การพัฒนาโปรแกรมควบคุมหลอดไฟตามจังหวะของเสียง

TONAL CONTROL LIGHT SYSTEM

เทวินทร์ ธนสถิตย์ชัย
ธีรัช พรปัทมภิญโญ
อภิชาติ แซ่อึ้ง

ปัญหาพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรบัณฑิต
ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2544

TONAL CONTROL LIGHT SYSTEM

TAYWIN TANASATHIDCHAI

TEERUSH PORNPATTAMAPINYO

APICHART SAE-OUNGE

A SPECIAL PROJECT SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF BACHELOR OF SCIENCE

DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE

FACULTY OF SCIENCE

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

ACADEMIC YEAR 2001

หัวข้อปัญหาพิเศษ	การพัฒนาโปรแกรมควบคุมหลอดไฟตามจังหวะของเสียง	
	TONAL CONTROL LIGHT SYSTEM	
ชื่อนักศึกษา	นาย เทวินทร์ ธนสถิตย์ชัย	41056034
	นาย ธีรัช พรปัทมภิญโญ	41056048
	นาย อภิชาติ แซ่อึ้ง	41056141
ภาควิชา	คณิตศาสตร์และวิทยาการคอมพิวเตอร์	
สาขาวิชา	วิทยาการคอมพิวเตอร์	
อาจารย์ที่ปรึกษา	อาจารย์วิสันต์ ตั้งวงษ์เจริญ	

ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อนุมัติให้นับปัญหาพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร วิทยาศาสตรบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์ ประจำปีการศึกษา 2544

คณะกรรมการสอบ		ลายมือชื่อ
ประธานกรรมการ	ผู้ช่วยศาสตราจารย์ธีรวัฒน์ ประกอบผล	
กรรมการ	อาจารย์วีระชัย ตันยะสิทธิ์	
กรรมการและอาจารย์ที่ปรึกษา	อาจารย์วิสันต์ ตั้งวงษ์เจริญ	

(ผู้ช่วยศาสตราจารย์ไพรบูลย์ พันธรักษ์พงษ์)
หัวหน้าภาควิชาวิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์

หัวข้อปัญหาพิเศษ	การพัฒนาโปรแกรมควบคุมหลอดไฟตามจังหวะของเสียง	
ชื่อนักศึกษา	นาย เทวินทร์ ธนสถิตย์ชัย	41056034
	นาย ธีรัช พรปัทมภิญโญ	41056048
	นาย อภิชาติ แซ่อึ้ง	41056141
ปริญญา	วิทยาศาสตรบัณฑิต	
ภาควิชา	คณิตศาสตร์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์	
สาขาวิชา	วิทยาการคอมพิวเตอร์	
ปีการศึกษา	2544	
อาจารย์ที่ปรึกษา	อาจารย์วิสันต์ ตั้งวงษ์เจริญ	

บทคัดย่อ

ปัญหาพิเศษนี้เป็นการพัฒนาโปรแกรมควบคุมหลอดไฟ ตามจังหวะสัญญาณเสียง โดยทำการวิเคราะห์ข้อมูลสัญญาณเสียง เพื่อทำการแปลงข้อมูลสัญญาณเสียงจากรูปแบบแอมปลิจูดต่อเวลา ให้เป็นรูปแบบของขนาดสเปกตรัมในแต่ละความถี่ โดยใช้ทฤษฎีการแปลง Fast Fourier Transform เพื่อนำข้อมูลที่ได้จากการแปลงมาแสดงผลทางหลอดไฟ และทางหน้าจอมอนิเตอร์ โปรแกรมที่พัฒนาขึ้นนี้สามารถทำการเล่นไฟล์เสียงประเภทไฟล์ MP3 และไฟล์ WAVE เพื่อใช้เป็นแหล่งของสัญญาณข้อมูลอินพุท

Special Project Title	Tonal Control Light System	
Students	Mr. Taywin Tanasathidchai	41056034
	Mr. Teerush Pornpattamapinyo	41056048
	Mr. Apichart Sae-ounge	41056141
Degree	Bachelor's Degree of Science	
Department	Mathematics and Computer Science, Faculty of Science	
Programme	Computer Science	
Academic Year	2001	
Special Project Advisor	Lecturer Wisan Tangwongcharoen	

ABSTRACT

This project is developing an application that using Fast Fourier Transform to analyses waveform signals in order to change their format from amplitude - time domain into spectrum of frequency domain. Transformed signals will be displayed on Light output device and on a monitor. In addition the project included developing a sound player which can playback MP3 files type and WAVE files type. Signals from the sound player are inputs for controlling Light process.

กิตติกรรมประกาศ

ในการทำปัญหาพิเศษเรื่องการควบคุมหลอดไฟตามจังหวะของเสียง (Tonal Control Light System) สามารถสำเร็จลงไปได้ด้วยดี คณะผู้จัดทำต้องขอขอบพระคุณ อาจารย์วิสันต์ ตั้งวงษ์เจริญ อาจารย์ผู้รับผิดชอบปัญหาพิเศษฉบับนี้ ที่กรุณาให้คำแนะนำ และเป็นที่ ปรึกษา ในเรื่องแนวทางการพัฒนาโปรแกรม การแก้ไขปัญหาต่างๆ รวมทั้งเป็นผู้ตรวจสอบความถูกต้องของปัญหาฉบับนี้

นอกจากนี้คณะผู้จัดทำขอขอบพระคุณ บิดา มารดา ที่ได้ให้ความสนับสนุนทางด้านกำลังใจ และทุนทรัพย์ จนการทำปัญหาพิเศษนี้สำเร็จด้วยดี รวมทั้งเพื่อนๆทุกคนที่ให้ความช่วยเหลือในด้านต่างๆเกี่ยวกับปัญหาพิเศษนี้ไว้ ณ ที่นี้ด้วย

คณะผู้จัดทำ

มีนาคม 2545

สารบัญ

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญรูป

สารบัญรูป(ต่อ)

รูปที่ **หน้า**

บทที่ 1
บทนำ

1.1 ที่มาของปัญหาพิเศษ

ในปัจจุบันเทคโนโลยีทางด้านมัลติมีเดียได้พัฒนาก้าวหน้าไปมาก โดยเฉพาะเทคโนโลยีทางเสียง และ เทคโนโลยี การเก็บข้อมูลเสียงในรูปดิจิตอล เช่น MPEG Layer ต่างๆ, MIDI เป็นต้น ซึ่งเป็นที่นิยมกันอย่างแพร่หลาย โปรแกรมที่ใช้ในการเล่นไฟล์เสียงเหล่านี้จึงมีการผลิตออกมามาก โดยมีรูปแบบ และลูกเล่นที่หลากหลาย เพื่อดึงดูดผู้ใช้ แต่ยังไม่มีโปรแกรม ที่ใช้ควบคุมอุปกรณ์ที่ใช้แสดงผลภายนอกที่แยกออกมาจากการแสดงผลบนหน้าจอคอมพิวเตอร์ เช่น อุปกรณ์แสดงผลทาง Equalizer กล่าวคือเป็นอุปกรณ์เสริมประเภท หลอด LED หรืออื่นๆ เพื่อเพิ่มความเพลิดเพลินและสร้างบรรยากาศ โดยอุปกรณ์นี้จะต้องทำงานร่วมกับโปรแกรมที่ใช้ในการเล่นไฟล์ทางดนตรีได้ จึงเป็นที่มาของการพัฒนาโปรแกรมควบคุมหลอดไฟตามจังหวะของเสียง ในการนี้จะต้องศึกษาครอบคลุมถึงลักษณะ, รูปแบบของไฟล์ mp3 และรวมไปถึงข้อมูลที่ถูกเก็บอยู่ในไฟล์ mp3 ด้วย

1.2 วัตถุประสงค์ของปัญหาพิเศษ

1.2.1 เพื่อพัฒนาโปรแกรมควบคุมอุปกรณ์แสดงผลทางทางหลอดไฟภายนอก ซึ่งโปรแกรมดังกล่าวสามารถเชื่อมต่อกับโปรแกรมที่ใช้ในการเล่นไฟล์ mp3

1.2.2 ศึกษาลักษณะ , รูปแบบ และ ข้อมูลที่ถูกเก็บอยู่ในไฟล์ mp3

1.2.3 ศึกษาการถอดรหัสและเข้ารหัสไฟล์ mp3

1.2.4 ศึกษาการใช้งาน API(Application Programming Interface) โดยใช้ Visual Basic 6.0

1.2.5 ศึกษาฟังก์ชันเกี่ยวกับเสียงของ DirectX 8.0

1.2.6 ศึกษาการเชื่อมต่ออุปกรณ์แสดงผลทางพอร์ตขนาน

1.3 ขอบเขตของปัญหาพิเศษ

ปัญหาพิเศษนี้จะทำการถอดรหัสข้อมูลไฟล์ mp3 และไฟล์เสียง (wave) มาโดยใช้ฟังก์ชันการทำงานของวินโดว์ (Application Programming Interface) ซึ่งถูกเรียกใช้โดย Visual Basic 6.0 และ ฟังก์ชันเกี่ยวกับเสียงภายใน DirectX 8.0 จากนั้นจะนำข้อมูลที่ได้มาทำการวิเคราะห์และทำการประมวลผลข้อมูลเสียงที่ได้นี้ โดยใช้ทฤษฎี การแปลงฟูเรียร์อย่างเร็ว(Fast Fourier Transform)เพื่อจะสามารถนำข้อมูลที่ผ่านการวิเคราะห์และประมวลผลนี้ ออกไปแสดงผลลัพธ์ผ่านทางพอร์ตขนานที่ซึ่งทำการเชื่อมต่อกับอุปกรณ์ภายนอก(หลอด LED)ที่จะทำการแสดงผลตามจังหวะและความดังของเสียง

