ภาควิชาครุศาสตร์วิศวกรรม

คณะครุศาสตร์อุตสาหกรรม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองปริญญานิพนธ์

ชื่อหัวข้อ กรณีศึกษาประสิทธิภาพของเวคเตอร์ควอนไตส์

Case Study of Vector Quantize Efficiency

ชื่อนักศึกษา 1. นางสาวอัสนีย์ ศิริพงษ์ รหัสประจำตัว 43035279

2. นางสาวปิยดา ปันชุน รหัสประจำตัว 43035596

หลักสูตร ครุศาสตร์อุตสาหกรรมบัณฑิต **สาขาวิชา** วิศวกรรมโทรคมนาคม

อาจารย์ที่ปรึกษา อาจารย์วรวิทย์ สมหา

อาจารย์ที่ปรึกษาร่วม อาจารย์สุชิน อาจหาญ

คณะกรรมการสอบปริญญานิพนธ์	ลายมือชื่อ
1. อาจารย์พงษ์เกียรติ เชษฐพิทักษ์สกุล	.....
2. อาจารย์สุชิน อาจหาญ	.....
3. อาจารย์ปิยะ จิตธรรมมาภิรมย์	.....
4. อาจารย์วรวิทย์ สมหา	.....
5. อาจารย์อำพล ทองระอา	.....

วัน/เดือน/ปีที่สอบ วันพฤหัสบดีที่ 2 พฤษภาคม พ.ศ. 2545 เวลา 15.00 น.

สถานที่สอบ ห้อง ค.317 คณะครุศาสตร์อุตสาหกรรม สจล.

ภาควิชารับรองแล้ว

ลงนาม.....

(ผศ.วิสุทธิ์ อธิพรธรรม)

หัวหน้าภาควิชาครุศาสตร์วิศวกรรม

วันที่ 21 เดือน พ.ค. พ.ศ. 2545

<BT4401202>

กรณีศึกษาประสิทธิภาพของเวคเตอร์ควอนไตส์

ปริญญานิพนธ์

กรณีศึกษาประสิทธิภาพของเวคเตอร์ควอนไตส์

CASE STUDY OF VECTOR QUANTIZE EFFICIENCY

นางสาวอัสนีย์ ศิริพงษ์
นางสาวปิยดา ปันชุน

ปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรครุศาสตร์อุตสาหกรรมบัณฑิต
สาขาวิชาวิศวกรรมโทรคมนาคม
ภาควิชาครุศาสตร์วิศวกรรม คณะครุศาสตร์อุตสาหกรรม
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2544

ปริญญานิพนธ์

เรื่อง กรณีศึกษาประสิทธิภาพของเวคเตอร์ควอนไตส์

Case Study of Vector Quantize Efficiency

วัตถุประสงค์

1. เพื่อศึกษาทฤษฎีและหลักการเข้ารหัสของเวคเตอร์ควอนไตส์
2. เพื่อออกแบบโครงสร้าง และวิธีการเปรียบเทียบสัญญาณ
3. เพื่อเขียนโปรแกรมการเข้ารหัสเวคเตอร์ควอนไตส์ และโปรแกรมทดสอบเปรียบเทียบสัญญาณ
4. เพื่อทดลองหาผลเปรียบเทียบการเข้ารหัสสัญญาณเสียงทั้งสามแบบ
5. เพื่อนำผลการทดลองไปใช้สรุปประสิทธิภาพของเวคเตอร์ควอนไตส์

ประโยชน์ที่คาดว่าจะได้รับ

1. มีความรู้ความเข้าใจทฤษฎีและหลักการเข้ารหัสแบบเวคเตอร์ควอนไตส์
2. ได้แปลนแบบโครงสร้าง และวิธีการเปรียบเทียบสัญญาณ
3. ได้โปรแกรมการเข้ารหัสเวคเตอร์ควอนไตส์ และโปรแกรมทดสอบเปรียบเทียบสัญญาณ
4. ได้ผลจากการเปรียบเทียบสัญญาณเข้าเสียงทั้งสามแบบ
5. ได้ผลสรุปประสิทธิภาพของเวคเตอร์ควอนไตส์

ชื่อหัวข้อ	กรณีศึกษาประสิทธิภาพของเวคเตอร์ควอนไตส์
นักศึกษา	นางสาวอัสนีย์ ศิริพงษ์
	นางสาวปิยดา ปันชุน
อาจารย์ที่ปรึกษา	อาจารย์วรวิทย์ สมหา
อาจารย์ที่ปรึกษาร่วม	อาจารย์สุชิน อาจหาญ
หลักสูตร	ครุศาสตร์อุตสาหกรรมบัณฑิต
สาขาวิชา	วิศวกรรมโทรคมนาคม
ปีการศึกษา	2544

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้ นำเสนอกรณีศึกษาประสิทธิภาพของเวคเตอร์ควอนไตส์ โดยการนำเอาสัญญาณเสียงใน MP3 , PCM และเวคเตอร์ควอนไตส์ มาเข้ารหัสก็จะได้ผลของ อัตราส่วนสัญญาณต่อสัญญาณรบกวน ในระดับที่ต่างกันเราสามารถนำสัญญาณเสียงที่เข้ารหัสทั้ง 3 แบบ มาเปรียบเทียบและศึกษาผลที่ได้เพื่อนำสัญญาณเสียง ไปเข้ารหัสในแบบที่เหมาะกับสัญญาณเสียงนั้นๆ เพื่อลดอัตราของข้อมูลที่สามารถจัดเก็บในรูปต่างๆ กันได้

Thesis	Case Study of Vector Quantize Efficiency
Students	Miss Asanee Siriphong
	Miss Piyada Panchun
Advisor	Mr. Worawit Somha
Co – Advisor	Mr. Suchin Adhan
Education Level	Bachelor of Science in Industrial Education
Program in	Telecommunication Engineering
Academic Year	2001

ABSTRACT

This thesis presents the project of Case Study of Vector Quantize Efficiency. To used MP3 , PCM voice signal and vector quantize are encoded to signal to noise ratio in different level . It 's to make encode voice signal 3 type to compare and education for apply.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้ สำเร็จลุล่วงไปด้วยดี ด้วยความช่วยเหลือ และให้คำแนะนำปรึกษาตลอดจนแนวความคิดต่างๆ พร้อมทั้งข้อเสนอแนะแนวทางแก้ไขปัญหา ในการดำเนินงานรวมทั้งด้านเวลาและสถานที่ เครื่องมือ อุปกรณ์ ต่างๆ จาก อาจารย์วรวิทย์ สมหา อาจารย์สุชิน อาจหาญ อาจารย์โกศล ตราชู และอาจารย์ประจำภาควิชาครุศาสตร์วิศวกรรมทุกท่าน

คณะผู้จัดทำ ขอกราบขอบพระคุณท่านบุพการี ซึ่งให้โอกาส ที่ได้กรุณาให้งบประมาณและให้การสนับสนุนในการศึกษามาตลอด รวมทั้งคณาจารย์ และเจ้าหน้าที่ ภาควิชาครุศาสตร์วิศวกรรม คณะครุศาสตร์อุตสาหกรรม สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง รวมถึงกำลังใจจากเพื่อนๆ ที่ทำให้ปริญญานิพนธ์นี้สำเร็จลุล่วงไปด้วยดี

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญตาราง

สารบัญรูป

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

บทที่ 1
บทนำ

1.1 ความเป็นมา และความสำคัญของปริญญานิพนธ์

การส่งสัญญาณเสียงหรือการบันทึกเสียงในระบบดิจิตอล ส่วนสำคัญประการหนึ่งที่ต้องคำนึงถึง คือ การลดจำนวนบิตเรต (Bitrate) ในการส่งข้อมูลซึ่งจะมีผลทำให้ความจุของช่องสัญญาณเพิ่มมากขึ้นสามารถกระทำได้โดยการที่เข้ารหัสเสียง (Speech Coding) ซึ่งการเข้ารหัสเสียงมีหลายวิธีด้วยกัน และการเวคเตอร์ควอนไตส์ เป็นวิธีที่มีประสิทธิภาพอีกวิธีหนึ่งที่ถูกนำมาประยุกต์ใช้ ในการเข้ารหัสเสียงเพื่อลดจำนวนบิตเรต ในการส่งข้อมูลให้ต่ำลง โดยจะอ้างอิงเปรียบเทียบกับวิธีที่ใช้อยู่เป็นมาตรฐาน 2 วิธี คือ MP3 และ PCM เพื่อหาข้อสรุปถึงประสิทธิภาพและความเหมาะสมในการประยุกต์ใช้งานต่อไป

1.2 ขีดความสามารถของโครงงาน

โครงงานนี้มีขีดความสามารถดังนี้

1) สามารถเข้ารหัสเสียงเวคเตอร์ควอนไตส์ได้

2) สามารถแสดงผลของสัญญาณ SNR เป็นกราฟบนจอมอนิเตอร์ได้

3) สามารถเปรียบเทียบค่า SNR และ Bitrate ในการเข้ารหัสแต่ละแบบได้

4) สามารถพิมพ์ ผลการทดลองได้

5) โปรแกรมเปรียบเทียบสัญญาณ ใช้งานระบบปฏิบัติการวินโดว์ 98 ขึ้นไป

6) ค่าพารามิเตอร์ของเวคเตอร์ควอนไตส์ ที่โปรแกรมสามารถปรับเปลี่ยนได้คือค่ามิติ และอีลีเมนต์

7) สามารถเปรียบเทียบประสิทธิภาพ ได้กับการเข้ารหัสแบบ PCM ทั้ง แบบ A-Law และ μ – Law ได้

1.3 เนื้อหาโดยสังเขป

ปริญญานิพนธ์ฉบับนี้มีเนื้อหาในบทต่างๆ ดังต่อไปนี้

บทที่ 2 ทฤษฎี และหลักการ จะกล่าวถึงเนื้อหาที่นำมาอ้างอิง และใช้เป็นแนวทางในการศึกษาและออกแบบเขียนโปรแกรมประสิทธิภาพเวคเตอร์ควอนไตส์

บทที่ 3 การออกแบบ การสร้าง และการทำงานจะเป็นเนื้อหาตั้งแต่ขั้นตอนในการออกแบบโปรแกรม และการนำโปรแกรมต่างๆ มาใช้เพื่อให้สามารถทำงานร่วมกันอย่างมีประสิทธิภาพ

บทที่ 4 การทดลอง และผลการทดลอง ในบทนี้เป็นการนำเสนอ การทดลอง และผลการทดลอง โดยการแบ่งการทดลองออกเป็นส่วนๆ ตามการออกแบบ และเขียนโปรแกรมพร้อมบันทึกผลการทดลองในแต่ละส่วน

บทที่ 5 บทสรุป ปัญหา แนวทางแก้ไข และการพัฒนา ซึ่งเป็นการสรุปเกี่ยวกับความสามารถของกรณีศึกษาประสิทธิภาพของเวคเตอร์ควอนไตส์ และกล่าวถึงปัญหาที่เกิดขึ้นนับตั้งแต่การเริ่มสร้างโครงงานจนกระทั่ง โครงงานเสร็จสมบูรณ์ ตลอดจนแนวทางแก้ไขปัญหาที่เกิดขึ้น พร้อมทั้งเสนอแนวทางการพัฒนาโปรแกรม ในกรณีศึกษาประสิทธิภาพของเวคเตอร์ควอนไตส์ ให้สามารถนำไปใช้งานอย่างกว้างขวาง และปรับปรุงให้มีประสิทธิภาพดียิ่งขึ้น

ภาคผนวก ก ผังงาน และโปรแกรม

ภาคผนวก ข คู่มือการใช้งาน

บทที่ 2
ทฤษฎี และหลักการ

2.1 หลักการของ Pulse Code Modulation

Pulse Code Modulation (PCM) จะมีลักษณะที่แตกต่างจาก ระบบการสวิตชิ่ง แบบแอนะลอก ซึ่งทำการสับเปลี่ยนสัญญาณแอนะลอกในรูปแบบที่เป็นแรงดัน โดยเครื่องโทรศัพท์ แต่ระบบสวิตชิ่งดิจิตอลพัลส์สัญญาณถูกแปลงมาจากสัญญาณแอนะลอก โดยอาศัยหลักการของ PCM , PAM หรือ DELTA – M

2.1.1 พัฒนาการของ PCM

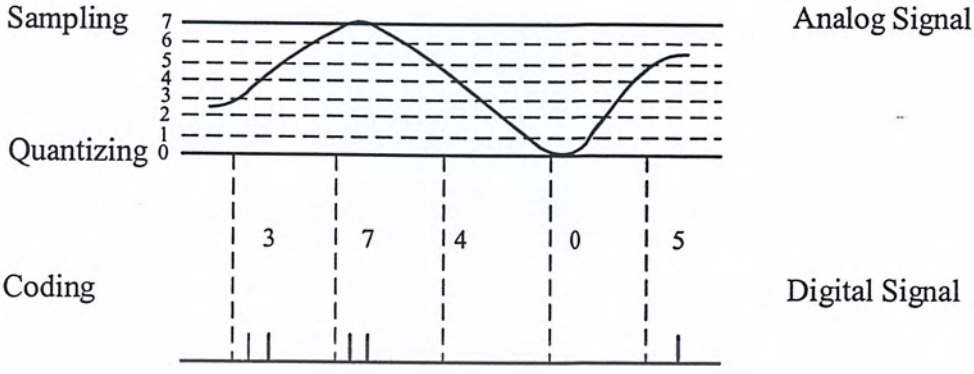
ระบบ PCM ถูกคิดค้นขึ้น โดย A.H.REEVES ในปี 1937 การลดทอนของสัญญาณที่มีในระบบแอนะลอก เทคนิค PCM ถูกใช้เริ่มแรกในการขนส่งระยะใกล้ โดยการแปลงสัญญาณดิจิตอลที่ความหนาแน่นของช่องสัญญาณถึง 24 ช่องสัญญาณ จะเห็นว่าในระยะต่างๆ มาได้มีการนำมาใช้ในการสื่อสารระยะไกลที่มีความจุของช่องสัญญาณสูงมาก

ขณะที่ PCM กำลังได้รับการพัฒนาอยู่นั้นระบบ PCM มีช่องความถี่กว้าง (Spectrum) ต่อช่องสัญญาณระหว่างสถานีสร้างสัญญาณ (Re – generative Repeaters) สั้นทำให้ค่าใช้จ่ายสำหรับสายส่งสัญญาณเพิ่มมากขึ้น

ระบบสื่อสารดิจิตอลทั้งหลายอาศัยเทคนิคการเข้ารหัสแบบ PCM เป็นส่วนมาก หรือเทคนิคที่พัฒนาจากระบบ PCM เช่น โทรศัพท์ ข้อมูล โทรสาร สัญญาณวิดีโอที่นำพา และสวิตชิ่งข่าวสารโดยเน็ตเวิร์คดิจิตอล

2.1.2 หลักการเบื้องต้นของ PCM

PCM เป็นเทคนิคในการแปลงแรงดันไฟฟ้าที่เปลี่ยนแปลงจากเวลาต่อเวลาเป็นขบวนพัลส์รหัสเลขฐานสองของช่วงเวลาที่เหมือนกัน ขบวนการเปลี่ยนแปลงนี้มีด้วยกัน 3 ขั้นตอน คือ การสุ่มตัวอย่าง (Sampling) การจัดระดับสัญญาณ (Quantizing) และการเข้ารหัส (Encoding) ดังแสดงในรูปที่ 2.1



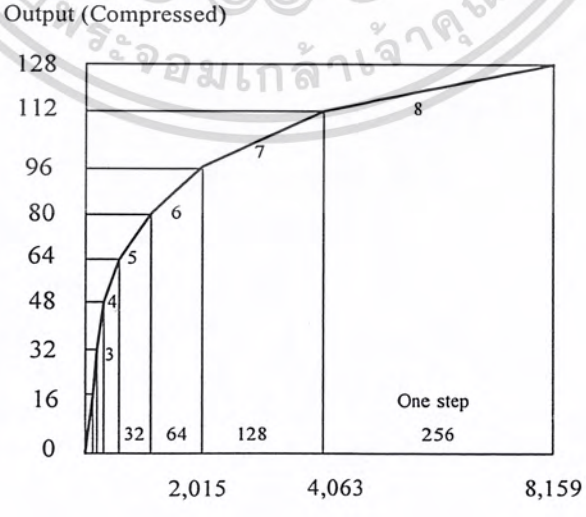
รูปที่ 2.1 หลักการของ PCM

1) ขบวนการ Sampling

เป็นขบวนการสุ่มอ่านค่าแรงดันสัญญาณเริ่มแรกด้วยช่วงเวลาที่เหมาะสม โดยอาศัยทฤษฎีการสุ่มตัวอย่างของ SHANNON ที่ว่า "ช่วงของการสุ่มตัวอย่างจำเป็นต้องมีค่าอย่างน้อยสองเท่าของความกว้างแถบความถี่สัญญาณที่ทำการสุ่มตัวอย่าง"

2) ขบวนการ Quantizing

เป็นการจัดระดับแรงดันของสัญญาณ PAM ที่จำนวนระดับแอมปลิจูดของสัญญาณแอนะลอกโดยระดับของการควอนไตส์ อย่างไรตาม โดยการปรับระดับของการควอนไตส์ให้เหมาะสมกับแรงดันสัญญาณอินพุตที่เข้ามา จะสามารถลดค่าสัญญาณรบกวน (Noise) ได้มากขึ้น เรียกเทคนิคนี้ว่าการควอนไตส์แบบไม่เป็นเชิงเส้น (Non-linear Quantization) ดังรูปที่ 2.2



รูปที่ 2.2 การควอนไตส์แบบไม่เป็นเชิงเส้น

3) ขบวนการเข้ารหัส

เป็นขบวนการที่แปลงค่าระดับการควอนไตส์ เป็นขบวนพัลส์รหัสเลขฐานสอง โดยอ้างอิงจากสมการข้างล่างเหล่านี้

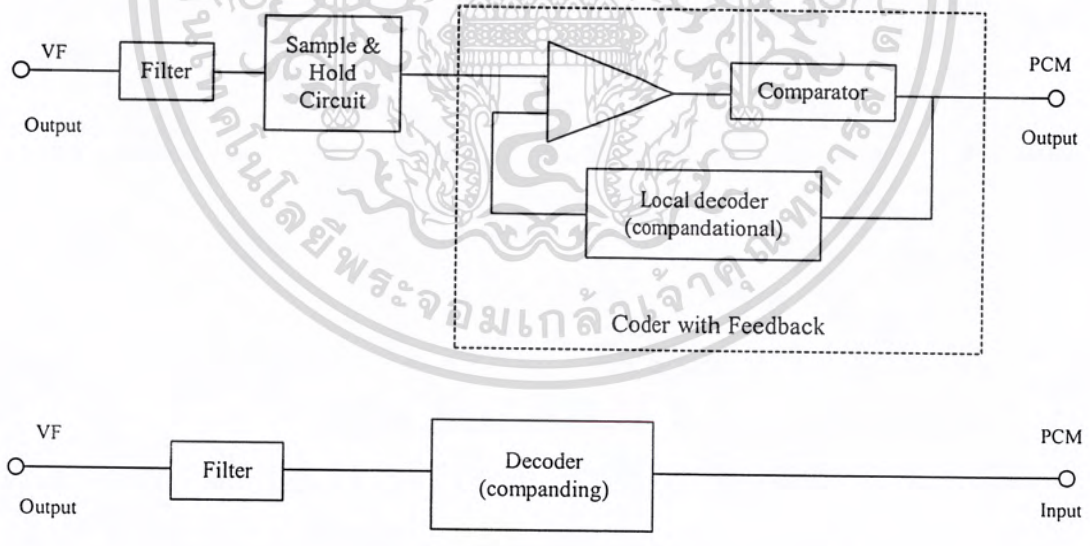
จำนวนของระดับการควอนไตส์ = $2^{(\text{จำนวนของบิตที่เข้ารหัส})}$

เช่น ถ้าจำนวนระดับของควอนไตส์ = 7

ถ้าจำนวนบิตที่เข้ารหัส = 8

จำนวนของระดับการควอนไตส์ = 2^8 = 128 ระดับ

จากสมการข้างบนจะเห็นว่าถ้าจำนวนของบิตที่เข้ารหัสเพิ่มขึ้น อัตราของสัญญาณต่อการรบกวนควอนไตส์จะเพิ่มขึ้น ปัจจุบันการส่งสัญญาณดิจิตอลของโครงข่าย นิยมใช้การเข้ารหัส 8 บิต และอัตราบิตของโทรศัพท์จะได้เป็นค่า 64 kBit/Sec จากสัญญาณโทรศัพท์ 4 kHz ตามข้อกำหนดของ CCITT RECS. G711 และรูปที่ 2.4 แสดงแผนผังการทำงานอุปกรณ์ส่งสัญญาณดิจิตอล PCM 24 ช่องสัญญาณ



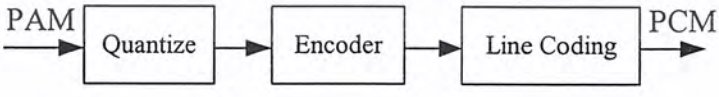
รูปที่ 2.3 อุปกรณ์ปลายทาง PCM สำหรับการเข้ารหัสสัญญาณเสียง

2.1.3 การมอดูเลตเชิงพัลส์

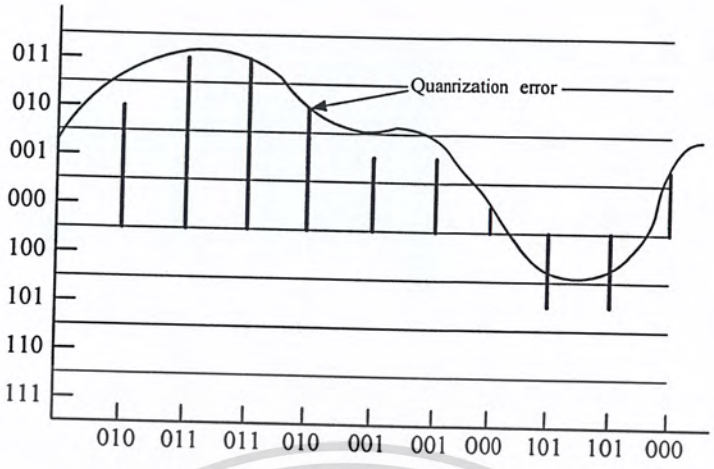
การมอดูเลตเชิงพัลส์ (Pulse Code Modulation) หรือที่เรียกย่อๆ ว่า PCM นั้น เป็นระบบสื่อสารแบบดิจิตอลที่ได้รับการพัฒนา และนำมาใช้ในระบบโทรศัพท์เมื่อต้นทศวรรษของ พ.ศ. 2500 ระบบ PCM เป็นระบบที่ใช้ประโยชน์จากทฤษฎีสุ่มตัวอย่างโดยจัดการกับสัญญาณ PAM ที่ได้จากการสุ่มตัวอย่างให้เป็นสัญญาณที่เหมาะกับการส่งผ่านระบบมากขึ้น ทั้งนี้เพราะสัญญาณ PAM นั้นเมื่อส่งผ่านระบบจะประสบกับปัญหาการผิดเพี้ยนซึ่งทำให้แถบความถี่ของสัญญาณเบสแบนด์ที่ปลายทางผิดเพี้ยนไปจากเดิม และเมื่อทำการดีมอดูเลตด้วยฟิลเตอร์ผ่านความถี่ต่ำ สัญญาณที่ดีมอดูเลตได้จะผิดเพี้ยนไปจากสัญญาณเดิมอย่างหลีกเลี่ยงไม่ได้ ระบบ PCM นั้นเป็นการจัดการกับสัญญาณ PAM นี้ โดยการนำสัญญาณ PAM ไปทำการเข้ารหัสเป็นสัญญาณดิจิตอล แล้วจึงนำสัญญาณดิจิตอลที่ได้นั้นส่งผ่านระบบต่อไป และทางภาครับจะทำการถอดรหัสเป็นสัญญาณ PAM และนำไปดีมอดูเลตเป็นสัญญาณเบสแบนด์กลับคืนมา

2.1.4 ขั้นตอนในการสร้างสัญญาณ PCM

ตามที่ได้กล่าวไว้ข้างต้น สัญญาณ PCM นั้นจะสร้างจากสัญญาณ PAM ที่ได้จากการสุ่มตัวอย่างของสัญญาณจริง ขั้นตอนที่เพิ่มเติมขึ้นนั้นเป็นดังที่แสดงไว้ในรูปที่ 2.4 กล่าว คือ สัญญาณ PAM ซึ่งมีขนาดตามค่าที่สุ่มตัวอย่างมานั้นจะถูกปัดขึ้น หรือปัดลงให้สามารถแทนระดับสัญญาณได้โดยใช้จำนวนบิตจำกัด โดยทั่วไปจำนวนบิตที่ใช้ในการเข้ารหัสจะเป็น 8 บิต ซึ่งสามารถแสดงระดับสัญญาณได้ทั้งหมด 256 ระดับ ดังนั้นจึงจำเป็นต้องปัดระดับของสัญญาณ PAM ให้ลงตามระดับที่เตรียมไว้ ขั้นตอนในการจัดให้ระดับลงตัวนี้เรียกว่าการควอนไตส์สัญญาณ ซึ่งถ้าให้แต่ละขั้นของระดับสัญญาณมีขนาดเท่ากันจะเป็นการควอนไตส์สัญญาณแบบเชิงเส้น ดังแสดงในรูปที่ 2.5 สัญญาณ PAM ที่ถูกควอนไตส์เรียบร้อยแล้ว จะถูกนำไปเข้ารหัสด้วยสัญญาณดิจิตอลขนาด 8 บิต ได้เป็นสัญญาณ PCM



รูปที่ 2.4 ขั้นตอนในการสร้างสัญญาณ PCM

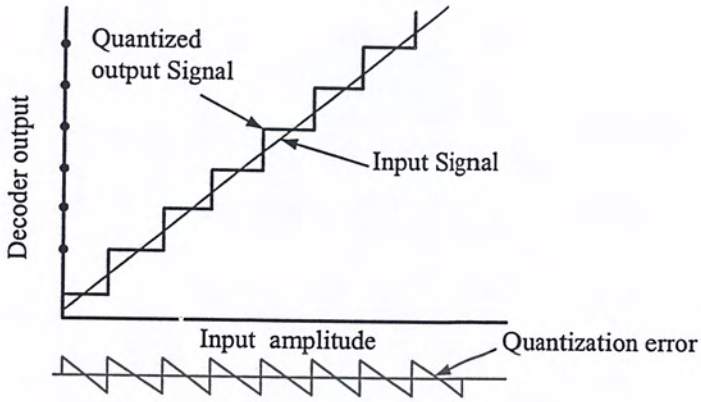


รูปที่ 2.5 ตัวอย่างการควอนไตส์สัญญาณแบบเชิงเส้น

ในการดีมอดูเลตสัญญาณ PCM จะทำได้โดยการย้อนกระบวนการที่กล่าวมาข้างต้น กล่าวคือ จากสัญญาณ PCM ที่รับมาได้จะทำการถอดรหัสออกมาเป็นสัญญาณ PAM ที่เป็นระดับลงตัว และเมื่อผ่านสัญญาณ PAM นี้เข้าสู่วงจรฟิลเตอร์ผ่านความถี่ต่ำ จะได้สัญญาณแอนะลอกที่ใกล้เคียงกับ สัญญาณเดิมออกมา ที่กล่าวว่าใกล้เคียงกับสัญญาณเดิมเพราะว่าสัญญาณ PAM ที่ได้จากการถอดรหัสนั้นจะมีระดับลงตัว และอาจจะไม่เท่ากับสัญญาณเดิมได้ ลักษณะเช่นนี้เหมือนกับการเกิดสัญญาณรบกวนแล้วทำให้ระดับของสัญญาณเปลี่ยนไป สัญญาณรบกวนในลักษณะนี้เรียกว่า การรบกวนที่เกิดจากการควอนไตส์ (Quantization Noise) นับเป็นข้อเสียที่สำคัญ จึงไม่นิยมใช้ในระบบโทรศัพท์

2.1.5 การรบกวนที่เกิดจากการควอนไตส์

ในหัวข้อนี้จะพิจารณาสัญญาณรบกวนที่เกิดจากการควอนไตส์แบบเชิงเส้น โดยให้ช่วงกว้างของแต่ละขั้นของระดับสัญญาณที่ถูกควอนไตส์เป็น q และในการวิเคราะห์หาการรบกวนที่เกิดจากการควอนไตส์นี้จะใช้สมมุติฐานดังต่อไปนี้



รูปที่ 2.6 การเกิดควอนไตส์เซชั่นนอยส์กรณีของควอนไตส์แบบเป็นเชิงเส้น

ถ้าให้ $v(t)$ เป็นสัญญาณแอนะลอก และ $v_D(t)$ เป็นสัญญาณที่ได้จากการถอดรหัส กำลังของสัญญาณรบกวนแบบ ดังรูปที่ 2.6 สัญญาณรบกวนที่เกิดจากการควอนไตส์นี้จะเขียนได้เป็น

$$N_Q = E\left[\left\{v_D\left(t - v(t)\right)\right\}^2\right]$$

$$= \int_{\frac{-1}{q}}^{1} \varepsilon^2 d\varepsilon$$

$$= \frac{q^2}{12} \tag{2.1}$$

ตามสมมุติฐานที่กล่าวไว้ข้างต้น กำลังของสัญญาณรบกวนที่เกิดจากการควอนไตส์จะเท่ากันที่ทุก ๆ ช่วงของการควอนไตส์ โดยไม่ขึ้นกับระดับของสัญญาณแอนะลอก ดังนั้นถ้าให้สัญญาณแอนะลอก มีค่า rms ของสัญญาณเป็น v_{rms} จะสามารถคำนวณค่ากำลังของสัญญาณต่อกำลังของสัญญาณรบกวนจากการควอนไตส์ หรือ SQR (Signal-to-quantization Noise Ratio) ได้จากสมการต่อไปนี้

$$SQR = 10\log\frac{E\left[v^2(t)\right]}{E\left[\left\{v_D(t) - v(t)\right\}^2\right]}$$

$$= 10\log\frac{v^2 rms}{q^2/12}$$

$$= 10.8 + 20\log\frac{v_{rms}}{q} \qquad (2.2)$$

กรณีที่สัญญาณแอนะลอกเป็นคลื่นรูปไซน์ที่มีแอมปลิจูดเป็น A สมการ (2.2) จะเขียนได้เป็น

$$SQR = 10\log\frac{A^2/2}{q^2/12} = 7.8 + 20\log\frac{A}{q} \qquad (2.3)$$

เนื่องจากตามเงื่อนไขข้อ (3) ช่วงของการควอนไตส์จะต้องครอบคลุมระดับสัญญาณสูงสุดของสัญญาณแอนะลอก ดังนั้นถ้าให้ $A_{\max}$ เป็นระดับสัญญาณสูงสุด แล้วทำการควอนไตส์ด้วยขั้นละ q เราจะต้องการจำนวนขั้นทั้งหมด คือ M เป็นดังนี้

$$M = \frac{2A_{\max}}{q} \qquad (2.4)$$

จำนวนขั้นที่ได้ตามสมการ (2.4) นั้นจะเป็นจำนวนลงตัวโดยการปัดขึ้น หรือปัดลง เมื่อใช้สัญญาณดิจิตอลแบบไบนารีซึ่งมี 2 ระดับสัญญาณ คือ "0" กับ "1" จะต้องใช้จำนวนบิต N เป็นดังนี้

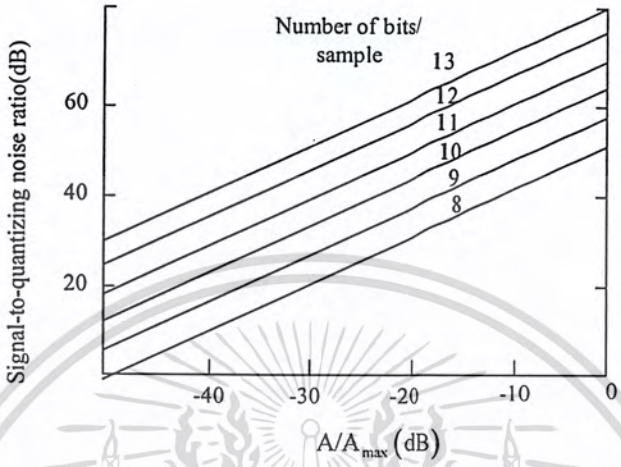
$$M = 2^N$$

หรือ

$$N = \log_2 M \qquad (2.5)$$

ในกรณีที่จำนวนระดับ M ไม่ทำให้ N เป็นจำนวนลงตัว ต้องปัดขึ้นให้เป็นจำนวนลงตัว ยกตัวอย่างเช่น ถ้า $M = 200$ เมื่อคำนวณ N จะได้ 7.6 ซึ่งเป็นจำนวนตัวเลขไม่ลงตัว และจำเป็นต้องทำการปัดขึ้นให้เป็นตัวเลขลงตัว คือ 8 ซึ่ง คือ ต้องใช้สัญญาณไบนารีจำนวน 8 บิต ในการควอนไตส์สัญญาณที่มี 200 ระดับ การใช้จำนวนบิตตามตัวอย่างที่กล่าวมานี้นับว่าเป็นการใช้บิตข้อมูลที่ไม่มีประสิทธิภาพ ฉะนั้นโดยทั่วไปมักจะแบ่งจำนวนระดับในการควอนไตส์ ให้

เหมาะสมกับจำนวนบิตที่จะใช้ในการเข้ารหัส ตัวอย่างเช่น ถ้าใช้สัญญาณไบนารี 8 บิต จะทำการควอนไตส์เป็น 256 ระดับ เป็นต้น



รูปที่ 2.7 SQR ของสัญญาณ PCM ที่มีขั้นของการควอนไตส์สม่ำเสมอ

เมื่อเราทำการหาค่า q โดยแทนสมการ (2.5) ลงในสมการ (2.4) ซึ่งจะได้ $q = 2A_{max}/2N$ และนำค่า q นี้แทนลงในสมการ (2.3) จะได้ SQR ของกรณีที่ระดับสัญญาณเป็น A ในขณะที่ระดับสัญญาณสูงสุดเป็น A_{max} ในรูปต่อไปนี้

$$SQR = 10\log\left[\frac{A^2/2\times 12}{4A^2{}_{max}/2^{2N}}\right]$$

$$= 1.76 + 6.02N + 20\log 10 \qquad (2.6)$$

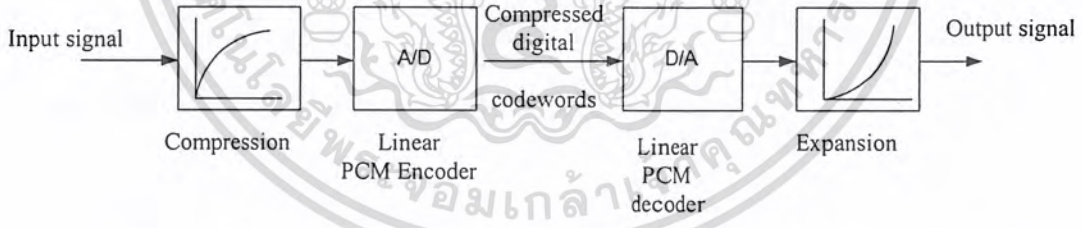
เมื่อนำสมการ (2.6) นี้ไปพล็อตกราฟจะได้ตามรูปที่ 2.7 จากรูปจะเห็นได้ว่า ค่า SQR จะสูงขึ้นเมื่อระดับของสัญญาณ A เข้าใกล้ระดับของสัญญาณสูงสุด คือ A_{max} และเนื่องจากในมาตรฐานของระบบโทรศัพท์กำหนดไว้ว่า SQR จะต้องสูงกว่า 26dB ตลอดช่วงไดนามิก (Dynamic Range) ซึ่งกำหนดให้เป็น 30 dB เมื่อพิจารณาจากกราฟจะเห็นได้ว่า ที่ตำแหน่ง A/A_{max} เป็น -30dB สัญญาณ 8 บิตจะไม่สามารถให้ SQR ที่เท่ากับ 26dB ได้ จำนวนบิตที่ต้องการใช้นั้นจะคำนวณได้จากสมการ (2.6) แล้วกำหนด N โดยการปัดขึ้น ผลที่ได้ให้เป็นเลขจำนวนเต็มลงตัว ซึ่งในกรณีนี้จะคำนวณออกมาได้ว่า $N = 10$ เมื่อใช้สัญญาณไบนารีจำนวน 10 บิต ในการเข้ารหัส

จะปรากฏว่า SQR ที่ได้ที่ระดับสัญญาณสูงสุดจะได้ถึงประมาณ 62 dB ซึ่งจัดว่าสูงเกินความจำเป็น และที่สำคัญ คือ สัญญาณระดับสูงเกือบเท่า A_{max} ไม่ค่อยจะเกิดขึ้นบ่อยนัก

สรุปได้ว่า มีข้อเสียหลัก 2 ประการ คือ มีโอกาสที่จะใช้จำนวนบิตในการลงรหัสอย่างไม่มีประสิทธิภาพ และค่า SQR นั้นเปลี่ยนแปลงไปตามระดับสัญญาณโดยที่ระดับสัญญาณที่สูงกว่า SQR สูงกว่า ข้อเสียดังกล่าวนี้ทำให้ระบบ PCM ที่มีการควอนไตส์สัญญาณแบบเป็นเชิงเส้นไม่ถูกนำมาใช้ และวิธีการที่ถูกนำมาใช้โดยทั่วไป คือ การทำคอมแพนดิง (Companding)

2.1.6 คอมแพนดิง

คอมแพนดิง ("Companding" มาจากคำภาษาอังกฤษสองคำมาผสมกัน คือ "Compressing" และ "Expanding") ซึ่งหมายถึงในภาคส่งนั้นก่อนที่จะทำการสุ่มตัวอย่างทำการอัดสัญญาณ (Compress) และในภาครับนั้น หลังจากที่ถอดรหัสแล้ว จะทำการยืดสัญญาณกลับคืนมา (Expand) รูปที่ 2.8 แสดงการแทรกขั้นตอนดังกล่าวนี้ลงในระบบ PCM อันที่จริงคอมแพนดิงเป็นวิธีการที่ใช้ในระบบโทรศัพท์แบบแอนะลอก คือ ระบบ FDM มาก่อน เพราะในระบบ FDM ที่ส่งผ่านสายทองแดงในเคเบิลเส้นเดียวกันนั้น จะมีปัญหาเรื่องของครอสทอล์ค (Crosstalk) ในขณะที่ระดับสัญญาณสูง การทำคอมแพนดิงเมื่อนำมาใช้ในระบบ PCM จะแก้ปัญหาหลักทั้ง 2 ประการที่กล่าวไว้ข้างต้น จะทำให้ประสิทธิภาพในการใช้จำนวนบิตในการเข้ารหัสสูงขึ้น และทำให้ค่า SQR ที่ระดับสัญญาณระดับต่างๆ มีโอกาสที่จะมีค่าเท่าๆ กันมากขึ้น

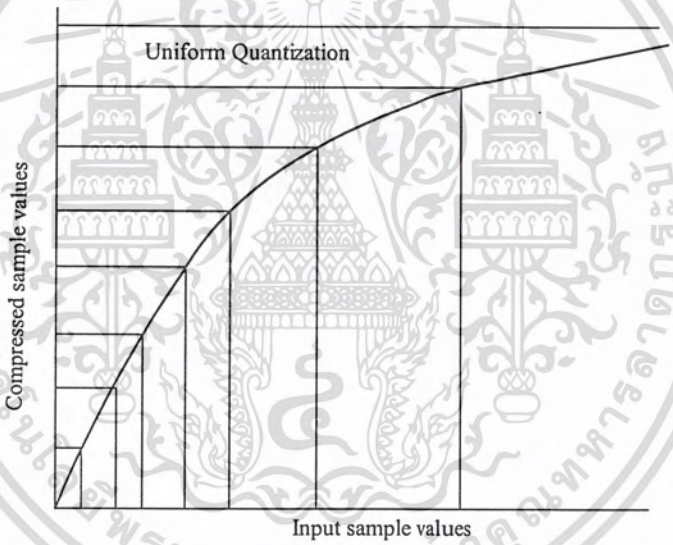


รูปที่ 2.8 การแทรกขั้นตอนของการอัดสัญญาณ และการยืดสัญญาณลงในระบบ PCM

วิธีการที่ใช้ในการทำคอมแพนดิงนั้น จะใช้ชิ้นส่วน หรือวงจรอิเล็กทรอนิกส์ที่มีการตอบสนองแบบไม่เป็นเชิงเส้น โดยที่ในภาคส่งนั้นจะมีความไม่เป็นเชิงเส้นในลักษณะอัดสัญญาณที่มีระดับสูงส่วน ในภาครับนั้น จะทำการยืดสัญญาณนั้นกลับคืนมา ลักษณะสมบัติของความไม่เป็นเชิงเส้นที่นำมาใช้งานได้นั้นมีได้หลายแบบด้วยกัน แต่แบบที่ถูกกำหนดเป็นมาตรฐาน และใช้งานกันอยู่ทั่วไปในขณะนี้มีอยู่ 2 แบบ และถูกเรียกว่าเป็น กฎ - μ(μ-law) และ กฎ - A(A-law) ซึ่งกฎ - μ นั้นจะมีลักษณะสมบัติของการอัดสัญญาณเขียนเป็นฟังก์ชันได้ในรูปต่อไปนี้

$$F\mu(x) = \text{sgn}(x)\frac{\ln(1+\mu|x|)}{\ln(1+\mu)} \tag{2.7}$$

โดยที่ x เป็นระดับสัญญาณที่ถูกนอร์มอไลซ์ไว้ให้มีค่าเป็น $-1 \le x \le 1$ และ $\text{sgn}(x)$ จะมีค่าเป็น +1 หรือ −1 ขึ้นอยู่กับค่า x ว่าเป็นบวก หรือเป็นลบ สำหรับ μ นั้นเป็นพารามิเตอร์ที่บอกถึงความมากน้อยของการอัดสัญญาณ รูปที่ 2.9 แสดงลักษณะสมบัติของการอัดสัญญาณนี้ เมื่อทำการอัดสัญญาณด้วยลักษณะสมบัติตามสมการ (2.7) จะต้องการยืดสัญญาณด้วยลักษณะสมบัติที่สอดคล้องกัน ซึ่งลักษณะสมบัตินี้จะหาได้จากสมการ (2.7) กล่าว คือ ถ้าให้ $F\mu(x) = y$ จะได้ x ในสมการ (2.7)



รูปที่ 2.9 ลักษณะสมบัติของการอัดสัญญาณตามกฎ - μ

$$x = F_{\mu}^{-1}(y) = \text{sgn}(y)\frac{1}{\mu}\left[(1+\mu)^{|y|} - 1\right] \tag{2.8}$$

กฎ - μ นี้ปัจจุบันใช้กันในทวีปอเมริกาเหนือ และประเทศญี่ปุ่นเป็นหลัก โดยที่ค่า μ ที่ใช้คือ 255 สำหรับในทวีปยุโรป และในประเทศไทยเราจะใช้กฎ -A ซึ่งโดยหลักการแล้วจะเหมือนกับกฎ - μ แต่จะแตกต่างกันในรายละเอียด โดยที่กฎ -A จะมีลักษณะสมบัติในการอัดสัญญาณเขียนเป็นรูป ฟังก์ชั่นได้ดังนี้

$$F_A(x) = \begin{cases} \operatorname{sgn}(x)\left(\dfrac{A|x|}{1+\ln(A)}\right) & 0 \le |x| < \dfrac{1}{A} \\ \operatorname{sgn}(x)\left(\dfrac{1+\ln(Ax)}{1+\ln(A)}\right) & \dfrac{1}{A} < |x| < 1 \end{cases} \tag{2.9}$$

ซึ่งจะต้องใช้ลักษณะสมบัติของการยืดสัญญาณในรูปต่อไปนี้

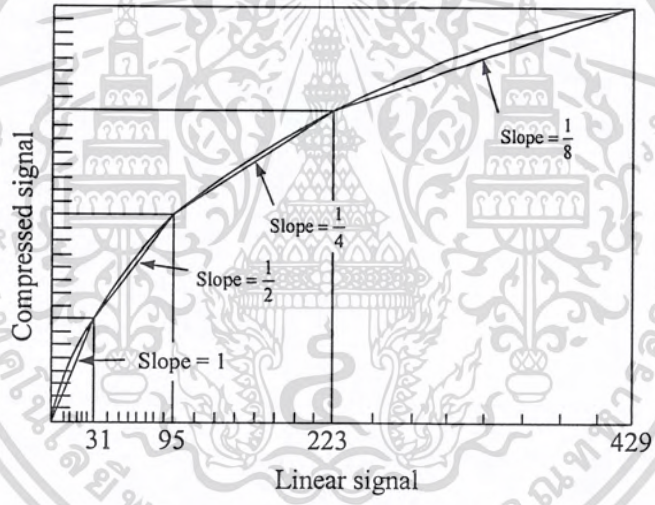
$$x = F_{AA}^{-1}(y) = \begin{cases} \operatorname{sgn}(y)\left(\dfrac{|y|\left[1+\ln(A)\right]}{A}\right) & 0 \le |x| < \dfrac{1}{A} \\ \operatorname{sgn}(y)\left(\dfrac{1+\ln|Ax|}{1+\ln(A)}\right) & \dfrac{1}{A} < |x| < 1 \end{cases} \tag{2.10}$$

ตามสมการ (2.9) นั้นจะเห็นได้ว่า $F_A(x)$ ในช่วงที่ $0 \le |x| \le 1/A$ นั้น จะเปลี่ยนแปลงตาม x แบบเป็นเชิงเส้น ส่วนในช่วงที่ $1/A < |x| \le 1$ นั้น $F_A(x)$ จะคล้ายคลึงกับของกฎ $-\mu$ ซึ่งแสดงไว้ในรูปที่ 2.9 เมื่อทำการอัดสัญญาณตามกฎ $-\mu$ หรือกฎ –A แล้ว ขั้นตอนต่อไป คือ ทำการสุ่มตัวอย่างเสร็จแล้วทำการควอนไตส์สัญญาณ และนำไปเข้ารหัสต่อไป

2.1.7 การควอนไตส์สัญญาณและการเข้ารหัสสัญญาณในกรณีที่มีการทำคอมแพนดิง

ในการควอนไตส์สัญญาณ PAM ในกรณีที่มีการอัดสัญญาณนั้น เพื่อให้การเข้ารหัสลดความยุ่งยากลงโดยทั่วไปจะทำการแบ่งช่วงของระดับสัญญาณเป็นช่วงๆ ซึ่งเรียกว่า เซ็กเมนต์ (Segment) โดยให้ในแต่ละเซ็กเมนต์สามารถประมาณเป็นการเปลี่ยนแปลงแบบเชิงเส้นได้ โดยที่ความชันของเส้นตรงในแต่ละเซ็กเมนต์ไม่เท่ากัน รูปที่ 2.10 แสดงการประมาณค่าดังกล่าวนี้ของกรณีที่เป็น กฎ -μ โดยมีค่า μ = 255 ในรูปแสดงเฉพาะ 4 เซ็กเมนต์ทางด้านบวกเท่านั้น โดยที่ความชันของเส้นตรงในแต่ละเซ็กเมนต์จะลดหลั่นลงไป ในการจัดแบ่งเซ็กเมนต์นั้นทั้งในกฎ -μ และในกฎ -A จะแบ่งเป็น 16 เซ็กเมนต์เหมือนกัน โดยที่ครึ่งหนึ่งอยู่ด้านบวก และอีกครึ่งหนึ่งอยู่ด้านลบ ดังนั้นจะต้องใช้สัญญาณไบนารี 4 บิตมาแสดงตำแหน่งของเซ็กเมนต์นั้น เมื่อทำการแบ่งเป็นเซ็กเมนต์แล้วการควอนไตส์สัญญาณภายในแต่ละเซ็กเมนต์จะสามารถใช้ควอนไตส์แบบ

เป็นเชิงเส้นได้ เนื่องจากระบบ PCM ใช้การเข้ารหัสด้วยสัญญาณไบนารี 8 บิต 4 บิตบนต้องใช้ในการแสดงตำแหน่งของเซ็กเมนต์ จึงเหลือ 4 บิตล่างไว้แสดงระดับภายในแต่ละเซ็กเมนต์ นั่น คือ ในแต่ละเซ็กเมนต์จะแบ่งออกเป็น 16 ขั้นได้ และเมื่อใช้ควอนไตส์เซอร์แบบเป็นเชิงเส้นแต่ละขั้นจะมีค่าเท่ากัน การแสดงตำแหน่งของเซ็กเมนต์นั้นจะใช้บิตบนสุด แสดงว่าเซ็กเมนต์นั้นอยู่ในซีกบวกหรือซีกลบ และใช้ 3 บิตถัดไปแสดงลำดับของเซ็กเมนต์ในซีกนั้น เมื่อใช้หลักเกณฑ์ตามที่กล่าวมานี้ในกรณีของกฎ $-\mu$ จะมีรายละเอียดของการควอนไตส์การเข้ารหัส และการถอดรหัสตามที่แสดงไว้ในตารางที่ 2.1 ในกรณีของกฎ $-\mu$ นี้ ระดับสัญญาณแอนะลอกซึ่งอยู่ในช่วง $0-1$ ขั้นนั้นจะถูกเข้ารหัสเป็น "00000000" และถ้าอยู่ทางลบจะถูกเข้ารหัสเป็น "10000000" และขั้นของการควอนไตส์ในแต่ละเซ็กเมนต์จะเป็นตามที่แสดงไว้ในตาราง และเมื่อรวมจำนวนขั้นของสัญญาณที่จะควอนไตส์และเข้ารหัสได้จะมีถึง 8,159 ขั้นในซีกบวกเพียงซีกเดียว



