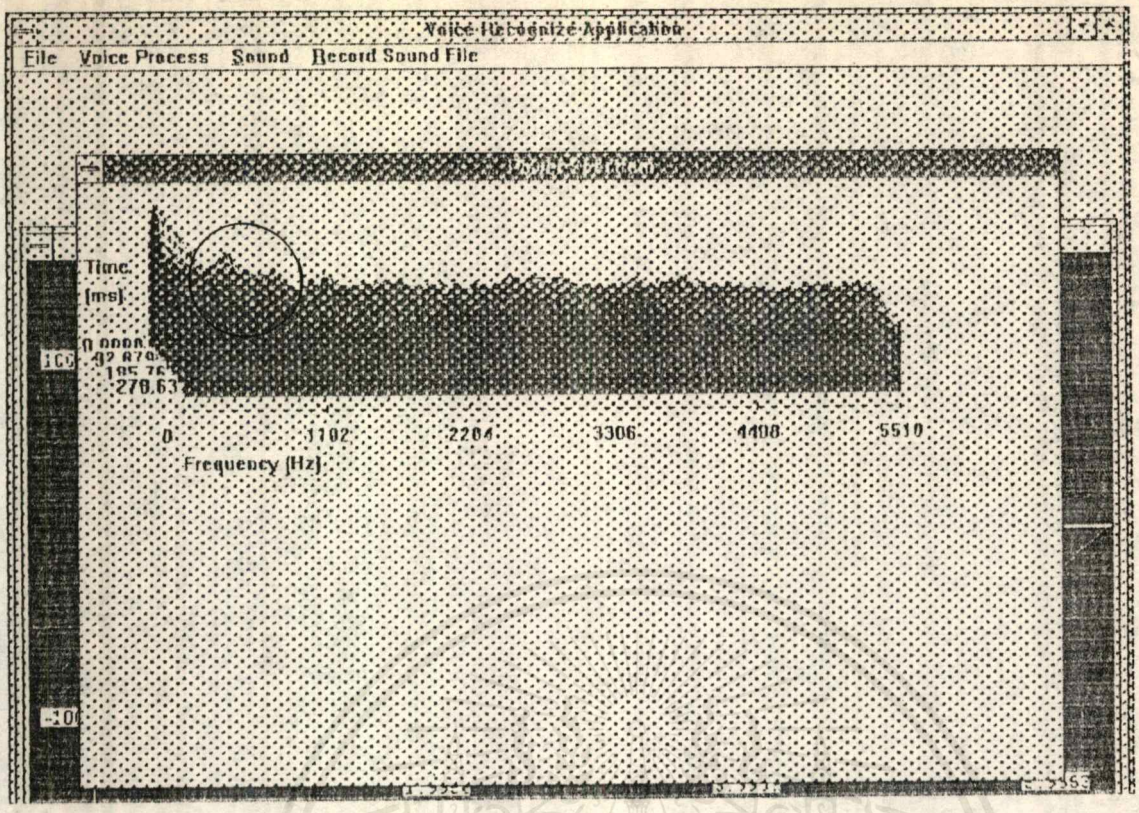
คำว่า "สาม" พูดโดยบุคคลที่ 2 ครั้งที่ 3



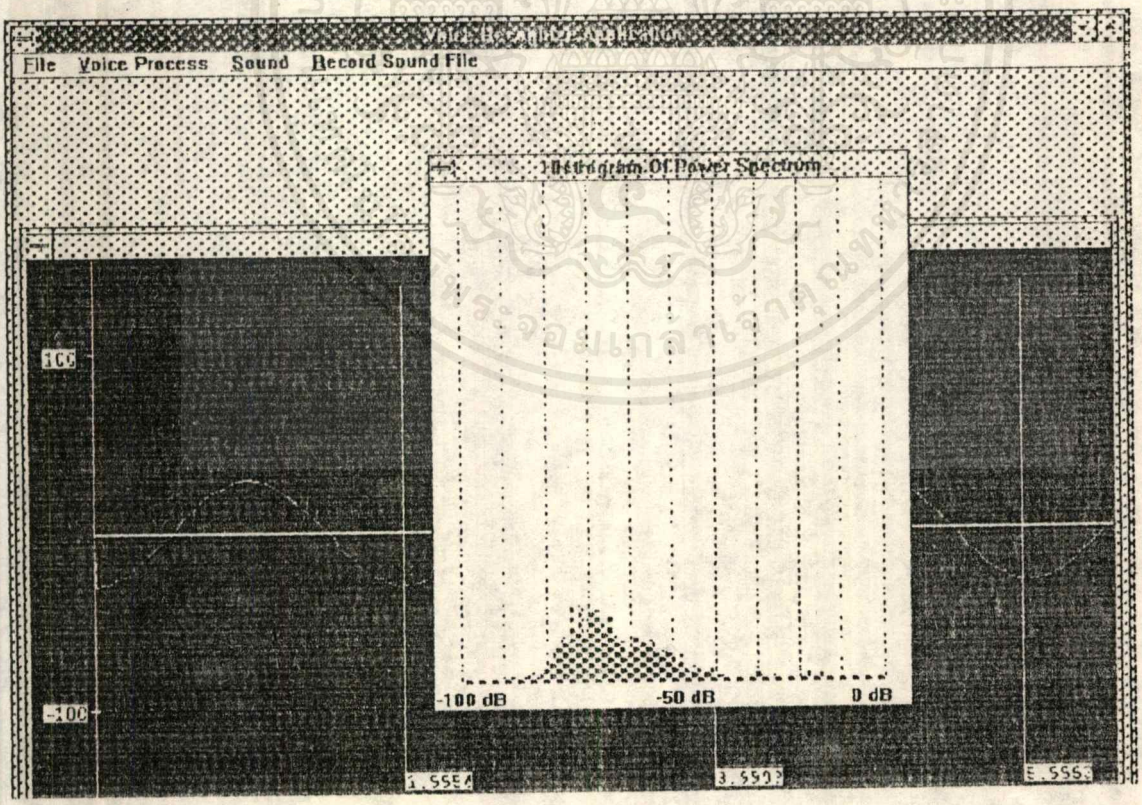


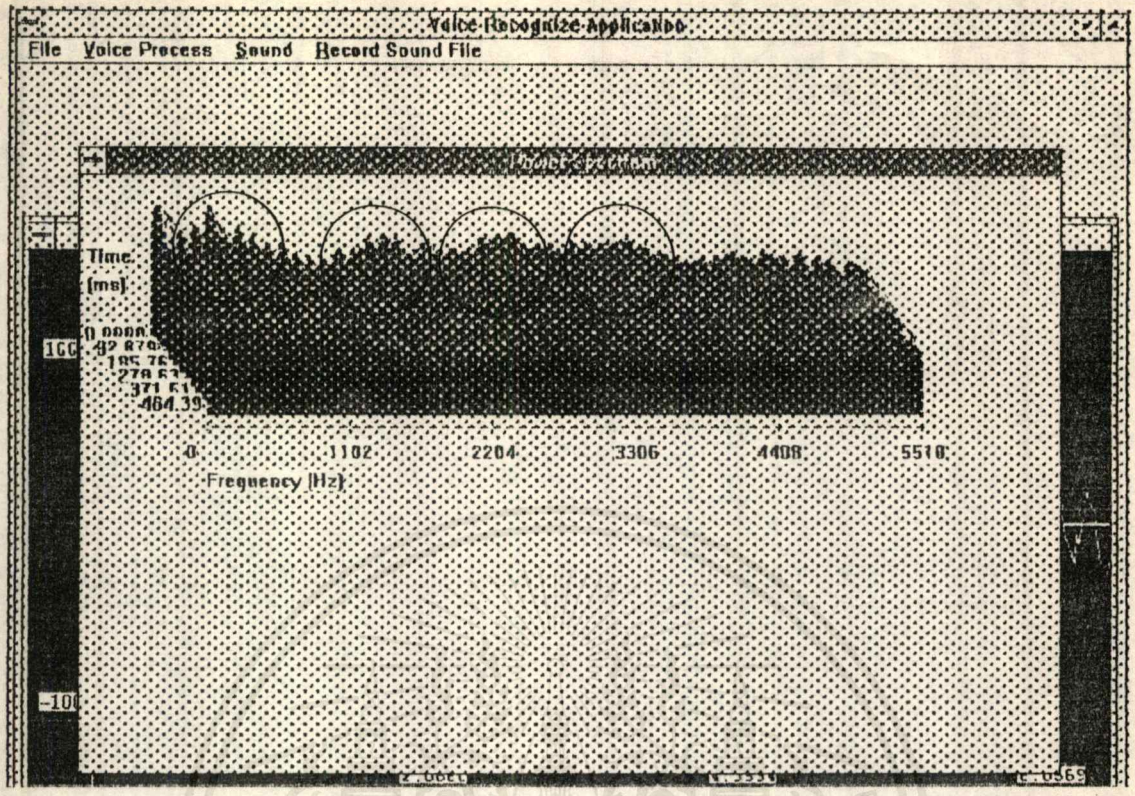
คำว่า "สี่" พูดโดยบุคคลที่ 1 ครั้งที่ 1



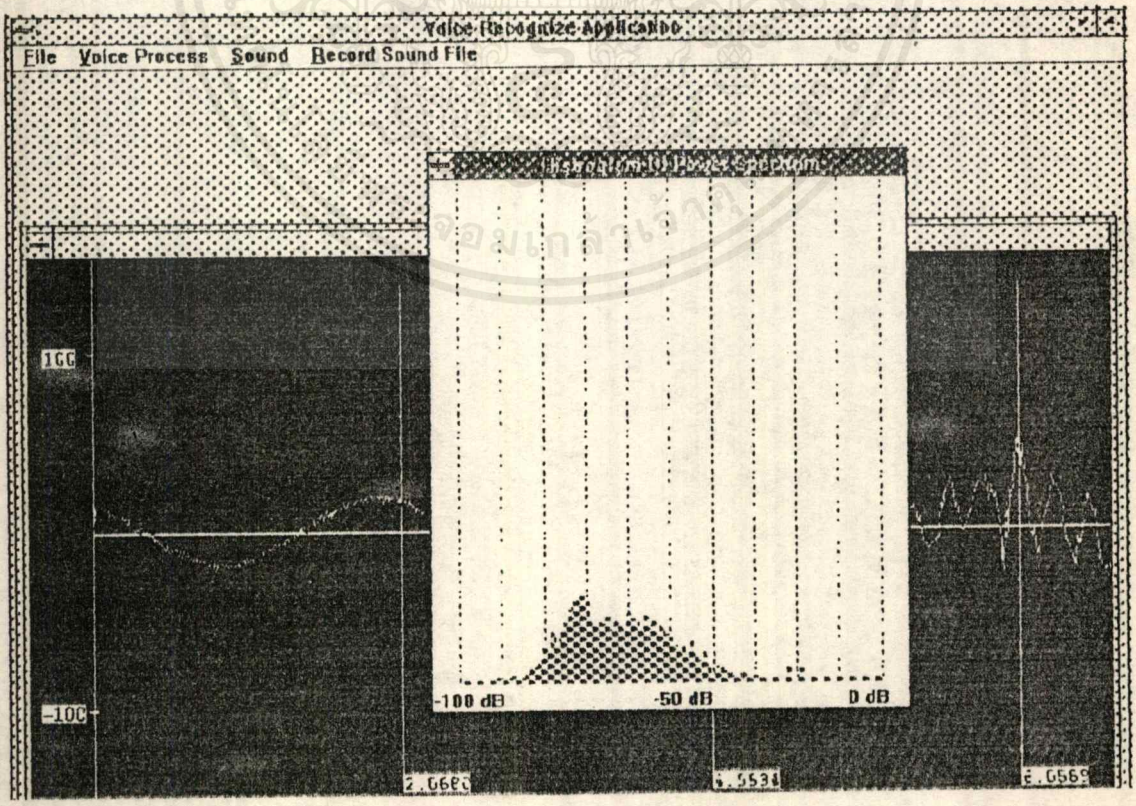


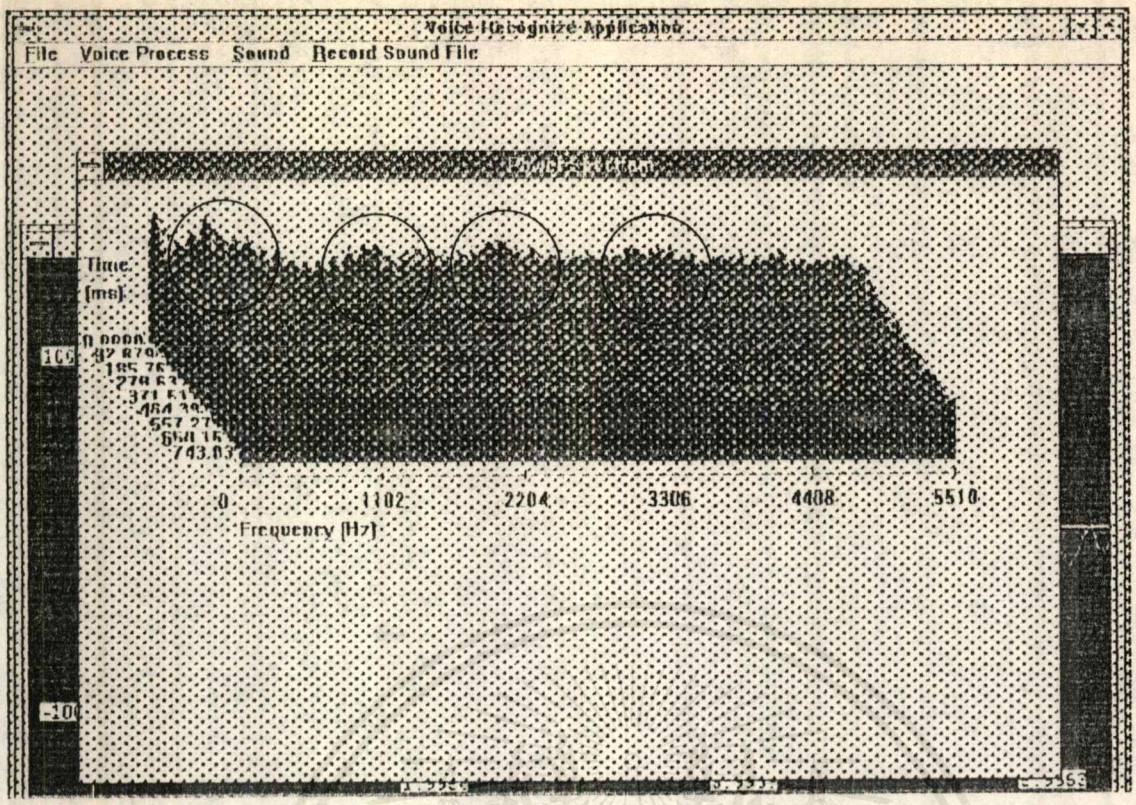
คำว่า "สี" พูดโดยบุคคลที่ 1 ครั้งที่ 2



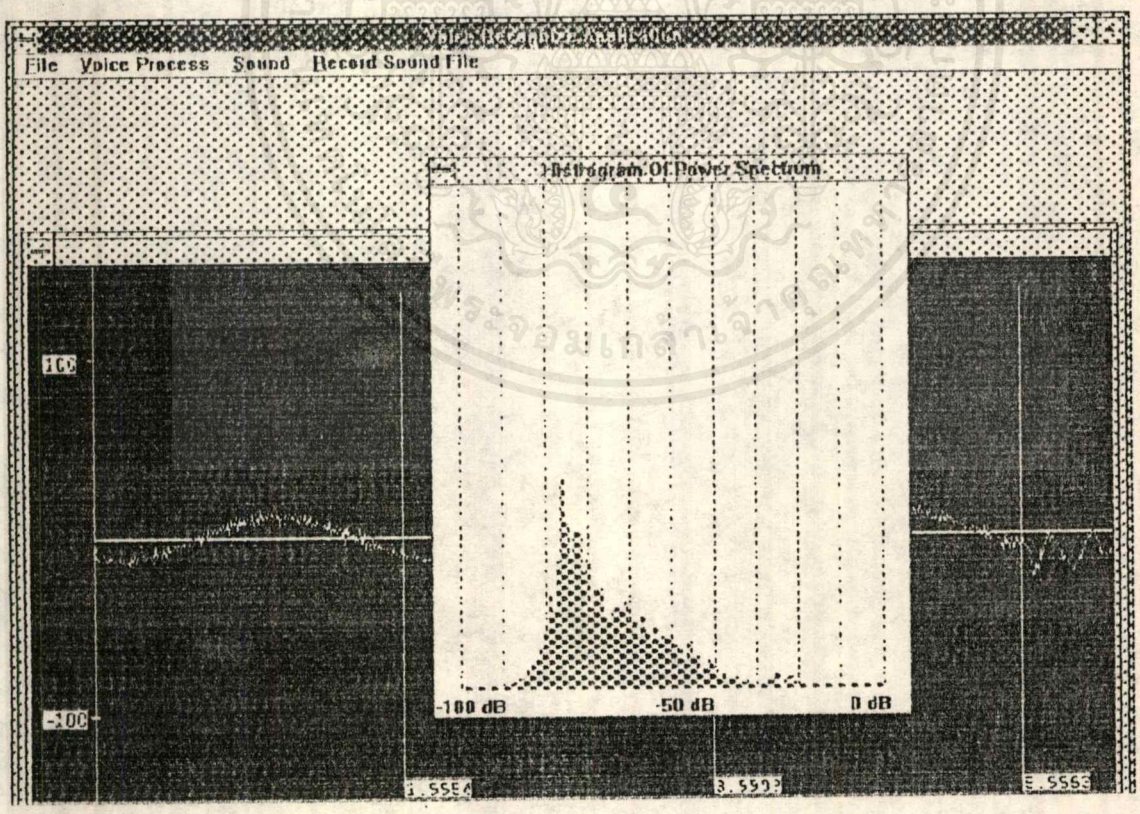


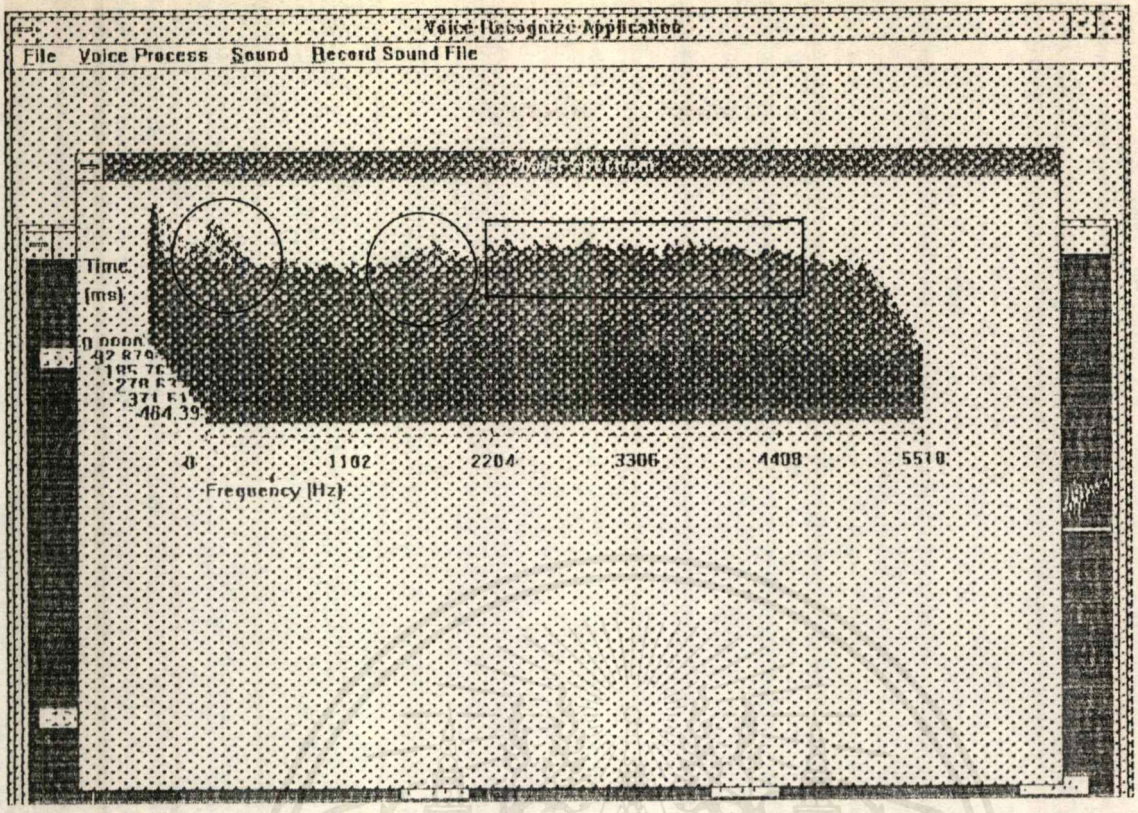
คำว่า "สี" พูดโดยบุคคลที่ 1 ครั้งที่ 3



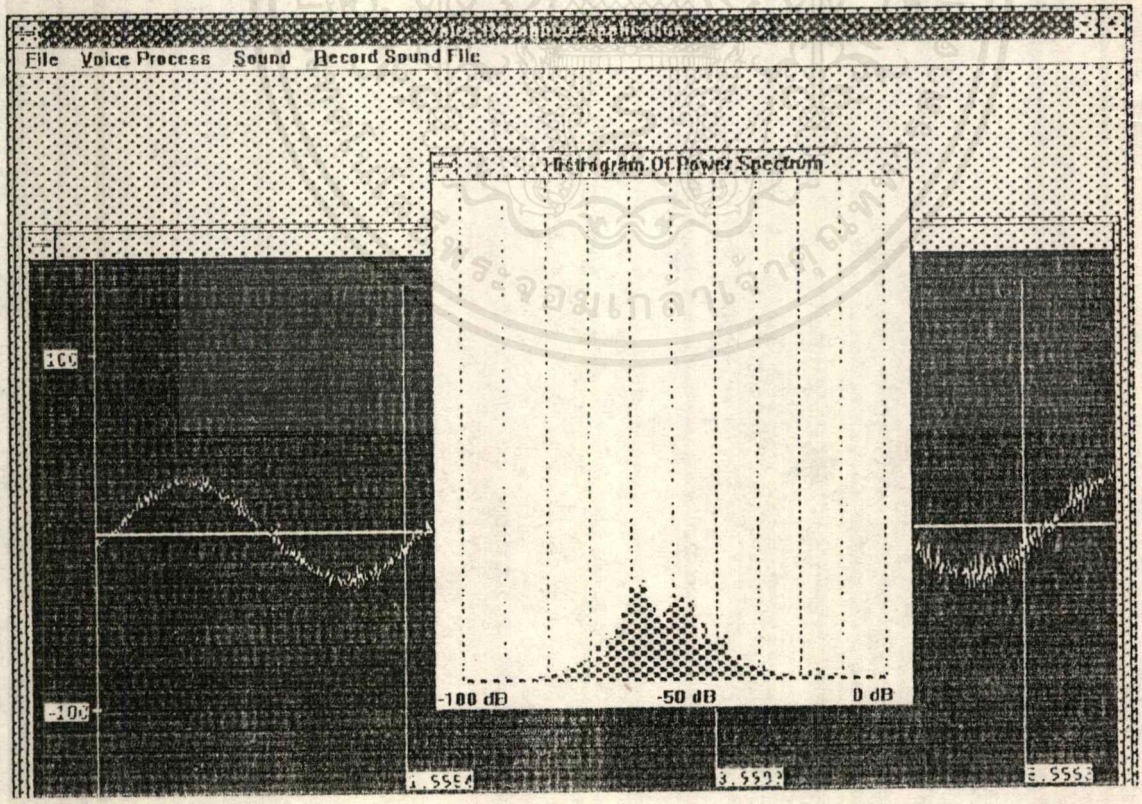


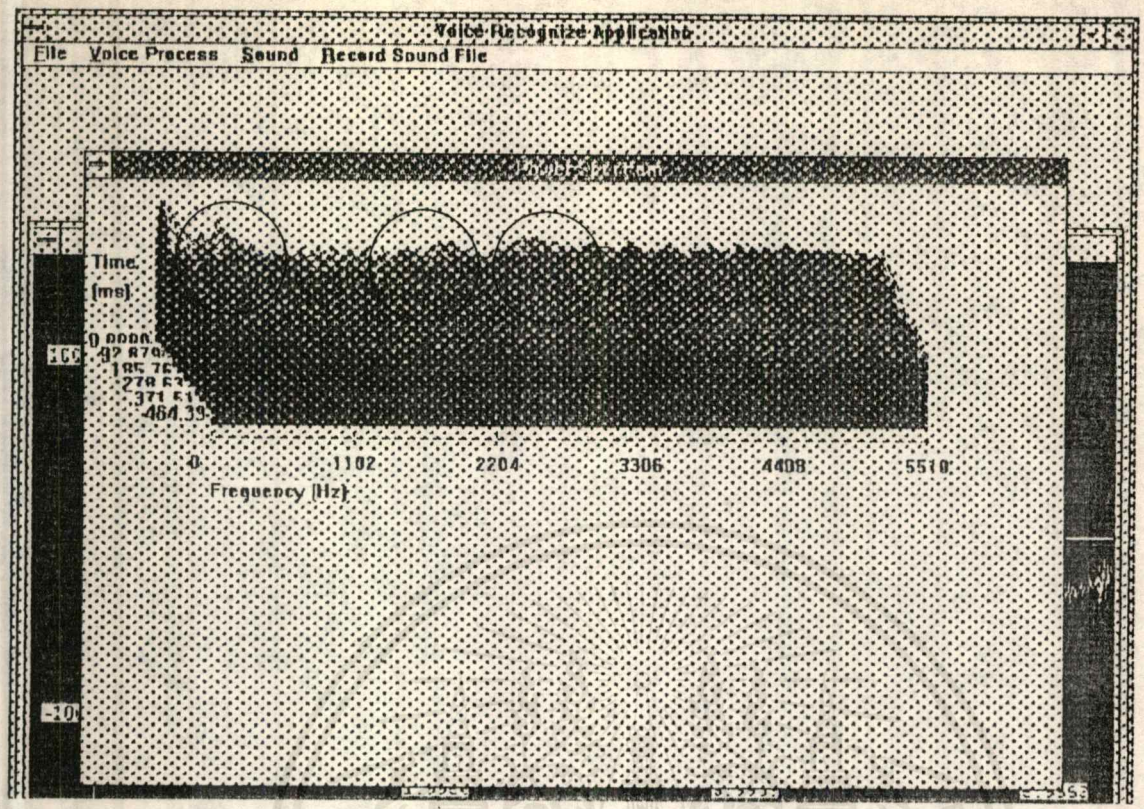
คำว่า "สี่" พูดโดยบุคคลที่ 2 ครั้งที่ 1



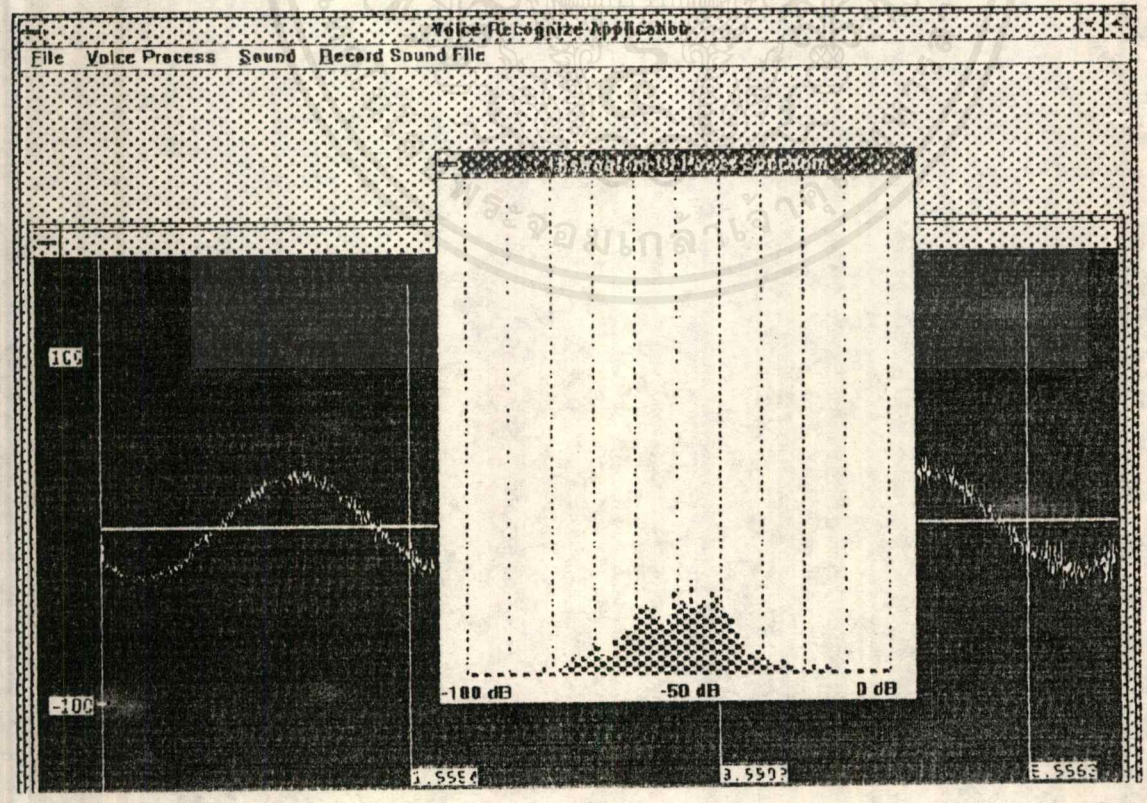


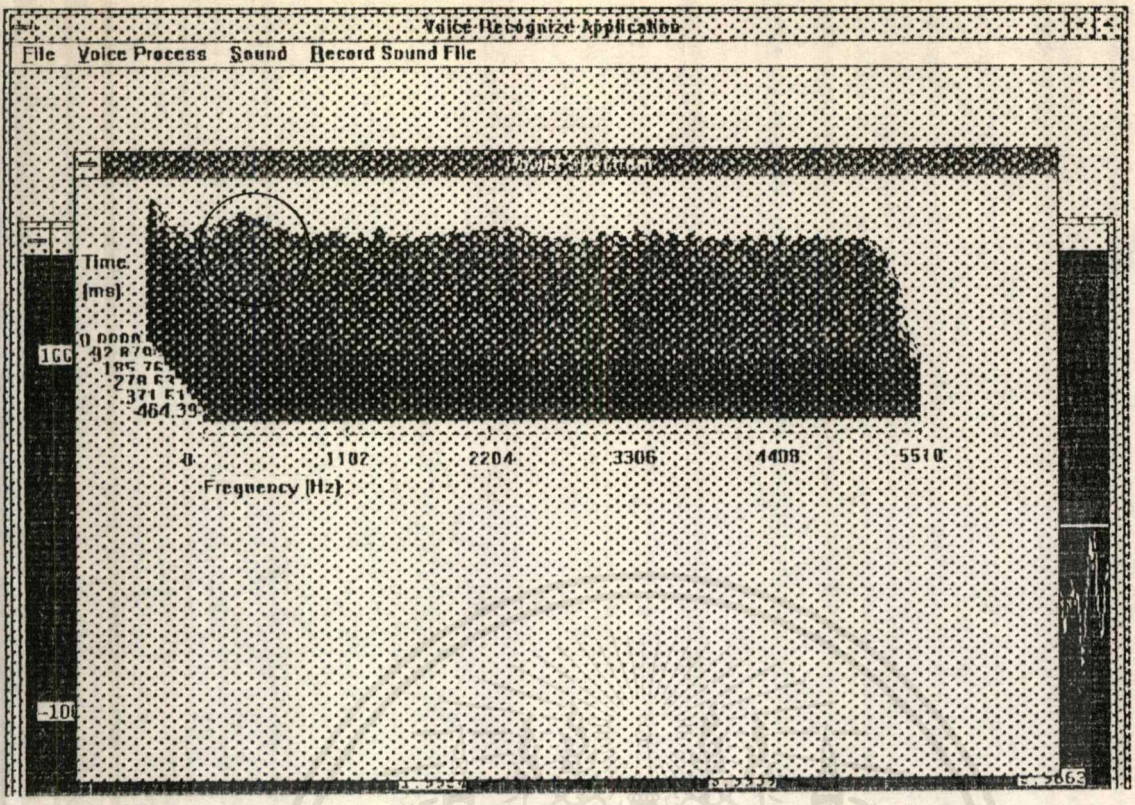
คำว่า "สี่" พูดโดยบุคคลที่ 2 ครั้งที่ 2



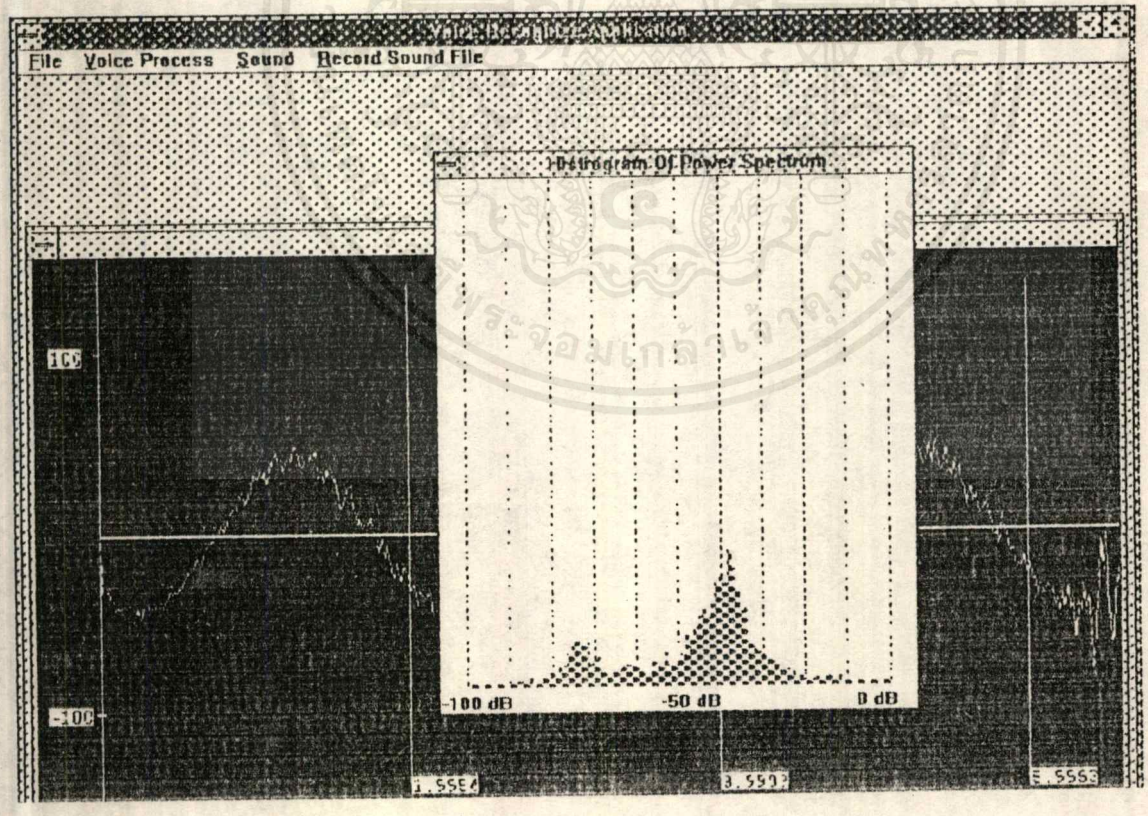


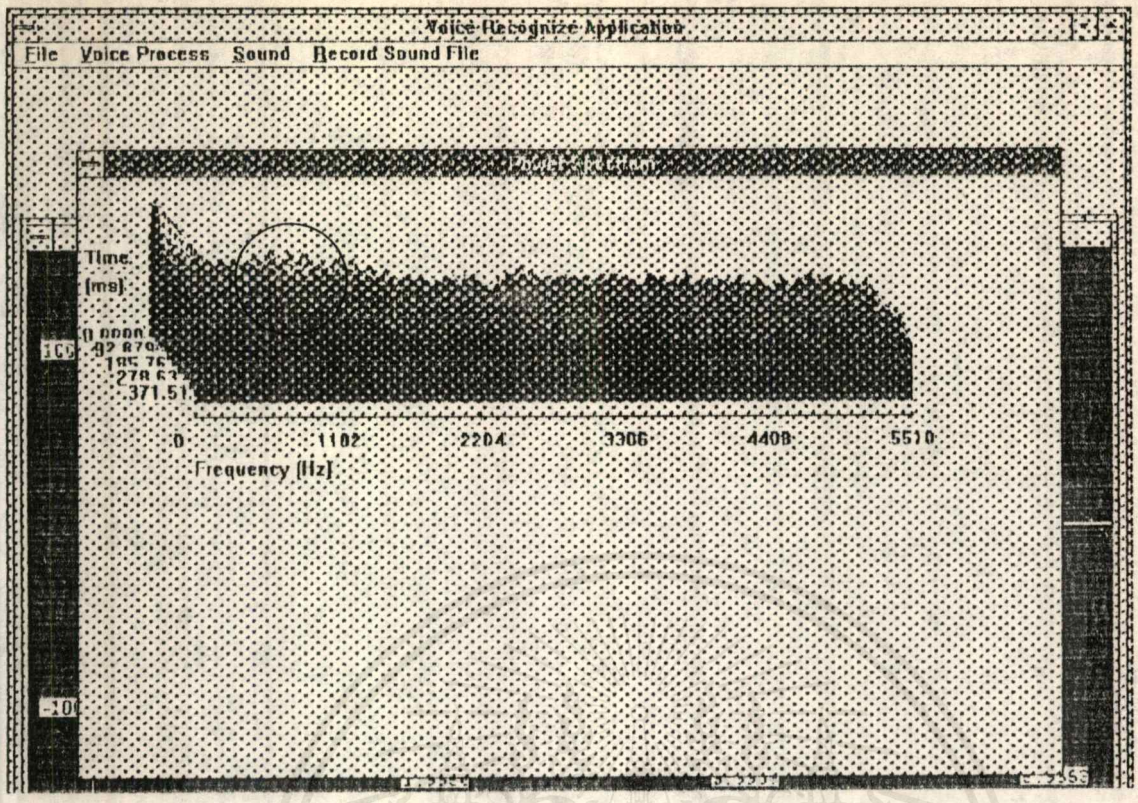
คำว่า "สี่" พูดโดยบุคคลที่ 2 ครั้งที่ 3