1.4 ระยะเวลาในการดำเนินงานปัญหาพิเศษ

ระยะเวลาในการดำเนินงานปัญหาพิเศษมีรายละเอียดดังนี้

ตารางที่ 1.1 แสดงระยะเวลาดำเนินงานปัญหาพิเศษ

วัน/ เดือน/ ปี	การทำงาน
1 มิ.ย. – 11 มิ.ย. 44	ศึกษาปัญหา และที่มาของหัวข้อปัญหาพิเศษ
12 มิ.ย. – 12 ก.ค. 44	รวบรวมข้อมูลในเรื่องของไฟล์ mp3
13 ก.ค. – 30 ก.ย. 44	ศึกษาการเข้ารหัส และถอดรหัสไฟล์ mp3
1 ต.ค. – 14 ต.ค. 44	จัดทำเอกสารประกอบโครงงานพิเศษ
15 ต.ค. – 25 ต.ค. 44	ศึกษาวิธีการเรียกใช้งาน API บน Visual Basic
26 ต.ค. – 8 พ.ย. 44	ศึกษาการใช้งานผ่าน DirectX, DirectSound
9 พ.ย. – 25 พ.ย. 44	ศึกษาทฤษฎีการแปลงฟูเรียร์อย่างเร็ว (FFT)
26 พ.ย. – 1 ธ.ค. 44	ศึกษาวิธีการเชื่อมต่อกับวงจรหลอดไฟ LED ผ่านทางพอร์ตขนาน
2 ธ.ค. – 10 ม.ค. 45	ออกแบบโปรแกรม และทำการเขียนโปรแกรม
11 ม.ค. – 21 ม.ค. 45	ออกแบบอินเตอร์เฟส ของโปรแกรม
22 ม.ค. – 25 ก.พ. 45	ทดสอบ และปรับปรุงโปรแกรม
26 ก.พ. 45 – 6 มี.ค. 45	จัดทำเอกสารประกอบโครงงานพิเศษ

1.5 ขั้นตอนการดำเนินงาน

1.5.1 ศึกษาหาข้อมูลเกี่ยวกับไฟล์ mp3 และการเชื่อมต่อกับอุปกรณ์ภายนอก

1.5.2 ศึกษาทฤษฎีต่างๆมี่เกี่ยวข้องกับไฟล์ mp3 เช่น ทฤษฎีการบีบอัดไฟล์ข้อมูล , การเข้ารหัสการถอดรหัสไฟล์ mp3 และโครงสร้างของไฟล์ mp3 เป็นต้น

1.5.3 ศึกษาการใช้งาน API และฟังก์ชันที่เกี่ยวกับเสียงต่างๆ ของ Visual Basic 6.0 และ DirectX 8.0

1.5.4 ศึกษาการวิเคราะห์ข้อมูลเสียงโดยใช้การแปลงฟูเรียร์อย่างเร็ว (Fast Fourier Transform)

1.5.5 ศึกษาการเชื่อมต่อกับอุปกรณ์แสดงผลภายนอกผ่านทางพอร์ตขนาน

1.5.6 ออกแบบระบบการทำงาน

1.5.7 ทำการเขียนโปรแกรม

1.5.8 ทดสอบระบบงานจริง และทำการปรับปรุงระบบงาน เป็นการทดลองใช้กับระบบงานจริง เพื่อดูความเหมาะสม และเพื่อทำการปรับปรุงให้ตรงกับความต้องการ

1.5.9 จัดทำเอกสารการวิจัยสรุปผลการทดสอบ ทำการสร้างเอกสารประกอบและเอกสารอ้างอิงในการทำปัญหาพิเศษ

1.6 ประโยชน์ที่คาดว่าจะได้รับ

1.6.1 เพื่อเป็นแนวทางในการพัฒนาโปรแกรมด้านมัลติมีเดียอื่นๆ

1.6.2 เพื่อเป็นแนวทางในการพัฒนาโปรแกรมเชื่อมต่อกับอุปกรณ์ภายนอกแบบอื่นๆ

1.6.3 เพื่อเรียนรู้วิธีการวิเคราะห์และแยกแยะข้อมูลเสียงต่างๆ โดยใช้ ทฤษฎีการแปลงฟูเรียร์อย่างเร็ว (Fast Fourier Transform)

1.6.4 เพื่อเรียนรู้วิธีการการควบคุมอุปกรณ์แสดงผลภายนอก

1.7 ข้อตกลงเบื้องต้น

1.7.1 ระบบควบคุมหลอดไฟตามจังหวะเสียงนี้ ต้องได้รับการสนับสนุนจากการ์ดเสียงที่สามารถทำการ recording ได้เท่านั้น

1.7.2 ระบบที่พัฒนาขึ้นนี้ไม่ได้ใช้การถอดรหัสไฟล์ mp3 โดยตรง ดังนั้นคุณภาพของเสียงที่ ได้อาจไม่ดีเทียบเท่าโปรแกรมที่ใช้ในการเล่นเพลงอื่นๆ

1.8 อุปกรณ์ที่ใช้ในการทำปัญหาพิเศษ

1.8.1 เครื่องคอมพิวเตอร์ Pentium II 350 MHz

1.8.2 หน่วยความจำ 64 MB

1.8.3 ฮาร์ดดิสก์ ความจุ 4.3 GB

1.8.4 การ์ดเสียง Yamaha S-YXG50

1.8.5 ไดร์ฟซีดีรอม 36 x

1.8.6 ลำโพง

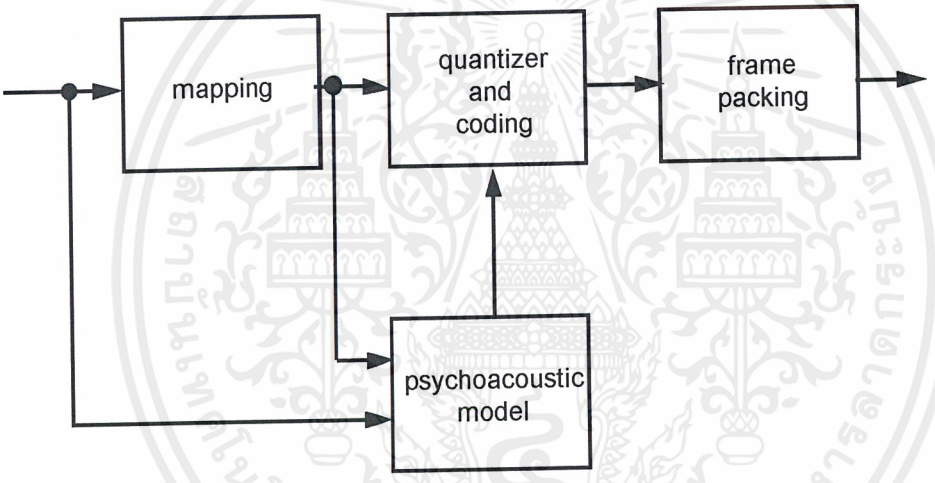
1.8.7 บอร์ดเชื่อมต่อพอร์ตขนาน P-Board

1.8.8 บอร์ด LED มอนิเตอร์ 16 ช่อง EX-01

บทที่ 2
ทฤษฎีและหลักการที่เกี่ยวข้อง

2.1 การเข้ารหัส

กระบวนการเข้ารหัสข้อมูลสัญญาณเสียงและสร้างการบีบอัดก่อนที่จะเก็บรวบรวมเป็นไฟล์ อัลกอริทึ่มที่ใช้ในการเข้ารหัสนั้นไม่มีการกำหนดมาตรฐาน และอาจจะใช้รูปแบบที่หลากหลายในการเข้ารหัสเช่น การประมาณของออดิทอรีมาสกิงเทรสโฮลด์ , ควอนไตซ์เซชันและ การสเกลลิ่ง อย่างไรก็ตามผลลัพธ์ที่ได้จากการเข้ารหัสก็จะต้องถูกถอดรหัสได้ด้วยตัวถอดรหัสใดๆตามรูปแบบของกระบวนการถอดรหัส ซึ่งจะทำการสร้างข้อมูลเสียงที่ยอมรับได้สำหรับแอปพลิเคชั่นตามที่ต้องการ ซึ่งสามารถแสดงกระบวนการเข้ารหัสพื้นฐานได้ดังแสดงในรูปที่ 2.1



รูปที่ 2.1 แสดงกระบวนการเข้ารหัสพื้นฐาน

ข้อมูลเสียงที่ถูกสุ่มมาก็จะเข้าสู่กระบวนการเข้ารหัส การแมปปิ้ง ทำการสร้างตัวกรองและทำการสุ่มข้อมูลย่อยเพื่อเข้ามาแทนข้อมูลเสียงที่เข้ามา การแมป แบบสุ่มอาจจะถูกเรียกได้สองแบบ คือ การสุ่มส่วนย่อย (subband sample) เช่น ในเลเยอร์ที่ 1 ด้านล่าง หรือ การแปลงการสุ่มส่วนย่อย เช่น ในเลเยอร์ที่ 3 รูปแบบไซโคอคูสติก (psychoacoustic model) ทำการสร้างเซตของข้อมูลเพื่อทำการควบคุม การควอนไตซ์เซอร์ และการเข้ารหัส ข้อมูลเหล่านี้จะแตกต่างกันไปขึ้นอยู่กับการประยุกต์ใช้คำสั่งต่างๆ ความเป็นไปได้อย่างหนึ่งนั่นก็คือการใช้ การประมาณของออดิทอรีมาสกิงเทรสโฮลด์ เพื่อที่จะทำการควบคุมการการควอนไตซ์เซอร์