รูปที่ 2.10 การประมาณค่าเพื่อแบ่งเป็นเซ็กเมนต์ของ 4 เซ็กเมนต์แรกในกรณีของกฎ $-\mu$ ที่มีค่า $\mu = 255$

ตารางที่ 2.1 การเข้ารหัส และถอดรหัสของระบบ PCM $-\mu 255$

Input Amplitude Range	Step Size	Segment Code S	Quantization Code Q	Code Value	Decoder Amplitude
0-1	1		0000	0	0
1-3	2	000	0001	1	2
3-5			0010	2	4
⋮			⋮	⋮	⋮
29-31			1111	15	30
31-35	4	001	0000	16	33
⋮			⋮	⋮	⋮
91-95			1111	31	93
95-103	8	010	0000	32	99
⋮			⋮	⋮	⋮
215-223			1111	47	219
223-239	16	001	0000	48	231
⋮			⋮	⋮	⋮
463-479			1111	63	471
479-511	32	100	0000	64	495
⋮			⋮	⋮	⋮
959-991			1111	79	975
991-1055	64	101	0000	80	1023
⋮			⋮	⋮	⋮
1951-2015			1111	95	1983
2015-2143	128	110	0000	96	2179
⋮			⋮	⋮	⋮
3935-4063			1111	111	3999
4063-4319	256	111	0000	112	4191
⋮			⋮	⋮	⋮
7903-8159			1111	127	8031

ตารางที่ 2.2 การเข้ารหัส และถอดรหัสของระบบ PCM แบบกฎ –A

Input Amplitude Range	Step Size	Segment Code S	Quantization Code Q	Code Value	Decoder Amplitude
0-2	2	000	0000	0	1
2-4			0001	1	3
⋮			⋮	⋮	⋮
30-32			1111	15	31
32-34		001	0000	16	33
⋮			⋮	⋮	⋮
62-64			1111	31	63
64-68	4	010	0000	32	66
⋮			⋮	⋮	⋮
124-128			1111	47	126
128-136	8	011	0000	48	136
⋮			⋮	⋮	⋮
248-256			1111	63	252
256-272	16	100	0000	64	264
⋮			⋮	⋮	⋮
496-512			1111	79	504
512-544	32	101	0000	80	528
⋮			⋮	⋮	⋮
992-1024			1111	95	1008
1024-1088	64	110	0000	96	1056
⋮			⋮	⋮	⋮
1984-2048			1111	111	2016
2048-2476	128	111	0000	112	2112
⋮			⋮	⋮	⋮
3968-4096			1111	127	4032

รหัสที่ได้จากตารางที่ 2.1 หรือตารางที่ 2.2 ซึ่งเป็นกรณีของกฎ -A นั้น ไม่ได้หมายความว่าสัญญาณดิจิตอลที่จะถูกขับและส่งออกไปตามสายนำสัญญาณจะมีระดับสัญญาณตามรหัสนั้นๆ ทุกประการ ทั้งนี้เพราะรหัสของสัญญาณระดับ 0 ซึ่งเกิดขึ้นบ่อยที่สุดนั้นไม่เหมาะที่จะส่งเป็นสัญญาณดิจิตอลออกไป เพราะสัญญาณ "00000000" และ "10000000" นั้น เป็นสัญญาณที่เกือบจะเป็น "0" ตลอดเวลา ซึ่งสัญญาณ "0" นี้จะไม่มีข่าวสารที่ใช้ในการซิงโครไนซ์บิตได้ ดังนั้นในระบบที่ใช้งานจริงมักจะสลับรหัส "0" กับ "1" เพื่อให้มี "1" มากขึ้น

สำหรับระบบ PCM แบบ -A นั้น จะมีการเข้ารหัสตามตารางที่ 2.2 ข้อสังเกตของ กฎ –A คือ จะมีขนาดของขั้นในการควอนไตส์ในเซ็กเมนต์ที่ 1 และที่ 2 เท่ากัน คือ เท่ากับ 2 และสัญญาณที่ถอดรหัสออกมาจะไม่มีระดับ 0 สำหรับจำนวนขั้นที่แบ่งในแต่ละซีกจะเป็น 4096 ขั้น ซึ่งน้อยกว่าจำนวนขั้นในกรณีของกฎ $-\mu$

2.1.8 SQR ในกรณีของการใช้คอมแพนดิงกฎ -μ และกฎ -A

ตามรายละเอียดของการควอนไตส์ และการเข้ารหัสของกฎ $-\mu$ และกฎ -A ตามที่กล่าวไว้ในหัวข้อก่อน เนื่องจากขั้นของการควอนไตส์ ในแต่ละเซ็กเมนต์ไม่เท่ากัน จึงทำให้สัญญาณรบกวนจากการควอนไตส์ในแต่ละเซ็กเมนต์ไม่เท่ากัน โดยที่เซ็กเมนต์ที่มีการอัดสัญญาณสูง จะมีสัญญาณรบกวนจากการควอนไตส์สูงด้วย แต่เนื่องจากระดับสัญญาณที่บริเวณนั้นสูงจึงทำให้ค่า SQR นั้นมีโอกาสสม่ำเสมอมากขึ้น เมื่อเปรียบเทียบกับการควอนไตส์แบบเชิงเส้นตลอดช่วงของสัญญาณ ในการหา SQR ของกรณีที่มีการคอมแพนดิงนี้จะใช้สมการ (2.3) ในการหาค่า SQR ได้ แต่จะต้องพิจารณารายละเอียดเพิ่มเติม กล่าว คือ ถ้าเราพิจารณาสัญญาณเบสแบนด์ที่เป็นคลื่นรูปไซน์ ถ้าแอมปลิจูดของคลื่นนั้นอยู่ภายในเซ็กเมนต์ที่ 1 จะต้องพิจารณาความน่าจะเป็นที่สัญญาณนั้นจะตกอยู่ใน เซ็กเมนต์ต่างๆ แล้วหาค่าเฉลี่ยของสัญญาณรบกวนจากการควอนไตส์ออกมาก่อน จึงจะนำไปหาค่า SQR ต่อไปได้ก่อนอื่นพิจารณากรณีแรก ที่แอมปลิจูดของสัญญาณไซน์เท่ากับค่าสูงสุดของเซ็กเมนต์ที่ 1 ในกรณีของกฎ $-\mu$ ซึ่งเท่ากับขั้นที่ 31 เราจะสามารถหาค่า SQR ในกรณีนี้ได้เป็น

$$SQR(A=31) = 7.8 + 20\log\left(\frac{31}{2}\right) = 31.6\,dB \qquad (2.11)$$

ต่อไปพิจารณากรณีที่ค่าแอมปลิจูดของสัญญาณไซน์ มีระดับเท่ากับค่าสูงสุดในเซ็กเมนต์ที่ 8 ซึ่งเท่ากับขั้นที่ 8,159 ในกรณีเช่นนี้ก่อนอื่นเราจะหาค่าควอนไตส์เซชั่นนอยส์เฉลี่ยซึ่งเขียนได้ในรูปต่อไปนี้

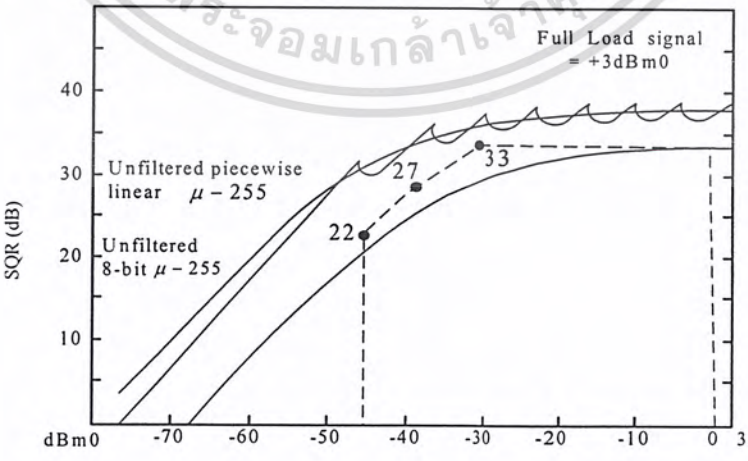
$$N_Q = \frac{1}{12}\sum_{i=1}^{8} p_i q_i^2 \tag{2.12}$$

โดยที่ p_i คือ พรอบบะบิลิตี้ที่ระดับสัญญาณจะตกอยู่ในเซ็กเมนต์ที่ i และ q_i เป็นขนาดของขั้นในการควอนไตส์ในเซ็กเมนต์ที่ i ซึ่งมีค่าเท่ากับ $2^i p_i$ นั้นสามารถหาได้จากการพิจารณาสัญญาณรูปไซน์กับการอัดสัญญาณตามตารางที่ 2.1 ซึ่งเราจะพบว่า พรอบบะลิตี้ที่จะไปอยู่ในส่วนเซ็กเมนต์ที่ 8 จะสูงที่สุด คือ $p_8 = 0.67$ เพราะเซ็กเมนต์ที่ 8 นั้นเริ่มต้นจากขั้นย่อยที่ 4,063 ถึง 8,159 ซึ่งกินช่วงกว้างมาก เมื่อทำการหาค่า SQR โดยพิจารณาตามรายละเอียดที่กล่าวมานี้จะได้ผลสำหรับกรณีนี้เป็นดังนี้

$$SQR(A = 8{,}159) = 39.3\,dB \tag{2.13}$$

สำหรับการคำนวนช่วงไดนามิกนั้นจะคำนวนจากอัตราส่วนของสัญญาณสูงสุดของเซ็กเมนต์ที่ 8 กับสัญญาณในเซ็กเมนต์ที่ SQR ยังสูงกว่าค่าที่ต้องการ อย่างไรตาม เนื่องจากขั้นย่อยสูงสุดของเซ็กเมนต์ที่ 1 สามารถให้ค่า SQR เป็น 31.6 dB ดังนั้นโดยทั่วไปการกำหนดช่วงไดนามิกจะกำหนดจากขั้นย่อยสูงสุดในเซ็กเมนต์ที่ 8 กับขั้นย่อยสูงสุดในเซ็กเมนต์ที่ 1 ดังต่อไปนี้

$$DR = 20\log\left(\frac{8{,}159}{31}\right) = 48.4\,dB \tag{2.14}$$

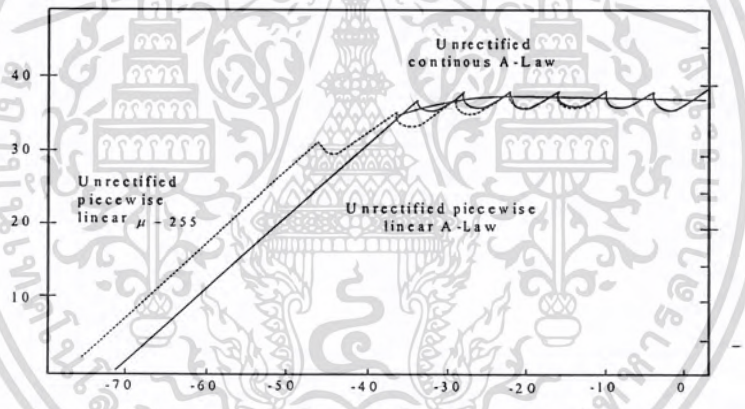


รูปที่ 2.11 SQR ของระบบ PCM กฎ - μ ที่สัญญาณขาเข้าเป็นสัญญาณไซน์ระดับต่างๆ

ซึ่งจะเห็นได้ว่า ช่วงไดนามิกที่ได้นี้กว้างเพียงพอสำหรับการใช้งานจริง รูปที่ 2.11 แสดงค่า SQR ของสัญญาณไซน์ระดับต่างๆ โดยที่ระดับสัญญาณสูงสุด คือ 3dBm

สำหรับในกรณีของกฎ -A นั้น เราจะสามารถพิจารณาได้ทำนองเดียวกับที่กล่าวมาข้างต้น และเมื่อทำการแสดงคุณสมบัติของ SQR ทำนองเดียวกับกรณีของกฎ $-\mu$ จะได้ตามรูปที่ 2.12 เมื่อเปรียบเทียบรูปที่ 2.12 กับรูปที่ 2.11 จะเห็นได้ว่าค่า SQR ที่ระดับสัญญาณสูง ๆ นั้นจะมีค่าใกล้เคียงกัน และช่วงไดนามิกนั้นถ้าคำนวณตามสมการ (2.14) กรณีของกฎ -A จะมีช่วงไดนามิกแคบกว่า แต่ถ้าพิจารณาจากรูปโดยใช้ช่วงที่ค่า SQR มีค่าสม่ำเสมอแล้ว กฎ -A จะมีช่วงไดนามิกกว้างกว่าเล็กน้อย และที่น่าสังเกต คือ กฎ -A จะมีค่า SQR ต่ำลงรวดเร็วกว่าเมื่อระดับสัญญาณต่ำลงทั้งเป็นเพราะการแบ่งขั้นย่อยของ กฎ - A หยาบกว่าของกฎ $-\mu$ ซึ่งจะส่งผลกระทบในกรณีที่ระดับของสัญญาณต่ำกล่าวนี้

ค่า SQR จะราบเรียบขึ้นเมื่อมีการใช้ฟิลเตอร์กรองเอาเฉพาะสัญญาณแบนด์ออกไปใช้



รูปที่ 2.12 ระบบ PCM และการมัลติเพลกซ์เชิงเวลา

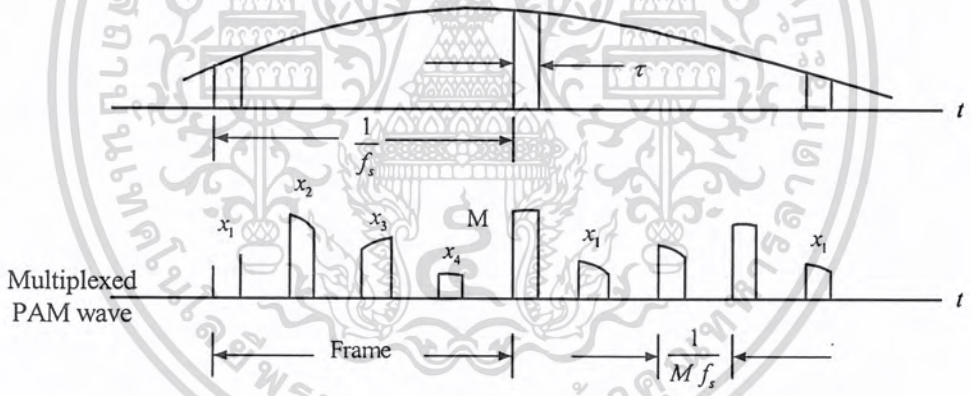
2.1.9 ระบบ PCM และการมัลติเพลกซ์เชิงเวลา

อันที่จริงระบบ PCM นั้นไม่ได้ใช้ประโยชน์สำหรับสัญญาณเสียงเพียงอย่างเดียว แต่จะใช้กับสัญญาณแอนะลอกอื่นๆ เช่น สัญญาณภาพนิ่ง หรือภาพเคลื่อนไหว เป็นต้น อย่างไรตามการแปลงสัญญาณจากสัญญาณแอนะลอก ให้เป็นสัญญาณ PCM นั้น เนื่องจากอัตราการสุ่มตัวอย่างต้องสูงกว่า 2 เท่าของความถี่เบสแบนด์ของสัญญาณแอนะลอก แล้วยังต้องทำการเข้ารหัสด้วยสัญญาณไบนารี 8 บิต ซึ่งหมายถึงอัตราการส่งสัญญาณที่คิดเป็นจำนวนบิตต่อวินาที จะต้องสูงขึ้นกว่าความถี่เบสแบนด์ไม่ต่ำกว่า 16 เท่า ยกตัวอย่างเช่น สัญญาณเสียงที่มีความถี่เบสแบนด์สูงสุดเป็น 3.4 kHz ถูกสุ่มตัวอย่างด้วยอัตราสุ่ม 8 kHz และเข้ารหัสด้วยสัญญาณเบสแบนด์ ที่กล่าวมานี้ถือว่า

เป็นของเสีย และเป็นค่าใช้จ่ายสำหรับการสร้างสัญญาณ PCM ที่หลีกเลี่ยงไม่ได้ ระบบ PCM ถึงแม้จะมีข้อเสียดังกล่าวนี้ แต่มีข้อดีที่เด่นมาก คือการที่สามารถทำการมัลติเพลกซ์เชิงเวลาได้ นอกจากนั้นยังมีความสามารถในการต้านทานสัญญาณรบกวนได้ดีอีกด้วยในหัวข้อนี้จึงกล่าวถึงการมัลติเพลกซ์เชิงเวลาและลำดับชั้นของระบบ PCM ที่ใช้ระบบโทรศัพท์ในปัจจุบัน

2.1.10 การมัลติเพลกซ์เชิงเวลา (Time division miltiplexing)

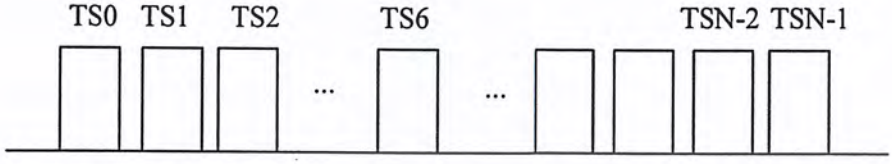
การมัลติเพลกซ์เชิงเวลา หรือ TDM นั้น หมายถึงการรวมสัญญาณเบสแบนด์หลายๆ สัญญาณเพื่อส่งผ่านช่องสัญญาณร่วมอันเดียวกัน โดยการจัดแบ่งช่องเวลา (Time Slot) ให้กับแต่ละสัญญาณเบสแบนด์อย่างเหมาะสม การมัลติเพลกซ์เชิงเวลานี้เป็นไปตามทฤษฎีสุ่มตัวอย่าง จะสามารถส่งเพียงสัญญาณที่สุ่มจากสัญญาณเบสแบนด์ซึ่งช่วงเวลาระหว่างสัญญาณสุ่มที่อยู่ติดกันจะเป็น T_S เพราะฉะนั้นถ้า T_S กว้างเพียงพอก็อาจจะแทรกสัญญาณสุ่มตัวอย่างของสัญญาณเบสแบนด์อื่นเข้าไปได้ แสดงดังรูปที่ 2.13



รูปที่ 2.13 การมัลติเพลกซ์เชิงเวลาสัญญาณ PAM

รูปที่ 2.13 แสดงการมัลติเพลกซ์เชิงเวลาจากสัญญาณ PAM โดยตรง แต่ในระบบ PCM นั้น สัญญาณสุ่มตัวอย่างแต่ละตัวอย่างจะถูกเข้ารหัสเป็น 8 บิต รูปที่ 2.14 แสดงการมัลติเพลกซ์เชิงเวลาในกรณีที่เป็นสัญญาณ PCM โดยที่แต่ละบล็อกนั้นแสดงช่องเวลาที่จัดสรรให้กับช่องสัญญาณโทรศัพท์แต่ละช่อง และจะเป็นสัญญาณ PCM 8 บิตของช่องสัญญาณเสียงอันดับนั้นๆ ในรูปที่ 2.14 แสดงกรณีที่มีช่องสัญญาณเสียงเป็นจำนวน N ช่องสัญญาณ โดยใน 1 เฟรมของสัญญาณเสียงนั้นจะมีจำนวนบิตเป็น $8 \times N$ บิต เนื่องจากในหนึ่งวินาทีต้องส่ง 8,000 ตัวอย่าง เพราะฉะนั้นในกรณีที่มัลติเพลกซ์กัน N ช่องสัญญาณเสียงนี้ อัตราการส่งข้อมูลจะเท่ากับ $8 \times N \times 8{,}000$ บิตต่อวินาที

หรือ 64× N kbps ในกรณีของการมัลติเพลกซ์ช่องสัญญาณเสียงนี้จะมีมาตรฐานที่กำหนดกันขึ้นและใช้อยู่ในประเทศต่างๆ



รูปที่ 2.14 การมัลติเพลกซ์เชิงเวลาในระบบ PCM

2.1.11 ลำดับชั้นของระบบ PCM

เพื่อให้การใช้งานของระบบ PCM มีมาตรฐานสอดคล้องกัน และมีความยืดหยุ่นในการใช้งาน CCITT ได้กำหนดมาตรฐานขึ้นมาชุดหนึ่ง ซึ่งตามมาตรฐานนี้จะแบ่งระบบ PCM ออกเป็นลำดับชั้น (Hierarchy) ตารางที่ 2.3 แสดงการแบ่งลำดับชั้นดังกล่าวนี้

ลำดับชั้นที่ 1 ซึ่งเป็นลำดับชั้นที่ต่ำที่สุดนั้น จะบรรจุช่องสัญญาณโทรศัพท์ไว้ 30 ช่อง เนื่องจากมาตรฐานนี้ได้จัดช่องเวลาที่ 0 และช่องเวลาที่ 16 ในแต่ละเฟรมให้เป็นช่องเวลาสำหรับส่งสัญญาณซิงโครไนซ์เฟรมและสัญญาณซิกแนลลิง (Signaling) ตามลำดับ ดังนั้นในแต่ละเฟรมจะมีทั้งหมด 32 ช่องเวลา ซึ่งในแต่ละช่องเวลาจะเป็นสัญญาณ 8 บิตเหมือนกัน ดังนั้นจะต้องใช้อัตราการส่งสัญญาณเป็น 64 × 32 = 2,048 kbps หรือ 2.048 Mbps สำหรับในลำดับชั้นที่ 2 และลำดับชั้นที่สูงขึ้นไปนั้น เมื่อพิจารณาที่จำนวนช่องสัญญาณโทรศัพท์จะเห็นได้ว่าสูงขึ้นที่ 4 เท่าตัว อย่างไรตามอัตราการส่งสัญญาณจะสูงขึ้นมากกว่า 4 เท่าตัวเล็กน้อย ทั้งนี้เพราะต้องมีค่าใช้จ่าย (Overhead) ในการ มัลติเพลกซ์ในลำดับชั้นที่สูงขึ้น การแบ่งเป็นลำดับชั้นจะทำให้มีความยืดหยุ่นในการใช้งาน

ยกตัวอย่างเช่น การติดต่อระหว่างชุมสายที่มีการจราจรต่ำอาจจะใช้ลำดับชั้นที่ 1 เป็นหลัก แต่ถ้าเป็นการติดต่อระหว่างชุมสายที่มีการจราจรสูงอาจจะต้องใช้ลำดับชั้นที่ 2 หรือสูงกว่านั้นขึ้นไปและอุปกรณ์อิเล็กทรอนิกส์ที่ใช้ในลำดับชั้นที่ต่ำกว่าจะสามารถใช้กับลำดับชั้นที่สูงขึ้นไปได้เพียงแต่เพิ่มมัลติเพลกเซอร์ลำดับชั้นที่สูงขึ้นเข้าไป

ตารางที่ 2.3 การแบ่งลำดับชั้นของระบบ PCM ตามมาตรฐาน CCITT

ลำดับชั้นที่	จำนวนช่องสัญญาณโทรศัพท์	อัตราการส่งสัญญาณ (Mbps)
1	30	2.048
2	120	8.448
3	480	34.368
4	1,920	139.264
5	7,680	565.148

การแบ่งลำดับชั้นของระบบ PCM นั้น นอกจากมาตรฐานของ CCITT แล้ว ยังมี มาตรฐานที่ใช้ในทวีปอเมริกาเหนือและในประเทศญี่ปุ่น ซึ่งเป็นไปตามที่แสดงไว้ในตารางที่ 2.4 และตารางที่ 2.5 ตามลำดับ จากตารางทั้งสองจะเห็นได้ว่า มาตรฐานทั้งสองระบบนี้จะแตกต่างกันตั้งแต่ลำดับชั้นที่ 3 ในปัจจุบันนี้ระบบที่เป็นมาตรฐานของ CCITT ถูกใช้มากที่สุด และประเทศไทยใช้ระบบ CCITT เช่นเดียวกัน

ตารางที่ 2.4 การแบ่งลำดับชั้นของระบบ PCM ที่ใช้ในทวีปอเมริกาเหนือ

ลำดับชั้นที่	จำนวนช่องสัญญาณโทรศัพท์	อัตราการส่งสัญญาณ (Mbps)
1(DS-1)	24	1.544
2(DS-2)	96	6.312
3(DS-3)	672	44.736
4(DS-4)	4,032	274.176
5(DS-5)	8,064	548.352

ตารางที่ 2.5 การแบ่งลำดับชั้นของระบบ PCM ที่ใช้ในประเทศญี่ปุ่น

ลำดับชั้นที่	จำนวนช่องสัญญาณโทรศัพท์	อัตราการส่งสัญญาณ (Mbps)
1	24	1.544
2	96	6.312
3	480	32.064
4	1,440	97.728
5	5,760	400.352

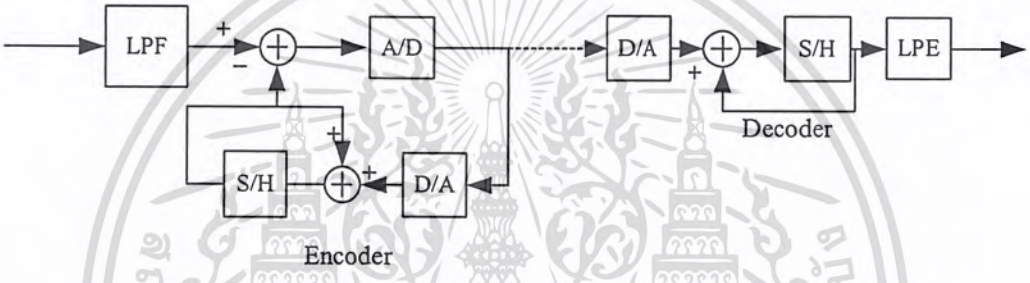
2.1.12 การลดอัตราการส่งสัญญาณข้อมูลและการลดความยุ่งยากของระบบ

สัญญาณ PCM ที่ทำการเข้ารหัสด้วยสัญญาณไบนารี 8 บิตนั้น จะต้องการอัตราการส่งสัญญาณดิจิตอลเป็น 16 เท่าของความถี่สูงสุดในเบสแบนด์เป็นอย่างน้อย อย่างไรตามสัญญาณ PAM ที่ได้จากการสุ่มตัวอย่างนั้น สัญญาณสุ่มตัวอย่างตัวที่อยู่ติดกันมักจะมีความสัมพันธ์กัน เช่น มีขนาดใกล้เคียงกัน เป็นต้น ลักษณะเช่นนี้เมื่อพิจารณาจากทฤษฎีข่าวสาร จะเข้าลักษณะที่มีความซ้ำซ้อนกัน (Redundancy) ของข่าวสาร ดังนั้นถ้าเราสามารถลดความซ้ำซ้อนอันนี้ลงได้มีโอกาสที่จะทำการเข้ารหัสด้วยจำนวนบิตที่น้อยลง ซึ่งหมายถึงจะสามารถลดอัตราการส่งสัญญาณดิจิตอลลงไปได้ การลดอัตราการส่งสัญญาณดิจิตอลลงไปได้นี้เท่ากับเป็นการลดค่าใช้จ่ายลงไปได้ โดยเฉพาะถ้าจะทำการเก็บสัญญาณเสียงลงในหน่วยความจำถึง 64 กิโลบิต หรือ 8 กิโลไบต์ ในการเก็บสัญญาณเสียงเพียง 1 วินาที ซึ่งจัดว่าเป็นการลงทุนที่สูงมาก ดังนั้นในหัวข้อนี้จึงขอกล่าวถึงวิธีการมอดูเลตแบบที่พยายามจะลดจำนวนบิตที่ต้องใช้ในการส่งต่อหนึ่งวินาที และแบบที่ลดความยุ่งยากของชุดเข้ารหัส และถอดรหัส

2.1.13 DPCM

DPCM นั้นเป็นระบบที่ดัดแปลงจากระบบ PCM โดยอาศัยการลดความซ้ำซ้อนของสัญญาณเสียงในสัญญาณสุ่มตัวอย่างที่กล่าวมาข้างต้น วิธีการที่ใช้ในการทำ DPCM นั้นจะเป็นตามแผนผังการทำงานที่แสดงไว้ในรูปที่ 2.15 คือ แทนที่จะทำการสุ่มตัวอย่างจากสัญญาณแอนะลอกโดยตรง จะทำการสุ่มตัวอย่างจากผลต่างของสัญญาณแอนะลอกในขณะนั้นกับสัญญาณแอนะลอกก่อนหน้านั้น ซึ่งจะทำให้ขนาดของสัญญาณที่สุ่มตัวอย่างออกมามีขนาดเล็กลง ซึ่งหมายถึงจะมีโอกาสเข้ารหัสด้วยจำนวนบิตที่ลดลงได้ เพื่อให้เห็นภาพที่ชัดเจน จะขอยกตัวอย่างกรณีของสัญญาณรูปไซน์ที่มีความถี่ 800 Hz ซึ่งเป็นสัญญาณที่บางครั้งใช้เป็น "เสียงทดลอง" ในระบบ

โทรศัพท์ สมมุติให้ในช่วงไดนามิกที่สนใจอยู่สามารถเข้ารหัสได้ด้วยระบบ PCM แบบเชิงเส้นได้ตลอดย่าน เมื่อนำสัญญาณนี้มาเข้ารหัสแบบ DPCM โดยให้มีขั้นของการควอนไตส์ คือ q เท่ากับของการเข้ารหัสแบบ PCM เราจะสามารถคำนวณจำนวนบิตที่จำเป็นต้องใช้ในการเข้ารหัสสัญญาณสุ่มตัวอย่างแต่ละสัญญาณ โดยพิจารณาดังต่อไปนี้ เนื่องจากในระบบ DPCM นั้นจะนำค่าความแตกต่างระหว่างสัญญาณที่เข้ามาใหม่กับสัญญาณก่อนหน้านั้นเล็กน้อยมาแปลงเป็นสัญญาณดิจิตอล ดังนั้นค่าสูงที่สุดที่จะต้องทำการแปลงเป็นสัญญาณดิจิตอลจะหาได้จากค่าสูงสุดของอนุพันธ์ของสัญญาณแอนะลอกนั้น ถ้าให้ $x(t)$ เป็นสัญญาณรูปไซน์ความถี่ 800 เฮิรตซ์ $x(t)$ จะเขียนได้เป็น $x(t) = A\sin(2\pi \times 800t)$

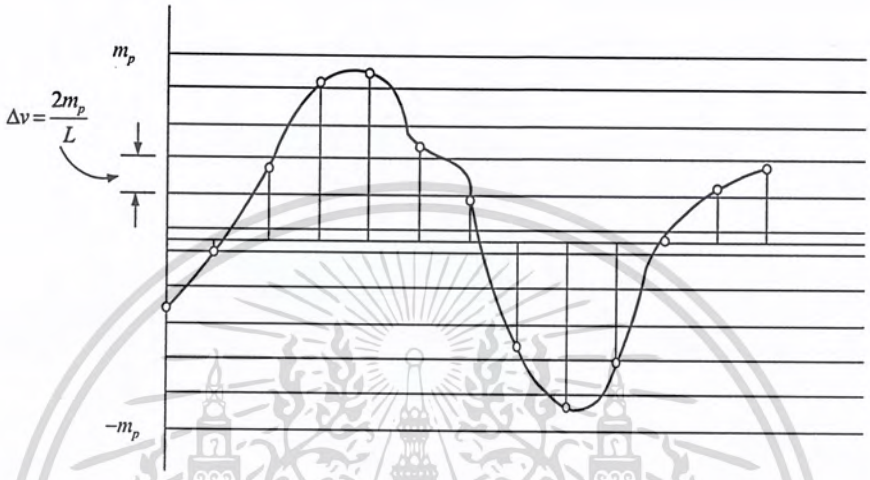


รูปที่ 2.15 แผนผังการทำงานของการทดลอง DPCM

2.1.14 การควอนไตส์สัญญาณ

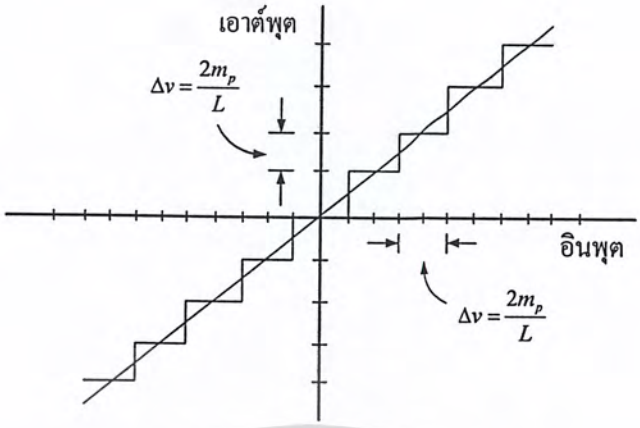
ค่าตัวอย่างสัญญาณแอนะลอกที่ชักค่าได้มานั้น จะมีขนาดความแรงที่มีค่าเปลี่ยนแปลงเป็นแอนะลอกด้วย กล่าวคือ จะมีค่าที่เป็นไปได้เป็นจำนวนนับไม่ถ้วน ดังนั้นถ้าจะกำหนดรหัสให้กับค่าตัวอย่างเหล่านี้ทั้งหมด จะต้องใช้รหัสจำนวนมหาศาล ซึ่งเป็นไปไม่ได้ในทางปฏิบัติ ดังนั้นจึงต้องมีการลดจำนวนรหัสที่ใช้ลงให้เหมาะสมกับสภาวะที่จะใช้งานจริงได้ เมื่อความจำเป็นมีดังนี้ จึงจำเป็นที่จะต้องจัดแบ่งค่าความแรงของตัวอย่างสัญญาณออกเป็นกลุ่มเท่าจำนวนรหัสที่ใช้ โดยการกำหนดให้ค่าตัวอย่างสัญญาณที่มีความแรงอยู่ในกลุ่มเดียวกันจะมีรหัสเดียวกัน จะทำให้เกิดการจัดแบ่งระดับเป็นระดับสัญญาณดิสครีตที่มีจำนวนจำกัดขึ้น ดังรูปที่ 2.16 กระบวนการที่ทำการปรับค่าของสัญญาณที่เกิดขึ้นได้อย่างต่อเนื่อง ไปเป็นค่าที่มีระดับเป็นดิสครีตที่มีจำนวนจำกัดนี้ เราได้อธิบายมาแล้วว่า คือ การทำควอนไตส์ ถ้าสัญญาณ $m(t)$ มีค่ายอดของความแรงอยู่ระหว่าง $(-m_p, +m_p)$ หากเราต้องการแบ่งกลุ่มความแรงของค่าตัวอย่างสัญญาณออกเป็น L กลุ่ม อาจทำได้โดยการแบ่งช่วงความแรงของสัญญาณออกเป็น L ช่วงเท่ากัน ซึ่งจะได้ความกว้างของแต่ละช่วง

เท่ากับ $\frac{2m_p}{L}$ แสดงในรูปที่ 2.16 ค่าตัวอย่างสัญญาณที่มีระดับความแรงที่อยู่ในช่วงเดียวกัน จะถือว่าอยู่ในกลุ่มเดียวกัน



รูปที่ 2.16 ค่าตัวอย่างสัญญาณที่ชักค่าออกมา จะถูกประมาณด้วยค่าระดับดิสครีตที่ใกล้ที่สุด

ถ้าใช้ระดับความแรงที่กึ่งกลางของแต่ละช่วงเป็นระดับการทำควอนไตส์ ค่าตัวอย่างของสัญญาณที่สุ่มได้จะถูกปรับให้มีค่าเท่ากับระดับกึ่งกลางช่วงที่บรรจุอยู่ ค่าความสัมพันธ์ระหว่างความแรงของสัญญาณที่อินพุต และเอาต์พุตของวงจรควอนไตส์เซอร์นี้ จะมีดังแสดงในรูปที่ 2.19 ค่าความแตกต่างของการปรับค่าตัวอย่างสัญญาณที่ได้ กับระดับการทำควอนไตส์แต่ละระดับนี้ จะมีค่าอยู่ในช่วง $\left(-\frac{m_p}{L}, +\frac{m_p}{L}\right)$ ค่าความแตกต่างนี้ คือ ค่าความผิดเพี้ยน หรือค่าความคลาดเคลื่อนของการทำควอนไตส์ ค่าตัวอย่างสัญญาณที่ผ่านการควอนไตส์แล้วจะถูกเข้ารหัส และส่งไปยังเครื่องรับในรูปกลุ่มของพัลส์ เครื่องรับจะมีวงจรถอดรหัส เพื่อนำค่าตัวอย่างสัญญาณกลับคืนในรูปสัญญาณดิสครีต และวงจรกรองความถี่ต่ำผ่านจะกรองสัญญาณแอนะลอก $\hat{m}(t)$ ที่ประมาณเท่ากับสัญญาณข่าวสาร $m(t)$ กลับคืนมา



รูปที่ 2.17 ความสัมพันธ์ระหว่างระดับสัญญาณอินพุต และเอาต์พุตของวงจรควอนไตส์เซอร์

เป็นธรรมดาที่สัญญาณที่ดีเทกต์ได้ทางเครื่องรับอาจจะผิดไปจากสัญญาณต้นกำเนิดทางเครื่องส่ง สาเหตุสำคัญมี 2 ประการ คือ

1) การผิดพลาดจากการทำควอนไตส์ (Quantization Error)

2) การผิดพลาดจากการดีเทกต์พัลส์ผิด (Pulse Detection Error)

โดยปกติแล้วการดีเทกต์พัลส์ผิดนั้นมีโอกาสเกิดขึ้นน้อยมาก เมื่อเทียบกับการผิดพลาดที่เกิดจากการทำควอนไตส์ จนกล่าวได้ว่าการดีเทกต์พัลส์ผิดไม่เป็นปัญหาสำคัญ และสามารถละทิ้งได้ สาเหตุสำคัญจึงมาจากการทำควอนไตส์

จะสรุปได้ว่า สามารถคำนวณได้ว่าค่าเอสเอ็นอาร์ที่เอาต์พุตของควอนไตส์เซอร์ที่มีการแบ่งระดับอย่างสม่ำเสมอนั้นจะมีค่า

$$\frac{S_o}{N_o} = 3L^2 \frac{m^2(t)}{m^2 p} \tag{2.15}$$

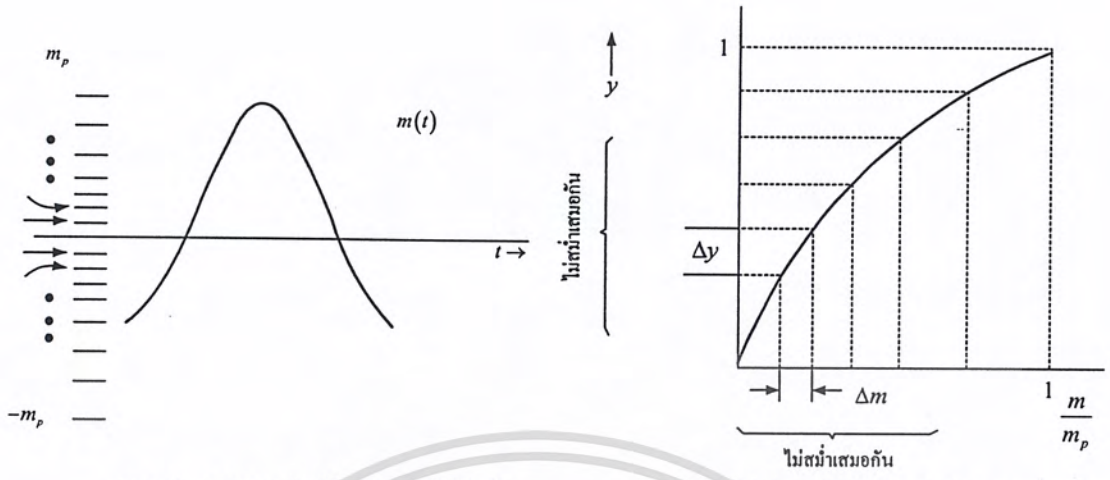
โดยในที่นี้ $m^2(t)$ คือ ค่าเฉลี่ยของกำลังสัญญาณ $m(t)$ จะเห็นได้ว่าค่าเอสเอ็นอาร์จะเพิ่มขึ้นอย่างมากในลักษณะเป็นฟังก์ชันเอกซ์โพเนนเชียล หรือฟังก์ชันการยกกำลังของจำนวนระดับ L

2.1.15 การควอนไตส์แบบไม่เชิงเส้น

เนื่องจากค่ากำลังเฉลี่ยของสัญญาณ มีค่าขึ้นกับคุณสมบัติของสัญญาณ $m(t)$ นั้น เช่นถ้าสัญญาณ m(t) คือ เสียงคน ค่า $m^2(t)$ จะแตกต่างกันไปขึ้นกับผู้พูด ซึ่งอาจมีระดับต่างกันมากถึง

40 เดซิเบล ดังนี้เป็นต้น ดังนั้นค่าเอสเอ็นอาร์ตาม จะแปรปรวนไปอย่างมากขึ้นอยู่กับชนิด และแหล่งกำเนิดของสัญญาณ $m(t)$ แม้ผู้พูดคนเดียวกัน ค่าของสัญญาณที่รับได้อาจจะมีค่า เอสเอ็นอาร์เลวลงอย่างมากเมื่อเขาพูดค่อยลง โดยปกติแล้วสัญญาณเกือบทุกชนิดเช่นเสียงพูด และสัญญาณภาพ จะมีคุณสมบัติที่ไม่เหมาะกับการทำควอนไตส์แบบเชิงเส้น กล่าวกันอย่างเคร่งครัด คือ ควอนไตส์เซอร์เชิงเส้นไม่ทำให้ค่าเฉลี่ยของกำลังสองของค่าผิดพลาดของการทำควอนไตส์สัญญาณทั่วไปนั้นเกิดมีค่าน้อยสุด ควอนไตส์เซอร์เชิงเส้น จะใช้ได้อย่างดีกับสัญญาณที่มีค่าพีเอสดีที่มีการกระจายแบบสม่ำเสมอเท่านั้น เพื่อที่จะปรับปรุงคุณสมบัติดังกล่าว จึงเกิดมีการทำควอนไตส์แบบไม่เชิงเส้น (Nonlinear Quantization) ขึ้น ในการทำควอนไตส์แบบไม่เชิงเส้นนี้ จะทำการจัดแบ่งกลุ่มของค่าตัวอย่างสัญญาณให้มีขนาดใหญ่ไม่เท่ากัน กล่าว คือ จะทำการแบ่งช่วงระหว่างระดับการควอนไตส์ให้มีค่าต่างกัน ที่ระดับความแรงสัญญาณต่ำๆ จะแบ่งช่วงให้แคบ และจะแบ่งให้กว้างขึ้นที่ระดับความแรงของสัญญาณที่มีค่ามากขึ้น การกระทำเช่นนี้จะช่วยให้ค่าเอสเอ็นอาร์ที่เอาต์พุตของควอนไตส์เซอร์ เมื่อสัญญาณ m(t) มีความแรงเฉลี่ยต่างๆ กัน มีค่าประมาณเท่ากันตลอดได้ การแบ่งช่วงระดับการควอนไตส์ให้ต่างกันนี้จะมีผลเทียบเท่ากับการทำการบีบสัญญาณอินพุต m(t) ที่ระดับความแรงต่างๆ กันด้วยอัตราที่ต่างกันก่อนที่จะป้อนสัญญาณที่ถูกบีบแล้วนั้นเข้าทำการควอนไตส์แบบเชิงเส้นอีกครั้งหนึ่ง ค่าความสัมพันธ์ระหว่างความแรงของสัญญาณอินพุตและเอาต์พุตของวงจรบีบสัญญาณ หรือคอมเพรสเซอร์ (Compressor) ดังแสดงในรูปที่ 2.20

จากผลการวิเคราะห์สัญญาณ จะพบว่า ถ้าต้องการให้ค่าเอสเอ็นอาร์ ของเอาต์พุตของ ควอนไตส์เซอร์ ไม่ขึ้นกับค่ากำลังเฉลี่ยของสัญญาณอินพุต $m^2(t)$ จะต้องทำค่าความสัมพันธ์ระหว่างเอาต์พุต y และอินพุต $\frac{m}{m_p}$ ของวงจรบีบสัญญาณให้เป็นลักษณะเชิงล็อก (Logarithmic) โดย m ในที่นี้ คือ ความแรงของ $m(t)$ รูปสมการของ y ที่เป็นฟังก์ชันของ $\frac{m}{m_p}$ ที่นิยมนำมาทำกันนั้นมีอยู่ 2 ฟังก์ชัน หรือ 2 กฎตามข้อแนะนำของซีซีไอทีที (CCITT) คือ



(ก) ระดับการทำควอนไตส์สัญญาณแบบไม่เชิงเส้น

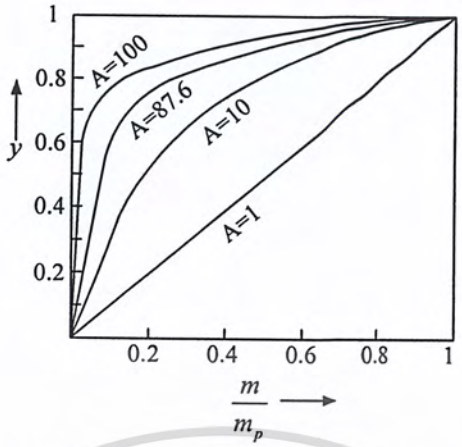
(ข) ความสัมพันธ์ระหว่างอินพุต และเอาต์พุตของวงจรบีบสัญญาณเพื่อใช้ทำการควอนไตส์ โดยวงจรควอนไตส์แบบเชิงเส้นให้ได้ผลเทียบเท่ากับการทำควอนไตส์ โดยการทำควอนไตส์เซอร์ไม่เชิงเส้น

รูปที่ 2.18 การทำควอนไตส์แบบไม่เชิงเส้น

ก) กฎเอ (A-Law) เป็นกฎ หรือฟังก์ชันที่นิยมใช้กันในยุโรป และสายการสื่อสารระหว่างประเทศส่วนใหญ่ ยกเว้นอเมริกาเหนือ และญี่ปุ่น มีรูปความสัมพันธ์ดังนี้ คือ

$$y = \begin{cases} \dfrac{A}{1+\ln A}\left(\dfrac{m}{m_p}\right), & \left|\dfrac{m}{m_p}\right| \le \dfrac{1}{A} \\ \dfrac{sgn(m)}{1+\ln A\left[1+\ln A\left|\dfrac{m}{m_p}\right|\right]}, & \left|\dfrac{1}{A}\right| \le \left|\dfrac{m}{m_p}\right| \le 1 \end{cases} \qquad (2.16)$$

โดยในที่นี้ A เป็นค่าพารามิเตอร์ที่ใช้สำหรับปรับลักษณะการบีบสัญญาณ เมื่อเลือกค่า A ต่างๆ กัน ลักษณะความสัมพันธ์ของอินพุต และเอาต์พุตของวงจรบีบสัญญาณที่ต่างกันไปดังแสดงในรูปที่ 2.21



รูปที่ 2.19 ความสัมพันธ์ของอินพุต และเอาต์พุตของวงจรบีบสัญญาณตามกฎ A

ข) กฎมิว (μ-Law) นิยมใช้กันในอเมริกาเหนือ และญี่ปุ่นมีความสัมพันธ์ดังนี้ คือ

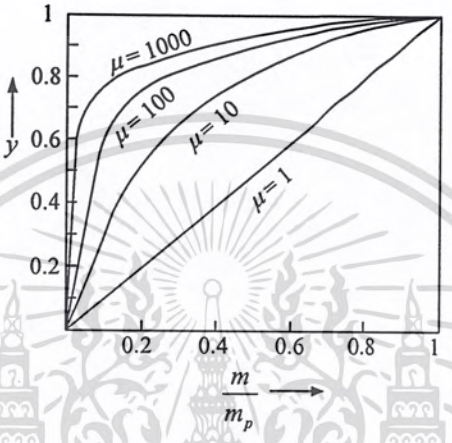
$$y = \frac{\operatorname{sgn}(m)}{\ln(1+\mu)} \ln\left(1+\mu\left|\frac{m}{m_p}\right|\right), \quad \left|\frac{m}{m_p}\right| \le 1 \tag{2.17}$$