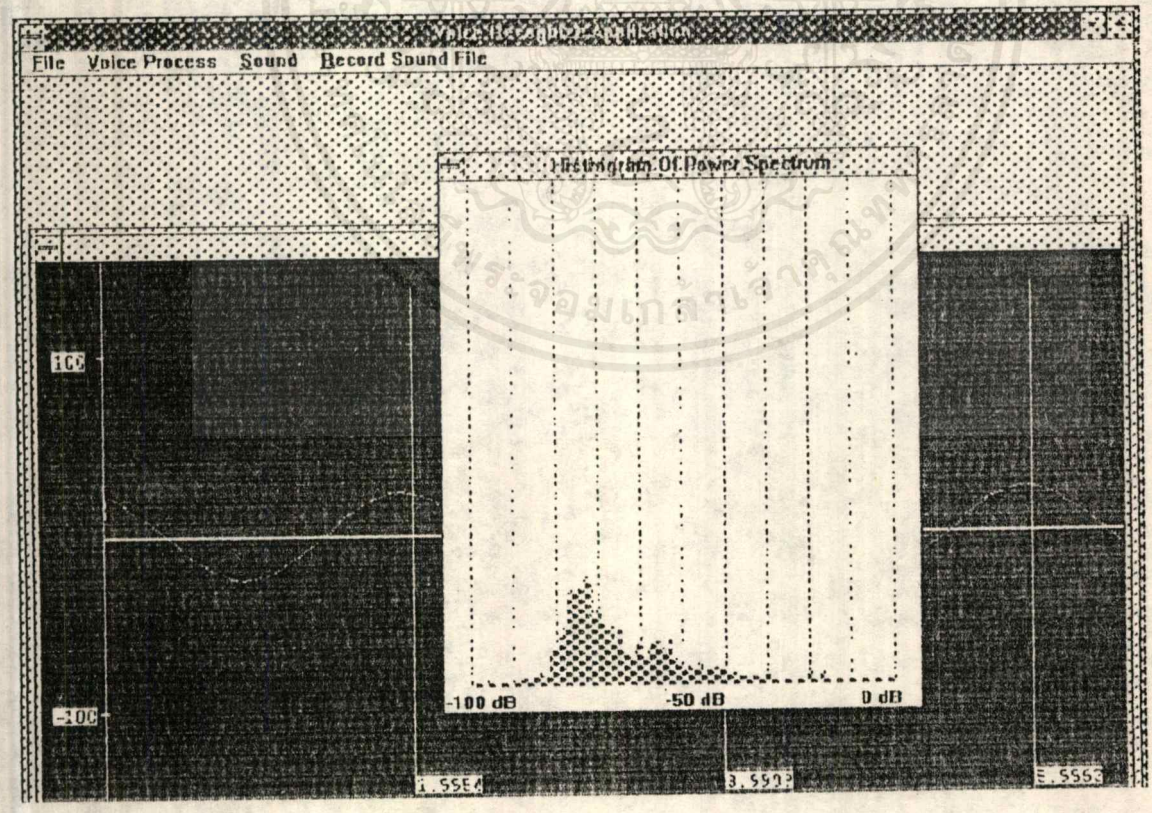


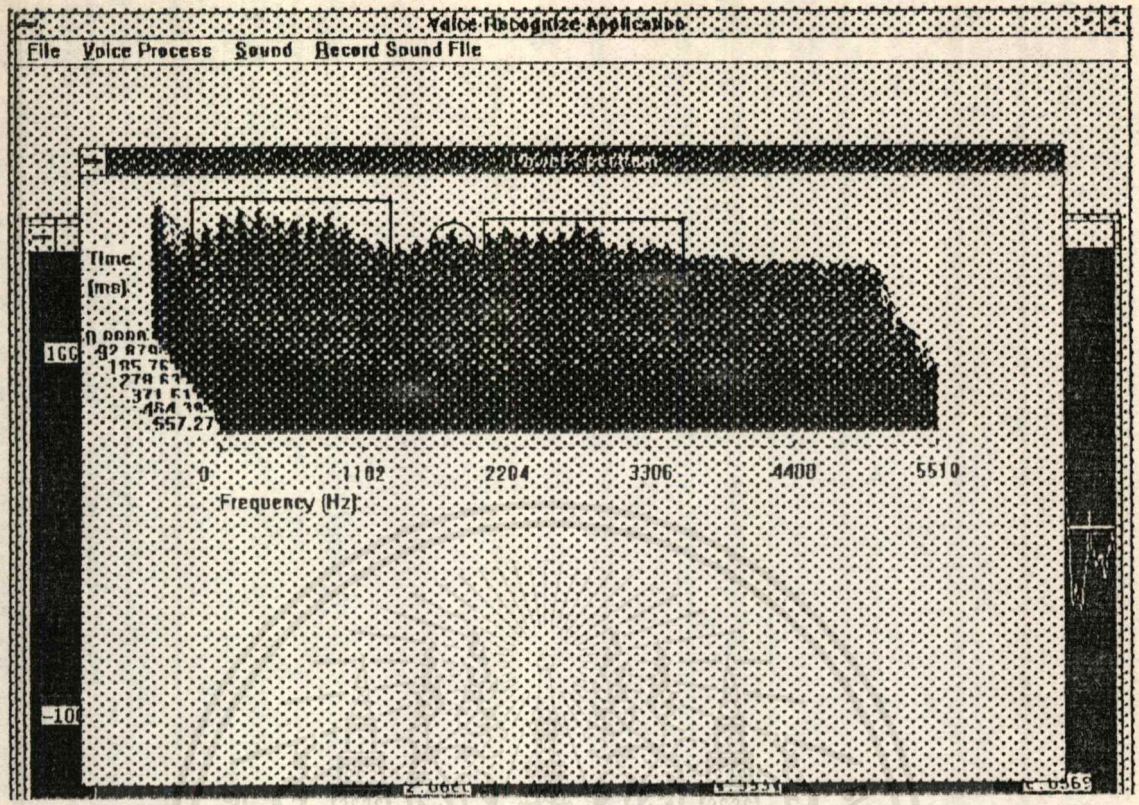
คำว่า "ห้า" พูดโดยบุคคลที่ 1 ครั้งที่ 1



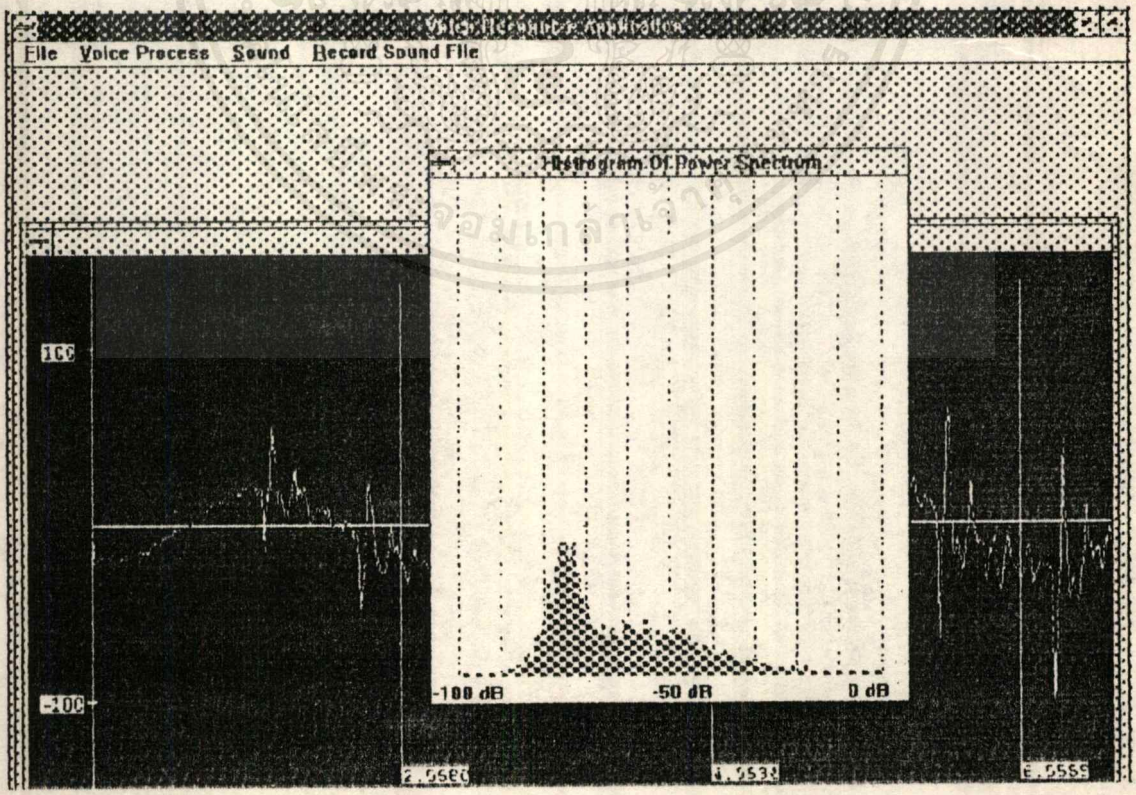


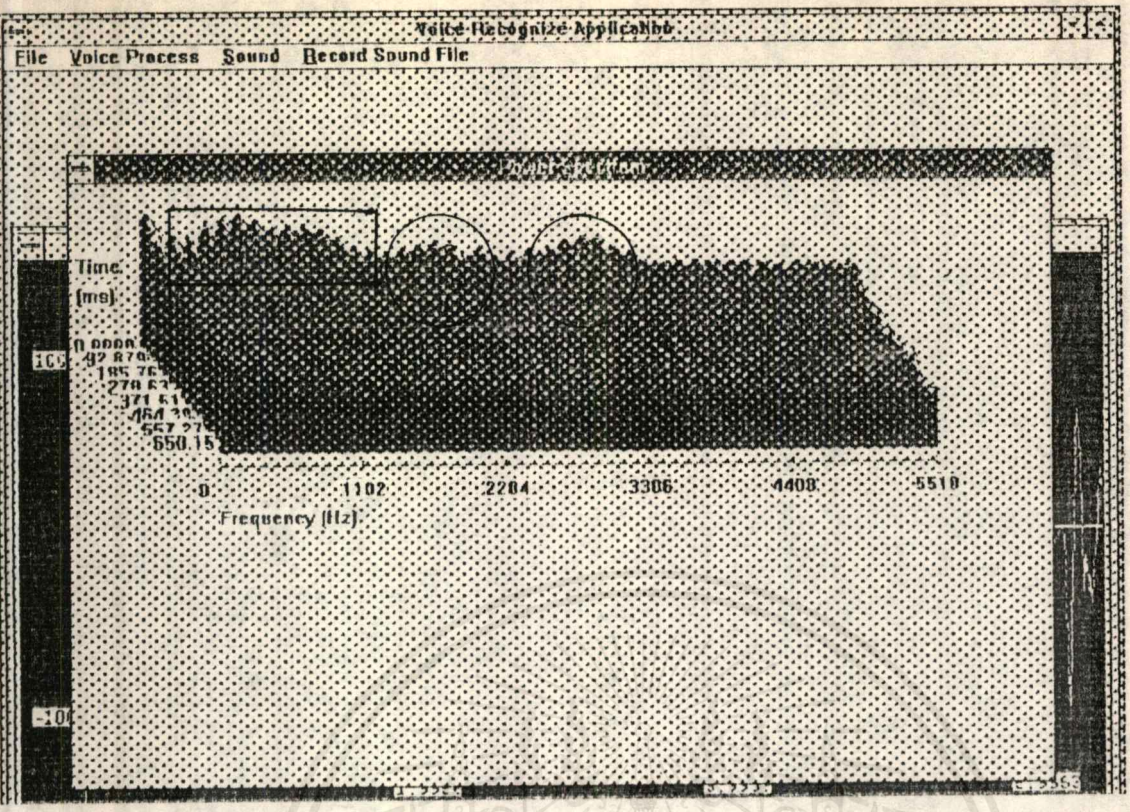
คำว่า "ห้า" พูดโดยบุคคลที่ 1 ครั้งที่ 2



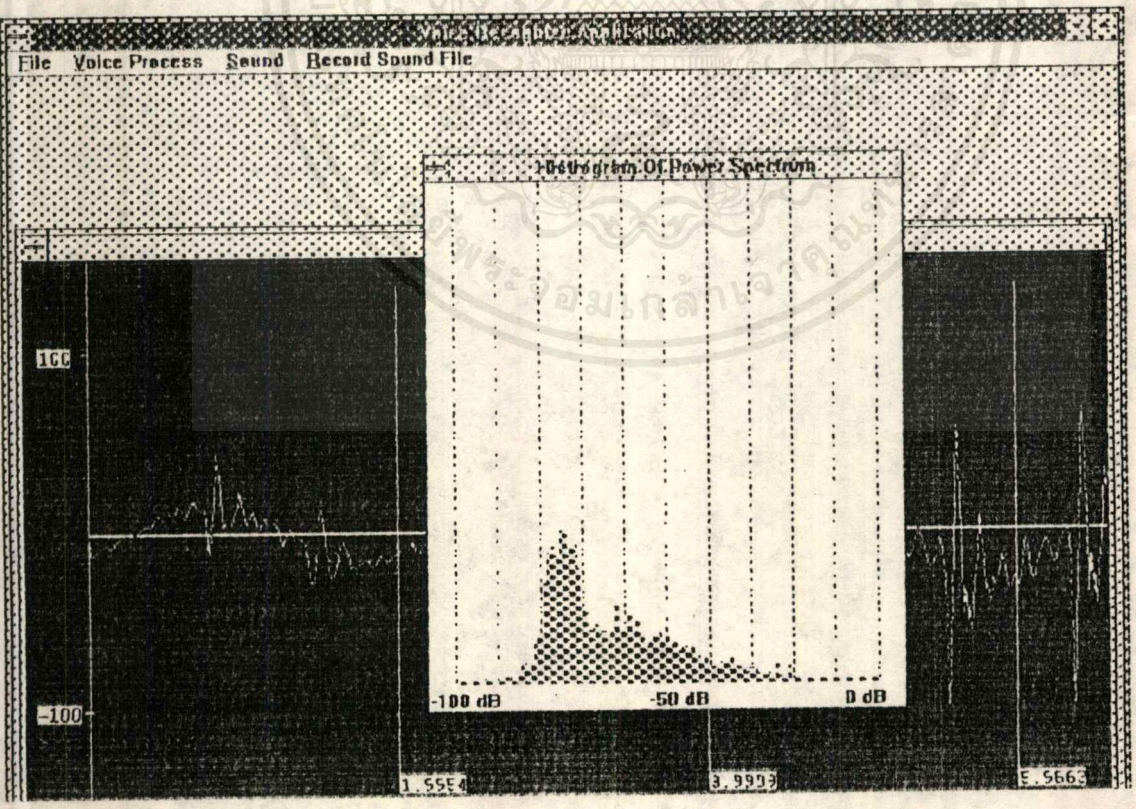


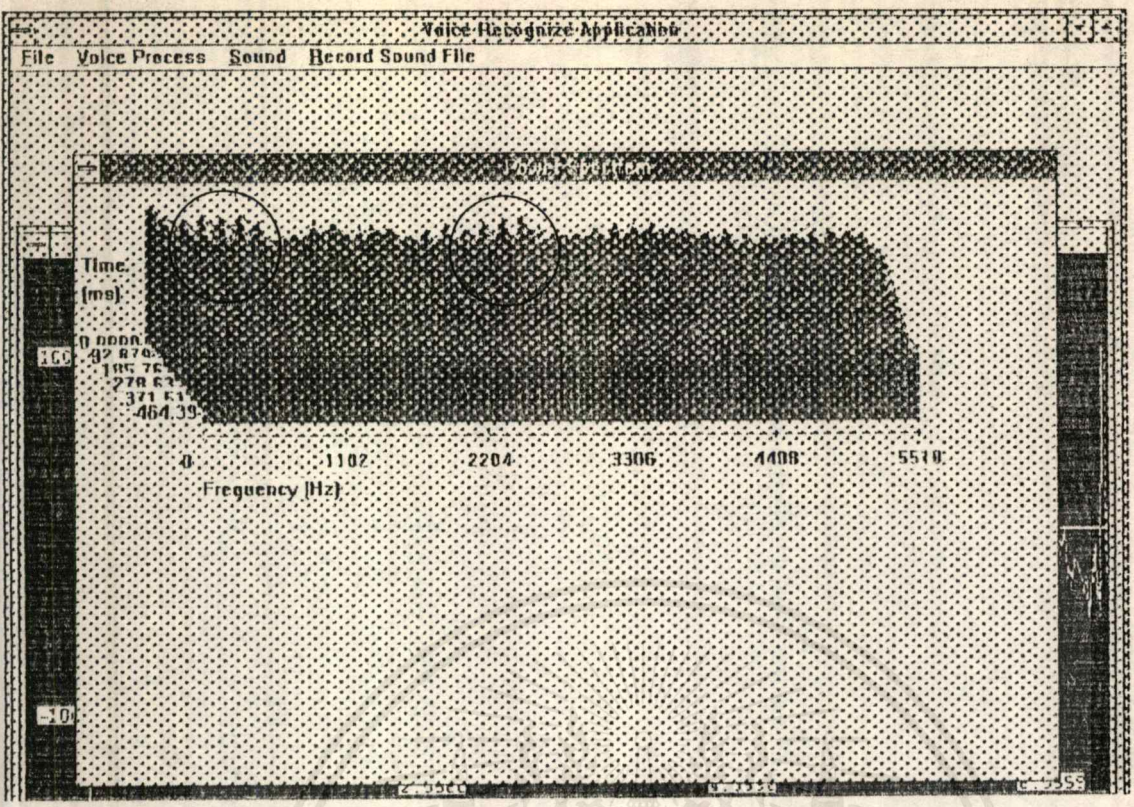
คำว่า "ห้า" พูดโดยบุคคลที่ 1 ครั้งที่ 3



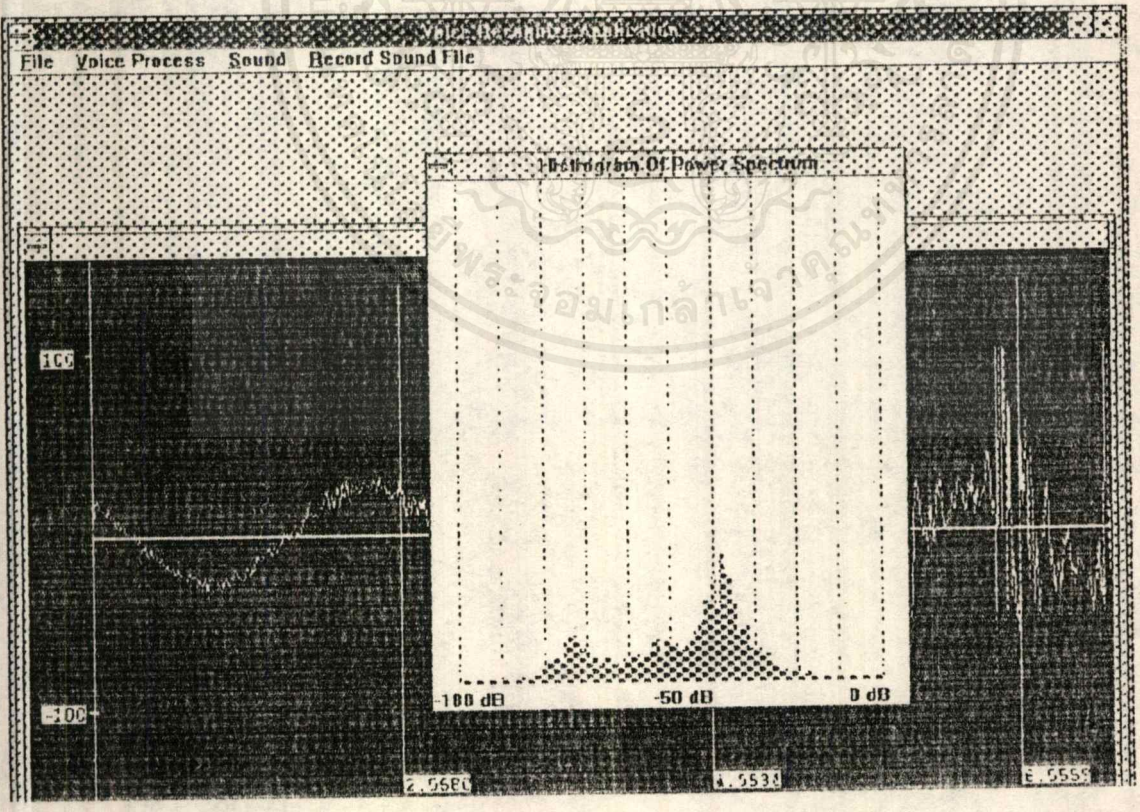


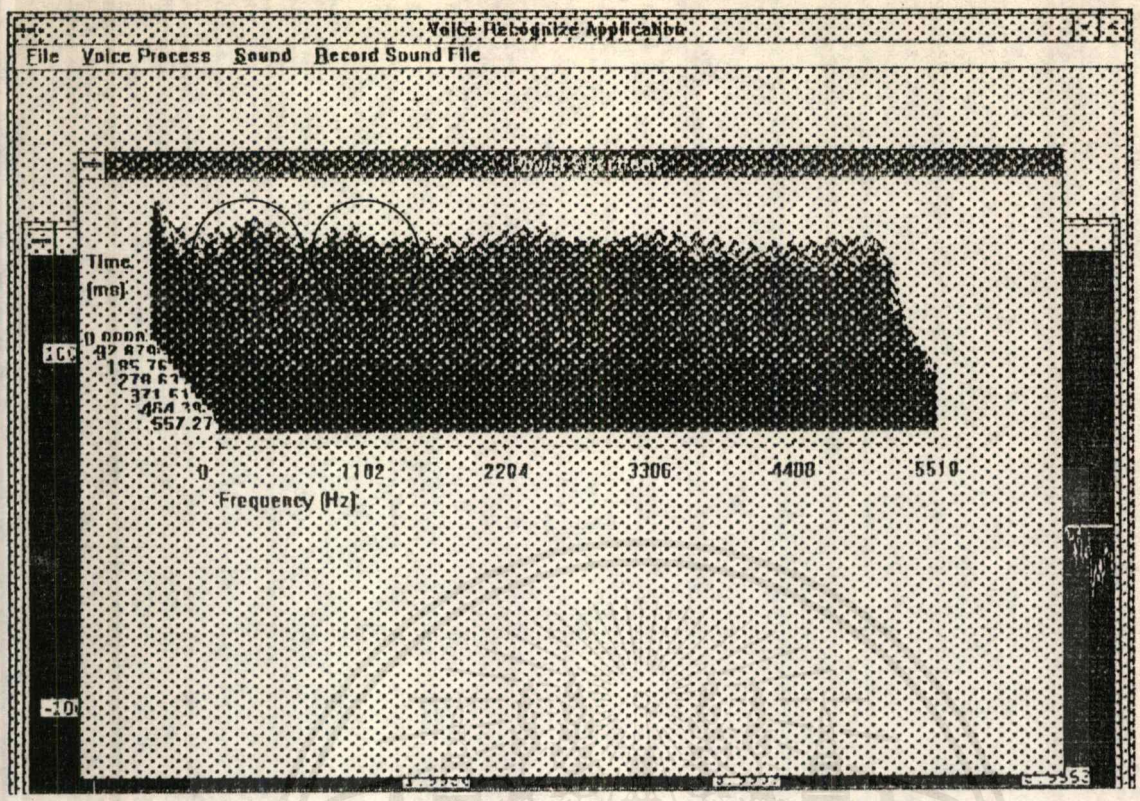
คำว่า "ห้า" พูดโดยบุคคลที่ 2 ครั้งที่ 1



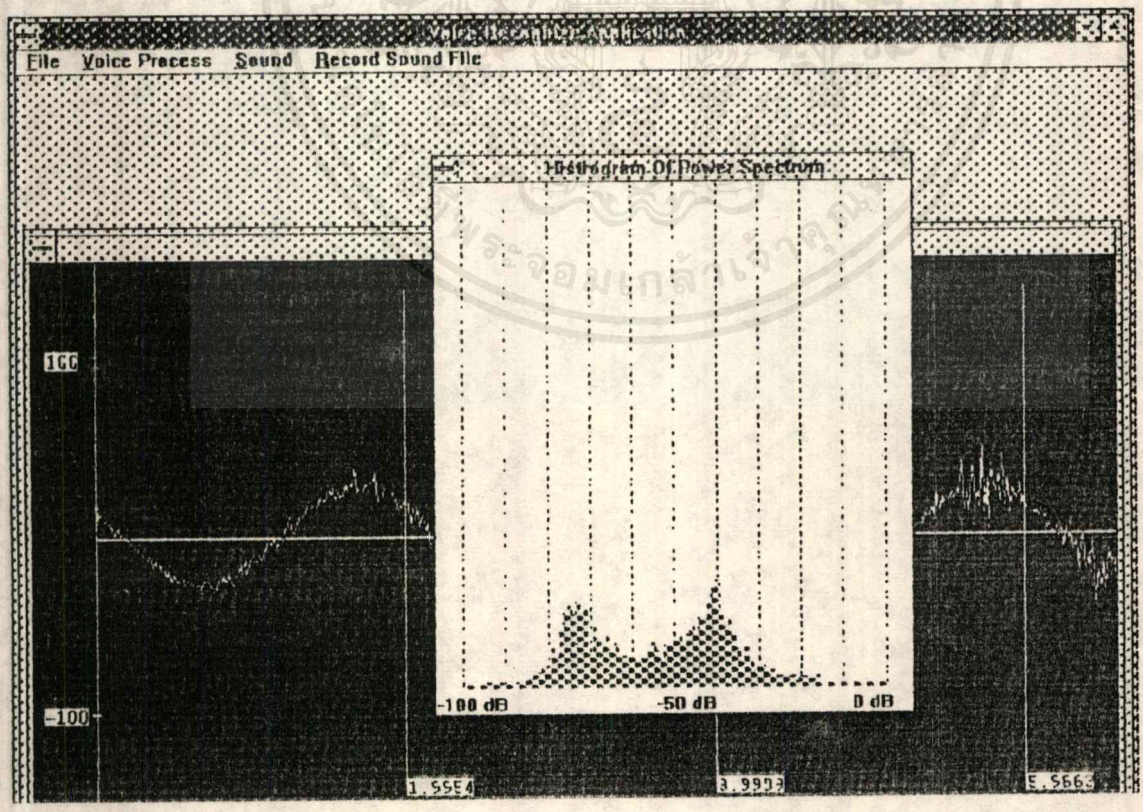


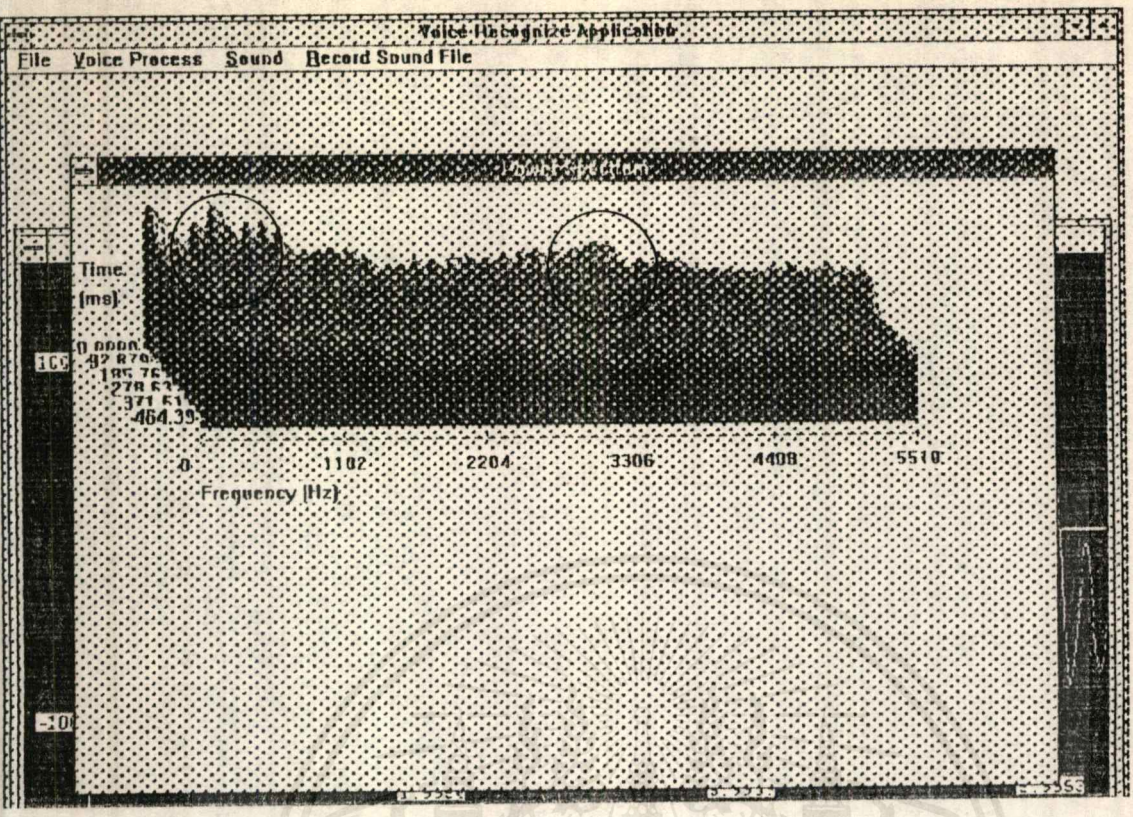
คำว่า "ห้า" พูดโดยบุคคลที่ 2 ครั้งที่ 2



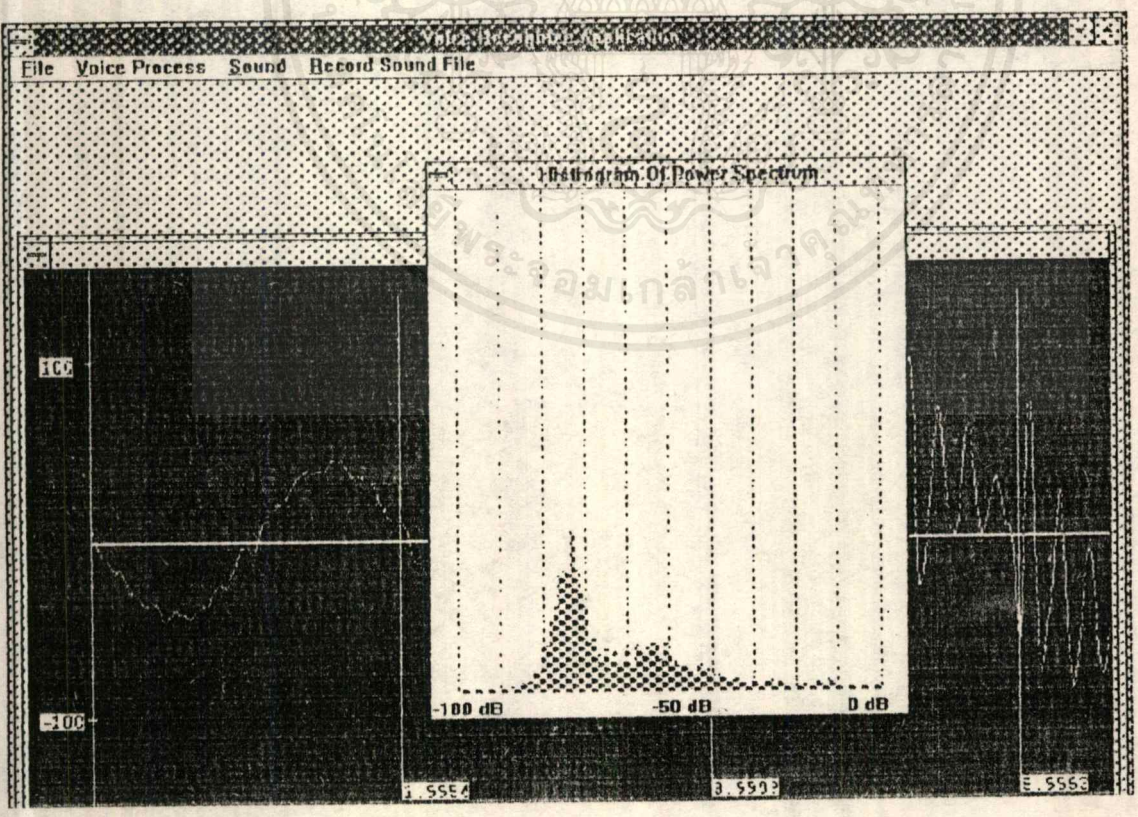


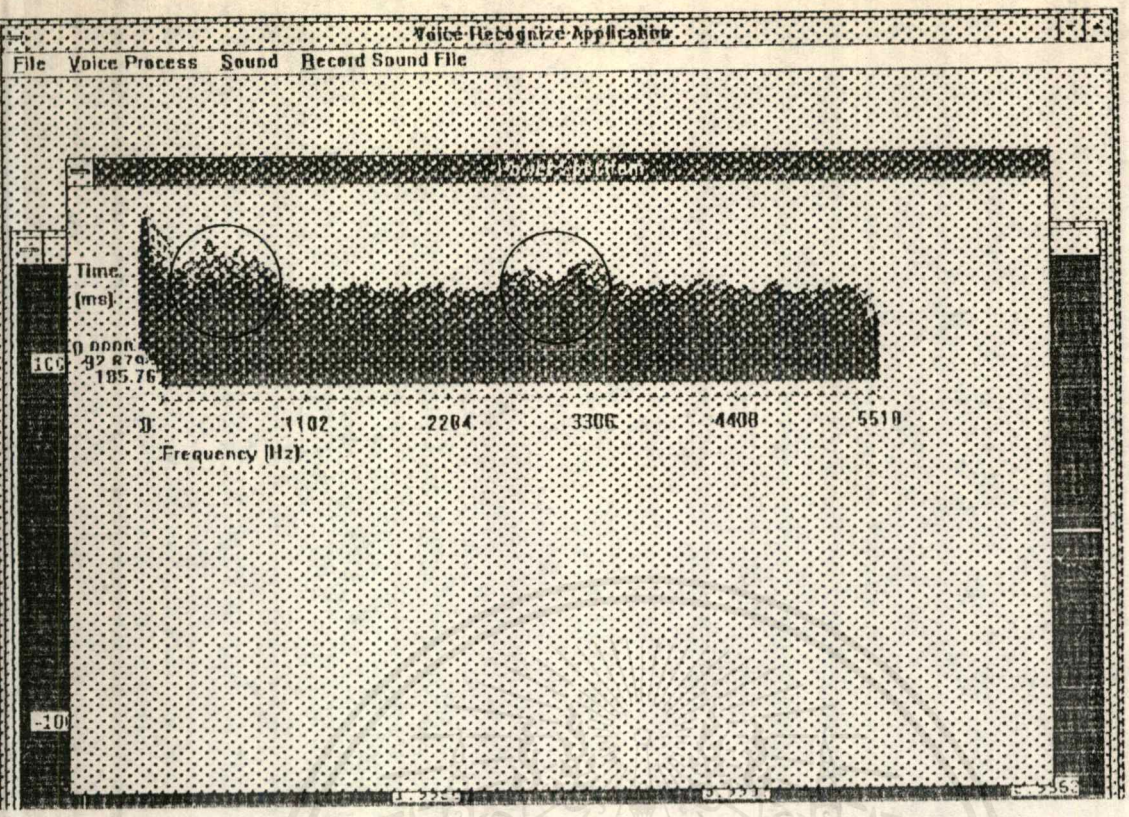
คำว่า "ห้า" พูดโดยบุคคลที่ 2 ครั้งที่ 3



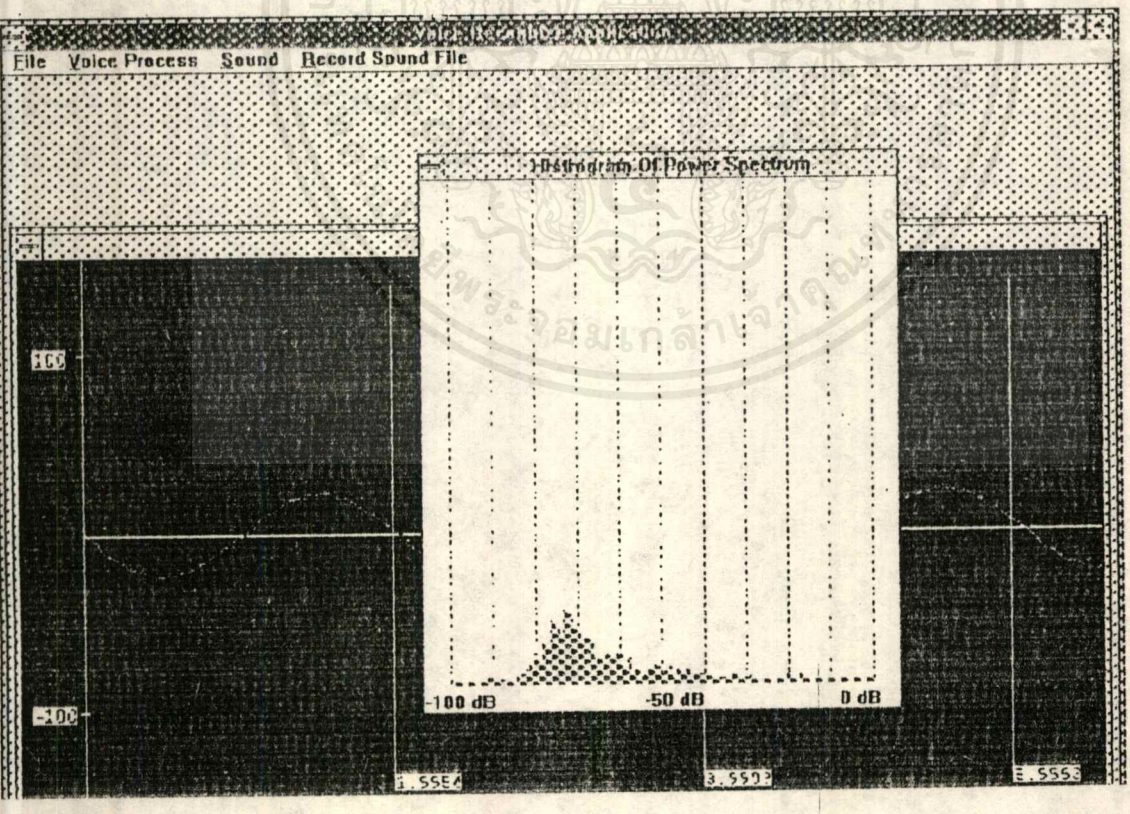