การควอนไตซ์เซอร์ และส่วนของการเข้ารหัส จะทำการสร้างเซตของสัญลักษณ์การเข้ารหัส จากการแมป กับข้อมูลสุ่มที่ส่งเข้ามา และส่วนของการเข้ารหัสนี้ก็จะขึ้นกับระบบการเข้ารหัสด้วย ในส่วนของเฟรมแพคกิ้ง ทำการรวบรวมบิตสตรีม จากข้อมูลที่เป็นผลลัพธ์ของส่วนอื่นๆ และอาจจะมีส่วนของข้อมูลที่เพิ่มเติม ถ้าจำเป็น เช่น การตรวจสอบความถูกต้อง (error correction)

2.2 เลเยอร์ต่างๆของ MPEG 1

ขึ้นกับแอปพลิเคชัน แต่ละเลเยอร์ ก็จะมีระบบการเข้ารหัสที่แตกต่างกันทำให้ความซับซ้อนและประสิทธิภาพของกระบวนการเข้ารหัสเพิ่มขึ้น ในมาตรฐาน ISO ตัวถอดรหัส MPEG Layer ที่ N ต้องสามารถทำการถอดรหัสข้อมูลบิตสตรีม ซึ่งถูกทำการเข้ารหัสในเลเยอร์ที่ N และเลเยอร์ ที่ต่ำกว่าลงไปได้ด้วย

2.2.1 เลเยอร์ที่1

ในเลเยอร์นี้ประกอบไปด้วยการแมปปิง พื้นฐานระหว่างข้อมูลเสียงที่เข้ามาในรูปแบบ 32 ซับแบนด์ , การกำหนดการแบ่งข้อมูลออกเป็นส่วนย่อยๆ (block) , รูปแบบไซโคอคูสติก (psychoacoustic model) ที่จะกำหนดและจัดการกับบิตต่างๆ และทำการควอนไตซ์เซชัน โดยการใช้การคอมแพนดิง และรูปแบบของส่วนย่อย (block)

2.2.2 เลเยอร์ที่2

ในเลเยอร์ นี้ทำการเพิ่มเติมการเข้ารหัสในส่วนของ การกำหนดบิต (bit allocation), สเกลแฟคเตอร์ และ การสุ่มมีการใช้เฟรมที่แตกต่างกันด้วย

2.2.3 เลเยอร์ที่3

ในเลเยอร์ นี้มีการเพิ่มความถี่ที่ใช้แสดงผลบนพื้นฐานของ ไฮบริดฟิลเตอร์แบงค์ ซึ่งทำการเพิ่มการควอนไตซ์เซอร์ที่แตกต่างออกไป (nonuniform quantizer), การพัฒนาการแบ่งเป็นส่วนย่อย และการเข้ารหัสของค่าการการควอนไตซ์เซอร์ ในส่วนของรูปแบบการเข้ารหัสจอยท์สเตริโอ ก็สามารถถูกเพิ่มเติมเข้าไปในเลเยอร์ใดๆ ก็ได้

2.3 คลังข้อมูล (Storage)

สตรีมที่หลากหลายของข้อมูลภาพและเสียงที่ถูกเข้ารหัส, ข้อมูลที่เข้าจังหวะ (synchronization data), ข้อมูลของระบบ และข้อมูลที่ช่วยสนับสนุน (auxiliary data) อาจจะถูกจัดเก็บไว้ด้วยกันในคลังข้อมูลส่วนกลาง (storage medium) การเปลี่ยนแปลงข้อมูลเสียงจะสามารถทำได้ง่ายขึ้นถ้าจุดที่ต้องการเปลี่ยนแปลงนั้นตรงกับจุดที่อยู่ตรงแอดเดรสนั้น

การเข้าถึงคลังข้อมูล (storage) อาจจะใช้การเข้าถึงข้อมูลระยะไกล (remote access) บนระบบการติดต่อสื่อสาร การเข้าถึงข้อมูลจะถูกควบคุมโดยส่วนควบคุมอื่นๆ ที่ไม่ได้อยู่ในตัวถอดรหัสเสีย ซึ่งส่วนควบคุมนี้จะรองรับคำสั่ง, อ่าน และทำการแปลข้อมูลบนพื้นฐานโครงสร้างข้อมูล, อ่านข้อมูลที่ถูกเก็บไว้ในสื่อต่างๆ , ทำการดีมัลติเพล็กซ์ ข้อมูลที่ไม่ใช่เสียง และส่งข้อมูลบิตสตรีมของเสียงที่ถูกเก็บไว้ไปยังตัวถอดรหัสตามอัตรา (rate) ที่ต้องการ

2.4 การถอดรหัส (Decoding)

ตัวถอดรหัสต้องใช้รูปแบบของกระบวนการถอดรหัส เพื่อสามารถยอมรับข้อมูลเสียงที่ถูกบีบอัดมาได้, ทำการถอดรหัสข้อมูล และใช้ข้อมูลที่ถอดรหัสมาได้นั้นไปสร้างเป็นข้อมูลเสียงที่เป็นผลลัพธ์ตามต้องการต่อไป ซึ่งสามารถแสดงกระบวนถอดรหัสพื้นฐานได้ดังแสดงในรูปที่ 2.2

รูปที่ 2.2 แสดงกระบวนการถอดรหัสพื้นฐาน

ข้อมูลบิตสตรีมจะถูกส่งเข้าไปในตัวถอดรหัส แล้วข้อมูลบิตสตรีมก็จะเข้าสู่กระบวนการอันแพ็กกิง และถอดรหัสส่วนย่อย (block) ทำการตรวจข้อผิดพลาดถ้าในการเข้ารหัสมีการใช้ส่วนของการตรวจสอบข้อผิดพลาด ซึ่งข้อมูลบิตสตรีมที่ถูกอันแพ็ก แล้วก็จะทำการรวบรวมให้เป็นส่วนของข้อมูล หลังจากนั้นจะเข้าสู่กระบวนการรีคอนสตรักชัน ซึ่งจะทำการนำรูปแบบของการควอนไตซ์ ของเซตของข้อมูลที่ถูกสุ่มขึ้นมาทำการสร้างใหม่ จากนั้นก็ทำกระบวนการแมปปิง ย้อนกลับแปลงข้อมูลสุ่มให้กลับเป็นรูปแบบของ PCM

2.5 ลักษณะของไฟล์ MPEG 1 Layer 3 (MP3)

หลังจากการเข้ารหัสไฟล์ข้อมูลเสียงในรูปแบบ MPEG 1 layer 3 จะได้ข้อมูลที่มีรูปแบบเฉพาะตามมาตราฐาน ISO/IEC 11172-3 ซึ่งมีรายละเอียดดังนี้ ภายในไฟล์ MPEG Audio นั้นไม่มีส่วนต้นไฟล์ (header file) ที่เป็นส่วนหลักเหมือนไฟล์อื่นๆ เนื่องจากไฟล์ MPEG Audio ถูกสร้างขึ้นโดยการนำส่วนประกอบย่อยๆ ที่เรียกว่า เฟรม มารวมกัน โดยเฟรมนั้นเป็นกลุ่มของข้อมูลซึ่งมีส่วนต้นเฟรม (header frame) และข้อมูลเสียงอยู่ภายใน

โดยในส่วนของ Layer I / II ในแต่ละเฟรมนั้นเป็นอิสระต่อกัน เป็นผลทำให้เราสามารถตัดส่วนใดส่วนหนึ่งของ MPEG ไฟล์แล้วยังสามารถทำการเล่นไฟล์เหล่านั้นได้ โดยโปรแกรมที่เล่นจะทำการเล่นเพลงโดยเริ่มต้นจากเฟรมที่ถูกต้องที่ถูกพบเป็นเฟรมแรก อย่างไรก็ตามในกรณีของ Layer III แต่ละเฟรมนั้นไม่เป็นอิสระต่อกัน ทำให้มีการใช้ไบต์รีเสิร์ฟวา ซึ่งเป็นบัฟเฟอร์ชนิดหนึ่งเนื่องมาจากการที่แต่ละเฟรมนั้นไม่เป็นอิสระต่อกันนั่นเอง ในกรณีแย่ที่สุดของการถอดรหัส อาจจะจำเป็นต้องใช้เฟรม 9 เฟรมในการถอดรหัสเฟรมหนึ่งเฟรม

เมื่อเราต้องการอ่านข้อมูลจาก ไฟล์ MPEG เราจำเป็นต้องค้นหาเฟรมเริ่มต้น ทำการอ่านส่วนต้นเฟรม (header) และกระทำซ้ำอย่างนี้กับเฟรมอื่นๆ แต่ก็ไม่แน่เสมอไป เช่นอาจถูกต่อต้านจากไฟล์ VBR (Variable Bitrate) ในไฟล์ VBR แต่ละเฟรมอาจจะมีบิตเรท ที่แตกต่างกัน ซึ่งจะถูกใช้ในกรณีที่ต้องการให้คุณภาพของเสียงคงที่ในการเล่นไฟล์นั้นๆ โดยเป็นผลมาจากการที่ต้องรองรับการเข้ารหัสเสียงดนตรีที่ต้องการใช้จำนวนบิทที่มากขึ้น

ในส่วนต้นเฟรม (header frame) นั้น มีความยาว 32 bits (4 Bytes) โดยใน11 บิตแรก (อาจจะเป็น 12 บิตในกรณีของ MPEG version 2.5 extension) ของส่วนต้นเฟรม (header frame) จะต้องถูกตั้งค่าให้เป็น 1 เสมอทุกๆ บิต เรียกว่าเป็น เฟรมซิงค์ ในส่วนต้นเฟรมอาจจะมีส่วนของ CRC (Cyclic Redundancy Check) ด้วย โดยจะมีความยาว 16 บิต ซึ่งถ้ามีจะอยู่หลังจากส่วนต้นเฟรม (header frame) และหลังจาก CRC checksum จะเป็นส่วนของข้อมูลเสียง และเราสามารถตรวจสอบความถูกต้องโดยการคำนวณ CRC และเปรียบเทียบกับข้อมูลว่า เฟรมมีการเปลี่ยนแปลงในระหว่างการส่งบิตสตรีม หรือไม่