ในที่นี้ μ เป็นค่าพารามิเตอร์ที่ใช้ปรับลักษณะของการบีบสัญญาณ เมื่อ μ มีค่าต่างๆ จะให้ความสัมพันธ์ระหว่างอินพุต และเอาต์พุตของวงจรบีบสัญญาณ ดังแสดงในรูป จะแสดงให้เห็นว่าค่าเอาต์พุตเอสเอ็นอาร์ของควอนไตส์เซอร์ที่มีการบีบสัญญาณตามกฎมิวนั้น มีค่าประมาณได้ดังนี้ คือ

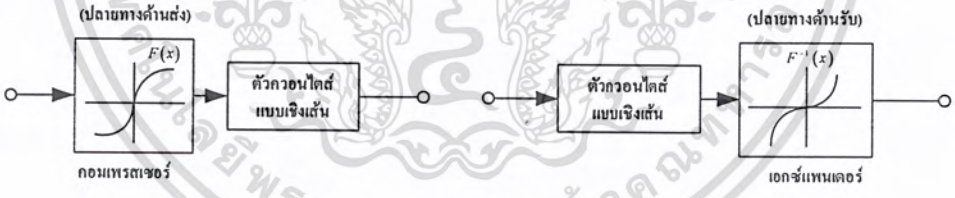
$$\frac{S_o}{N_o} \approx \frac{3L^2}{\left|\ln(1+\mu)^2\right|} \quad \text{เมื่อ} \quad \mu \gg \frac{m_p^2}{m^2(t)} \tag{2.18}$$

เป็นที่น่าสังเกตว่าคุณสมบัติตามกฎทั้งสองตามรูปและจะมีคุณสมบัติบีบสัญญาณของอินพุตที่มีความแรงมากกว่าการบีบสัญญาณอินพุตที่มีความค่อย เพราะในระบบการทำควอนไตส์แบบไม่เชิงเส้นนี้ ใช้วงจรบีบสัญญาณก่อนทำการควอนไตส์แบบเชิงเส้นดังนั้นในทางเครื่องรับ จึงมีความจำเป็นจะต้องใช้ วงจรถ่างสัญญาณ หรือ เอกซ์แพนเดอร์ ที่มีคุณสมบัติกลับกันกับคุณสมบัติ

ของวงจรบีบสัญญาณ เพื่อที่จะทำให้สัญญาณที่ถูกบีบมานั้นถูกถ่างกลับคืนสู่สภาพเหมือนเดิม ระบบที่ใช้วงจรบีบ และถ่างสัญญาณนี้มีชื่อเรียกว่าระบบที่ใช้วงจรบีบ-ถ่างสัญญาณ หรือคอมแพนเดอร์ ซึ่งส่วนประกอบของระบบนี้จะมีดังแสดงในรูปที่ 2.23 รายละเอียดเพิ่มเติมเกี่ยวกับระบบพีซีเอ็มที่ใช้วงจรบีบ-ถ่างสัญญาณนี้จะกล่าวถึงอีกในหัวข้อต่อไป



รูปที่ 2.20 ความสัมพันธ์ของอินพุต และเอาต์พุตของวงจรบีบสัญญาณตามกฎ μ



รูปที่ 2.21 ส่วนประกอบของวงจรคอมแพนเดอร์

2.1.16 แบนด์วิดท์ในการส่งสัญญาณ PCM

ในกรณีที่ส่งสัญญาณพีซีเอ็ม โดยให้รหัสคำใช้กลุ่มพัลส์ที่มีความยาว n พัลส์ จะได้รหัสที่แตกต่างกันทั้งหมดเท่ากับ 2^n รหัส ดังนั้นถ้าจำนวนระดับในการทำควอนไตส์สัญญาณทั้งหมดมี L ระดับ เราจะหาค่าจำนวนพัลส์ที่ต้องใช้ต่อรหัสได้เป็น $n = \log_2 L$ พัลส์ โดยจะต้องปัด n ขึ้นเป็นค่าจำนวนเต็มเสมอ

สัญญาณข่าวสาร $m(t)$ ที่มีแบนด์วิดท์ B เฮิรตซ์ เมื่อมีการสุ่มตัวอย่าง ด้วยอัตราไนควิสต์ คือ $2B$ ค่าต่อวินาที และใช้รหัสที่มีความยาว n พัลส์ จะทำให้สัญญาณพีซีเอ็มมีอัตราการส่งพัลส์ทั้งหมดเท่ากับ $2nB$ พัลส์ต่อวินาที

เมื่อพิจารณาใจความของทฤษฎีการสุ่มตัวอย่างว่าสัญญาณที่มีแบนด์จำกัดอยู่ B เฮิรตซ์ สามารถที่จะกำหนดได้ด้วยค่าตัวอย่างของมันที่เวลาห่างกันไม่เกิน $1/(2B)$ วินาทีนั้น ได้บอกให้รู้ว่าสัญญาณที่มีแบนด์จำกัดอยู่ B เฮิรตซ์นี้ สามารถจะกำหนดได้ด้วยข้อมูลจากตัวมันเองเพียงจำนวน $2B$ อนุภาค (Element) ใน 1 วินาที หรือกล่าวอีกนัยหนึ่งได้ว่าสัญญาณที่มีแบนด์จำกัดอยู่ B เฮิรตซ์นั้นมีข้อมูลที่สำคัญ ซึ่งเป็นอิสระไม่ขึ้นแก่กันอยู่อย่างมากสุดเพียง $2B$ อนุภาคเท่านั้นใน 1 วินาที ซึ่งอาจกล่าวอีกทัศนะหนึ่งได้ว่าในการส่งข้อมูลนั้น เราสามารถที่จะส่งข้อมูลอิสระได้เป็นอัตราสูงถึง 2 อนุภาคใน 1 วินาทีในแบนด์วิดท์ 1 เฮิรตซ์ ในที่นี้การส่ง 1 อนุภาคของข้อมูล อาจหมายถึงการส่งพัลส์ 1 พัลส์ได้ ดังนั้นคำว่าอนุภาคของข้อมูลอิสระ จึงอาจหมายถึง พัลส์ที่มีขนาดไม่ขึ้นแก่กันได้ เพราะฉะนั้นโดยอาศัยทฤษฎีการชักตัวอย่าง เราสามารถกล่าวได้ว่า เราสามารถที่จะรับสัญญาณพัลส์ที่มีขนาดไม่ขึ้นแก่กัน จำนวนมากถึง $2B$ พัลส์ต่อวินาทีผ่านช่องสัญญาณ ซึ่งมีแบนด์วิดท์ B เฮิรตซ์ได้โดยไม่ผิดพลาด ซึ่งที่กล่าวมานี้เป็นอัตราสูงสุดของการส่งสัญญาณตามทฤษฎี ส่วนในทางปฏิบัตินั้นจะมีข้อจำกัดอื่นๆ มาทำให้เราไม่สามารถส่งพัลส์ได้มากเท่าตามทฤษฎี ปัจจุบันทางปฏิบัติที่ทำได้สามารถส่งพัลส์ผ่านช่องสัญญาณที่มีแบนด์วิดท์ B เฮิรตซ์ ได้ด้วยอัตรา k พัลส์ต่อวินาที $(1 \le k \le 2)$ โดยไม่มีการรบกวนกัน และกัน ทั้งนี้ขึ้นกับรูปแบบของการส่งสัญญาณ และรูปร่างของพัลส์ที่ใช้ สำหรับการส่งสัญญาณแบบไบโพลาร์ที่นิยมใช้กันในระบบพีซีเอ็มนั้นมีค่า $k = 1$ ดังนั้นเพื่อที่จะส่งสัญญาณจำนวน $2nB$ พัลส์ต่อวินาทีจึงต้องใช้แบนด์วิดท์ $2nB$ เฮิรตซ์ และสำหรับแบนด์วิดท์ที่จำเป็นในการส่งสัญญาณแบบอื่นจะได้กล่าวถึงต่อไป

2.1.17 เอสเอ็นอาร์ของระบบ PCM

ค่าเอสเอ็นอาร์ที่เอาต์พุตของระบบพีซีเอ็มหาได้โดยอาศัยที่เรารู้ว่า $L = 2^n$ ประกอบกับ (2.15) และ (2.18) สมการทั้งสองนี้จะเขียนในรูปสมการใหม่ได้เป็น

$$\frac{S_o}{N_o} \approx 3M\left(2\right)^{2n} \tag{2.19}$$

โดยในที่นี้ $M = \frac{\overline{m^2(t)}}{m_p^2}$ สำหรับระบบที่ใช้การทำควอนไตส์แบบเชิงเส้นตาม (2.15) หรือ $M = \frac{1}{\left[\ln(1+\mu)\right]^2}$ สำหรับระบบที่ใช้การทำควอนไตส์แบบไม่เชิงเส้นตาม (2.18) ดังนั้นค่าเอสเอ็นอาร์ที่คิดในหน่วยเดซิเบลจะมีค่าเป็น

$$\left.\frac{S_o}{N_o}\right|_{dB} = 10\log\left(\frac{S_o}{N_o}\right)$$

$$= \alpha + 6n \text{ เดซิเบล}$$

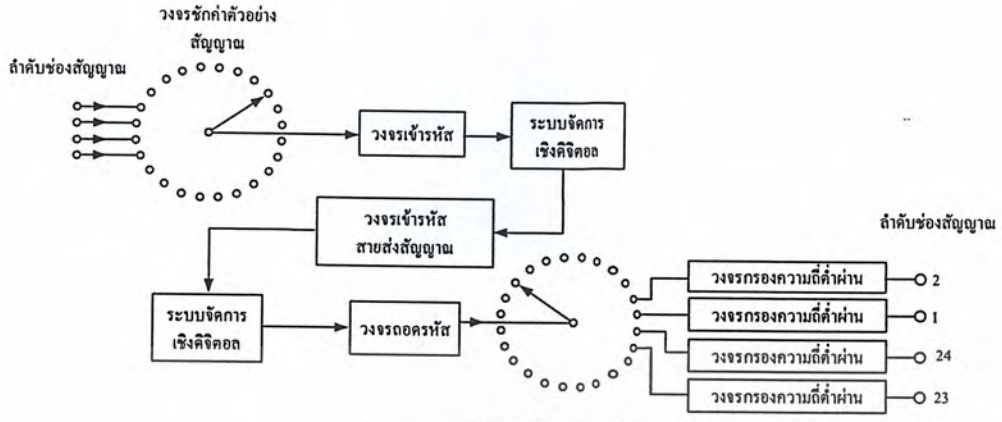
โดยในที่นี้

$$\alpha = 10\log 3(M)$$

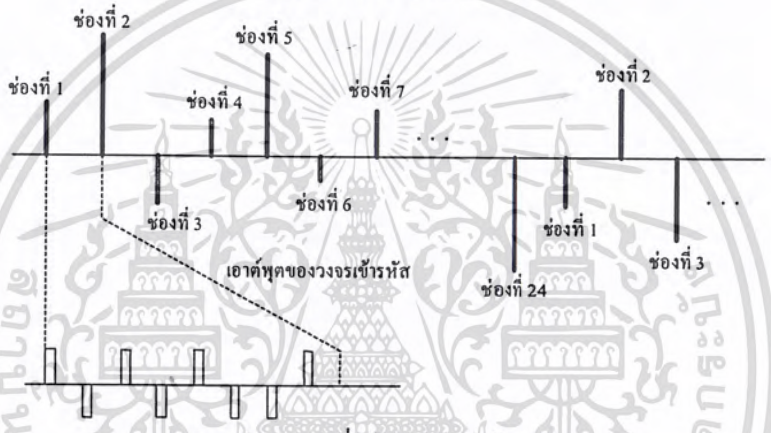
เมื่อคำนึงถึงคำอธิบายจากหัวข้อที่ผ่านมาจะรู้ว่าระบบพีซีเอ็มนี้ เป็นระบบที่มีอัตราการส่งพัลส์เท่ากับ $2nB$ เฮิรตซ์ ซึ่งอัตราการส่งพัลส์ของระบบนี้เป็นตัวแสดงถึงค่าแบนด์วิดท์ของระบบ จะเห็นได้ว่าค่าแบนด์วิดท์จะแปรผันโดยตรงกับค่า n และเมื่อพิจารณา จะพบว่าการเพิ่มค่าพัลส์ต่อวินาทีนั้น n จะทำให้ค่าเอสเอ็นอาร์ของระบบเพิ่มขึ้นอย่างเอกซ์โพเนนเซียล ดังนั้นจึงกล่าวได้ว่าการยอมเพิ่มแบนด์วิดท์ของระบบเพียงเล็กน้อย จะทำให้ค่าเอสเอ็นอาร์ของระบบเพิ่มขึ้นได้อย่างมาก ซึ่งจะเห็นได้ชัดว่าการเพิ่ม n (จำนวนพัลส์ หรือหลักในคำรหัส) ขึ้นอีกหนึ่ง (เป็น $1+n$) จะทำให้สามารถเพิ่มค่าเอสเอ็นอาร์ขึ้นได้ถึง 4 เท่าตัว (6 เดซิเบล)

2.1.18 การจัดจังหวะสัญญาณในระบบ PCM

สัญญาณพีซีเอ็มนั้นสามารถนำมาทำการมัลติเพลกซ์แบบทีดีเอ็มได้ เช่นเดียวกับสัญญาณพีเอเอ็ม แผนผังระบบการมัลติเพลกซ์สัญญาณพีซีเอ็มแบบง่ายๆ มีดังแสดงในรูป 2.24 (ก) ตามรูปนี้ได้จัดระบบไว้เพื่อการมัลติเพลกซ์สัญญาณจำนวน 24 ช่อง รูปแบบของสัญญาณที่ออกมาจากสวิตช์หมุน (Commutator) จะมีลักษณะดังแสดงในรูปที่ 2.24 (ข) ค่าตัวอย่างสัญญาณที่ถูกชักค่าออกมาแต่ละค่าจะถูกเข้ารหัสไปทีละค่าจนครบ ถ้าเป็นการใช้สัญญาณชนิด 8 บิต จะได้รหัสของค่าตัวอย่างสัญญาณแต่ละตัวเรียงกันอยู่ในช่วงเวลาดังแสดงในส่วนล่างของรูป



(ก) ระบบ พีซีเอ็ม-ทีดีเอ็ม



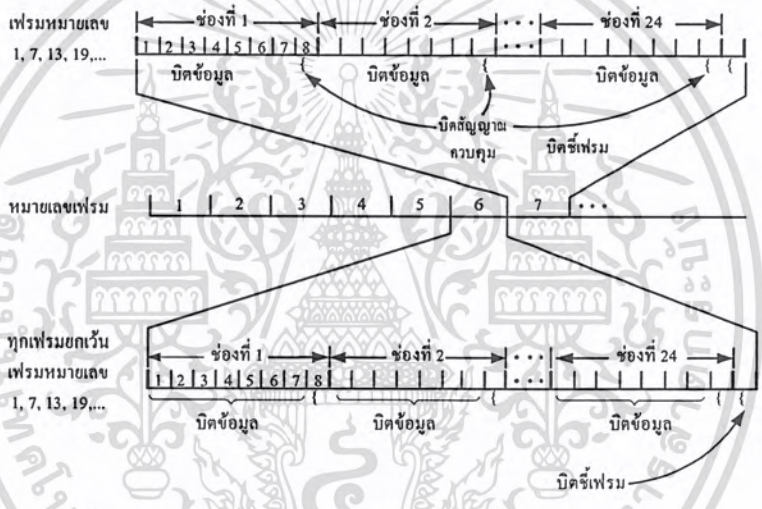
(ข) สัญญาณที่เข้าสู่วงจรเข้ารหัส

รูปที่ 2.22 ระบบการมัลติเพลกซ์ในระบบพีซีเอ็ม

รหัสไบนารีที่สมนัยกับค่าตัวอย่างสัญญาณ แต่ละค่าจากช่องการสื่อสารทั้ง 24 ช่องจะถูกมัลติเพลกซ์กันเป็นลำดับ ช่วงเวลาที่บรรจุรหัสพัสส์ที่เกิดจากค่าตัวอย่างสัญญาณจาก 24 ช่องสัญญาณช่องละ 1 ตัวอย่าง มีชื่อเรียกว่า เฟรม (Frame) กล่าวได้ว่าแต่ละเฟรมจะบรรจุบิตข้อมูลเท่ากับ $24 \times 8 = 192$ บิต ถ้าระบบมัลติเพลกซ์นี้ใช้อัตราการสุ่มตัวอย่างสัญญาณ 8,000 ครั้งต่อวินาที แต่ละเฟรมจะมีคาบเวลา 125 ไมโครวินาที และเนื่องจากในกระบวนการส่งสัญญาณนั้นทางเครื่องรับมีความจำเป็นที่จะต้องรู้ว่าจุดเริ่มต้นของเฟรมอยู่ตรงตำแหน่งเวลาใด เพื่อที่จะได้ทำการแยกชุดบิตของข้อมูลได้อย่างถูกต้อง ด้วยเหตุนี้ในทางปฏิบัติจึงต้องเพิ่มบิตชี้เฟรม หรือ เฟรมมิงบิต (Framing Bit) ที่ปลายสุดของแต่ละเฟรมขึ้นอีก 1 บิต ซึ่งหมายความว่าใน 1 เฟรมจะต้องบรรจุบิตทั้งหมด 193 บิตต่อเฟรม บิตชี้เฟรมนี้จะถูกกำหนดโดยมีหลักเกณฑ์การเลือกให้ลำดับของบิตชี้เฟรมนี้ที่เกิดขึ้นที่ปลายของแต่ละเฟรม ก่อเกิดเป็นรูปแบบของรหัสที่ไม่ปรากฏซ้ำกับค่ารหัสของ

ตัวอย่างสัญญาณที่ใช้อยู่เช่น ลำดับ 101010.... นั้นไม่น่าจะเกิดขึ้นในระบบสัญญาณเสียงเพราะการที่เกิดเช่นนี้ได้หมายถึงมันจะต้องมีส่วนประกอบของความถี่ 4 กิโลเฮิรตซ์

ลำดับรหัสที่เกิดจากบิตชี้เฟรมนี้ จะถูกตรวจโดยวงจรตรรกของเครื่องรับ ถ้าเครื่องตรวจจับหาไม่พบรูปแบบของลำดับดังกล่าวเท่ากับว่าเครื่องรับได้ตรวจพบว่า การซิงโครไนซ์สัญญาณของระบบนั้นเสียไป เครื่องรับจะต้องสำรวจหาบิตชี้เฟรมนี้ใหม่ ซึ่งมันจะต้องเสียเวลาไปบ้าง ยกตัวอย่างระบบพีซีเอ็มจริงนั้นจะใช้เวลาประมาณ 0.4 ~ 0.6 มิลลิวินาที ที่จะค้นหาบิตชี้เฟรมได้ และจะเสียเวลาถึงประมาณ 50 มิลลิวินาที ในการสร้างเฟรมใหม่รูปลักษณะการจัดเฟรมของสัญญาณมีดังแสดงในรูปที่ 2.23



รูปที่ 2.23 รูปแบบการจัดสัญญาณในระบบ T-1

2.2 MP3 (MPEG Audio Layer3)

MP3 คือ ฟอร์แมตของไฟล์ชนิดหนึ่ง ซึ่งบรรจุข้อมูลเสียง ที่ใช้คอมพิวเตอร์เล่นไฟล์ชนิดนี้ ใช้เทคโนโลยีบีบอัดข้อมูลให้มีขนาดเล็กมากๆ โดยการแปลงไฟล์นามสกุล WAV ในอัตราส่วน 1 : 12 แต่คุณภาพเสียงไม่ได้สูญเสียไปจากการลดขนาดนั้น และมีคุณภาพเทียบเท่ากับเพลงที่เปิดจากแผ่นซีดีโดยทั่วไป

อย่างไรตาม เมื่อนำไฟล์ MP3 ไปเล่น ในเครื่องคอมพิวเตอร์รุ่นเก่า ๆ ตั้งแต่ซีพียู 486/100 MHz. ลงไป จะได้เสียงในลักษณะกระตุกๆ หรือใช้การไม่ได้เลย เพราะการเล่น MP3 จะไปถ่วงความเร็วของซีพียูลงถึง 5 – 20 % ซึ่งขึ้นอยู่กับโปรแกรมที่ใช้งานนั้นด้วย

MP3 เป็นเทคโนโลยีบีบอัดข้อมูลที่ถูกพัฒนามาจาก MPEG รุ่นก่อนๆ โดยเผยแพร่ออกสู่สาธารณะครั้งแรกผ่านอินเทอร์เน็ตเมื่อราวปี คศ. 1996 ทางเว็บไซต์ http://layer3.org/

รูปที่ 2.24 โปรแกรม Winamp Minibrowser

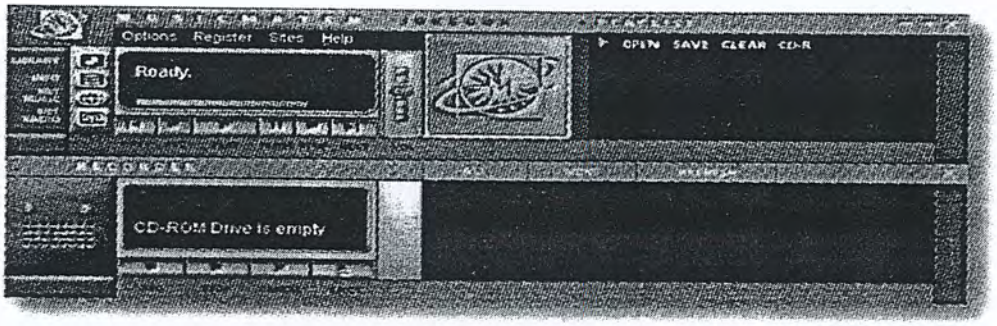
จากคุณภาพเสียงที่ดี และขนาดที่เล็กมาก ทำให้ MP3 ได้รับความนิยมสูงสุดจากผู้ใช้คอมพิวเตอร์ทั้งหลาย มีการเผยแพร่อย่างกว้างขวางในเครือข่ายอินเทอร์เน็ตโดยการแจกจ่ายให้ดาวน์โหลดฟรีทางเว็บไซต์ต่างๆ

2.2.1 อัดข้อมูลเสียงแล้วแปลงให้เป็นไฟล์ MP3

ขั้นตอนในการอัดข้อมูลเสียงให้เป็นไฟล์ MP3 โดยจะกล่าวโดยคร่าว ๆ ดังต่อไปนี้

1) อัดเสียงของเข้าสู่เครื่องคอมพิวเตอร์ โดยใช้โปรแกรมอัดเสียงพวก Rippers ซึ่งจะแปลงเสียงให้อยู่ในรูปของไฟล์ WAV ซึ่งคุณภาพของเสียงต้นฉบับ

2) จากนั้น แปลงไฟล์ WAV มาเป็นไฟล์ MP3 ด้วยโปรแกรม Encoder แม้ว่าไฟล์ WAV จะเป็นมาตรฐานของไฟล์เสียงที่ใช้ในเครื่องคอมพิวเตอร์ตาม แต่ไฟล์ WAV มีขนาดประมาณ 10 MB ต่อเวลา 1 นาที นั่นคือหากเสียงยาว 4 นาที จะต้องใช้พื้นที่ฮาร์ดดิสก์ถึง 40 MB และขนาดของขนาดไฟล์ WAV จะทำให้เกิดความล่าช้าในการส่งข้อมูลผ่านเครือข่ายคอมพิวเตอร์ ในขณะที่การแปลงเป็นไฟล์ MP3 ข้อมูลดังกล่าวจะใช้พื้นที่ประมาณ 4 MB เท่านั้น จึงมีความเหมาะสมในการใช้งานมากกว่าไฟล์ WAV



รูปที่ 2.25 โปรแกรม Musicmatch Jukebox

สำหรับโปรแกรม Ripper และ Encoder มีอยู่หลายตัว บางตัวเป็นแค่ Ripper บางตัวเป็นแค่ Encoder แต่มีบางตัวที่สามารถทำงานได้ทั้งสองแบบ

2.2.2 การบีบอัดข้อมูลเสียงแบบเอ็มเป็ก (MPEG Audio Compression)

การบีบอัดข้อมูลเสียงแบบเอ็มเป็ก (MPEG Audio Compression) เป็นวิธีการบีบอัดข้อมูลเสียงแบบดิจิตอล (Digital Compression) ที่มีความเหมือนจริงสูง (High Fidelity : HiFi) การบีบอัดข้อมูลเสียงแบบเอ็มเป็กสำเร็จลงในปี 1993 โดยคณะกรรมการสากลแห่งการเชี่ยวชาญการบีบอัดเสียงอย่างเหมือนจริง ซึ่งรู้จักกันดีในชื่อว่า Motion Picture Expert Group (MPEG) และมาตรฐาน ISO/IEC 1172-3

2.2.3 หลักการพื้นฐานและการใช้งาน

หลักการบีบอัดข้อมูลแบบเอ็มเป็ก จะใช้ประโยชน์จากขีดจำกัดในการได้ยินเสียงของมนุษย์โดยไม่จัดเก็บข้อมูลเสียงที่มนุษย์ฟังไม่ได้ยิน เพื่อให้ข้อมูลมีขนาดเล็กลง ซึ่งใช้หลักการว่าถ้าได้ยินเสียง 2 เสียงที่มีความถี่ใกล้เคียงกัน โดยให้เสียงที่ 2 ดังกว่า จะได้ยินเสียงที่ 2 เพียงเสียงเดียว ดังนั้นในการจัดเก็บข้อมูลสามารถตัดข้อมูลเสียงที่ค่อยกว่าออกไปได้บางส่วน การบีบอัดข้อมูลแบบเอ็มเป็กสามารถเลือกวิธีการบีบอัดได้หลายๆ แบบ ดังนี้

อัตราการสุ่มข้อมูล (Sampling Rate) 32, 44.1 หรือ 48 กิโลเฮิรตซ์

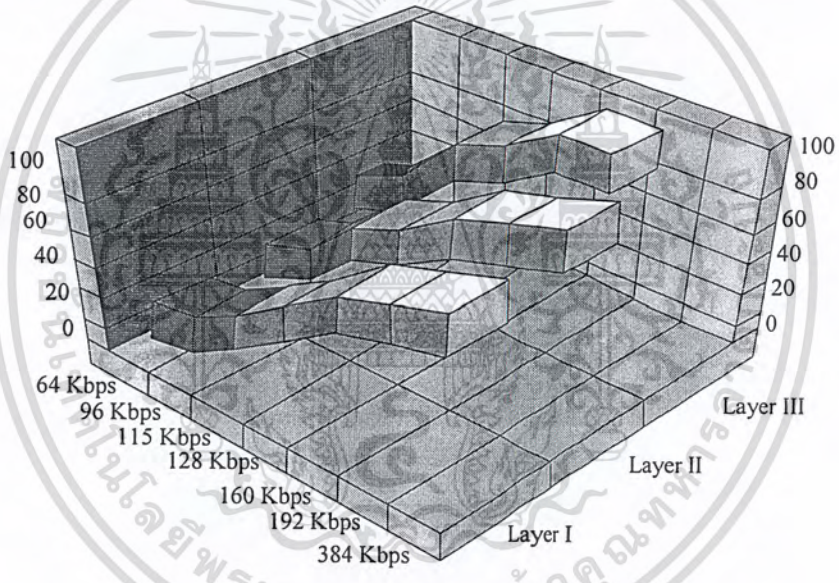
รองรับระบบเสียงได้ทั้ง 1 และ 2 ช่องสัญญาณเสียง ซึ่งมีอยู่ 4 แบบ คือ แบบโมโนเดี่ยว (Monophonic mode) แบบโมโนคู่ (Dual Monophonic Mode) แบบสเตอริโอ (Stereo Mode) แบบจอยท์สเตอริโอ (Joint Stereo Mode)

สามารถเลือกค่าอัตราการส่งข้อมูล (Bitrate) ได้ตั้งแต่ 32 ถึง 244 Kbit/sec ต่อหนึ่งช่องสัญญาณเสียง ขึ้นอยู่กับอัตราการสุ่มตัวอย่าง (Sampling Rate)

รูปแบบการเข้ารหัสไฟล์เอ็มเป็ก มี 3 เลเยอร์ (Layer) คือ เลเยอร์ 1 เลเยอร์ 2 และ เลเยอร์ 3 แต่ละเลเยอร์มีลักษณะต่างกันดังนี้

เลเยอร์ 1 มีความซับซ้อนน้อยที่สุด ต้องใช้อัตราการส่งข้อมูลสูงถึง 384 กิโลบิตต่อวินาที (kbps) เพื่อที่จะให้ได้เสียงคุณภาพเสียงสูง

เลเยอร์ 2 มีความซับซ้อนมากขึ้น คุณภาพเสียงสูงกว่า เลเยอร์ 1 ใช้อัตราการส่งข้อมูลที่ 160 กิโลบิตต่อวินาทีจะได้เสียงคุณภาพสูง และที่ 192 กิโลบิตต่อวินาทีจะได้คุณภาพเสียงที่ไม่แตกต่างจากเสียงต้นแบบ

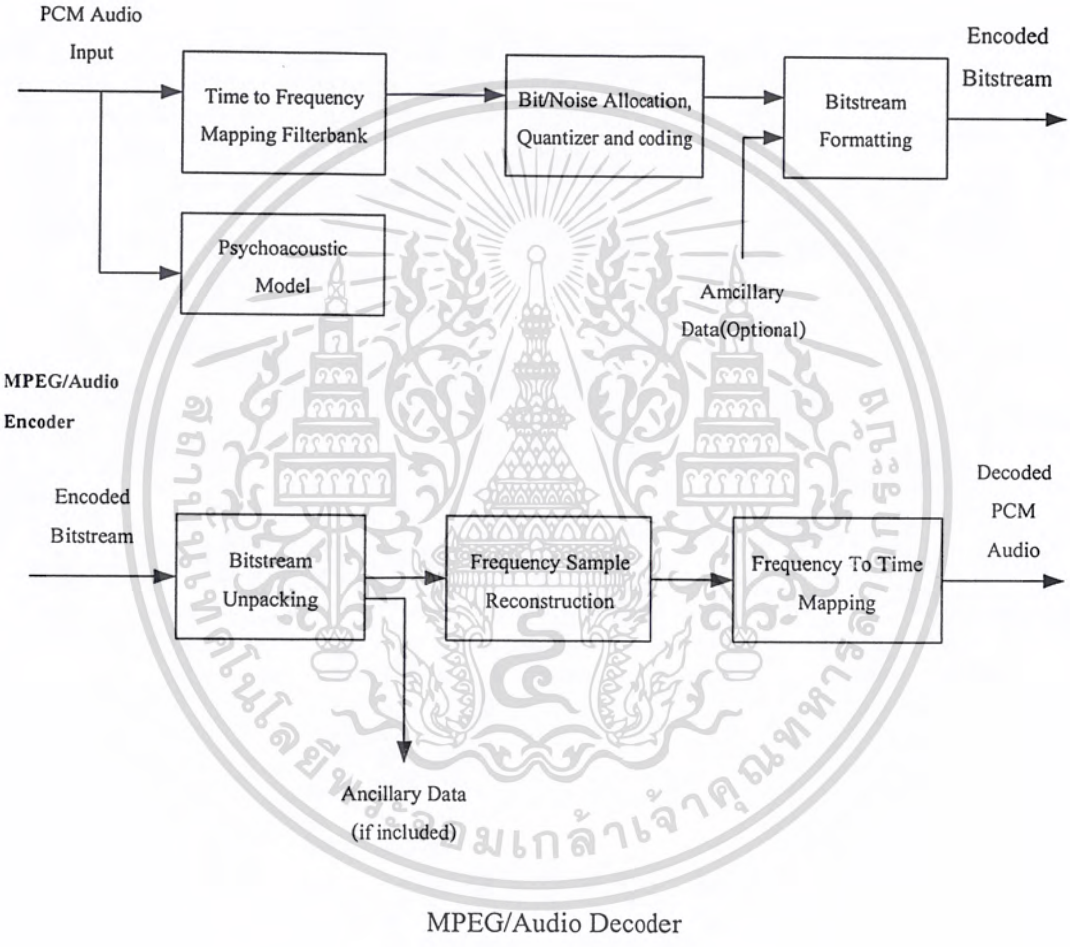


รูปที่ 2.26 กราฟเปรียบเทียบอัตราส่วนการบีบอัดเสียงของการเข้ารหัสเลเยอร์ต่างๆ เมื่อคุณภาพเสียงต้นแบบจากคอมแพ็คดิสก์

เลเยอร์ 3 มีความซับซ้อนมากที่สุดแต่สามารถให้คุณภาพเสียงที่ดีที่สุด ใช้อัตราการส่งข้อมูลที่ 160 กิโลบิตต่อวินาทีให้คุณภาพเสียงที่ไม่แตกต่างจากเสียงต้นแบบ สามารถแสดงความสัมพันธ์ของอัตราการส่งข้อมูลเสียง (กิโลบิตต่อวินาที) สำหรับการเข้ารหัสเลเยอร์ต่างๆ กับคุณภาพเสียงต้นแบบได้ดังรูปที่ 2.26

2.2.4 การเข้ารหัสแบบเอ็มเป็กโดยทั่วไป

การเข้ารหัสเอ็มเป็กโดยทั่วไป จะตัดข้อมูลเสียงที่จัดเก็บบางส่วนออกแต่สามารถคงรายละเอียดของเสียงที่ได้ยินไว้เท่าเดิม ดังรูปที่ 2.27 แสดงแผนผังการทำงาน (Block Diagram) ของการเข้ารหัส และถอดรหัสแบบเอ็มเป็ก



รูปที่ 2.27 แผนผังการทำงานของการเข้ารหัส และถอดรหัสแบบเอ็มเป็ก

กระบวนการเข้ารหัสมีขั้นตอนดังต่อไปนี้

1) เริ่มจากข้อมูลเสียงจะถูกส่งไปยังโพลีเฟสฟิลเตอร์แบงค์ (The Polyphase Filter Bank) เพื่อแบ่งข้อมูลเสียงออกเป็นหลายๆ ช่วงความถี่ (Subband of Frequency)

2) แบ่งค่าของสัญญาณต่อการกำหนดค่ามาสก์กิงเทรดโฮล (Masking Threshold : SMR) ของแต่ละช่วงความถี่

3) ข้อมูลจากโพลีเฟส ฟิลเตอร์แบงค์ ถูกส่งไปส่วนแยกบิต (Bit/Noise Allocation) และปรับข้อมูล (Quantizing)

กระบวนการถอดรหัสบิตสตรีม (Bitstream) จะนำข้อมูลมาผ่านกระบวนการปรับข้อมูลย้อนกลับ และทำการรวมข้อมูลแต่ละช่วงความถี่กลับเป็นข้อมูลเสียง

2.2.5 โพลีเฟส ฟิลเตอร์แบงค์

โพลีเฟสฟิลเตอร์แบงค์ (The Polyphase Filter Bank) เป็นหลักการทั่วไปของการเข้ารหัสแบบเอ็มเป็ก โพลีเฟส ฟิลเตอร์แบงค์ จะแบ่งสัญญาณเสียงออกเป็นช่วงความถี่ที่มีความกว้าง (Bandwidth) ออกเป็น 32 ช่วงความถี่ย่อย (Subband) ที่เท่ากัน

1) คุณสมบัติของโพลีเฟสฟิลเตอร์แบงค์

1.1) ความกว้างของช่วงความถี่แต่ละช่วงเท่ากัน

1.2) ฟิลเตอร์แบงค์ และการแปลงกลับเป็นการแปลงกลับแบบมีการสูญเสีย (Lossy)

1.3) ย่านความถี่แต่ละย่านจะมีการเหลื่อมซึ่งกันและกัน

2) ขั้นตอนการทำโพลีเฟสฟิลเตอร์แบงค์

การทำโพลีเฟสฟิลเตอร์แบงค์ เป็นกระบวนการแยกสัญญาณที่ถูกสุ่มตัวอย่าง (Sampling) ด้วยความถี่สุ่มตัวอย่าง ออกเป็น 32 ย่านความถี่ โดยแต่ละช่วงความถี่นี้จะมีช่วงความถี่เท่ากัน Fs/32 ขั้นตอนการวิเคราะห์มีดังนี้

1) ข้อมูลเสียงเข้า 32 ข้อมูลสุ่มตัวอย่าง (Sampling)

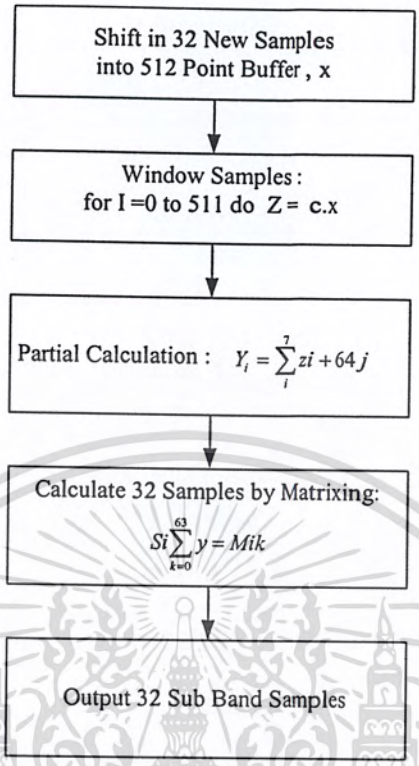
2) สร้างข้อมูลบัฟเฟอร์ (Buffer) ไว้สำหรับข้อมูลสุ่มตัวอย่าง 512 ข้อมูล โดยจะเลื่อนข้อมูลเข้า 32 ข้อมูล เข้ามาในบัฟเฟอร์ ในตำแหน่งที่ 0-31

3) คูณค่าข้อมูลอินพุต 512 ค่า กับสัมประสิทธิ์ (C) 512 ค่า

4) คำนวณค่า Y ตามสมการที่ให้มา (64 ค่า)

5) คำนวณค่า Subband Sample : S 32 ค่าออกมา

$$St(i) = \sum_{k=0}^{63} \sum_{j=0}^{7} M[i][k] \times \left(C[k+64] \times X[k+64j] \right) \qquad (2.20)$$



รูปที่ 2.28 ผังงานของโปรแกรมอธิบายขั้นตอนการทำโพลีเฟส ฟิลเตอร์แบงค์

โดย

i คือ ดรรชนีด้านความถี่ตั้งแต่ 0-31

St คือ สัญญาณเอาต์พุตที่ถูกกรองความถี่แล้ว สำหรับย่านความถี่ i ที่เวลา t โดยที่ t เป็นเลขจำนวนเต็ม

$C[n]$ คือ สัมประสิทธิ์ของวินโดว์ ลำดับที่ n ถูกนิยามไว้ในมาตรฐาน

$X[n]$ คือ สัญญาณเสียงที่ถูกอ่านจากอินพุตบัฟเฟอร์ที่เก็บอินพุตไว้ 512 สัญญาณความถี่

$$M[i][k] = \cos\frac{(2i+1)\times(n-16)\times\pi}{64}$$

สมการที่ 2.20 เป็นการคำนวณที่สามารถลดจำนวนครั้งการคูณได้มากที่สุด เนื่องจากพจน์ที่อยู่ในวงเล็บไม่เป็นฟังก์ชั่นของ i และ $M[i][k]$ ไม่ขึ้นกับค่า j ดังนั้นการคำนวณทุกๆ 32 สัญญาณสุ่มตัวอย่างเอาต์พุตจะคูณเพียง $512+32\times64=2560$ ครั้ง และการบวก $64\times7+32\times63=2464$ ครั้ง หรือประมาณ 80 ครั้งต่อสัญญาณเอาต์พุต 1 สัญญาณ

ข้อสังเกตจากการทำฟิลเตอร์แบงค์ทุกๆ 32 สัญญาณอินพุตจะได้สัญญาณเอาต์พุต จำนวน 32 สัญญาณ ด้วยเหตุนี้ทั้ง 32 ย่าน เมื่ออินพุตเข้า 32 สัญญาณ

จากสมการที่ 2.20 สามารถเขียนเป็นสมการพื้นฐานในรูปแบบผลบวกประสาน (Convolution) ได้ คือ

$$St[i]=\sum_{n=0}^{511} x[t-n]\times Hi[n] \tag{2.21}$$

$$Hi[n]=h[n]\times\cos\left[\frac{(2\times i+1)\times(n-16)\times\pi}{64}\right]$$

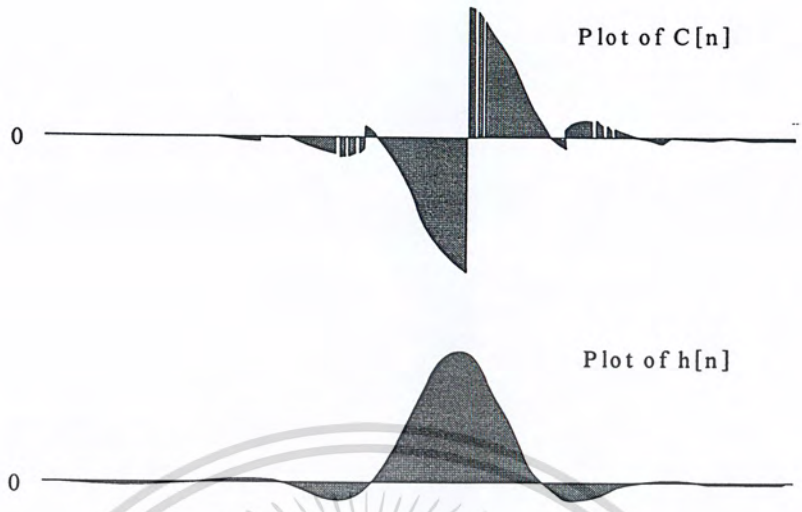
โดย

$x[t]$ คือ สัญญาณเสียงสุ่มความถี่

$$h[n]=\begin{cases} -C[n] & \text{เมื่อ } n/64 \text{ เป็นเลขคี่} \\ C[n] & \text{กรณีอื่น} \end{cases}$$

ในรูปแบบนี้แต่ละย่านความถี่ย่อยจะมีผลตอบสนองความถี่ของตนเอง คือ $H_i[n]$ แม้ว่ารูปแบบนี้จะง่ายต่อการวิเคราะห์แต่จะมีจำนวนครั้งของการคูณ และบวกในการคำนวณมากกว่าสมการที่ (2.20) คือ จะคูณเลข $512\times32=16,384$ ครั้ง และการบวก $512\times32=16,384$ ครั้ง เพื่อที่จะคำนวณสัญญาณเอาต์พุต 32 สัญญาณ

ค่าสัมประสิทธิ์ $h[n]$ เป็นผลตอบสนองตัวกรองความถี่ต่ำสำหรับโพลีเฟส ฟิลเตอร์แบงค์ รูปที่ 2.21 แสดงสัมประสิทธิ์ $C[n]$ และ $h[n]$ สมการสำหรับ $H[n]$ แสดงให้เห็นว่า ค่า $H[n]$ ได้มาจาก $h[n]$ คูณกับเทอมโคไซน์ (Cosine) เพื่อที่จะเลื่อนเฟสของผลตอบสนองตัวกรองความถี่ต่ำ $h[n]$ ให้เหมาะสมกับย่านความถี่ ดังนั้น จึงเรียกการกรองความถี่แบบนี้ว่า “โพลีเฟส” (Polyphase) การกรองความถี่แบบนี้จะมีความถี่กลางอยู่ที่ค่าเลขคี่คูณด้วย $\pi/(64T)$ ซึ่งค่า T นี้เป็นคาบของการสุ่มตัวอย่าง(Sampling Period)

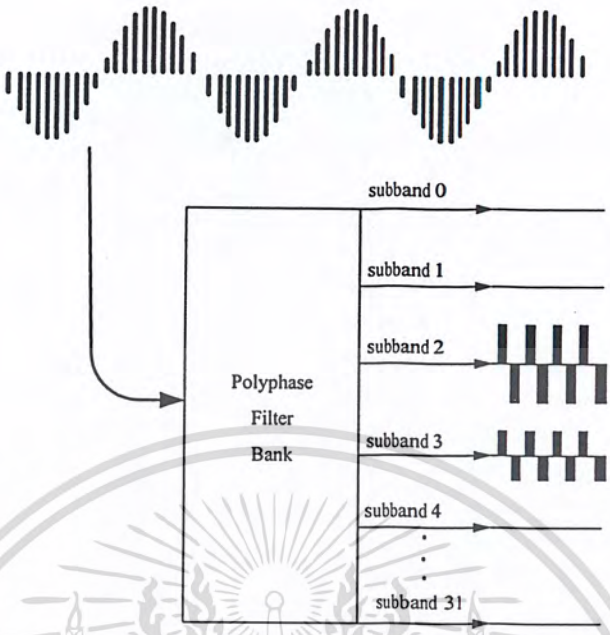


รูปที่ 2.29 การเปรียบเทียบสัมประสิทธิ์ C[n] และ h[n]

เนื่องจากความถี่ต่ำไม่มีความคมในความถี่คัทออฟ (แถบนำกว้าง) ดังนั้น เมื่อแบ่งช่วงความถี่ที่ใช้งานทั้งหมดเป็น 32 ช่วง จะเกิดการเหลื่อมซึ่งกันและกัน ดังแสดงตัวอย่างในรูปที่ 2.30

รูปที่ 2.30 การเหลื่อมซึ่งกันและกันระหว่างย่านความถี่ย่อยที่ติดกัน

ซึ่งผลการเหลื่อมกันของย่านความถี่ที่ติดกัน จะทำให้ประสิทธิภาพของการบีบอัดข้อมูลลดลงเนื่องจากพลังงานของสัญญาณที่มีความถี่ใกล้ๆ กับขอบของย่านความถี่หนึ่งๆ จะปรากฏเป็นเอาต์พุตของ โพลีเฟส ฟิลเตอร์แบงค์ 2 เอาต์พุตที่อยู่ติดกัน ดังรูปที่ 2.31 แสดงตัวอย่างของสัญญาณรูปไซน์หนึ่งความถี่ เมื่อผ่านกระบวนการ โพลีเฟส ฟิลเตอร์แบงค์ แล้วปรากฏเอาต์พุตออกมา 2 ย่านความถี่



รูปที่ 2.31 การเกิดเอาต์พุตของโพลีเฟส ฟิลเตอร์แบงค์
เมื่อสัญญาณเข้าเป็นสัญญาณหนึ่งความถี่

3) ไซโคอะคูสติก โมเดล (Psychoacoustic Model)

การเข้ารหัสข้อมูลเสียงแบบเอ็มเป็ก ใช้หลักการตัดเสียงบางส่วนที่ฟังไม่ได้ยินออก เนื่องจากความได้เปรียบในการรับฟังเสียงของมนุษย์ที่ไม่สามารถได้ยินเสียงรบกวนภายใต้เงื่อนไขการปิดกั้นการได้ยิน (Auditory Masking) การปิดกั้นการได้ยินเป็นคุณสมบัติอย่างหนึ่งในการรับฟังเสียงของมนุษย์ เกิดขึ้นเมื่อเรารับฟังเสียงจากแหล่งกำเนิดสองแหล่งจะไม่สามารถได้ยินเสียงจากแหล่งกำเนิดที่ค่อยกว่าถ้าเครื่องกำเนิดทั้งสองมีความถี่ใกล้เคียงกัน

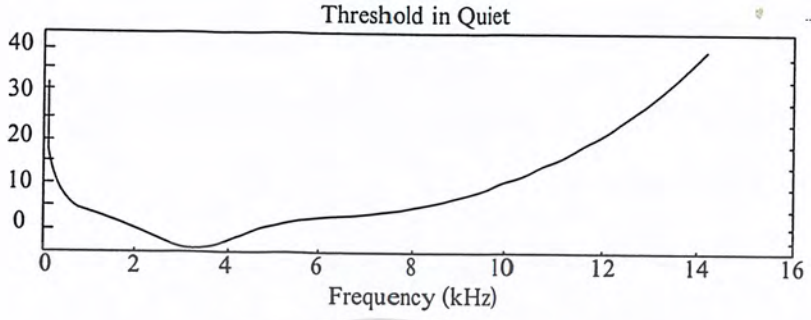
ความสามารถในการได้ยินของมนุษย์มีข้อจำกัดต่างๆ ดังนี้

1) ช่วงความถี่ที่สามารถรับฟังประมาณ 20-20,000 เฮิรตซ์

2) ไดนามิคเรนจ์ (Dynamic Range) คือ เสียงเบาสุดถึงดังสุดที่สามารถรับฟังได้ประมาณ 96 เดซิเบล

3) ความถี่เสียงพูดปกติ 500-2,000 เฮิรตซ์

4) ความไวในการรับฟังเสียงของมนุษย์ จากการทดลองให้คนหนึ่งคนเข้าไปนั่งในห้องที่เงียบสนิทแล้วค่อยๆ เพิ่มความดังเสียง จนกระทั่งคนๆ นั้นเริ่มได้ยิน (Threshold Level) วัดระดับความดังเสียง แปรความถี่แล้วนำค่ามาวาดกราฟ แสดงได้ดังรูปที่ 2.32 จะพบว่ามนุษย์สามารถรับฟังเสียงช่วงความถี่ 2-4 กิโลเฮิรตซ์ได้ไวที่สุด