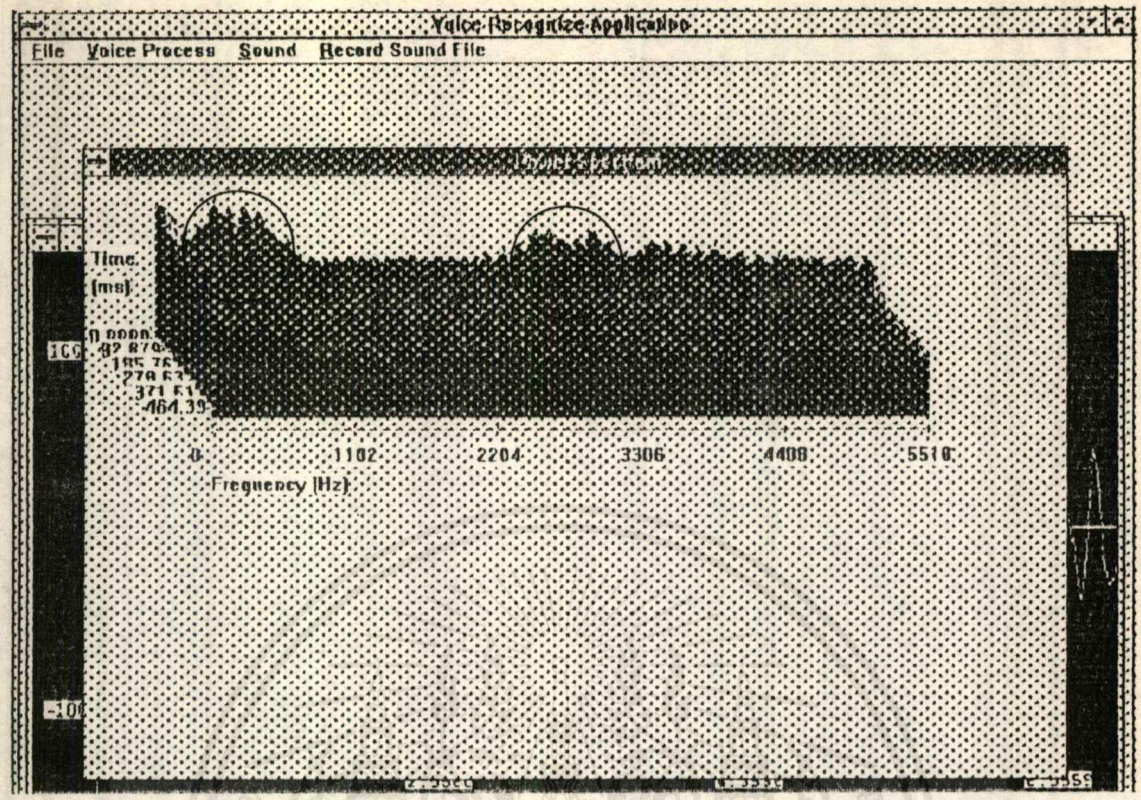
คำว่า "หก" พูดโดยบุคคลที่ 1 ครั้งที่ 1



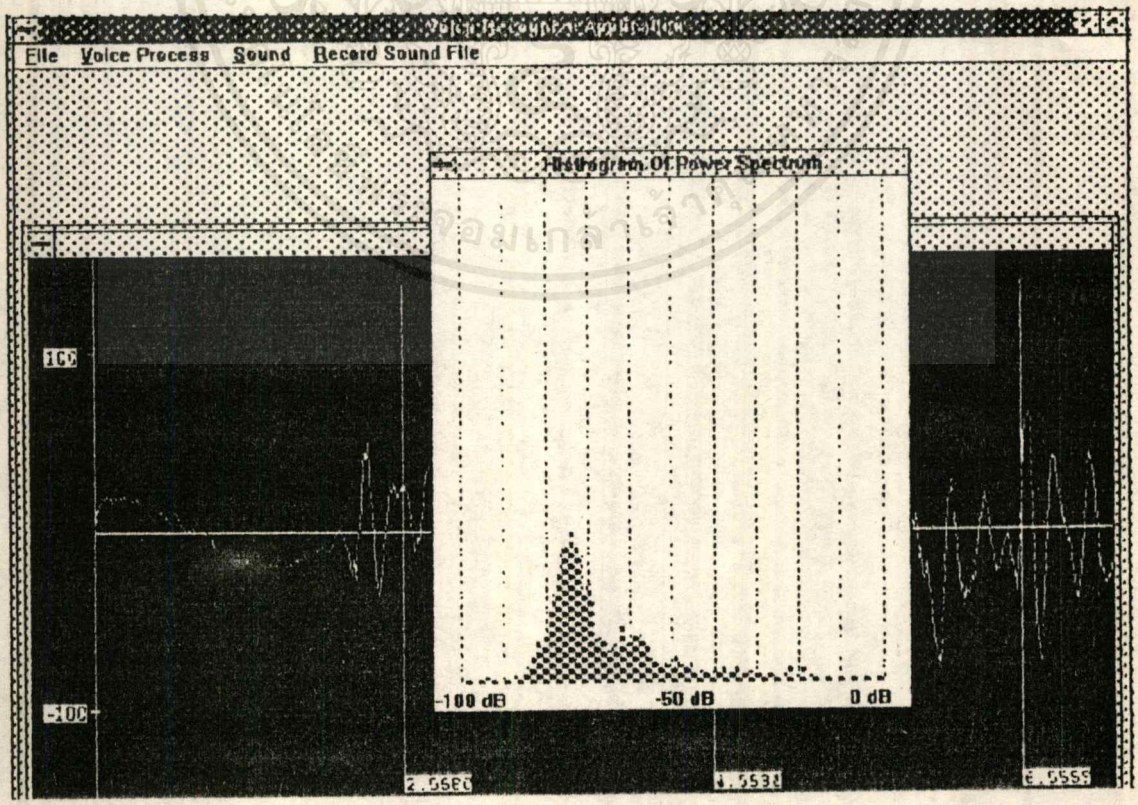


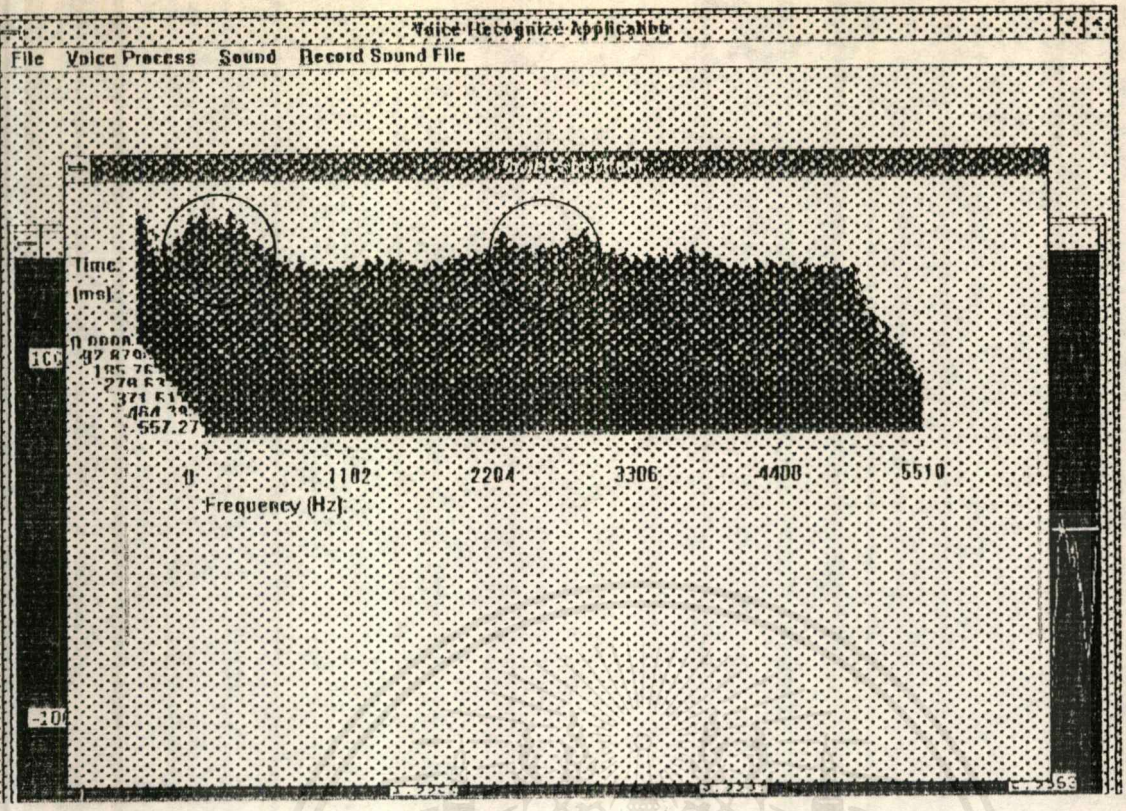
คำว่า "หก" พูดโดยบุคคลที่ 1 ครั้งที่ 2



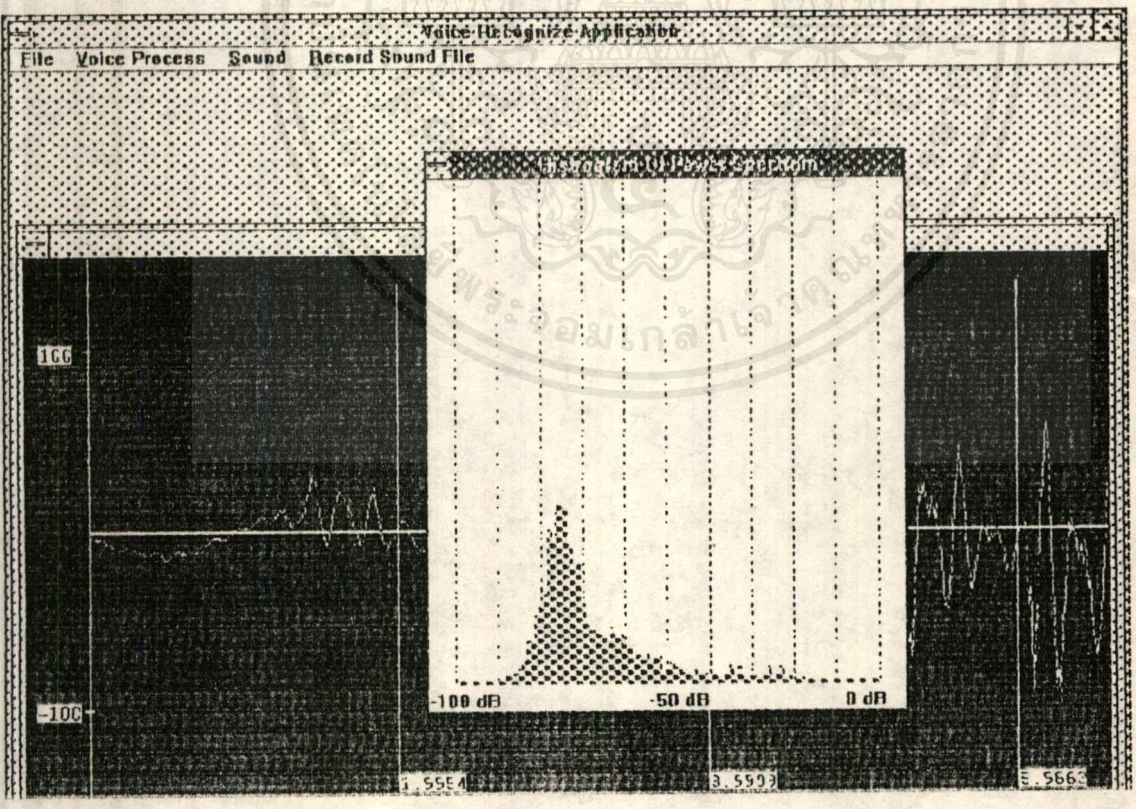


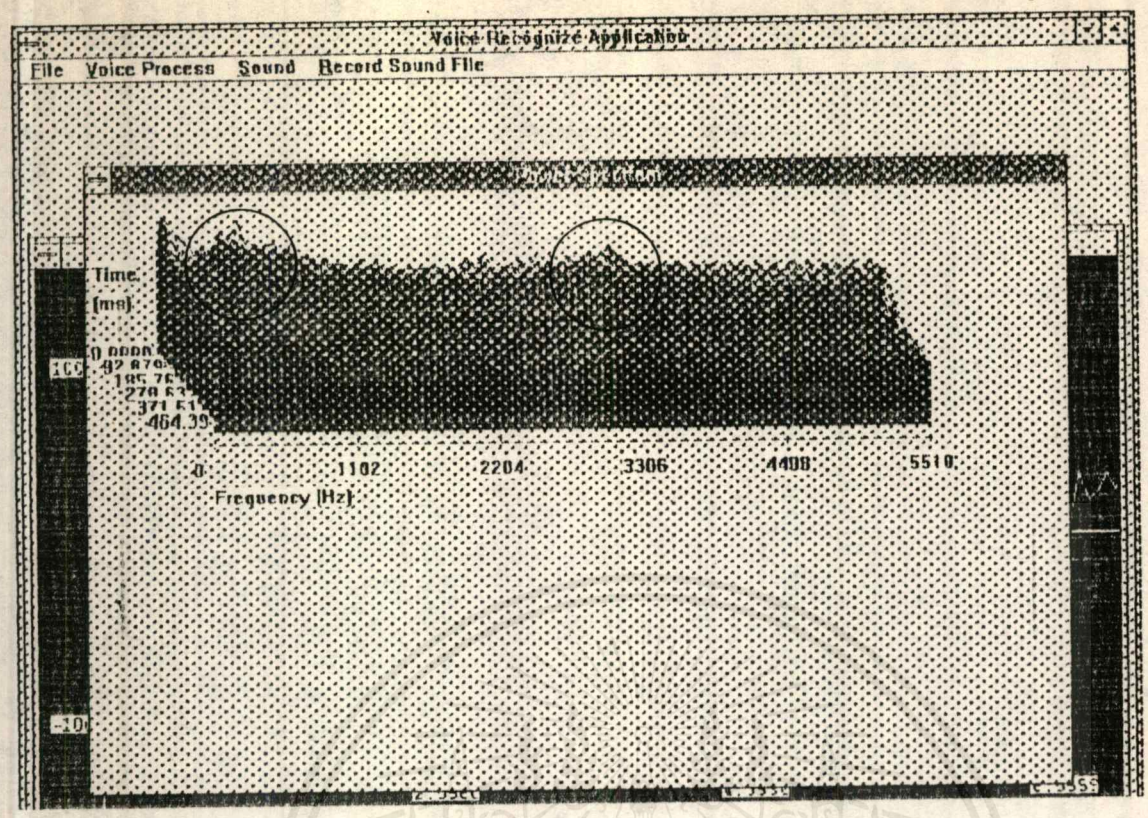
คำว่า "หก" พูดโดยบุคคลที่ 1 ครั้งที่ 3



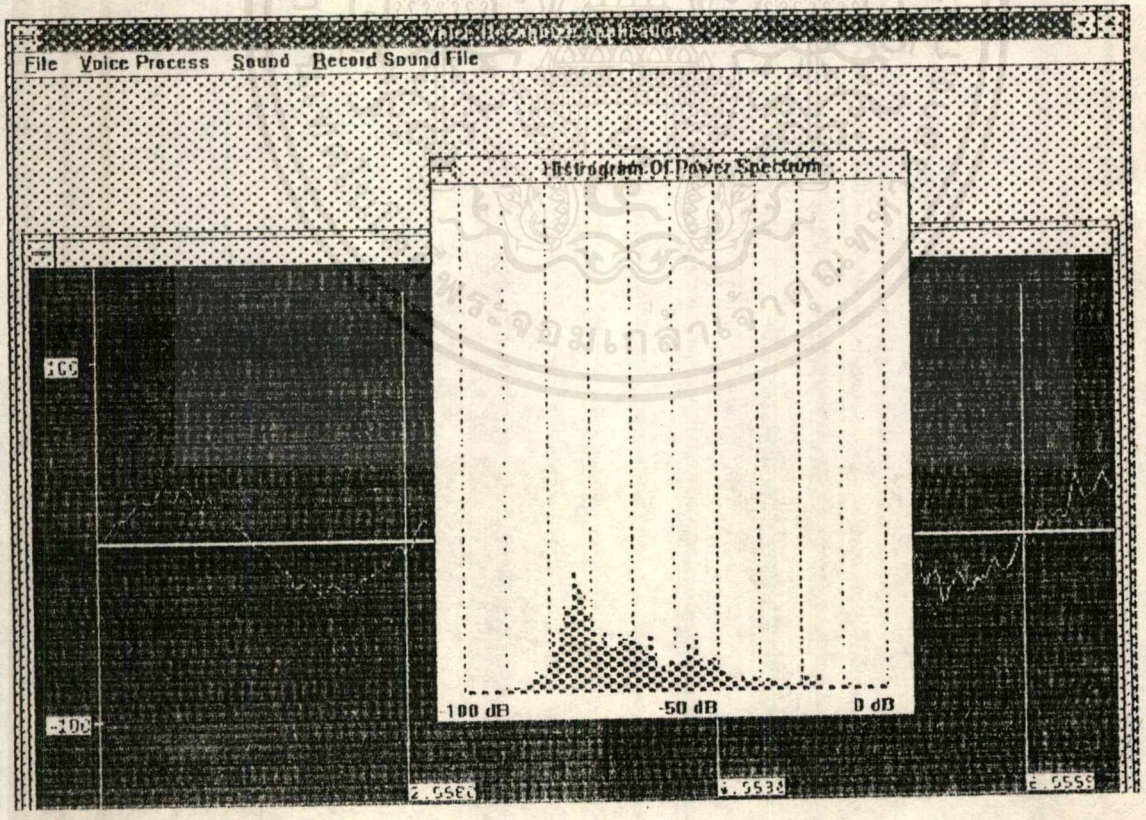


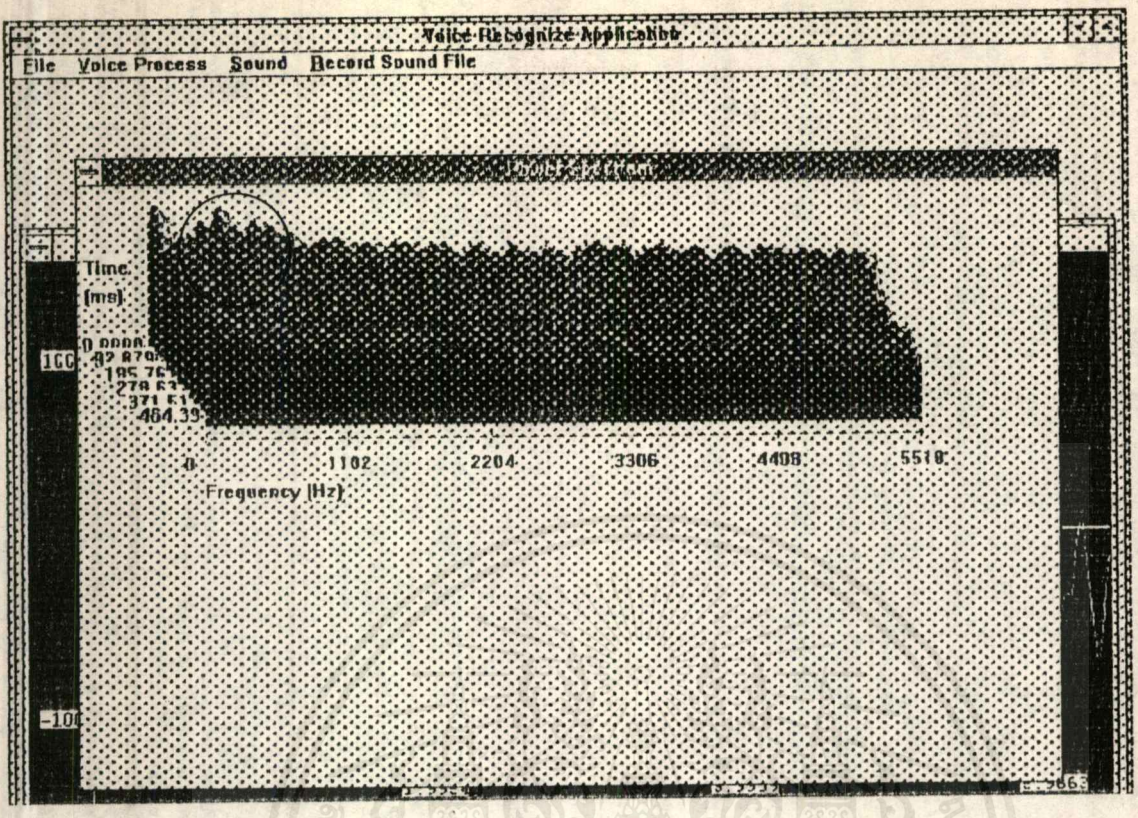
คำว่า "หก" พูดโดยบุคคลที่ 2 ครั้งที่ 1



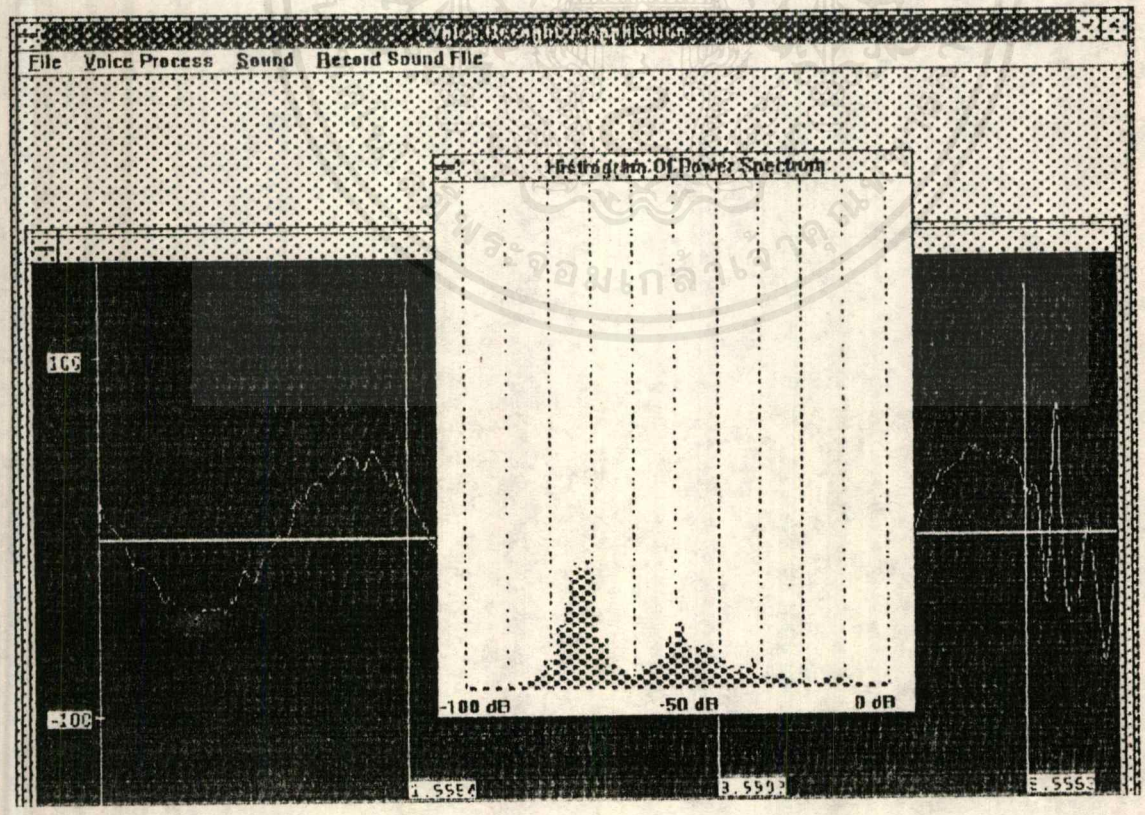


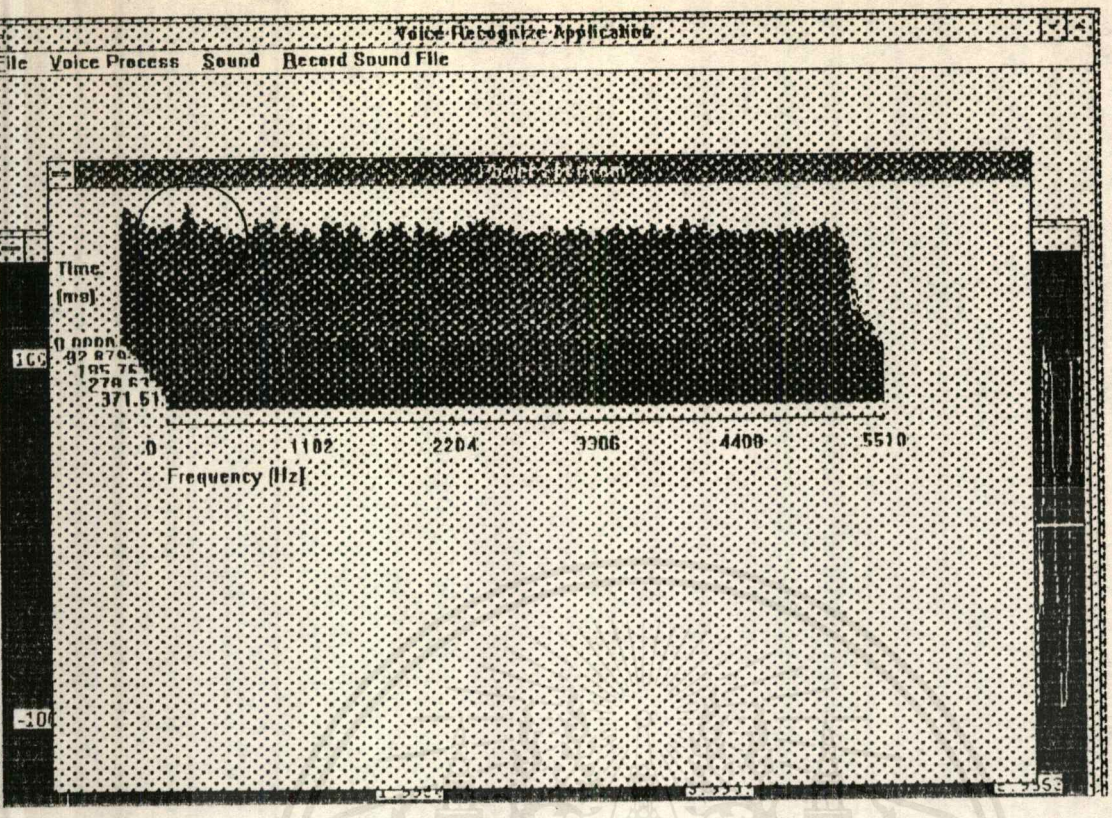
คำว่า "หก" พูดโดยบุคคลที่ 2 ครั้งที่ 2



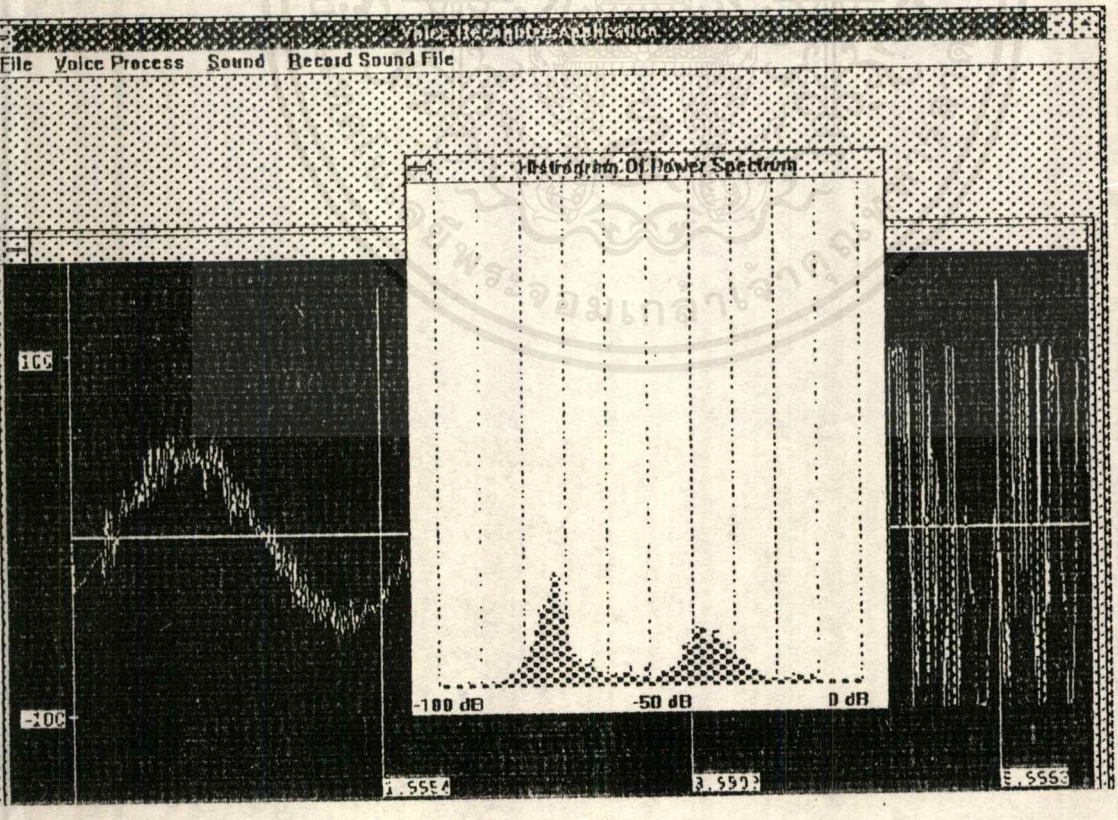


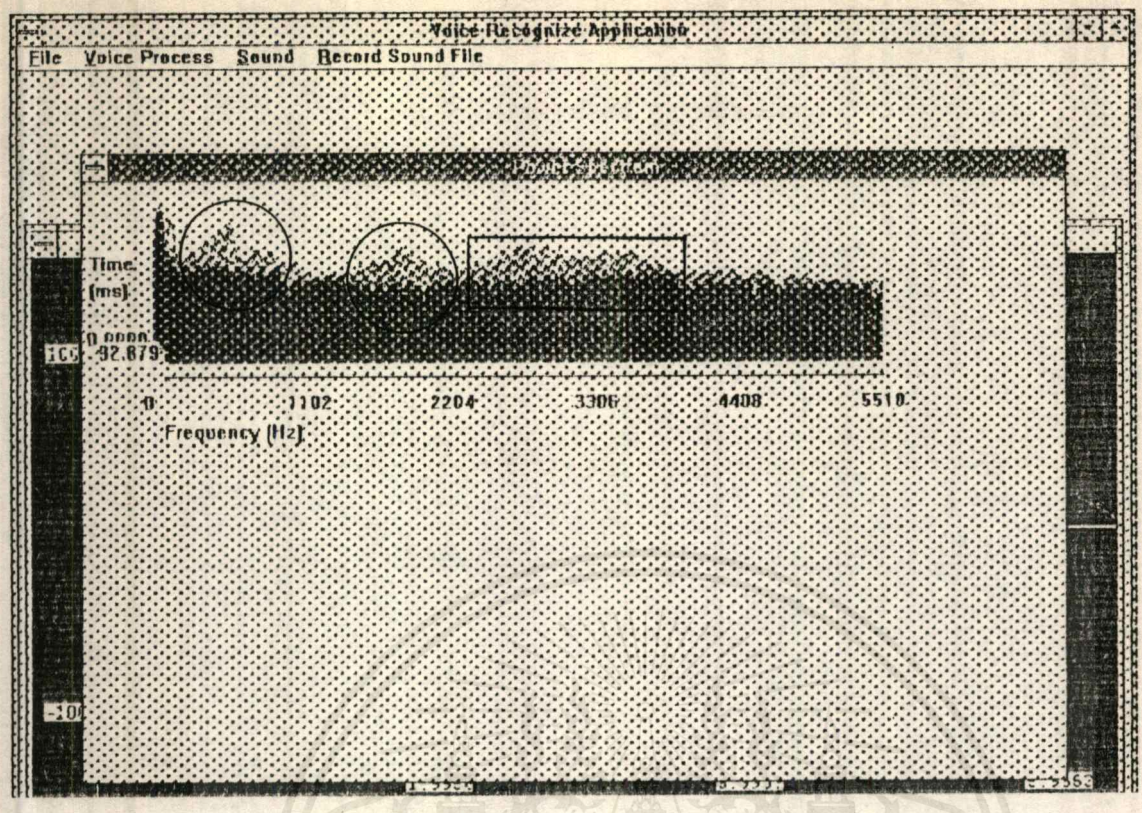
คำว่า "หก" พูดโดยบุคคลที่ 2 ครั้งที่ 3



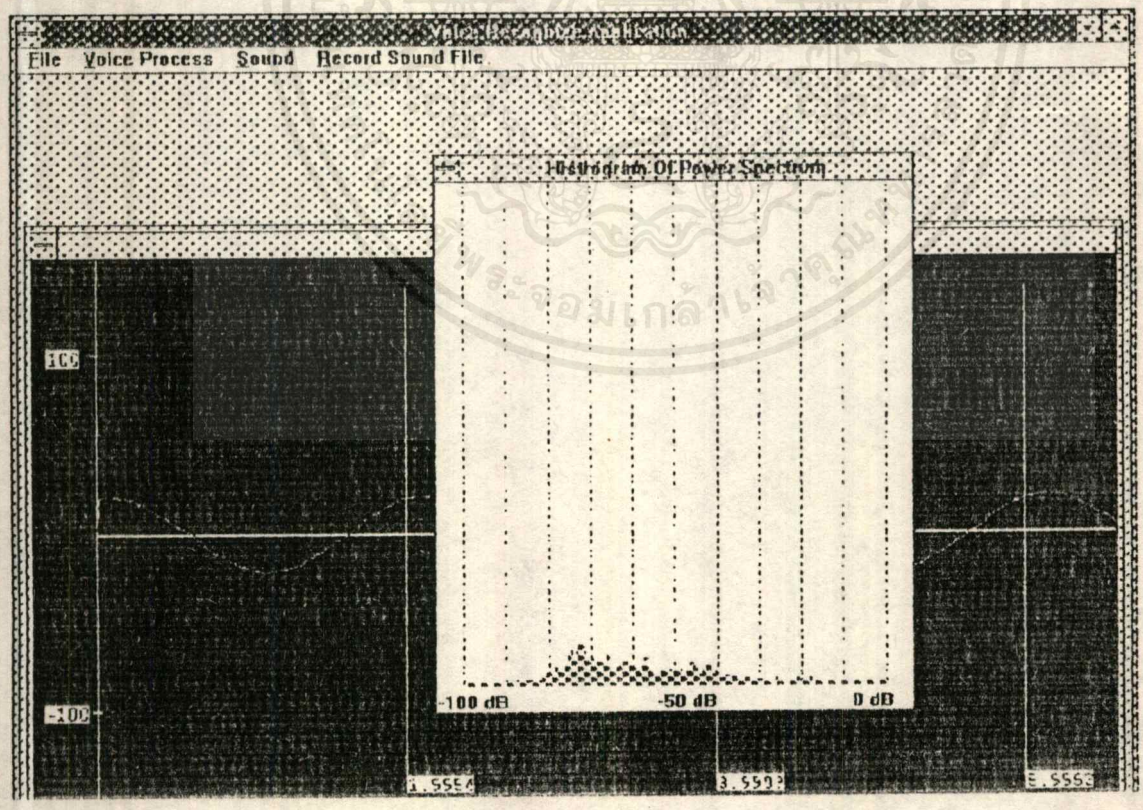


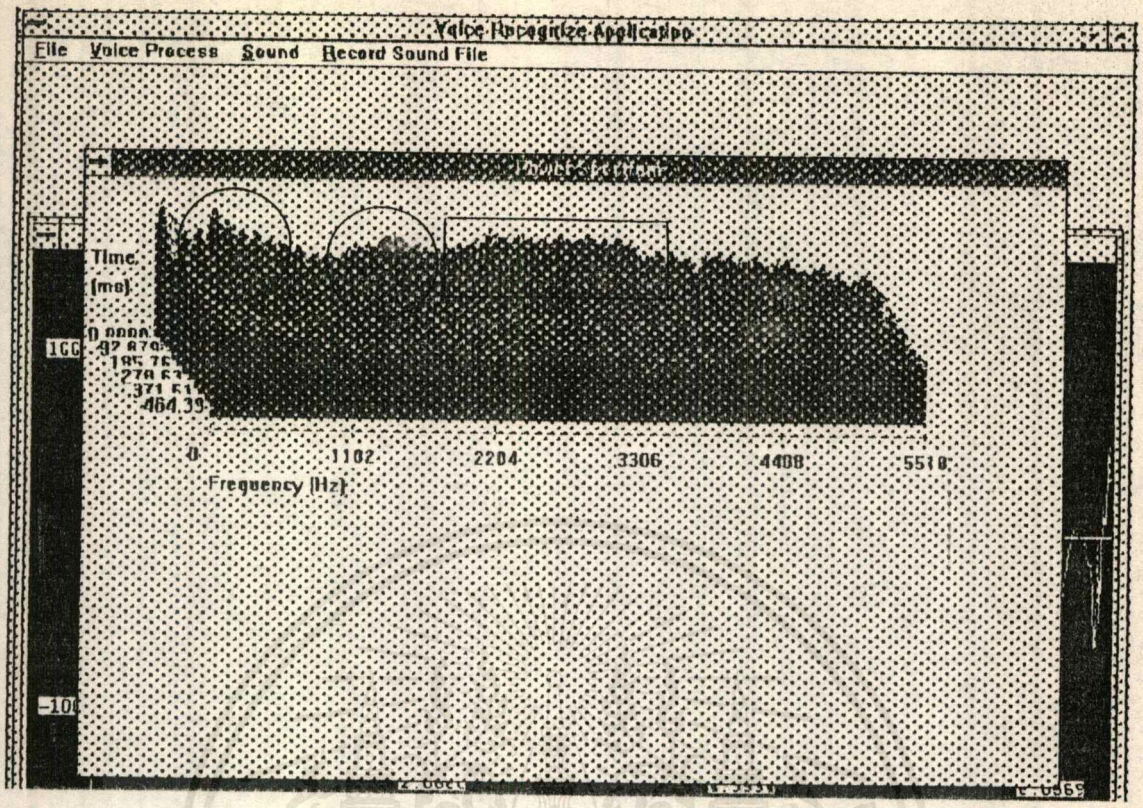
คำว่า "เจ็ด" พูดโดยบุคคลที่ 1 ครั้งที่ 1