2.5.1 ส่วนต้นเฟรม (Header Frame)

สามารถแสดงแผนภาพคร่าวๆ ของส่วนประกอบของส่วนต้นเฟรม (frame header) ได้ดังนี้

AAAA AAAA AAAB BCCD EEEE FFGH IIJJ KLMM

ซึ่งตัวอักษร A–M ใช้แสดงถึงข้อมูลที่แตกต่างกันในแต่ละฟิลด์ โดยจะอธิบายรายละเอียดของแต่ละฟิลด์ได้ดังนี้

ตัวอักษร A หรือ บิตที่ 31-21 (จำนวน 11 บิต) แทน เฟรมซิงค์ ซึ่งทุกบิตจะถูกกำหนดให้เป็น 1

ตัวอักษร B หรือ บิตที่ 20-19 (จำนวน 2 บิต) แทน รหัสของเวอร์ชัน MPEG Audio ซึ่งสามารถแสดงได้ดังตารางที่ 2.1

ตารางที่ 2.1 แสดงรหัสแทนเวอร์ชันของ MPEG Audio

บิตที่ 20-19	เวอร์ชันของ MPEG Audio
00	MPEG เวอร์ชัน 2.5
01	Reserved
10	MPEG เวอร์ชัน 2 (ISO/IEC 13818-3)
11	MPEG เวอร์ชัน 1 (ISO/IEC 11172-3)

ตัวอักษร C หรือ บิตที่ 18-17 (จำนวน 2 บิต) แทน รหัสของเลเยอร์ ซึ่งสามารถแสดงได้ ดัง ตารางที่ 2.2

ตารางที่ 2.2 แสดงรหัสแทนเลเยอร์ของ MPEG Audio

บิตที่ 18-17	เลเยอร์ของ MPEG Audio
00	Reserved
01	Layer III
10	Layer II
11	Layer I

ตัวอักษร D หรือ บิตที่ 16 (จำนวน 1 บิต) แทน บิตโปรเทคชัน ค่าของบิตมีความหมาย ดังตารางที่ 2.3

ตารางที่ 2.3 แสดงความหมายของบิตโปรเทคชัน

บิตที่ 16	ความหมายของบิตโปรเทคชัน
0	มีส่วนขยายสำหรับตรวจสอบข้อผิดพลาด คือมีส่วนของCyclicRedundancy Check –16 bits
1	ไม่มีส่วนขยายสำหรับตรวจสอบข้อผิดพลาด

ตัวอักษร E หรือ บิตที่ 15-12 (จำนวน 4 บิต) แทน ค่าบิตเรทของ MPEG ซึ่งสามารถแสดงได้ดังตารางที่ 2.4

ตารางที่ 2.4 รหัสแทนบิตเรท ของ MPEG เวอร์ชันต่างๆ ภายในส่วนต้นเฟรม (header frame)

บิตที่ 15 –12	MPEG v.1 Layer 1	MPEG v.1 Layer 3	MPEG v.1 Layer 3	MPEG v.2 Layer 1	MPEG v.2 Layer 2, 3
0000	Free	Free	Free	Free	Free
0001	32	32	32	32	8
0010	64	48	40	48	16
0011	96	56	48	56	24
0100	128	64	56	64	32
0101	160	80	64	80	40
0110	192	96	80	96	48
0111	224	112	96	112	56
1000	256	128	112	128	64
1001	288	160	128	144	80
1010	320	192	160	160	96
1011	352	224	192	176	112
1100	384	256	224	192	128
1101	416	320	256	224	144
1110	448	384	320	256	160
1111	Bad	Bad	Bad	Bad	Bad

<u>หมายเหตุ</u> – Free คือ รูปแบบอิสระ หรือค่าบิตเรทที่กำหนดเอง แต่อัตราจะต้องคงที่ตลอดและต้องน้อยกว่าอัตราสูงสุดที่รองรับได้ โดยที่ตัวถอดรหัสไม่จำเป็นต้องรองรับการถอดรหัสของบิตเรทนั้น

Bad คือ ไม่มีการกำหนดไว้ ซึ่งถ้าหากค่าที่ได้เป็น Bad แล้วถือว่าข้อมูลนั้นผิดพลาด

ตารางที่ 2.5 แสดงความเป็นไปได้ในการใช้เทคนิคต่างๆ ในแต่ละบิตเรท

ค่าบิตเรท	หนึ่งช่องทาง	สเตริโอ	อินเทนซิตี สเตริโอ	สองช่องทาง
Free	O	O	O	O
32	O	X	X	X
48	O	X	X	X
56	O	X	X	X
64	O	O	O	O
80	O	X	X	X
96	O	O	O	O
112	O	O	O	O
128	O	O	O	O
160	O	O	O	O
192	O	O	O	O
224	X	O	O	O
256	X	O	O	O
320	X	O	O	O
384	X	O	O	O

ตัวอักษร F หรือ บิตที่ 11-10 (จำนวน 2 บิต) แทน อัตราความถี่ของการสุ่ม (sampling rate frequency) ซึ่งสามารถแสดงได้ดังตารางที่ 2.6

ตารางที่ 2.6 อัตราความถี่ของการสุ่มของ MPEG เวอร์ชันต่างๆ ภายในส่วนต้นเฟรม

บิตที่ 11 – 10	อัตราความถี่ของการสุ่ม (หน่วย : เฮิร์ต (Hz))		
	MPEG v.1	MPEG v.2	MPEG v.2.5
00	44100	22050	11025
01	48000	24000	12000
10	32000	16000	8000
11	Reserved	Reserved	Reserved

ตัวอักษร G หรือ บิตที่ 9 (จำนวน 1 บิต) แทน บิตแพดดิง ดังตารางที่ 2.7

ตารางที่ 2.7 แสดงความหมายของบิตแพดดิง

บิตที่ 9	ความหมายของบิตโปรเทคชัน
0	ไม่มีการเพิ่มสล๊อตพิเศษ
1	มีการเพิ่มสล๊อตพิเศษ เพื่อให้ได้บิตเรทที่ต้องการ

สำหรับการเพิ่มสล๊อตนี้ จะจำเป็นกับอัตราความถี่ของการสุ่มที่ 11.025 kHz, 22.05 kHz, 44.1 kHz และ ในรูปแบบอิสระที่กำหนดเอง

ตัวอักษร H หรือ บิตที่ 8 (จำนวน 1 บิต) แทน บิตไพรเวท เป็นบิตเฉพาะตัว ซึ่งในอนาคต ISO/IEC จะไม่สนับสนุนแล้ว

ตัวอักษร I หรือ บิตที่ 7-6 (จำนวน 2 บิต) แทน บิตโหมด ใช้บอกโหมดแสดงผล ดังตารางที่ 2.8

ตารางที่ 2.8 แสดงรหัสบอกโหมดแสดงผล

บิตที่ 7-6	โหมดแสดงผล
00	สเตริโอ
01	จ๊อยท์สเตริโอ
10	สองช่องทาง
11	หนึ่งช่องทาง

ตัวอักษร J หรือ บิตที่ 5-4 (จำนวน 2 บิต) แทน ส่วนขยายโหมด (mode extension) ใช้ บ่งบอกข้อมูลเพิ่มเติมในโหมดจอนท์สเตริโอเท่านั้น โดยจะบอกว่าใช้วิธีในการเข้ารหัสการจ๊อยท์สเตริโอแบบใดบ้าง ดังตารางที่ 2.9

ตารางที่ 2.9 แสดงรหัสแทนการเข้ารหัสจ๊อยท์สเตอริโอแบบต่างๆ

บิตที่ 5-4	แบบอินเทนซิตี	แบบ ms
00	ไม่ใช้	ไม่ใช้
01	ใช้	ไม่ใช้
10	ไม่ใช้	ใช้
11	ใช้	ใช้

ตัวอักษร K หรือ บิตที่ 3 (จำนวน 1 บิต) แทน การบ่งบอกการมีลิขสิทธิ์ ดังตารางที่ 2.10

ตารางที่ 2.10 แสดงการมีลิขสิทธิ์

บิตที่ 3	แสดงการมีลิขสิทธิ์
0	ไม่มีการป้องกันทางลิขสิทธิ์
1	มีการถูกป้องกันทางลิขสิทธิ์

ตัวอักษร L หรือ บิตที่ 2 (จำนวน 1 บิต) แทน การเป็นต้นฉบับกับสำเนา ดังตารางที่ 2.11

ตารางที่ 2.11 แสดงการเป็นต้นฉบับหรือสำเนา

บิตที่ 2	การเป็นต้นฉบับหรือสำเนา
0	การเป็นสำเนา
1	การเป็นต้นฉบับ

ตัวอักษร M หรือ บิตที่ 1-0 (จำนวน 2 บิต) แทน การเอ็มฟราซิส ซึ่งบ่งบอกว่าต้องทำการดี-เอ็มฟราไซซ์ แบบใด ซึ่งในปัจจุบันไม่นิยมใช้แล้ว โดยจะใช้ 00 แสดงว่าไม่มีการเอ็มฟราซิส

2.5.2 ข้อมูลเสียง (Audio Data)

ข้อมูลเสียงประกอบด้วยข้อมูลร่วม (side information) และ ข้อมูลหลัก โดยข้อมูลหลัก (main data) จะมี ความยาวไม่แน่นอน และขนาดของข้อมูลร่วมขึ้นอยู่กับโหมด และเวอร์ชันของ MPEG ดังตารางที่ 2.12