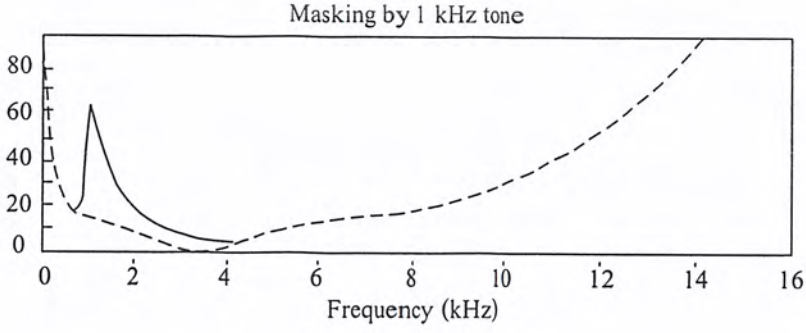


รูปที่ 2.32 ระดับความดังเสียงที่เริ่มได้ยินที่ความถี่ต่างๆ

5) การปิดกั้นเสียงเมื่อสัญญาณเสียงหลายๆ สัญญาณที่มีความถี่ใกล้เคียงกันเข้ามามนุษย์สามารถได้ยินเสียงที่มีความดัง (แรง) กว่าเท่านั้น หรือกล่าวได้ว่าสัญญาณที่แรงกว่าจะปิดกั้นสัญญาณที่อ่อนกว่า และเรียกสัญญาณที่แรงกว่านั้นว่า "ตัวปิดกั้น" (Masker)

จากการทดลองเรื่องการปิดกั้นเสียง โดยให้คนฟังเสียงในห้องที่มีเสียงที่มีความถี่ 1 กิโลเฮิรตซ์ ความดัง 60 เดซิเบล แล้วเล่นเสียงจากแหล่งกำเนิดอีกแหล่ง วัดระดับความดังเสียงที่ได้ยินจากแหล่งกำเนิดที่สอง แปรความถี่ วาดกราฟ ได้ดังรูปที่ 2.33 ซึ่งแสดงให้เห็นว่ามีความถี่ใกล้กับความถี่ 1 กิโลเฮิรตซ์ ระดับความดังเสียงของแหล่งกำเนิดที่ 2 ต้องมีค่าสูงเพื่อให้รับฟังเสียงได้

สามารถสรุปจากข้อมูลทั้งหมดได้ว่า มนุษย์สามารถรับฟังเสียงเป็นช่วงความถี่ และรับพลังงานเสียงได้ไม่เท่ากันในแต่ละย่านความถี่ซึ่งเราเรียกว่าย่านความถี่วิกฤติ (Critical Band) และในแต่ละย่านความถี่มนุษย์จะแยกสัญญาณจากแหล่งกำเนิดต่างๆ ออกจากกันได้ยาก ตารางที่ 2.6 แสดงความกว้างของย่านความถี่วิกฤติย่านต่างๆ



รูปที่ 2.33 ผลของการปิดกั้น

ตารางที่ 2.6 ความกว้างของความถี่วิกฤติย่านต่างๆ

Band Number	Frequency (Hz)	Band Number	Frequency (Hz)
0	50	14	1,970
1	95	15	2,340
2	140	16	2,720
3	235	17	3,280
4	330	18	3,840
5	420	19	4,690
6	560	20	5,440
7	660	21	6,375
8	800	22	7,690
9	940	23	9,375
10	1,125	24	11,625
11	1,265	25	15,375
12	1,500	26	20,250
13	1,735		

เนื่องด้วยเหตุผลที่กล่าวมาทั้งหมดข้างต้น สามารถปิดกั้นเสียงที่ไม่ได้ยินภายในย่านความถี่หนึ่งๆ ได้ ดังรูปที่ 2.34 หลักการนี้จะแบ่งสัญญาณเสียงออกเป็นย่านความถี่ย่อย ประมาณเท่าย่านความถี่วิกฤติแล้วจึงปรับข้อมูลในแต่ละย่านตามความสามารถในการได้ยินในแต่ละย่านความถี่

ไซโคอะคูสติก โมเดล (Psychoacoustic Model) ทำหน้าที่วิเคราะห์หาจุด(ความถี่)ที่ต้องปิดกั้นในย่านความถี่หนึ่ง ตัวเข้ารหัสจะใช้ข้อมูลนี้ในการตัดสินใจว่าควรแทนสัญญาณอินพุตที่เข้ามาด้วยจำนวนบิตมากน้อยเพียงใด

การแบ่งแยกบิต (Bit or Noise Allocation) ทำหน้าที่หาจำนวนของบิตเพื่อที่จะแยกในแต่ละย่านความถี่ย่อยขึ้นอยู่กับข้อมูลที่ได้มาจากไซโคอะคูสติกโมเดล

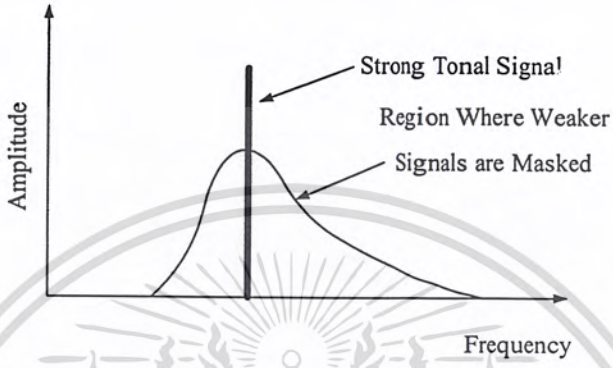
การแบ่งแยกบิตของการเข้ารหัสเอ็มเป็กเลเยอร์ 3 มีรายละเอียดดังนี้

1) ปรับข้อมูลให้เป็นค่า Quantize

2) นับจำนวนบิตของการเข้ารหัสแบบฮัฟแมน (Huffman Coding)

3) คำนวณผลของเสียงที่ตัดออกจริงๆ (Actual Calculates The Result Noise)

4) ทำขั้นตอนที่ 1-3 ซ้ำเพื่อปรับค่าให้อยู่ในระดับที่ยอมรับได้ (Iterative Loop)



รูปที่ 2.34 การปิดกั้นเสียงที่ไม่ได้ยินในย่านความถี่ย่อยหนึ่งๆ ของการเข้ารหัสเอ็มเป็ก

4) การเข้ารูปแบบ (Formatting)

เป็นขั้นตอนการจัดรูปแบบข้อมูลให้ตรงตามมาตรฐาน ซึ่งมาตรฐานการจัดเรียงข้อมูลแบบเอ็มเป็กของเลเยอร์ 3 แสดงได้ดังรูปที่ 2.35

หัวข้อมูล (Header)	ตรวจสอบความผิดพลาด (CRC)	ข้อมูลข้างเคียง (Side Information)	ข้อมูลหลัก (Main Data)

รูปที่ 2.35 รูปแบบข้อมูลเอ็มเป็กเลเยอร์ 3

รายละเอียดของรูปแบบข้อมูลเอ็มเป็กเลเยอร์ 3 มีดังนี้

1) ส่วนหัวข้อมูล (Header) เป็นข้อมูลขนาด 32 บิต แสดงลักษณะทั่วๆ ไปของไฟล์นั้นๆ ข้อมูลในส่วนนี้ประกอบด้วยบิตต่างๆ ดังรูปที่ 2.36 ซึ่งแต่ละบิตจะมีรายละเอียดดังนี้

รายละเอียดของส่วนหัว

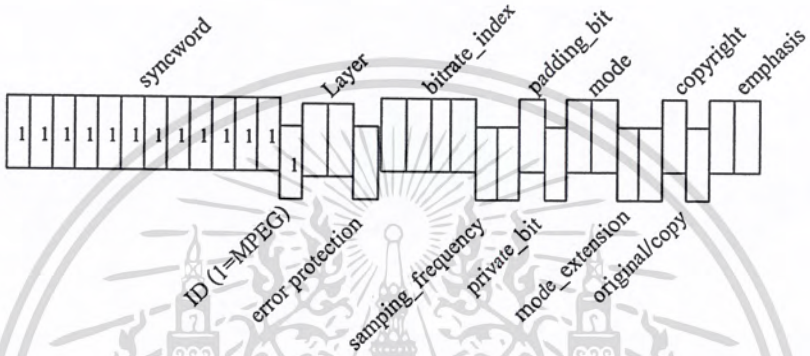
Syncword	ส่วนรหัสของลำดับข้อมูลเอ็มเป็ก เท่ากับ 1111 1111 1111
ID	บิตแสดงถึงมาตรฐานการเข้ารหัส "1" หมายถึง เข้ารหัสตามมาตรฐาน ISO 1172-3, "0" ไม่ใช่มาตรฐาน ISO

Layer แสดงเลเยอร์ที่กำลังถอดรหัสอยู่ดังตารางที่ 2.7

Protect_bit บอกให้ทราบว่า ข้อมูลเสริมเกี่ยวกับการตรวจสอบความผิดพลาดได้บันทึกมาด้วยหรือไม่

"1" ไม่มีข้อมูลเสริม

"0" มีข้อมูลเสริม



รูปที่ 2.36 บิตต่างๆ ในส่วนหัวข้อมูล

ตารางที่ 2.7 ความหมายของรหัสข้อมูลในเลเยอร์

เลเยอร์	ความหมาย
11	เลเยอร์ 1
10	เลเยอร์ 2
01	เลเยอร์ 3
00	ไม่ใช้

Bitrate_index บอกอัตราการบีบอัดข้อมูล (Bitrate) ที่ใช้

Sampling_frequency บอกความถี่ในการสุ่มตัวอย่าง (Sampling)

Padding_bit เท่ากับ "1" แสดงว่าเฟรมนั้นๆ บรรจุสล็อต (Slot) ในเลเยอร์3=1 ไบต์เพิ่มเติมมา ถ้าเท่ากับ "0" แสดงว่าไม่ได้บรรจุ การบบรจุสล็อตเพื่อปรับอัตราส่วนระหว่างอัตราส่วนการบีบอัดกับความถี่ในการสุ่มตัวอย่าง Sampling Frequency ให้ลงตัว

Private ไม่ใช้ในการเข้ารหัสตามมาตรฐาน ISO/IEC 11172-3

Copy right "1" ป้องกันการ copy, "0" ไม่ป้องกันการ copy

Originl/copy "0" ข้อมูลนั้นถูก copy มา, "1" ข้อมูลนั้นเป็นต้นฉบับ

Mode บอกเกี่ยวกับรูปแบบของข้อมูลว่าเป็นหนึ่งช่องเสียง (Single Channel) สองช่องเสียง (Double Channel) สเตอริโอ (Stereo) และจอยท์-สเตอริโอ (Joint Stereo) แสดงดังตารางที่ 2.8

ตารางที่ 2.8 ความหมายของรหัสข้อมูลใน Mode

Mode	ความหมาย
00	สเตอริโอ
01	จอยท์-สเตอริโอ
10	สองช่องเสียง
11	หนึ่งช่องเสียง

Mode_extention ใช้บอกชนิดของจอยท์-สเตอริโอ (Joint Stereo Mode) ว่ามี Ms-stereo, Intensity-stereo หรือไม่อย่างไร ดังตารางที่ 2.9

ตารางที่ 2.9 ความหมายของรหัสข้อมูลใน Mode_extention

Mode_extention	Intensity	Ms_stereo
00	OFF	OFF
01	ON	OFF
10	OFF	ON
11	ON	ON

Emphasis แสดงถึงชนิดเอ็มฟาไซส์ (Emphasis) ที่ใช้งาน แสดงในตารางที่ 2.10

ตารางที่ 2.10 ชนิดเอ็มฟาไซส์

Emphasis	ความหมาย
00	ไม่มีใช้
01	50/15 ไมโครวินาที
10	สงวน
11	CCITT J.17

2) ตรวจสอบความผิดพลาดเป็นข้อมูลขนาด 16 บิต ตรวจสอบพาริตี้ข้อมูลที่ถูกเข้ารหัส

3) ส่วนข้อมูลข้างเคียงเป็นข้อมูลขนาด 17 หรือ 32 ไบต์ข้อมูลในส่วนนี้จะเก็บองค์ประกอบต่างๆ ที่ใช้ในการถอดรหัสโดยตรง

4) ข้อมูลหลักจะไม่ถูกจำกัดความยาวข้อมูล ขึ้นอยู่กับความถี่สุ่มตัวอย่าง และอัตราการส่งข้อมูล ดังสมการ

$$N=144,000\times\frac{\text{Bit rate}}{\text{Sampling Frequency}} \qquad (2.23)$$

เมื่อ N คือ ความยาวข้อมูลระหว่าง Syncword ที่อยู่ติดกัน

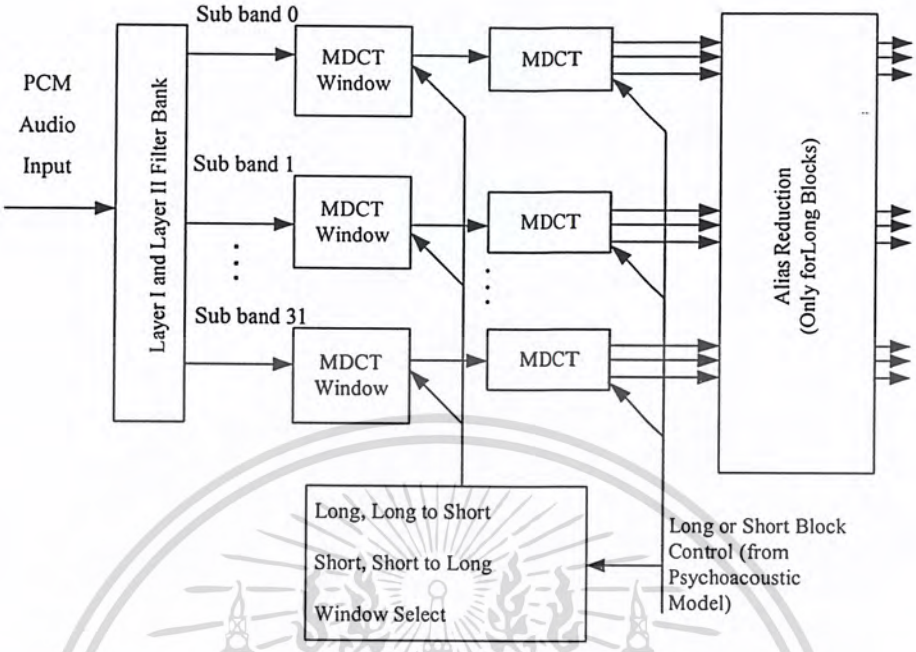
Bitrate คือ อัตราการส่งข้อมูล

Sampling Frequency คือ ความถี่สุ่มตัวอย่าง

2.2.6 การเข้ารหัสข้อมูลเอ็มเป็กเลเยอร์ 3

การเข้ารหัสข้อมูลเอ็มเป็กเลเยอร์ 3 มีหลักการเหมือนการเข้ารหัสแบบเอ็มเป็กโดยทั่วไปดังกล่าวมาแล้วในหัวข้อ 2.1.3 มีเพียงบางเทคนิคที่เพิ่มเข้ามาเพื่อให้การบีบอัดข้อมูลมีประสิทธิภาพสูงขึ้น ดังนี้

1) การใช้การแปลงแบบ MDCT (Modified Discrete Cosine Transform) เพื่อเพิ่มความละเอียดของสเปกตรัม หลังจากที่ผ่านขั้นตอนการทำโพลีเฟส ฟิลเตอร์แบงค์ แสดงแผนผังการทำ MDCT ดังรูปที่ 2.30



รูปที่ 2.37 แผนผังการทำ IMDCT

มีรูปแบบการทำ MDCT อยู่ 2 ลักษณะ โดยแบ่งตามความยาวของบล็อก คือ

1.1) บล็อกยาวมี 18 สัญญาณสุ่มตัวอย่างแต่มีการเหลื่อมของย่านความถี่อยู่ร้อยละ 50 ดังนั้น บล็อกจึงมีความยาว 36 สัญญาณสุ่มตัวอย่าง

1.2) บล็อกสั้นมี 6 สัญญาณสุ่มตัวอย่าง แต่มีการเหลื่อมของย่านความถี่อยู่ที่ร้อยละ 50 ดังนั้น บล็อกสั้นจึงมีความยาว 12 สัญญาณสุ่มตัวอย่าง

ความแตกต่างของบล็อกยาว และบล็อกสั้น คือ บล็อกยาวให้ความละเอียดในด้านความถี่มาก ส่วนบล็อกสั้นให้รายละเอียดด้านเวลาสำหรับสัญญาณชั่วขณะ (Transient) สังเกตุว่าความยาวของบล็อกสั้นเท่ากับหนึ่งส่วนสามของบล็อกยาว ดังนั้น ถ้าใช้บล็อกสั้นต้องใช้สามบล็อกเพื่อให้ความยาวข้อมูลเท่ากับเมื่อใช้บล็อกยาว

ในการเข้ารหัสเฟรม (หนึ่งเฟรม คือ 1,152 สัญญาณสุ่มตัวอย่างอินพุต) หนึ่งๆ สามารถใช้บล็อกชนิดเดียวกันหรือบล็อกผสม (Mixed Block) ได้ ถ้าเข้ารหัสแบบผสม 2 ย่านความถี่ต่ำจะต้องใช้บล็อกยาว อีก 30 ย่านความถี่ที่เหลือเป็นบล็อกสั้น ซึ่งถ้าการเข้ารหัสแบบบล็อกผสมจะให้ความละเอียดด้านความถี่สูงที่ความถี่ต่ำ และความละเอียดด้านเวลาสูงที่ความถี่สูง

2) การลดผลจากการเหลื่อมของย่านความถี่ติดกัน (Alias Reduction)

3) การปรับค่าแบบไม่เป็นรูปแบบ (Non-uniform Quantization)

4) ย่านสเกลแฟคเตอร์ (Scalefactor Band) ใช้สเกลแฟคเตอร์ต่างกันเมื่อต่างย่านความถี่

5) การเข้ารหัสข้อมูลแบบเอ็นทรอปี้ เอ็มเป็กเลเยอร์ 3 ใช้การเข้ารหัสข้อมูลแบบฮัฟแมน (Huffman) มาช่วยในการบีบอัดข้อมูล

2.2.7 การเข้ารหัสและถอดรหัสฮัฟแมน

1) การเข้ารหัสฮัฟแมน

การเข้ารหัสแบบฮัฟแมน (Huffman Coding) จะใช้ทฤษฎีความน่าจะเป็นในการเข้ารหัส ความน่าจะเป็นที่กล่าวถึงนี้ คือ ความซ้ำซ้อนของข้อมูลที่จะเกิดขึ้นในข้อมูลชุดใหญ่ๆ ชุดหนึ่ง แทนที่เราจะแทนรหัสข้อมูลที่มีความซ้ำซ้อนกันมากๆ แทนที่จะเก็บด้วยข้อมูลบิตปกติ เราจะทำการเข้ารหัสฮัฟแมนด้วยข้อมูลบิตน้อยลง หรือเพิ่มขึ้นตามความน่าจะเป็นของข้อมูลที่จะเกิดขึ้นว่ามีสูงขนาดใด ถ้าความน่าจะเป็นเกิดขึ้นมากแทนด้วยข้อมูลบิตน้อยๆ และถ้าความน่าจะเป็นเกิดขึ้นน้อยแทนด้วยรหัสที่มีจำนวนบิตมากขึ้นตามหลักการของฮัฟแมนซึ่งจะกล่าวถึงโดยละเอียดต่อไป

ตัวอย่างของการเข้ารหัสฮัฟแมน มีข้อมูลอยู่ชุดหนึ่งมีความน่าจะเป็นที่เกิดขึ้นตามตารางที่ 2.11

การเข้ารหัสฮัฟแมน สามารถกระทำได้ดังนี้ คือ

1) เริ่มจับคู่จากข้อมูลที่มีความน่าจะเป็นน้อยที่สุดไปยังมากที่สุด

2) เอาข้อมูลที่มีความน่าจะเป็นน้อยไว้ทางขวามือ

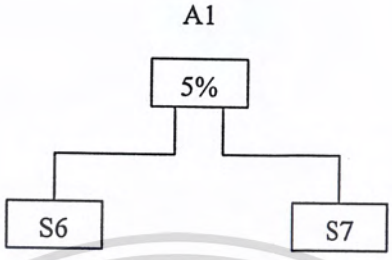
3) เมื่อทำการจับคู่ข้อมูลแล้วจะได้ความน่าจะเป็นของข้อมูลเพิ่มขึ้น ให้ทำการเลื่อนความน่าจะเป็นเพิ่มขึ้นไปเพื่อจับคู่กับข้อมูลตัวที่มีความน่าจะเป็นเรียงลำดับกัน

ตารางที่ 2.11 ข้อมูลที่ต้องการเข้ารหัสฮัฟแมน

Symbols	Probability
S0	40%
S1	24%
S2	10%
S3	9%
S4	7%
S5	5%
S6	3%
S7	2%
Total	100%

4) ทำการจับคู่ชุดข้อมูลจนกว่าจะมีความน่าจะเป็น 100%

การเขียน Binary Tree เพื่อเข้ารหัสแบบฮัฟแมน



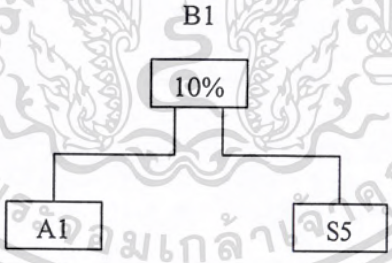
รูปที่ 2.38 การจับคู่ข้อมูลที่มีความน่าจะเป็นระหว่าง S6 และ S7

A1 เป็นรหัสเพื่อให้การจับคู่ดูง่ายในการที่จะไปจับคู่กับตัวถัดไป

5% เป็นความน่าจะเป็นใหม่ที่เกิดขึ้นจากการรวมความน่าจะเป็นของ S6 และ S7 คือ 3% และ 2%

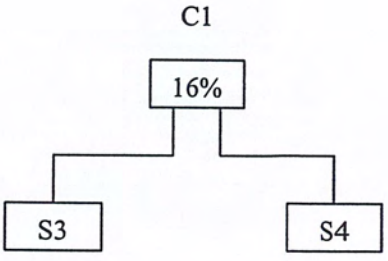
S7 อยู่ทางขวามือเพราะความน่าจะเป็นต่ำกว่า S6

เมื่อทำการจับคู่ระหว่าง S6 และ S7 จะได้ความน่าจะเป็นเพิ่มขึ้นแล้วนำไปจับกับ S5



รูปที่ 2.39 การจับคู่ข้อมูลที่มีความน่าจะเป็นระหว่าง A1 และ S5

B1 เกิดจากการรวมของ S5 และ A1 ซึ่งทั้งสองรวมกันได้ความน่าจะเป็น 10% จะได้ความน่าจะเป็นไปอยู่ต้นๆ จากนั้นทำการจับคู่ S3, S4 ซึ่งมีความน่าจะเป็น 9% และ 7%



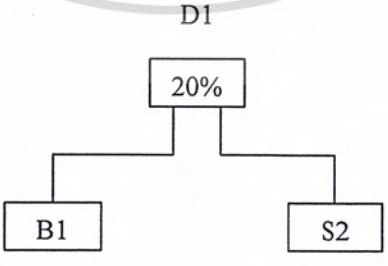
รูปที่ 2.40 การจับคู่ข้อมูลที่มีความน่าจะเป็นระหว่าง S3 และ S4

จะได้ตารางใหม่ดังนี้

ตารางที่ 2.12 ผลการจับคู่ความน่าจะเป็น

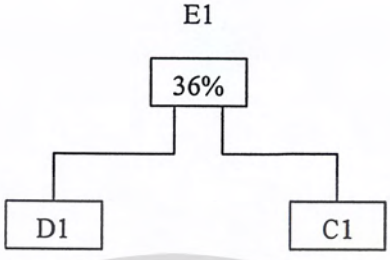
Symbols	Probability
S0	40%
S1	24%
C1	16%
B1	10%
S2	10%

ทำการจับคู่ลำดับความน่าจะเป็นระหว่าง S2 และ B1 โดยเรา S2 ไว้ทางขวามือเพราะความน่าจะเป็นน้อยกว่า



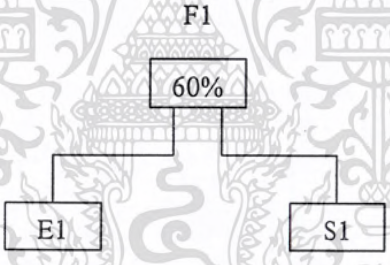
รูปที่ 2.41 การจับคู่ข้อมูลที่มีความน่าจะเป็นระหว่าง B1 และ S2

จากนั้นทำการจับคู่ตามลำดับความน่าจะเป็นระหว่าง C1 และ D1 โดยเอา C1 ไว้ทางขวามือเพราะความน่าจะเป็นน้อยกว่า



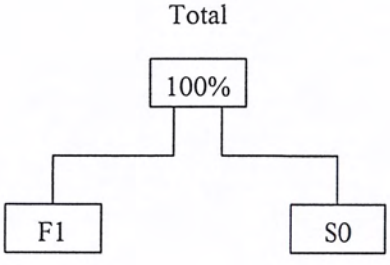
รูปที่ 2.42 การจับคู่ข้อมูลที่มีความน่าจะเป็นระหว่าง D1 และ C1

จากนั้นทำการจับคู่ตามลำดับความน่าจะเป็นระหว่าง S1 และ E1 โดยเอา S1 ไว้ทางขวามือเพราะความน่าจะเป็นน้อยกว่า



รูปที่ 2.43 การจับคู่ข้อมูลที่มีความน่าจะเป็นระหว่าง E1 และ S1

ลำดับสุดท้ายในการจับคู่ คือ S0 และ F1 โดยเอา S0 ไว้ทางขวามือเพราะความน่าจะเป็นน้อยกว่า เป็นการจับคู่อันดับสุดท้ายเพราะทั้งคู่รวมความน่าจะเป็นได้ 100%



รูปที่ 2.44 การจับคู่ข้อมูลที่มีความน่าจะเป็นระหว่าง F1 และ S0

จากนั้นนำการจับคู่อันดับสุดท้ายมาขยายย่อยเพื่อทำการแทนรหัส

F1 คือ S1,E1

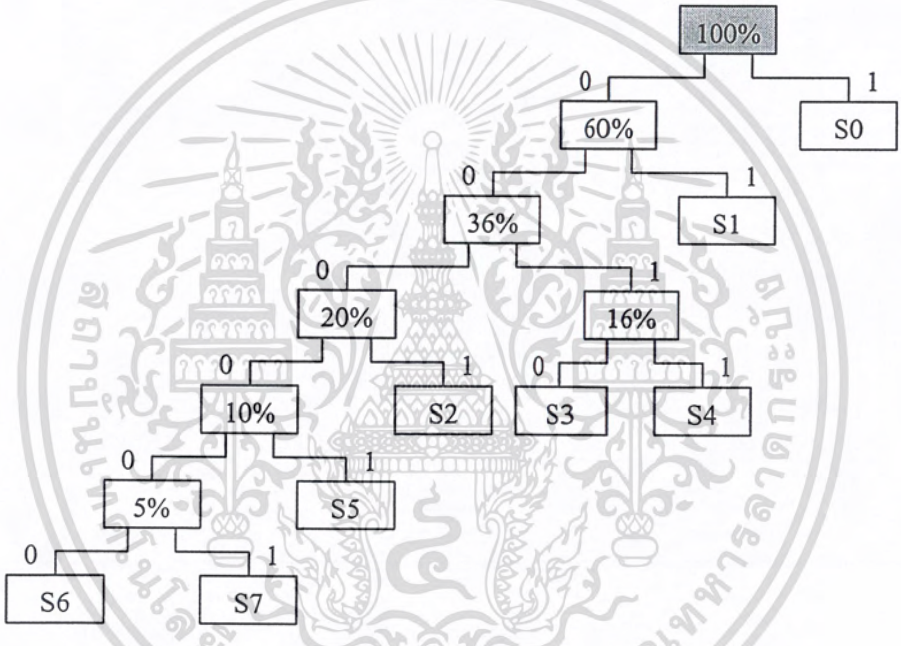
E1 คือ C1,D1

C1 คือ S4,S3

D1 คือ S2,B1

B1 คือ S5,A1

A1 คือ S7,S6



รูปที่ 2.45 ผลการนำการจับคู่ครั้งสุดท้ายมาขยายย่อย

การแทนรหัสฮัฟแมน

จากฮัฟแมนทรี (Huffman Tree) สามารถแทนรหัสฮัฟแมนได้ดังนี้ คือ

ตารางที่ 2.13 การแทนข้อมูลด้วยรหัสฮัฟแมน

Symbols	Probability	Huffman Codes
S0	40%	1
S1	24%	01
S2	10%	0001
S3	9%	0010
S4	7%	0011
S5	5%	00001
S6	3%	000000
S7	2%	000001
Total	100%	

2) การถอดรหัสฮัฟแมน (Hufaman Decode)

สำหรับการถอดรหัสฮัฟแมนที่เก็บไว้ในหน่วยความจำ ซึ่งรหัสฮัฟแมนจะเรียงกันอยู่เป็นอนุกรม การถอดรหัสต้องดึงข้อมูลออกมาแล้วเช็คข้อมูลเป็นระดับบิตเพื่อนำไปชี้ว่ารหัสดังกล่าวเป็นรหัสของข้อมูลใด

ตัวอย่าง การถอดรหัสฮัฟแมนต่อไปนี้ 0000001 10000110001

000000 เป็นรหัสของ S6

จะเหลือข้อมูล 110000110001

1 เป็นรหัสของ S0

จะเหลือข้อมูล 10000110001

1 เป็นรหัสของ S0

จะเหลือข้อมูล 0000110001

00001 เป็นรหัสของ S5

จะเหลือข้อมูล 10001

1 เป็นรหัสของ S0

จะเหลือข้อมูล 0001

0001 เป็นรหัสของ S2

ดังนั้น ข้อมูลที่อนุกรมกันมา คือ S6, S0, S0, S5, S0, S2

2.3 การควอนไตส์เวคเตอร์แบบมาตรฐาน

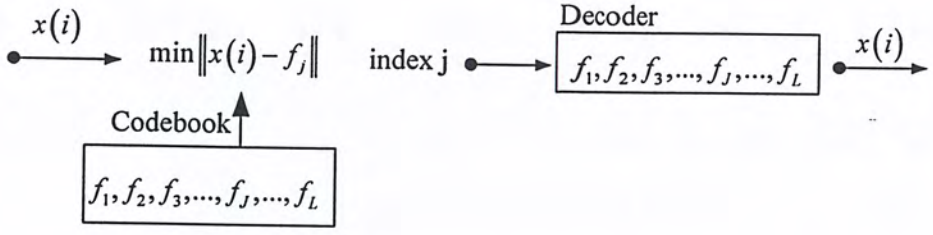
การควอนไตส์เวคเตอร์แบบมาตรฐาน เป็นการควอนไตส์เวคเตอร์แบบแรกสุด ที่ได้มีการนำไป ประยุกต์ใช้งานในการเข้ารหัสสัญญาณเสียง และเป็นพื้นฐาน ในการพัฒนาการควอนไตส์เวคเตอร์ในเวลาต่อมา ซึ่งวิธีการควอนไตส์เวคเตอร์แบบมาตรฐานนี้ได้นำเสนอเป็นครั้งแรก โดย Yoseph Linde, Andres Buzo และ Robert M. Gray โดยทั้ง 3 คน นำเสนอวิธีการ ออกแบบโค๊ดบุ้ค คือ วิธีการแบบ LBG Algorithms หรือในบางครั้งเราเรียกว่าวิธีการแบบ Spliting Algorithms ซึ่งได้พัฒนามาจากวิธีการของ Lloyd Algorithms เสนอการควอนไตส์เวคเตอร์แบบมาตรฐาน โดยจะกล่าวถึง หลักการพื้นฐานของการควอนไตส์เวคเตอร์ และการออกแบบโค๊ดบุ้คสำหรับการควอนไตส์เวคเตอร์แบบมาตรฐาน

2.3.1 หลักการควอนไตส์เวคเตอร์แบบมาตรฐาน

จากรูปที่ 2.46(ก) เป็นหลักการพื้นฐานของการควอนไตส์เวคเตอร์แบบมาตรฐาน วงจรเครื่องส่ง (Transmitter) และถูกจัดเก็บในหน่วยความจำในลักษณะของอะเรย์ 2 มิติ ซึ่งแสดงได้ดังรูปที่ 2.47 โดยโค๊ดบุ้คเป็นส่วนที่มีความสำคัญ เนื่องจากจะต้องเป็นที่เก็บคุณลักษณะของเสียงทั้งหมด และจากรูปที่ 2.46(ข) ในขบวนการเริ่มต้นสัญญาณอินพุตเวคเตอร์ $x(i)$ จะถูกสุ่มเข้ามาจำนวนรหัสจะแทนสัญญาณเวคเตอร์ ตัวอย่าง (เท่ากับมิติของโค๊ดบุ้ค หรือหนึ่งเวคเตอร์) โดยวงจรเข้ารหัส $x(i)$ ด้วยค่าเวคเตอร์ f_j จากโค๊ดบุ้คซึ่งจะมีเพียงหนึ่งเวคเตอร์ที่มีคุณลักษณะใกล้เคียงกับสัญญาณอินพุตเวคเตอร์ $x(i)$ มากที่สุด โดยการหาค่าต่ำสุด (Minimize The Euclidean Distance) ดังนี้

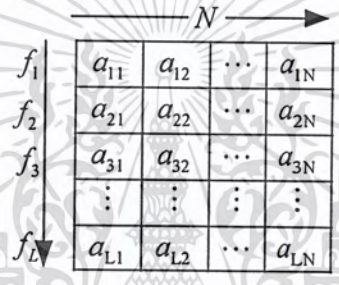
$$min\left\|x(i) - f_j\right\| \tag{2.24}$$

โดยที่ $j = 1, 2, 3, ..., L$



(ก) วงจรเครื่องส่ง (ข) วงจรเครื่องรับ

รูปที่ 2.46 การควอนไตส์เวคเตอร์แบบมาตรฐาน



รูปที่ 2.47 การจัดเก็บโค๊ดบุ้คในหน่วยความจำ

จะได้ค่าดัชนี (Index) ของเวคเตอร์จากโค๊ดบุ้ค คือ j เพื่อทำการส่งไปที่วงจรเครื่องรับ ในส่วนของวงจรเครื่องรับจะนำเอาค่าดัชนี j มาทำการถอดรหัสจากโค๊ดบุ้คที่เครื่องรับ โดยที่โค๊ดบุ้คที่เครื่องส่ง และเครื่องรับจะต้องมีข้อมูลที่เหมือนกัน หลังจากนั้นจะได้สัญญาณเอาต์พุตเวคเตอร์ที่ถูกสร้างขึ้นใหม่ คือ เวคเตอร์ $\hat{x}(i)$ ซึ่งมีค่าเท่ากับเวคเตอร์ f_j ที่เก็บอยู่ในโค๊ดบุ้คนั่นเอง โดยที่เวคเตอร์ f_j จะต้องมีค่าใกล้เคียงกับอินพุตเวคเตอร์ $x(i)$ มากที่สุด

สิ่งสำคัญอีกประการหนึ่งของการควอนไตส์เวคเตอร์ คือ ความละเอียดของจำนวนบิตที่ใช้ในหนึ่งอีลีเมนท์ของเวคเตอร์ (R) หรืออาจกล่าวได้ว่าในหนึ่งตัวอย่างที่ได้จากการสุ่มสัญญาณจากเครื่องส่งต้องการจำนวนบิตของข้อมูลเท่าไรเพื่อที่จะส่งไปที่เครื่องรับ และสามารถคำนวณได้ดังนี้

$$R = \frac{\log_2 L}{N} \text{ (bit/element)} \tag{2.25}$$

สมมุติถ้ากำหนดให้โค๊ดบุ้คมีมิติ $N=5$ และจำนวนเวคเตอร์ในโค๊ดบุ้ค $L=256$ จะได้

$$R=\frac{\log 2L}{N}=\frac{8}{5}=1.6 \text{ (bit/element)} \tag{2.26}$$

นั่นหมายความว่าในหนึ่งตัวอย่างของการสุ่มสัญญาณที่ส่งไปยังเครื่องรับต้องการข้อมูลเพียง 1.6 บิต ถ้าหากเรานำหลักการควอนไตส์เวคเตอร์ไปประยุกต์ใช้ในการเข้ารหัสสัญญาณเสียง และจาก สมการที่ 2.22 สามารถคำนวณหาค่าบิตเรทในการส่งข้อมูล หรือจำนวนบิตที่ต้องส่งจากเครื่องส่งไปที่เครื่องรับภายในเวลา 1 วินาที ซึ่งคำนวณหาได้ดังนี้

$$\text{Bit rate}=R\times F_s \quad \text{(bit/sec)}$$

$$F_s = \text{ความถี่ในการสุ่มสัญญาณ } (\text{H}_z) \tag{2.27}$$

ถ้าอัตราการสุ่มสัญญาณ $F_s=8kHz$ มิติของโค๊ตบุ้ค $L=256$ (หมายถึงการเข้ารหัสในแต่ละครั้งใช้สัญญาณจำนวน 5 ตัวอย่าง) และจำนวนของโค๊ดบุ้คเวคเตอร์ $L=256$ นั่นหมายความว่าค่าดัชนีเวคเตอร์ j ที่ส่งไปยังเครื่องรับเท่ากับ $\log_2 256=8\log_2$ บิต สามารถสร้างสัญญาณเอาต์พุตเวคเตอร์ $x(i)$ ได้จำนวน 5 ตัวอย่าง ดังนั้นจากสมการที่ จำนวนบิตเรตในการส่งข้อมูลจะได้เป็น

$$Bit\ rate=1.6\times 8{,}000=12{,}800 \text{ bps}$$

เพื่อแสดงให้เห็น ถึงหลักการเข้ารหัส และการลดบิตเรต ในการส่งข้อมูลให้ชัดเจนยิ่งขึ้น จึงได้ยกตัวอย่างหลักการควอนไตส์เวคเตอร์แบบมาตรฐาน โดยกำหนดให้โค๊ตบุ้คมีมิติ $N=2$ และ จำนวนเวคเตอร์ในโค๊ตบุ้ค $L=4$ ดังนี้

$$f_1=\begin{bmatrix}3\\5\end{bmatrix},\ f_2=\begin{bmatrix}4\\9\end{bmatrix},f_3=\begin{bmatrix}7\\-5\end{bmatrix},f_4=\begin{bmatrix}0\\1\end{bmatrix}$$

สัญญาณอินพุตเวคเตอร์ที่ถูกสุ่มเข้ามาจำนวน 2 ตัวอย่าง (เท่ากับมิติของโค๊ตบุ้ค)

$$x(i) = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$$

ความถี่ในการสุ่มสัญญาณ Fs = 800 Hz

บิตเรตในการส่งข้อมูลจะได้ดังนี้

$$Bit\ rate = \frac{\log_2 L}{N} \times F_s = \frac{\log_2 4}{2} \times 8{,}000 = 8{,}000 \text{ bps}$$

คำนวณหาสัญญาณเอาต์พุตเวคเตอร์ $x(i)$ เพื่อแสดงให้เห็นถึงหลักการเข้ารหัส และการลดบิตเรตในการส่งข้อมูลให้ชัดเจนยิ่งขึ้น จำนวนของเวคเตอร์ในโค๊ตบุ้ค $L = 4$ ดังนี้

$$f_1 = \begin{bmatrix} 3 \\ 5 \end{bmatrix}, f_1 = \begin{bmatrix} 4 \\ 9 \end{bmatrix}, f_1 = \begin{bmatrix} 7 \\ -5 \end{bmatrix}, f_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

สัญญาณอินพุต เวคเตอร์ที่ถูกสุ่มเข้ามาจำนวน 2 ตัวอย่าง (เท่ากับมิติของโค๊ตบุ้ค)

$$\mathrm{x(i)} = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$$

ความถี่ในการสุ่มสัญญาณ $F_s = 8{,}000\,Hz$

บิตเรตในการส่งข้อมูลจะได้ดังนี้

$$Bit\ rate = \frac{\log_2 L}{N} \times F_s = \frac{\log_2 4}{2} \times 8{,}000 = 8{,}000$$

คำนวณหาสัญญาณเอาต์พุตเวคเตอร์ $x(i)$ ได้จาก $\min \left\| x(i) - f_j \right\|; j = 1, 2, 3, ..., L$

$$\|x(i)-f_1\|=\left\|\begin{bmatrix}3\\4\end{bmatrix}-\begin{bmatrix}3\\5\end{bmatrix}\right\|=\left\|\begin{bmatrix}0\\-1\end{bmatrix}\right\|=\sqrt{\begin{bmatrix}0 & -1\end{bmatrix}\begin{bmatrix}0\\-1\end{bmatrix}}=1$$

$$\|x(i)-f_2\|=\left\|\begin{bmatrix}3\\4\end{bmatrix}-\begin{bmatrix}4\\9\end{bmatrix}\right\|=\left\|\begin{bmatrix}-1\\-5\end{bmatrix}\right\|=\sqrt{\begin{bmatrix}-1 & -5\end{bmatrix}\begin{bmatrix}-1\\-5\end{bmatrix}}=5.09$$

$$\|x(i)-f_3\|=\left\|\begin{bmatrix}3\\4\end{bmatrix}-\begin{bmatrix}7\\-5\end{bmatrix}\right\|=\left\|\begin{bmatrix}-4\\9\end{bmatrix}\right\|=\sqrt{\begin{bmatrix}-4 & -9\end{bmatrix}\begin{bmatrix}-4\\9\end{bmatrix}}=9.84$$

$$\|x(i)-f_4\|=\left\|\begin{bmatrix}3\\4\end{bmatrix}-\begin{bmatrix}0\\1\end{bmatrix}\right\|=\left\|\begin{bmatrix}3\\3\end{bmatrix}\right\|=\sqrt{\begin{bmatrix}3 & -3\end{bmatrix}\begin{bmatrix}3\\3\end{bmatrix}}=4.24$$

เมื่อพิจารณาผลของการเปรียบเทียบสัญญาณอินพุตเวคเตอร์ $x(i)$ เทียบกับโค๊ตบุ้คทั้ง 4 เวคเตอร์ $\min\|x(i)-f_j\|$ จะเห็นได้ว่าผลของการเปรียบเทียบนั้น สัญญาณอินพุตเวคเตอร์ $x(i)$ มีค่า ใกล้เคียงกับเวคเตอร์ f_1 มากที่สุด (ผลจากการลบระหว่าง $x(i)$ และ f_1 มีค่าน้อยที่สุด) ดังนั้น ข้อมูลที่ส่งไปที่เครื่องรับ คือ 00 (Binary) สัญญาณเอาต์พุตที่เครื่องรับจะมีค่าเป็น $x(i)=\begin{bmatrix}3\\4\end{bmatrix}$

2.3.2 การออกแบบโค๊ตบุ้ค

คุณภาพของสัญญาณเสียงที่เอาต์พุตจะขึ้นอยู่กับโค๊ตบุ้คดังนั้นขั้นตอนการออกแบบโค๊ตบุ้ค จึงเป็นขั้นตอนที่มีความสำคัญเป็นอย่างยิ่งที่จะต้องเก็บคุณลักษณะของสัญญาณเสียงทั้งหมด ในขั้นตอนการออกแบบโค๊ตบุ้คจะนำเอาสัญญาณเสียงต้นฉบับ หรือสัญญาณที่ใช้ในการออกแบบโค๊ตบุ้ค $\{x(i)\}$ (Training Vector)มาทำการแยกกลุ่ม (Clustering) เพื่อหาคุณลักษณะของสัญญาณเสียงที่เหมาะสมที่สุด และให้มีความผิดพลาด (Distortion) น้อยที่สุด โดยใช้วิธีการของ LBG ซึ่งได้พิจารณามาจากวิธีการของ Lloyd มาประยุกต์ใช้ในการออกแบบ และเขียนเป็นผังการทำงานของการออก แบบได้ดังแสดงในรูปที่ 2.48 และส่วนที่มีความสำคัญสำหรับการออกแบบโค๊ตบุ้คนั้น คือ การหาค่า Centroid ของโค๊ตบุ้คเวคเตอร์ f_j โดยการพิจารณาจากค่าความเพี้ยนระหว่างสัญญาณอินพุต และเอาต์พุต $D(i)$ (Local Distrotion) จะได้

$$D(i)=\left\|x(i)-\overline{x}(i)\right\|^2 \tag{2.28}$$

จากหลักการควอนไตส์เวคเตอร์แบบมาตรฐานที่ได้กล่าวมาแล้วสัญญาณเอาต์พุตเวคเตอร์ $\overline{x}(i)$ จะมีค่าเท่ากับโค๊ตบุ้คเวคเตอร์ f_j ที่มีค่าใกล้เคียงกับสัญญาณอินพุตเวคเตอร์ $x(i)$ มากที่สุด ดังนั้นค่าของความเพี้ยน $D(i)$ จะได้ดังนี้

$$D(i) = \|x(i) - f_j\|^2 \tag{2.29}$$

ค่าความเพี้ยนรวมในแต่ละ Partition ของโค๊ตบุ้คซึ่งเขียนแทนด้วย D_j (Partition Distortion) เป็นค่าความเพี้ยนรวมที่เกิดจากจากการจัดกลุ่มของเวคเตอร์ $x(i)$ ให้เข้ากลุ่มของ $P(f_j)\{x(i) \in P(f_j)\}$ เมื่อ $P(f_j)$ คือ Voronoi cell หรือกลุ่มของเวคเตอร์ที่มีสมาชิก คือ เวคเตอร์โดยมีเวคเตอร์ เป็นค่า Centriod สามารถหาได้ดังนี้

$$\begin{aligned} D_j &= \sum_{\substack{t \\ x(i) \in P(f_j)}} D(i) \\ &= \sum_{\substack{t \\ x(i) \in P(f_j)}} \left\| x(i) - f_j \right\|^2 \\ &= \sum_{\substack{t \\ x(i) \in P(f_j)}} \left[\left[x(i) - f_j \right]^t \left[x(i) - f_j \right] \right] \\ &= \sum_{\substack{t \\ x(i) \in P(f_j)}} \left[x^t(i)x(i) - 2f_j^t x(i) + f_j^t f_j \right] \end{aligned} \tag{2.30}$$

เมื่อนำค่าความเพี้ยนรวมในแต่ละ Partition (D_j) มาทำการอนุพันธ์ $\frac{\partial D_j}{\partial f_j}$ เพื่อหาโค๊ตบุ้คเวคเตอร์ f_j ที่เกิดขึ้นใหม่ในแต่ละ Partition และจากสมการที่ (2.30) จะได้

$$\sum_{\substack{t \\ x(i) \in P(f_j)}} \left[\frac{\partial x^t(i)x(i)}{\partial f_j} \right] - \sum_{\substack{t \\ x(i) \in P(f_j)}} \left[\frac{\partial 2f_j^t x(i)}{\partial f_j} \right] + \sum_{\substack{t \\ x(i) \in P(f_j)}} \left[\frac{\partial f_j^t f_j}{\partial f_j} \right] = 0$$

$$2 \sum_{\substack{t \\ x(i) \in P(f_j)}} x(i) + 2 \sum_{\substack{t \\ x(i) \in P(f_j)}} f_j^{new} = 0 \qquad (2.31)$$

จากสมการที่ (2.31) จะได้ค่า Centroid ของโค๊ดบุ๊คเวคเตอร์ที่เกิดขึ้นใหม่ในแต่ละ Partition หรือในแต่ละ P (f_j) ดังนี้

$$f_j^{new} = \frac{1}{2 \sum_{\substack{t \\ x(i) \in P(f_j)}} 1} 2 \sum_{\substack{t \\ x(i) \in P(f_j)}} x(i)$$

$$f_j^{new} = \frac{1}{m_j} \sum_{\substack{t \\ x(i) \in P(f_j)}} x(i) \qquad (2.32)$$

ซึ่ง M_j คือ จำนวนของเวคเตอร์ x(i) ที่ถูกจัดเข้ากลุ่มของ $P(f_j)$ และจากที่กล่าวมา เป็นการหาค่า Centroid ของโค๊ดบุ๊คที่เกิดขึ้นในแต่ละ Parition โดยหลักการออกแบบโค๊ดบุ๊คทั้งหมดสามารถเขียนเป็นขั้นตอนได้ดังต่อไปนี้

กำหนดให้โค๊ดบุ๊คที่ต้องการจะออกแบบมีจำนวนของเป็น L เวคเตอร์ในแต่ละเวคเตอร์มีจำนวนอีลีเมนต์เท่ากับ N

$P(f_j)$ = voronoi cell หรือกลุ่มของเวคเตอร์ที่มีสมาชิก คือ เวคเตอร์ โดยมีเวคเตอร์ f_j เป็นค่า centriod ของ $P(f_j)$ $\{j = 1,2,3,\ldots, L\}$