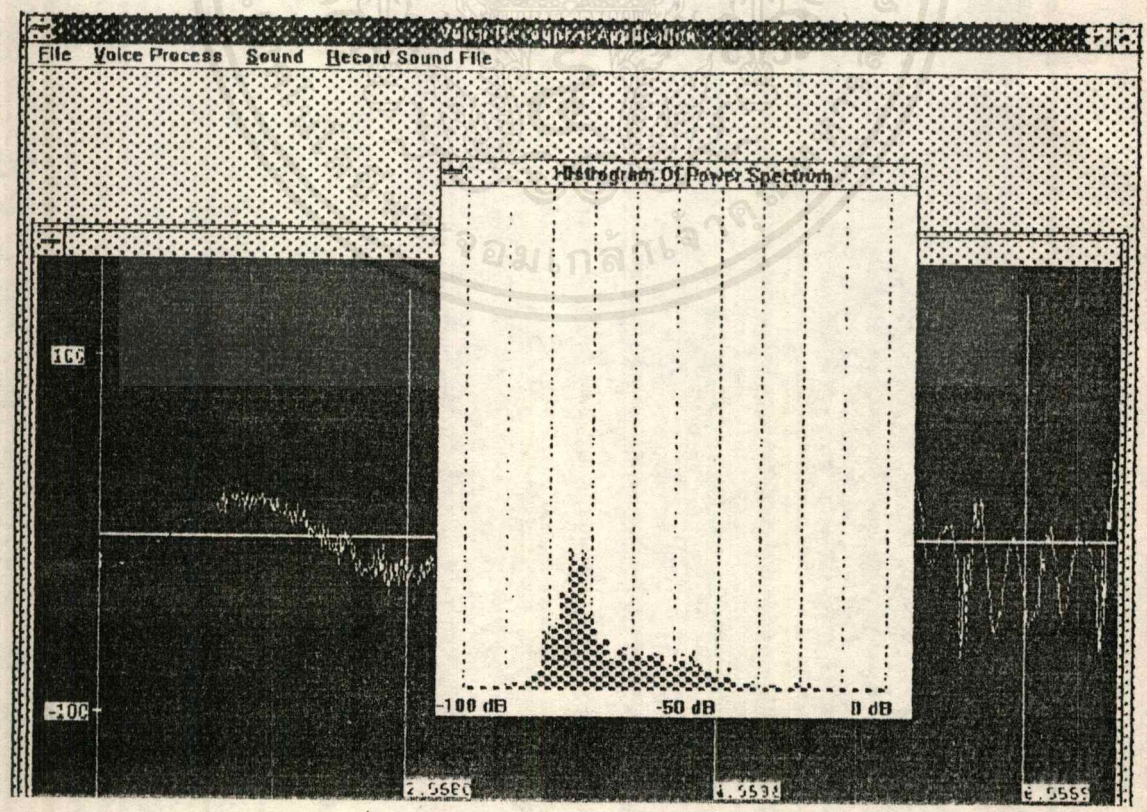


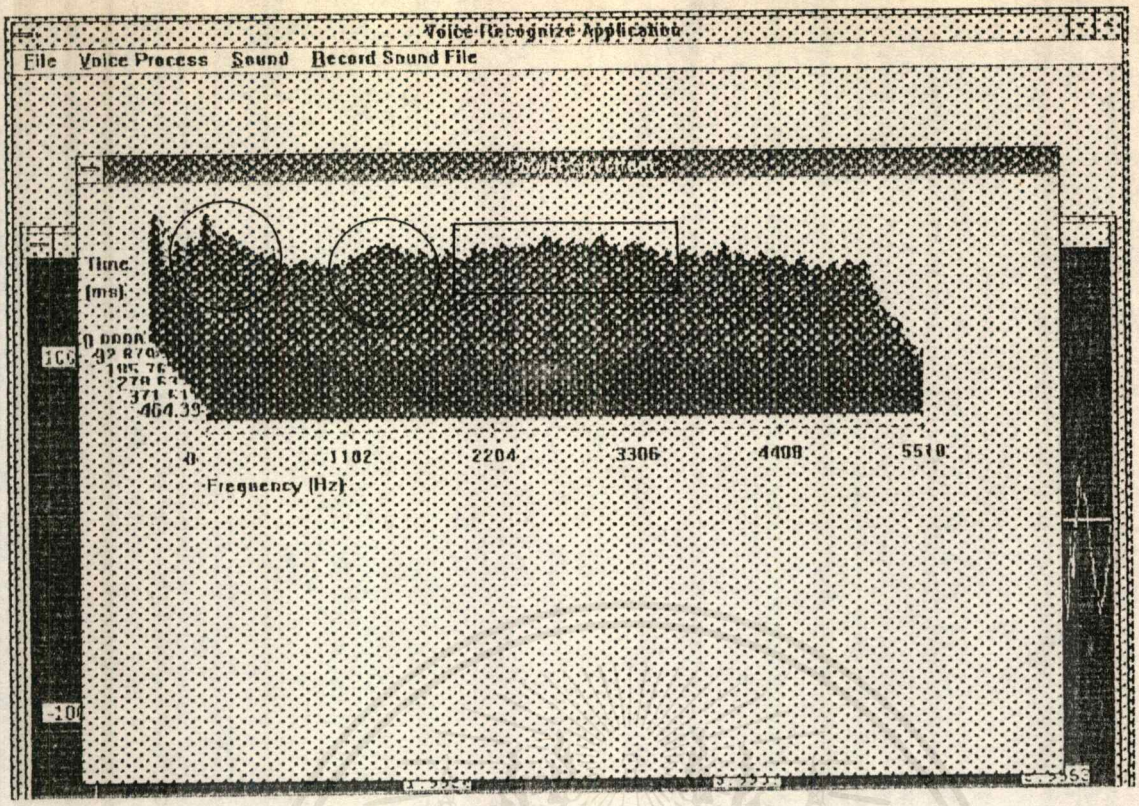
คำว่า "เจ็ด" พูดโดยบุคคลที่ 1 ครั้งที่ 2



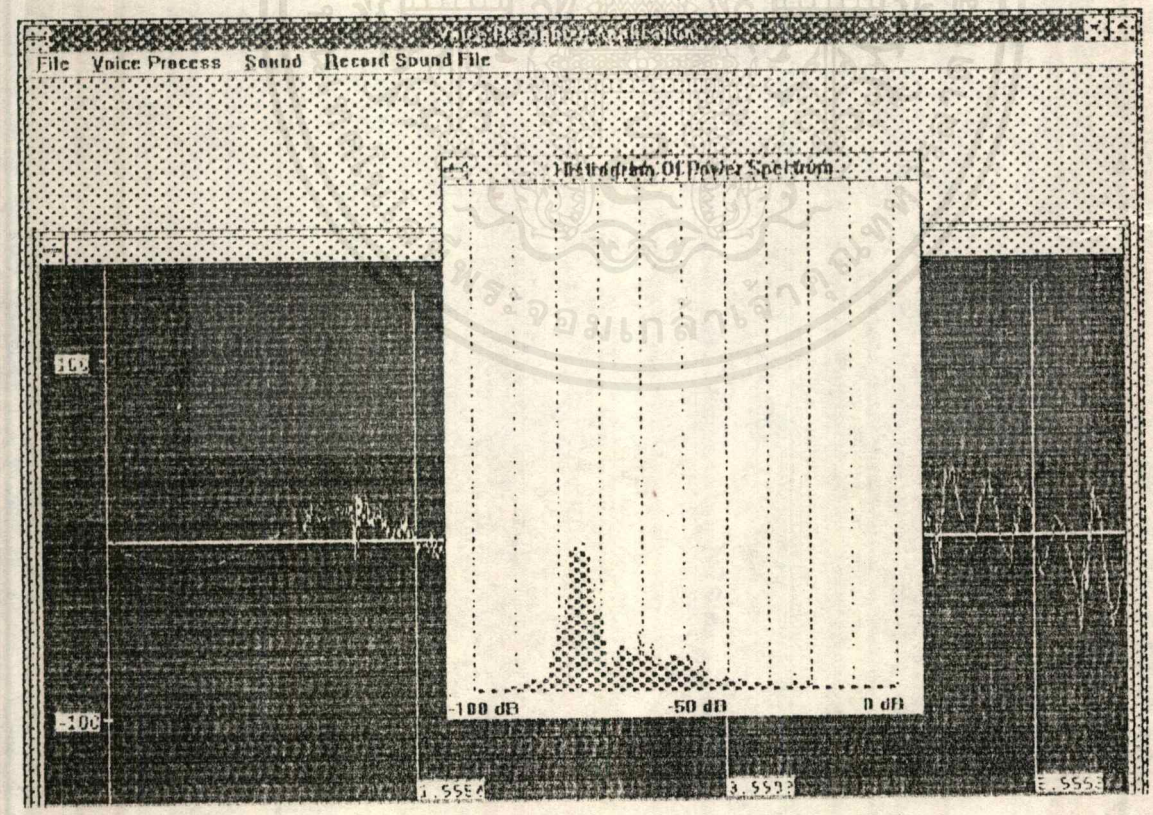


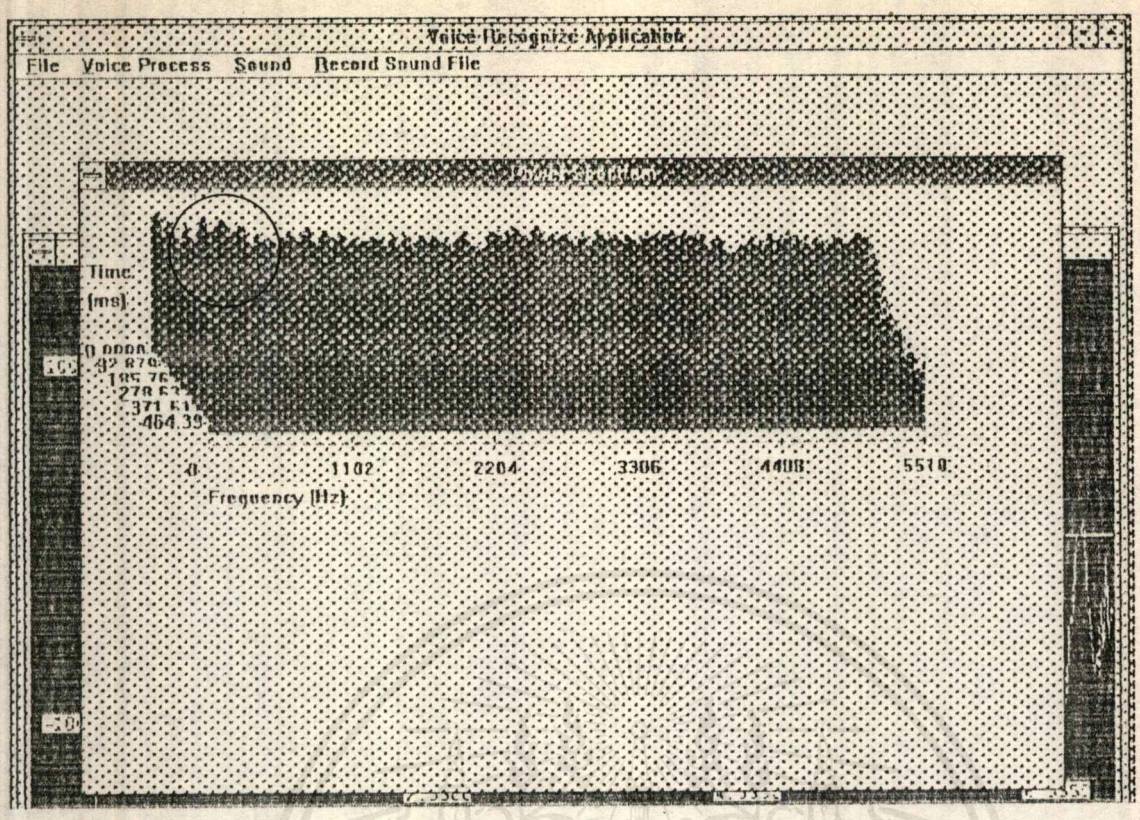
คำว่า "เจ็ด" พูดโดยบุคคลที่ 1 ครั้งที่ 3



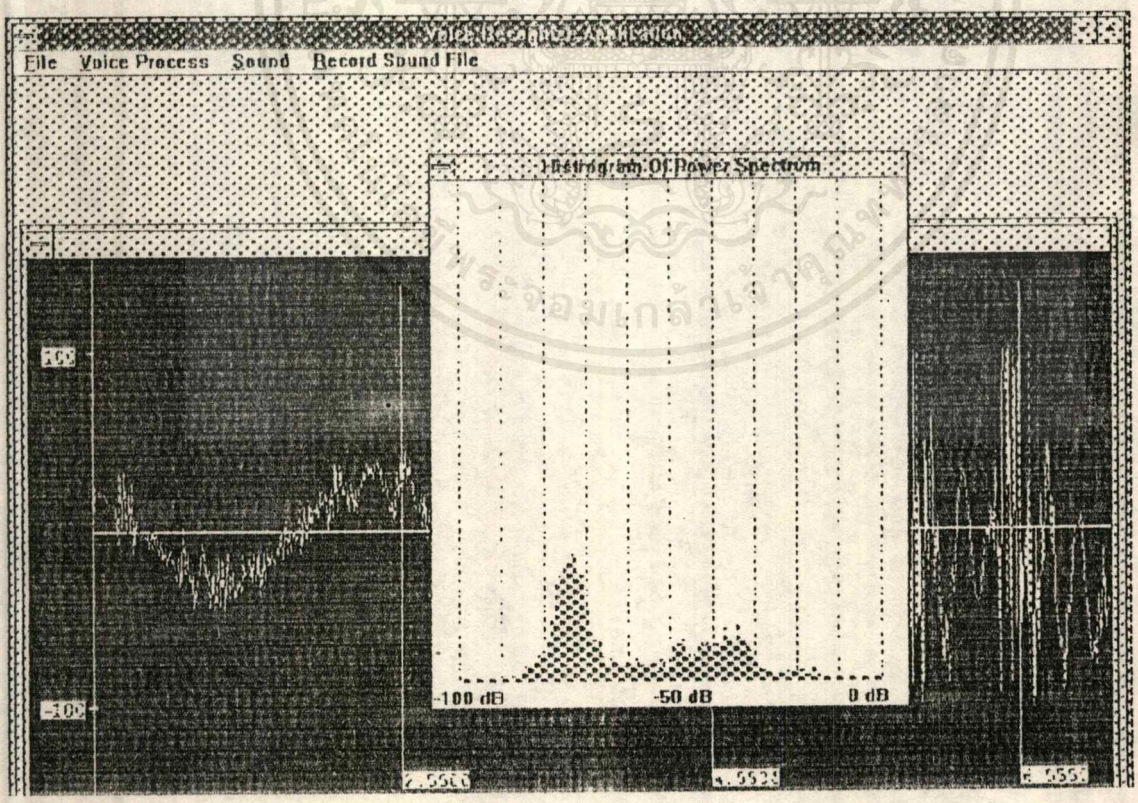


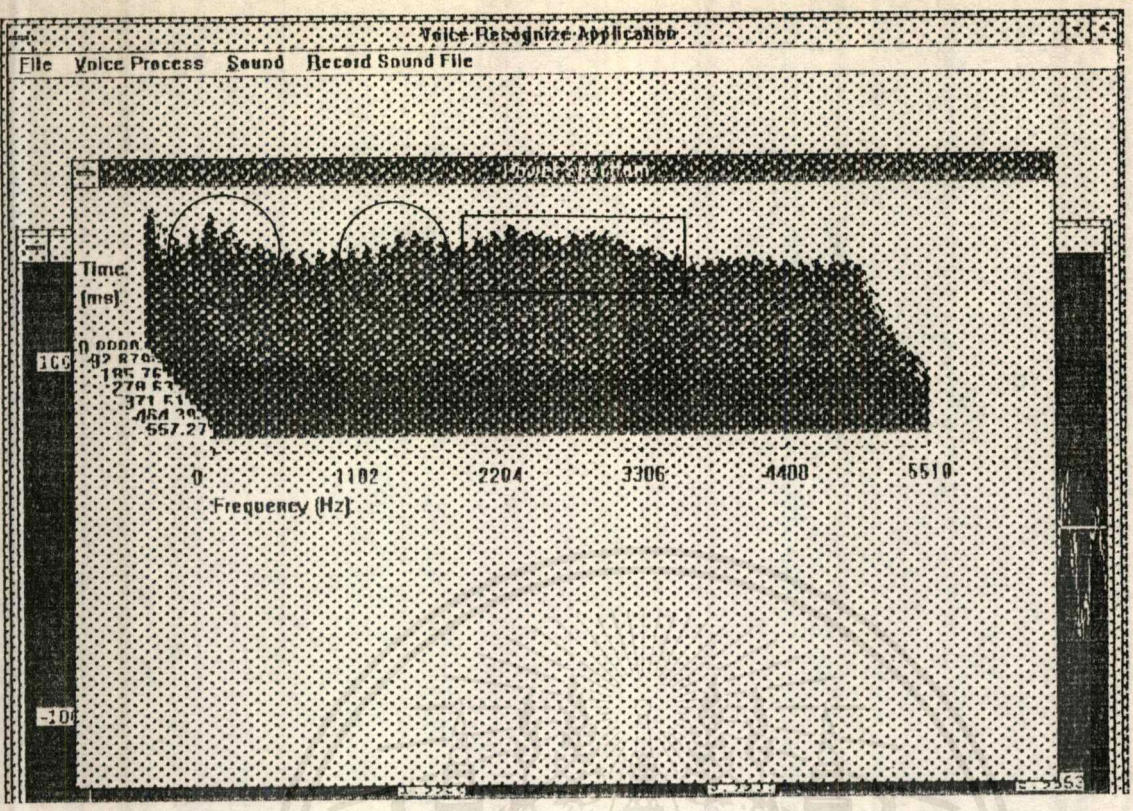
คำว่า "เจ็ด" พูดโดยบุคคลที่ 2 ครั้งที่ 1



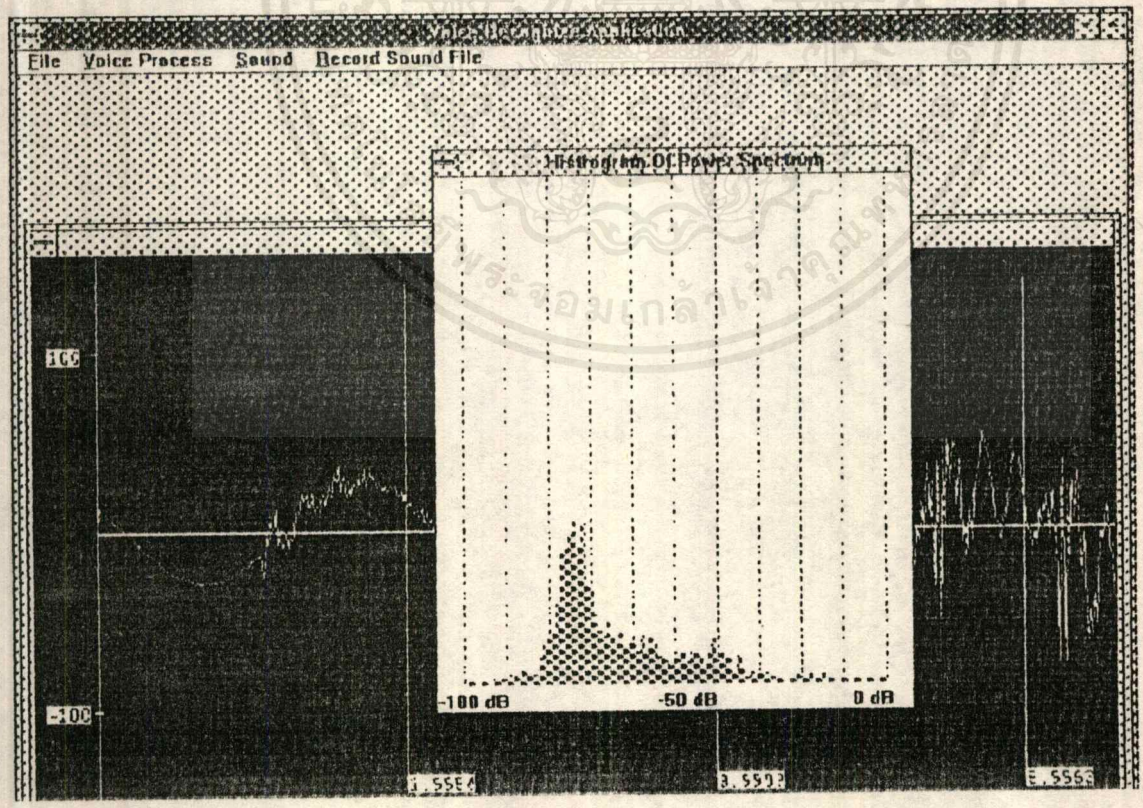


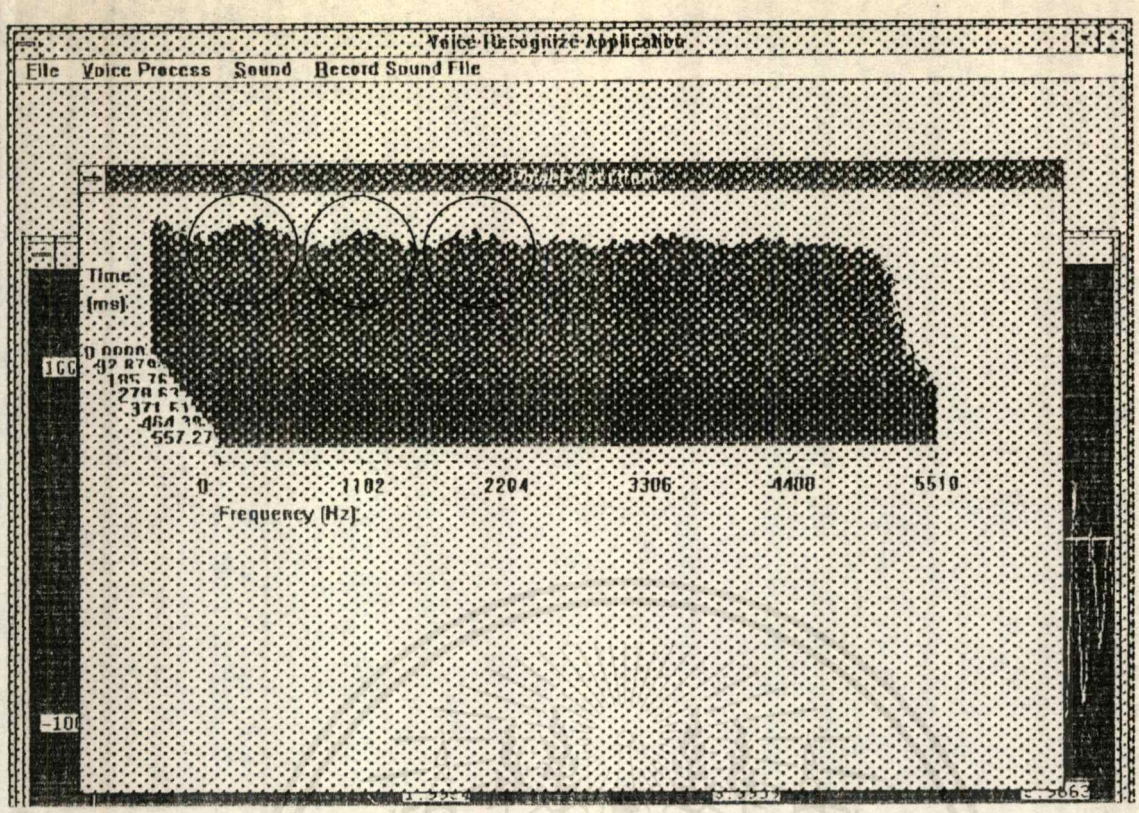
คำว่า "เจ็ด" พูดโดยบุคคลที่ 2 ครั้งที่ 2



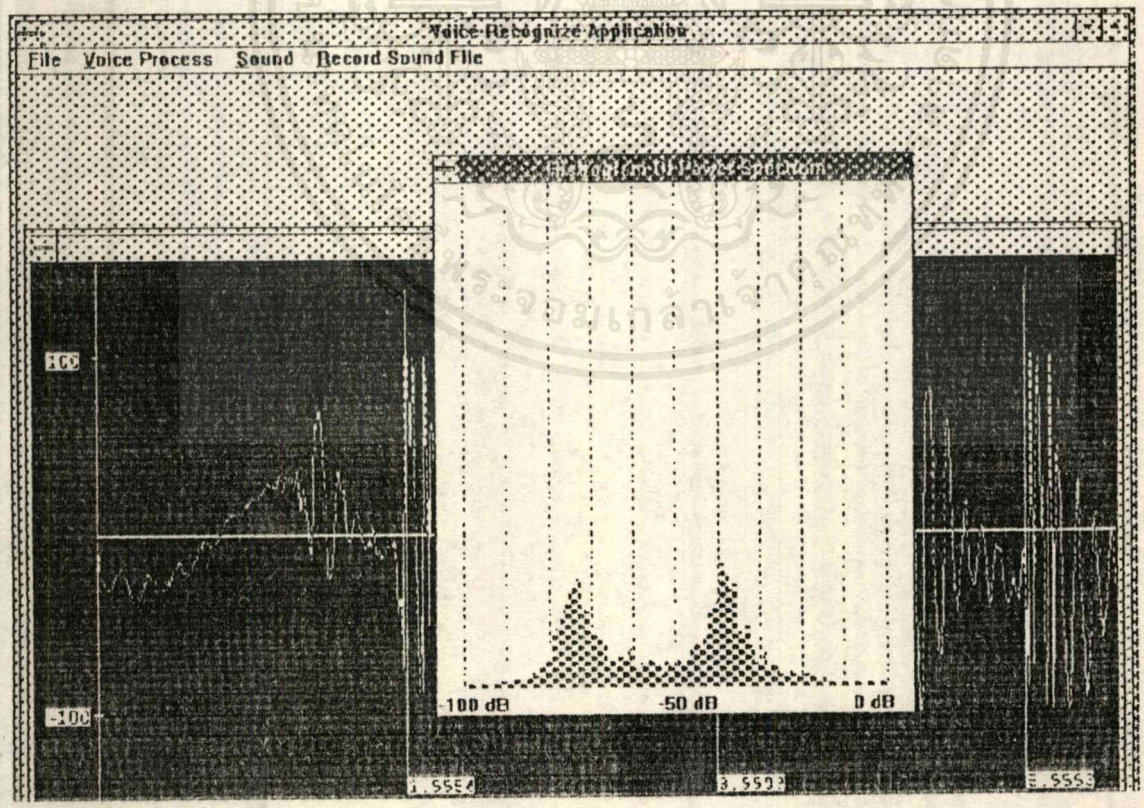


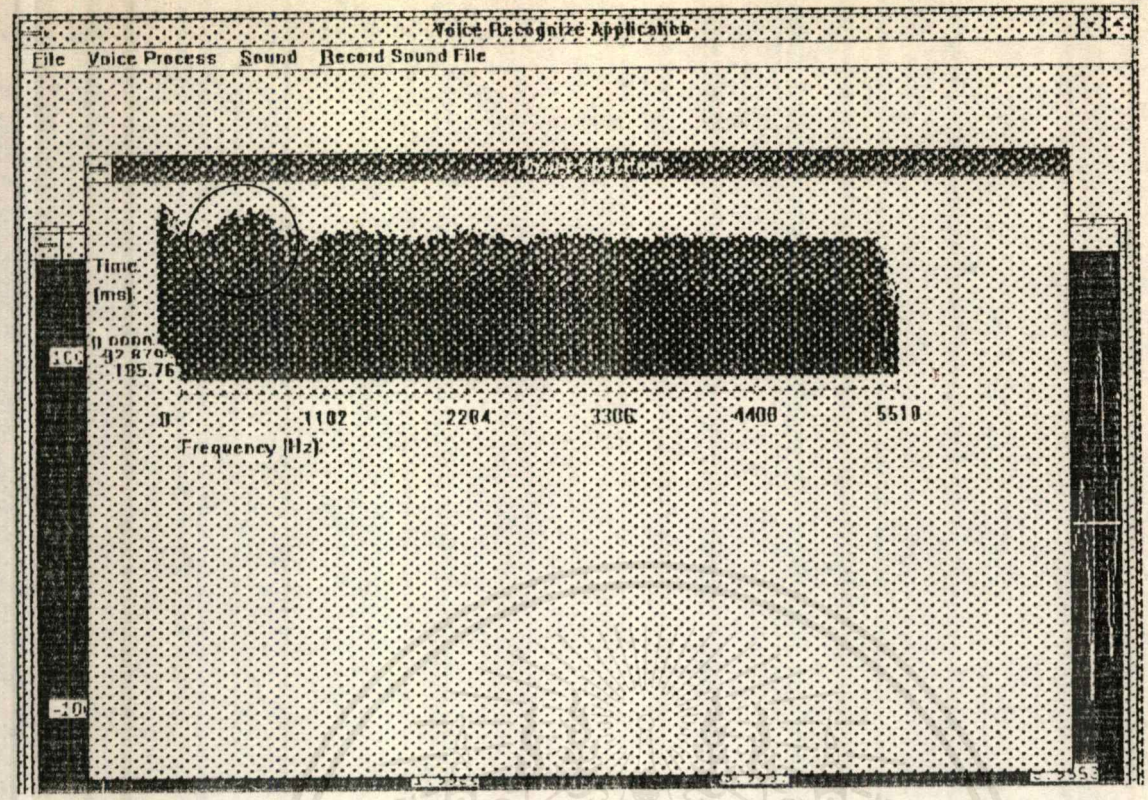
คำว่า "เจ็ด" พูดโดยบุคคลที่ 2 ครั้งที่ 3