ตารางที่ 2.12 แสดงขนาดของข้อมูลเสียงในโหมดต่างๆ

โหมด	เวอร์ชัน	ขนาด
หนึ่งช่องทาง	MPEG v.1	17 bytes
	MPEG v.2 , 2.5	9 bytes
โหมดอื่น ๆ (Dual channel , Stereo	MPEG v.1	32 bytes
, Joint Stereo)	MPEG v. 2 , 2.5	17 bytes

2.6 การใช้ API (Application Programming Interface) ต่างๆ

API เป็นลิบราลี ของวินโดว์ ซึ่งเป็นที่เก็บฟังก์ชันการทำงานต่างๆ ของวินโดว์ โดยแอพลิเคชัน ต่างๆ ที่ทำงานอยู่บนวินโดว์ สามารถเรียกไปใช้งานเพื่อทำงานใดๆ ตามแต่หน้าที่ และการทำงานของฟังก์ชัน หรือโพรซีเยอร์ ซึ่งเก็บอยู่ในลิบรารี นั้นๆ สามารถทำได้ โดย API นั้นมักจะถูกเก็บอยู่ในรูปแบบไฟล์นามสกุล DLL

DLL หรือ Dynamic Link Library เป็นไฟล์ลิบราลี ประเภทหนึ่ง ซึ่งแตกต่างจากลิบราลีทั่วๆ ไป เนื่องจากลิบราลี โดยทั่วไปจะถูกลิงค์ เก็บไว้ในโปรแกรมระหว่างที่คอมไพล์ โปรแกรมนั้น เป็นเอ็กซีคิวท์โปรแกรม แต่ไฟล์ DLL เป็นลิบราลี ที่ถูกอ่านมาเก็บในหน่วยความจำในขณะที่ถูกเรียกใช้โดยโปรแกรมต่างๆ ซึ่งจะทำให้โปรแกรมแอพลิเคชันมีขนาดเล็กลง

2.6.1 การเรียกใช้งาน API

ในการเรียกใช้งานฟังก์ชัน และโพรซีเยอร์ ของ API มีขั้นตอนดังนี้

- ประกาศชื่อฟังก์ชัน และโพรซีเยอร์ ของ API ซึ่งมีรูปแบบดังนี้
  - การประกาศเรียกใช้งานโพรซีเยอร์ ของ API

[Public/Private] Declare Sub *name* **Lib** "*libname*" **[Alias** "*aliasname*"**]**[(**[***arglist***]**)]

  - การประกาศเรียกใช้งานฟังก์ชัน ของ API

[Public/Private] Declare Function *name* **Lib** "*libname*" **[Alias** "*aliasname*"**]**[(**[***arglist***]**)] **[As** *type***]**

โดยที่ *name* หมายถึง ชื่อโพรซีเยอร์ หรือฟังก์ชันที่ต้องการเรียกใช้

libname หมายถึง ชื่อไฟล์ลิบรารี

aliasname หมายถึง ชื่อที่กำหนดขึ้นแทนชื่อของโพรซีเยอร์ หรือฟังก์ชันที่ต้องการเรียกใช้ ใช้ในกรณีที่โพรซีเยอร์ หรือฟังก์ชันมีชื่อเหมือนกับรีเซิร์ฟเวิร์ด

arglist หมายถึง รายชื่ออากิวเมนท์

type หมายถึง ประเภทของข้อมูลที่ฟังก์ชันจะคืนค่ากลับไป

ในการกำหนดประเภทของข้อมูลให้กับอากิวเมนท์ ต่างๆ ที่ต้องส่งผ่านไปยังโพรซีเยอร์ หรือฟังก์ชันที่ต้องการเรียกใช้จะต้องคำนึงรายละเอียดที่กำหนดไว้ของแต่ละโพรซีเยอร์ หรือฟังก์ชัน นั้นๆ ด้วย รวมทั้งต้องกำหนดประเภทของอากิวเมนท์ ต่างๆ อย่างเหมาะสมด้วย เนื่องจากอากิวเมนท์ ต่างๆ ที่ใช้ใน API โดยทั่วไปนั้นจะถูกสร้างด้วยภาษา C, Assembly หรือ FORTRAN เป็นส่วนใหญ่ เมื่อนำมาใช้กับ Visual Basic จึงต้องมีการเปลี่ยนแปลงประเภทของอากิวเมนท์ ให้เหมาะสมด้วย หรืออาจกำหนดประเภทโดยใช้ "As Any" แทนได้ จะเห็นว่าการประกาศโพรซีเยอร์ หรือฟังก์ชันนั้น จะต้องทราบถึงรูปแบบในการประกาศ และอากิวเมนท์ ต่างๆ ที่โพรซีเยอร์ หรือฟังก์ชันนั้นต้องการ ดังนั้นเราสามารถใช้ยูทิลิตี ของ Visual Basic ที่มีชื่อว่า API Viewer เข้ามาช่วย ซึ่งจะกล่าวต่อไป

- เรียกใช้งานโพรซีเยอร์ หรือฟังก์ชันของ API ที่ประกาศไว้ได้ตามต้องการ

2.6.2 API Viewer

เป็นยูทิลิตี ของ Visual Basic ซึ่งช่วยในการประกาศการเรียกใช้งาน API โดยในการเรียกใช้ เราจะต้อง "แอด-อินส์" API Viewer เข้ามาไว้ใน Visual Basic ก่อน

2.7 การใช้งาน DirectX 8.0

2.7.1 หลักการเบื้องต้นของ DirectX

DirectX คือชุดคำสั่งประเภท API ระดับต่ำที่ใช้สำหรับสร้างเกมส์และทำงานกับแอปพลิเคชั่นทางมัลติมีเดียได้อย่างมีประสิทธิภาพ รวมไปถึง การสนับสนุนภาพกราฟฟิคในแบบ 2 มิติ และ 3มิติ, รวมไปถึงดนตรีและเสียงประกอบ , การรับข้อมูลจากอุปกรณ์ , และสนับสนุนโปรแกรมการทำงานด้านเครือข่าย เช่น เกมส์แบบผู้เล่นเล่นพร้อมกันหลายๆคน

การเรียกใช้ DirectX8 ในโปรแกรมทำได้โดย ประกาศ " dx As New DirectX8 " ในโปรแกรมก่อนที่จะเรียกใช้ ในส่วนอื่นๆ dx เป็นอ็อบเจ็ค ที่ใช้เรียกแทน DirectX8

เมทธอดของDirectX8 ที่ใช้ในงานนี้คือ DirectSoundCaptureCreate() ทำการสร้างอ็อบเจ็คของอุปกรณ์บันทึกเสียงขึ้น

2.7.2 การใช้งาน DirectSoundCapture8

เป็นอ๊อบเจคที่สร้างจากเมทธอด DirectX8 โดยที่ DirectSoundCapture8 ใช้เป็นตัวแทนของอุปกรณ์ในการบันทึกเสียง โดยเมทธอดของอ๊อบเจคนี้จะใช้เพื่อสร้างบัฟเฟอร์

การใช้งาน DirectSoundCapture ทำได้โดยประกาศตัวแปร ds As DirectSoundCapture8

สร้างด้วย คำสั่ง Set ds = dx.DirectSoundCreate(vbNullString) ส่งค่าพารามิเตอร์เป็น vbnullstring หมายถึงใช้อุปกรณ์บันทึกที่เป็น Default ของระบบ

เมทธอดในการสร้าง บัฟเฟอร์คือ CreateCaptureBuffer(DSCBUFFERDESC)

2.7.3 การใช้งาน DirectSoundCaptureBuffer8

เป็นอ๊อบเจ็ค ที่สร้างมาจากเมทธอดของ DirectSoundCapture8 โดยใช้เป็นตัวแทนของบัฟเฟอร์ที่ใช้ในการบันทึกเสียง

ทำการสร้างได้โดยรวบรวมคุณสมบัติของบัฟเฟอร์ (DSCBUFFERDESC) ที่จะสร้างอันได้แก่

ประกาศ dscd As DSCBUFFERDESC (รายละเอียดของบัฟเฟอร์บันทึก)

- ค่า fxFormat คือ รายละเอียดเกี่ยวกับรูปแบบของสัญญาณเสียงบนบัฟเฟอร์
- ค่า lBufferBytes คือ ขนาดของบัฟเฟอร์
- ค่า lFlag กำหนดเป็น DSCBCAPS_WAVEMAPPED จะมีการใช้ Wave Mapper ของ Win32 ถ้าไม่สนับสนุนรูปแบบของข้อมูลนั้นๆ

โดยค่า fxFormat มีรายละเอียดที่ต้องรวบรวมมาก่อนดังนี้

- ค่า nFormatTag กำหนดเป็น WAVE_FORMAT_PCM คือส่วนที่บ่งบอกว่าเป็น Wave รูปแบบใด การใช้งานใน DirectSound และ DirectSoundCapture นั้น ใช้ได้กับ nFormatTag ที่เป็น WAVE_FORMAT_PCM เท่านั้น
- ค่า nChannels คือ จำนวนช่องของ Wave ถ้าเป็นเสียง Stereo จะมีค่าเป็น 2 และ 1 สำหรับ Mono
- ค่า lSamplesPerSec คือ ค่าความถี่สุ่ม
- ค่า nBitsPerSample คือ จำนวน Bit ที่ใช้ในการเก็บค่าสุ่ม 1 ค่า
- ค่า nBlockAlign คำนวณจาก Channels * BITS / 8 คือ ค่าขนาดบล็อกหนึ่งๆของการบันทึก
- ค่า lAvgBytesPerSec คำนวณจาก lSamplesPerSec * nBlockAlign คือ จำนวนข้อมูล (ไบต์)เฉลี่ยใน 1 วินาที

ทำการประกาศ dscb As DirectSoundCaptureBuffer8

จากนั้นสร้างด้วยคำสั่ง dscb = dsc.CreateCaptureBuffer(dscd)