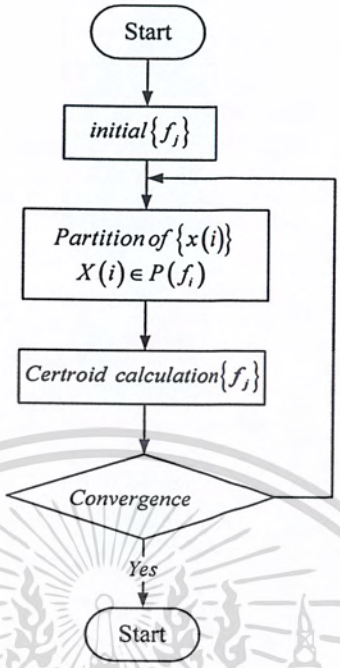
$\{x(i)\}$ = สัญญาณเสียงต้นฉบับสำหรับการออกแบบโค๊ดบุ๊คโดย $\{i = 1,2,3,\ldots, M\}$ และ M คือ จำนวนของเวคเตอร์ใน $\{x(i)\}$

α = เวคเตอร์สำหรับแยกโค๊ดบุ๊ค f_j ให้เป็น 2 ส่วน โดยมีจำนวนอีลีเมนท์เท่ากับ N ใน ที่นี้ได้เลือกใช้ค่า $\alpha = [0.001, 0.001,\ldots,0.001]$

ε = Theshold สำหรับตรวจสอบจุดวิกฤติ (Centical) ในที่นี้ได้เลือกใช้ค่า $\varepsilon = 0.005$

ขั้นตอนการออกแบบ

1) กำหนดให้ Iteration $m = 0$



รูปที่ 2.48 วิธีการของ LBG โดยการประยุกต์จากวิธีการของ Lloyd

2) ติดตั้งโค๊ดบุ้คเวคเตอร์ $f_j(m)$ ที่ $j=1$ โดยการหาค่า Centroid ของสัญญาณเสียงต้นฉบับ $\{x(i)\}$ ได้จาก

$$f_j(m)\frac{1}{M}\sum_{i=1}^{M}x(i) \tag{2.33}$$

$$\{i=1,2,3,..,M\}$$

3) กำหนดเวคเตอร์ $f_j(m)$ ให้เป็น $f_{j(m)}^{old}$ และทำการแยกโค๊ดบุ้คเวคเตอร์ $f_{j(m)}^{old}$ ออกเป็น 2 ส่วน โดยการบวก และลบด้วยเวคเตอร์ α ซึ่งมีจำนวนอีลีเมนท์เท่ากับโค๊ดบุ้คเวคเตอร์ และจะได้โค๊ดบุ้คเวคเตอร์ใหม่ $f_{j(m)}^{new}$ เป็น 2 เท่า ของจำนวนโค๊ดบุ้คเวคเตอร์ เดิมดังนี้

$$f_{j(m)}^{new}=f_{j(m)}^{old}+\alpha$$

$$f_{j(m)}^{new}=f_{j(m)}^{old}-\alpha \tag{2.34}$$

4) กำหนดเวคเตอร์ $f_{j(m)}^{new}$ ให้เป็นเวคเตอร์

5) กำหนดให้ $m = m+1$

6) แยกสัญญาณเสียงต้นฉบับ $\{x(i)\}$ ให้เข้ากลุ่มของ $P(f_j(m))$ ดังนี้

$$x(i) \in P(f_j(m)) \quad \text{if } \|x(i-f_j(m))\| < \|x(i)-f_k\| \tag{2.35}$$

โดยในการจัดเข้ากลุ่มในแต่ละ $Pf_j(m)$ มีเงื่อนไข คือ

$$\|x(i-f_{j(m)})\| < \|x(i)-f_k\|$$

7) นำเอา $\{x(i)\}$ ในขั้นตอนที่ 6 มาทำการหาค่า Centroid ในแต่ละ $P(f_{j(m)})$ ดังนี้

$$f_{j(m)} = \frac{1}{M_j} \sum_{x(i) \in P(f_{j(m)})} x(i) \tag{2.36}$$

$$\{j = 1,2,3,\ldots,L\}$$

โดยที่ M_j คือ จำนวนของเวคเตอร์จากสัญญาณเสียงของต้นฉบับที่ถูกจัดเข้ากลุ่มในแต่ละ $P(f_{j(m)})$

8) คำนวณค่าความเพี้ยนรวมทั้งหมด $D(m)$ (Average Distortion) ของโค๊ตบุ้คเวคเตอร์ $f_{j(m)}$

$$D(m) = \sum_{j=1}^{L} \frac{M_j}{M} D_j(m) \tag{2.37}$$

$$\{j = 1,2,3,\ldots,L\}$$

โดยที่ $D_j(m)$ คือ ความเพี้ยนรวมในแต่ละ Partition $P(f_{j(m)})$ ซึ่งเกิดจากการเปรียบเทียบระหว่างสัญญาณเสียงต้นฉบับ $\{x(i)\}$ และโค๊ตบุ้คเวคเตอร์ $f_{i(m)}$ ในแต่ละ $P(f_{j(m)})$

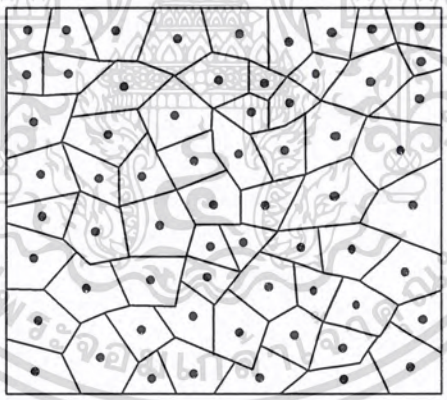
$$D_j(m) = \frac{1}{M_j} \sum_{x(i) \in P(f_{j(m)})} \left\| x(i) - f_{j(m)} \right\|^2 \tag{2.38}$$

$$\{j = 1,2,3,...,L\}$$

9) ตรวจสอบค่าความเพี้ยนรวมทั้งหมดเทียบกับค่า Theshold

$$\frac{D(m-1)-D(m)}{D(m-1)} < \varepsilon \tag{2.39}$$

เมื่อ $D(m-1)$ คือ ค่าความผิดเพี้ยนรวมทั้งหมดที่เกิดขึ้นในครั้งก่อน ถ้าข้อกำหนดจากสมการที่ 2.35 ไม่เป็นจริงแสดงว่าการจัดกลุ่มของสัญญาณเสียงต้นฉบับ $\{x(i)\}$ ให้เข้ากลุ่มของ $P(f_{j(m)})$ ยังเข้าสู่จุดวิกฤติหรือยังไม่ดีพอ และให้กลับไปกระทำในขั้นตอนที่ 5 ถ้าข้อกำหนดจากสมการที่ 2.15 เป็นจริงจะกำหนดให้ $m = 0$ และให้กลับไปกระทำในขั้นตอนที่ 3 โดยการกระทำในแต่ละรอบที่ขั้นตอนที่ 3 จำนวนของโค๊ตบุ้คเวคเตอร์ f_j จะเพิ่มเป็น 2 เท่า คือ 1,2,4,8... จน กระทั่งเวคเตอร์ f_j มีจำนวนเท่ากับ L โดย f_j เป็นค่า Centroid ของ $P(f_j)$ จึงเป็นการสิ้นสุดการออกแบบโค๊ตบุ้ค



รูปที่ 2.49 กลุ่มของ voronoi cell $P(f_j)$ โดยมี f_j เป็นค่า Centroid ซึ่งแสดงด้วยจุดดำ

ในรูปที่ 2.49 แสดงให้เห็นถึงภาพ 2 มิติของกลุ่มของ $P(f_j)$ ที่ได้หลังจากการออกแบบโค๊ตบุ้คโดยมี f_j เป็นค่า Centroid ซึ่งแทนด้วยจุดสีดำที่อยู่ช่วงกลางของเซลล์ สัญญาณอินพุตเวคเตอร์ $x(i)$ ที่ถูกเข้ารหัสแล้วจะถูกจัดเข้ากลุ่มของ $P(f_j)$ ตัวใดตัวหนึ่งที่มีค่าใกล้เคียงกันมากที่สุด และจะสังเกตเห็นว่ากลุ่มของ $P(f_j)$ จะมีรูปแบบที่ไม่แน่นอน ทั้งนี้เนื่องจากในแต่ละ $P(f_j)$ เกิดขึ้นจากสัญญาณ Trainning Vector $\{x(i)\}$ ที่มีค่าใกล้เคียงกันรวมกลุ่มกันแล้วหาค่าเฉลี่ยซึ่ง คือ เวคเตอร์ f_j ที่เก็บอยู่ในโค๊ดบุ้คนั่นเอง

บทที่ 3
การออกแบบ การสร้าง และการทำงาน

3.1 เครื่องมือที่ใช้ในการออกแบบสร้างโปรแกรม

3.1.1 ทางด้านฮาร์ดแวร์

1.1) เครื่องคอมพิวเตอร์

1.2) หน่วยความจำ 256 Mbytes

1.3) ฮาร์ดดิสก์ 40 GB

3.1.2 ทางด้านซอฟต์แวร์

2.1) Microsoft Visual C++ 6.0

2.2) Creative Wave Studio

2.3) Microsoft Sound Recorder

3.2 ขั้นตอนการออกแบบโปรแกรม

3.2.1 ขั้นตอนการออกแบบโปรแกรม มีดังนี้

1) ศึกษาข้อมูลเกี่ยวกับการใช้งานโปรแกรม Microsoft Visual C++ 6.0

2) ศึกษาการเขียนโปรแกรม Visual C++ แบบ MDI

3) ศึกษาการเขียนกราฟฟิกด้วย Visual C++

4) ออกแบบโปรแกรมการทำงานโดยรวม

5) ออกแบบส่วนการเข้า และถอดรหัสแบบ SVQ

6) ออกแบบส่วนการเข้า และถอดรหัสแบบ MP3 และ PCM โดยใช้โปรแกรมสำเร็จรูปประกอบด้วย

7) ออกแบบส่วนการเปรียบเทียบค่าต่างๆ

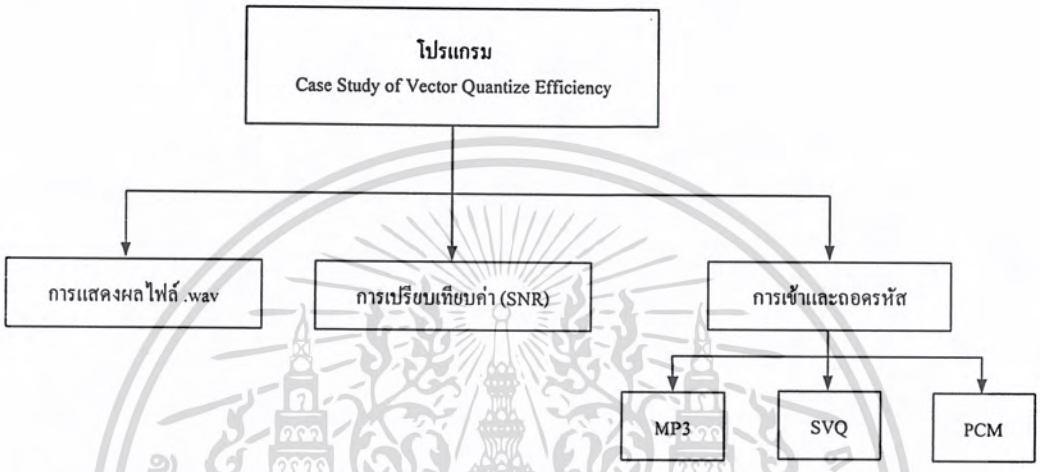
8) ทดสอบโปรแกรมการใช้งานทั้งหมด

9) ปรับปรุงแก้ไขส่วนที่ยังไม่สมบูรณ์

10) จัดทำเป็นแผ่น CD-ROM

3.3 โครงสร้างของโปรแกรม

โครงสร้างของโปรแกรมจะแบ่งออกเป็น 3 ส่วนหลักๆ คือ ส่วนการแสดงผลไฟล์ .WAV ส่วนการเข้า และถอดรหัส และส่วนการเปรียบเทียบค่า SNR ซึ่งสามารถมองออกมาเป็นโครงสร้างของโปรแกรมได้ดังนี้



รูปที่ 3.1 โครงสร้างของโปรแกรม Case Study of Vector Quantize Efficiency

3.4 ผังงานของโปรแกรม Case Study of Vector Quantize Efficiency

ผังงานของโปรแกรม Case Study of Vector Quantize Efficiency แสดงดังรูปที่ 3.2

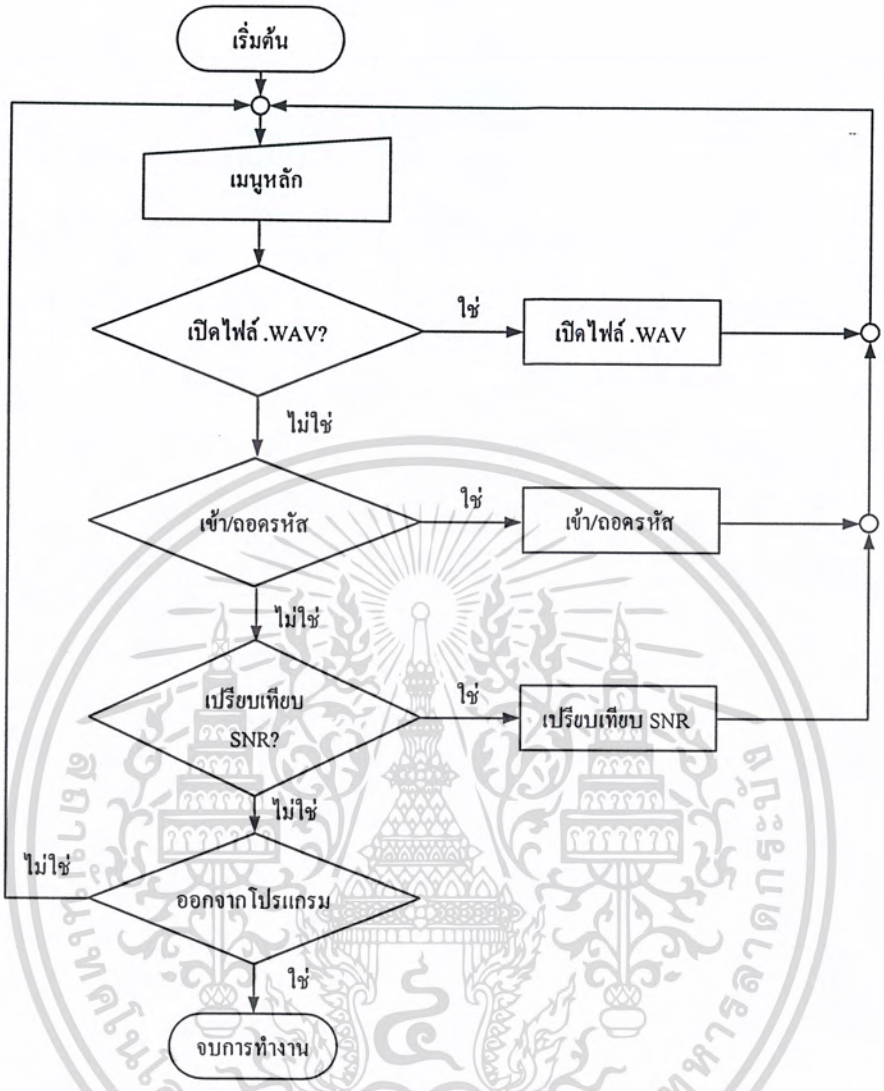
เริ่มโปรแกรมจะเข้าสู่เมนูหลัก จะทำการเลือกการทำงานในแต่ละส่วน ซึ่งแบ่งเป็น 3 ส่วน ดังนี้

1) การเปิดไฟล์

2) การเข้าและถอดรหัส

3) การเปรียบเทียบสัญญาณต่อสัญญาณรบกวน

โดยในการใช้งานจะเป็นส่วนที่แยกออกจากกันในแต่ละเมนู สามารถเลือกใช้งานในแต่ละส่วนตามต้องการ หากเข้าไปในส่วนการใช้งานแล้วต้องการยกเลิกการทำงาน ก็สามารถยกเลิกได้โดยไม่มีผลกระทบต่อการใช้งานส่วนอื่น

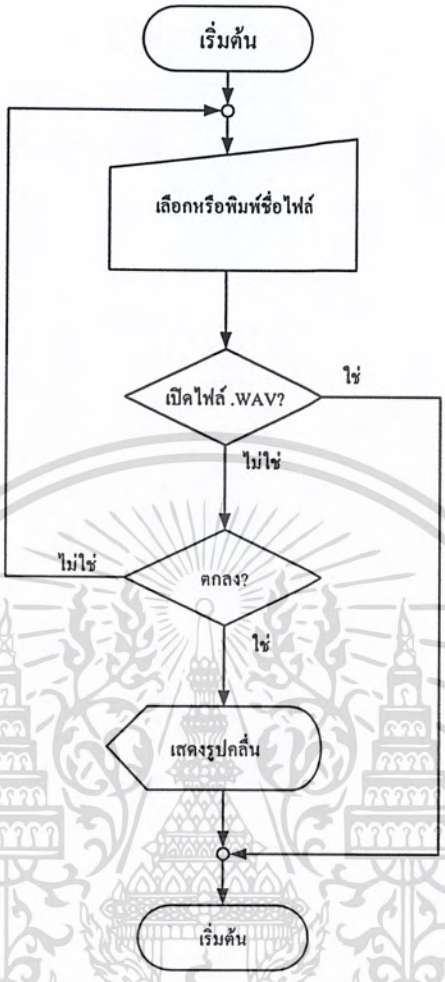


รูปที่ 3.2 ผังงานของโปรแกรม Case Study of Vector Quantize Efficiency

3.5 ผังงาน ของส่วนการเปิดไฟล์ .WAV

ผังงาน ส่วนการเปิดไฟล์ .WAV สามารถแสดงดังรูปที่ 3.3

การเปิดไฟล์ของสัญญาณ เมื่อเปิดโปรแกรมสามารถเลือกหรือพิมพ์ชื่อไฟล์ ที่ต้องการให้แสดงสัญญาณขึ้นมาได้ หลังจากนั้นเลือกตกลง เพื่อแสดงภาพสัญญาณ หากต้องการยกเลิกการเปิดไฟล์ ให้เลือกยกเลิก จะจบการทำงานของส่วนการเปิดไฟล์

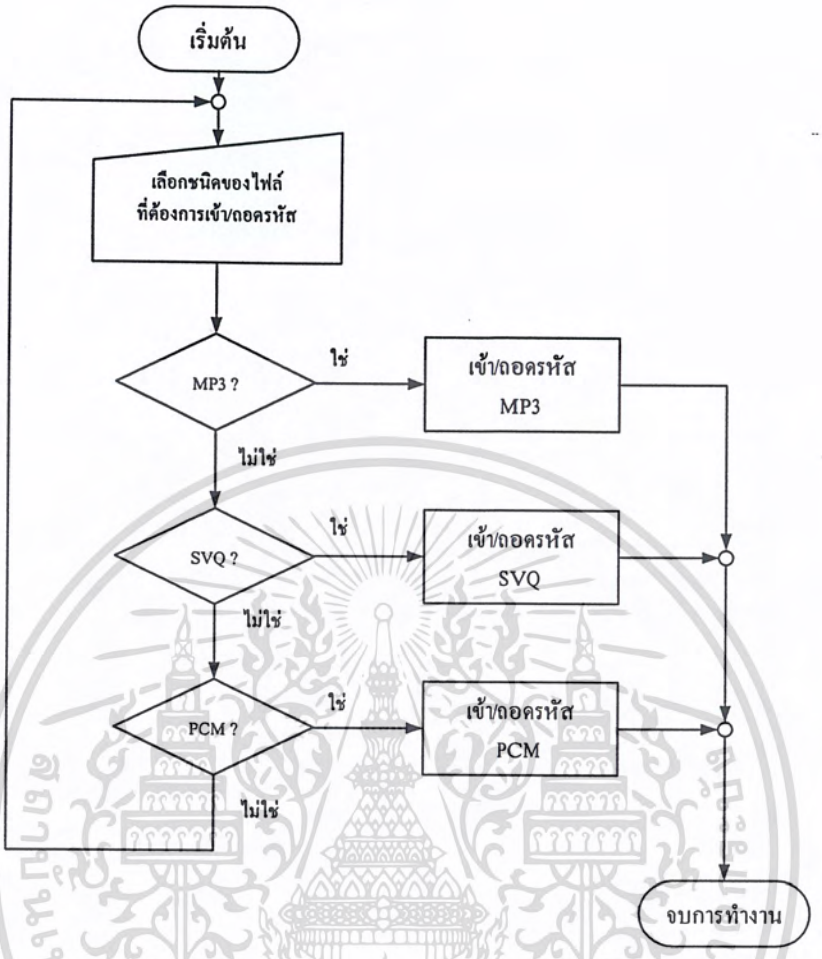


รูปที่ 3.3 ผังงาน ส่วนการเปิดไฟล์ .WAV

3.6 ผังงาน ของส่วนการเข้า / ถอดรหัส

ผังงาน ส่วนการเข้า / ถอดรหัส สามารถแสดงดังรูปที่ 3.4

การเข้ารหัส และถอดรหัสในโปรแกรมแบ่งได้เป็น 3 ส่วน โดยเราจะเลือกตามชนิดที่ต้องการเข้ารหัส เมื่อเลือก MP3 จะเข้าสู่ส่วนของโปรแกรมเข้ารหัสและถอดรหัส MP3 แบ่งย่อยเป็น ส่วนของการเข้ารหัส/ถอดรหัส และส่วนของการปรับเฟส เมื่อเลือก SVQ จะเข้าสู่ส่วนของโปรแกรมเข้ารหัสและถอดรหัสเวคเตอร์ควอนไตส์ แบ่งย่อยออกเป็นส่วนของการออกแบบโค๊ดบุ้คและส่วนของการสร้างสัญญาณจากโค๊ดบุ้ค ส่วนสุดท้ายจะเป็นการเข้ารหัส/ถอดรหัสแบบ PCM แบ่งย่อยเป็นกฎ μ- law และกฎ A – law หากไม่ต้องการเข้ารหัส และถอดรหัสสัญญาณใดๆ ให้ออกจากโปรแกรม

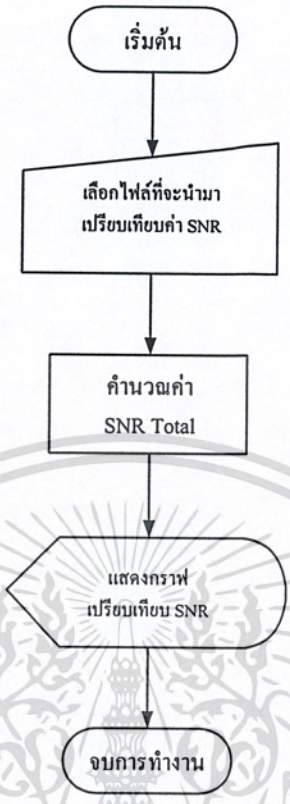


รูปที่ 3.4 ผังงาน ส่วนการเข้า / ถอดรหัส

3.7 ผังงาน ของส่วนการเปรียบเทียบค่า SNR

ผังงาน ส่วนการเปรียบเทียบค่า SNR สามารถแสดงดังรูปที่ 3.5

การทำงานโดยเลือกไฟล์ที่จะนำมาเปรียบเทียบอัตราสัญญาณต่อสัญญาณรบกวน ซึ่งเลือกได้สูงสุด 3 สัญญาณ เทียบกับสัญญาณต้นฉบับที่กำหนดเข้าไป เมื่อตกลง โปรแกรมจะทำการคำนวณค่าอัตราสัญญาณต่อสัญญาณรบกวนออกมา และแสดงกราฟของสัญญาณแต่ละตัว เทียบกับสัญญาณต้นฉบับ เป็นการสิ้นสุดการทำงานของโปรแกรม



รูปที่ 3.5 ผังงาน ส่วนการเปรียบเทียบค่า SNR

บทที่ 4
การทดลอง และผลการทดลอง

4.1 กล่าวนำ

ผู้ใช้งานโปรแกรมคอมพิวเตอร์กรณีศึกษาประสิทธิภาพของเวคเตอร์ควอนไตส์นี้ควรมีพื้นฐานความรู้เรื่องการใช้เครื่องคอมพิวเตอร์มาบ้าง สำหรับผู้ที่ไม่มีพื้นฐานเลย โปรแกรมจะมีการอธิบายการใช้งานต่างๆ ไว้ในโปรแกรมแล้ว สำหรับการทดลอง และผลการทดลองโปรแกรมกรณีศึกษาประสิทธภาพของเวคเตอร์ควอนไตส์นี้ จะแบ่งเป็น 2 ส่วนใหญ่ๆ คือ ส่วนของการทดลอง และผลของการทดลอง การทดลองจะทำให้ทราบข้อผิดพลาด และทราบว่าข้อผิดพลาดอยู่ในส่วนใดของโปรแกรม

4.2 ความต้องการของโปรแกรม

1) เครื่องคอมพิวเตอร์ Pentium 166 ขึ้นไป
2) หน่วยความจำต้องไม่ต่ำกว่า 32 Mbyte
3) พื้นที่ว่างในฮาร์ดดิสก์ 100 Mbyte
4) เมาส์และคีย์บอร์ด
5) ระบบปฏิบัติการ Windows 98 ขึ้นไป

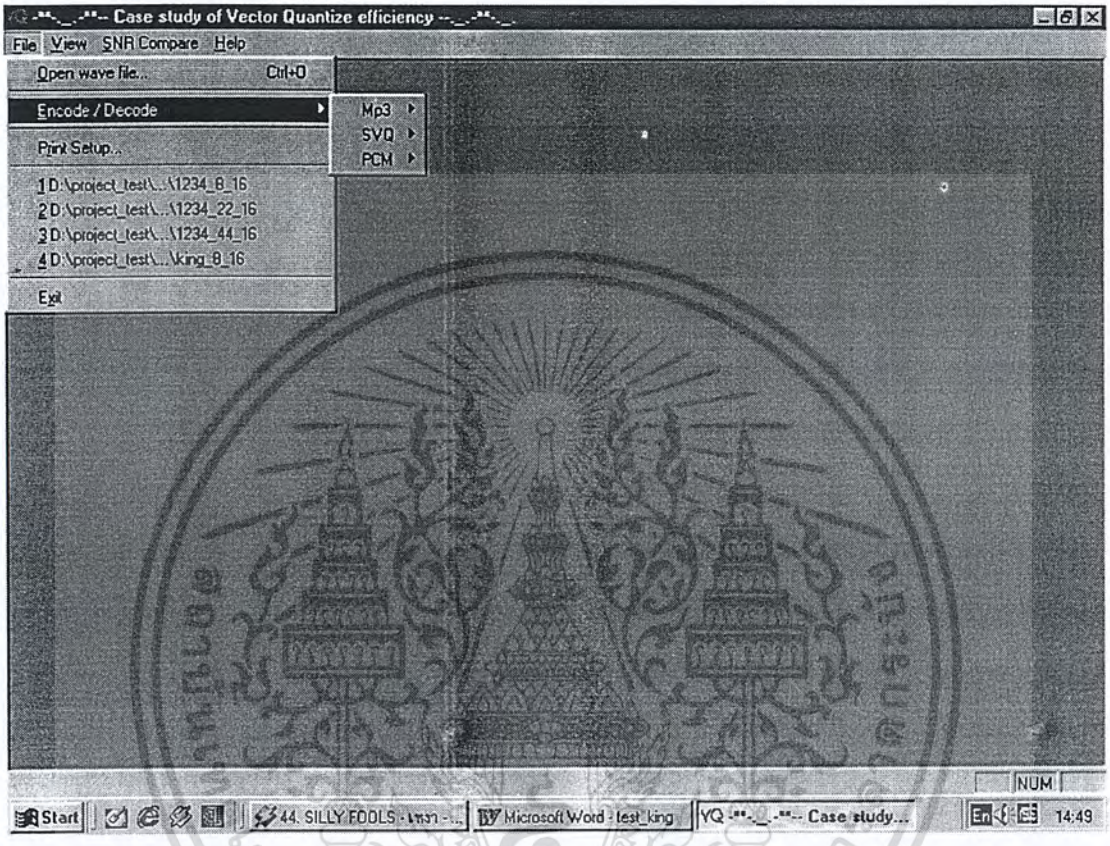
4.3 การเข้าสู่โปรแกรม

การเข้าสู่โปรแกรมกรณีศึกษาประสิทธิภาพของเวคเตอร์ควอนไตส์เมื่อติดตั้งลงบนฮาร์ดดิสก์ในคอมพิวเตอร์แต่ละเครื่องเรียบร้อยแล้วก็จะสามารถเรียกใช้งานได้ทันที

เมื่อเรียกโปรแกรมกรณีศึกษาประสิทธิภาพของเวคเตอร์ควอนไตส์แล้ว จะแสดงเมนูหลักขึ้นมา ดังรูปที่ 4.1

4.4 เมนูหลัก

ในเมนูหลักจะประกอบไปด้วยเนื้อหา 3 เรื่องด้วยกันดังรูป



รูปที่ 4.1 เมนูหลัก

4.5 การทดลองกรณีศึกษาประสิทธิภาพของเวคเตอร์ควอนไตส์

4.5.1 ลำดับขั้นการทดลองการเข้ารหัส – ถอดรหัส MP3

เมื่อทำการเปิดโปรแกรม Case Study of Vector Quantize Efficiency ขึ้นมาแล้ว สามารถใช้งานโปรแกรมตามลำดับขั้น ดังนี้

1) เลือกที่ File → Encode/Decode → MP3

2) เลือกที่ Add

3) Browse ไฟล์ต้นฉบับ . WAV ที่ต้องการเปรียบเทียบ

4) เลือก Output ที่ต้องการ

5) เลือก Directory ที่ต้องการให้ผลของการเข้ารหัสของสัญญาณไปอยู่ที่ใด

6) เลือก Encode เพื่อแปลงจากไฟล์ . WAV เป็น MP3

7) ปิดโปรแกรม WAV to MP3 โปรแกรม MP3 to WAV จะปรากฏ

8) เลือกที่ Add แล้ว Browse ไฟล์ . MP3 ที่ได้จากการ Encode จากโปรแกรมข้างต้น

9) เลือก Output ที่ต้องการให้ผลของการถอดรหัสของสัญญาณไปอยู่ที่ใด

10) เลือก Decode เพื่อทำการ Converse สัญญาณ . MP3 ให้เป็น . WAV

11) ปิดโปรแกรม MP3 to WAV

12) เลือก SNR Compare→SNR Graph

13) Browse สัญญาณไฟล์ . WAV ต้นฉบับ

14) Browse สัญญาณที่ต้องการทำการเปรียบเทียบ

4.5.2 ลำดับขั้นการทดลองการเข้ารหัส – ถอดรหัส SVQ

เมื่อทำการเปิดโปรแกรม Case Study of Vector Quantize Efficiency ขึ้นมาแล้ว จะสามารถใช้งานโปรแกรมตามลำดับขั้น ดังนี้

1) เลือกที่ File → Encode/Decode →SVQ→ Codebook เพื่อทำการสร้าง Codebook

2) Browse สัญญาณต้นฉบับขึ้นมา

3) กำหนดค่าต่างๆ ดังที่กำหนดไว้ดังนี้

3.1) กำหนด Dimension ค่าระหว่าง 1 – 10

3.2) กำหนด Size ค่าระหว่าง 1 – 512 (ยิ่งละเอียดมากความถูกต้องของสัญญาณก็จะมีมาก)

3.3) กำหนด Splitting Value ค่าระหว่าง 0.1 – 100

3.4) กำหนด Theshold ค่าระหว่าง 0.00001 – 1.0

4) กำหนด Output File ของ Codebook เพื่อเก็บ Codebook ที่ทำการสร้างเสร็จแล้ว

5) เลือกที่ File → Encode/Decode →SVQ→ ENSVQ

6) เลือก File ต้นฉบับ เพื่อนำไปสร้าง ไฟล์ WAV ใหม่ขึ้นมา

7) เลือก Codebook ที่สร้างเสร็จแล้ว

8) กำหนด Output File จากการสร้าง ไฟล์ . WAV จาก Codebook

9) เลือก SNR Compare→SNR Graph

10) Browse สัญญาณไฟล์ . WAV ต้นฉบับ

11) Browse สัญญาณที่ต้องการทำการเปรียบเทียบ

4.5.3 ลำดับขั้นการทดลองการเข้ารหัส – ถอดรหัส PCM แบบ A – Law

1) File → Encode/Decode → PCM → A-Law เพื่อทำการเข้า และถอดรหัส PCM แบบ A – Law
2) Browse สัญญาณต้นฉบับขึ้นมา
3) Browse เพื่อตั้งชื่อไฟล์ปลายทาง
4) เลือก OK. เพื่อทำการเข้า และถอดรหัส

4.5.4 ลำดับขั้นการทดลองการเข้ารหัส – ถอดรหัส PCM แบบ μ – Law

1) File → Encode/Decode → PCM → μ-Law เพื่อทำการเข้า และถอดรหัส PCM แบบ μ – Law
2) Browse สัญญาณต้นฉบับขึ้นมา
3) Browse เพื่อตั้งชื่อไฟล์ปลายทาง
4) เลือก OK. เพื่อทำการเข้า และถอดรหัส

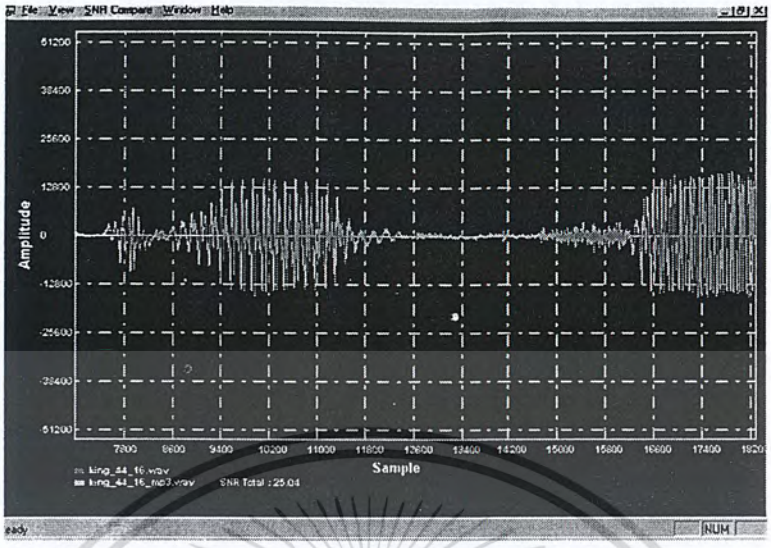
4.6 ผลการทดลองของกรณีศึกษาประสิทธิภาพของเวคเตอร์ควอนไตส์

การเปรียบเทียบแบ่งเป็น 2 ส่วน คือ SVQ กับ A – law , μ - law และ SVQ กับ MP3 โดยบิตเรต A – law , μ - law มีค่าเท่ากับ 64 Kb/sและ บิตเรต MP3 มีค่าเท่ากับ 128 Kb/s สาเหตุที่ต้องแยกการเปรียบเทียบออกเป็น 2 ส่วน คือ บิตเรตของ MP3 กับ A – law , μ - law ไม่เท่ากัน จำเป็นที่จะต้องปรับ SVQ ให้ใกล้เคียงกับ MP3 และ A – law , μ - law ให้ใกล้เคียง มากที่สุด

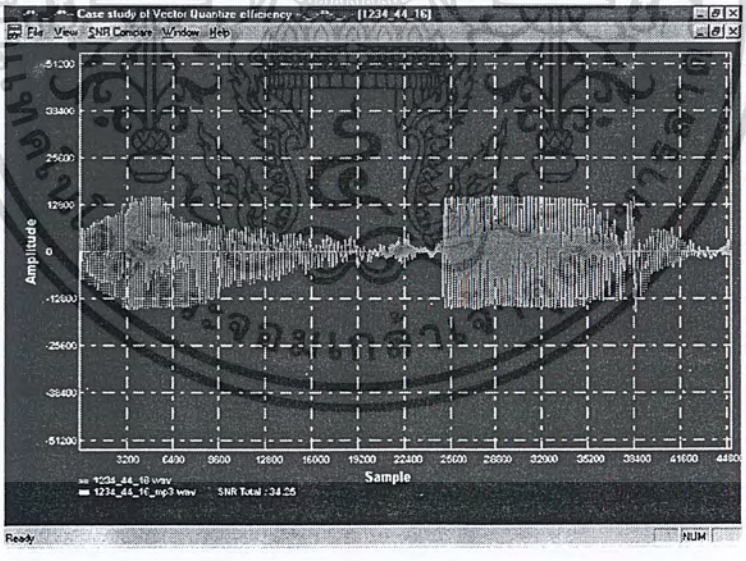
4.6.1 ผลการเปรียบเทียบของการเข้ารหัส – ถอดรหัส ระหว่าง MP3 และ SVQ

กำหนดให้สัญญาณไฟล์ king_44_16.wav (สัญญาณต้นฉบับ)โดยพูดคำว่า "พระจอมเกล้า" ซึ่งไฟล์นี้มีความถี่ Sampling Rate ที่ 44.1 khz การเข้ารหัสเป็นแบบ 16 Bit แบบ Mono ซึ่งในขั้นตอนแรกจะทำการทดสอบหาค่า SNR เพื่อหา ประสิทธิภาพของการเข้ารหัส และถอด รหัสแบบ MP3 จะได้บิตเรตในการส่งข้อมูลที่ 128 kb/s ซึ่งผลของค่า SNR ได้ค่า เท่ากับ 25.04 แสดงได้ ดังรูปที่ 4.2

ในขั้นตอนต่อมากำหนดให้ไฟล์1234_44_16.wavโดยพูดคำว่า "หนึ่ง...สอง...สาม...สี่" โดยมีความถี่ Sampling Rate ที่ 44.1 khz การเข้ารหัสเป็นแบบ 16 Bit แบบ Mono จะทำการทดสอบหาค่า SNR เพื่อหาประสิทธิภาพของการเข้ารหัส และถอดรหัสแบบ MP3 ซึ่งผลของค่า SNR ได้ค่า เท่ากับ 34.25 แสดงได้ดังรูปที่ 4.3



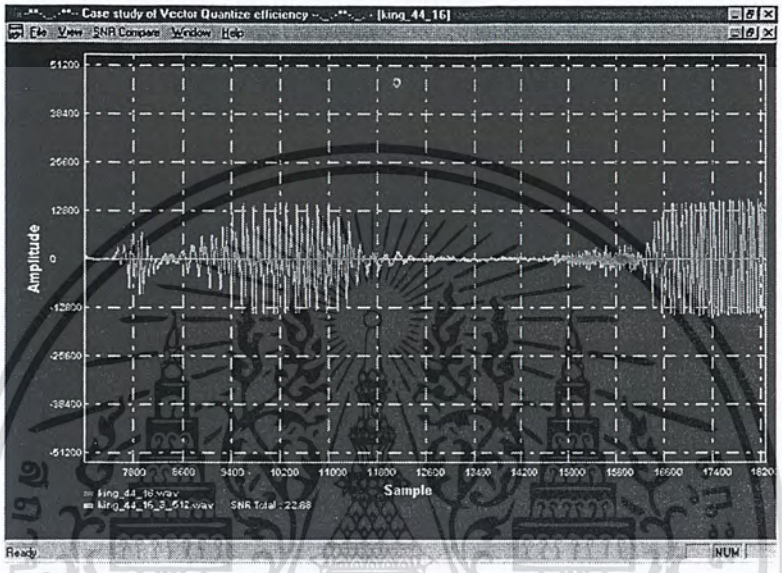
รูปที่ 4.2 ผลการเปรียบเทียบการเข้า และถอดรหัสของไฟล์ king_44_16.wav ในแบบ MP3 ซึ่งได้ค่า SNR otal : 25.04



รูปที่ 4.3 ผลการเปรียบเทียบการเข้า และถอดรหัสของไฟล์ 1234_44_16.wav ในแบบ MP3 ซึ่งได้ค่า SNR Total : 34.25

ในส่วนของ SVQ ได้กำหนด สัญญาณไฟล์ king_44_16.wav (สัญญาณต้นฉบับ หรือ สัญญาณที่ใช้ในการออกแบบ Coodbook) โดยพูดคำว่า "พระจอมเกล้า" ซึ่งไฟล์นี้มีความถี่

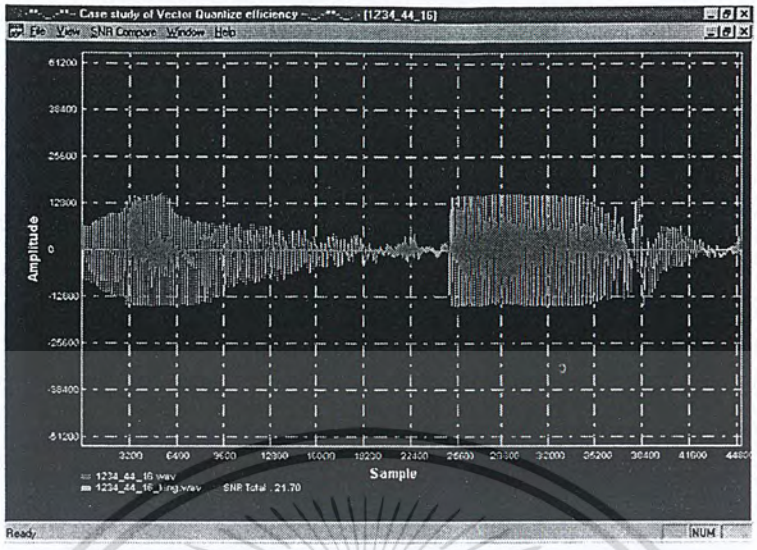
Sampling Rate ที่ 44.1 khz การเข้ารหัสเป็นแบบ 16 Bit แบบ Mono การเข้ารหัส และถอดรหัส SVQ ไม่สามารถกำหนดบิตเรตในการส่งข้อมูลได้เท่ากับ 128 kb/s ซึ่งกำหนดได้ใกล้เคียงที่ 132 kb/s และ ประสิทธิภาพของการเข้ารหัส และถอดรหัสแบบ SVQ ผลของค่า SNR ได้ เท่ากับ 22.88 แสดงได้ดังรูปที่ 4.4



รูปที่ 4.4 ผลการเปรียบเทียบการเข้า และถอดรหัสของไฟล์ king_44_16.wav ในแบบ SVQ ซึ่งได้ค่า SNR Total : 22.88

ในกรณีที่ใช้สัญญาณที่ไม่ใช่ต้นฉบับมาทำการเข้ารหัสและถอดรหัสแบบ SVQ ซึ่งใช้ไฟล์ 1234_44_16.wav โดยพูดคำว่า "หนึ่ง...สอง...สาม...สี่" ซึ่งไฟล์นี้มีความถี่ Sampling Rate ที่ 44.1 khz การเข้ารหัสเป็นแบบ 16 Bit แบบ Mono โดยกำหนดบิตเรตในการส่งข้อมูลไว้ที่ 132 kb/sผลของค่า SNR ได้ค่า เท่ากับ 21.70 ซึ่งแสดงได้ดังรูปที่ 4.5

จากการทดสอบหาค่า SNR ของการเข้ารหัส และการถอดรหัสแบบ MP3 ที่บิตเรตในการส่งข้อมูล 128 kb/s เทียบกับการเข้า และการถอดรหัสแบบ SVQ ที่บิตเรตในการส่งข้อมูล 128 kb/s จะสังเกตได้ว่าค่า SNR ของ MP3 มากกว่า SVQ ซึ่งสรุปได้ดังตารางที่ 4.1



รูปที่ 4.5 ผลของการเปรียบเทียบระหว่างสัญญาณ 1234_44_16.wav ซึ่งใช้ Code book ของไฟล์ king_44_16.w ค่า SNR Total : 21.70

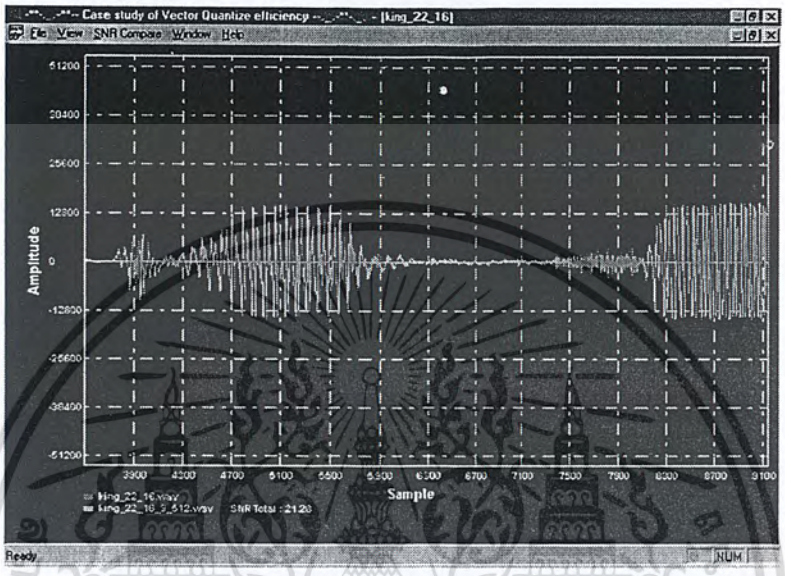
ตารางที่ 4.1 การเปรียบเทียบค่า SNR ของการเข้ารหัส-ถอดรหัสสัญญาณแบบเวคเตอร์ควอนไตส์ และ MP3

SVQ (132 kb/s)	MP3 (128 kb/s)
king_44_16_3_512.wav SNR :22.88	king_44_16_mp3.wav SNR :25.04
1234_44_16_king.wav SNR :21.70	1234_44_16_mp3.wav SNR :34.25

4.6.2 ผลการเปรียบเทียบของการเข้ารหัส – ถอดรหัส ระหว่าง SVQ และPCM แบบ A – Law และ μ-Law

ในระบบ PCM แบบ A-Law และ แบบ μ-Law มีบิตเรตในการส่งข้อมูลที่ 64 kb/s ซึ่งไม่สามารถนำไปเปรียบเทียบค่า SNR กับการเข้า และการถอดรหัสแบบ MP3 ซึ่งมีบิตเรต ในการส่งข้อมูลที่ 128 kb/s ดังนั้นจึงปรับบิตเรตของ SVQ ให้ได้ใกล้เคียง 64 kb/s ซึ่งกระทำได้โดยกำหนดให้สัญญาณไฟล์ king_22_16.wav (สัญญาณต้นฉบับ) มีความถี่ Sampling Rate ที่22.05 khz การเข้ารหัสเป็นแบบ 16 Bit แบบ Mono และได้กำหนดให้มิติของ Codebook เท่ากับ 3 ขนาด

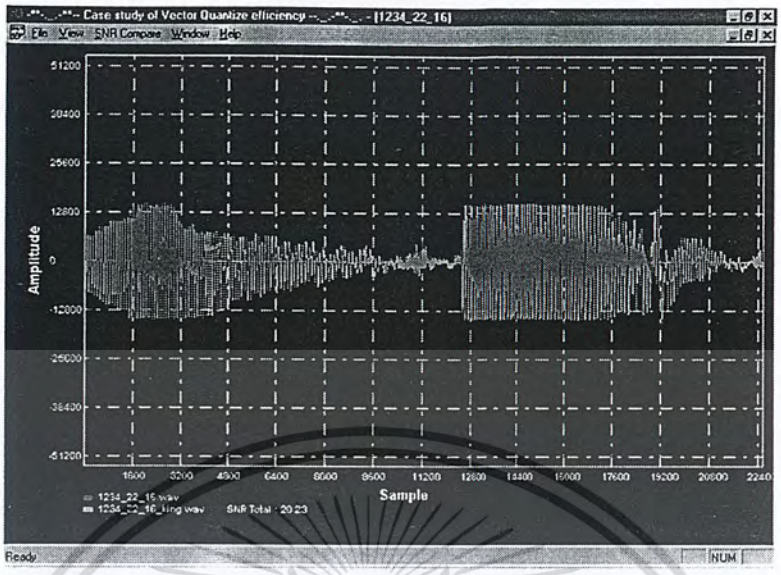
เท่ากับ 512 จะได้บิตเรตในการส่งข้อมูลที่ 66 kb/s เมื่อทำการออกแบบ Codebook ด้วยสัญญาณเสียงต้นฉบับนี้แล้วสามารถคำนวณหาค่า SNR ของการเข้ารหัส และถอดรหัสแบบ SVQ ซึ่งผลของค่า SNR ได้เท่ากับ 21.28 และแสดงได้ดังรูปที่ 4.6