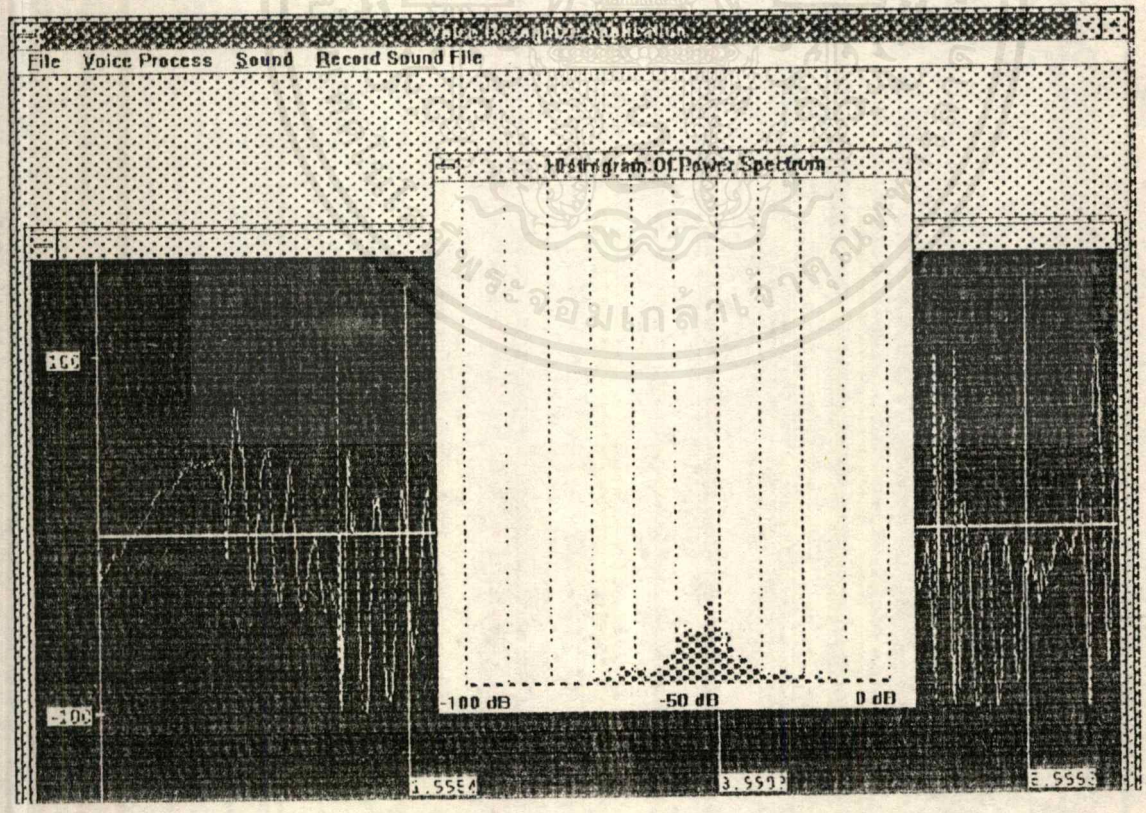


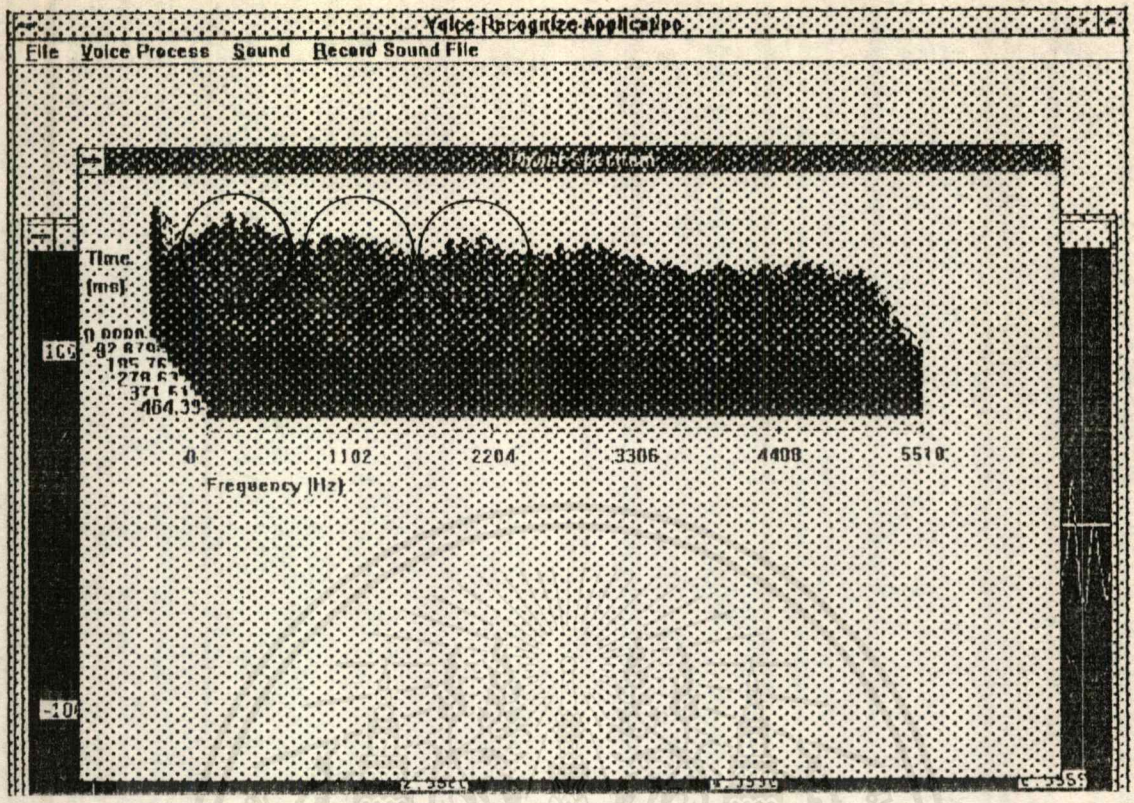
คำว่า "แปด" พูดโดยบุคคลที่ 1 ครั้งที่ 1



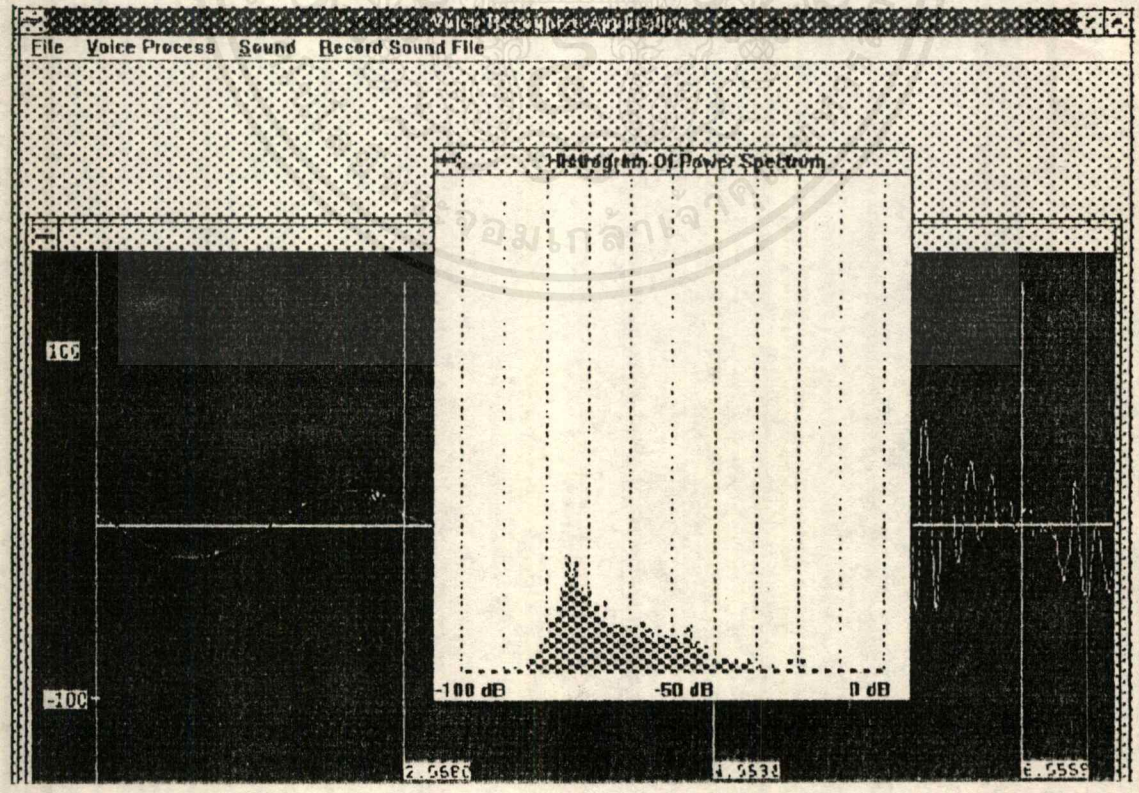


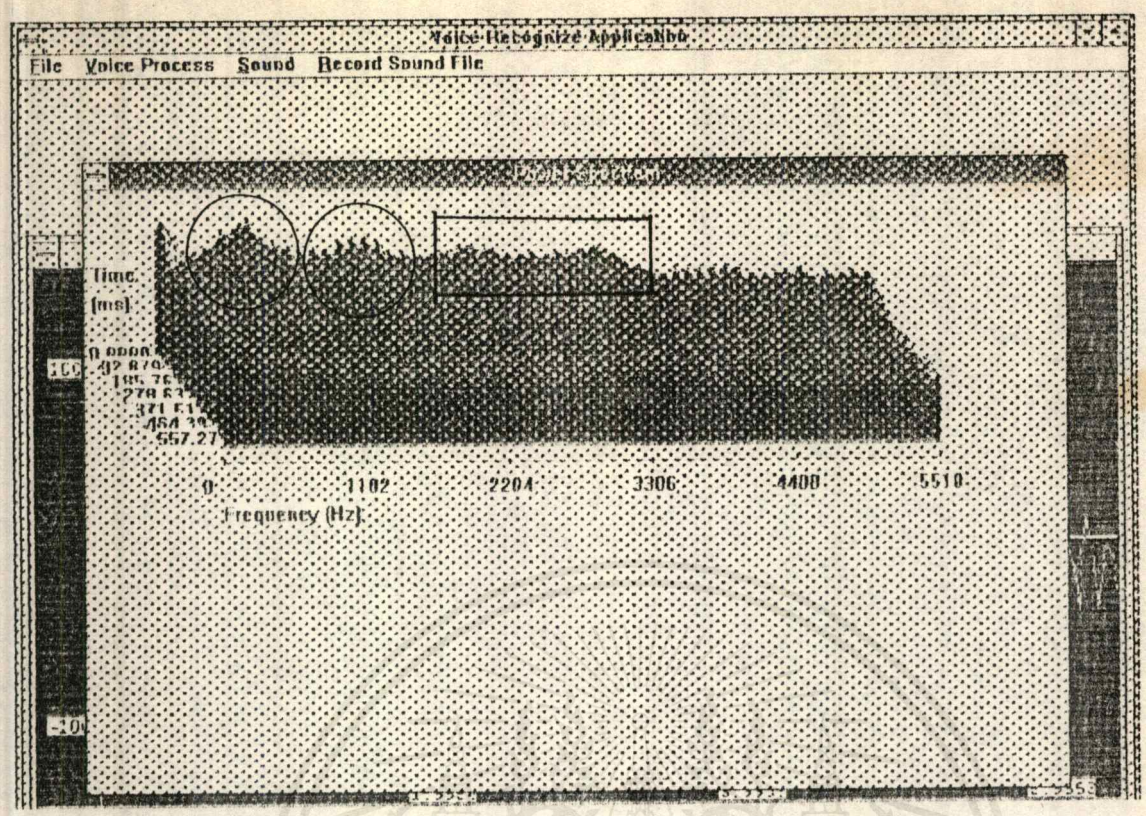
คำว่า "แปด" พูดโดยบุคคลที่ 1 ครั้งที่ 2



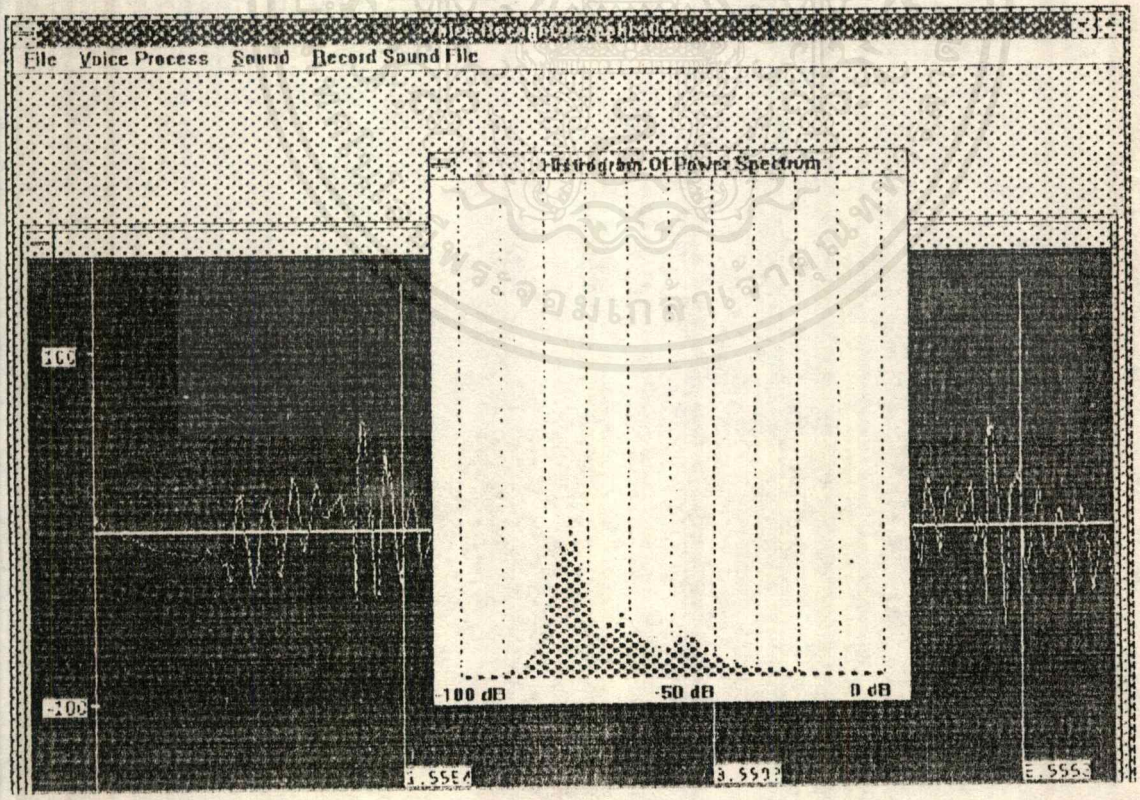


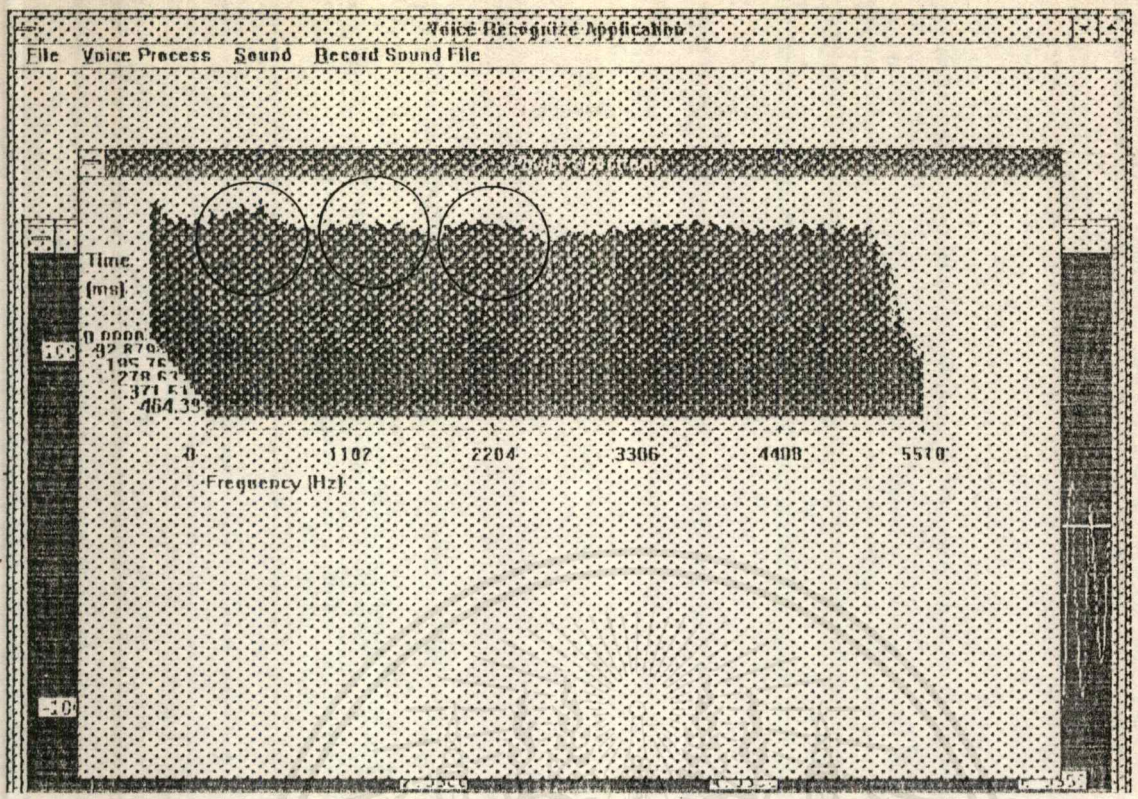
คำว่า "แปด" พูดโดยบุคคลที่ 1 ครั้งที่ 3



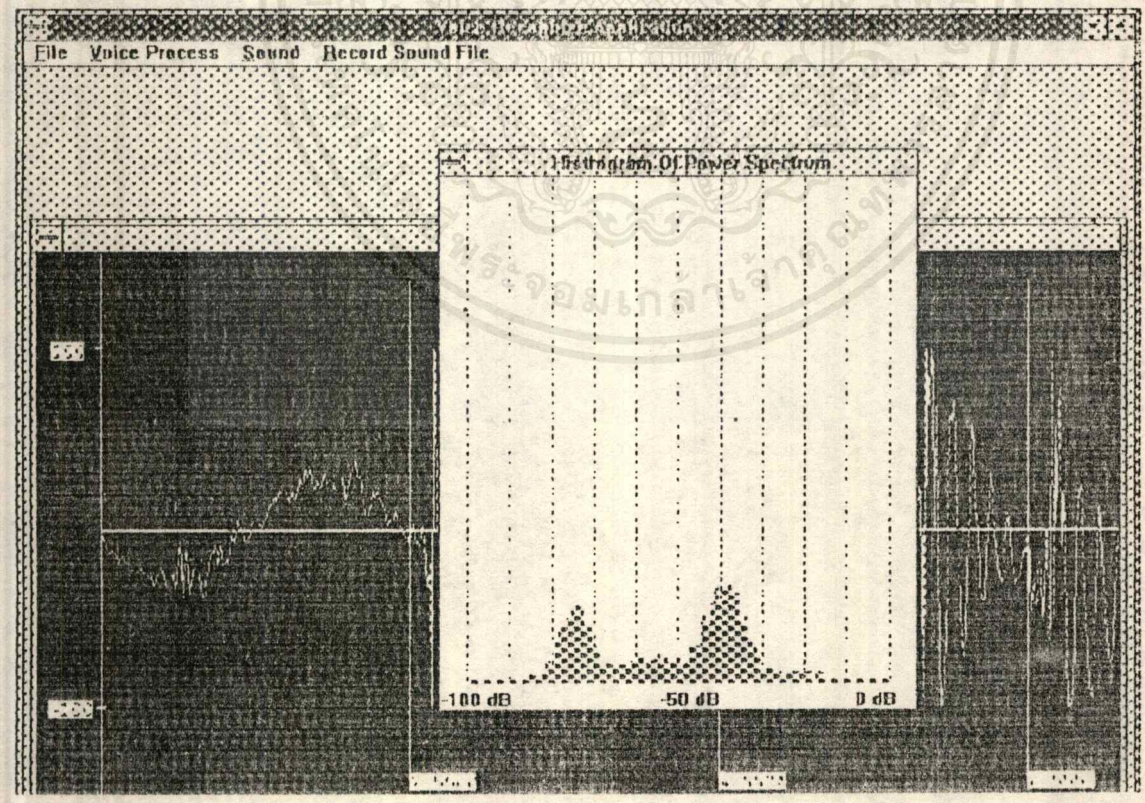


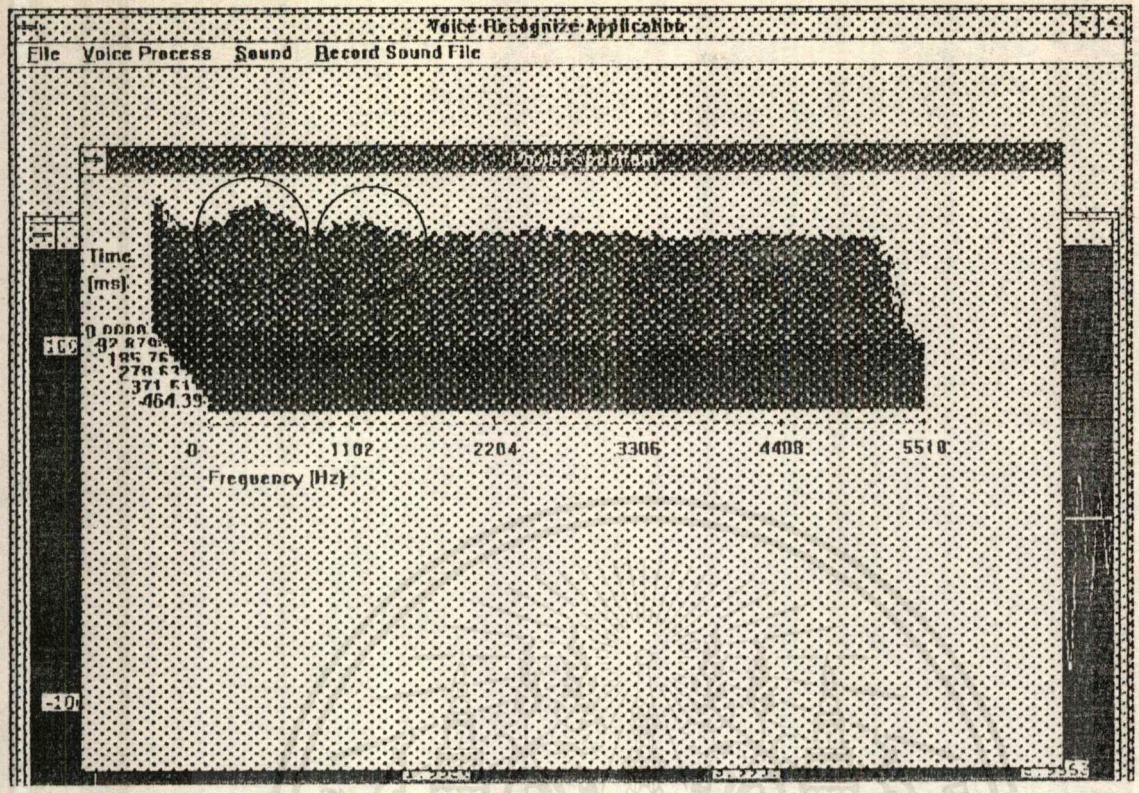
คำว่า "แปด" พูดโดยบุคคลที่ 2 ครั้งที่ 1



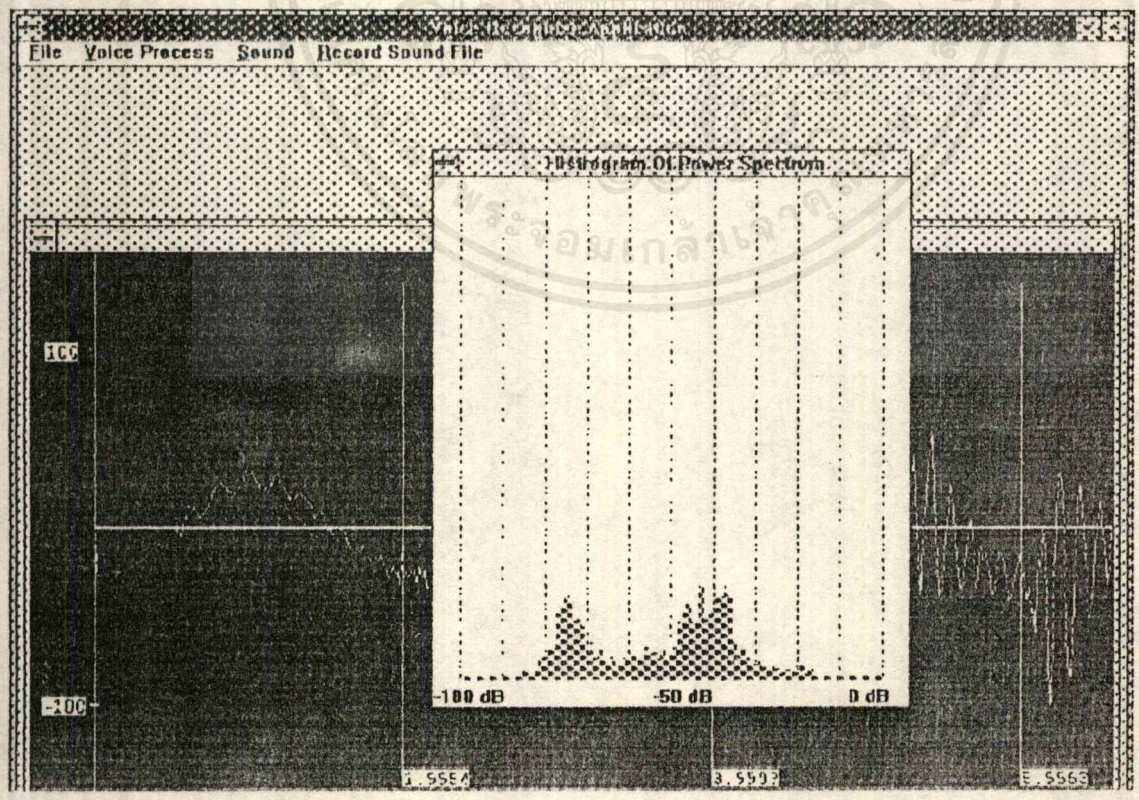


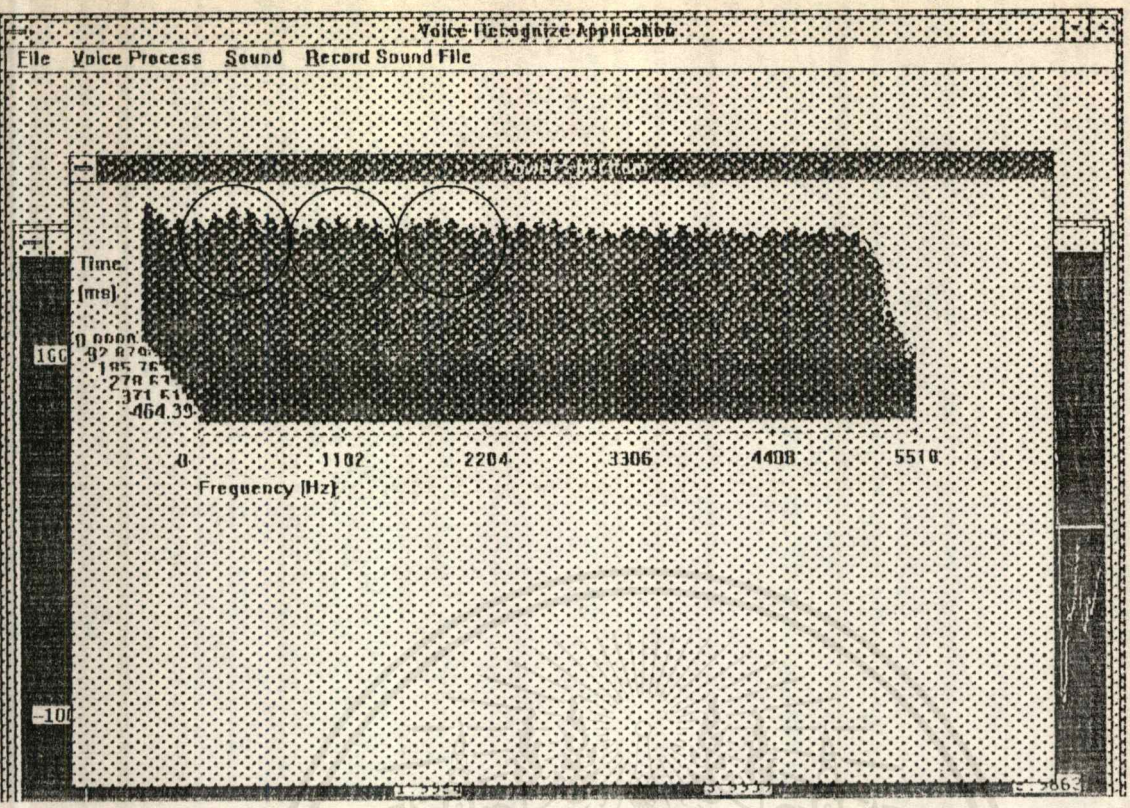
คำว่า "แปด" พูดโดยบุคคลที่ 2 ครั้งที่ 2



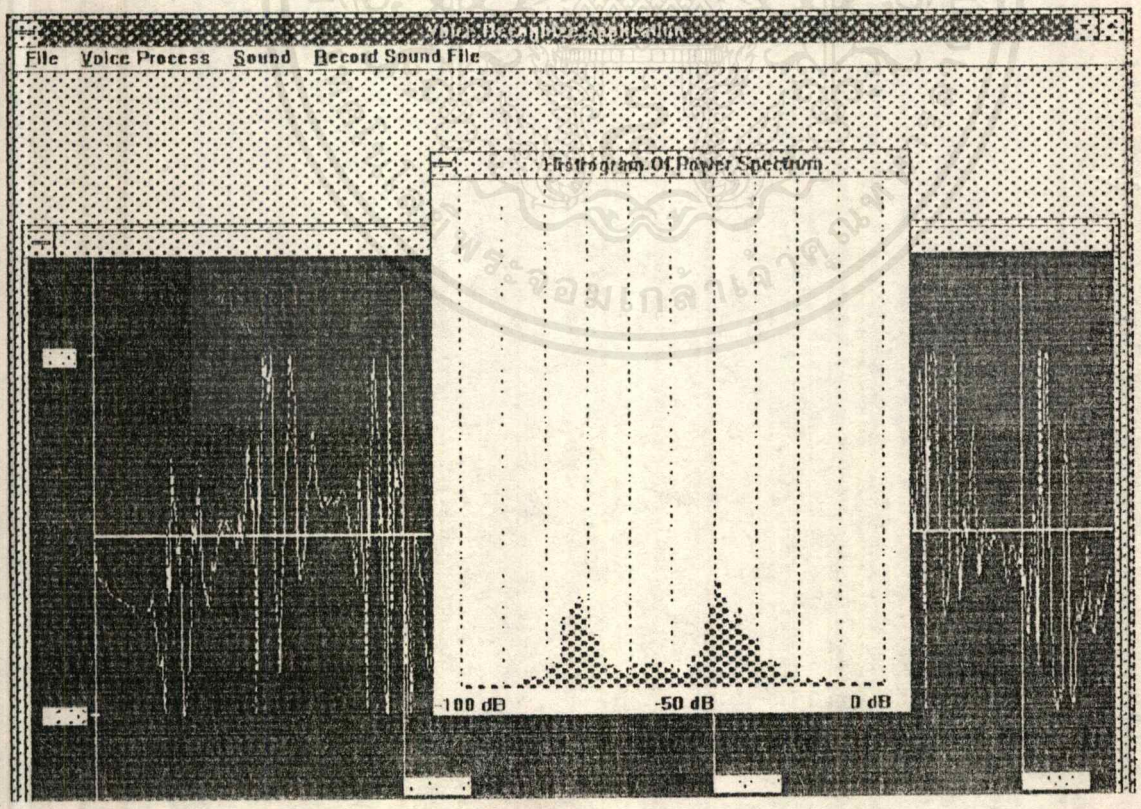


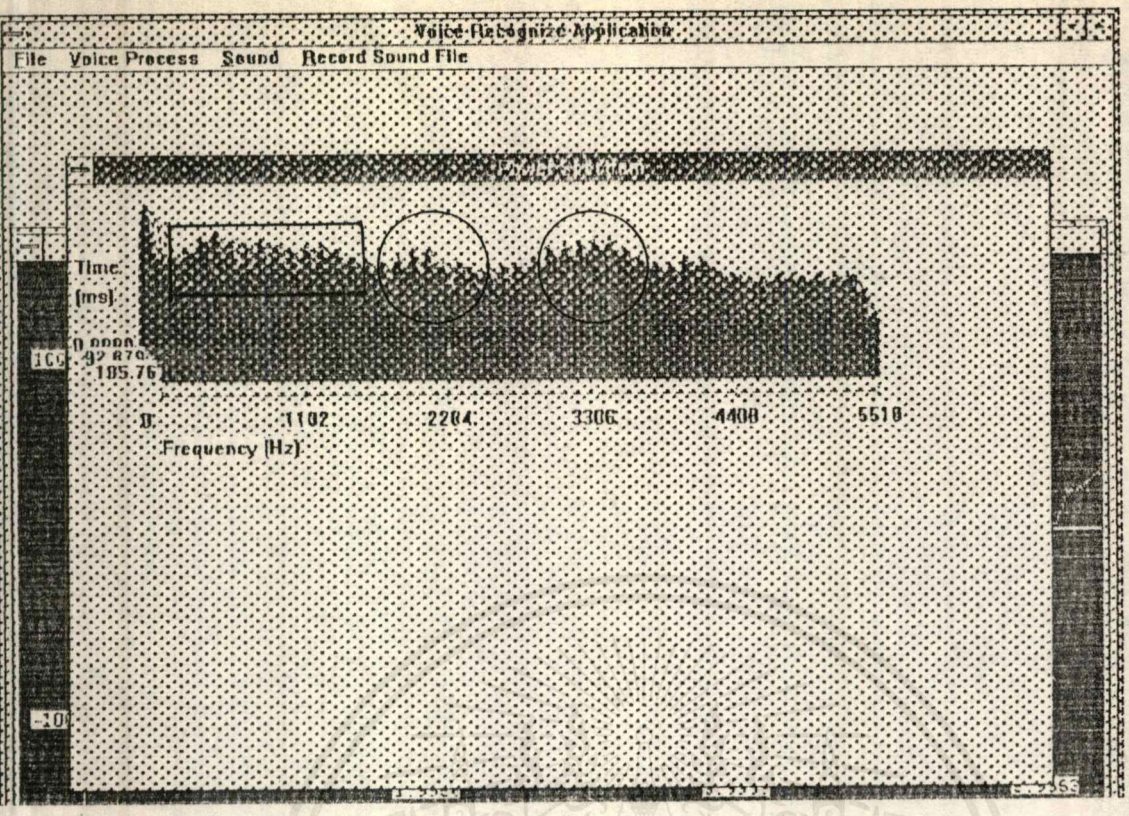
คำว่า "แปด" พูดโดยบุคคลที่ 2 ครั้งที่ 3



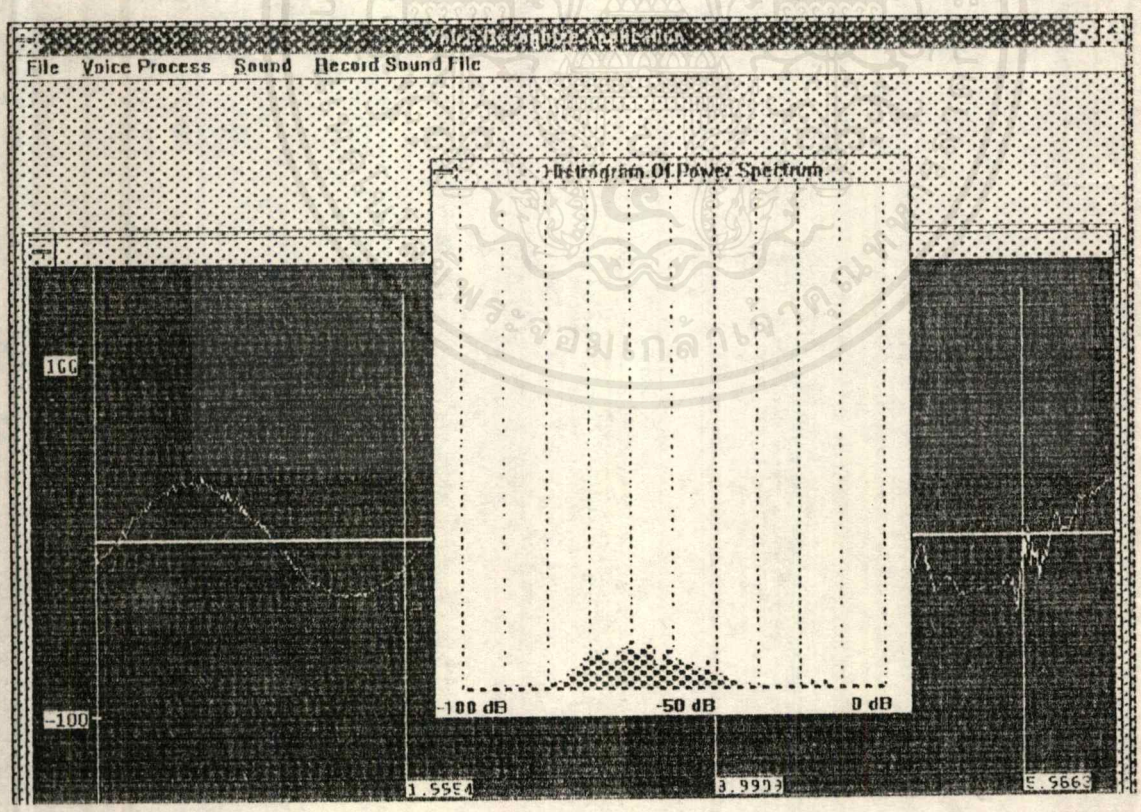


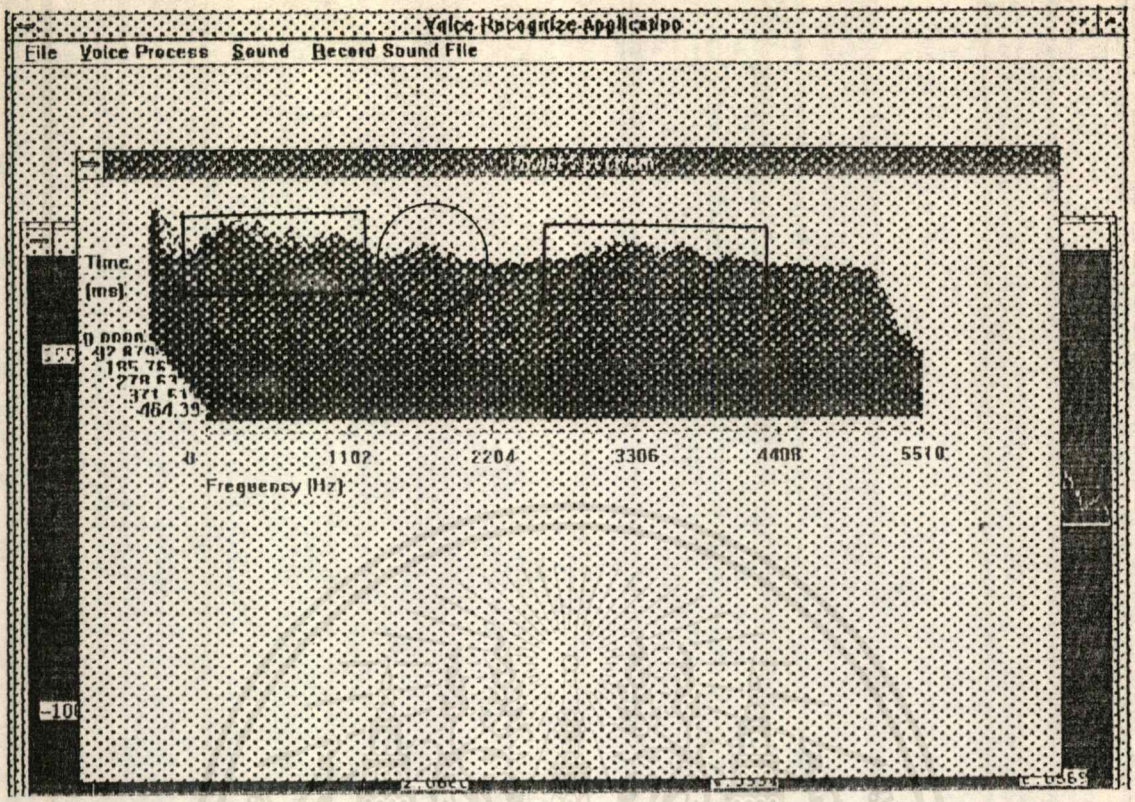
คำว่า "เก้า" พูดโดยบุคคลที่ 1 ครั้งที่ 1



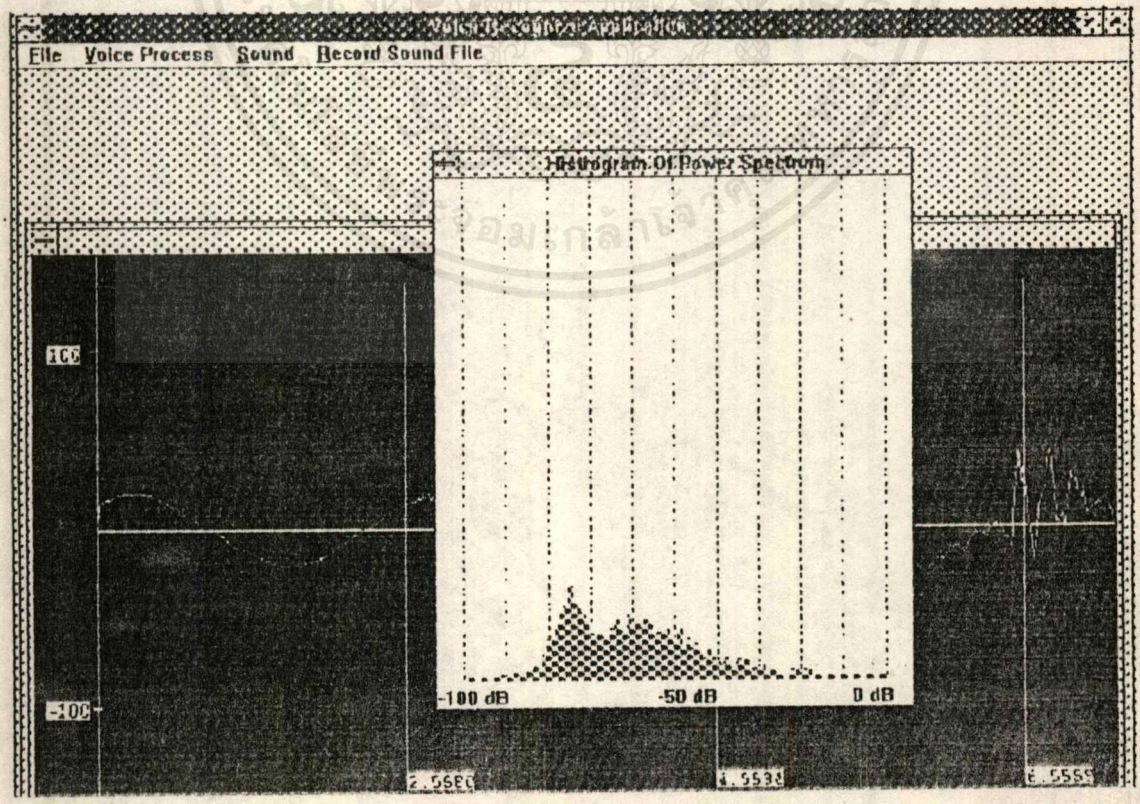


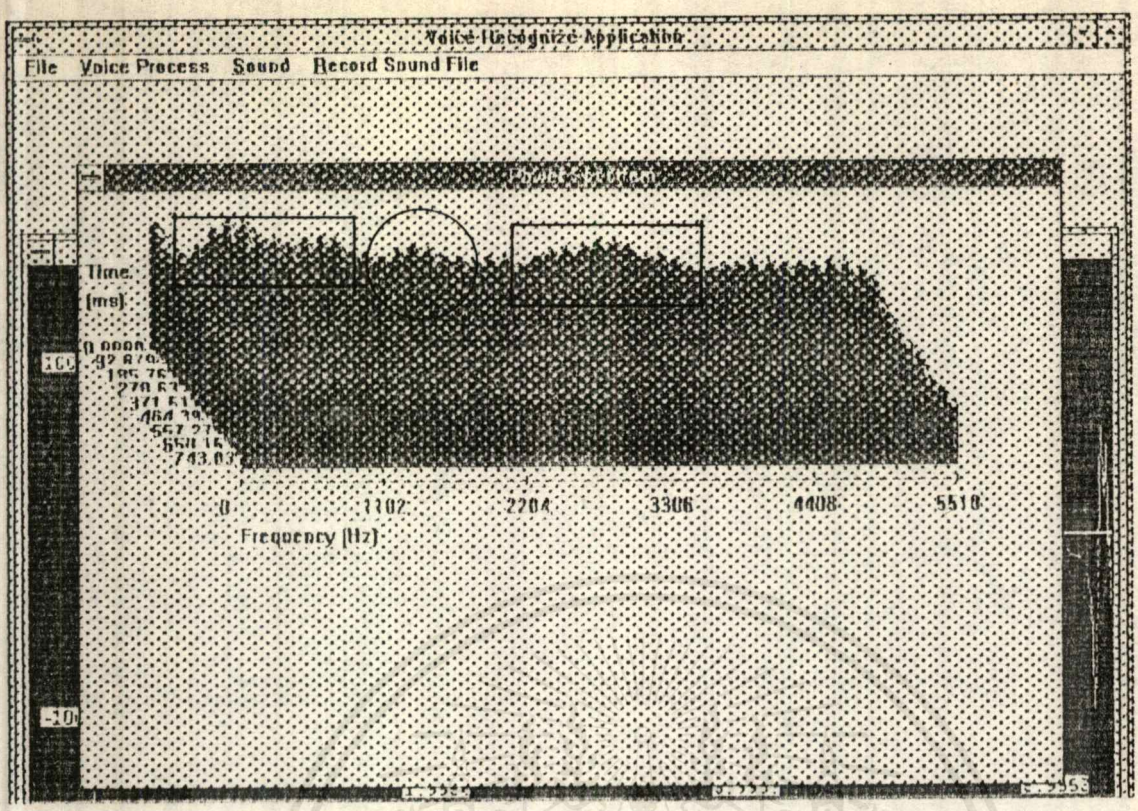
คำว่า "เก้า" พูดโดยบุคคลที่ 1 ครั้งที่ 2