เมทธอดที่ใช้ในงานนี้ได้แก่

- Start เริ่มการบันทึกข้อมูลลงบัฟเฟอร์ เมื่อมีการเรียก Start ข้อมูลเดิมจะถูกเริ่มใหม่ที่ต้นบัฟเฟอร์ ไม่ได้บันทึกต่อจากจุดที่ถูก Stop และจะทำการบันทึกจนถูกเรียก Stop
- Stop หยุดการบันทึกบัฟเฟอร์
- Readbuffer ใช้ในการอ่านข้อมูลจากบัฟเฟอร์มาเก็บไว้ภายในบัฟเฟอร์ของโปรแกรม (มักจะเป็นอาร์เรย์)

2.8 การแปลงฟูเรียร์แบบเต็มหน่วย (Discrete Fourier Transform : DFT)

ถ้าเรามีข้อมูลเสียงอยู่ชุดหนึ่งซึ่งมีโดเมนเป็นค่าของเวลา (Time Domain) และเราต้องการที่จะวิเคราะห์ข้อมูลนี้ เราสามารถที่จะวิเคราะห์ได้เลยเพียงแต่จะมีความยุ่งยากมาก เราจึงใช้ทฤษฎีการแปลงฟูเรียร์แบบเต็มหน่วย (Discrete Fourier Transform : DFT) เพื่อแปลงชุดของข้อมูลนั้นให้อยู่ในรูปของโดเมนที่เป็นค่าของความถี่แทน (Frequency Domain) ซึ่งจะทำให้การวิเคราะห์เป็นไปโดยง่ายและสะดวกยิ่งขึ้น ต่อมาก็ได้มีการพัฒนาขั้นตอนวิธี เพื่อให้ใช้เวลาในการคำนวณน้อยลง ซึ่งเรียกกันว่าวิธีการแปลงฟูเรียร์อย่างเร็ว (Fast Fourier Transform : FFT)

2.8.1 นิยามของการแปลงฟูเรียร์แบบเต็มหน่วย (Discrete Fourier Transform : DFT)

การแปลงฟูเรียร์แบบเต็มหน่วยมีนิยามดังนี้คือ

$$A_r = \sum_{K=0}^{N-1} X_k \exp(-2\pi jrk/N) \qquad (1)$$

โดย $r = 0, 1, 2, \ldots, N-1$

และเมื่อ A_r คือสัมประสิทธิ์เทอมที่ r ของ DFT

X_k คือตัวอย่าง(Sample) ตัวที่ k ของอนุกรมเวลาที่มี N เทอม

$j = \sqrt{-1}$

สำหรับค่า X นั้นอาจจะเป็นจำนวนเชิงซ้อนก็ได้แต่ A มักจะเป็นจำนวนเชิงซ้อนทั้งหมด เพื่อความสะดวก เราอาจเขียนสมการใหม่ได้เป็น

$$A_r = \sum_{K=0}^{N-1} X_k W^{rk} \qquad (2)$$

โดย $r = 0, 1, 2, \ldots, N-1$

และเมื่อ $W^{rk} = \exp(-2\pi jk/N)$

เพราะว่าโดยปกติแล้ว X_k จะเป็นค่าของฟังก์ชัน ณ discrete time points ดังนั้นในบางครั้งดัชนี r จึงมักเรียกว่าความถี่ (frequency) ของ DFT

นอกจากนี้เรายังสามารถขยายนิยามของ A_r ไปใช้กับจำนวนเต็มทุกจำนวนไม่ว่าจะเป็นบวกหรือลบ ได้ซึ่งนำไปใช้ประโยชน์ได้มากมายต่อไป จากนิยามนี้จะได้ว่า

$$A_r = A_{N+r} = A_{2N+r} = \dots \quad (3)$$

และ

$$X_1 = A_{N+1} = A_{2N+1} = \dots \quad (4)$$

2.8.2 การแปลงฟูเรียร์อย่างเร็ว (Fast Fourier Transform : FFT)

การแปลงฟูเรียร์อย่างเร็ว (Fast Fourier Transform : FFT) สามารลดปริมาณการคูณและการบวกในการใช้วิธีการแปลงฟูเรียร์แบบเต็มหน่วย (DFT) ซึ่งต้องคำนวณจาก N^2 ครั้ง ให้ลดลงเหลือ $N \log_2 N$ ครั้ง เมื่อ N คือข้อมูลที่สุ่มตัวอย่าง

กำหนดให้ f(x) เป็นชุดของค่าข้อมูลจำนวนจำกัด N ค่า และ FFT ของ f(x) คือ F(s) จะได้สมการเป็นดังนี้

$$F(k) = \sum_{K=0}^{N-1} f(n)\, W_N^{kn} \quad (5)$$

เมื่อ $k = 0, 1, \dots, N-1$

ซึ่งค่า W_N^{kn} เป็นค่าสำคัญที่ต้องคำนวณทุกครั้งของการคำนวณ และเป็นตัวการสำคัญที่ทำให้สิ้นเปลืองเวลา เราจะเลี่ยงการคำนวณค่า W_N^{kn} ในทุกรอบ โดยการคำนวณค่าของ W_N^{kn} สำหรับทุกค่าของ k และ n แล้วนำไปเก็บในหน่วยความจำของเครื่องคอมพิวเตอร์ก่อน เมื่อจะใช้ก็ค่อยเรียกออกมาใช้ได้เลย แต่จะมีปัญหาคือต้องใช้หน่วยความจำปริมาณมากถ้า N มีค่ามากๆ

และลักษณะสำคัญของ FFT คือ N จะต้องไม่เป็นจำนวนเฉพาะ (prime number) และจะดีที่สุดถ้า N มีตัวประกอบเป็น 2 ทั้งหมด (เขียนในรูปของ 2^x ได้) มีขั้นตอนในการคำนวณดังต่อไปนี้

สมมติว่า f(x) เป็นจุดค่าสุ่มตัวอย่างของ f(x) ค่าสุ่มตัวอย่างนี้เป็นค่าทางเวลาซึ่งเราเรียกว่า Decimation in time ถ้าเป็นค่าสุ่มตัวอย่างทางความถี่จะเรียกว่า Decimation in frequency ที่มีอยู่ N ค่า โดยที่ N ต้องมีตัวประกอบเป็น 2 ทั้งหมด

จาก f(x) จำนวน N ค่านี้ เราสามารถแยกออกเป็น 2 กลุ่มๆละ N/2 ค่าเท่าๆกันได้ โดยสมมุติเป็น g(x) และ h(x) โดยที่ g(x) เป็นค่าที่เลือกออกมาจาก f(x) ในตำแหน่งเลขคู่

h(x) เป็นค่าที่เลือกออกมาจาก f(x) ในตำแหน่งเลขคี่

กล่าวคือ $g(x) = \{ f(0), f(2), f(4), \dots \} = \{ g(0), g(1), g(2), \dots \}$

$h(x) = \{ f(1), f(3), f(5), \dots \} = \{ h(0), h(1), h(2), \dots \}$

หรือ

$$g(x) = f(2r) \quad ; r = 0, 1, 2, \dots, N/2 - 1 \qquad (6)$$

$$h(x) = f(2r+1) \; ; r = 0, 1, 2, \dots, N/2 - 1 \qquad (7)$$

เราสามารถคำนวณ DFT ของ g(x) และ h(x) ได้จากสมการ (5)

$$F(k) = \sum_{K=0}^{N-1} f(n) W_N^{kn} \qquad \text{เมื่อ } k = 0, 1, \dots, N-1$$

$$= \sum_{N \text{ เป็นเลขคู่}} f(n) W_N^{kn} + \sum_{N \text{ เป็นเลขคี่}} f(n) W_N^{kn}$$

$$= \sum_{K=0}^{N/2-1} f(2r) W_N^{2rk} + \sum_{K=0}^{N/2-1} f(2r+1) W_N^{(2r+1)k}$$

จาก $W_N^2 = W_{N/2}$

$$F(k) = \sum_r f(2r) W_{N/2}^{rk} + W_N^k \sum_r f(2r+1) W_{N/2}^{rk}$$

จะได้ว่า

$$F(k) = \sum g(r) W + W \sum h(r) W \qquad (8)$$

พิจารณาโดยละเอียดตามคุณสมบัติฟังก์ชันคาบของ $W_{N/2}$ จะได้

$$W_{N/2}^{rk} = W_{N/2}^{r(k+N/2)}$$

และ $W_N^k = - W_N^{(k+N/2)}$

เมื่อ k เปลี่ยนค่าตั้งแต่ 0 ถึง N-1 จะทำให้ $\sum_r f(2r)\, W_{N/2}^{rk}$ และ $\sum_r f(2r+1)\, W_{N/2}^{rk}$ มีลักษณะการกระจายเหมือนกัน กล่าวคือสามารถใช้ค่า $W_{N/2}^{rk}$ ร่วมกันได้ แทนที่เราจะต้องทำการคำนวณทุกครั้งเหมือนในสมการที่ (5)

ถ้ากำหนดให้ G(k) และ H(k) เป็นค่าสุ่มตัวอย่างที่ k ของ g(s) และ h(s) ตามลำดับแล้ว สมการที่ (8) สามารถเขียนให้อยู่ในรูป

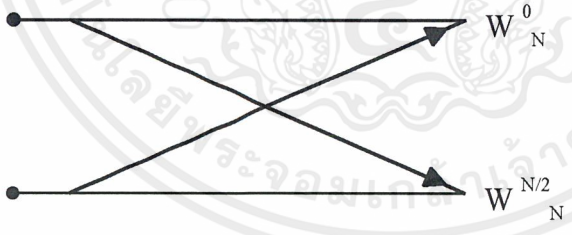
$$F(k) = G(k) + W_N^{k} H(k) \qquad (9)$$