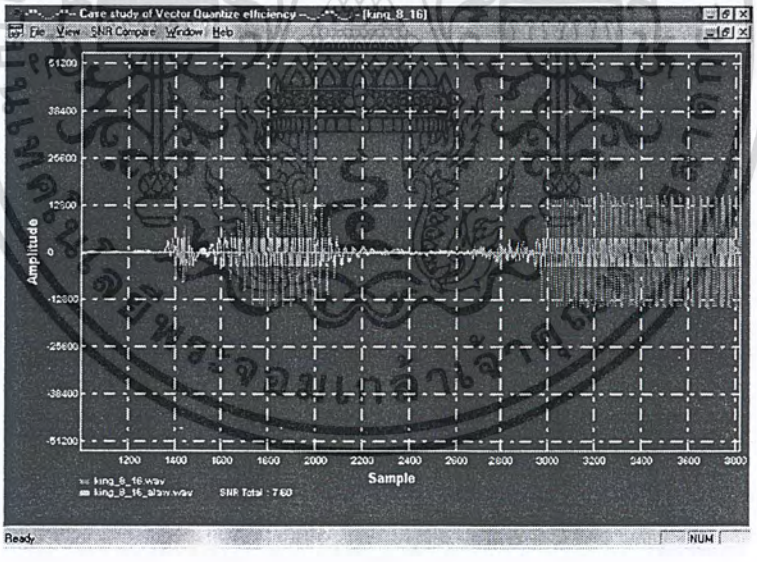
รูปที่ 4.6 ผลของการเปรียบเทียบระหว่างสัญญาณ king_22_16.wav และ king_22_16_3_512.wav ค่า SNR Total : 21.28

เช่นเดียวกันกับในกรณีแรกเมื่อต้องการทดสอบค่า SNR กับสัญญาณเสียงที่ไม่ใช่ต้นฉบับ คือไฟล์ 1234_22_16.wav จำเป็นจะต้องกำหนด Sampling Rate ไว้ที่ 22.05 khz เช่นกัน และเมื่อทำการเข้ารหัส และถอดรหัสแบบ SVQ ด้วยสัญญาณเสียงนี้แล้วผลของค่า SNR ที่ได้คือ 20.23 แสดงได้ดังรูปที่ 4.7

ในขั้นตอนต่อมาจะทำการทดสอบหาค่า SNR ของการเข้ารหัส และถอดรหัสแบบ A– Law โดยใช้ไฟล์เสียงชนิดเดียวกันกับ SVQ แต่ต้องกำหนดให้ Sampling Rate เท่ากับ 8 khz จึงจะได้บิตเรตในการส่งข้อมูล 64 kb/s ซึ่งค่า SNR ของไฟล์ king_8_16.wav ได้เท่ากับ 7.60 และค่า SNR ของไฟล์ 1234_8_16 . wav มีค่าเท่ากับ 9.62 ดังแสดงในรูปที่ 4.8 และรูปที่ 4.9 ตามลำดับ

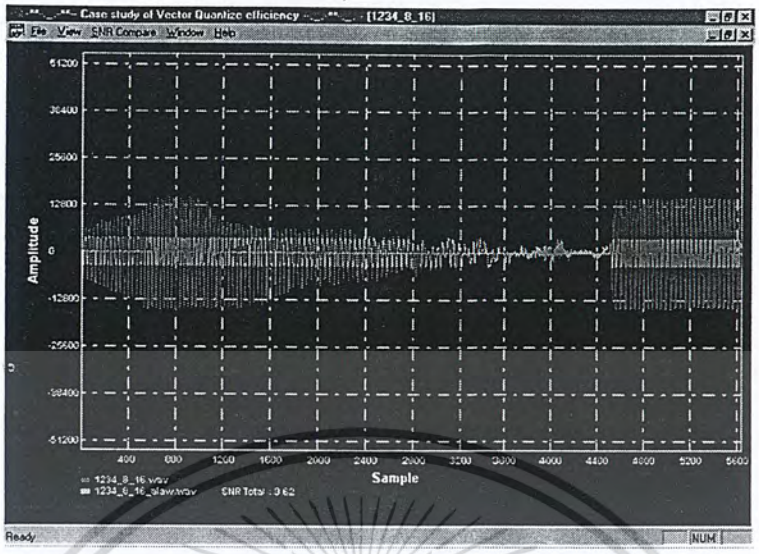


รูปที่ 4.7 ผลของการเปรียบเทียบระหว่างสัญญาณ 1234_22_16.wav และ 1234_22_16_king.wav ค่า SNR Total : 20.23



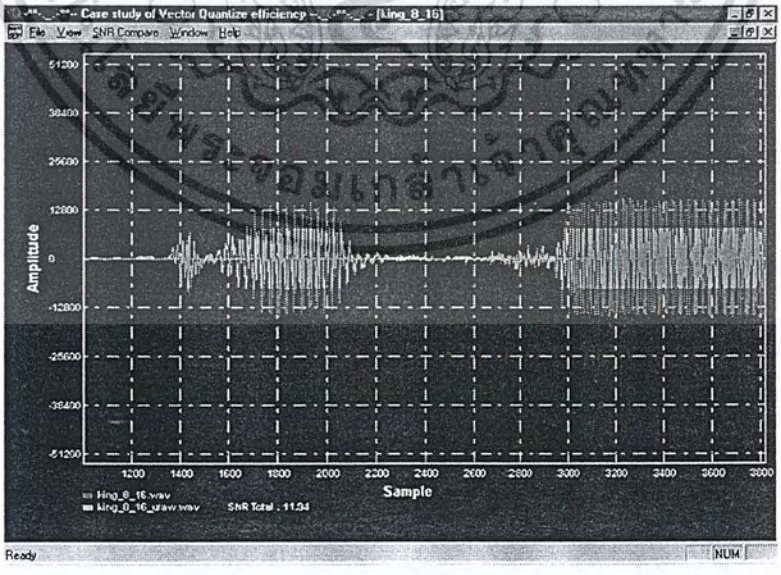
รูปที่ 4.8 ผลของการเปรียบเทียบระหว่างสัญญาณ king_8_16.wav และ king_8_16_alaw.wav ค่า SNR Total : 7.60

ในขั้นตอนต่อมาจะทำการทดสอบหาค่า SNR ของการเข้ารหัส และถอดรหัสแบบ μ – Law โดยใช้ไฟล์เสียงชนิดเดียวกันกับ SVQ แต่ต้องกำหนดให้ Sampling Rate เท่ากับ 8 khz

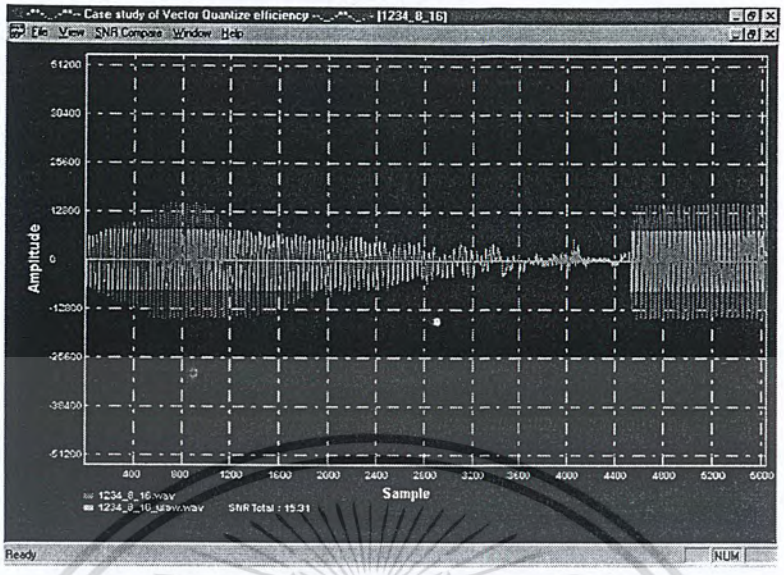


รูปที่ 4.9 ผลของการเปรียบเทียบระหว่างสัญญาณ 1234_8_16.wav และ 1234_8_16_alaw.wav ค่า SNR Total : 9.62

จึงจะได้บิตเรตในการส่งข้อมูล 64 kb/s ซึ่งค่า SNR ของไฟล์ king_8_16.wav ได้เท่ากับ 11.94 และค่า SNR ของไฟล์ 1234_8_16 . wav มีค่าเท่ากับ 15.31 ดังแสดงในรูปที่ 4.10 และรูปที่ 4.11 ตามลำดับ



รูปที่ 4.10 ผลของการเปรียบเทียบระหว่างสัญญาณ king_8_16.wav และ king_8_16_ulaw.wav ค่า SNR Total : 11.94



รูปที่ 4.11 ผลของการเปรียบเทียบระหว่างสัญญาณ 1234_8_16.wav และ 1234_8_16_ulaw.wav ค่า SNR Total : 15.31

จากการทดสอบหาค่า SNR ของการเข้ารหัส และการถอดรหัสแบบ A- Law และ μ - Law ที่บิตเรตข้อมูล 64 kb/s เทียบกับการเข้า และการถอดรหัสแบบ SVQ ที่บิตเรตข้อมูล 66.15 kb/s จะสังเกตุได้ว่าค่า SNR ของ SVQ มากกว่า A- Lawและ μ - Law ซึ่งสรุปได้ดังตารางที่ 4.2

ตารางที่4.2 การเปรียบเทียบค่า SNR ของการเข้ารหัส-ถอดรหัสสัญญาณแบบเวคเตอร์ควอนไตส์ และ PCM แบบ A-Law และ μ-Law

SVQ (66.15 kb/s)	A-Law (64 kb/s)	u-Law (64 kb/s)
king_22_16_3_512.wav SNR :21.28	king_8_16_alaw.wav SNR : 7.60	king_8_16_ulaw.wav SNR : 11.92
1234_22_16_king.wav SNR :20.23	1234_8_16_alaw.wav SNR : 9.62	1234_8_16_ulaw.wav SNR : 15.31

บทที่ 5
บทสรุป ปัญหา แนวทางการแก้ไข และพัฒนา

5.1 บทสรุป

ปริญญานิพนธ์ฉบับนี้ ได้เสนอโปรแกรม การเปรียบเทียบสัญญาณเสียงที่ทำการเข้ารหัสแบบต่างๆ ซึ่งจะทำการเปรียบเทียบว่าการเข้ารหัสแบบใดมีผลของสัญญาณที่คล้ายกับสัญญาณต้นฉบับมากที่สุด เพื่อที่จะศึกษา และทราบถึงผลของการเข้ารหัสแต่ละแบบว่ามีการทำงานอย่างไร และได้ผลเป็นอย่างไร ซึ่งจากการศึกษา จัดทำ และทดลอง ก็เป็นที่น่าพอใจ และสามารถทำงานได้อย่างมีประสิทธิภาพในระดับหนึ่ง แต่อย่างไรก็ตามก็ยังมีส่วนที่เป็นปัญหาของโปรแกรมอยู่บ้างบางประการ ซึ่งจะได้กล่าวต่อไป

5.2 ปัญหาและแนวทางการแก้ไข

สำหรับปัญหาที่เกิดขึ้นในการทำโครงงาน ตั้งแต่เริ่มการจัดทำ จนสำเร็จเป็นโปรแกรมชุดนี้ มีปัญหา และแนวทางการแก้ไขดังนี้

1) **ปัญหา** ในส่วนการศึกษาข้อมูลเกี่ยวกับการใช้งาน โปรแกรม Visual C++ นั้น หนังสือที่ใช้อ้างอิงมีจำนวนน้อยมาก ทำให้การเขียนโปรแกรมในส่วนที่ต้องการนั้นต้องใช้เวลานานในการศึกษาหาความรู้

แนวทางการแก้ไข ศึกษาจากตัวอย่างโปรแกรมที่เขียนขึ้นจาก VC++ ประกอบกับหนังสือภาษาไทยที่มีอยู่ และแผ่นโปรแกรมช่วยสอนวิธีการใช้โปรแกรม VC++

2) **ปัญหา** เทคนิคบางอย่างของ VC++ เช่น การแสดง Scroll Bar นั้นจากแหล่งข้อมูลที่มีอยู่นั้นเมื่อทดลองทำตามตัวอย่างในหนังสือแล้ว ไม่สามารถนำมาใช้งานได้

แนวทางการแก้ไข ทำการเขียนโค้ดเพิ่มเติมขึ้นมาเองเพื่อให้สามารถทำงานแทน Scroll Bar ทำให้สามารถเลื่อนรูปคลื่นที่แสดงออกมาทางหน้าจอได้

3) **ปัญหา** ในส่วนของการเข้ารหัสแบบ MP3 นั้นมีความซับซ้อนมาก และยากที่จะหาแหล่งความรู้เกี่ยวกับการเข้า และถอดรหัสแบบ MP3 จึงไม่สามารถเขียนโปรแกรมในส่วนนี้ได้

แนวทางการแก้ไข ใช้โปรแกรมสำเร็จรูปที่สามารถเข้า และถอดรหัสแบบ MP3 ได้มาใช้แทน แล้วจึงนำผลมาทดสอบ

4) **ปัญหา** การเข้ารหัสแบบ MP3 นั้นเมื่อถอดรหัสกลับคืนมาเป็นไฟล์ต้นฉบับ (.WAV) แล้วจะมีรูปร่างและขนาดเลื่อนเฟสไปจากไฟล์ต้นฉบับ จึงไม่สามารถเปรียบเทียบได้

แนวทางการแก้ไข เขียนโปรแกรมส่วนของการปรับเฟสขึ้นมาเพิ่มเติม

5.3 แนวทางการพัฒนา

เนื่องด้วยเทคโนโลยีในปัจจุบัน มีการพัฒนาอยู่เสมอ ปริญญานิพนธ์ฉบับนี้ยังสามารถที่จะเพิ่มขีดความสามารถในการทำงานได้ดังนี้คือ

1) สามารถนำการเข้ารหัส และถอดรหัสแบบอื่นๆ มาร่วมเปรียบเทียบด้วย ทำให้โปรแกรมมีประสิทธิภาพมากยิ่งขึ้น

2) สามารถศึกษาเทคนิคต่างๆ ในการเขียนโปรแกรมด้วย VC++ เพื่อให้โปรแกรมมีความสะดวก และสมบูรณ์มากกว่าเดิม เช่นมี Scroll Bar เพื่อให้สะดวกมากขึ้น

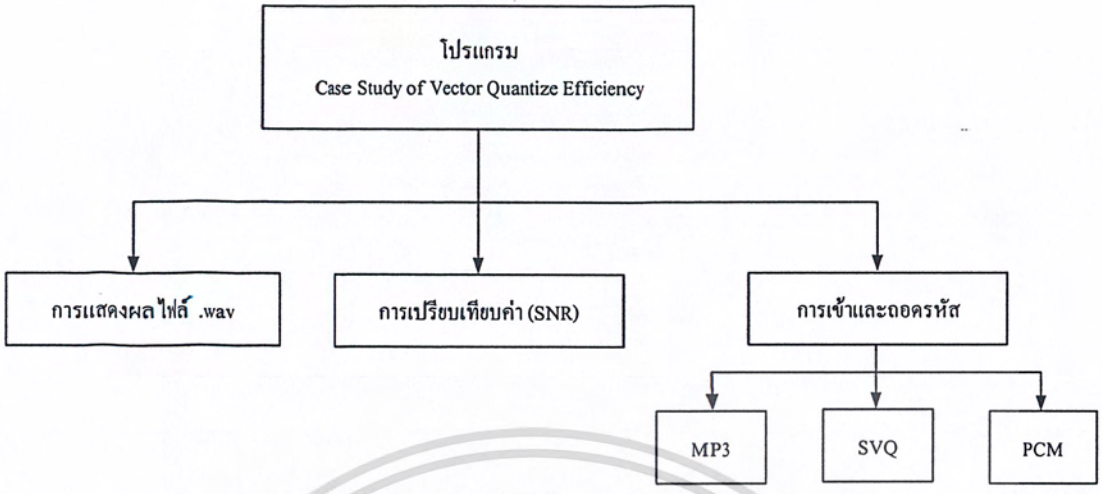
3) สามารถเปิดไฟล์เสียงที่เป็นลักษณะอื่น (นามสกุลอื่น) ได้ ซึ่งโปรแกรมตอนนี้สามารถเปิดได้เฉพาะไฟล์นามสกุล .WAV เท่านั้น

4) สามารถพัฒนาให้ใช้งานแบบ Real Time ได้ จะทำให้นำไปประยุกต์ใช้งานต่อไปได้

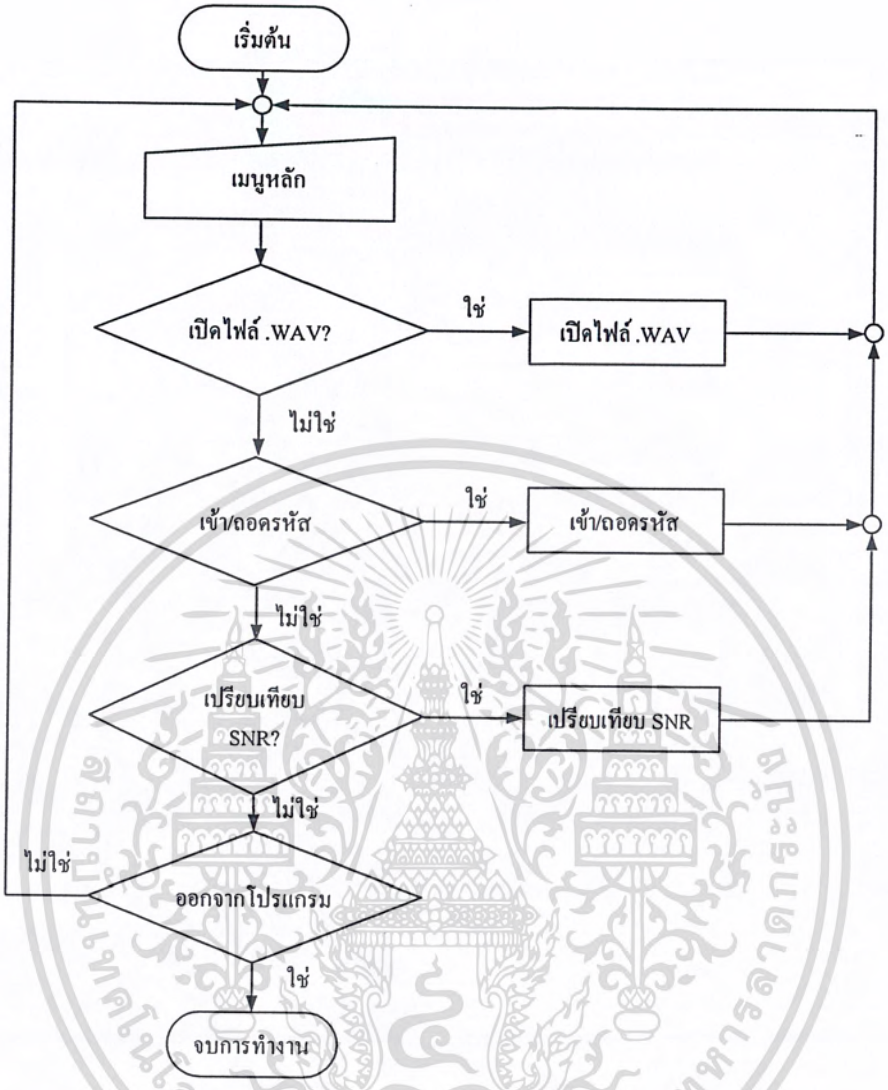
5) สามารถประยุกต์ใช้งานในการเข้ารหัสจริงได้ เพราะในปัจจุบันเทคโนโลยีที่ใช้งาน ยังต้องให้เวลาในการทำงานของการออกแบบ Codebook อยู่มาก

ภาคผนวก ก

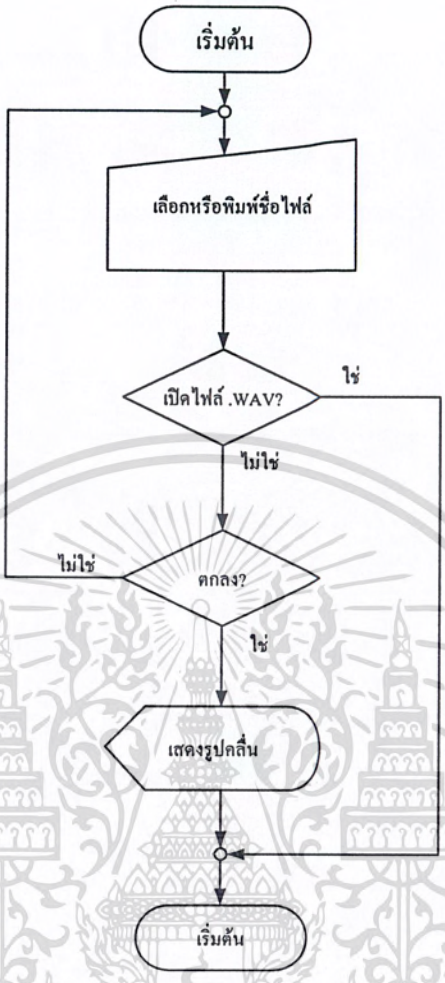
ผังงาน และโปรแกรม



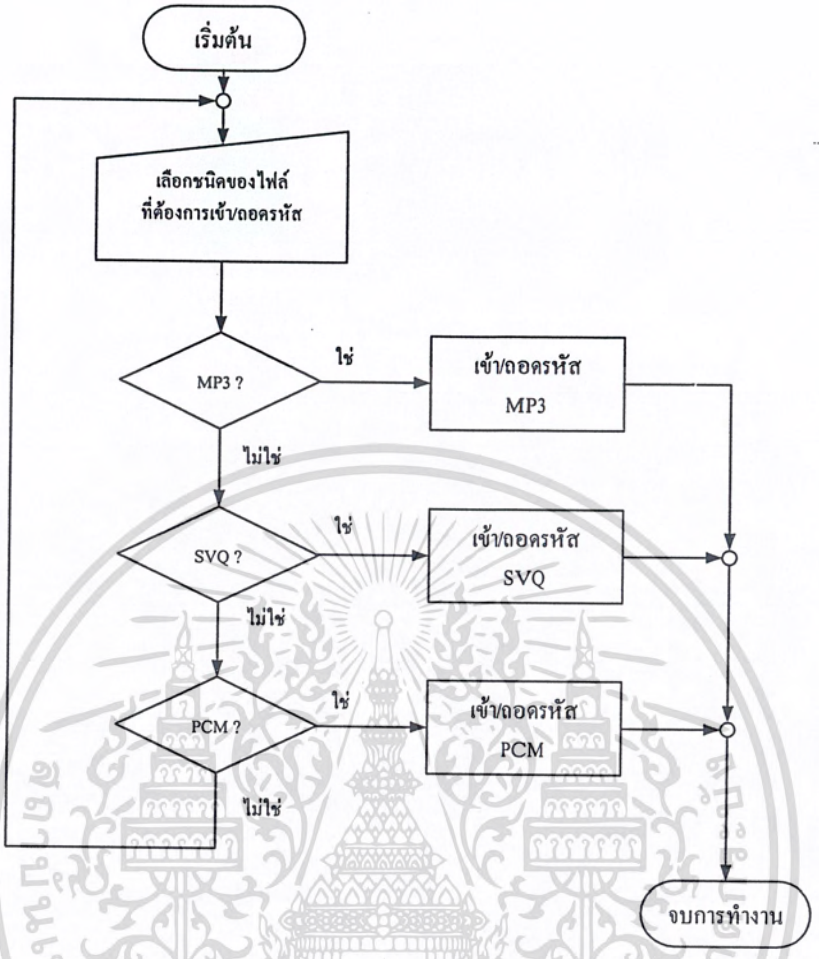
รูปที่ ก.1 โครงสร้างของโปรแกรม Case Study of Vector Quantize Efficiency



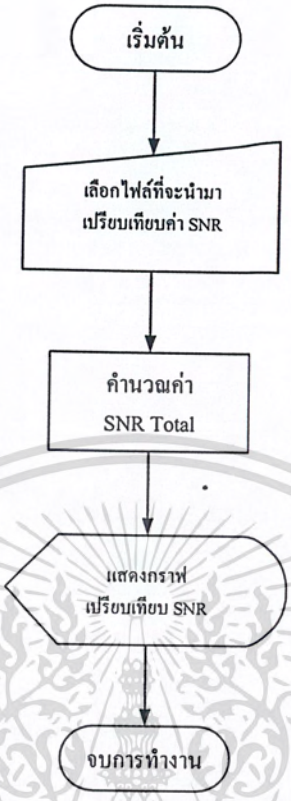
รูปที่ ก.2 ผังงานของโปรแกรม Case Study of Vector Quantize Efficiency



รูปที่ ก.3 ผังงาน ส่วนการเปิดไฟล์ .WAV



รูปที่ ก.4 ผังงาน ส่วนการเข้า และถอดรหัส



รูปที่ ก.5 ผังงาน ส่วนการเปรียบเทียบค่า SNR

```
//VectorQ.h : main header file for the VECTORQ application
//

#if !defined(AFX_VECTORQ_H__569A5BBA_7CAA_48C1_93C9_61A57FCDA1FA__INCLUDED_)
#define AFX_VECTORQ_H__569A5BBA_7CAA_48C1_93C9_61A57FCDA1FA__INCLUDED_

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

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

#include "resource.h"		// main symbols

/////////////////////////////////////////////////////////////////////////////
// CVectorQApp:
// See VectorQ.cpp for the implementation of this class
//

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

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

// Implementation
protected:

public:
	//{{AFX_MSG(CVectorQApp)
	afx_msg void OnAppAbout();
	afx_msg void OnSnrGraph();
	afx_msg void OnFileOpen();
	afx_msg void OnAppCal();
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};


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

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

#endif // !defined(AFX_VECTORQ_H__569A5BBA_7CAA_48C1_93C9_61A57FCDA1FA__INCLUDED_)

// VectorQ.cpp : Defines the class behaviors for the application.
//

#include "stdafx.h"
#include "VectorQ.h"

#include "MainFrm.h"
#include "ChildFrm.h"
#include "VectorQDoc.h"
#include "VectorQView.h"

#include "WavChar.h"
#include "SNR.h"
```

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

/////////////////////////////////////////////////////////////////////////////
// CVectorQApp

BEGIN_MESSAGE_MAP(CVectorQApp, CWinApp)
        //{{AFX_MSG_MAP(CVectorQApp)
        ON_COMMAND(ID_APP_ABOUT, OnAppAbout)
        ON_COMMAND(ID_SNR_GRAPH, OnSnrGraph)
        ON_COMMAND(ID_FILE_OPEN, OnFileOpen)
        ON_COMMAND(ID_APP_CAL, OnAppCal)
        //}}AFX_MSG_MAP
        // Standard file based document commands
        ON_COMMAND(ID_FILE_OPEN, CWinApp::OnFileOpen)
        // Standard print setup command
        ON_COMMAND(ID_FILE_PRINT_SETUP, CWinApp::OnFilePrintSetup)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CVectorQApp construction

CVectorQApp::CVectorQApp()
{
        // TODO: add construction code here,
        // Place all significant initialization in InitInstance
}

/////////////////////////////////////////////////////////////////////////////
// The one and only CVectorQApp object

CVectorQApp theApp;

/////////////////////////////////////////////////////////////////////////////
// CVectorQApp initialization

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

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

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

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

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

//      CMultiDocTemplate* pDocTemplate;
//      pDocTemplate = new CMultiDocTemplate(
//              IDR_VECTORTYPE,
//              RUNTIME_CLASS(CVectorQDoc),
```

```
//              RUNTIME_CLASS(CChildFrame), // custom MDI child frame
//              RUNTIME_CLASS(CVectorQView));
//      AddDocTemplate(pDocTemplate);

        // create main MDI Frame window
        CMainFrame* pMainFrame = new CMainFrame;
        if (!pMainFrame->LoadFrame(IDR_MAINFRAME))
                return FALSE;
        m_pMainWnd = pMainFrame;

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

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

        // The main window has been initialized, so show and update it.
        pMainFrame->ShowWindow(SW_SHOWMAXIMIZED);
        pMainFrame->UpdateWindow();

        return TRUE;
}


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

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

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

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

// Implementation
protected:
        //{{AFX_MSG(CAboutDlg)
                // No message handlers
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};

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

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

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
        //{{AFX_MSG_MAP(CAboutDlg)
```

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

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

/////////////////////////////////////////////////////////////////////////////
// CVectorQApp message handlers


void CVectorQApp::OnSnrGraph()
{
        // TODO: Add your command handler code here
        CWavChar WavSnr;

        if(WavSnr.SNRNum < 1){
                CSNR SNRDlg;
                if(SNRDlg.DoModal() == IDOK){
                        BOOL Err = TRUE;

                        int NeverSnr = WavSnr.SNR;
                        WavSnr.SNR = 0;

                        for(int i=2; i<5; i++) if(SNRDlg.GetPathName(i) != "")
WavSnr.SNR++;

                        if(WavSnr.SNR > 0){
                                WavSnr.SNR++;  // plus for Orginal file
                                for(int j=2; j<=WavSnr.SNR; j++){
                                        if(WavSnr.SetWavData(SNRDlg.GetPathName(j),
5+j)) Err = FALSE;
                                        else{
                                                Err = TRUE;
                                                MessageBox(NULL,"Some File doesn't WAVE
Format","Error",MB_OK|MB_ICONSTOP);
                                                break;
                                        }
                                }

                                if(!Err){
                                        WavSnr.IsSnr = TRUE;

                                        CMultiDocTemplate* pDocTemplate;
                                        pDocTemplate = new CMultiDocTemplate(
                                                IDR_VECTORTYPE,
                                                RUNTIME_CLASS(CVectorQDoc),
                                                RUNTIME_CLASS(CChildFrame), // custom MDI
child frame
                                                RUNTIME_CLASS(CVectorQView));
                                        AddDocTemplate(pDocTemplate);

                                        CWinApp::OnFileOpen();
                                }
                        }
                        else{
                                WavSnr.SNR = NeverSnr;
                                MessageBox(NULL,"Wave file less than 1
flie","Error",MB_OK|MB_ICONSTOP);
                        }
                }
        }
        else MessageBox(NULL,"Please, Close old SNR Graph","Error",MB_OK|MB_ICONSTOP);
```

```
}

void CVectorQApp::OnFileOpen()
{
        CWavChar Wav;

        if(Wav.WavNum < 5 ){
                Wav.IsSnr = FALSE;

                CMultiDocTemplate* pDocTemplate;
                pDocTemplate = new CMultiDocTemplate(
                        IDR_VECTORTYPE,
                        RUNTIME_CLASS(CVectorQDoc),
                        RUNTIME_CLASS(CChildFrame), // custom MDI child frame
                        RUNTIME_CLASS(CVectorQView));
                AddDocTemplate(pDocTemplate);

                CWinApp::OnFileOpen();
        }
        else MessageBox(NULL,"File doesn't WAVE Format","Error",MB_OK|MB_ICONSTOP);
}

void CVectorQApp::OnAppCal()
{
        // TODO: Add your command handler code here
        ShellExecute(NULL, "Open", "svqbitrate.txt", NULL, NULL, SW_SHOWNORMAL);

}


// ChildFrm.h : interface of the CChildFrame class
//
/////////////////////////////////////////////////////////////////////////////

#if !defined(AFX_CHILDFRM_H__2BFE1C18_1B79_4622_8725_8A255D0CB17D__INCLUDED_)
#define AFX_CHILDFRM_H__2BFE1C18_1B79_4622_8725_8A255D0CB17D__INCLUDED_

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


class CChildFrame : public CMDIChildWnd
{
        DECLARE_DYNCREATE(CChildFrame)
public:
        CChildFrame();

// Attributes
public:

// Operations
public:

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

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

```
// Generated message map functions
protected:
        //{{AFX_MSG(CChildFrame)
                // NOTE - the ClassWizard will add and remove member functions here.
                //    DO NOT EDIT what you see in these blocks of generated code!
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};

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

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

#endif // !defined(AFX_CHILDFRM_H__2BFE1C18_1B79_4622_8725_8A255D0CB17D__INCLUDED_)

// ChildFrm.cpp : implementation of the CChildFrame class
//

#include "stdafx.h"
#include "VectorQ.h"

#include "ChildFrm.h"

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

/////////////////////////////////////////////////////////////////////////////
// CChildFrame

IMPLEMENT_DYNCREATE(CChildFrame, CMDIChildWnd)

BEGIN_MESSAGE_MAP(CChildFrame, CMDIChildWnd)
        //{{AFX_MSG_MAP(CChildFrame)
                // NOTE - the ClassWizard will add and remove mapping macros here.
                //    DO NOT EDIT what you see in these blocks of generated code !
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CChildFrame construction/destruction

CChildFrame::CChildFrame()
{
        // TODO: add member initialization code here

}

CChildFrame::~CChildFrame()
{
}

BOOL CChildFrame::PreCreateWindow(CREATESTRUCT& cs)
{
        // TODO: Modify the Window class or styles here by modifying
        //  the CREATESTRUCT cs

        if( !CMDIChildWnd::PreCreateWindow(cs) )
                return FALSE;

        return TRUE;
}
```

```
/////////////////////////////////////////////////////////////////////////////
// CChildFrame diagnostics

#ifdef _DEBUG
void CChildFrame::AssertValid() const
{
        CMDIChildWnd::AssertValid();
}

void CChildFrame::Dump(CDumpContext& dc) const
{
        CMDIChildWnd::Dump(dc);
}

#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CChildFrame message handlers

#if !defined(AFX_CUTHEADER_H__0B45B4BB_690F_4805_91F5_947C7513006C__INCLUDED_)
#define AFX_CUTHEADER_H__0B45B4BB_690F_4805_91F5_947C7513006C__INCLUDED_

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

/////////////////////////////////////////////////////////////////////////////
// CCutHeader dialog

class CCutHeader : public CDialog
{
// Construction
public:
        CCutHeader(CWnd* pParent = NULL);   // standard constructor
        bool CutHeader(CString);
// Dialog Data
        //{{AFX_DATA(CCutHeader)
        enum { IDD = IDD_MP3_CH };
        CString m_CutHd;
        //}}AFX_DATA


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

// Implementation
protected:

        // Generated message map functions
        //{{AFX_MSG(CCutHeader)
        afx_msg void OnChangeEditCbf();
        afx_msg void OnBrowseCbf();
        virtual void OnOK();
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};

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

```
#endif // !defined(AFX_CUTHEADER_H__0B45B4BB_690F_4805_91F5_947C7513006C__INCLUDED_)
// CutHeader.cpp : implementation file
//

#include "stdafx.h"
#include "VectorQ.h"
#include "CutHeader.h"

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

/////////////////////////////////////////////////////////////////////////////
// CCutHeader dialog


CCutHeader::CCutHeader(CWnd* pParent /*=NULL*/)
        : CDialog(CCutHeader::IDD, pParent)
{
        //{{AFX_DATA_INIT(CCutHeader)
        m_CutHd = _T("");
        //}}AFX_DATA_INIT
}


void CCutHeader::DoDataExchange(CDataExchange* pDX)
{
        CDialog::DoDataExchange(pDX);
        //{{AFX_DATA_MAP(CCutHeader)
        DDX_Text(pDX, IDC_EDIT_CBF, m_CutHd);
        //}}AFX_DATA_MAP
}


BEGIN_MESSAGE_MAP(CCutHeader, CDialog)
        //{{AFX_MSG_MAP(CCutHeader)
        ON_EN_CHANGE(IDC_EDIT_CBF, OnChangeEditCbf)
        ON_BN_CLICKED(IDC_BROWSE_CBF, OnBrowseCbf)
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CCutHeader message handlers

void CCutHeader::OnChangeEditCbf()
{
        // TODO: Add your control notification handler code here
        m_CutHd = m_CutHd;
        UpdateData(FALSE);
}

void CCutHeader::OnBrowseCbf()
{
        // TODO: Add your control notification handler code here
        UpdateData(TRUE);
        CFileDialog fd(TRUE, NULL, NULL, OFN_HIDEREADONLY,"WAVE Files
(*.wav)|*.wav||");
        if(fd.DoModal() == IDOK){               // open file dialog
                if(fd.GetFileExt()!="wav"){
                        MessageBox("The lastname must be .wav
only","Error",MB_OK|MB_ICONEXCLAMATION);
                        m_CutHd = "";
                }
                else m_CutHd = fd.GetPathName();
        }
       UpdateData(FALSE);
```

```
}

void CCutHeader::OnOK()
{
        // TODO: Add extra validation here
        CutHeader(m_CutHd);
        CDialog::OnOK();
}

bool CCutHeader::CutHeader(CString fname)
{
        CStdioFile wf(fname,CFile::modeRead|CFile::typeBinary);     // open Wave file


        BYTE Header[58];
        BYTE* Data;
        LONG Size;
        wf.Read(Header,58);

        wf.Seek(2448,0);        // Cut Header

        Size = wf.GetLength()-2448;
        Data = new BYTE[Size];

        wf.Read(Data,Size);
        wf.Close();

        CStdioFile opf(fname,CFile::modeCreate|CFile::modeWrite|CFile::typeBinary);
        // create temp file

        opf.Write(Header,58);  // Write Header
        opf.Write(Data,Size);  // Write Data
/*
        WavSize = wf.GetLength()-58;  // 58 is Wave File Header
        BYTE *wfbuff;

        wfbuff = new BYTE[WavSize];
        wf.Read(wfbuff,WavSize);

        if(!BitRate){
                WavData = new int[WavSize];
                for(int a=0; a<WavSize; a++) WavData[a] = wfbuff[a];


        CStdioFile opf(fname1,CFile::modeCreate|CFile::modeWrite|CFile::typeBinary);
        // create temp file

        opf.Write(Header,58);  // Write Header

*/ return 1;
}

// MainFrm.h : interface of the CMainFrame class
//
/////////////////////////////////////////////////////////////////////////////

#if !defined(AFX_MAINFRM_H__CFF302BC_BE26_4C14_A0E6_C8CA36FCACEC__INCLUDED_)
#define AFX_MAINFRM_H__CFF302BC_BE26_4C14_A0E6_C8CA36FCACEC__INCLUDED_

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


class CMainFrame : public CMDIFrameWnd
{
        DECLARE_DYNAMIC(CMainFrame)
```

```
public:
        CMainFrame();

// Attributes
public:

// Operations
public:

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

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

protected:  // control bar embedded members
        CStatusBar  m_wndStatusBar;

// Generated message map functions
protected:
        //{{AFX_MSG(CMainFrame)
        afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
        afx_msg void OnEndecMp3();
        afx_msg void OnCbSvq();
        afx_msg void OnEndecSvq();
        afx_msg void OnEndecW2pcmA();
        afx_msg void OnEndecW2pcmU();
        afx_msg void OnEndecHmp3();
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};

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

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

#endif // !defined(AFX_MAINFRM_H__CFF302BC_BE26_4C14_A0E6_C8CA36FCACEC__INCLUDED_)

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

#include "stdafx.h"
#include "VectorQ.h"

#include "MainFrm.h"
#include "VEndec.h"
#include "VCodeBook.h"
#include "SNR.h"
#include "PcmAlaw.h"
#include "PcmUlaw.h"
#include "CutHeader.h"

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

```
/////////////////////////////////////////////////////////////////////////////
// CMainFrame

IMPLEMENT_DYNAMIC(CMainFrame, CMDIFrameWnd)

BEGIN_MESSAGE_MAP(CMainFrame, CMDIFrameWnd)
        //{{AFX_MSG_MAP(CMainFrame)
        ON_WM_CREATE()
        ON_COMMAND(ID_ENDEC_MP3, OnEndecMp3)
        ON_COMMAND(ID_CB_SVQ, OnCbSvq)
        ON_COMMAND(ID_ENDEC_SVQ, OnEndecSvq)
        ON_COMMAND(ID_ENDEC_W2PCM_A, OnEndecW2pcmA)
        ON_COMMAND(ID_ENDEC_W2PCM_U, OnEndecW2pcmU)
        ON_COMMAND(ID_ENDEC_HMP3, OnEndecHmp3)
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()

static UINT indicators[] =
{
        ID_SEPARATOR,           // status line indicator
        ID_INDICATOR_CAPS,
        ID_INDICATOR_NUM,
        ID_INDICATOR_SCRL,
};

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

CMainFrame::CMainFrame()
{
        // TODO: add member initialization code here

}

CMainFrame::~CMainFrame()
{
}

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

        if (!m_wndStatusBar.Create(this) ||
                !m_wndStatusBar.SetIndicators(indicators,
                  sizeof(indicators)/sizeof(UINT)))
        {
                TRACE0("Failed to create status bar\n");
                return -1;      // fail to create
        }

        return 0;
}

BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)
{
        if( !CMDIFrameWnd::PreCreateWindow(cs) )
                return FALSE;
        // TODO: Modify the Window class or styles here by modifying
        //  the CREATESTRUCT cs

        return TRUE;
}

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

#ifdef _DEBUG
```

```
void CMainFrame::AssertValid() const
{
        CMDIFrameWnd::AssertValid();
}

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

#endif //_DEBUG

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

void CMainFrame::OnEndecMp3()
{
        // TODO: Add your command handler code here
        CFileDialog fd(TRUE, NULL, NULL, OFN_HIDEREADONLY,"All Files (*.*)|*.*||");
        if(fd.DoModal() == IDOK) ShellExecute(NULL, "Open", fd.GetPathName(), NULL,
NULL, SW_SHOWNORMAL);
}

void CMainFrame::OnCbSvq()
{
        // TODO: Add your command handler code here
        CVCodebook VCBookDlg;
        VCBookDlg.DoModal();
}

void CMainFrame::OnEndecSvq()
{
        // TODO: Add your command handler code here
        CVEndec VEndecDlg;
        VEndecDlg.DoModal();
}

void CMainFrame::OnEndecW2pcmA()
{
        // TODO: Add your command handler code here
        CPcmAlaw PcmAlawDlg;
        PcmAlawDlg.DoModal();
}

void CMainFrame::OnEndecW2pcmU()
{
        // TODO: Add your command handler code here
        CPcmUlaw PcmUlawDlg;
        PcmUlawDlg.DoModal();
}

void CMainFrame::OnEndecHmp3()
{
        // TODO: Add your command handler code here
        CCutHeader CutHdDlg;
        CutHdDlg.DoModal();
}

// PcmAlaw.h : header file
//

#if !defined(AFX_PCMALAW_H__04904165_18F1_44B5_B764_1DC8CC6379C5__INCLUDED_)
#define AFX_PCMALAW_H__04904165_18F1_44B5_B764_1DC8CC6379C5__INCLUDED_

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

```
//////////////////////////////////////////////////////////////////////////
// CPcmAlaw dialog

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

// Dialog Data
        //{{AFX_DATA(CPcmAlaw)
        enum { IDD = IDD_PCM_ALAW };
        CString m_Inpf;
        CString m_Outf;
        //}}AFX_DATA


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

// Implementation
protected:

        LONG WavSize;
        BOOL BitRate;
        int* WavData;
        BYTE Header[58];

        bool MakeData(CString);
        bool AlawEnDec(CString, CString);
        unsigned char int2alaw(short);
        int alaw2int(unsigned char);

        // Generated message map functions
        //{{AFX_MSG(CPcmAlaw)
        afx_msg void OnBrowseIpf();
        afx_msg void OnBrowseOpf();
        virtual void OnOK();
        afx_msg void OnChangeEditIpf();
        afx_msg void OnChangeEditOpf();
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};

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

#endif // !defined(AFX_PCMALAW_H__04904165_18F1_44B5_B764_1DC8CC6379C5__INCLUDED_)


// PcmAlaw.cpp : implementation file
//

#include "stdafx.h"
#include "VectorQ.h"
#include "PcmAlaw.h"

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

```
////////////////////////////////////////////////////////////////////////////
// CPcmAlaw dialog


CPcmAlaw::CPcmAlaw(CWnd* pParent /*=NULL*/)
        : CDialog(CPcmAlaw::IDD, pParent)
{
        //{{AFX_DATA_INIT(CPcmAlaw)
        m_Inpf = _T("");
        m_Outf = _T("");
        //}}AFX_DATA_INIT
}


void CPcmAlaw::DoDataExchange(CDataExchange* pDX)
{
        CDialog::DoDataExchange(pDX);
        //{{AFX_DATA_MAP(CPcmAlaw)
        DDX_Text(pDX, IDC_EDIT_IPF, m_Inpf);
        DDX_Text(pDX, IDC_EDIT_OPF, m_Outf);
        //}}AFX_DATA_MAP
}


BEGIN_MESSAGE_MAP(CPcmAlaw, CDialog)
        //{{AFX_MSG_MAP(CPcmAlaw)
        ON_BN_CLICKED(IDC_BROWSE_IPF, OnBrowseIpf)
        ON_BN_CLICKED(IDC_BROWSE_OPF, OnBrowseOpf)
        ON_EN_CHANGE(IDC_EDIT_IPF, OnChangeEditIpf)
        ON_EN_CHANGE(IDC_EDIT_OPF, OnChangeEditOpf)
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()

////////////////////////////////////////////////////////////////////////////
// CPcmAlaw message handlers

void CPcmAlaw::OnBrowseIpf()
{
        // TODO: Add your control notification handler code here
        UpdateData(TRUE);
        CFileDialog fd(TRUE, NULL, NULL, OFN_HIDEREADONLY,"WAVE Files
(*.wav)|*.wav||");
        if(fd.DoModal() == IDOK){                       // open file dialog
                if(fd.GetFileExt()!="wav"){
                        MessageBox("The lastname must be .wav
only","Error",MB_OK|MB_ICONEXCLAMATION);
                        m_Inpf = "";
                }
                else m_Inpf = fd.GetPathName();
        }
        UpdateData(FALSE);

}

void CPcmAlaw::OnBrowseOpf()
{
        // TODO: Add your control notification handler code here
        UpdateData(TRUE);
        CFileDialog
fd(FALSE,"wav","Untitled.wav",OFN_HIDEREADONLY|OFN_OVERWRITEPROMPT,"WAVE Files
(*.wav)|*.wav||");
        if(fd.DoModal()==IDOK){                         // open saveas dialog
                if(fd.GetFileExt()!="wav"){
                        MessageBox("The lastname must be .wav
only","Error",MB_OK|MB_ICONEXCLAMATION);
                        m_Outf = "";
                }
                else m_Outf = fd.GetPathName();         // set text of saveas
```

```
        }
        UpdateData(FALSE);

}

unsigned char CPcmAlaw::int2alaw(short linear)
{
        char segment;
        unsigned char i, sign,quant;
        unsigned short output,absol, temp;
        temp=absol=abs(linear);
        sign=(linear>=0)?1:0;
        for(i=0;i<16;i++){
                output=temp&0x8000;
                if(output)break;
                temp<<=1;
        }
        segment=11-i;
        if(segment<=0){
                segment=0;
                quant=(absol>>1)&0x0F;
        }
        else
                quant=(absol>>segment)&0x0F;
        segment<<=4;
        output=segment+quant;
        if(absol>4095) output=0x7F;
        if(sign)
                return output^=0xD5;
        else
                return output^=0x55;
}

int CPcmAlaw::alaw2int(unsigned char log)
{
        unsigned char sign,segment;
        unsigned short temp,quant;

        temp=log^0xD5;
        sign=(temp&0x80)>>7;
        segment=(temp&0x70)>>4;
        quant=temp&0x0F;
        quant<<=1;
        if(!segment)
                quant+=1;
        else{
                quant+=33;
                quant<<=segment-1;
        }
        if(sign)
                return -quant;
        else
                return quant;
}

void CPcmAlaw::OnOK()
{
        // TODO: Add extra validation here
        if(AlawEnDec(m_Inpf, m_Outf)){
                MessageBox("Encode / Decode PCM A-law complete","Encode / Decode PCM A-
law",MB_OK);
                CDialog::OnOK();
        }
}

bool CPcmAlaw::MakeData(CString fname)
{
```

```
        CStdioFile wf(fname,CFile::modeRead|CFile::typeBinary);     // open Wave file


        wf.Seek(24,wf.GetPosition()); // Seek to bit rate
        BYTE bitbuff[5];

        wf.Read(bitbuff,5);    // Read bit rate
        if(bitbuff[0] == bitbuff[4])  BitRate  = FALSE;      //      8  bit
        else if(bitbuff[4] == bitbuff[0]*2) BitRate = TRUE; //      16 bit
        else {wf.Close(); return 0;}

        wf.SeekToBegin();      // Cut Header
        wf.Read(Header,58);

        WavSize = wf.GetLength()-58;  // 58 is Wave File Header
        BYTE *wfbuff;

        wfbuff = new BYTE[WavSize];
        wf.Read(wfbuff,WavSize);

        if(!BitRate){
                WavData = new int[WavSize];
                for(int a=0; a<WavSize; a++) WavData[a] = wfbuff[a];
        }
        else{
                WavSize /= 2;
                WavData = new int[WavSize];
                for(int b=0; b<WavSize; b++) WavData[b] = wfbuff[b*2] +
wfbuff[b*2+1]*256;
        }

        return TRUE;
}

bool CPcmAlaw::AlawEnDec(CString fname, CString fname1){

        unsigned char* data;
        BYTE buff[2];

        if(!MakeData(fname)){
                MessageBox("A Wave file format is
wrong","Error",MB_OK|MB_ICONEXCLAMATION);
        return FALSE;
        }

        CStdioFile opf(fname1,CFile::modeCreate|CFile::modeWrite|CFile::typeBinary);
        // create temp file

        opf.Write(Header,58); // Write Header

        data = new unsigned char[WavSize];

        for(int i=0; i<WavSize; i++){
                data[i] = int2alaw(WavData[i]);              // Encode
                WavData[i] = alaw2int(data[i]);              // Decode

                if(!BitRate){
                        buff[0] = (BYTE)WavData[i];
                        opf.Write(buff,1);
                }
                else{
                        buff[0] = (BYTE)(WavData[i]%256);
                        opf.Write(buff,1);
                        buff[1] = (BYTE)(WavData[i]/256);
                        opf.Write(buff+1,1);
                }
        }
```