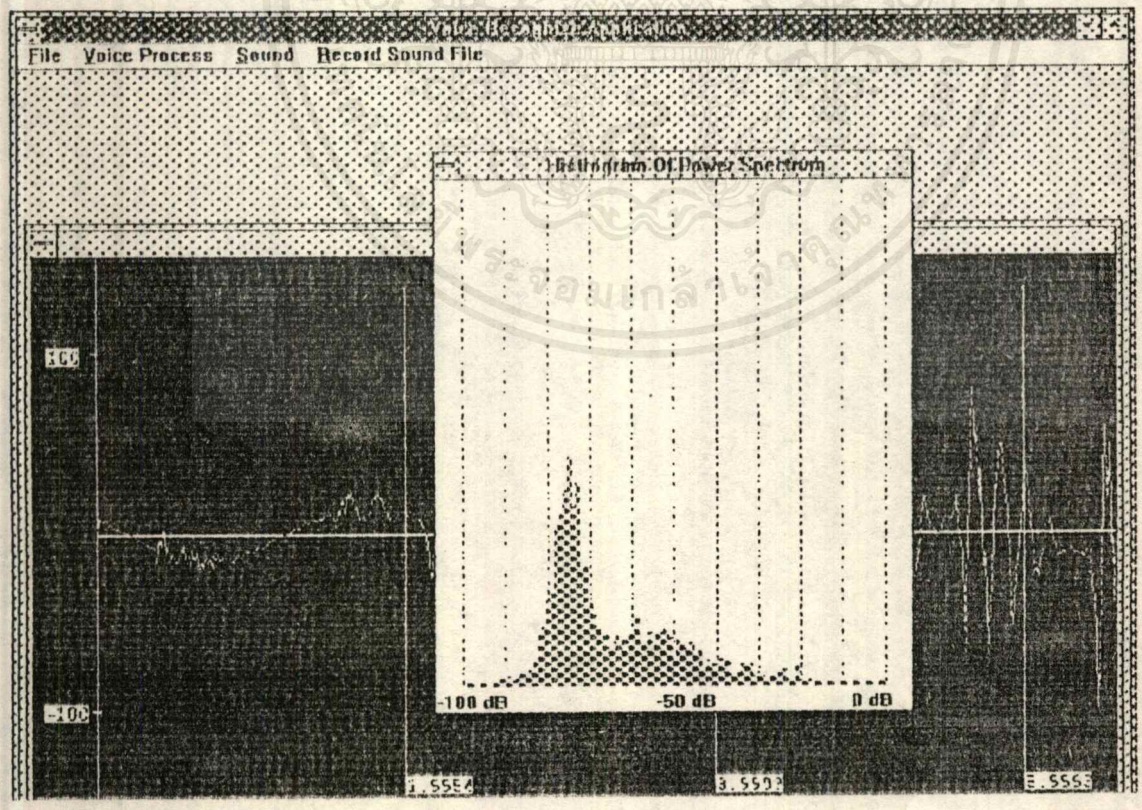


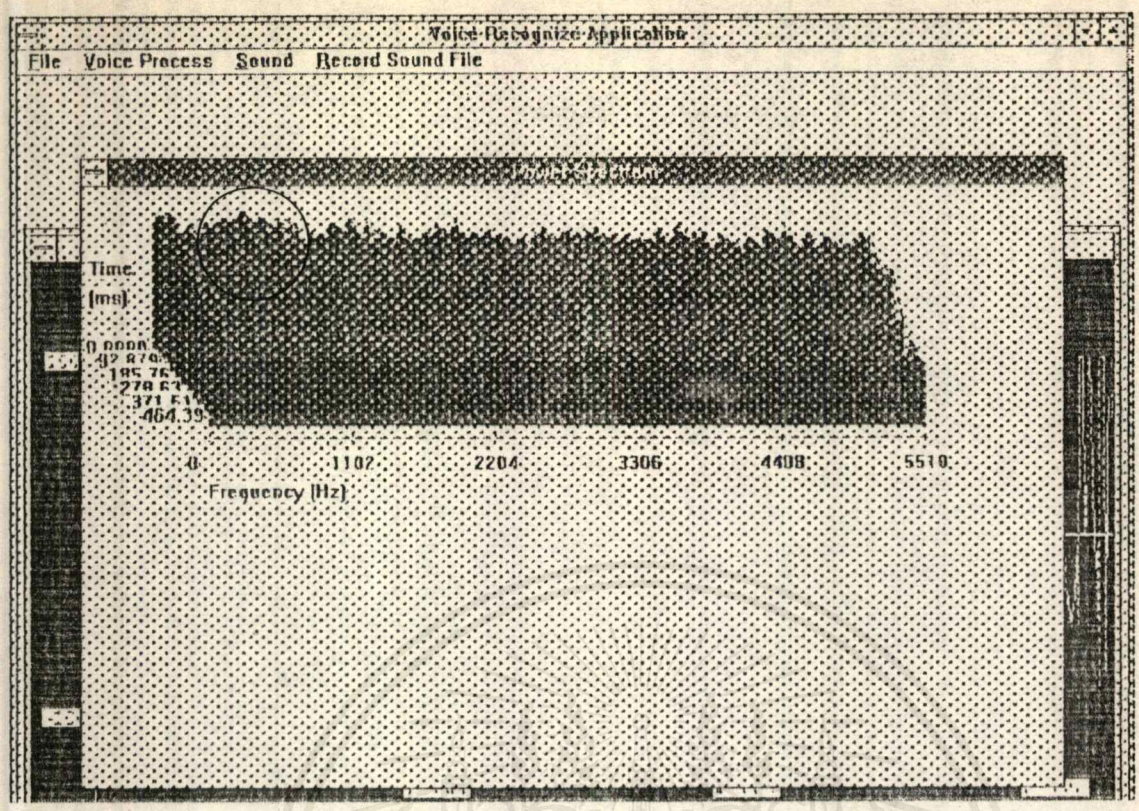
คำว่า "เก้า" พูดโดยบุคคลที่ 1 ครั้งที่ 3



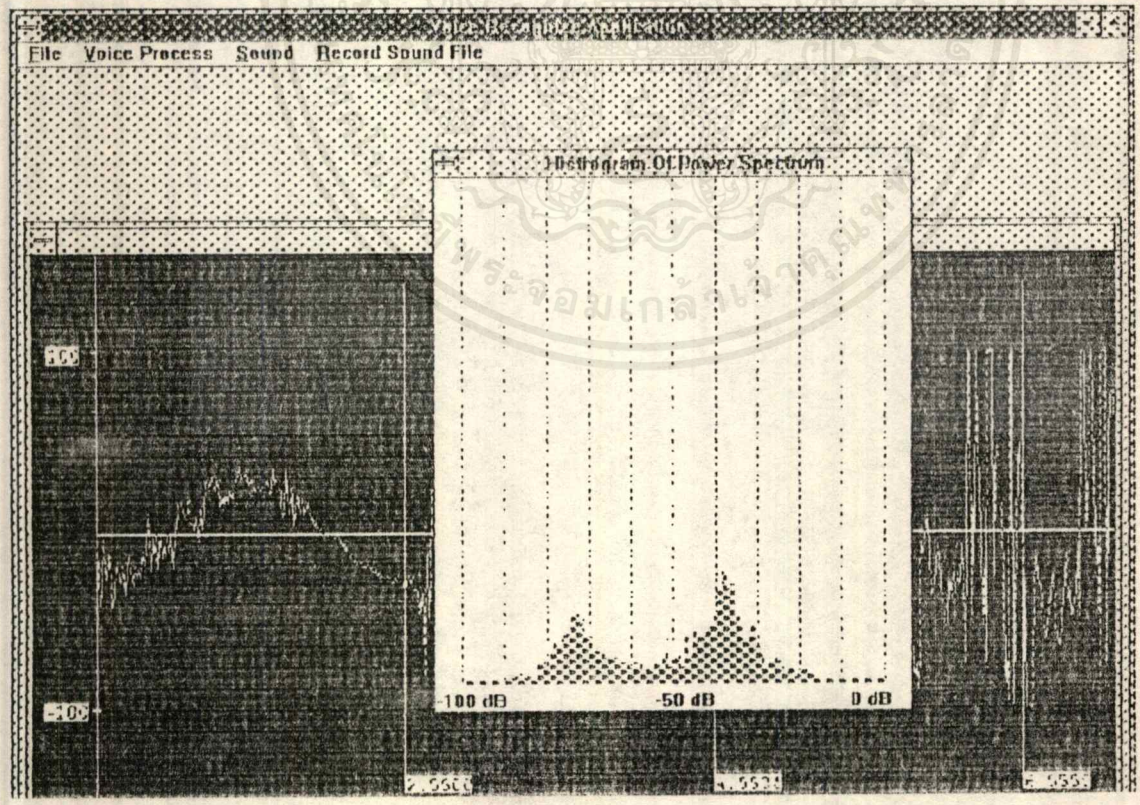


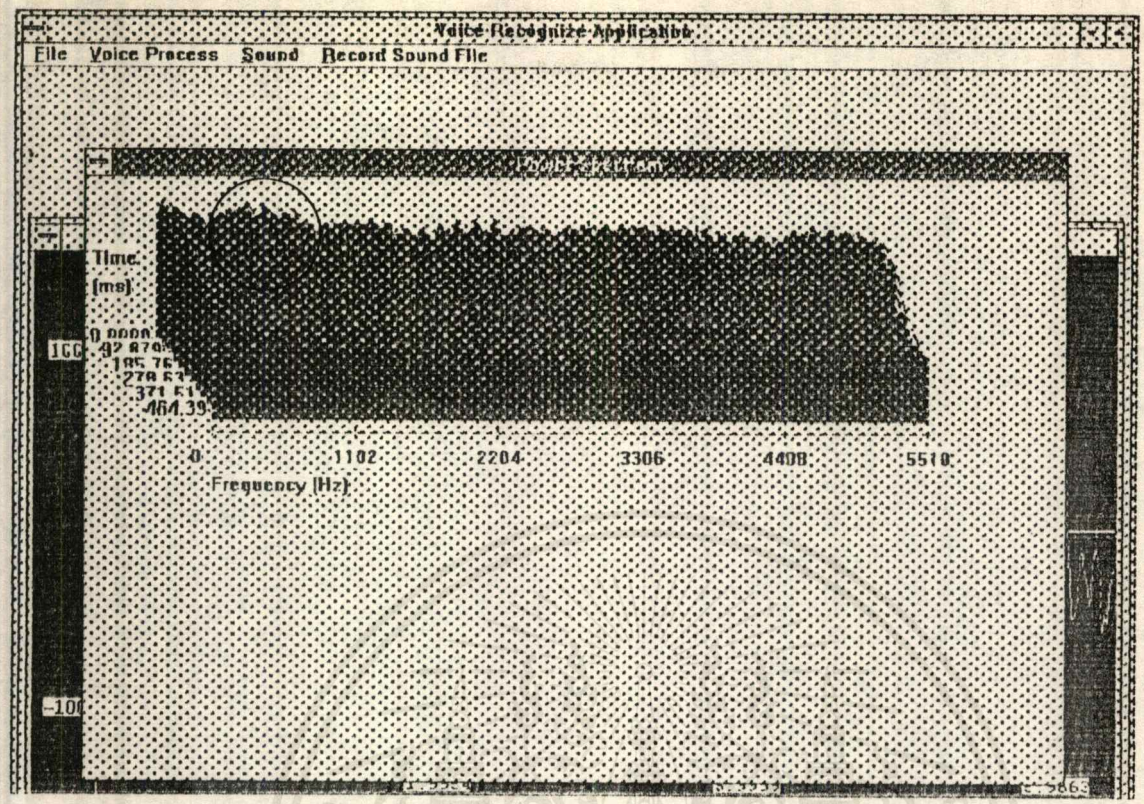
คำว่า "เก้า" พูดโดยบุคคลที่ 2 ครั้งที่ 1



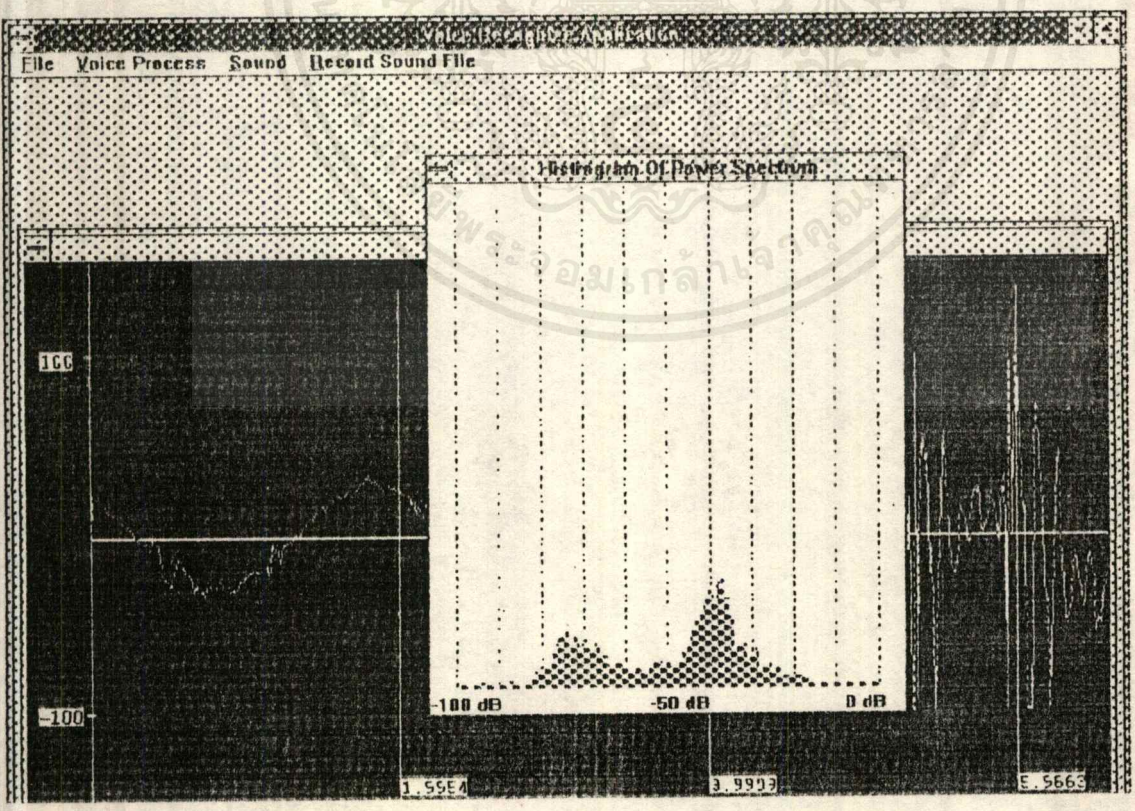


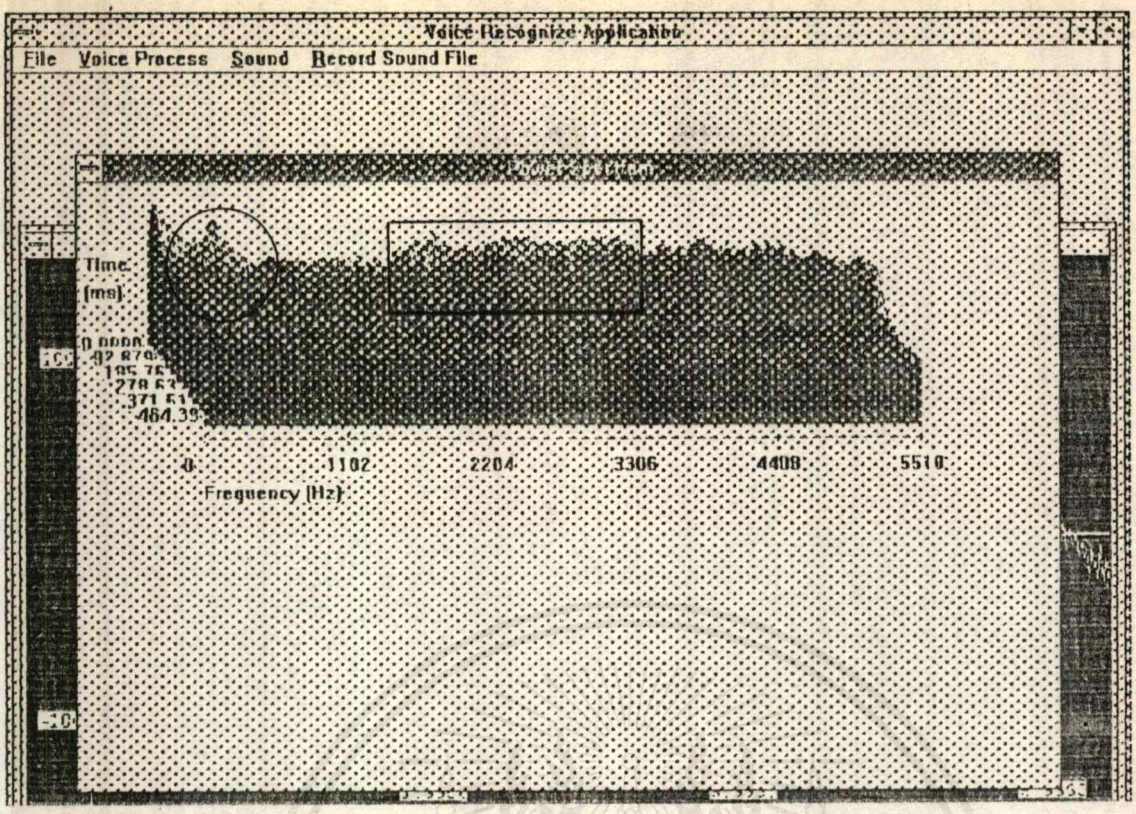
คำว่า "เก้า" พูดโดยบุคคลที่ 2 ครั้งที่ 2



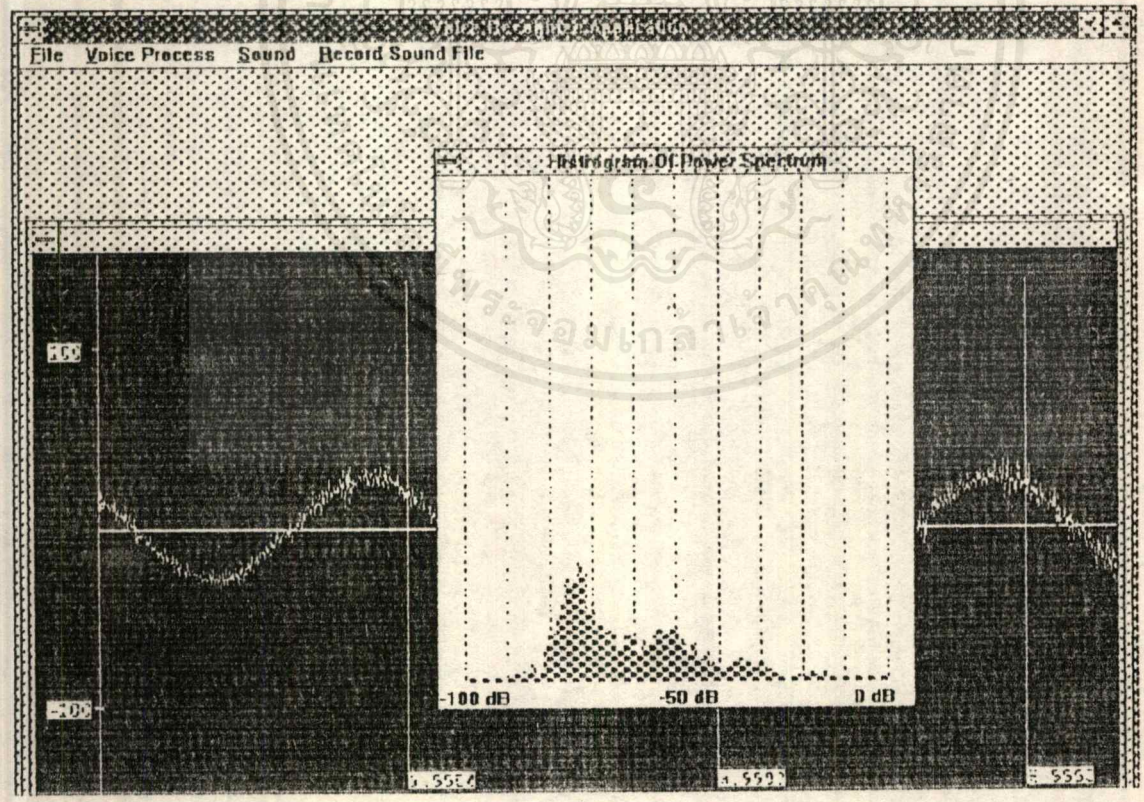


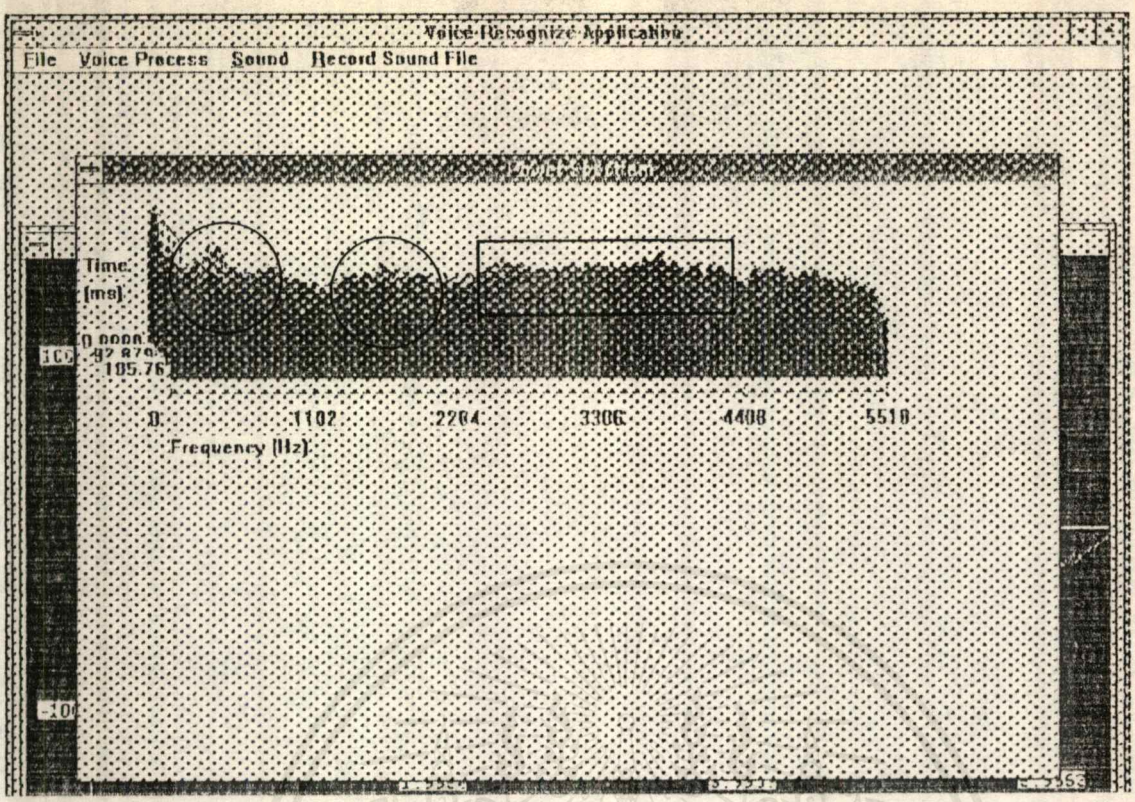
คำว่า "เก้า" พูดโดยบุคคลที่ 2 ครั้งที่ 3