และ
$$\begin{aligned} F(k+N/2) &= G(k+N/2) + W_N^{k+N/2} H(k+N/2) \\ &= G(k) - W_N^{k} H(k) \end{aligned} \qquad (10)$$

เพราะว่า
$$\begin{aligned} G(k+N/2) &= \sum_r g(r)\, W_{N/2}^{r(k+N/2)} \\ &= \sum_r g(r)\, W_{N/2}^{rk} \\ &= G(k) \end{aligned}$$

ในทำนองเดียวกัน $H(k+N/2) = H(k)$ และ $W_N^{k+N/2} = - W_N^{k}$

สมการ (9) และ (10) ถ้าให้ค่า k มีค่าเป็น 0 , 1 , 2 , … , N/2 –1 ก็จะสามารถให้ค่า k ชุดเดียวกันกับสมการ (5) ในขณะที่สมการ(1) ต้องใช้ค่า k = 0 , 1 , 2 , … , N–1



รูปที่ 2.3 กราฟแสดงการไหลของสมการ (9) และ (10)

ค่า W เป็นตัวประกอบที่ต้องนำไปคูณกับสัญญาณที่มาจาก H(k) เสมอ เช่น

$$F(0) = G(0) + W_N^{0} H(0) = G(0) + H(0)$$

$$F(4) = G(0) + W_N^{0} H(0) = G(0) - H(0)$$

$$F(1) = G(1) + W_N^{1} H(1)$$

$$F(5) = G(1) + W_N^{1} H(1)$$

จากสมการที่(5) $F(k) = \sum f(n) W_N^{kn}$, $k = 0, 1, \ldots, N-1$

จะเห็นว่าค่า F(k) หนึ่งค่าจะมีการคูณเกิดขึ้น N ครั้ง และการบวก N-1 ครั้ง ระหว่าง f(n) และ W_N^{kn} เพราะฉะนั้นสำหรับ F(k) ทั้งหมด N ค่า จะมีจำนวนการคูณเกิดขึ้น N^2 ครั้งและจำนวนการบวก N^2-N ครั้ง ระหว่าง f(n) และ W_N^{kn} นั่นคือ DFT ของ N จุดจะมีจำนวนการคูณเกิดขึ้น N^2 ครั้งและจำนวนการบวก N^2-N ครั้ง

แต่ถ้าจากรูปที่ 2.3 นั้นเป็นเทคนิคที่เกิดจากการแบ่งกลุ่มออกเป็น 2 กลุ่มดังกล่าว ซึ่งจะมีขั้นตอนในการคำนวณดังนี้ DFT ของ N/2 จุด 1 ชุด จะมีการคูณเกิดขึ้น $(N/2)^2$ ครั้ง และการบวก $(N/2)^2 - (N/2)$ ครั้ง

DFT ของ N/2 จุด 2 ชุด จะมีการคูณเกิดขึ้น $2(N/2)^2$ ครั้ง และการบวก $2[(N/2)^2 - (N/2)]$ ครั้ง มีการคูณกันระหว่าง W_N^k และ H(k) อีก N ครั้ง การบวกและลบกันของ $G(k) \pm W_N^{kn} H(k)$ อีก N ครั้ง

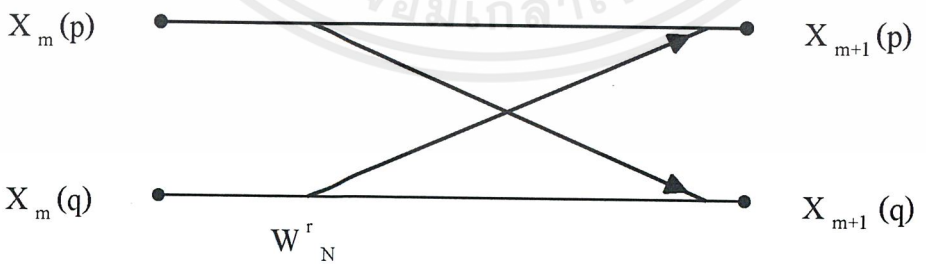
นั่นคือมีการคูณ $2(N/2)^2 + N$ ครั้ง และการบวก $2[(N/2)^2 - (N/2)] + N^2 = 2(N/2)$ ครั้ง

นั่นคือเราสามารถลดการคำนวณลงไปได้มากเมื่อ $N > 2$

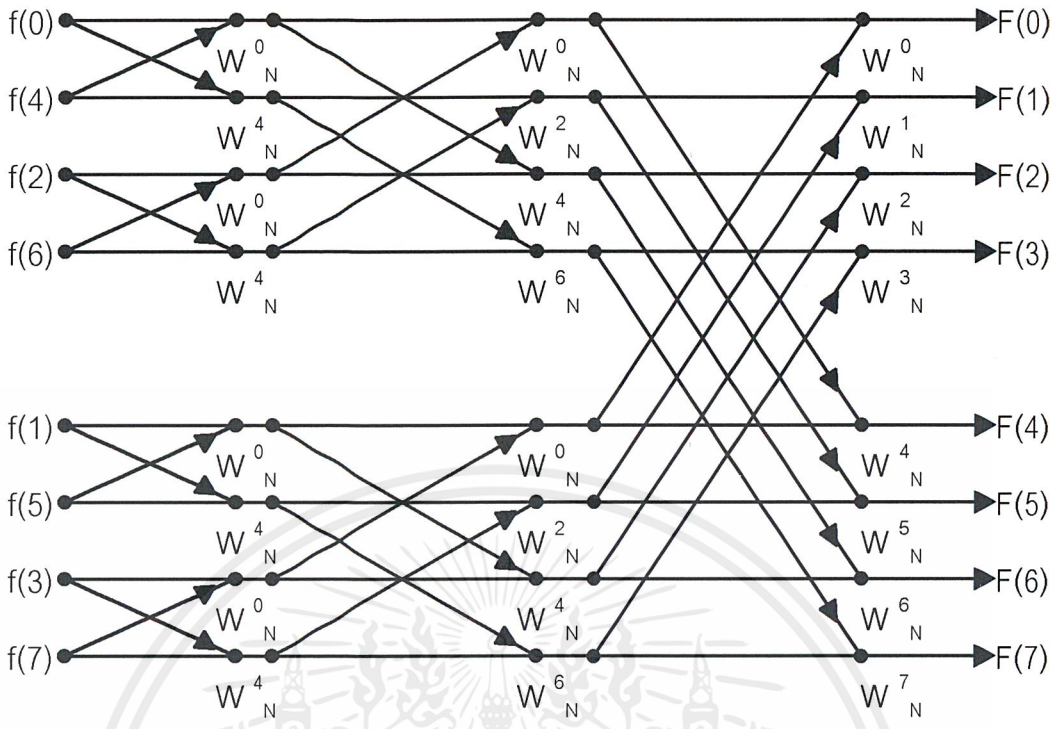
และในทำนองเดียวกันนี้ ถ้า N/2 จุดนี้ยังหาร 2 ได้ลงตัวอีกเราก็สามารถดำเนินขั้นตอนเดียวได้เช่นเดียวกันคือ แยกออกเป็น 2 กลุ่มย่อยๆลงไปได้อีก จึงทำให้ประสิทธิภาพของการแปลงฟูเรียร์เพิ่มขึ้นไปอีก ตราบเท่าที่ N ยังคงหารด้วย 2^a ได้ลงตัว(a เป็นเลขจำนวนเต็ม) นั่นก็คือ เราทำการแบ่งกลุ่มย่อยลงมาทั้งหมด a ขั้นซึ่งต้องใช้จำนวนครั้งของการคูณเป็น $2^a(N/2^a)^2 + aN = N^2/2^a + aN$

นั่นคือ ในกรณีที่ N เป็นค่าของ 2 ยกกำลัง b , $N = 2^b$

เราก็จะสามารถทอนการคำนวณ DFT ของข้อมูล N จุดลงไปเหลือเพียง DFT ของ 2 จุดทั้งหมด 2^{b-1} ชุด และมีการคูณ(ของจำนวนเชิงซ้อน) $2^b(N/2^b)^2 + (b-1)2^b = b2^b$ หรือ $N \log_2 N$ ครั้ง



รูปที่ 2.4 แสดงการไหลของสัญญาณในกลุ่มย่อยสุด



รูปที่ 2.5 แสดงการไหลของสัญญาณสำหรับ FFT ของ 8 จุด

จากรูปที่ 2.5 จัดว่าเป็นวิธีการคำนวณที่มีประสิทธิภาพมากที่สุดวิธีหนึ่ง(เพราะใช้เวลาในการคำนวณเพียง 2% ของการคำนวณแบบตรงๆ) ดังที่ได้กล่าวมาปัญหาในการคำนวณของ DFT ก็คือเวลาที่ต้องเสียไปกับการรับส่งข้อมูล และการสำรองหน่วยความจำไว้ในขณะทำการคำนวณ แต่ปัญหาต่างๆเหล่านี้ก็สามารถทำให้หมดไปได้ ถ้าเราสังเกตจากรูปที่ จะเห็นว่าผลการคำนวณในแต่ละขั้นตอนนั้นสามารถแทนที่ข้อมูลเดิมได้ทันที เช่น ข้อมูล f(0) และ f(4) จะให้ผลการคำนวณ A1 และ B1 ซึ่ง f(0) และ f(4) จะไม่เกี่ยวข้องกับข้อมูลอื่นๆอีกเลย ดังนั้นเราจึงสามารถนำค่า A1 แทนลงไปที่หน่วยเก็บความจำ f(0) และแทน B1 ลงไปที่หน่วยเก็บความ จำ f(4) ได้โดยตรง สำหรับในขั้นที่ 2 นั้นก็เช่นเดียวกันข้อมูลใหม่ A1 และ C1 จะให้ผล A2 และ C2 แล้วนำ A2 ไปแทนที่ในหน่วยความจำ A1 และ C2 แทนที่ในหน่วยความจำ C1 ตามลำดับ และจะเป็นไปในทำนองนี้ตลอดการคำนวณ ซึ่งอาจจะสรุปได้ว่า