```
        opf.Close();

        return TRUE;

}

void CPcmAlaw::OnChangeEditIpf()
{
        // TODO: Add your control notification handler code here
        m_Inpf = m_Inpf;
        UpdateData(FALSE);
}

void CPcmAlaw::OnChangeEditOpf()
{
        // TODO: Add your control notification handler code here
        m_Outf = m_Outf;
        UpdateData(FALSE);
}

// PcmUlaw.h : header file
//
#if !defined(AFX_PCMULAW_H__CF8C4A44_7E1A_4D93_8DC0_55944A0C44F6__INCLUDED_)
#define AFX_PCMULAW_H__CF8C4A44_7E1A_4D93_8DC0_55944A0C44F6__INCLUDED_

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

/////////////////////////////////////////////////////////////////////////////
// CPcmUlaw dialog

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

// Dialog Data
        //{{AFX_DATA(CPcmUlaw)
        enum { IDD = IDD_PCM_ULAW };
        CString m_Inpf;
        CString m_Outf;
        //}}AFX_DATA


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

// Implementation
protected:

        LONG WavSize;
        BOOL BitRate;
        int* WavData;
        BYTE Header[58];

        bool MakeData(CString);
        bool UlawEnDec(CString, CString);
        unsigned char int2ulaw(short);
        int ulaw2int(unsigned char);

        // Generated message map functions
        //{{AFX_MSG(CPcmUlaw)
```

```
        afx_msg void OnBrowseIpf();
        afx_msg void OnBrowseOpf();
        virtual void OnOK();
        afx_msg void OnChangeEditIpf();
        afx_msg void OnChangeEditOpf();
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};

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

#endif // !defined(AFX_PCMULAW_H__CF8C4A44_7E1A_4D93_8DC0_55944A0C44F6__INCLUDED_)

// pcmUlaw.cpp : implementation file
//

#include "stdafx.h"
#include "VectorQ.h"
#include "PcmUlaw.h"

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

/////////////////////////////////////////////////////////////////////////////
// CPcmUlaw dialog


CPcmUlaw::CPcmUlaw(CWnd* pParent /*=NULL*/)
        : CDialog(CPcmUlaw::IDD, pParent)
{
        //{{AFX_DATA_INIT(CPcmUlaw)
        m_Inpf = _T("");
        m_Outf = _T("");
        //}}AFX_DATA_INIT
}


void CPcmUlaw::DoDataExchange(CDataExchange* pDX)
{
        CDialog::DoDataExchange(pDX);
        //{{AFX_DATA_MAP(CPcmUlaw)
        DDX_Text(pDX, IDC_EDIT_IPF, m_Inpf);
        DDX_Text(pDX, IDC_EDIT_OPF, m_Outf);
        //}}AFX_DATA_MAP
}


BEGIN_MESSAGE_MAP(CPcmUlaw, CDialog)
        //{{AFX_MSG_MAP(CPcmUlaw)
        ON_BN_CLICKED(IDC_BROWSE_IPF, OnBrowseIpf)
        ON_BN_CLICKED(IDC_BROWSE_OPF, OnBrowseOpf)
        ON_EN_CHANGE(IDC_EDIT_IPF, OnChangeEditIpf)
        ON_EN_CHANGE(IDC_EDIT_OPF, OnChangeEditOpf)
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CPcmUlaw message handlers


void CPcmUlaw::OnBrowseIpf()
{
        // TODO: Add your control notification handler code here
```

```
        UpdateData(TRUE);
        CFileDialog fd(TRUE, NULL, NULL, OFN_HIDEREADONLY,"WAVE Files
(*.wav)|*.wav||");
        if(fd.DoModal() == IDOK){              // open file dialog
                if(fd.GetFileExt()!="wav"){
                        MessageBox("The lastname must be .wav
only","Error",MB_OK|MB_ICONEXCLAMATION);
                        m_Inpf = "";
                }
                else m_Inpf = fd.GetPathName();
        }
        UpdateData(FALSE);

}

void CPcmUlaw::OnBrowseOpf()
{
        // TODO: Add your control notification handler code here
        UpdateData(TRUE);
        CFileDialog
fd(FALSE,"wav","Untitled.wav",OFN_HIDEREADONLY|OFN_OVERWRITEPROMPT,"WAVE Files
(*.wav)|*.wav||");
        if(fd.DoModal()==IDOK){                // open saveas dialog
                if(fd.GetFileExt()!="wav"){
                        MessageBox("The lastname must be .wav
only","Error",MB_OK|MB_ICONEXCLAMATION);
                        m_Outf = "";
                }
                else m_Outf = fd.GetPathName();      // set text of saveas
        }
        UpdateData(FALSE);

}

unsigned char CPcmUlaw::int2ulaw(short linear)
{
        unsigned char i, sign, segment, quant;
        unsigned short output, absol, temp;

        absol=abs(linear)+33;
        temp=absol;
        sign=(linear >= 0)?1:0;
        for(i=0; i<16; i++){
                output=temp&0x8000;
                if(output) break;
                temp<<=1;
        }
        segment=11-i;
        quant=(absol>>segment)&0x0F;
        segment--;
        segment<<=4;
        output=segment+quant;
        if(absol>8191) output=0x7F;
        if(sign)
                return output^=0xFF;
        else
                return output^=0x7F;
}

int CPcmUlaw::ulaw2int(unsigned char log)
{
        unsigned char sign, segment;
        unsigned short temp, quant;

        temp=log^0xFF;
        sign=(temp&0x80)>>7;
        segment=(temp&0x70)>>4;
        quant=temp&0x0F;
```

```
        quant<<=1;
        quant+=33;
        quant<<=segment;
        if(sign)
                return 33-quant;
        else
                return quant-33;
}

void CPcmUlaw::OnOK()
{
        // TODO: Add extra validation here
        if(UlawEnDec(m_Inpf, m_Outf)){
                MessageBox("Encode / Decode PCM u-law complete","Encode / Decode PCM u-
law",MB_OK);
                CDialog::OnOK();
        }
}

bool CPcmUlaw::MakeData(CString fname)
{
        CStdioFile wf(fname,CFile::modeRead|CFile::typeBinary);     // open Wave file


        wf.Seek(24,wf.GetPosition()); // Seek to bit rate
        BYTE bitbuff[5];

        wf.Read(bitbuff,5);    // Read bit rate
        if(bitbuff[0] == bitbuff[4])  BitRate  = FALSE;      //      8  bit
        else if(bitbuff[4] == bitbuff[0]*2) BitRate = TRUE; //      16 bit
        else {wf.Close(); return 0;}

        wf.SeekToBegin();       // Cut Header
        wf.Read(Header,58);

        WavSize = wf.GetLength()-58;  // 58 is Wave File Header
        BYTE *wfbuff;

        wfbuff = new BYTE[WavSize];
        wf.Read(wfbuff,WavSize);

        if(!BitRate){
                WavData = new int[WavSize];
                for(int a=0; a<WavSize; a++) WavData[a] = wfbuff[a];
        }
        else{
                WavSize /= 2;
                WavData = new int[WavSize];
                for(int b=0; b<WavSize; b++) WavData[b] = wfbuff[b*2] +
wfbuff[b*2+1]*256;
        }

        return TRUE;
}

bool CPcmUlaw::UlawEnDec(CString fname, CString fname1){

        unsigned char* data;
        BYTE buff[2];

        if(!MakeData(fname)){
                MessageBox("A Wave file format is
wrong","Error",MB_OK|MB_ICONEXCLAMATION);
        return FALSE;
        }

        CStdioFile opf(fname1,CFile::modeCreate|CFile::modeWrite|CFile::typeBinary);
        // create temp file
```

```
        opf.Write(Header,58); // Write Header

        data = new unsigned char[WavSize];

        for(int i=0; i<WavSize; i++){
                data[i] = int2ulaw(WavData[i]);              // Encode
                WavData[i] = ulaw2int(data[i]);              // Decode

                if(!BitRate){
                        buff[0] = (BYTE)WavData[i];
                        opf.Write(buff,1);
                }
                else{
                        buff[0] = (BYTE)(WavData[i]%256);
                        opf.Write(buff,1);
                        buff[1] = (BYTE)(WavData[i]/256);
                        opf.Write(buff+1,1);
                }
        }

        opf.Close();

        return TRUE;

}

void CPcmUlaw::OnChangeEditIpf()
{
        // TODO: Add your control notification handler code here
        m_Inpf = m_Inpf;
        UpdateData(FALSE);
}

void CPcmUlaw::OnChangeEditOpf()
{
        // TODO: Add your control notification handler code here
        m_Outf = m_Outf;
        UpdateData(FALSE);
}



// SNR.h : header file
//
#if !defined(AFX_SNR_H__306B8AA5_6CC0_41E8_8205_7D10CDB9C3CF__INCLUDED_)
#define AFX_SNR_H__306B8AA5_6CC0_41E8_8205_7D10CDB9C3CF__INCLUDED_

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

/////////////////////////////////////////////////////////////////////////////
// CSNR dialog

class CSNR : public CDialog
{
// Construction
public:
        CSNR(CWnd* pParent = NULL);   // standard constructor
        CString GetPathName(int);
// Dialog Data
        //{{AFX_DATA(CSNR)
        enum { IDD = IDD_SNR };
        CString m_Snr2;
        CString m_Snr3;
        CString m_Snr4;
        //}}AFX_DATA
```

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

// Implementation
protected:

        // Generated message map functions
        //{{AFX_MSG(CSNR)
        afx_msg void OnBrowseWf2();
        afx_msg void OnBrowseWf3();
        afx_msg void OnBrowseWf4();
        virtual void OnOK();
        afx_msg void OnChangeEditWf2();
        afx_msg void OnChangeEditWf3();
        afx_msg void OnChangeEditWf4();
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};

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

#endif // !defined(AFX_SNR_H__306B8AA5_6CC0_41E8_8205_7D10CDB9C3CF__INCLUDED_)

// SNR.cpp : implementation file
//

#include "stdafx.h"
#include "VectorQ.h"
#include "SNR.h"
#include "WavChar.h"

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

/////////////////////////////////////////////////////////////////////////////
// CSNR dialog


CSNR::CSNR(CWnd* pParent /*=NULL*/)
        : CDialog(CSNR::IDD, pParent)
{
        //{{AFX_DATA_INIT(CSNR)
        m_Snr2 = _T("");
        m_Snr3 = _T("");
        m_Snr4 = _T("");
        //}}AFX_DATA_INIT
}


void CSNR::DoDataExchange(CDataExchange* pDX)
{
        CDialog::DoDataExchange(pDX);
        //{{AFX_DATA_MAP(CSNR)
        DDX_Text(pDX, IDC_EDIT_WF2, m_Snr2);
        DDX_Text(pDX, IDC_EDIT_WF3, m_Snr3);
        DDX_Text(pDX, IDC_EDIT_WF4, m_Snr4);
        //}}AFX_DATA_MAP
```

```
}


BEGIN_MESSAGE_MAP(CSNR, CDialog)
        //{{AFX_MSG_MAP(CSNR)
        ON_BN_CLICKED(IDC_BROWSE_WF2, OnBrowseWf2)
        ON_BN_CLICKED(IDC_BROWSE_WF3, OnBrowseWf3)
        ON_BN_CLICKED(IDC_BROWSE_WF4, OnBrowseWf4)
        ON_EN_CHANGE(IDC_EDIT_WF2, OnChangeEditWf2)
        ON_EN_CHANGE(IDC_EDIT_WF3, OnChangeEditWf3)
        ON_EN_CHANGE(IDC_EDIT_WF4, OnChangeEditWf4)
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CSNR message handlers

CString CSNR::GetPathName(int num)
{
        switch(num){
        case 2 : return m_Snr2;
        case 3 : return m_Snr3;
        case 4 : return m_Snr4;
        }
        return "Error";
}

void CSNR::OnBrowseWf2()
{
        // TODO: Add your control notification handler code here
        UpdateData(TRUE);
        CFileDialog fd(TRUE, NULL, NULL, OFN_HIDEREADONLY,"WAVE Files
(*.wav)|*.wav||");
        if(fd.DoModal() == IDOK){                    // open file dialog
                if(fd.GetFileExt()!="wav"){
                        MessageBox("The lastname must be .wav
only","Error",MB_OK|MB_ICONEXCLAMATION);
                        m_Snr2 = "";
                }
                else m_Snr2 = fd.GetPathName();
        }
        UpdateData(FALSE);

}

void CSNR::OnBrowseWf3()
{
        // TODO: Add your control notification handler code here
        UpdateData(TRUE);
        CFileDialog fd(TRUE, NULL, NULL, OFN_HIDEREADONLY,"WAVE Files
(*.wav)|*.wav||");
        if(fd.DoModal() == IDOK){                    // open file dialog
                if(fd.GetFileExt()!="wav"){
                        MessageBox("The lastname must be .wav
only","Error",MB_OK|MB_ICONEXCLAMATION);
                        m_Snr3 = "";
                }
                else m_Snr3 = fd.GetPathName();
        }
        UpdateData(FALSE);

}

void CSNR::OnBrowseWf4()
{
        // TODO: Add your control notification handler code here
        UpdateData(TRUE);
```

```
        CFileDialog fd(TRUE, NULL, NULL, OFN_HIDEREADONLY,"WAVE Files
(*.wav)|*.wav||");
        if(fd.DoModal() == IDOK){              // open file dialog
                if(fd.GetFileExt()!="wav"){
                        MessageBox("The lastname must be .wav
only","Error",MB_OK|MB_ICONEXCLAMATION);
                        m_Snr4 = "";
                }
                else m_Snr4 = fd.GetPathName();
        }
        UpdateData(FALSE);

}

void CSNR::OnOK()
{
        // TODO: Add extra validation here

        CDialog::OnOK();

}

void CSNR::OnChangeEditWf2()
{
        // TODO: Add your control notification handler code here
        m_Snr2 = m_Snr2;
        UpdateData(FALSE);

}

void CSNR::OnChangeEditWf3()
{
        // TODO: Add your control notification handler code here
        m_Snr3 = m_Snr3;
        UpdateData(FALSE);

}

void CSNR::OnChangeEditWf4()
{
        // TODO: Add your control notification handler code here
        m_Snr4 = m_Snr4;
        UpdateData(FALSE);

}

// VCodebook.h : header file
//
#if !defined(AFX_VCODEBOOK_H__D5AD9425_5160_44D1_B6B9_B8CE132F7B7F__INCLUDED_)
#define AFX_VCODEBOOK_H__D5AD9425_5160_44D1_B6B9_B8CE132F7B7F__INCLUDED_

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

/////////////////////////////////////////////////////////////////////////////
// CVCodebook dialog

class CVCodebook : public CDialog
{
// Construction
public:
        CVCodebook(CWnd* pParent = NULL);   // standard constructor
        bool MakeCodeBook();
        void Spliting(int);
        bool Partition(int);
        bool MakeData(CString);
```

```
protected:
        virtual BOOL OnInitDialog();
        void InitProgress();
// Dialog Data
        //{{AFX_DATA(CVCodebook)
        enum { IDD = IDD_VQ_CB };
        int             m_Dimension;
        int             m_Size;
        float   m_Splitting;
        float   m_Theshould;
        CString m_WavFile;
        CString m_CBFile;
        CProgressCtrl m_progress;     // object of progress control
        CString m_ProgStr;
        int m_IT;
        int m_CDB;
        double m_ERR;
        int m_ER;

        CString m_Bitrate;
        CComboBox       m_Cbitrate;
                // NOTE: the ClassWizard will add data members here
        //}}AFX_DATA

// Code Book Data
private:
        CString fname, fname1;
        int CodebookSize, CodebookDimension, Frame;
        double Stepsize, Theshould;
        double Error, ErrorOld;
        double Codebook[10][512], Newbook[10][512], New[10];
        long *data;
        int bit;

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

// Implementation
protected:

        // Generated message map functions
        //{{AFX_MSG(CVCodebook)
        afx_msg void OnBrowseWF();
        afx_msg void OnBrowseCBF();
        afx_msg void OnChangeEditDM();
        afx_msg void OnChangeEditSZ();
        afx_msg void OnChangeEditSP();
        afx_msg void OnChangeEditTH();
        afx_msg void OnChangeEditWF();
        afx_msg void OnChangeEditCBF();
        virtual void OnOK();
        afx_msg void OnKillfocusEditDM();
        afx_msg void OnKillfocusEditSz();
        afx_msg void OnKillfocusEditSp();
        afx_msg void OnKillfocusEditTh();


//      afx_msg void OnCloseupCombo1();
        afx_msg void OnSelchangeCombo1();
//      afx_msg void OnSelendokCombo1();

        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};
```

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

#endif // !defined(AFX_VCODEBOOK_H__D5AD9425_5160_44D1_B6B9_B8CE132F7B7F__INCLUDED_)

// VCodebook.cpp : implementation file
//

#include "stdafx.h"
#include "VectorQ.h"
#include "VCodebook.h"

#include "fcntl.h"
#include "io.h"
#include "math.h"

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

/////////////////////////////////////////////////////////////////////////////
// CVCodebook dialog


CVCodebook::CVCodebook(CWnd* pParent /*=NULL*/)
        : CDialog(CVCodebook::IDD, pParent)
{
        //{{AFX_DATA_INIT(CVCodebook)
        m_Dimension = 1;
        m_Size = 1;
        m_Splitting = 0.1f;
        m_Theshould = 0.0001f;
        m_WavFile = _T("");
        m_CBFile = _T("");
        m_ProgStr = _T("Ready ...");
        m_IT = 0;
        m_CDB = 0;
        m_ERR = 0.0;
        m_ER = 0;

        m_Bitrate = _T("");
        //}}AFX_DATA_INIT

        ErrorOld = 9.9E+6;
}


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

        InitProgress();                         // initial progress

        return TRUE;
}

void CVCodebook::InitProgress(){
        m_progress.SetRange(0,100);
        m_progress.SetStep(1);
        m_progress.SetPos(0);
}

void CVCodebook::DoDataExchange(CDataExchange* pDX)
{
        CDialog::DoDataExchange(pDX);
```

```
        //{{AFX_DATA_MAP(CVCodebook)
        DDX_Text(pDX, IDC_EDIT_DM, m_Dimension);
        DDX_Text(pDX, IDC_EDIT_SZ, m_Size);
        DDX_Text(pDX, IDC_EDIT_SP, m_Splitting);
        DDX_Text(pDX, IDC_EDIT_TH, m_Theshould);
        DDX_Text(pDX, IDC_EDIT_WF, m_WavFile);
        DDX_Text(pDX, IDC_EDIT_CBF, m_CBFile);
        DDX_Control(pDX,IDC_PROGRESS,m_progress);
        DDX_Text(pDX, IDC_STATIC_PG1, m_ProgStr);
        DDX_Text(pDX, IDC_STATIC_IT2, m_IT);
        DDX_Text(pDX, IDC_STATIC_CDB2, m_CDB);
        DDX_Text(pDX, IDC_STATIC_ERR2, m_ERR);
        DDX_Text(pDX, IDC_STATIC_ER2, m_ER);

        DDX_CBString(pDX, IDC_COMBO1, m_Bitrate);
        DDX_Control(pDX, IDC_COMBO1, m_Cbitrate);
        //}}AFX_DATA_MAP
}


BEGIN_MESSAGE_MAP(CVCodebook, CDialog)
        //{{AFX_MSG_MAP(CVCodebook)
        ON_BN_CLICKED(IDC_BROWSE_WF, OnBrowseWF)
        ON_EN_CHANGE(IDC_EDIT_WF, OnChangeEditWF)
        ON_EN_CHANGE(IDC_EDIT_CBF, OnChangeEditCBF)
        ON_EN_KILLFOCUS(IDC_EDIT_DM, OnKillfocusEditDM)
        ON_EN_KILLFOCUS(IDC_EDIT_SZ, OnKillfocusEditSz)
        ON_EN_KILLFOCUS(IDC_EDIT_SP, OnKillfocusEditSp)
        ON_EN_KILLFOCUS(IDC_EDIT_TH, OnKillfocusEditTh)
        ON_BN_CLICKED(IDC_BROWSE_CBF, OnBrowseCBF)

//      ON_CBN_CLOSEUP(IDC_COMBO1, OnCloseupCombo1)
        ON_CBN_SELCHANGE(IDC_COMBO1, OnSelchangeCombo1)
//      ON_CBN_SELENDOK(IDC_COMBO1, OnSelendokCombo1)
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CVCodebook message handlers

void CVCodebook::OnBrowseWF()
{
        // TODO: Add your control notification handler code here
        UpdateData(TRUE);
        CFileDialog fd(TRUE, NULL, NULL, OFN_HIDEREADONLY,"WAVE Files
(*.wav)|*.wav||");
        if(fd.DoModal() == IDOK){                    // open file dialog
                if(fd.GetFileExt()!="wav"){
                        MessageBox("The lastname must be .wav
only","Error",MB_OK|MB_ICONEXCLAMATION);
                        m_WavFile = "";
                }
                else m_WavFile = fd.GetPathName();
        }
        UpdateData(FALSE);
}

void CVCodebook::OnBrowseCBF()
{
        // TODO: Add your control notification handler code here
        UpdateData(TRUE);
        CFileDialog
fd(FALSE,"cdb","Untitled.cdb",OFN_HIDEREADONLY|OFN_OVERWRITEPROMPT,"Code Book file
(*.cdb)|*.cdb||");
        if(fd.DoModal()==IDOK){                  // open saveas dialog
                if(fd.GetFileExt()!="cdb"){
                        MessageBox("The lastname must be .cdb
only","Error",MB_OK|MB_ICONEXCLAMATION);
```

```
                    m_CBFile = "";
                }
                else m_CBFile = fd.GetPathName();     // set text of saveas
        }
        UpdateData(FALSE);
}

void CVCodebook::OnChangeEditWF()
{
        // TODO: Add your control notification handler code here
        m_WavFile = m_WavFile;
        UpdateData(FALSE);
}

void CVCodebook::OnChangeEditCBF()
{
        // TODO: Add your control notification handler code here
        m_CBFile = m_CBFile;
        UpdateData(FALSE);
}

void CVCodebook::OnChangeEditDM()
{
        // TODO: Add your control notification handler code here
        m_Dimension = m_Dimension;
        UpdateData(FALSE);
}

void CVCodebook::OnChangeEditSP()
{
        // TODO: Add your control notification handler code here
        m_Splitting = m_Splitting;
        UpdateData(FALSE);
}

void CVCodebook::OnChangeEditSZ()
{
        // TODO: Add your control notification handler code here
        m_Size = m_Size;
        UpdateData(FALSE);
}

void CVCodebook::OnChangeEditTH()
{
        // TODO: Add your control notification handler code here
        m_Theshould = m_Theshould;
        UpdateData(FALSE);
}

void CVCodebook::OnKillfocusEditDM()
{
        // TODO: Add your control notification handler code here
        if(!UpdateData(TRUE)) m_Dimension = 1;
        else if (m_Dimension < 1 || m_Dimension > 10){
                MessageBox("Please, key value between 1 -
10","Error",MB_OK|MB_ICONEXCLAMATION);
                m_Dimension = 1;
        }
        UpdateData(FALSE);
}

void CVCodebook::OnKillfocusEditSz()
{
        // TODO: Add your control notification handler code here
        if(!UpdateData(TRUE)) m_Size = 1;
        else if (m_Size < 1 || m_Size > 512){
                MessageBox("Please, key value between 1 -
512","Error",MB_OK|MB_ICONEXCLAMATION);
```

```
                m_Size = 1;
        }
        UpdateData(FALSE);
}

void CVCodebook::OnKillfocusEditSp()
{
        // TODO: Add your control notification handler code here
        if(!UpdateData(TRUE)) m_Splitting = 0.1f;
        else if (m_Splitting < 0.1f || m_Splitting > 100.0f){
                MessageBox("Please, key value between 0.1 -
100.0","Error",MB_OK|MB_ICONEXCLAMATION);
                m_Splitting = 0.1f;
        }
        UpdateData(FALSE);
}

void CVCodebook::OnKillfocusEditTh()
{
        // TODO: Add your control notification handler code here
        if(!UpdateData(TRUE)) m_Theshould = 0.0001f;
        else if (m_Theshould < 0.00001f || m_Theshould > 1.0f){
                MessageBox("Please, key value between 0.00001 -
1.0","Error",MB_OK|MB_ICONEXCLAMATION);
                m_Theshould = 0.0001f;
        }
        UpdateData(FALSE);
}

void CVCodebook::OnOK()
{
        // TODO: Add extra validation here

        fname = m_WavFile;
        fname1 = m_CBFile;
        CodebookDimension = m_Dimension;
        CodebookSize = m_Size;
        Stepsize = m_Splitting;
        Theshould = m_Theshould;

        if(MakeCodeBook()){
                MessageBox("SVQ Code book complete","Make SVQ Code Book",MB_OK);
                CDialog::OnOK();
        }
}

bool CVCodebook::MakeData(CString fname)
{
        CStdioFile wf(fname,CFile::modeRead|CFile::typeBinary);     // open Wave file


        wf.Seek(24,wf.GetPosition()); // Seek to bit rate
        BYTE bitbuff[5];

        wf.Read(bitbuff,5);    // Read bit rate
        if(bitbuff[0] == bitbuff[4])  bit  = 8;       //      8  bit
        else if(bitbuff[4] == bitbuff[0]*2) bit = 16;        //      16 bit
        else {wf.Close(); return 0;}

        wf.SeekToBegin();      // Cut Header
        wf.Seek(58,wf.GetPosition());

        long size = wf.GetLength()-58;        // 58 is Wave File Header
        BYTE *wfbuff;

        wfbuff = new BYTE[size];
        wf.Read(wfbuff,size);
```

```
        if(bit == 8){
                data = new long[size];
                for(int a=0; a<size; a++){
                        if(wfbuff[a] <= 127) data[a] = wfbuff[a];
                        else data[a] = wfbuff[a] - 256;
                }
        }
        else if(bit == 16){
                data = new long[size/2];
                for(int b=0; b<size/2; b++){
                        if((wfbuff[b*2] + wfbuff[b*2+1]*256) <= 32767) data[b] =
wfbuff[b*2] + wfbuff[b*2+1]*256;
                        else data[b] = (wfbuff[b*2] + wfbuff[b*2+1]*256) - 65536;
                }
        }
        else return FALSE;

        if(bit == 8) Frame = size/(CodebookDimension);
    else if(bit == 16) Frame = size/(CodebookDimension*2);

        return TRUE;
}


bool CVCodebook::MakeCodeBook()
{
        FILE *record_file;
        int  a, b, L;
        long pt = 0;

        UpdateData(TRUE);
        m_ProgStr = "Running    ";
        UpdateData(FALSE);

        if(!MakeData(fname)){
                MessageBox("A Wave file format is
wrong","Error",MB_OK|MB_ICONEXCLAMATION);
         return FALSE;
        }

        m_progress.SetRange(0,Frame);

    for (b=0; b < CodebookDimension; b++)    // Initial Array 'New'
        New[b]=0.0;

    for (a=0; a<Frame; a++)
    {
                pt = (long)(a*CodebookDimension);

                for (b=0; b<CodebookDimension; b++)
                        New[b] += (float)data[pt+b];

    }
    for (b=0; b<CodebookDimension; b++)
        Codebook[b][0]=(float)(New[b]/Frame);


       L=1;

    while (L<=CodebookSize/2)
    {
        Spliting(L);
        L*=2;
        ErrorOld=9E+15;
        Error=1.0;
        a=0;

                while (Theshould < Error)
```

```
        {
                        UpdateData(TRUE);
                        m_IT = a++;
                        m_CDB = L;
                        m_ERR = Error;
                        UpdateData(FALSE);
            if (!Partition(L)) return FALSE;
        }
    }

    if( m_CBFile == "" || (record_file = fopen(fname1,"wb")) == NULL)
    {
                MessageBox("Can not write output
file","Error",MB_OK|MB_ICONEXCLAMATION);
        return FALSE;
    }                                                    // SYNCHRONIZE TO DSP

    fprintf (record_file, "Codebook-dimension\t%d\n", CodebookDimension);
    fprintf (record_file, "Codebook-size\t\t%d\n", CodebookSize);

    for (a=0; a<CodebookSize; a++)
    {
        for (b=0; b<CodebookDimension; b++)
        {
            fprintf (record_file, "%0.02f ", Codebook[b][a]);
        }
        fprintf (record_file, "\n");
    }
    fclose (record_file);

       return TRUE;
}


void CVCodebook::Spliting (int Index)
{
    int a, b, c;
    c=0;
    for (a=0; a<Index; a++)
    {
        for (b=0; b<CodebookDimension; b++)
            Newbook[b][c]=(Codebook[b][a]+Stepsize);
        c++;
        for (b=0; b<CodebookDimension; b++)
            Newbook[b][c]=(Codebook[b][a]-Stepsize);
        c++;
    }
    for (a=0; a<c; a++)
        for (b=0; b<CodebookDimension; b++)
            Codebook[b][a]=Newbook[b][a];
}


bool CVCodebook::Partition (int Count)
{
    int a, b, c, x;
       long pt = 0;
       int C[512];

    double x1, x2, res1, res2, s;

    Error=0;

    for (a=0; a<512; a++){
                C[a]=0;
        for (b=0; b<10; b++)
            Newbook[b][a]=0;
       }
```

```
for (c=0; c<Frame; c++)
{

        if (c != 0 && c % 500 == 0){              // Show progress

                    UpdateData(TRUE);

                    if(m_ProgStr == "Running    ") m_ProgStr = "Running .  ";
                    else if(m_ProgStr == "Running .  ") m_ProgStr = "Running .. ";
                    else if(m_ProgStr == "Running .. ") m_ProgStr = "Running ...";
                    else m_ProgStr = "Running    ";

                    UpdateData(FALSE);
        }

    m_progress.StepIt();


           pt = (long)(c*CodebookDimension);

    res2=9.9E+15;
    for (a=0; a<Count; a++)
    {
        res1=0;
        for (b=0; b<CodebookDimension; b++)
        {
            x1=(float)data[pt+b];
            x2=Codebook[b][a];
            res1+=(x1-x2)*(x1-x2);
        }

        res1=sqrt (res1);

        if  (res1<res2)
        {
            for (b=0; b<CodebookDimension; b++)
            {
               x1=(float)data[pt+b];
               New[b]=x1;
            }
            res2=res1;
            x=a;
        }
    }

    Error+=res2;

    for (b=0; b<CodebookDimension; b++)
        Newbook[b][x]+=New[b];
    C[x]++;

}

for (a=0; a<Count; a++)
{
    if  (C[a]!=0)
    {
        for (b=0; b<CodebookDimension; b++)
        {
            x1=(float)C[a];
            x2=Newbook[b][a];
            x2/=x1;
            Newbook[b][a]=x2;
            Codebook[b][a]=Newbook[b][a];
        }
    }
    else{
```

```
                        UpdateData(TRUE);
                        m_ER = a;
                        UpdateData(FALSE);
                }
    }

    s=ErrorOld-Error;
    if  (s!=0)
    {
        s=fabs (s);
        ErrorOld=Error;
        Error=s/Error;
    }
    else
        Error=ErrorOld=0;

        return TRUE;
}

void CVCodebook::OnSelchangeCombo1()
{
        // TODO: Add your control notification handler code here
        UpdateData(TRUE);
        int x = m_Cbitrate.GetCurSel();
        if(x == 0) MessageBox("64");
        else if(x == 1) MessageBox("128");
        else if(x == 2) MessageBox("192");
        UpdateData(FALSE);
}

// VectorQView.h : interface of the CVectorQView class
//
/////////////////////////////////////////////////////////////////////////////

#if !defined(AFX_VECTORQVIEW_H__038FCF9C_249C_4122_879F_6C7921B31F3D__INCLUDED_)
#define AFX_VECTORQVIEW_H__038FCF9C_249C_4122_879F_6C7921B31F3D__INCLUDED_

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

#include "WavChar.h"

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

// Attributes
public:
        CVectorQDoc* GetDocument();

        CWavChar Wchar;                 // Create Wave Characterisitc

private:
        int Zoom;
        int IncDec;
        int Scale;
        CPoint Border;
        BOOL WavStyle;
        ULONG Shift;
        COLORREF bgColor;
        COLORREF bdColor;
        COLORREF gdColor;
        COLORREF ttColor;
        COLORREF w1Color;
        COLORREF w2Color;
```

```
        COLORREF w3Color;
        COLORREF w4Color;

        int WNum;
        BOOL Snr;
        BOOL NewWav;
        BOOL IsWavfile;

// Operations
public:
        bool SetWavFile(CString);

        void ClearFrame(CDC*);
        void PaintBorder(CDC*);
        void PaintGrid(CDC*);
        void PaintTitle(CDC*);
        void PaintGraph(CDC*, COLORREF);
        void PaintSNR(CDC*);
        void PaintSnrValue(CDC*);
        float ComputeSnrValue(int);

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

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

protected:

// Generated message map functions
protected:
        //{{AFX_MSG(CVectorQView)
        afx_msg void OnScrollR();
        afx_msg void OnViewWstyle();
        afx_msg void OnScrollL();
        afx_msg void OnViewZoomOut();
        afx_msg void OnViewZoomIn();
        afx_msg void OnViewZoom1_1();
        afx_msg void OnViewZoom1_Max();
        afx_msg void OnViewIncdecInc();
        afx_msg void OnViewIncdecDec();
        afx_msg void OnViewIncdec1_1();
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};

#ifndef _DEBUG  // debug version in VectorQView.cpp
inline CVectorQDoc* CVectorQView::GetDocument()
   { return (CVectorQDoc*)m_pDocument; }
#endif

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

//{{AFX_INSERT_LOCATION}}
```

```
// Microsoft Visual C++ will insert additional declarations immediately before the
previous line.

#endif // !defined(AFX_VECTORQVIEW_H__038FCF9C_249C_4122_879F_6C7921B31F3D__INCLUDED_)

// VectorQView.cpp : implementation of the CVectorQView class
//

#include "stdafx.h"
#include "VectorQ.h"

#include "VectorQDoc.h"
#include "VectorQView.h"
#include "math.h"

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

/////////////////////////////////////////////////////////////////////////////
// CVectorQView

IMPLEMENT_DYNCREATE(CVectorQView, CView)

BEGIN_MESSAGE_MAP(CVectorQView, CView)
        //{{AFX_MSG_MAP(CVectorQView)
        ON_COMMAND(ID_SCROLL_R, OnScrollR)
        ON_COMMAND(ID_VIEW_WSTYLE, OnViewWstyle)
        ON_COMMAND(ID_SCROLL_L, OnScrollL)
        ON_COMMAND(ID_VIEW_ZOOM_OUT, OnViewZoomOut)
        ON_COMMAND(ID_VIEW_ZOOM_IN, OnViewZoomIn)
        ON_COMMAND(ID_VIEW_ZOOM_1_1, OnViewZoom1_1)
        ON_COMMAND(ID_VIEW_ZOOM_1_MAX, OnViewZoom1_Max)
        ON_COMMAND(ID_VIEW_INCDEC_INC, OnViewIncdecInc)
        ON_COMMAND(ID_VIEW_INCDEC_DEC, OnViewIncdecDec)
        ON_COMMAND(ID_VIEW_INCDEC_1_1, OnViewIncdec1_1)
        //}}AFX_MSG_MAP
        // Standard printing commands
        ON_COMMAND(ID_FILE_PRINT, CView::OnFilePrint)
        ON_COMMAND(ID_FILE_PRINT_DIRECT, CView::OnFilePrint)
        ON_COMMAND(ID_FILE_PRINT_PREVIEW, CView::OnFilePrintPreview)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CVectorQView construction/destruction

CVectorQView::CVectorQView()
{
        // TODO: add construction code here

        Shift = 0;      // Shift next screen
        Zoom = 1;
        IncDec = 0;
        Scale = 50;
        Border.x = 80;
        Border.y = 80;
        WavStyle = false;

        bgColor = RGB(0,0,0);
        bdColor = RGB(255,255,255);
        gdColor = RGB(170,170,170);
        ttColor = RGB(255,255,255);
        w1Color = RGB(255,0,0);
        w2Color = RGB(0,255,0);
        w3Color = RGB(0,0,255);
        w4Color = RGB(255,255,0);
```

```
        WNum = Wchar.WavNum;
        Snr = Wchar.IsSnr;
        NewWav = TRUE;

        if(Snr) Wchar.SNRNum++;        // plus for Orginal file
}

CVectorQView::~CVectorQView()
{
        if(!Snr && IsWavfile) Wchar.WavNum--;
        else if(Snr){
                Wchar.SNR = 0;
                Wchar.SNRNum--;
        }
}

BOOL CVectorQView::PreCreateWindow(CREATESTRUCT& cs)
{
        // TODO: Modify the Window class or styles here by modifying
        //  the CREATESTRUCT cs

        return CView::PreCreateWindow(cs);
}

/////////////////////////////////////////////////////////////////////////////
// CVectorQView drawing

void CVectorQView::OnDraw(CDC* pDC)
{
        CVectorQDoc* pDoc = GetDocument();
        ASSERT_VALID(pDoc);

        // TODO: add draw code for native data here

        if(NewWav){
                IsWavfile = SetWavFile(pDoc->GetPathName());
                NewWav = FALSE;
        }

        if(IsWavfile){
                ClearFrame(pDC);
                PaintBorder(pDC);
                PaintGrid(pDC);

                if(!Snr){
                        PaintTitle(pDC);
                        PaintGraph(pDC, w1Color);
                }
                else{
                        PaintSnrValue(pDC);
                        PaintSNR(pDC);
                }
        }
}

/////////////////////////////////////////////////////////////////////////////
// CVectorQView printing

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

void CVectorQView::OnBeginPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
{
```

```
        // TODO: add extra initialization before printing
}

void CVectorQView::OnEndPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
{
        // TODO: add cleanup after printing
}

/////////////////////////////////////////////////////////////////////////////
// CVectorQView diagnostics

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

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

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

/////////////////////////////////////////////////////////////////////////////
// CVectorQView message handlers


bool CVectorQView::SetWavFile(CString fpath)
{
        if(!Wchar.IsSnr){
                if(!Wchar.SetWavData(fpath, Wchar.WavNum)){
                        MessageBox("File doesn't WAVE
Format","Error",MB_OK|MB_ICONSTOP);
                        return 0;
                }
        }
        else{
                if(!Wchar.SetWavData(fpath, 6)){
                        MessageBox("File doesn't WAVE
Format","Error",MB_OK|MB_ICONSTOP);
                        return 0;
                }
        }

        return 1;

}

void CVectorQView::ClearFrame(CDC* pDC)
{
        CPen pen;
        CBrush brush;
        CRect rect;
        GetClientRect(&rect);

        pen.CreatePen(PS_SOLID,1,bgColor);
        brush.CreateSolidBrush(bgColor);
        pDC->SelectObject(&pen);
        pDC->SelectObject(&brush);

        pDC->Rectangle(0,0,rect.Width(),rect.Height());             // Clear Screen
```

```
        pen.DeleteObject();
        brush.DeleteObject();
}

void CVectorQView::PaintBorder(CDC* pDC)
{
        CPen pen;
        CBrush brush;
        CRect rect;
        GetClientRect(&rect);

        pen.CreatePen(PS_SOLID,1,bdColor);
        brush.CreateSolidBrush(bgColor);
        pDC->SelectObject(&pen);
        pDC->SelectObject(&brush);
        pDC->Rectangle(Border.x,10,rect.Width()-10,rect.Height()-Border.y);// Border

        pen.DeleteObject();
        brush.DeleteObject();
}

void CVectorQView::PaintGrid(CDC* pDC)
{
        CPen pen;
        CBrush brush;
        CFont title;
        CRect rect;
        GetClientRect(&rect);

        char buff[5];
        int bit;

        title.CreateFont(12,0,0,0,FW_BOLD,false,false,0,ANSI_CHARSET,OUT_DEFAULT_PRECI
S,
                                CLIP_DEFAULT_PRECIS, DEFAULT_QUALITY,
DEFAULT_PITCH | FF_SWISS, "AngsanaNew");

        pen.CreatePen(PS_SOLID,1,bdColor);

        pDC->SelectObject(&title);
        pDC->SelectObject(&pen);
        pDC->SetTextColor(ttColor);
        pDC->SetBkColor(bgColor);

        pDC->MoveTo(Border.x,rect.CenterPoint().y-((Border.y-10)/2));
        pDC->LineTo(rect.Width()-10,rect.CenterPoint().y-((Border.y-10)/2));        //
Center Line
        pen.DeleteObject();

        pen.CreatePen(PS_SOLID,1,bgColor);
        brush.CreateSolidBrush(bgColor);
        pDC->SelectObject(&pen);
        pDC->SelectObject(&brush);
        pDC->Rectangle(Border.x,rect.Height()-(Border.y-2),rect.Width(),rect.Height()-
(Border.y-15));        // Clear Old Scale
        pDC->Rectangle(42,10,Border.x-1,rect.Height()-Border.y);
        pDC->TextOut(45,rect.CenterPoint().y-((Border.y-10)/2)-6,"0");      // Zero
Amplitude
        pen.DeleteObject();

        pen.CreatePen(PS_DASHDOT,1,gdColor);
        pDC->SelectObject(&pen);
        for(int i = 0; i < rect.Width()-(Border.x+Scale+12); i = i+Scale){
                pDC->MoveTo(Border.x+Scale+i,11);
                pDC->LineTo(Border.x+Scale+i,rect.Height()-(Border.y+1));   //
Horizontal Scale
```

```
                if(!Wchar.BitRate[WNum]) pDC->TextOut(Border.x+Scale+i-
10,rect.Height()-(Border.y-3),itoa((((i+Scale)*Zoom)+Shift),buff,10));     // Sample
Number
                else pDC->TextOut(Border.x+Scale+i-10,rect.Height()-(Border.y-
3),itoa((((i+Scale)*Zoom)+(Shift/2)),buff,10));      // Sample Number
        }

        if(!Wchar.BitRate[WNum]) bit = 1;
        else bit = 256;

        int k, j;
        j = k = rect.CenterPoint().y-((Border.y-10)/2);
        for(int l = 1; j > Scale+11; j = j-Scale, k = k+Scale, l++){                //
Vertical Scale
                pDC->MoveTo(Border.x+1,j-Scale);
                pDC->LineTo(rect.Width()-11,j-Scale);
                pDC->TextOut(45,j-Scale-6,itoa((Scale*bit*l)/(IncDec+1),buff,10));
        // Positive Amplitude Number
                pDC->MoveTo(Border.x+1,k+Scale);
                pDC->LineTo(rect.Width()-11,k+Scale);
                pDC->TextOut(45,k+Scale-6,itoa(((-1)*Scale*bit*l)/(IncDec+1),buff,10));
        // Negative Amplitude Number
        }

        pen.DeleteObject();
        brush.DeleteObject();
        title.DeleteObject();

}

void CVectorQView::PaintTitle(CDC* pDC)
{
        CFont title;
        CRect rect;
        GetClientRect(&rect);

        char buff[5];
        CString strbuff;

        title.CreateFont(16,0,0,0,FW_BOLD,false,false,0,ANSI_CHARSET,OUT_DEFAULT_PRECI
S,
                                        CLIP_DEFAULT_PRECIS, DEFAULT_QUALITY,
DEFAULT_PITCH | FF_SWISS, "AngsanaNew");
        pDC->SelectObject(&title);
        pDC->SetTextColor(ttColor);
        pDC->SetBkColor(bgColor);

        pDC->TextOut(rect.CenterPoint().x-10,rect.Height()-Border.y+20,"Sample");  //
Axis X title

        title.DeleteObject();

        title.CreateFont(16,0,900,0,FW_BOLD,false,false,0,ANSI_CHARSET,OUT_DEFAULT_PRE
CIS,
                                        CLIP_DEFAULT_PRECIS, DEFAULT_QUALITY,
DEFAULT_PITCH | FF_SWISS, "AngsanaNew");
        pDC->SelectObject(&title);
        pDC->TextOut(20,rect.CenterPoint().y,"Amplitude");  // Axis Y title

        title.DeleteObject();

        title.CreateFont(12,0,0,0,FW_BOLD,false,false,0,ANSI_CHARSET,OUT_DEFAULT_PRECI
S,
                                        CLIP_DEFAULT_PRECIS, DEFAULT_QUALITY,
DEFAULT_PITCH | FF_SWISS, "AngsanaNew");
        pDC->SelectObject(&title);

        int bit;
```

```
        CStdioFile WavFile(Wchar.WavPointer[WNum],CFile::modeRead|CFile::typeBinary);
        pDC->TextOut(Border.x,rect.Height()-40,"File Name :    " +
WavFile.GetFileName());
        WavFile.Close();
        if(!Wchar.BitRate[WNum]){                       // Wave Format
                bit = 1;
                strbuff = itoa(Wchar.WavFreq[WNum],buff,10);
                pDC->TextOut(Border.x,rect.Height()-30,"Format :    " + strbuff + " Hz,
8 bits");
                strbuff = itoa(Wchar.WavSize[WNum],buff,10);
                pDC->TextOut(Border.x,rect.Height()-20,"Sample Numbers :    " + strbuff
+ " samples");
        }
        else{
                bit = 256;
                strbuff = itoa(Wchar.WavFreq[WNum],buff,10);
                pDC->TextOut(Border.x,rect.Height()-30,"Format :    " + strbuff + " Hz,
16 bits");
                strbuff = itoa((Wchar.WavSize[WNum]/2),buff,10);
                pDC->TextOut(Border.x,rect.Height()-20,"Sample Numbers :    " + strbuff
+ " samples");
        }

}

void CVectorQView::PaintGraph(CDC* pDC, COLORREF cr)
{
        CRect rect;
        CPen pen;
        GetClientRect(&rect);

        CPoint WavPoint;
        CPoint StartPoint;
        int axis_x = 0;

        pen.CreatePen(PS_SOLID,1,cr);

        pDC->SelectObject(&pen);
        StartPoint.x = Border.x+1;
        StartPoint.y = rect.CenterPoint().y-((Border.y-10)/2);

        if(!Wchar.BitRate[WNum]){       // 8 bit
                for(ULONG j = 0, i = Shift; (axis_x < rect.Width()-12) && (i <
Wchar.WavSize[WNum]); j++,i++){
                        WavPoint.x = axis_x = Border.x+1 + j/Zoom;
                        WavPoint.y =  rect.CenterPoint().y + (128-((Border.y-10)/2)) -
Wchar.WavData[WNum][i];

                        if(Wchar.WavData[WNum][i]>128) WavPoint.y = WavPoint.y -
((rect.CenterPoint().y-((Border.y-10)/2) - WavPoint.y) * IncDec);
                        else if(Wchar.WavData[WNum][i]<128) WavPoint.y = WavPoint.y +
(((WavPoint.y)-(rect.CenterPoint().y-((Border.y-10)/2))) * IncDec);

                        if(WavPoint.y > 10 && WavPoint.y < rect.Height()-(Border.y+1)){
        // Check over border
                                if(WavStyle) pDC->SetPixel(WavPoint,cr);     // Paint
Dot Wave
                                else{
                                        pDC->MoveTo(StartPoint.x,StartPoint.y);
        // Paint Line Wave
                                        pDC->LineTo(WavPoint);
                                        StartPoint.x = WavPoint.x;
                                        StartPoint.y = WavPoint.y;
                                }
                        }
                }
        }
        else{                           // 16 bit
```

```
                int DataBuff;
                for(ULONG j = 0, i = Shift; (axis_x < rect.Width()-12) && (i <
Wchar.WavSize[WNum]); j++,i+=2){
                        WavPoint.x = axis_x = Border.x+1 + j/Zoom;
                        DataBuff = Wchar.WavData[WNum][i+1]*256+Wchar.WavData[WNum][i];
                        if(DataBuff > 32767) DataBuff -= 65536;
                        DataBuff /= 256;
                        WavPoint.y =  rect.CenterPoint().y - ((Border.y-10)/2) -
DataBuff;

                        if(DataBuff>0) WavPoint.y = WavPoint.y -
((rect.CenterPoint().y-((Border.y-10)/2) - WavPoint.y) * IncDec);
                        else if(DataBuff<0) WavPoint.y = WavPoint.y + (((WavPoint.y)-
(rect.CenterPoint().y-((Border.y-10)/2))) * IncDec);