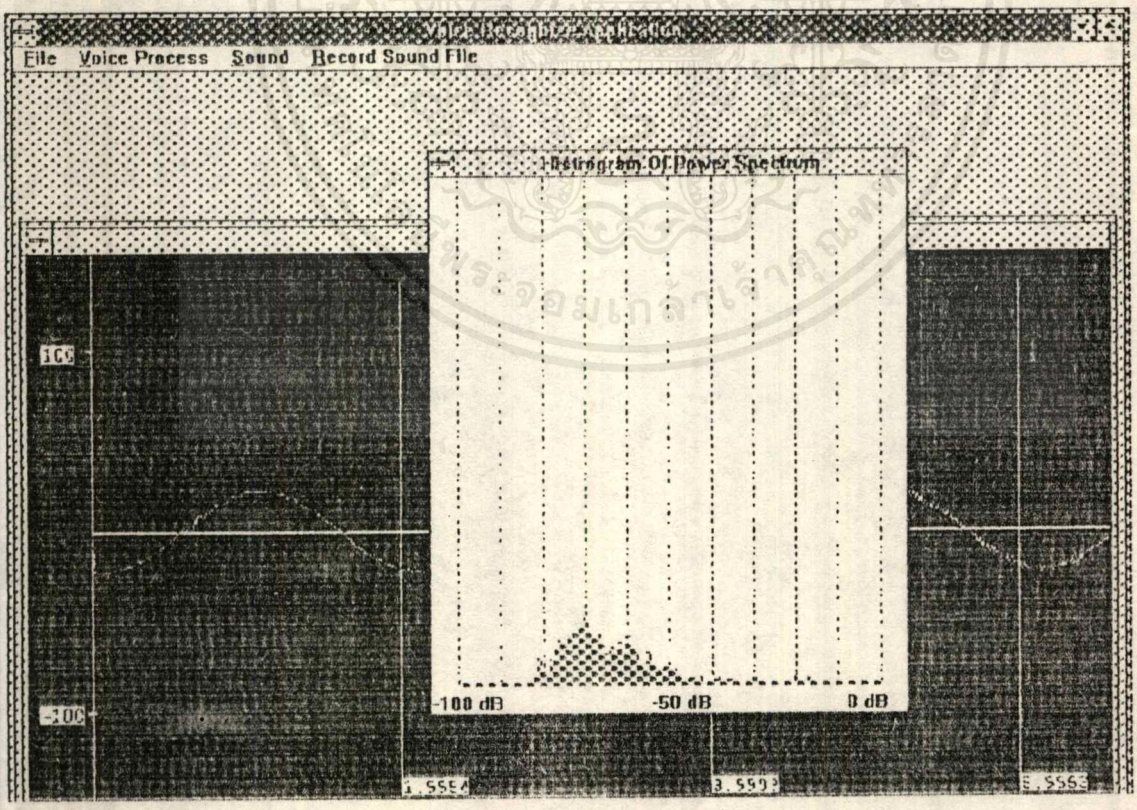


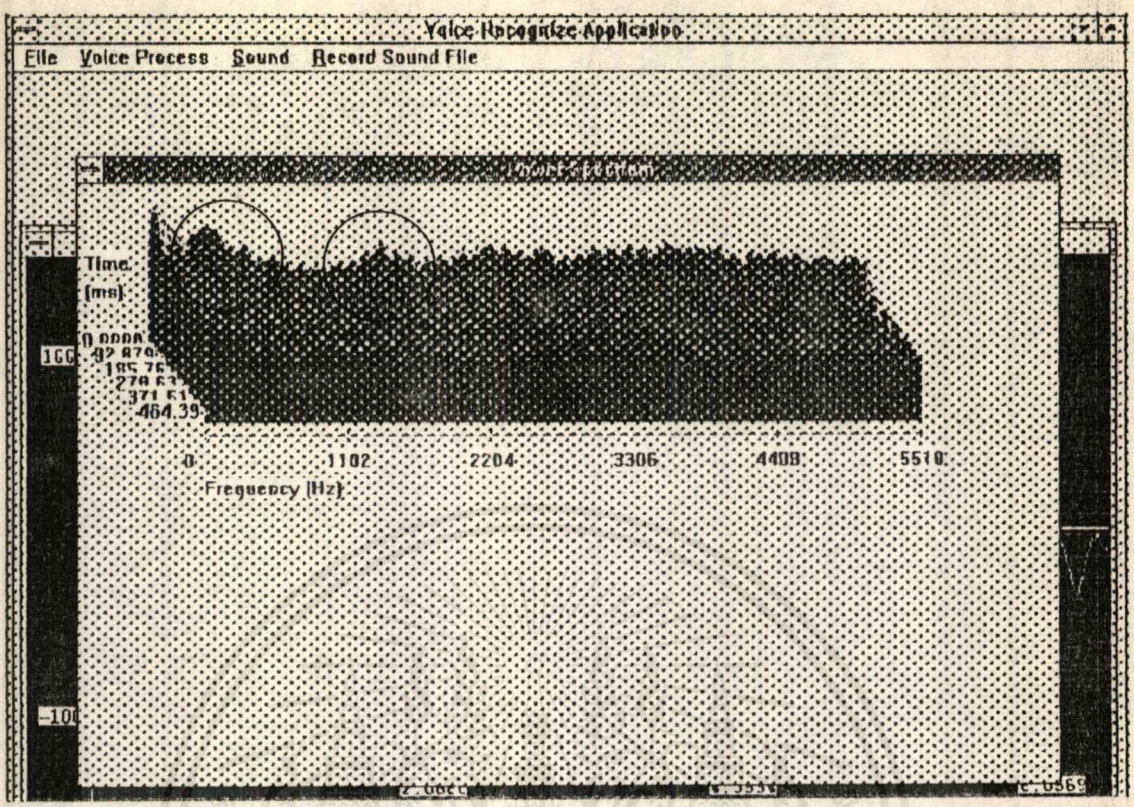
คำว่า "สิบ" พูดโดยบุคคลที่ 1 ครั้งที่ 1



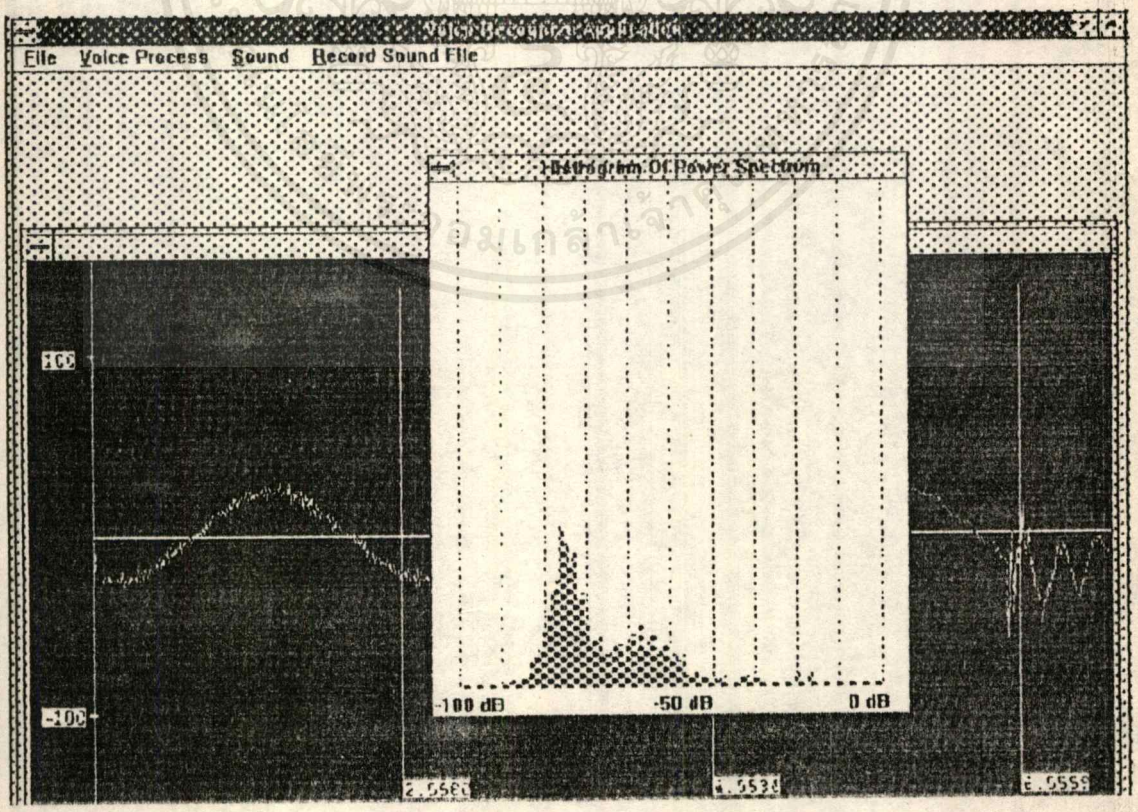


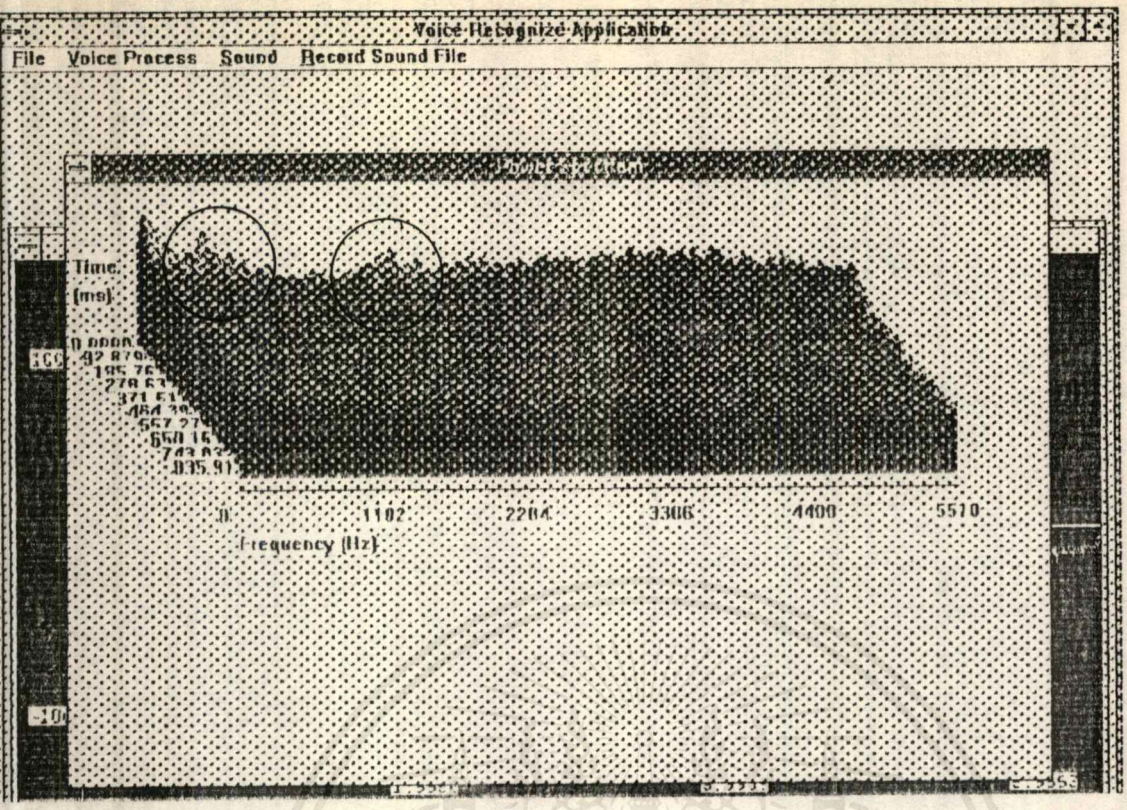
คำว่า "สิบ" พูดโดยบุคคลที่ 1 ครั้งที่ 2



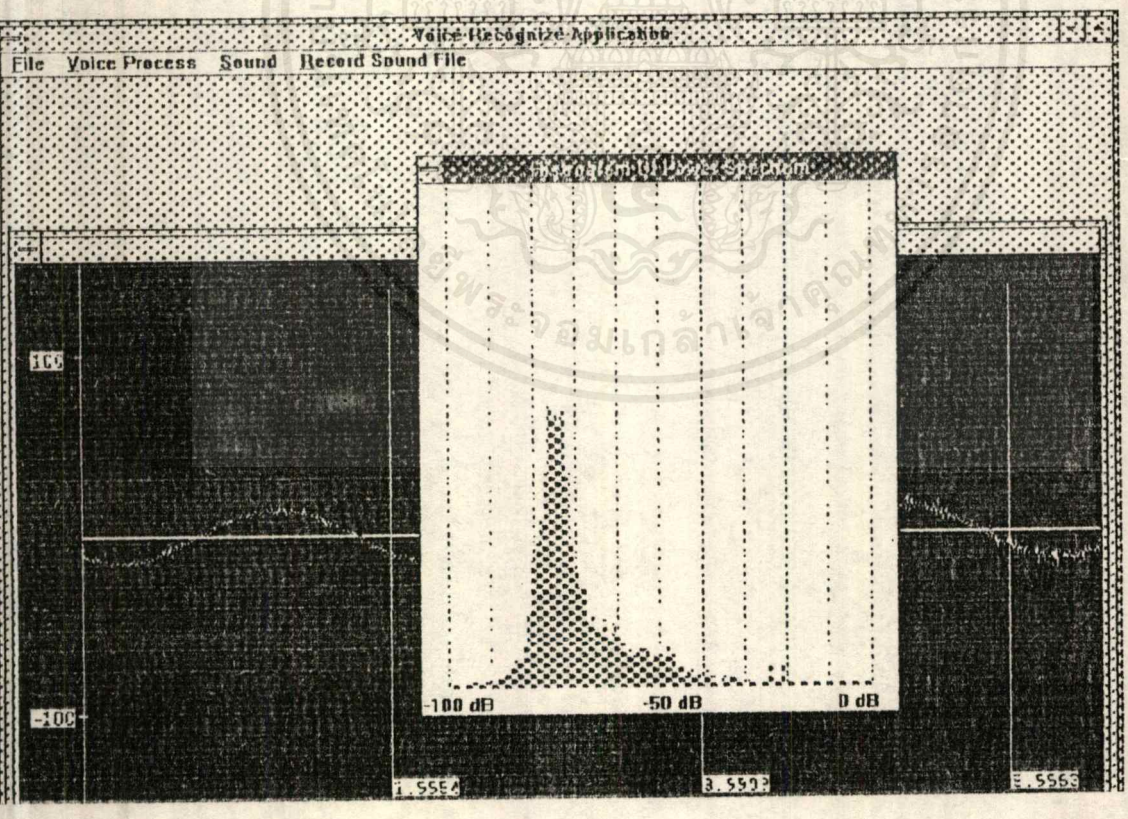


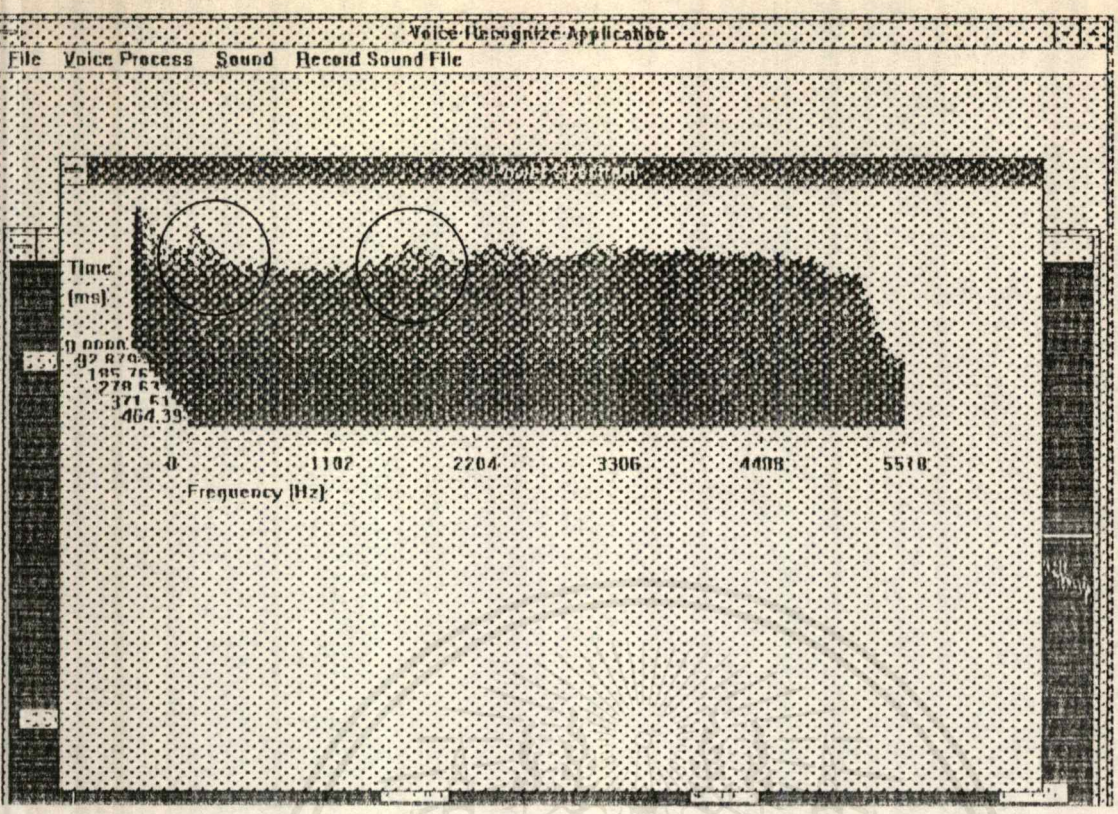
คำว่า "สิบ" พูดโดยบุคคลที่ 1 ครั้งที่ 3



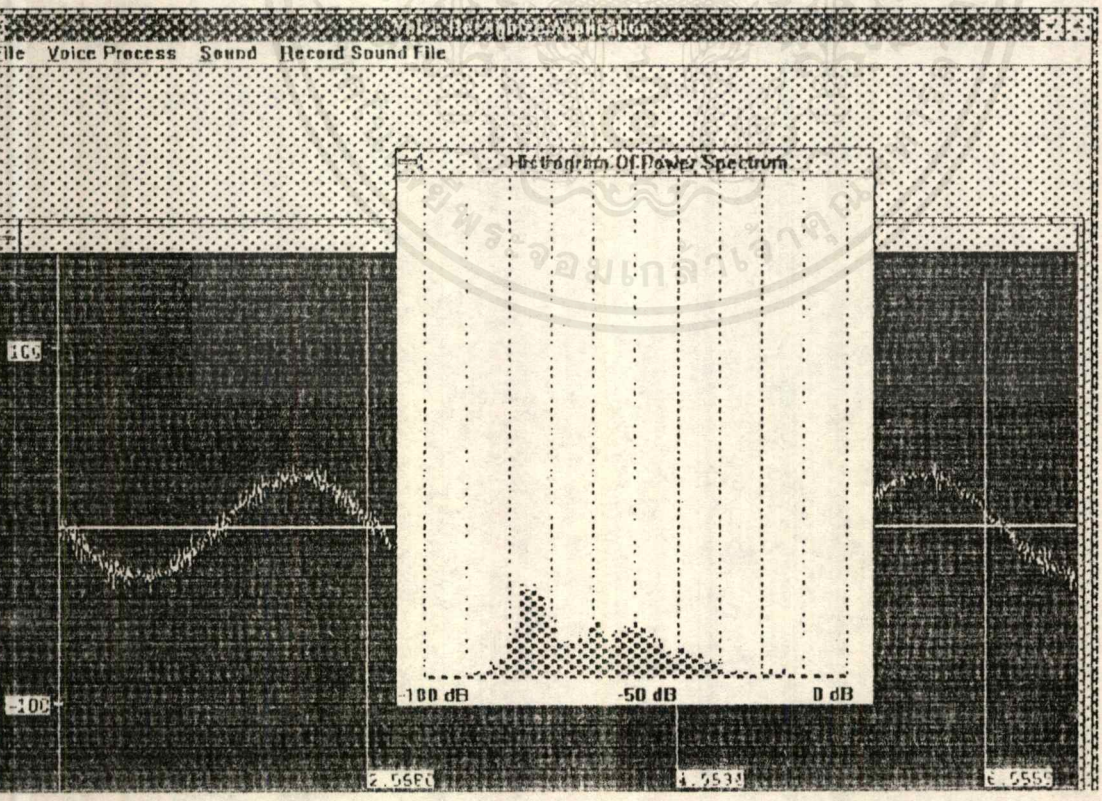


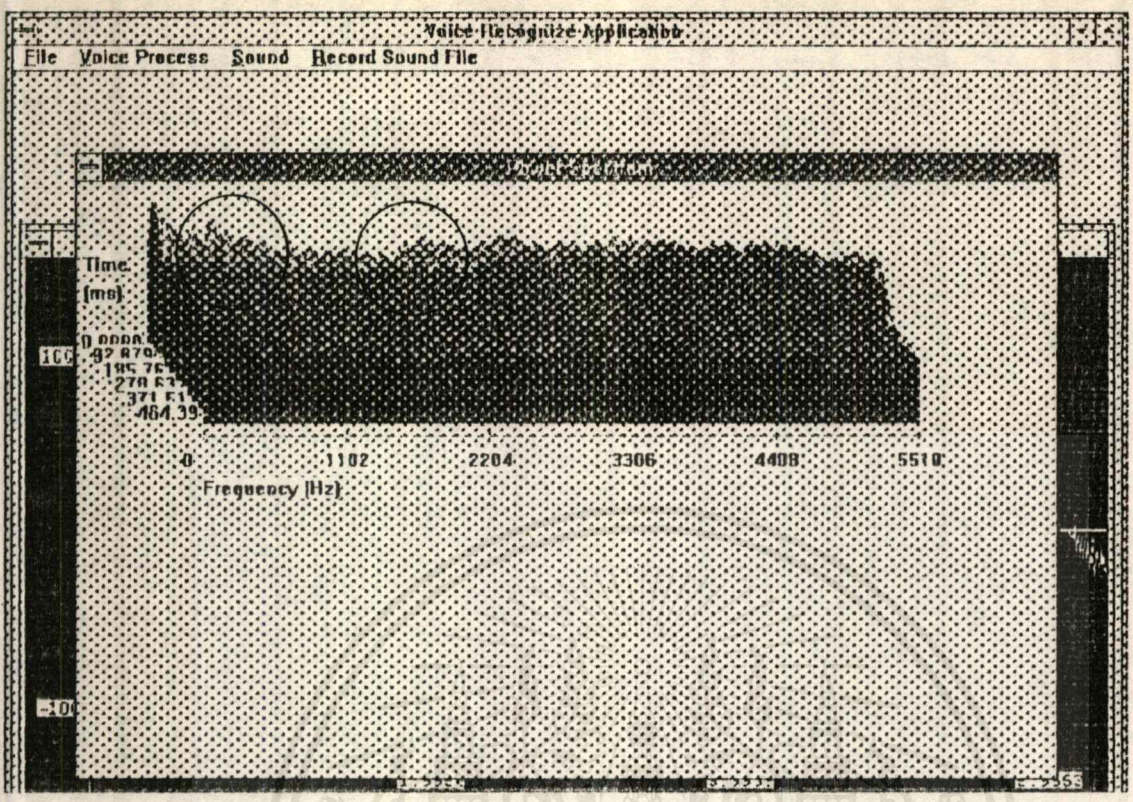
คำว่า "สิบ" พูดโดยบุคคลที่ 2 ครั้งที่ 1



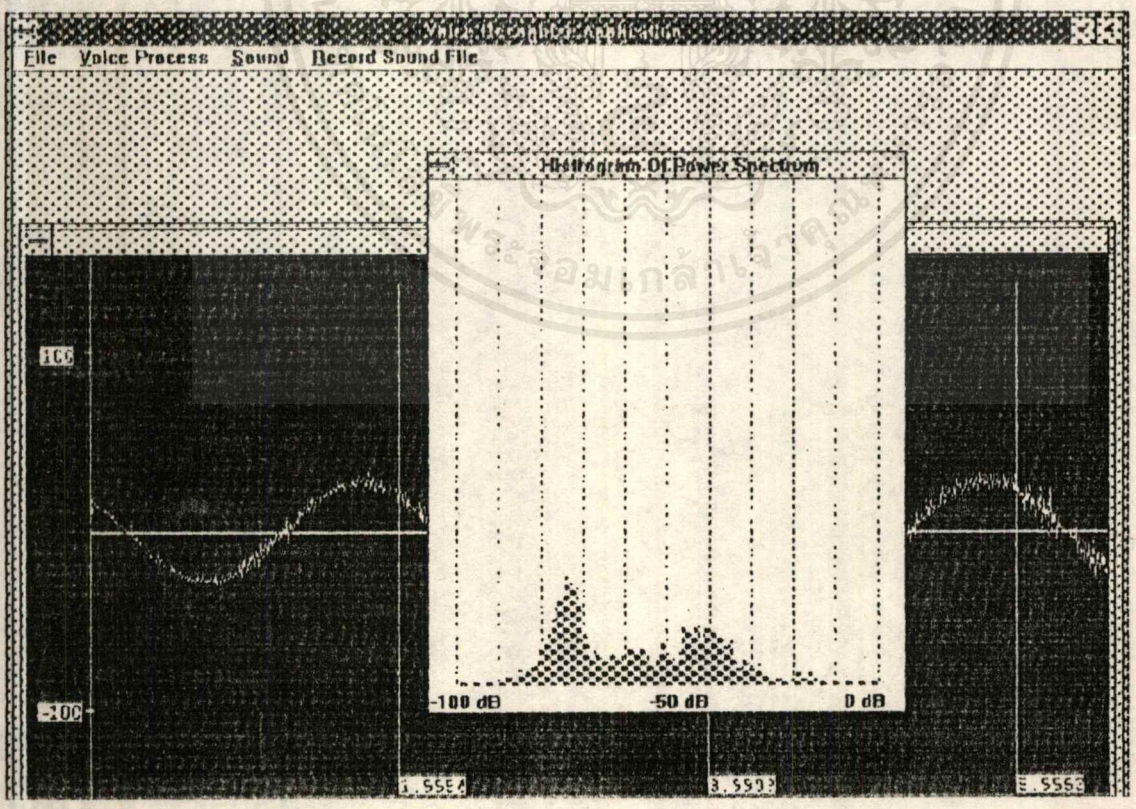


คำว่า "สิบ" พูดโดยบุคคลที่ 2 ครั้งที่ 2





คำว่า "สิบ" พูดโดยบุคคลที่ 2 ครั้งที่ 3



บทสรุป

จากบทความการวิจัยทั้งหมดที่กล่าวมาแล้ว ได้แสดงให้เห็นถึงขั้นตอน ในการที่จะทำให้ คอมพิวเตอร์สามารถ รับรู้คำสั่งได้โดยผ่านทางเสียงพูด ซึ่งประกอบไปด้วยหลายขั้นตอน ดังได้กล่าวมาแล้ว จากการศึกษาตามโครงงานนี้ ทำให้สามารถ แยกความแตกต่าง ของเสียงได้ ซึ่งจะนำไปใช้ในการ Recognized ต่อไป สำหรับปัญหา ของการตรวจรู้จำเสียง นั้น จะเป็นในเรื่องของ สัญญาณรบกวนของเสียง ที่อาจเกิดขึ้นจากเสียงที่ไม่ต้องการ หรืออาจเกิดขึ้นจาก ขั้นตอนในการแปลงสัญญาณเสียง ให้เป็นสัญญาณทางดิจิตอล แล้วเกิด ข้อมูลที่ผิดผลาดขึ้น ซึ่งจากสัญญาณรบกวนดังกล่าวนั้นต้องมีการกำจัดออกไปในหลายขั้นตอน นอกจากนี้อีกปัญหาหนึ่งคือ ความคล้ายคลึงกันของเสียง ทำให้ คอมพิวเตอร์ ไม่สามารถ หาความแตกต่าง ของเสียงเหล่านั้นได้

จากการศึกษาถึงการ การแปลงสัญญาณเสียง ที่อยู่ในรูปโดเมนของเวลา ให้อยู่ในโดเมนของความถี่และเวลา นั้นต้องใช้การคำนวณอย่างมาก จึงจำเป็นต้องใช้เครื่องคอมพิวเตอร์ที่มี ความเร็วสูงพอสมควร จึงจะสามารถคำนวณได้ทันตามความต้องการ

ภาคผนวก
Program Listing

```
/*************************************************************************
        PROGRAM: Voice.c

        PURPOSE: Voice Processer

*************************************************************************/
#include <windows.h>
#include <mmsystem.h>
#include "sbtest1.h"
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <io.h>
#include <math.h>
#include <sys\types.h>
#include <sys\stat.h>
#include <fcntl.h>

static HANDLE hInst;
static float sqrarg;

HANDLE hAccTable;               /* handle to accelerator table */
HWND hMainFrame;
HWND hPaintWnd;
HWND hSpctrmWnd;
HWND hScrollWnd;
HWND hFreqWnd;
HWND hSoundWnd;
HWND hHistroWnd;

FARPROC  lpProcAbout;
/* Additional includes needed for the fstat() function */
#define MAXWIDTH 60
char FileName[128];
char PathName[128];
```

```
char OpenName[128];
char DefPath[128];
char DefSpec[13] = "*.wav";
char DefExt[] = ".wav";
char str[255];
char soundstr[128];
static char szAppName[]="Draw";

HANDLE hEditBuffer;                 /* handle to editing buffer      */
HANDLE hOldBuffer;                  /* old buffer handle             */
HANDLE hHourGlass;                  /* handle to hourglass cursor    */
HANDLE hSaveCursor;                 /* current cursor handle         */
int hFile;                          /* file handle                   */
int count;                          /* number of chars read or written */
PSTR pBuffer;                       /* address of read/write buffer  */
OFSTRUCT OfStruct;                  /* information from OpenFile()   */
struct stat FileStatus;             /* information from fstat()      */
BOOL bChanges = FALSE;              /* TRUE if the file is changed   */
BOOL bSaveEnabled = FALSE;          /* TRUE if text in the edit buffer */
BOOL bOpen = FALSE;
LPSTR pEditBuffer;
LPSTR datafour;                     /* address of the edit buffer    */
RECT Rect;                          /* dimension of the client window */
HWND hwnd;                          /* handle to main window         */
int IOStatus;
int nHSize=1;
int nVSize =0;
int nFTime = 0;
int count = 0;
int nMaxRange=30000;

char Untitled[] =                   /* default window title          */
        "Edit File - (untitled)";
float p[257];
float Oldp[257];
```

```
int             aHisBuffer[] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
                 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
                 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
                 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
                 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
                 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
                 0,0,0,0 };


/*************************************************************************
   FUNCTION : Message(HWND,LPSTR)
   PURPOSE  : Display Messagebox
*************************************************************************/
int Message(HWND hwndMessage,LPSTR lpMessage)
{
   return(MessageBox(hwndMessage,lpMessage,"MessageBox",MB_OK|MB_APPLMODAL|
      MB_ICONSTOP));
}
/*************************************************************************

      FUNCTION: WinMain(HANDLE, HANDLE, LPSTR, int)

      PURPOSE: calls initialization function, processes message loop

*************************************************************************/
int PASCAL WinMain(hInstance, hPrevInstance, lpCmdLine, nCmdShow)
HANDLE hInstance;
HANDLE hPrevInstance;
LPSTR lpCmdLine;
int nCmdShow;
{
      MSG msg;
  hInst = hInstance;
  if(!InitClass())
      {
```

```
            Message(NULL,"Init Class Error");
            return(FALSE);
        }
        if (!hPrevInstance)
            if (!InitMainFrame())
            {
                Message(NULL,"InitApplication Error");
                return (FALSE);
            }
        while (GetMessage(&msg, NULL, NULL, NULL)) {
        /* Only translate message if it is not an accelerator message */
                TranslateMessage(&msg);
                DispatchMessage(&msg);
        }
        lpCmdLine = lpCmdLine;
        nCmdShow = nCmdShow;
        return (msg.wParam);
}
/*****************************************************************************
        Function : InitClass(void);
*****************************************************************************/
BOOL InitClass(void)
{
        WNDCLASS  wc;
        RECT  Rect;
        short   xScreen,yScreen;
        wc.style = CS_HREDRAW|CS_VREDRAW;
        wc.lpfnWndProc = MainWndProc;
        wc.cbClsExtra = 0;
        wc.cbWndExtra = 0;
        wc.hInstance = hInst;
        wc.hIcon = LoadIcon(hInst, "sbtest");
        wc.hCursor = LoadCursor(NULL, IDC_ARROW);
        wc.hbrBackground = GetStockObject(LTGRAY_BRUSH);
        wc.lpszMenuName =  "EditFileMenu";
```

```
wc.lpszClassName = "EditFileWClass";
if(!RegisterClass(&wc))
        return  FALSE;

wc.style = CS_HREDRAW|CS_VREDRAW;
wc.lpfnWndProc = PaintWndProc;
wc.cbClsExtra = 0;
wc.cbWndExtra = 0;
wc.hInstance = hInst;
wc.hIcon = NULL;
wc.hCursor = LoadCursor(NULL,IDC_ARROW );
wc.hbrBackground = GetStockObject(LTGRAY_BRUSH);
wc.lpszMenuName =  NULL;
wc.lpszClassName = "PaintWClass";
if(!RegisterClass(&wc))
        return  FALSE;

wc.style = CS_HREDRAW|CS_VREDRAW;
wc.lpfnWndProc = SpctrmWndProc;
wc.cbClsExtra = 0;
wc.cbWndExtra = 0;
wc.hInstance = hInst;
wc.hIcon = NULL;
wc.hCursor = LoadCursor(NULL,IDC_ARROW );
wc.hbrBackground = GetStockObject(LTGRAY_BRUSH);
wc.lpszMenuName =  NULL;
wc.lpszClassName = "SpctrmWClass";
if(!RegisterClass(&wc))
        return  FALSE;

wc.style = CS_HREDRAW|CS_VREDRAW;
wc.lpfnWndProc = HistroWndProc;
wc.cbClsExtra = 0;
wc.cbWndExtra = 0;
wc.hInstance = hInst;
```

```
        wc.hIcon = NULL;
        wc.hCursor = LoadCursor(NULL,IDC_ARROW );
        wc.hbrBackground = GetStockObject(WHITE_BRUSH);
        wc.lpszMenuName =  NULL;
        wc.lpszClassName = "HistroWClass";
        if(!RegisterClass(&wc))
                return  FALSE;

        return (TRUE);
}
/*************************************************************************
                Function: InitMainFrame(void)
*************************************************************************/
BOOL InitMainFrame(void)
{
   WNDCLASS  wc;
   RECT      Rect;
   short     xScreen,yScreen;
   xScreen = GetSystemMetrics (SM_CXSCREEN);
   yScreen = GetSystemMetrics (SM_CYSCREEN);
   hwnd = CreateWindow(
                "EditFileWClass",
                "Voice Recognize Application",
                WS_OVERLAPPEDWINDOW|WS_THICKFRAME,
                0,
                0,
                xScreen,
                yScreen,
                NULL,
                NULL,
                hInst,
                NULL
        );
        if (!hwnd)
                return FALSE;
```

```
        lpProcAbout = MakeProcInstance(About, hInst);
        DialogBox(hInst, "About",hMainFrame , lpProcAbout);
        FreeProcInstance(lpProcAbout);
        ShowWindow(hwnd, SW_SHOWNORMAL);
        UpdateWindow(hwnd);
        hMainFrame = hwnd;
    return TRUE;
}
/*************************************************************************
        Function: InitPaint(void)
*************************************************************************/
BOOL InitPaintWnd(void)
{
        WNDCLASS  wc;
        RECT  Rect;
        short    xScreen,yScreen;

        xScreen = GetSystemMetrics (SM_CXSCREEN);
        yScreen = GetSystemMetrics (SM_CYSCREEN);
        hwnd = CreateWindow("PaintWClass",FileName,
                WS_POPUP|WS_THICKFRAME|WS_BORDER|WS_SYSMENU |
WS_CAPTION |
                WS_HSCROLL,
                10,
                yScreen-450,
                xScreen-20,
                yScreen-165,
                hMainFrame,
                NULL,
                hInst,
                NULL);
        if(!hwnd)
                return (FALSE);
        ShowWindow(hwnd, SW_SHOWNORMAL);
        UpdateWindow(hwnd);
```

```
        hPaintWnd = hwnd;
        return TRUE;
}
/*************************************************************************
        Function : InitSpectrm(void);
*************************************************************************/
BOOL InitSpctrmWnd(void)
{
        WNDCLASS  wc;
        RECT  Rect;

        hwnd = CreateWindow("SpctrmWClass","Power Spectrum",
                WS_POPUP|WS_BORDER|WS_CAPTION|WS_SYSMENU,
                50,100,
                700,450,
                hMainFrame,
                NULL,
                hInst,
                NULL);
        if(!hwnd)
                return (FALSE);
        ShowWindow(hwnd,SW_SHOWNORMAL);
        UpdateWindow(hwnd);
        hSpctrmWnd = hwnd;
        return TRUE;
}
/*************************************************************************
        Function: InitHistro(void);
*************************************************************************/
BOOL InitHistroWnd(void)
{
        WNDCLASS  wc;
        RECT  Rect;

        hwnd = CreateWindow("HistroWClass","Histrogram Of Power Spectrum",
```

```
            WS_POPUP|WS_BORDER|WS_CAPTION|WS_SYSMENU,
            300,100,
            342,400,
            hMainFrame,
            NULL,
            hInst,
            NULL);
    if(!hwnd)
            return (FALSE);
    ShowWindow(hwnd,SW_SHOWNORMAL);
    UpdateWindow(hwnd);
    hHistroWnd = hwnd;
    return TRUE;
}
/****************************************************************************
    FUNCTION: MainWndProc(HWND, unsigned, WORD, LONG)
    PURPOSE:  Processes messages

    MESSAGES:
            WM_COMMAND    - application menu (About dialog box)
            WM_DESTROY    - destroy window
            WM_SIZE       - window size has changed
            WM_QUERYENDSESSION - willing to end session?
            WM_ENDSESSION - end Windows session
            WM_CLOSE      - close the window
            WM_SIZE       - window resized
    COMMENTS:

            WM_COMMAND processing:
                    IDM_OPEN - query to save current file if there is one and it
                                    has been changed, open a new file.
                    IDM_EXIT - query to save current file if there is one and it
                                    has been changed, then exit.
                    IDM_ABOUT - display "About" box.
```

When more then one string needs to be sent to a message box, sprintf() is used to combine the strings into str[], and then str[] is passed to the MessageBox() function. A message box string cannot exceed 255 characters, but may contain \n to generate separate lines.