                        if(WavPoint.y > 10 && WavPoint.y < rect.Height()-(Border.y+1)){
        // Check over border
                                if(WavStyle) pDC->SetPixel(WavPoint,cr);      // Paint
Dot Wave
                                else{
                                        pDC->MoveTo(StartPoint.x,StartPoint.y);
        // Paint Line Wave
                                        pDC->LineTo(WavPoint);
                                        StartPoint.x = WavPoint.x;
                                        StartPoint.y = WavPoint.y;
                                }
                        }
                }
        }

        pen.DeleteObject();
}

float CVectorQView::ComputeSnrValue(int wnum){

        double res1, res2, res3, res4, snr;
        int total;
        long *xn, *xxn;
        int a, b;
        int pt=0;
        long size;

        snr = 0;
        total = 0;

        if(Wchar.BitRate == FALSE){
                size = Wchar.WavSize[6];

                xn = new long[size];
                for(b=0; b<size; b++){
                        if(Wchar.WavData[6][b] <= 127) xn[b] = Wchar.WavData[6][b];
                        else xn[b] = Wchar.WavData[6][b] - 256;
                }

                xxn = new long[size];
                for(b=0; b<size; b++){
                        if(Wchar.WavData[wnum][b] <= 127) xn[b] =
Wchar.WavData[wnum][b];
                        else xn[b] = Wchar.WavData[wnum][b] - 256;
                }
        }

        else{
                size = Wchar.WavSize[6]/2;

                xn = new long[size];
                for(b=0; b<size; b++){
```

```
                        if((Wchar.WavData[6][b*2] + Wchar.WavData[6][b*2+1]*256) <=
32767) xn[b] = Wchar.WavData[6][b*2] + Wchar.WavData[6][b*2+1]*256;
                        else xn[b] = (Wchar.WavData[6][b*2] +
Wchar.WavData[6][b*2+1]*256) - 65536;
                }

                xxn = new long[size];
                for(b=0; b<size; b++){
                        if((Wchar.WavData[wnum][b*2] + Wchar.WavData[wnum][b*2+1]*256)
<= 32767) xxn[b] = Wchar.WavData[wnum][b*2] + Wchar.WavData[wnum][b*2+1]*256;
                        else xxn[b] = (Wchar.WavData[wnum][b*2] +
Wchar.WavData[wnum][b*2+1]*256) - 65536;
                }
        }

    while (size > 128)
    {
        res1 = 0;
                res2 = 0;

        for (a=0; a<128; a++)
        {
            res3=(float)xn[pt+a];
            res3=res3*res3;
            res4=(float)xn[pt+a]-(float)xxn[pt+a];
            res4=res4*res4;
            res1+=res3;
            res2+=res4;
        }

        res1/=res2;
        res2=10*log10 (res1);

        if  (res2>0.0)
        {
            total++;
            snr+=res2;
        }

                pt += 128;
                size -= 128;
    }

        return (float)(snr/total);
}

void CVectorQView::PaintSnrValue(CDC* pDC){

        CPen pen;
        CBrush brush;
        CFont title;
        CRect rect;
        GetClientRect(&rect);

        int sint;
        float snr;
        int sfloat;
        char buff[10];
        CString strbuff1;
        CString strbuff2;

        title.CreateFont(16,0,0,0,FW_BOLD,false,false,0,ANSI_CHARSET,OUT_DEFAULT_PRECI
S,
                                        CLIP_DEFAULT_PRECIS, DEFAULT_QUALITY,
DEFAULT_PITCH | FF_SWISS, "AngsanaNew");
        pDC->SetBkColor(bgColor);
        pDC->SetTextColor(ttColor);
        pDC->SelectObject(&title);
```

```
        pDC->TextOut(rect.CenterPoint().x-10,rect.Height()-Border.y+20,"Sample");  //
Axis X title

        title.DeleteObject();

        title.CreateFont(16,0,900,0,FW_BOLD,false,false,0,ANSI_CHARSET,OUT_DEFAULT_PRE
CIS,
                                        CLIP_DEFAULT_PRECIS, DEFAULT_QUALITY,
DEFAULT_PITCH | FF_SWISS, "AngsanaNew");
        pDC->SelectObject(&title);
        pDC->TextOut(20,rect.CenterPoint().y,"Amplitude");   // Axis Y title

        title.DeleteObject();

        title.CreateFont(12,0,0,0,FW_BOLD,false,false,0,ANSI_CHARSET,OUT_DEFAULT_PRECI
S,
                                        CLIP_DEFAULT_PRECIS, DEFAULT_QUALITY,
DEFAULT_PITCH | FF_SWISS, "AngsanaNew");
        pDC->SelectObject(&title);

        COLORREF cr;

        CStdioFile Wf;

        for(int i=0; i<Wchar.SNR; i++){
                switch(i){
                case 0: cr = w1Color; break;
                case 1: cr = w2Color; break;
                case 2: cr = w3Color; break;
                case 3: cr = w4Color; break;
                }

                pen.CreatePen(PS_SOLID,1,cr);
                brush.CreateSolidBrush(cr);
                pDC->SelectObject(&pen);
                pDC->SelectObject(&brush);
                pDC->Rectangle(Border.x,rect.Height()-
50+(i*12),Border.x+10,rect.Height()-45+(i*12));
                pen.DeleteObject();
                brush.DeleteObject();

                Wf.Open(Wchar.WavPointer[6+i],CFile::modeRead|CFile::typeBinary);
                pDC->TextOut(Border.x+15,rect.Height()-55+(i*12),Wf.GetFileName());
                Wf.Close();
                if(i>0){
                        snr = ComputeSnrValue(6+i);
                        sint = (int)snr;
                        snr -= sint;
                        snr *= 100;
                        sfloat = (int)snr;

                        strbuff1 = itoa(sint,buff,10);
                        strbuff2 = itoa(sfloat,buff,10);
                        if (strbuff2.GetLength() == 1) strbuff2 = "0" + strbuff2;

                        pDC->TextOut(Border.x+150,rect.Height()-55+(i*12),"SNR Total :
"+strbuff1+"."+strbuff2 );

                }
        }

}

void CVectorQView::PaintSNR(CDC* pDC){
        for(int i=0; i<Wchar.SNR; i++){
                WNum = 6+i;
                switch(WNum){
                case 6: PaintGraph(pDC, w1Color); break;
```

```
                case 7: PaintGraph(pDC, w2Color); break;
                case 8: PaintGraph(pDC, w3Color); break;
                case 9: PaintGraph(pDC, w4Color); break;
                }
        }
}

void CVectorQView::OnViewZoomOut()
{
        // TODO: Add your command handler code here
        CRect rect;
        GetClientRect(&rect);
        if(!Wchar.BitRate[WNum]){
                if((ULONG)rect.Width() <= Wchar.WavSize[WNum]/Zoom){
                        Zoom = Zoom * 2;  // 8 bit
                        RedrawWindow();
                }
        }
        else{
                ULONG sizebuff = Wchar.WavSize[WNum]/2;
                if((ULONG)rect.Width() <= sizebuff/Zoom){
                        Zoom = Zoom * 2;  // 16 bit
                        RedrawWindow();
                }
        }
}

void CVectorQView::OnViewZoomIn()
{
        // TODO: Add your command handler code here
        if(Zoom != 1){
                Zoom = Zoom / 2;
                RedrawWindow();
        }
}

void CVectorQView::OnViewZoom1_1()
{
        // TODO: Add your command handler code here
        if(Zoom != 1){
                Zoom = 1;
                RedrawWindow();
        }
}

void CVectorQView::OnViewZoom1_Max()
{
        // TODO: Add your command handler code here
        CRect rect;
        GetClientRect(&rect);

        if(!Wchar.BitRate[WNum]){
                if((ULONG)rect.Width() <= Wchar.WavSize[WNum]/Zoom){
                        while((ULONG)rect.Width() <= Wchar.WavSize[WNum]/Zoom)      Zoom
= Zoom * 2;  // 8 bit
                        RedrawWindow();
                }
        }
        else{
                ULONG sizebuff = Wchar.WavSize[WNum]/2;
                if((ULONG)rect.Width() <= sizebuff/Zoom){
                        while((ULONG)rect.Width() <= sizebuff/Zoom) Zoom = Zoom * 2;
// 16 bit
                        RedrawWindow();
                }
        }
}
```

```
void CVectorQView::OnViewWstyle()
{
        // TODO: Add your command handler code here
        if(WavStyle) WavStyle = false;
        else WavStyle = true;
        RedrawWindow();
}


void CVectorQView::OnScrollR()
{
        // TODO: Add your command handler code here
        CRect rect;
        GetClientRect(&rect);

        if(!Wchar.BitRate[WNum]){
                if((ULONG)(Shift + (rect.Width()-Border.x-10)) < Wchar.WavSize[WNum]){
                        Shift += 500;
                        RedrawWindow();
                }
        }
        else{
                if((ULONG)(Shift + (rect.Width()-Border.x-10)) <
(Wchar.WavSize[WNum])){
                        Shift += 1000;
                        RedrawWindow();
                }
        }

}

void CVectorQView::OnScrollL()
{
        // TODO: Add your command handler code here
        if(Shift > 0){
                if(!Wchar.BitRate[WNum])        Shift -= 500;
                else Shift -= 1000;
                RedrawWindow();
        }
}

void CVectorQView::OnViewIncdecInc()
{
        // TODO: Add your command handler code here
        IncDec++;
        RedrawWindow();
}

void CVectorQView::OnViewIncdecDec()
{
        // TODO: Add your command handler code here
        if(IncDec != 0){
                IncDec--;
                RedrawWindow();
        }

}

void CVectorQView::OnViewIncdec1_1()
{
        // TODO: Add your command handler code here
        if(IncDec != 0){
                IncDec = 0;
                RedrawWindow();
        }

}
```

```
// VEndec.h : header file
//
#if !defined(AFX_VEndec_H__E267C9C4_A422_4485_B7B6_E8581D0D77D8__INCLUDED_)
#define AFX_VEndec_H__E267C9C4_A422_4485_B7B6_E8581D0D77D8__INCLUDED_

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

/////////////////////////////////////////////////////////////////////////////
// CVEndec dialog

class CVEndec : public CDialog
{
// Construction
public:
        CVEndec(CWnd* pParent = NULL);   // standard constructor
        bool MakeData(CString);
        void Encode(long);
        void Decode();
        bool EnDecSVQ();

protected:
        virtual BOOL OnInitDialog();
        void InitProgress();

// Dialog Data
        //{{AFX_DATA(CVEndec)
        enum { IDD = IDD_VQ_ENDEC };
        CString m_WavFile;
        CString m_CBFile;
        CString m_SVQFile;
        CProgressCtrl m_progress;     // object of progress control
        CString m_ProgStr;
        int m_VC;
        int m_SI;
        //}}AFX_DATA

// Endec Data
        CString fname, fname1, fname2;
        int CodebookSize, CodebookDimension, Frame, findex;
        float Codebook[10][512], New[10];

        FILE *record_file;                      /* File to accept record-data */
        FILE *read_file;                        /* File to accept record-data */
        long *data;
        BYTE header[58];
        BYTE wfbuff[100];
        int bit;

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

// Implementation
protected:

        // Generated message map functions
        //{{AFX_MSG(CVEndec)
        afx_msg void OnBrowseWF();
        afx_msg void OnBrowseCBF();
        afx_msg void OnBrowseSVQF();
        afx_msg void OnChangeEditWF();
        afx_msg void OnChangeEditCBF();
        afx_msg void OnChangeEditSVQF();
```

```
        virtual void OnOK();
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};

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

#endif // !defined(AFX_VENDEC_H__E267C9C4_A422_4485_B7B6_E8581D0D77D8__INCLUDED_)
// VEndec.cpp : implementation file
//

#include "stdafx.h"
#include "VectorQ.h"
#include "VEndec.h"

#include "math.h"

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

/////////////////////////////////////////////////////////////////////////////
// CVEndec dialog


CVEndec::CVEndec(CWnd* pParent /*=NULL*/)
        : CDialog(CVEndec::IDD, pParent)
{
        //{{AFX_DATA_INIT(CVEndec)
        m_WavFile = _T("");
        m_CBFile = _T("");
        m_SVQFile = _T("");
        m_ProgStr = _T("Ready ...");
        m_VC = 0;
        m_SI = 0;

        //}}AFX_DATA_INIT
}

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

        InitProgress();                         // initial progress

        return TRUE;
}

void CVEndec::InitProgress(){
        m_progress.SetRange(0,100);
        m_progress.SetStep(1);
        m_progress.SetPos(0);
}

void CVEndec::DoDataExchange(CDataExchange* pDX)
{
        CDialog::DoDataExchange(pDX);
        //{{AFX_DATA_MAP(CVEndec)
        DDX_Text(pDX, IDC_EDIT_WF, m_WavFile);
        DDX_Text(pDX, IDC_EDIT_CBF, m_CBFile);
        DDX_Text(pDX, IDC_EDIT_SVQF, m_SVQFile);
        DDX_Control(pDX,IDC_PROGRESS1,m_progress);
        DDX_Text(pDX, IDC_STATIC_PG, m_ProgStr);
        DDX_Text(pDX, IDC_STATIC_VC2, m_VC);
        DDX_Text(pDX, IDC_STATIC_SI2, m_SI);
```

```
        //}}AFX_DATA_MAP
}


BEGIN_MESSAGE_MAP(CVEndec, CDialog)
        //{{AFX_MSG_MAP(CVEndec)
        ON_BN_CLICKED(IDC_BROWSE_WF, OnBrowseWF)
        ON_BN_CLICKED(IDC_BROWSE_CBF, OnBrowseCBF)
        ON_BN_CLICKED(IDC_BROWSE_SVQF, OnBrowseSVQF)
        ON_EN_CHANGE(IDC_EDIT_WF, OnChangeEditWF)
        ON_EN_CHANGE(IDC_EDIT_CBF, OnChangeEditCBF)
        ON_EN_CHANGE(IDC_EDIT_SVQF, OnChangeEditSVQF)
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CVEndec message handlers

void CVEndec::OnBrowseWF()
{
        // TODO: Add your control notification handler code here
        UpdateData(TRUE);
        CFileDialog fd(TRUE, NULL, NULL, OFN_HIDEREADONLY,"WAVE Files
(*.wav)|*.wav||");
        if(fd.DoModal() == IDOK){                    // open file dialog
                if(fd.GetFileExt()!="wav"){
                        MessageBox("The lastname must be .wav
only","Error",MB_OK|MB_ICONEXCLAMATION);
                        m_WavFile = "";
                }
                else m_WavFile = fd.GetPathName();
        }
        UpdateData(FALSE);
}

void CVEndec::OnBrowseCBF()
{
        // TODO: Add your control notification handler code here
        UpdateData(TRUE);
        CFileDialog fd(TRUE, NULL, NULL,OFN_HIDEREADONLY,"Code Book file
(*.cdb)|*.cdb||");
        if(fd.DoModal()==IDOK){                      // open file dialog
                if(fd.GetFileExt()!="cdb"){
                        MessageBox("The lastname must be .cdb
only","Error",MB_OK|MB_ICONEXCLAMATION);
                        m_CBFile = "";
                }
                else m_CBFile = fd.GetPathName();    // set text of saveas
        }
        UpdateData(FALSE);
}

void CVEndec::OnBrowseSVQF()
{
        // TODO: Add your control notification handler code here
        UpdateData(TRUE);
        CFileDialog
fd(FALSE,"wav","Untitled.wav",OFN_HIDEREADONLY|OFN_OVERWRITEPROMPT,"WAVE Files
(*.wav)|*.wav||");
        if(fd.DoModal()==IDOK){                      // open saveas dialog
                if(fd.GetFileExt()!="wav"){

                        MessageBox("The lastname must be .wav
only","Error",MB_OK|MB_ICONEXCLAMATION);
                        m_SVQFile = "";
                }
                else m_SVQFile = fd.GetPathName();   // set text of saveas
        }
```

```
        UpdateData(FALSE);
}

void CVEndec::OnChangeEditWF()
{
        // TODO: Add your control notification handler code here
        m_WavFile = m_WavFile;
        UpdateData(FALSE);
}

void CVEndec::OnChangeEditCBF()
{
        // TODO: Add your control notification handler code here
        m_CBFile = m_CBFile;
        UpdateData(FALSE);
}

void CVEndec::OnChangeEditSVQF()
{
        // TODO: Add your control notification handler code here
        m_SVQFile = m_SVQFile;
        UpdateData(FALSE);
}

void CVEndec::OnOK()
{
        // TODO: Add extra validation here
        fname = m_WavFile;
        fname1 = m_SVQFile;
        fname2 = m_CBFile;

        if(EnDecSVQ()){
                MessageBox("Encode / Decode SVQ complete","Encode / Decode SVQ",MB_OK);
                CDialog::OnOK();
        }
}

bool CVEndec::MakeData(CString fname)
{
        CStdioFile wf(fname,CFile::modeRead|CFile::typeBinary);     // open Wave file


        wf.Seek(24,wf.GetPosition()); // Seek to bit rate
        BYTE bitbuff[5];

        wf.Read(bitbuff,5);    // Read bit rate
        if(bitbuff[0] == bitbuff[4])  bit  = 8;        //      8  bit
        else if(bitbuff[4] == bitbuff[0]*2) bit = 16;        //      16 bit
        else {wf.Close(); return 0;}

        wf.SeekToBegin();      // Cut Header
        wf.Read(header,58);

        long size = wf.GetLength()-58;        // 58 is Wave File Header
        BYTE *wfbuff;

        wfbuff = new BYTE[size];
        wf.Read(wfbuff,size);

        if(bit == 8){
                data = new long[size];
                for(int a=0; a<size; a++){
                        if(wfbuff[a] <= 127) data[a] = wfbuff[a];
                        else data[a] = wfbuff[a] - 256;
                }
        }
        else if(bit == 16){
                data = new long[size/2];
```

```
                for(int b=0; b<size/2; b++){
                        if((wfbuff[b*2] + wfbuff[b*2+1]*256) <= 32767) data[b] =
wfbuff[b*2] + wfbuff[b*2+1]*256;
                        else data[b] = (wfbuff[b*2] + wfbuff[b*2+1]*256) - 65536;
                }
        }
        else return FALSE;

    if(bit == 8) Frame = size/(CodebookDimension);
        else if(bit == 16) Frame = size/(CodebookDimension*2);

        return TRUE;
}


bool CVEndec::EnDecSVQ(){

        int a, b, i;
        float cb;


    if((read_file = fopen(fname2,"rb")) == NULL)
    {
                MessageBox("Can not read Code book
file","Error",MB_OK|MB_ICONEXCLAMATION);
        return FALSE;
    }

    fscanf (read_file, "%s", wfbuff);
    fscanf (read_file, "%d", &CodebookDimension);
    fscanf (read_file, "%s", wfbuff);
    fscanf (read_file, "%d", &CodebookSize);

    for (a=0; a<CodebookSize; a++)
    {
        for (b=0; b<CodebookDimension; b++)
        {
            fscanf (read_file, "%f", &cb);
            Codebook[b][a]=cb;
        }
    }
    fclose (read_file);

    if((record_file = fopen(fname1,"wb")) == NULL)
    {
                MessageBox("Can not write output
file","Error",MB_OK|MB_ICONEXCLAMATION);
        return FALSE;
    }

       if(!MakeData(fname)){
                MessageBox("A Wave file format is
wrong","Error",MB_OK|MB_ICONEXCLAMATION);
        return FALSE;
       }

       m_progress.SetRange(0,Frame);

    fwrite((char *)header, 1, 58, record_file);

    for (i=0; i<Frame; i++)
    {
        Encode((long)(i*CodebookDimension));

        Decode();

                if(bit == 8) fwrite((char *)wfbuff, 1,CodebookDimension , record_file);
```

```
                else if(bit == 16) fwrite((char *)wfbuff, 1,CodebookDimension*2 ,
record_file);

                UpdateData(TRUE);
                m_VC = i;
                m_SI = findex;
                UpdateData(FALSE);

                m_progress.StepIt();
    }

    fclose(record_file);

        return TRUE;
}


void CVEndec::Encode (long pt)

{
        double res1, res2;
        int     a, b ;

    res1=9.0E+15;
    for (a=0; a<CodebookSize; a++)
    {
        res2=0;
        for (b=0; b<CodebookDimension; b++)
        {
            New[b]=(float)data[pt+b]-Codebook[b][a];
            res2+=New[b]*New[b];
        }
        res2=sqrt(res2);
        if  (res2<res1)
        {
            res1=res2;
            findex=a;
        }
    }
}


void CVEndec::Decode ()
{
        int cbbuff;

        if(bit == 8){
                for (int b=0; b<CodebookDimension; b++){
                        cbbuff = (int)Codebook[b][findex];
                        if(cbbuff < 0) cbbuff += 256;
                        wfbuff[b] = cbbuff;
                }
        }
        else if(bit == 16){
                for (int b=0; b<CodebookDimension; b++){
                        cbbuff = (int)Codebook[b][findex];
                        if(cbbuff < 0) cbbuff += 65536;
                        wfbuff[b*2+1] = cbbuff/256;
                        wfbuff[b*2] = cbbuff%256;
                }
        }

}

// WavChar.h: interface for the CWavChar class.
//
//////////////////////////////////////////////////////////////////////
```

```
#if !defined(AFX_WAVCHAR_H__7C8771D6_56EC_4405_86D1_A6FE2DD11E6E__INCLUDED_)
#define AFX_WAVCHAR_H__7C8771D6_56EC_4405_86D1_A6FE2DD11E6E__INCLUDED_

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

class CWavChar : public CObject
{
public:
        CWavChar();
        virtual ~CWavChar();

        BOOL SetWavData(CString, int);

        static CString WavPointer[15];
        static BYTE* WavData[15];
        static ULONG WavSize[15];
        static BOOL BitRate[15];
        static int WavFreq[15];
        static int WavNum;

        static int SNR;
        static int SNRNum;
        static BOOL IsSnr;

};

#endif // !defined(AFX_WAVCHAR_H__7C8771D6_56EC_4405_86D1_A6FE2DD11E6E__INCLUDED_)

// WavChar.cpp: implementation of the CWavChar class.
//
//////////////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "VectorQ.h"
#include "WavChar.h"

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

//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////
CString CWavChar::WavPointer[15] = {""};
BYTE* CWavChar::WavData[15] = {NULL};
ULONG CWavChar::WavSize[15] = {0};
BOOL CWavChar::BitRate[15] = {TRUE};
int CWavChar::WavFreq[15] = {0};
int CWavChar::WavNum = 0;
int CWavChar::SNR = 0;
int CWavChar::SNRNum = 0;
BOOL CWavChar::IsSnr = FALSE;

CWavChar::CWavChar()
{

}

CWavChar::~CWavChar()
{

}

BOOL CWavChar::SetWavData(CString WavPath, int Wnum)
```

```
{
        WavPointer[Wnum] = WavPath;
        CStdioFile WavFile(WavPath,CFile::modeRead|CFile::typeBinary);      // open
Wave file
        WavFile.Seek(24,WavFile.GetPosition());      // Seek to bit rate
        BYTE bitbuff[5];

        WavFile.Read(bitbuff,5);      // Read bit rate
        if(bitbuff[0] == bitbuff[4])  BitRate[Wnum]  = FALSE;       //      8  bit
        else if(bitbuff[4] == bitbuff[0]*2) BitRate[Wnum] = TRUE;   //      16 bit
        else{ WavFile.Close(); return 0; }    // Wrong format

        int freqbuff = bitbuff[0]*256 + bitbuff[1];
        if              (freqbuff % 16415 == 0) WavFreq[Wnum] = 8000 * (freqbuff /
16415); // multiple 8,000 hz
        else if (freqbuff % 4395 == 0) WavFreq[Wnum] = 11025 * (freqbuff / 4395);
        // multiple 11,025 hz
        else if (freqbuff % 57390 == 0) WavFreq[Wnum] = 12000 * (freqbuff / 57390);
        // multiple 12,000 hz
        else if (freqbuff % 125 == 0) WavFreq[Wnum] = 32000;        // 32,000 hz
        else if (freqbuff % 32955 == 0) WavFreq[Wnum] = 48000;      // 48,000 hz
        else{ WavFile.Close(); return 0; }

        WavFile.SeekToBegin(); // Cut Header
        WavFile.Seek(58,WavFile.GetPosition());

        WavSize[Wnum] = WavFile.GetLength()-58;              // Size of Wave File
        WavData[Wnum] = new BYTE[WavSize[Wnum]];     // Create Wave Data
        WavFile.Read(WavData[Wnum],WavSize[Wnum]);   // Read Data From Wave File
        WavFile.Close();

        if(!IsSnr) WavNum++;   // Increase Number of Wave file

        return 1;
}
```

รูปที่ ก.6 โปรแกรมการทำงานของระบบ

ภาคผนวก ข
คู่มือการใช้งาน

ค.1 คู่มือการใช้งาน

ค.1.1 ลำดับขั้นการทดลองส่วนของการเข้ารหัส – ถอดรหัส MP3

1) ลำดับขั้นการทดลองการเข้ารหัส – ถอดรหัส MP3

1.1) เปิดโปรแกรม Case Study of Vector Quantize Efficiency

1.2) เลือกที่ File ⟶ Encode / Decode ⟶ MP3 เลือก EnDecMp3 ดังรูปที่ ข.1

รูปที่ ข.1 ลำดับขั้นการทดลองการเข้ารหัส – ถอดรหัส MP3

1.3) เมื่อโปรแกรมแสดงขึ้นมาแล้วให้เลือกที่ Add

1.4) Browse ไฟล์ต้นฉบับ . WAV ที่ต้องการเปรียบเทียบ

1.5) เลือก Output ที่ต้องการ

1.6) เลือก Directory ที่ต้องการให้ผลของการเข้ารหัสของสัญญาณไปอยู่ที่ใด

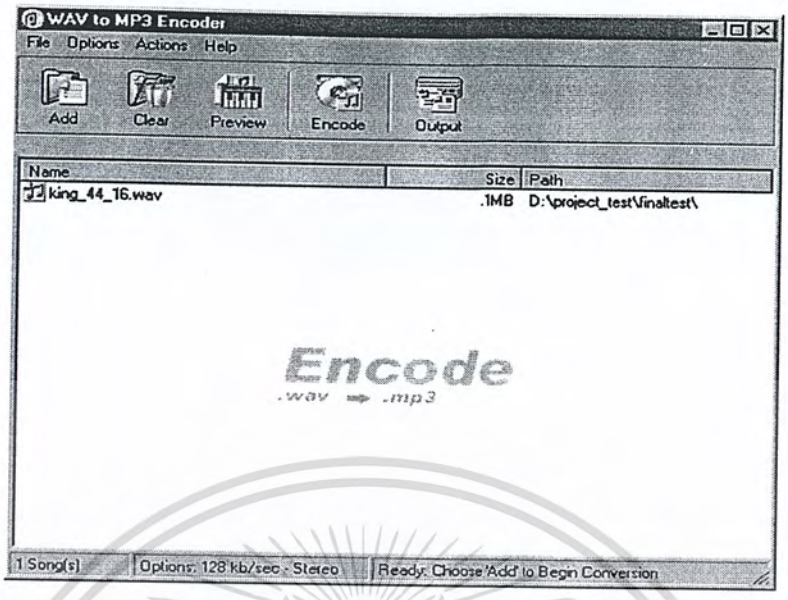
1.7) เลือก Encode เพื่อแปลงจากไฟล์ . WAV เป็น MP3

1.8) ปิดโปรแกรม WAV to MP3 โปรแกรม MP3 to WAV จะปรากฏ

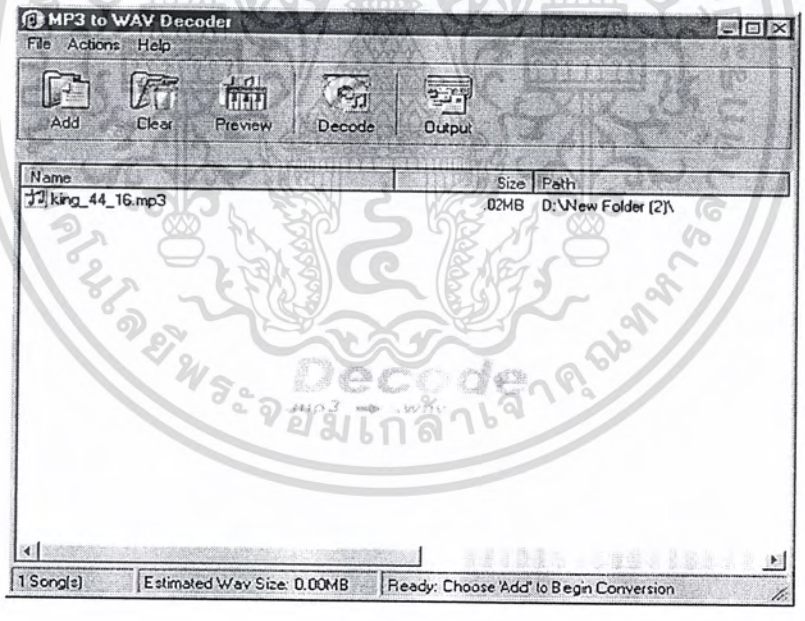
1.9) เลือกที่ Add แล้วBrowseไฟล์ . MP3 ที่ได้จากการ Encode จากโปรแกรมข้างต้น

1.10) เลือก Output ที่ต้องการให้ผลของการถอดรหัสของสัญญาณว่าไปอยู่ที่ใด

1.11) เลือก Decode เพื่อทำการ Converse สัญญาณ . MP3 ให้เป็น . WAV



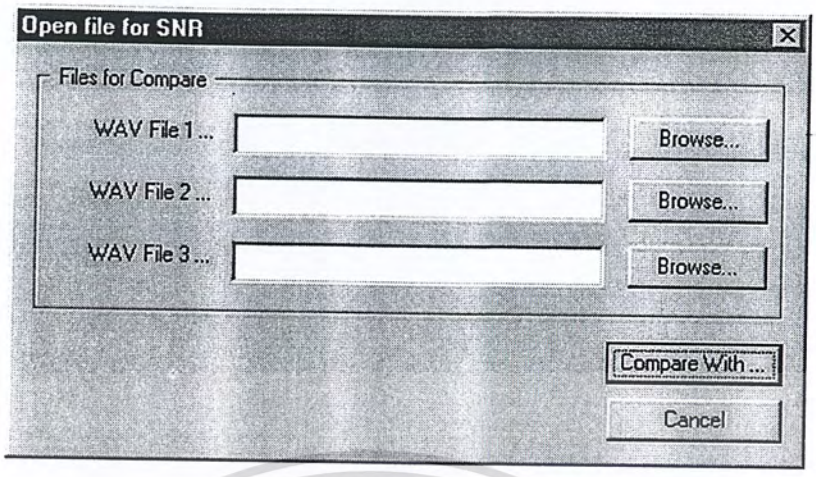
รูปที่ ข.2 โปรแกรม WAV to MP3



รูปที่ ข.3 โปรแกรม MP3 to WAV

1.12) ปิดโปรแกรม MP3 to WAV

1.13) เลือก SNR Compare ⟶ SNR Graph



รูปที่ ข.4 การเปรียบเทียบค่าอัตราส่วนสัญญาณต่อสัญญาณรบกวน

1.14) Browse สัญญาณที่ต้องการทำการเปรียบเทียบ โดยเลือกได้สูงสุด 3 สัญญาณ

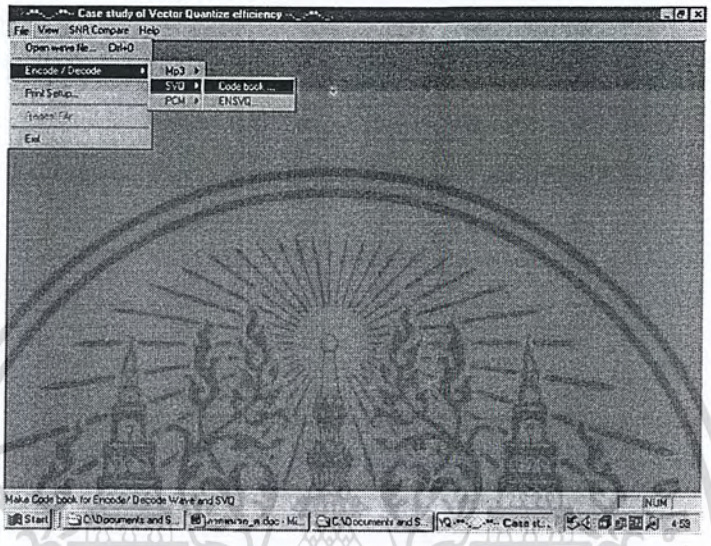
1.15) Browse สัญญาณไฟล์ .WAV ต้นฉบับ โดยเลือกที่ Compare With....

1.16) เลือก OK. กราฟของผลการเปรียบเทียบจะแสดงที่หน้าจอมอนิเตอร์

ค.1.2 ลำดับขั้นการทดลองและผลการทดลองส่วนของการเข้ารหัส – ถอดรหัส SVQ

1) ลำดับขั้นการทดลองของการเข้ารหัส – ถอดรหัส SVQ

1.1) เลือกที่ File ⟶ Encode / Decode ⟶ SVQ⟶ Codebook เพื่อทำการสร้าง Codebook



รูปที่ ข.5 ลำดับขั้นการทดลองของการเข้ารหัส – ถอดรหัส SVQ

1.2) Browse สัญญาณต้นฉบับขึ้นมา

1.3) กำหนดค่าต่างๆ ดังที่กำหนดไว้ดังนี้

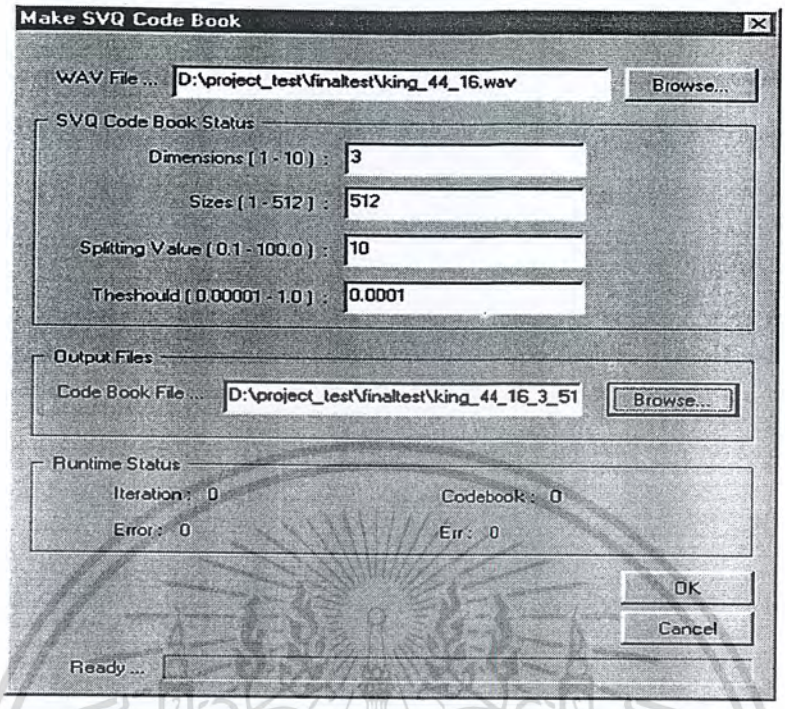
1.3.1) กำหนด Dimension ค่าระหว่าง 1 – 10

1.3.2) กำหนด Size ค่าระหว่าง 1 – 512 (ยิ่งละเอียดมากความถูกต้องของสัญญาณก็จะมีมาก)

1.3.3) กำหนด Splitting Value ค่าระหว่าง 0.1 – 100

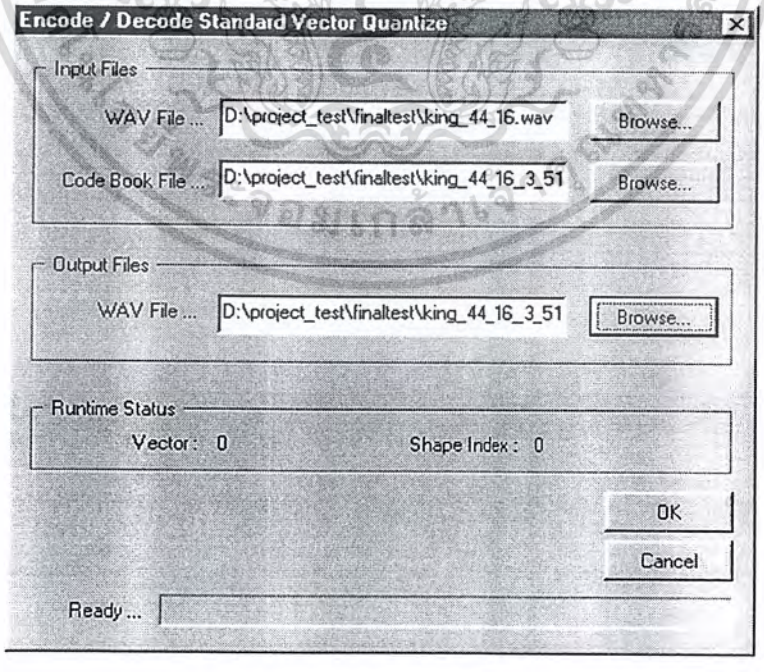
1.3.4) กำหนด Theshold ค่าระหว่าง 0.00001 – 1.0

1.4) กำหนด Output File ของ Codebook เพื่อเก็บ Codebook ที่ทำการสร้างเสร็จแล้ว กด OK.เพื่อทำการสร้าง Codebook



รูปที่ ข.6 การสร้าง Codebook

1.5) เลือกที่ File ⟶ Encode / Decode ⟶ SVQ ⟶ ENSVQ



รูปที่ ข.7 การสร้างสัญญาณจาก Codebook

1.6) เลือก File ต้นฉบับ เพื่อนำไปสร้าง ไฟล์ WAV ใหม่ขึ้นมา

1.7) เลือก Codebook ที่สร้างเสร็จแล้ว

1.8) กำหนด Output File จากการสร้าง ไฟล์ . WAV จาก Codebook

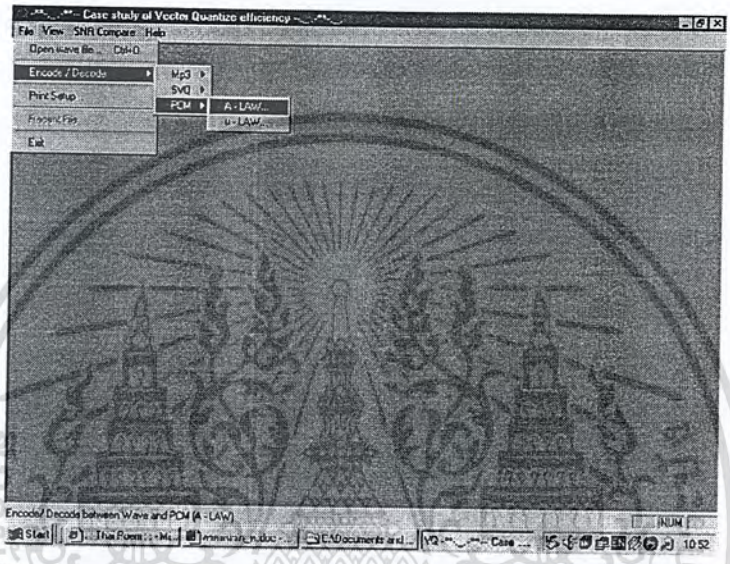
1.9) ทำการเปรียบเทียบอัตราส่วนสัญญาณต่อสัญญาณรบกวน

ค.1.3 ลำดับขั้นการทดลองและผลการทดลองส่วนของการเข้ารหัส – ถอดรหัส PCM แบบ A – Law และ μ - Law

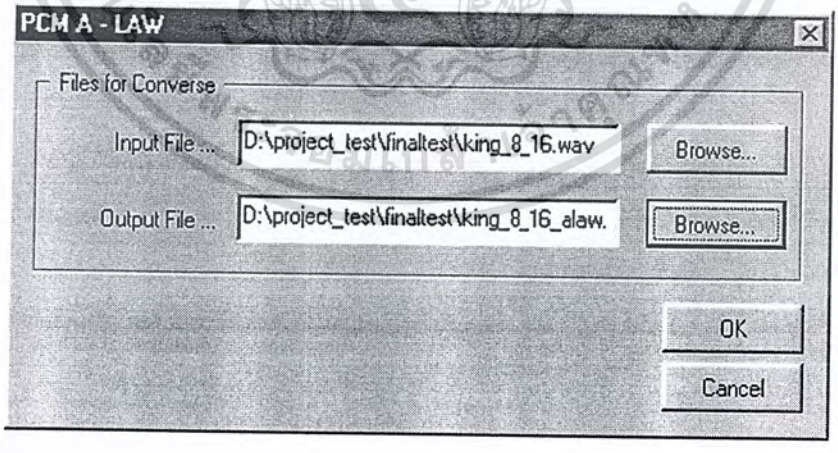
1) ลำดับขั้นการทดลองของการเข้ารหัส – ถอดรหัส PCM แบบ A – Law

1.1) เปิดโปรแกรม Case Study of Vector Quantize Efficiency

1.2) เลือกเข้าส่วนของการเข้ารหัสและถอดรหัส PCM แบบ A – Law



รูปที่ ข.8 ลำดับขั้นการทดลองของการเข้ารหัส – ถอดรหัส PCM แบบ A – Law



รูปที่ ข.9 การเข้ารหัส – ถอดรหัส PCM แบบ A – Law

1.3) Browse ไฟล์ต้นฉบับ . WAV

1.4) เลือก Output ตั้งชื่อเองให้มี .wav ด้วย

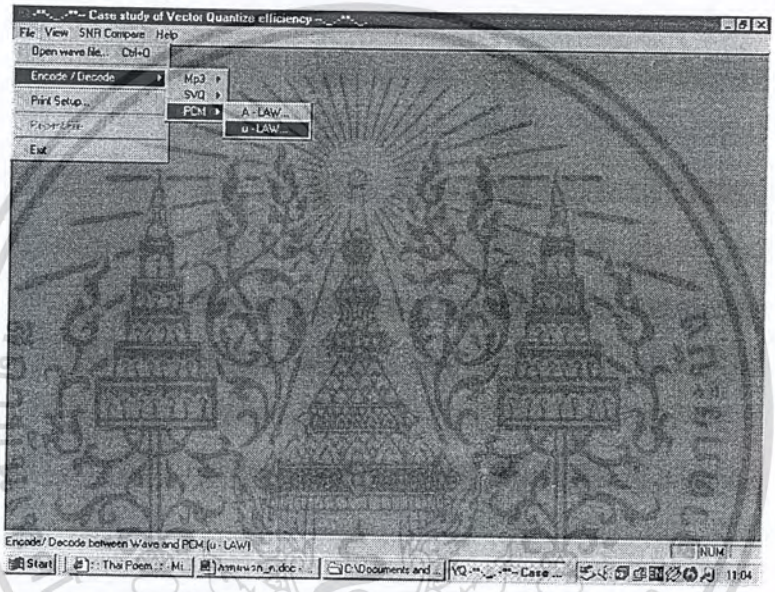
1.5) เลือก OK. แล้วโปรแกรมจะทำการเข้ารหัส – ถอดรหัส ออกมา

1.6) ทำการเปรียบเทียบอัตราส่วนสัญญาณต่อสัญญาณรบกวน

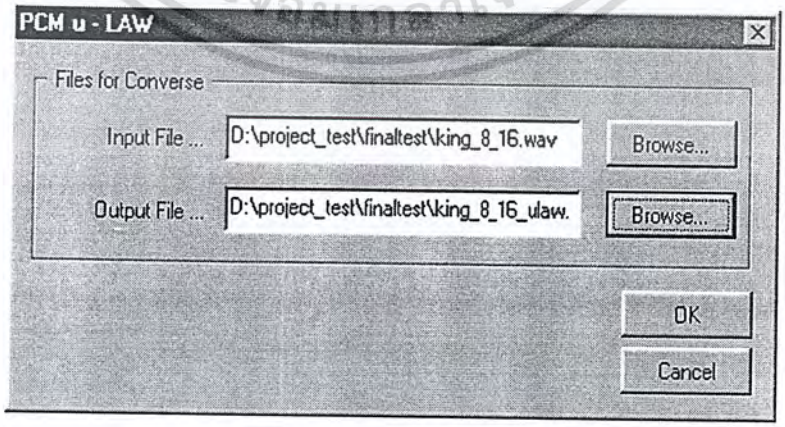
2) ลำดับขั้นการทดลองของการเข้ารหัส – ถอดรหัส PCM แบบ μ - Law

2.1) เปิดโปรแกรม Case Study of Vector Quantize Efficiency

2.2) เลือกเข้าส่วนของการเข้ารหัสและถอดรหัส PCM แบบ μ – Law



รูปที่ ข.10 ลำดับขั้นการทดลองของการเข้ารหัส – ถอดรหัส PCM แบบ μ – Law



รูปที่ ข.11 การเข้ารหัส – ถอดรหัส PCM แบบ μ – Law

2.3) Browse ไฟล์ต้นฉบับ . WAV

2.4) เลือก Output ตั้งชื่อเองให้มี .wav ด้วย

2.5) เลือก OK. แล้วโปรแกรมจะทำการเข้ารหัส – ถอดรหัสออกมา

2.6) ทำการเปรียบเทียบอัตราส่วนสัญญาณต่อสัญญาณรบกวน

บรรณานุกรม

ฑนะ รสยะพาห์. **แลกเปลี่ยน + แบ่งปัน MP3 ผ่าน Napsterเพื่อความบันเทิงแห่งเสียงเพลง:** วิตตี้กรุ๊ป จำกัด. 2543

บัณฑิตโรจน์ อารยานนท์.**หลักการไฟฟ้าสื่อสาร** ชุดความรู้เฉพาะอันดับ 6 . 2538

รัชนัย อินทุใส.**การสื่อสารโทรคมนาคม:** โครงการตำราเรียนPhysics Center. 2541

วรวิทย์ สมหา. **การคอนไตส์เวกเตอร์แบบมาตรฐานที่เวลาจริง.**วิทยานิพนธ์ วิศวกรรมไฟฟ้า สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

วิวัฒน์ กิรานันท์. **วิศวกรรมการสื่อสาร.**คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง. 2540

วิวัฒน์ กิรานนท์. **พื้นฐานการสื่อสาร.** วิศวกรรมสถานแห่งประเทศไทยในพระบรมราชูปถัมภ์. 2538

สุรเดช ตรีไตรลักษณะ. **การใช้กระบวนการเชิงเส้นในการควอนไตซ์ข้อมูลแบบเวกเตอร์.** วิทยานิพนธ์ สาขาวิชาวิศวกรรมไฟฟ้า สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง.2540

Kenny Chan. **สุดขั้วความสะใจกับ MP3 เพื่อความบันเทิงแห่งเสียงเพลง:**วิตตี้กรุ๊ปจำกัด2542

ประวัติผู้แต่ง

ชื่อผู้ทำปริญญานิพนธ์	นางสาวอัสนีย์ ศิริพงษ์
วันเดือนปีเกิด	30 กรกฎาคม พ.ศ.2521
สถานที่เกิด	โรงพยาบาลพุทธชินราช
ภูมิลำเนาเดิม	จังหวัดพิษณุโลก
ที่อยู่ปัจจุบัน	40/1 หมู่ 3 ตำบลเนินกุ่ม
	อำเภอบางกระทุ่ม จังหวัดพิษณุโลก
	65210
โทรศัพท์	0-9533-5454
ประวัติการศึกษา	
ประถมศึกษา	โรงเรียนบ้านท่ายาง
มัธยมศึกษาตอนต้น	โรงเรียนจ่านกร้อง
มัธยมศึกษาตอนปลาย	โรงเรียนเฉลิมขวัญสตรี
ประกาศนียบัตรวิชาชีพชั้นสูง (ปวส.)	วิทยาลัยเทคนิคพิษณุโลก
ปริญญาตรี	สาขาวิชาวิศวกรรมโทรคมนาคม
	ภาควิชาครุศาสตร์วิศวกรรม
	คณะครุศาสตร์อุตสาหกรรม
	สถาบันเทคโนโลยีพระจอมเกล้า
	เจ้าคุณทหารลาดกระบัง
ผลงานที่ได้รับรางวัล	-
ทุนการศึกษา	-
คติพจน์	-

ประวัติผู้แต่ง

ชื่อผู้ทำปริญญานิพนธ์	นางสาวปิยดา ปันชุน
วันเดือนปีเกิด	27 มกราคม พ.ศ.2523
สถานที่เกิด	โรงพยาบาลน่าน
ภูมิลำเนาเดิม	จังหวัดน่าน
ที่อยู่ปัจจุบัน	103 หมู่ 11 ตำบลดู่ใต้
	อำเภอเมือง จังหวัดน่าน
	55000
โทรศัพท์	0-9671-8741

ประวัติการศึกษา

ประถมศึกษา	โรงเรียนบ้านดอน (ศรีเสริมกสิกร)
มัธยมศึกษาตอนต้น	โรงเรียนสตรีศรีน่าน
ประกาศนียบัตรวิชาชีพ (ปวช.)	วิทยาลัยเทคนิคน่าน
ประกาศนียบัตรวิชาชีพชั้นสูง (ปวส.)	วิทยาลัยเทคนิคน่าน
ปริญญาตรี	สาขาวิชาวิศวกรรมโทรคมนาคม
	ภาควิชาครุศาสตร์วิศวกรรม
	คณะครุศาสตร์อุตสาหกรรม
	สถาบันเทคโนโลยีพระจอมเกล้า
	เจ้าคุณทหารลาดกระบัง
ผลงานที่ได้รับรางวัล	-
ทุนการศึกษา	-
คติพจน์	-