After the size of the file is determined, only enough memory to store the file is allocated for the Edit buffer. The edit control will automatically expand this memory as needed. Once the file has been read into the edit buffer, unlock the memory. Use whatever was obtained from the read() function, even if an error occured. This allows partial salvage of a file with a bad sector.

```
*************************************************************************/
LRESULT CALLBACK MainWndProc(hWnd, message, wParam, lParam)
HWND hWnd;
UINT message;
WPARAM wParam;
LPARAM lParam;
{
        FARPROC lpOpenDlg, lpSaveAsDlg;  /* return value from SaveAsDlg() */
        int Success;                     /* result of file i/o     */


switch (message) {
        case WM_COMMAND:
                switch (wParam) {
                        case IDM_ABOUT:
                                lpProcAbout = MakeProcInstance(About, hInst);
                                DialogBox(hInst, "About", hWnd, lpProcAbout);
                                FreeProcInstance(lpProcAbout);
                                break;
                        case IDM_OPEN:
                                if(bOpen == TRUE)
                                        MessageBox(hMainFrame,"Can't Open File. Close
```

```
            Before",," Error Message",
            MB_OK|MB_SYSTEMMODAL |MB_ICONSTOP);
    else {
    bOpen = TRUE;
            lpOpenDlg = MakeProcInstance((FARPROC)
                        OpenDlg, hInst);
            /* Open the file and get its handle */
            hFile = DialogBox(hInst, "Open", hWnd, lpOpenDlg);
            FreeProcInstance(lpOpenDlg);
            if (!hFile)
                    return (NULL);
            /* Allocate edit buffer to the size of the file + 1 */
            hEditBuffer =LocalAlloc(LMEM_MOVEABLE |
             LMEM_ZEROINIT,(WORD)(FileStatus.st_size+1));
            if (!hEditBuffer) {
                    MessageBox(hWnd, "Not enough memory.",
                            NULL, MB_OK | MB_ICONHAND);
                    return (NULL);
            }
            pEditBuffer = LocalLock(hEditBuffer);
             IOStatus = read(hFile, pEditBuffer, (WORD)

                        FileStatus.st_size);
            close(hFile);
            nMaxRange = IOStatus;
            /* # bytes read must equal file size */
            if (IOStatus != (int) FileStatus.st_size) {
                    sprintf(str, "Error reading %s.", FileName);
                    MessageBox(hWnd, str, NULL, MB_OK |
                            MB_ICONEXCLAMATION);
            }
            LocalUnlock(hEditBuffer);
            /* Set up a new buffer and window title */
            if (!InitPaintWnd())
                    return (FALSE);
```

```
                    LocalFree(hEditBuffer);
            }break;
            case IDM_EXIT:
                    DestroyWindow(hWnd);
                    break;
            case IDM_FREQ:
                    MessageBox(NULL,"Frequency",NULL,MB_OK);
                    break;
            case IDM_SPECTRUM:
                    if(bOpen!=TRUE)
                            MessageBox(hMainFrame,"File not ready",
                            " Error Message",MB_OK|MB_SYSTEMMODAL |
                             MB_ICONSTOP);
                    else {
                            if(!InitSpctrmWnd())
                                    return (FALSE);
                    }
            break;
            case IDM_HISTRO:
                    DestroyWindow(hSpctrmWnd);
                    if(!InitHistroWnd())
                            return (FALSE);
            break;
            case IDM_PLAY:
                    lpProcAbout = MakeProcInstance(PlaySound, hInst);
                    DialogBox(hInst, "PlaySound", hWnd, lpProcAbout);
                    FreeProcInstance(lpProcAbout);
                    break;
            case IDM_RECORD:
                    lpProcAbout = MakeProcInstance(Record,hInst);
                    DialogBox(hInst,"Record",hWnd,lpProcAbout);
                    FreeProcInstance(lpProcAbout);
            break;
    }
    break;
```

```
            case WM_CLOSE:                  /* message: close the window   */
                    DestroyWindow(hWnd);
                    break;
            case WM_DESTROY:
                    sndPlaySound("bye.wav",SND_SYNC);
                    PostQuitMessage(0);
                    break;
            default:
                    return (DefWindowProc(hWnd, message, wParam, lParam));
        }
        return (NULL);
}
/*************************************************************************/
/*                              Display Graph                            */
/*************************************************************************/
LRESULT CALLBACK PaintWndProc (hPaintWnd,iMessage,wParam,lParam)
HWND        hPaintWnd;
UINT        iMessage;
WPARAM      wParam;
LPARAM      lParam;
{
static short    nDrawingMode = IDM_DISPLAY;
HDC         hDC;
HMENU       hMenu;
HPEN        hPenWhite,hPenBlue,hPenGreen;

RECT        rect;
int         temp,i,p,x;

        static short    nHscrollMax;
        static short    nHscrollPos;
        char            szBuffer[80];
        HDC             hdc;
        PAINTSTRUCT     ps;
        short           nVscrollInc,nHscrollInc;
```

```
short           nPaintBeg,nPaintEnd;
long float      fTime,fTime1,fTime2;
char            sTime[8];
TEXTMETRIC          tm;

switch(iMessage) {
        case WM_COMMAND:
                switch(wParam) {
                        case IDM_DISPLAY:
                                hMenu = GetMenu(hPaintWnd);
                                CheckMenuItem(hMenu,nDrawingMode,
                                                MF_UNCHECKED);
                                nDrawingMode = wParam;
                                CheckMenuItem(hMenu,nDrawingMode,
                                                MF_CHECKED);
                                InvalidateRect (hPaintWnd,NULL,FALSE);
                        break;
                        case IDM_HISTRO:
                                hMenu = GetMenu(hPaintWnd);
                                CheckMenuItem(hMenu,nDrawingMode,
                                                MF_UNCHECKED);
                                nDrawingMode = wParam;
                                CheckMenuItem(hMenu,nDrawingMode,
                                                MF_CHECKED);
                                InvalidateRect (hPaintWnd,NULL,FALSE);
                        break;
                        case IDM_QUIT:
                                DestroyWindow(hPaintWnd);
                        break;
                }
                break;
          case WM_SIZE:
                nHscrollMax = max (0,nMaxRange);
                nHscrollPos = min (nHscrollPos,nHscrollMax);
                SetScrollRange(hPaintWnd,SB_HORZ,0,nHscrollMax,FALSE);
```

```
            SetScrollPos(hPaintWnd,SB_HORZ,nHscrollPos,TRUE);
    break;
    case WM_HSCROLL:
    switch(wParam)
    {
            case SB_LINEUP:
                    nHscrollInc = -20;
                    nFTime +=nHscrollInc;
                    if(nFTime < 0)
                            nFTime = 0;
            break;
            case SB_LINEDOWN:
                    nHscrollInc = 20;
                    nFTime +=nHscrollInc;
            break;
            case SB_PAGEUP:
                    nHscrollInc = -8;
            break;
            case SB_PAGEDOWN:
                    nHscrollInc = 8;
            break;
            case SB_THUMBPOSITION:
                    nHscrollInc = LOWORD (lParam) - nHscrollPos;
                    nFTime += nHscrollInc;
            break;
            default:
                    nHscrollInc = 0;
    }
    if(nHscrollInc == max(-nHscrollPos,min(nHscrollInc,nHscrollMax -
                            nHscrollPos)))
    {
            nHscrollPos += nHscrollInc;
            ScrollWindow (hPaintWnd,-rect.right*nHscrollInc,0,NULL,NULL);
            SetScrollPos(hPaintWnd,SB_HORZ,nHscrollPos,TRUE);
            UpdateWindow(hPaintWnd);
```

```
}
break;
 case WM_PAINT:
        hDC = BeginPaint (hPaintWnd,&ps);
        hPenWhite = CreatePen (0,1,RGB(250,250,250));
        hPenBlue  = CreatePen(0,1,RGB(250,250,0));
        hPenGreen =  CreatePen(0,1,RGB(250,250,250));
        SetMapMode(hDC,MM_ANISOTROPIC);
        GetClientRect(hPaintWnd,&rect);
        SetViewportExt(hDC,rect.right,rect.bottom);
        SetWindowExt(hDC,rect.right,rect.bottom);
        switch(nDrawingMode){
                case IDM_DISPLAY:
                        FillRect(hDC,&rect,
                                GetStockObject(BLACK_BRUSH));
                        SelectObject(hDC,hPenWhite);
                        MoveTo(hDC,45,(rect.bottom/10)*5);
                        LineTo(hDC,rect.right,(rect.bottom/10)*5);
                        MoveTo(hDC,45,rect.top);
                        LineTo(hDC,45,rect.bottom);
                        MoveTo(hDC,42,rect.bottom/2+(128));
                        LineTo(hDC,48,rect.bottom/2+128);
                        MoveTo(hDC,42,rect.bottom/2-128);
                        LineTo(hDC,48,rect.bottom/2-128);
                        MoveTo(hDC,265,rect.top+20);
                        LineTo(hDC,265,rect.bottom-20);
                        MoveTo(hDC,485,rect.top+20);
                        LineTo(hDC,485,rect.bottom-20);
                        MoveTo(hDC,705,rect.top+20);
                        LineTo(hDC,705,rect.bottom-20);
                        SelectObject(hDC,
                                GetStockObject(ANSI_FIXED_FONT));
                        SetTextColor(hDC,hPenGreen);
                switch(nVSize) {
                        case 0:
```

```
            TextOut(hDC,10,rect.bottom/2-132,"100",3);
            TextOut(hDC,7,rect.bottom/2+123,"-100",4);
        break;
        case 26:
                TextOut(hDC,20,rect.bottom/2-132,"80",2);
            TextOut(hDC,15,rect.bottom/2+123,"-80",3);
        break;
        case 52:
                TextOut(hDC,19,rect.bottom/2-132,"60",2);
            TextOut(hDC,14,rect.bottom/2+123,"-60",3);
        break;
        case 78:
                TextOut(hDC,20,rect.bottom/2-132,"40",2);
            TextOut(hDC,13,rect.bottom/2+123,"-40",3);
        break;
        case 104:
        TextOut(hDC,20,rect.bottom/2-132,"20",2);
        TextOut(hDC,14,rect.bottom/2+123,"-20",3);
        break;
    }
        fTime = (nHscrollPos+220.00)/110.25;
        gcvt(fTime,8,sTime);
        TextOut(hDC,265,rect.bottom-20,sTime,6);
        fTime1 = (nHscrollPos +440.00)/110.25;
        gcvt(fTime1,8,sTime);
        TextOut(hDC,485,rect.bottom-20,sTime,6);
        fTime2 = (nHscrollPos +660.00)/110.25;
        gcvt(fTime2,8,sTime);
        TextOut(hDC,705,rect.bottom-20,sTime,6);
        SelectObject(hDC,hPenBlue);
        MoveTo(hDC,45,rect.bottom/2);
        p=45;
        for(i=45+ nHscrollPos;i<IOStatus;i++){
                p+=nHSize;
                if( pEditBuffer[i]==127){
```

```
                                LineTo(hDC,p,rect.bottom/2);
                                MoveTo(hDC,p,rect.bottom/2);
                        }
                        else if(pEditBuffer[i] <0 ) {
                        LineTo(hDC,p,rect.bottom/2-(pEditBuffer[i]
                                +128+nVSize));
                        MoveTo(hDC,p,rect.bottom/2-(pEditBuffer[i]
                                +128+nVSize));
                        }
                        else {
                                LineTo(hDC,p,rect.bottom/2
                                +(-(pEditBuffer[i]-128-nVSize)));
                                MoveTo(hDC,p,rect.bottom/2+
                                (-(pEditBuffer[i]-128-nVSize)));
                                }
                        }
                        DeleteObject(hPenWhite);
                        DeleteObject(hPenBlue);
                        DeleteObject(hPenGreen);
                        DeleteObject(SelectObject(hDC,
                                GetStockObject(ANSI_VAR_FONT)));
                        EndPaint(hPaintWnd,&ps);
                        break;
                }
        break;
        case WM_CLOSE:
                bOpen = FALSE;
                for(i=0;i<100;i++)
                        aHisBuffer[i] = 0;
        default:
                return DefWindowProc (hPaintWnd,iMessage,wParam,lParam);
        }
    return 0L;
}
/******************** SpctrmWndProc ************************************/
```

```
LRESULT CALLBACK SpctrmWndProc (hSpctrmWnd,iMessage,wParam,lParam)
HWND            hSpctrmWnd;
UINT            iMessage;
WPARAM          wParam;
LPARAM          lParam;
{
HDC             hDC;
HMENU           hMenu;
HPEN            hPenBlue,hPenRed;

RECT            rect;
int             temp,i=0,x;
char            szBuffer[80];
HDC             hdc;
PAINTSTRUCT     ps;
short           nPaintBeg,nPaintEnd;
long float      fTime,fTime1,fTime2;
char            sTime[8];
int             iPageNum,j,k,color=0,iHistro;
HBRUSH          hBrush;
TEXTMETRIC      tm;

        switch(iMessage)  {
                case WM_PAINT:
                        hDC = BeginPaint (hSpctrmWnd,&ps);
                        SetMapMode(hDC,MM_ANISOTROPIC);
                        GetClientRect(hSpctrmWnd,&rect);
                        SetViewportExt(hDC,rect.right,rect.bottom);
                        SetWindowExt(hDC,rect.right,rect.bottom);
                        hHourGlass= LoadCursor(NULL,IDC_WAIT);
                        SetCursor(hHourGlass);
                        for(i=0;i<100;i++)          /* Clear Histrogram */
                                aHisBuffer[i] = 0;
                         datafour = pEditBuffer;
                         datafour+= 45;
```

```
iPageNum = (IOStatus-45)/1024;
hPenRed =  CreatePen(0,1,RGB(255,0,0));
hPenBlue =  CreatePen(0,1,RGB(0,0,255));
hBrush = CreateSolidBrush(RGB(0,0,255));
for(j=0; j<iPageNum; j++) {
        SelectObject(hDC,hPenRed);
        spctrm(datafour,p,256,1);
        for(k=1;k<=256;k++) {
                p[k]= 10*log10(p[k]);
                iHistro = (int)(-p[k]);
                aHisBuffer[iHistro]++;
        }
         datafour +=1024 ;
         i=0;
        fTime = j*92.8798;
        gcvt(fTime,6,sTime);
        if(j==0) strcpy(sTime,"0.000000");
TextOut(hDC,rect.left+(j*8)+3,rect.top+(j*10)+105,sTime,6);
        MoveTo(hDC,rect.left+(j*8)+TAB,rect.top+(-p[1])+(j*10));
        for(k=2;k<=256;k++) {
                i+=2;
        LineTo(hDC,rect.left+i+(j*8)+TAB,rect.top+(-p[k])+(j*10));
                if(j!=0) {
                MoveTo(hDC,rect.left+i+((j-1)*8)+
                        TAB,rect.top+(-Oldp[k])+((j-1)*10));
                LineTo(hDC,rect.left+i+(j*8)+
                        TAB,rect.top+(-p[k])+(j*10));
                }
        }
        SelectObject(hDC,hPenBlue);
        LineTo(hDC,rect.left+i+(j*8)+TAB,rect.top+(j*10)+120);
        LineTo(hDC,rect.left+(j*8)+TAB,rect.top+(j*10)+120);
        SelectObject(hDC,hPenRed);
        LineTo(hDC,rect.left+(j*8)+TAB,rect.top+(-p[1])+(j*10));
        SelectObject(hDC,hBrush);
```

```
                        ExtFloodFill(hDC,rect.left+(j*8)+TAB+1,rect.top+(j*10)+
                                119,RGB(192,192,192),FLOODFILLSURFACE);
                        for(k=0; k<=256; k++)
                                Oldp[k] = p[k];
                }
                MoveTo(hDC,rect.left+((j-1)*8)+TAB,rect.top+(j*10)+120);
        LineTo(hDC,rect.left+((j-1)*8)+TAB+(2*256),rect.top+(j*10)+120);
                x=rect.left+((j-1)*8)+TAB;
                k=rect.top+(j*10)+120;
                for(i=0; i<=5; i++) {
                        LineScale(hDC,x,k,i);
                }
                TextOut(hDC,rect.left+5,rect.top+50,"Time",4);
                TextOut(hDC,rect.left+5,rect.top+70,"(ms)",4);
                TextOut(hDC,x,k+30,"Frequency (Hz)",14);
                DeleteObject(hPenRed);
                DeleteObject(hPenBlue);
                DeleteObject(SelectObject(hDC,
                        GetStockObject(ANSI_VAR_FONT)));
                DeleteObject(hBrush);
                EndPaint(hSpctrmWnd,&ps);
        break;
        default:
                return DefWindowProc (hSpctrmWnd,iMessage,wParam,lParam);
    }
    return 0L;
}
/*****************************************************************************
    Function: HistroWndProc(hHistroWnd,iMessage,wParam,lParam)
*****************************************************************************/
LRESULT CALLBACK HistroWndProc(hHistroWnd,iMessage,wParam,lParam)
HWND            hHistroWnd;
UINT            iMessage;
WPARAM          wParam;
LPARAM          lParam;
```

```
{
HDC             hDC;
RECT            rect;
PAINTSTRUCT     ps;
HPEN            hPenRed,hPenGreen;
int             i,iIndex=0;
        switch(iMessage) {
                case WM_PAINT:
                        hDC = BeginPaint (hHistroWnd,&ps);
                        SetMapMode(hDC,MM_ANISOTROPIC);
                        GetClientRect(hHistroWnd,&rect);
                        SetViewportExt(hDC,rect.right,rect.bottom);
                        SetWindowExt(hDC,rect.right,rect.bottom);
                        hPenRed = CreatePen(1,2,RGB(255,0,0));
                        SelectObject(hDC,hPenRed);
                        MoveTo(hDC,rect.left+20,rect.bottom-20);
                        LineTo(hDC,rect.right-20,rect.bottom-20);
                        DeleteObject(hPenRed);
                        hPenRed = CreatePen(2,1,RGB(255,0,0));
                        SelectObject(hDC,hPenRed);
                        for(i=0;i<=10;i++) {
                                MoveTo(hDC,((rect.right-40)/10*i)+20,rect.bottom-20);
                                LineTo(hDC,((rect.right-40)/10*i)+20,rect.top);
                        }
                        DeleteObject(hPenRed);
                        hPenGreen = CreatePen(1,3,RGB(0,255,0));
                        SelectObject(hDC,hPenGreen);
                        TextOut(hDC,2,rect.bottom-15,"-100 dB",7);
                        TextOut(hDC,158,rect.bottom-15,"-50 dB",6);
                        TextOut(hDC,296,rect.bottom-15,"0 dB",4);
                        for(i=0;i<300;i+=3) {
                                MoveTo(hDC,((rect.right-20)-i),rect.bottom-21);
                                LineTo(hDC,((rect.right-20)-i),rect.bottom-21-
                                        (aHisBuffer[iIndex]));
                                        iIndex++;
```

```
                }
                DeleteObject(hPenGreen);
                EndPaint(hHistroWnd,&ps);
            break;
            default:
                return DefWindowProc(hHistroWnd,iMessage,wParam,lParam);
            }
        return 0L;
}
/*************************************************************************
            Function: LineScale(HDC hDC,int x,int y,int j);
*************************************************************************/
void LineScale(HDC hDC,int x,int y,int j)
{
        unsigned long value;
        char Buffer[4];
            MoveTo(hDC,x+(j*102),y-3);
            LineTo(hDC,x+(j*102),y+3);
            value=j*1102;
            ultoa(value,Buffer,10);
            TextOut(hDC,(x+(j*102))-15,y+10,Buffer,(j==0)?1:4);
}
/*************************************************************************

        FUNCTION: OpenDlg(HWND, unsigned, WORD, LONG)

        PURPOSE: Let user select a file, and open it.

*************************************************************************/
HANDLE FAR PASCAL OpenDlg(hDlg, message, wParam, lParam)
HWND hDlg;
unsigned message;
WORD wParam;
LONG lParam;
{
```

```
    WORD index;
    PSTR pTptr;
    HANDLE hFile;
    switch (message) {
        case WM_COMMAND:
            switch (wParam) {
                case IDC_LISTBOX:
                    switch (HIWORD(lParam)) {
                        case LBN_SELCHANGE:
                        /* If item is a directory name, append "*.*" */
                        if (DlgDirSelect(hDlg, str, IDC_LISTBOX))
                            strcat(str, DefSpec);
                            SetDlgItemText(hDlg, IDC_EDIT, str);
                            SendDlgItemMessage(hDlg,
                                IDC_EDIT,
                                EM_SETSEL,
                                NULL,
                                MAKELONG(0, 0x7fff));
                        break;
                        case LBN_DBLCLK:
                        goto openfile;
                    }
                return (TRUE);
                case IDOK:
openfile:
                GetDlgItemText(hDlg, IDC_EDIT, OpenName, 128);
                if (strchr(OpenName, '*') || strchr(OpenName, '?')) {
                    SeparateFile(hDlg, (LPSTR) str, (LPSTR) DefSpec,
                        (LPSTR) OpenName);
                    if (str[0])
                        strcpy(DefPath, str);
                    ChangeDefExt(DefExt, DefSpec);
                    UpdateListBox(hDlg);
                    return (TRUE);
                }
```

```
            if (!OpenName[0]) {
                MessageBox(hDlg, "No filename specified.",
                    NULL, MB_OK | MB_ICONHAND);
                return (TRUE);
            }
            AddExt(OpenName, DefExt);     /* Open the file */
            if ((hFile = open(OpenName, O_BINARY,S_IREAD)) == 0)
            {
                sprintf(str, "Error %d opening %s.",OfStruct.nErrCode,
                    OpenName);
                MessageBox(hDlg, str, NULL,MB_OK |
                        MB_ICONHAND);
            }
            else {
            /* Make sure there's enough room for the file */
                fstat(hFile, &FileStatus);
                if (FileStatus.st_size > MAXFILESIZE) {
                sprintf(str,
                "Not enough memory to load %s.\n%s exceeds %ld
                bytes.",
                OpenName, OpenName, MAXFILESIZE);
                MessageBox(hDlg, str, NULL,MB_OK |
                        MB_ICONHAND);
                return (TRUE);
            }
            strcpy(FileName, OpenName);
            EndDialog(hDlg, hFile);
            return (TRUE);
        }
        return (TRUE);
        case IDCANCEL:
            EndDialog(hDlg, NULL);
            return (TRUE);
        }
        break;
```

```
            case WM_INITDIALOG:                    /* message: initialize   */
                    UpdateListBox(hDlg);
                    SetDlgItemText(hDlg, IDC_EDIT, DefSpec);
                    SendDlgItemMessage(hDlg,          /* dialog handle     */
                            IDC_EDIT,                 /* where to send message */
                            EM_SETSEL,                /* select characters     */
                            NULL,                     /* additional information */
                            MAKELONG(0, 0xffff));     /* entire contents     */
                    SetFocus(GetDlgItem(hDlg, IDC_EDIT));
                    return (FALSE); /* Indicates the focus is set to a control */
        }
        return FALSE;
}
/*****************************************************************************

        FUNCTION: UpdateListBox(HWND);

        PURPOSE: Update the list box of OpenDlg

*****************************************************************************/
void UpdateListBox(hDlg)
HWND hDlg;
{
        strcpy(str, DefPath);
        strcat(str, DefSpec);
        DlgDirList(hDlg, str, IDC_LISTBOX, IDC_PATH, 0x4010);
        /* To ensure that the listing is made for a subdir. of
         * current drive dir...
         */
        if (!strchr (DefPath, ':'))
        DlgDirList(hDlg, DefSpec, IDC_LISTBOX, IDC_PATH, 0x4010);
        /* Remove the '..' character from path if it exists, since this
         * will make DlgDirList move us up an additional level in the tree
         * when UpdateListBox() is called again.
         */
```

```
        if (strstr (DefPath, "..'))
        DefPath[0] = '\0';
        SetDlgItemText(hDlg, IDC_EDIT, DefSpec);
}
/*****************************************************************************

        FUNCTION: ChangeDefExt(PSTR, PSTR);

        PURPOSE: Change the default extension

*****************************************************************************/
void ChangeDefExt(Ext, Name)
PSTR Ext, Name;
{
        PSTR pTptr;
        pTptr = Name;
        while (*pTptr && *pTptr != '.')
                pTptr++;
        if (*pTptr)
                if (!strchr(pTptr, '*') && !strchr(pTptr, '?'))
                        strcpy(Ext, pTptr);
}
/*****************************************************************************

        FUNCTION: SeparateFile(HWND, LPSTR, LPSTR, LPSTR)

        PURPOSE: Separate filename and pathname
*****************************************************************************/
void SeparateFile(hDlg, lpDestPath, lpDestFileName, lpSrcFileName)
HWND hDlg;
LPSTR lpDestPath, lpDestFileName, lpSrcFileName;
{
        LPSTR lpTmp;
        char  cTmp;
        lpTmp = lpSrcFileName + (long) lstrlen(lpSrcFileName);
```

```
        while (*lpTmp != ':' && *lpTmp != '\\' && lpTmp > lpSrcFileName)
                lpTmp = AnsiPrev(lpSrcFileName, lpTmp);
        if (*lpTmp != ':' && *lpTmp != '\\') {
                lstrcpy(lpDestFileName, lpSrcFileName);
                lpDestPath[0] = 0;
                return;
        }
        lstrcpy(lpDestFileName, lpTmp + 1);
        cTmp = *(lpTmp + 1);
        lstrcpy(lpDestPath, lpSrcFileName);
         *(lpTmp + 1) = cTmp;
        lpDestPath[(lpTmp - lpSrcFileName) + 1] = 0;
      hDlg = hDlg;
}
/*************************************************************************

        FUNCTION: AddExt(PSTR, PSTR);

        PURPOSE: Add default extension

*************************************************************************/
void AddExt(Name, Ext)
PSTR Name, Ext;
{
        PSTR pTptr;
        pTptr = Name;
        while (*pTptr && *pTptr != '.')
                pTptr++;
        if (*pTptr != '.')
                strcat(Name, Ext);
}
/*************************************************************************

        FUNCTION: CheckFileName(HWND, PSTR, PSTR)
```

```
        PURPOSE: Check for wildcards, add extension if needed

        COMMENTS:

                Make sure you have a filename and that it does not contain any
                wildcards.  If needed, add the default extension.  This function is
                called whenever your application wants to save a file.

*************************************************************************/
BOOL CheckFileName(hWnd, pDest, pSrc)
HWND hWnd;
PSTR pDest, pSrc;
{
        PSTR pTmp;
        if (!pSrc[0])
                return (FALSE);             /* Indicates no filename was specified */
        pTmp = pSrc;
        while (*pTmp) {                     /* Searches the string for wildcards */
                switch (*pTmp++) {
                        case '*':
                        case '?':
                                MessageBox(hWnd, "Wildcards not allowed.",
                                        NULL, MB_OK | MB_ICONEXCLAMATION);
                                return (FALSE);
                }
        }
        AddExt(pSrc, DefExt);               /* Adds the default extension if needed */
        if (OpenFile(pSrc, (LPOFSTRUCT) &OfStruct, OF_EXIST) >= 0) {
                sprintf(str, "Replace existing %s?", pSrc);
                if (MessageBox(hWnd, str, "EditFile",
                                MB_OKCANCEL | MB_ICONHAND) == IDCANCEL)
                        return (FALSE);
        }
        strcpy(pDest, pSrc);
        return (TRUE);
```

```
}
/*****************************************************************************

        FUNCTION: SetNewBuffer(HWND, HANDLE, PSTR)

        PURPOSE: Set new buffer for edit window

        COMMENTS:

            Point the edit window to the new buffer, update the window title, and
            redraw the edit window.  If hNewBuffer is NULL, then create an empty
            1K buffer, and return its handle.

*****************************************************************************/
void SetNewBuffer(hWnd, hNewBuffer, Title)
HWND hWnd;
HANDLE hNewBuffer;
PSTR Title;
{
        HANDLE hOldBuffer;
        hOldBuffer = (HANDLE) SendMessage(hMainFrame, EM_GETHANDLE, 0, 0L);
        LocalFree(hOldBuffer);
        if (!hNewBuffer)                    /* Allocates a buffer if none exists */
                hNewBuffer = LocalAlloc(LMEM_MOVEABLE | LMEM_ZEROINIT, 1);
        /* Updates the buffer and displays new buffer */
        SendMessage(hMainFrame, EM_SETHANDLE, hNewBuffer, 0L);
        /*SetWindowText(hWnd, Title);
        SetFocus(hMainFrame); */
        bChanges = FALSE;
        hWnd = hWnd;
      Title = Title;
}
/*****************************************************************************

        FUNCTION: About(HWND, unsigned, WORD, LONG)
```

```
        PURPOSE:  Processes messages for "About" dialog box

        MESSAGES:

                WM_INITDIALOG - initialize dialog box
                WM_COMMAND    - Input received

****************************************************************************/
BOOL FAR PASCAL About(hDlg, message, wParam, lParam)
HWND hDlg;
unsigned message;
WORD wParam;
LONG lParam;
{
        switch (message) {
                case WM_INITDIALOG:
                        sndPlaySound("mam.wav",SND_SYNC);
                        return (TRUE);
                case WM_CLOSE:
                   EndDialog(hDlg,TRUE);
                   return (TRUE);

                case WM_COMMAND:
                        if (wParam == ID_OK) {
                                EndDialog(hDlg, TRUE);
                                return (TRUE);
                        }
                        break;
        }
       lParam = lParam;
        return (FALSE);
}
/*****************************************************************
                Function:Playsound(hDlg,message,wParam,lParam);
```

```
*************************************************************************/
BOOL FAR PASCAL PlaySound(hDlg, message, wParam, lParam)
HWND        hDlg;
unsigned    message;
WORD        wParam;
LONG        lParam;
{
    switch (message) {
        case WM_INITDIALOG:
            return (TRUE);
        case WM_CLOSE:
            EndDialog(hDlg,TRUE);
            return (TRUE);
        case WM_COMMAND:
            switch(wParam)
            {
                case ID_PLAY:
                    sndPlaySound(FileName,SND_SYNC);
                    break;
                case ID_START:
                    sndPlaySound(FileName,SND_LOOP|SND_ASYNC);
                    break;
                case ID_STOP:
                    sndPlaySound(NULL,0);
                break;
                case IDM_EXIT:
                    sndPlaySound(NULL,0);
                    EndDialog(hDlg,TRUE);
        return (TRUE);
        }
    break;
    }
    lParam = lParam;
    return (FALSE);
}
```

```
/*************************************************************************
            Function: Record(hDlg,message,wParam,lParam)
*************************************************************************/
BOOL FAR PASCAL Record(hDlg, message, wParam, lParam)
HWND        hDlg;
unsigned    message;
WORD        wParam;
LONG        lParam;
{
    switch (message) {
        case WM_INITDIALOG:
            return (TRUE);
        case WM_CLOSE:
            EndDialog(hDlg,TRUE);
          return (TRUE);
        case WM_COMMAND:
            switch(wParam)
            {
            case ID_RECORD:
                recordWAVEFile(1000,hDlg);
                break;
            case IDCANCEL:
                EndDialog(hDlg,TRUE);
                break;
             }
            break;
     }
    lParam = lParam;
    return (FALSE);
}
/*************************************************************/
HANDLE FAR PASCAL SaveDlg(hDlg, message, wParam, lParam)
HWND hDlg;
unsigned message;
WORD wParam;
```

```
LONG lParam;
{
        WORD index;
        PSTR pTptr;
        HANDLE hFile;
        switch (message) {
                case WM_COMMAND:
                        switch (wParam) {
                                case IDC_LISTBOX:
                                        switch (HIWORD(lParam)) {
                                                case LBN_SELCHANGE:
                                                /* If item is a directory name, append "*.*" */
                                                if (DlgDirSelect(hDlg, soundstr,
                                                                IDC_LISTBOX))
                                                strcat(soundstr, DefSpec);
                                                SetDlgItemText(hDlg, IDC_EDIT, soundstr);
                                                SendDlgItemMessage(hDlg,
                                                        IDC_EDIT,
                                                        EM_SETSEL,
                                                        NULL,
                                                        MAKELONG(0, 0x7fff));
                                        break;
                        }
                        return (TRUE);
                        case ID_SAVE:
                                GetDlgItemText(hDlg, IDC_EDIT, OpenName, 128);
                                if (strchr(OpenName, '*') || strchr(OpenName, '?')) {
                                        SeparateFile(hDlg, (LPSTR) soundstr, (LPSTR)
                                        DefSpec,(LPSTR) OpenName);
                                        if (soundstr[0])
                                                strcpy(DefPath, soundstr);
                                        ChangeDefExt(DefExt, DefSpec);
                                        UpdateListBox(hDlg);
                                        return (TRUE);
                                }
```

```
                if (!OpenName[0]) {
                        MessageBox(hDlg, "No filename specified.",
                                NULL, MB_OK | MB_ICONHAND);
                                return (TRUE);
                        }
                AddExt(OpenName, DefExt);
                strcpy(soundstr,OpenName);
                EndDialog(hDlg, TRUE);
                        return (TRUE);

                case IDCANCEL:
                        EndDialog(hDlg, NULL);
                return (TRUE);
            } /*  case WM_COMMAND */
            break;
            case WM_INITDIALOG:                 /* message: initialize   */
                UpdateListBox(hDlg);
                SetDlgItemText(hDlg, IDC_EDIT, DefSpec);
                SendDlgItemMessage(hDlg,        /* dialog handle     */
                    IDC_EDIT,                   /* where to send message */
                    EM_SETSEL,                  /* select characters     */
                    NULL,                       /* additional information */
                    MAKELONG(0, 0x7fff));       /* entire contents       */
                SetFocus(GetDlgItem(hDlg, IDC_EDIT));
                return (FALSE); /* Indicates the focus is set to a control */
        } /*switch Message */
        return FALSE;
}
/*************************************************************************
            Function: recordWAVEFILE(DWORD,HWND)
*************************************************************************/
DWORD recordWAVEFile(DWORD dwMilliSeconds,HWND hDlg)
{
        WORD                    wDeviceID=0;
        DWORD                   dwReturn;
```

```
MCI_OPEN_PARMS          mciOpenParms;
MCI_RECORD_PARMS        mciRecordParms;
MCI_SAVE_PARMS          mciSaveParms;
MCI_PLAY_PARMS          mciPlayParms;
int                     i;
FARPROC                 lpProcSave;

mciOpenParms.lpstrDeviceType        = "waveaudio";
mciOpenParms.lpstrElementName       = ""
if (dwReturn = mciSendCommand(0,MCI_OPEN,MCI_OPEN_ELEMENT|
            MCI_OPEN_TYPE,(DWORD)(LPVOID)&mciOpenParms))
{
        return (dwReturn);
}
wDeviceID = mciOpenParms.wDeviceID;
mciRecordParms.dwTo = dwMilliSeconds;
for(i=0;i<500;i++);  /* Delay */
if(dwReturn = mciSendCommand(wDeviceID,MCI_RECORD,MCI_TO|MCI_WAIT,
                        (DWORD)(LPVOID)&mciRecordParms))
{
        mciSendCommand(wDeviceID,MCI_CLOSE,0,NULL);
        return (dwReturn);
}
mciPlayParms.dwFrom = 0L;
if (dwReturn = mciSendCommand(wDeviceID,MCI_PLAY, MCI_FROM |
        MCI_WAIT,  (DWORD)(LPVOID) &mciPlayParms))
{
        mciSendCommand(wDeviceID,MCI_CLOSE,0,NULL);
        return(dwReturn);
}
if (MessageBox(hDlg,"Do you want to save this recording?",
                        "",MB_YESNO)==IDNO)
{
        mciSendCommand(wDeviceID,MCI_CLOSE,0,NULL);
        return(01);
```

```
    }
    lpProcSave = MakeProcInstance((FARPROC)SaveDlg,hInst);
    DialogBox(hInst,"Save",hDlg,lpProcSave);
    FreeProcInstance(lpProcSave);
    mciSaveParms.lpfilename = soundstr;
    if(dwReturn = mciSendCommand(wDeviceID,MCI_SAVE, MCI_SAVE_FILE |
                        MCI_WAIT, (DWORD)(LPVOID)&mciSaveParms))
    {
        mciSendCommand(wDeviceID,MCI_CLOSE,0,NULL);
        return(dwReturn);
    }
    mciSendCommand(wDeviceID,MCI_CLOSE,0,NULL);
    return(0L);
}
/************************************************************************
        Function: ScrollWndProc(HWND,UINT,WPARAM,LPARAM)
************************************************************************/
LRESULT CALLBACK ScrollWndProc(hScrollWnd,iMessage,wParam,lParam)
HWND        hScrollWnd;
UINT        iMessage;
WPARAM      wParam;
LPARAM      lParam;
{
    static short    yChar;          /* pointer to the "About" function */
    static short    yCilent;
    static short    nVscrollMax,nHscrollMax;
    static short    nVscrollPos,nHscrollPos;
    short           nVscrollInc,nHscrollInc;
    switch(iMessage) {
        case WM_SIZE:
        nVscrollMax = max (0,4);
        nVscrollPos = min (nVscrollPos,nVscrollMax);
        nHscrollMax = max (0,10);
        nHscrollPos = min (nHscrollPos,nHscrollMax);
```

```
        SetScrollRange(hScrollWnd,SB_VERT,0,nVscrollMax,FALSE);
        SetScrollPos(hScrollWnd,SB_VERT,nVscrollPos,TRUE);
        SetScrollRange(hScrollWnd,SB_HORZ,0,nHscrollMax,FALSE);
        SetScrollPos(hScrollWnd,SB_HORZ,nHscrollPos,TRUE);
break;

case WM_VSCROLL:
switch (wParam)
{
        case SB_TOP:
                nVscrollInc = -nVscrollPos;
                break;
        case SB_BOTTOM:
                nVscrollInc = nVscrollMax - nVscrollPos;
                break;
        case SB_LINEUP:
                nVscrollInc = -1;
                nVSize += (nVscrollInc*26);
                if(nVSize < 0)
                        nVSize = 0;
                break;
        case SB_LINEDOWN:
                nVscrollInc = 1;
                nVSize += (nVscrollInc*26);
                if(nVSize >180)
                        nVSize = 104;
                break;
        case SB_PAGEUP:
                nVscrollInc = min (-1,-yCilent/yChar);
                break;
        case SB_PAGEDOWN:
                nVscrollInc  = max (1,yCilent/yChar);
                break;
        case SB_THUMBTRACK:
                nVscrollInc = LOWORD(lParam) - nVscrollPos ;
```

```
                nVSize += (nVscrollInc*13);
                break;
        default:
                nVscrollInc = 0;
    }
    if (nVscrollInc == max(-nVscrollPos,min(nVscrollInc,nVscrollMax - nVscrollPos)))
        {
                nVscrollPos += nVscrollInc;
                ScrollWindow(hScrollWnd,0,60*nVscrollInc,NULL,NULL);
                SetScrollPos (hScrollWnd,SB_VERT,nVscrollPos,TRUE);
                UpdateWindow (hScrollWnd);
        }
    break;
    case WM_HSCROLL:
        switch(wParam)
        {
                case SB_LINEUP:
                        nHscrollInc = -1;
                        nHSize+=-1;
                        if(nHSize<1)
                                nHSize=1;
                        break;
                case SB_LINEDOWN:
                        nHscrollInc = 1;
                        nHSize+=1;
                        break;
                case SB_PAGEUP:
                        nHscrollInc = -8;
                        break;
                case SB_PAGEDOWN:
                        nHscrollInc = 8;
                        break;
                case SB_THUMBPOSITION:
                        nHscrollInc = LOWORD (lParam) - nHscrollPos;
                        nHSize+=nHscrollInc;
```

```
                        break;
                default:
                        nHscrollInc = 0;
                }
        if(nHscrollInc == max(-nHscrollPos,min(nHscrollInc,nHscrollMax - nHscrollPos)))
                {
                        nHscrollPos += nHscrollInc;
                        ScrollWindow (hScrollWnd,120*nHscrollInc,0,NULL,NULL);
                        SetScrollPos(hScrollWnd,SB_HORZ,nHscrollPos,TRUE);
                        UpdateWindow(hScrollWnd);
                }
                        break;
         default:
                return DefWindowProc (hScrollWnd,iMessage,wParam,lParam);
        }
      return (NULL);
}
/*************************************************************************
        Function : FreqWndProc(HWND,UINT,WPARAM,LPARAM)
*************************************************************************/
LRESULT CALLBACK FreqWndProc(hFreqWnd,iMessage,wParam,lParam)
HWND        hFreqWnd;
UINT        iMessage;
WPARAM      wParam;
LPARAM      lParam;
{
        switch(iMessage) {
                case WM_CLOSE:                  /* message: close the window    */
                        DestroyWindow(hFreqWnd);
                        break;
                default:
                return (DefWindowProc(hFreqWnd, iMessage, wParam, lParam));
        }
      return(NULL);
}
```

```
/*************************************************************************
                Function: four1(float,int,int);
                PurPose :
                        Calculate  Frequency by FFT
*************************************************************************/
void four1(float data[],int nn,int isign)
/* Replaces data by its discrete Fourier transkform, if isign is input as;
   or replaces data by nn times its inverse discrete Fourier transform, if
   isign is input as - , data is a complex array of length nn, input as a
   real array data[1..2*nn]. nn MUST be an integer power of 2( this is not
   checked for!).*/
{
  int n,mmax,m,j,istep,i;
  double wtemp,wr,wpr,wpi,wi,theta;  /* Double precision for the  trigonometric recurrents */
  float tempr,tempi;

  n = nn << 1;
  j=1;
  for (i=1;i<n;i+=2)  /* This is the bit-reversal section of the routine */
        {
         if (j>i)
                {
                 SWAP(data[j],data[i]);/*Exchange the two complex numbers */
                 SWAP(data[j+1],data[i+1]);
                }
         m = n >> 1;
         while(m >= 2 && j > m)
                {
                 j-=m;
                 m >>=1;
                }
         j+=m;
        }
        mmax=2;   /* Here begin the Danieson-Lanczos section of the routine*/
```

```
    while ( n> mmax)
     { /* Outer loop executed log(2)nn times*/
          istep = 2*mmax;
          theta = 6.28318530717959/(isign*mmax);
          /*Initialize for the  trigonometric recurrence*/
          wtemp = sin(0.5*theta);
          wpr = -2.0*wtemp*wtemp;
          wpi = sin(theta);
          wr = 1.0;
          wi = 0.0;
          for (m=1;m<mmax;m+=2) /*Here are the two nested inner loops*/
           {
                for(i=m;i<=n;i+=istep)
                 {
                      j = i+mmax
                      /* This is the Danielson-Lanczos formula*/
                      tempr = wr*data[j]   - wi*data[j+1];
                      tempi = wr*data[j+1] + wi*data[j];
                      data[j] =   data[i]-tempr;
                      data[j+1] = data[i+1]-tempi;
                      data[i] += tempr;
                      data[i+1] += tempi;
                  } /* Trigonometric recurrence*/
                wr = (wtemp=wr)*wpr-wi*wpi+wr;
                wi = wi*wpr+wtemp*wpi+wi;
           }
          mmax = istep;
     }
}
/*********************************************************************
     Function: window(int,float,float)
*********************************************************************/
float window(int j,float a,float b)
{
     return (float)(1.0-fabs((((j)-1)-(a))*(b)));
```

```
}
/*****************************************************************************
        Function : spctrm(char,int,int)
        PurPose  :
                Calculate Power Spectrum
*****************************************************************************/
void spctrm(char far fp[],float p[],int m,int k)
{
        int mm,m44,m43,m4,kk,joff,j2,j,count=0;
        float w,facp,facm,sumw=0.0,den=0.0,w1[1025];
        unsigned char *a;

        mm=m+m;
        m43=(m4=mm+mm)+3;
        m44=m43+1;
        facm=(float)(m-0.5);
        facp=(float)(1.0/(m+0.5));
        for(j=1;j<=mm;j++)  sumw+= pow(window(j,facm,facp),2);
        for(j=1;j<=m;j++) p[j]=0.0; /* initiallize the spectrum to zero */

        a = fp;
        for(kk=1;kk<=k;kk++)
                {
                 for(joff = -1;joff<=0;joff++)
                        {
                                for (j=joff+2;j<=m4;j+=2)
                                        {
                                          w1[j] = (((float)(*a))-127.0)/127.0;
                                          a++;
                                        }
                        }
                 for (j=1;j<=mm;j++)   /*.Apply the window to the data*/
                 {
                        j2=j+j;
                        w = window(j,facm,facp);
```

```
                w1[j2] *=w;
                w1[j2-1]*=w;
        }
        four1(w1,mm,1); /* fourier transform the windowed data */
        p[1] +=pow(w1[1],2)+pow(w1[2],2);
            /*Sum results into previous segments*/
        for (j=2;j<=m;j++)
        {
                j2=j+j;
                p[j] += (pow(w1[j2],2)+pow(w1[j2-1],2) +
                        pow(w1[m44-j2],2)+pow(w1[m43-j2],2));
        }
        den+=sumw;
    }
den *=m4;                   /* Correct normalization*/
for(j=1;j<=m;j++)
        p[j]/=den; /* Normallize the output*/
}
```

```
/*********************************************

        File : Voice.def

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

NAME            SBTEST


DESCRIPTION  'Sample Microsoft Windows Application'

EXETYPE      WINDOWS

PROTMODE

CODE        PRELOAD MOVEABLE DISCARDABLE
DATA        PRELOAD MOVEABLE MULTIPLE

HEAPSIZE     0xAFFF  ; 45k
STACKSIZE    10290
SEGMENTS
        _RES:PRELOAD MOVEABLE DISCARDABLE

EXPORTS
        MainWndProc
        About
        OpenDlg
        PaintWndProc
         ScrollWndProc
        FreqWndProc
        SpctrmWndProc
```

```
/*************************************************************************
                File : Voice.h
*************************************************************************/
#define         TAB                     50
/* file menu items */
#define         IDM_OPEN    100
#define         IDM_EXIT    101
#define         IDM_ABOUT   102
/*---------------- button IDs ------------------*/
#define         ID_OK           103
#define         ID_START    104
#define         ID_STOP     105
#define         ID_PLAY     106

#define     ID_RECORD       107
#define         ID_SAVE             108

#define         IDM_DISPLAY         200
#define         IDM_HISTRO  201
#define         IDM_FREQUENCY 205
#define         IDM_POWER    206
#define         IDM_QUIT     202
#define         IDM_SOUND   203
#define         IDM_PLAY    204
#define         IDM_SPECTRUM        205
#define         IDM_FREQ    206
#define         IDM_RECORD  207
/* edit menu items */
/* Control IDs */
#define     IDC_FILENAME  400
#define     IDC_EDIT      401
#define     IDC_FILES     402
#define     IDC_PATH      403
#define     IDC_LISTBOX   404
#define         IDC_PAINT     405
```

```
#define MAXFILESIZE 0x7fff          /* maximum file size that can be loaded */

/* spctrm */
#define SQR(a) (sqrarg=(a),sqrarg*sqrarg)
#define WINDOW(j,a,b) (1.0-fabs((((j)-1)-(a))*(b)))
#define SWAP(a,b) tempr=(a);(a)=(b);(b)=tempr

int PASCAL WinMain(HANDLE, HANDLE, LPSTR, int);
BOOL InitApplication(HANDLE);
BOOL InitInstance(HANDLE, int);
LRESULT CALLBACK MainWndProc(HWND, unsigned, WPARAM, LPARAM);
BOOL FAR PASCAL About(HWND, unsigned, WORD, LONG);
HANDLE FAR PASCAL OpenDlg(HWND, unsigned, WPARAM, LPARAM);
HANDLE FAR PASCAL SaveDlg(HWND, unsigned, WPARAM, LPARAM);
BOOL CheckFileName(HWND, PSTR, PSTR);
void SeparateFile(HWND, LPSTR, LPSTR, LPSTR);
void UpdateListBox(HWND);
void SetNewBuffer(HWND, HANDLE, PSTR);
void AddExt(PSTR, PSTR);
void ChangeDefExt(PSTR, PSTR);
LRESULT CALLBACK PaintWndProc(HWND,UINT,WPARAM,LPARAM);
LRESULT CALLBACK ScrollWndProc(HWND,UINT,WPARAM,LPARAM);
LRESULT CALLBACK FreqWndProc(HWND,UINT,WPARAM,LPARAM);
LRESULT CALLBACK HistroWndProc(HWND,UINT,WPARAM,LPARAM);
int Message(HWND,LPSTR);
BOOL InitMainFrame(void);
BOOL InitClass(void);
BOOL InitPaintWnd(void);
BOOL InitHistroWnd(void);
/*long FAR PASCAL SoundWndProc(HWND,unsigned,WORD,LONG); */
BOOL FAR PASCAL PlaySound(HWND,unsigned,WORD,LONG);
BOOL FAR PASCAL Record(HWND,unsigned,WORD,LONG);
DWORD recordWAVEFile(DWORD,HWND);
void spctrm(char far fp[],float p[],int m,int k);
```

```
float window(int j,float a,float b);
void four1(float data[],int nn,int isign);
BOOL InitSpctrmWnd(void);
LRESULT CALLBACK SpctrmWndProc (HWND,UINT,WPARAM,LPARAM);

void SpctrmPaint(float PowerSpctrm[]);
void LineScale(HDC,int,int,int);
```

```
/*****************************************************************

        File : Voice.rc

*****************************************************************/
#include "windows.h"
#include "sbtest1.h"

About1 ICON  about.ico
About2 ICON  start.ico
Play   ICON  play.ico
Start  ICON  start.ico
Stop   ICON  stop.ico


#include "sbtest1.dlg"

EditFileMenu MENU
BEGIN
   POPUP       "&File"
   BEGIN
       MENUITEM  "&Open...",              IDM_OPEN
       MENUITEM  SEPARATOR
       MENUITEM  "E&xit",          IDM_EXIT
       MENUITEM  SEPARATOR
       MENUITEM  "&About Voice Recognize...", IDM_ABOUT
   END
   POPUP             "&Voice Process"
   BEGIN
       MENUITEM  "&Spectrum",        IDM_SPECTRUM
       MENUITEM  SEPARATOR
       MENUITEM  "&Histrogram",          IDM_HISTRO
   END
   POPUP     "&Sound"
   BEGIN
```

```
    MENUITEM   "&Play",                    IDM_PLAY
        END
    POPUP      "&Record Sound File"
        BEGIN
        MENUITEM   "&Record",                      IDM_RECORD
    END
END


PaintMenu MENU
BEGIN
        POPUP          "&Play"
        BEGIN
                MENUITEM    "&Display..",IDM_DISPLAY
                MENUITEM    "&Histrogram",IDM_HISTRO
                MENUITEM    SEPARATOR
                MENUITEM    "&Quit",IDM_QUIT
        END
        POPUP          "&Tool"
        BEGIN
                MENUITEM    "&Frequency Analyze",IDM_FREQUENCY
                MENUITEM    SEPARATOR
                MENUITEM    "Power Frequency",IDM_POWER
        END
END
Open DIALOG 10, 10, 148, 112
STYLE DS_MODALFRAME | WS_CAPTION | WS_SYSMENU
CAPTION "Open "
BEGIN
      LTEXT "Open File &Name:", IDC_FILENAME, 4,  4,  60, 10
      EDITTEXT                  IDC_EDIT,     4, 16, 100, 12, ES_AUTOHSCROLL
      LTEXT "&Files in",        IDC_FILES,    4, 40,  32, 10
      LISTBOX,                  IDC_LISTBOX,  4, 52,  70, 56,
WS_TABSTOP|WS_VSCROLL
      LTEXT "",                 IDC_PATH,     40, 40, 100, 10
```

```
    DEFPUSHBUTTON "&Open" ,  IDOK,       87, 60,  50, 14
    PUSHBUTTON "Cancel",     IDCANCEL,   87, 80,  50, 14
END

Record DIALOG 100, 15, 148, 112
STYLE DS_MODALFRAME | WS_CAPTION | WS_SYSMENU
CAPTION "Record "
BEGIN
            LTEXT "Record Sound File ",IDM_RECORD, 4,  4,  60, 10
    LTEXT "You can record sound for 1000 ms.",IDM_RECORD, 4, 16, 130, 12
            LTEXT "It records immediately after you push ",IDM_RECORD, 4, 29, 145,
                                                    10
            LTEXT "Record button.",IDM_RECORD,4,39,100,10
            LTEXT "Afer Recording it plays the sound ",IDM_RECORD,4,49,130,10
            LTEXT "that is recorded.",IDM_RECORD,4,59,100,10
            PUSHBUTTON "&Record" ,  ID_RECORD, 87, 69,  50, 14
            PUSHBUTTON "Cancel",    IDCANCEL,  87, 89,  50, 14
END
Save DIALOG 100,100,148,112
STYLE DS_MODALFRAME | WS_CAPTION | WS_SYSMENU
CAPTION " Save Sound File"
BEGIN
            LTEXT " &Save File Name:",  ID_SAVE,4,4,60,10
            EDITTEXT
IDC_EDIT,4,16,100,12,ES_AUTOHSCROLL
            LTEXT " &File in:"                  ,IDC_FILES,4,40,32,10
            LISTBOX,
IDC_LISTBOX,4,52,70,56,WS_TABSTOP|WS_VSCROLL
            LTEXT "",                                   IDC_PATH,40,40,100,10
            DEFPUSHBUTTON "&Save",          ID_SAVE,87,60,50,14
            PUSHBUTTON "&Cancel",           IDCANCEL,87,80,50,14
END
```

```
/*******************************************************
        File : Dialog.dlg
*******************************************************/
DLGINCLUDE RCDATA DISCARDABLE
BEGIN
   "SBTEST1.H\0"
END

IDM_OPEN DIALOG 93, 65, 160, 135
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
CAPTION "Play Sound"
FONT 8, "Helv"
BEGIN
   CONTROL        "", IDM_EXIT, "Static", SS_BLACKFRAME, 5, 5, 150, 30
   DEFPUSHBUTTON  "Play", ID_PLAY, 15, 10, 40, 20
   ICON           ID_START, ID_OK, 100, 11, 18, 20
   CONTROL        "", ID_PLAYS, "Static", SS_BLACKFRAME, 5, 40, 150, 30
   PUSHBUTTON     "Start", 106, 15, 45, 40, 20
   CONTROL        "", 107, "Static", SS_BLACKFRAME, 5, 75, 150, 30
   PUSHBUTTON     "Stop", ID_STOP, 15, 80, 40, 20
   ICON           110, 109, 100, 46, 18, 20
   ICON           112, 111, 100, 81, 18, 20
   ICON           114, 113, 118, 46, 18, 20
   PUSHBUTTON     "&Exit", IDD, 5, 113, 150, 14
END
```

กิตติกรรมประกาศ

ตลอดการทำโครงงาน ครั้งนี้ ได้รับคำแนะนำเป็นอย่างดี จากพี่ ธันวา ศรีประโมง ซึ่งต้องขอขอบคุณ พี่ ธันวา ไว้นะโอกาสนี้ รวมทั้ง อาจารย์ สมศักดิ์ มิตะถา ที่คอยแนะนำ การทำโครงงาน เป็นอย่างดี และให้ความกรุณาให้ยืม ตำราต่าง ๆ เป็นอย่างดี นอกจากนี้ ยังมีเพื่อน ๆ อีกหลายคนที่ไม่ได้กล่าวชื่อไว้นะที่นี้ ที่ให้คำปรึกษามาโดยตลอด ในนามของผู้จัดทำโครงงานในครั้งนี้ ต้องขอขอบคุณทุกท่านไว้นะโอกาสนี้ด้วย

บรรณานุกรม

1. William H.Press, Brain P.Flannery, Sual A.Teukolsky, William T.Vetterling, "Numerical Recipes in C", Cambridge University Press , 1988
2. Peter Norton , Paul L. Yau, "Windows 3.0 Power Programming Techniques", BANTAM BOOKS, 1990
3. CHARLES PETZOLD, "Programming Windows", Microsoft Corporation 1988
4. ศิราณี จัสวชิรกุล, สุวรรณ ศรีหะวรรณ์,"การทำนายเชิงเส้น", วิทยานิพนธ์ภาควิชาคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ ลาดกระบัง พ.ศ. 2534
5. ธันวา ศรีประโมง,"การแยกหน่วยเสียงของพยางค์ภาษาไทยโดยการวิเคราะห์ในแกนความถี่ฮาโมนิค" รายงานสัมมนา M II ระดับปริญญามหาบัณฑิต คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, พ.ศ. 2535