ในการคำนวณครั้งที่ m ข้อมูลตำแหน่งที่ p และ q คือ $x_m(p)$ และ $x_m(q)$ เมื่อนำมาคำนวณจะได้ผลเป็นขั้นที่ m+1 ด้วยข้อมูลตำแหน่งที่ $x_{m+1}(p)$ และ $x_{m+1}(q)$ ตามลำดับโดยมีสมการเป็น

$$x_{m+1}(p) = x_m(p) + W_N^{พ} x_m(q)$$

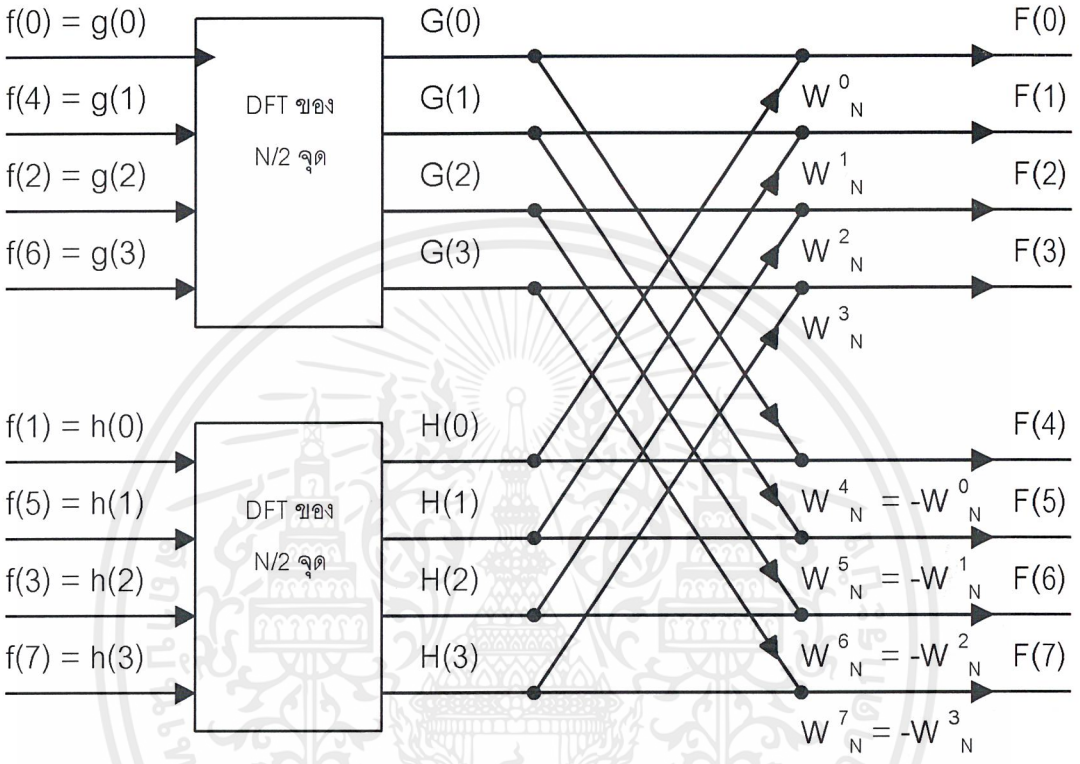
$$x_{m+1}(q) = x_m(q) + W_N^{พ+N/2} x_m(p)$$

แต่เนื่องจาก $W_N^{N/2} = -1$ ดังนั้นเขียนสมการใหม่ได้ดังนี้

$$x_{m+1}(p) = x_m(p) + W_N^{พ} x_m(q)$$

$$x_{m+1}(q) = x_m(q) - W_N^{พ} x_m(p)$$

และการไหลของสัญญาณจะได้ดังรูปที่ 2.6



รูปที่ 2.6 แสดงการไหลของสัญญาณ

และจากรูปที่ 2.6 นั้นมีข้อสังเกตที่สำคัญของข้อมูล f(0) , f(1) , f(2) , ... ทางซ้ายมือและค่า F(0) , F(1) , F(2) , ... ทางขวามือไม่ได้อยู่ตรงกันเลยกล่าวคือจะอยู่ในรูปของ bit-reversed order ของเลขฐานสอง

ตัวอย่างเช่นข้อมูล 8 จุด

f(0) = f(000) --------- F(0) = F(000)

f(4) = f(100) --------- F(1) = F(001)

f(2) = f(010) --------- F(2) = F(010)

f(6) = f(110) --------- F(3) = F(011)

f(1) = f(001) --------- F(4) = F(100)

f(5) = f(101) --------- F(5) = F(101)

f(3) = f(001) --------- F(6) = F(100)

f(7) = f(111) --------- F(7) = F(111)

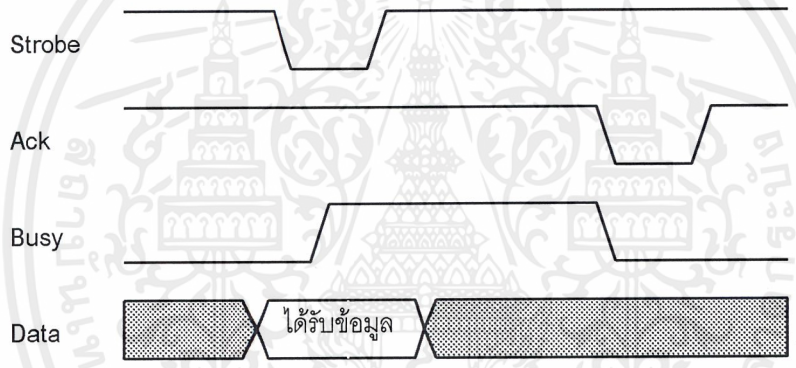
2.9 การเชื่อมต่อกับบอร์ด LED มอนิเตอร์ 8 ช่องโดยผ่านทางพอร์ตขนาน

2.9.1 การใช้งานบอร์ดเชื่อมต่อพอร์ตขนาน (P-Board)

1) ความรู้เบื้องต้นเกี่ยวกับพอร์ตขนาน

พอร์ตขนาน (Parallel Port) หรือก็คือพอร์ตที่มีการถ่ายเทข้อมูลในแบบขนาน ทำให้มีการถ่ายเทข้อมูลสูงกว่าแบบอนุกรม 8 –10 เท่า และการประมวลผลข้อมูลส่วนใหญ่จะมีขนาด 8 บิต

2) ลักษณะทางกายภาพของพอร์ตขนาน



รูปที่ 2.7 แสดงไดอะแกรมเวลาของการส่งข้อมูลไปยังอุปกรณ์ที่ต่อกับพอร์ตขนาน

ดังรูปที่ 2.7 จะเห็นได้ว่ามีสัญญาณที่ใช้งานจริงๆ มีไม่มากเริ่มจากสัญญาณพอร์ตเดต้า ถูกส่งออกไปยังอุปกรณ์ที่ต่อกับพอร์ตขนาน พร้อมทั้งส่งสัญญาณสโทรป ออกไปด้วย เพื่อให้อุปกรณ์ดังกล่าวรับรู้ว่ามีการส่งข้อมูลมาที่ขาเดต้า แล้ว จากนั้นคอมพิวเตอร์จะต้องรอการตอบกลับจากอุปกรณ์ดังกล่าว นั่นก็คือ อุปกรณ์ที่ต่อกับพอร์ตขนานจะสร้างสัญญาณบีซี่ ขึ้นมาเพื่อบอกว่ายังไม่พร้อมจะรับข้อมูลใหม่ แล้วเมื่อพร้อมจะรับข้อมูลใหม่ก็จะสร้างสัญญาณ Ack (Acknowledge) ส่งออกไปยังคอมพิวเตอร์ เพื่อแจ้งความพร้อมที่จะรับข้อมูล

นอกจากสัญญาณข้อมูล (ขนาด 8 บิต), สัญญาณสโทรป , สัญญาณ Ack (Acknowledge) ซึ่งเป็นสัญญาณที่สำคัญในการส่งข้อมูลจากคอมพิวเตอร์ไปยังอุปกรณ์ที่ต่อกับพอร์ตขนาน แล้วส่วนใหญ่ในการติดต่อจะมีสัญญาณอื่นร่วมด้วย เช่น ในการติดต่อกับเครื่องพิมพ์ที่มีทั้งการรับข้อมูลจากคอมพิวเตอร์, พิมพ์ข้อมูลที่รับเข้ามา, และตอบสนอง

ต่อการใช้ต่างๆ ของผู้ใช้ เช่น การเปลี่ยนฟอนต์ เป็นต้น และบางครั้งอาจเกิดเหตุการณ์ไม่ปกติ เช่น บัฟเฟอร์สำหรับรับข้อมูลเต็ม เครื่องพิมพ์จะต้องแจ้งไปยังคอมพิวเตอร์ให้หยุดส่งข้อมูลชั่วคราว เนื่องจากไม่สามารถรับข้อมูลได้มากกว่านี้อีกแล้ว ก็จะส่งสัญญาณบีซี่ไปยังคอมพิวเตอร์ หรือเมื่อเครื่องพิมพ์เกิดข้อผิดพลาดขึ้น เช่น กระดาษติด เครื่องพิมพ์ก็จะแจ้งไปยังคอมพิวเตอร์เป็นสัญญาณ Error และ ยังมีสัญญาณที่คอมพิวเตอร์ส่งไปยังเครื่องพิมพ์ เมื่อคอมพิวเตอร์ต้องการรีเซ็ตเครื่องพิมพ์ คือสัญญาณรีเซ็ต

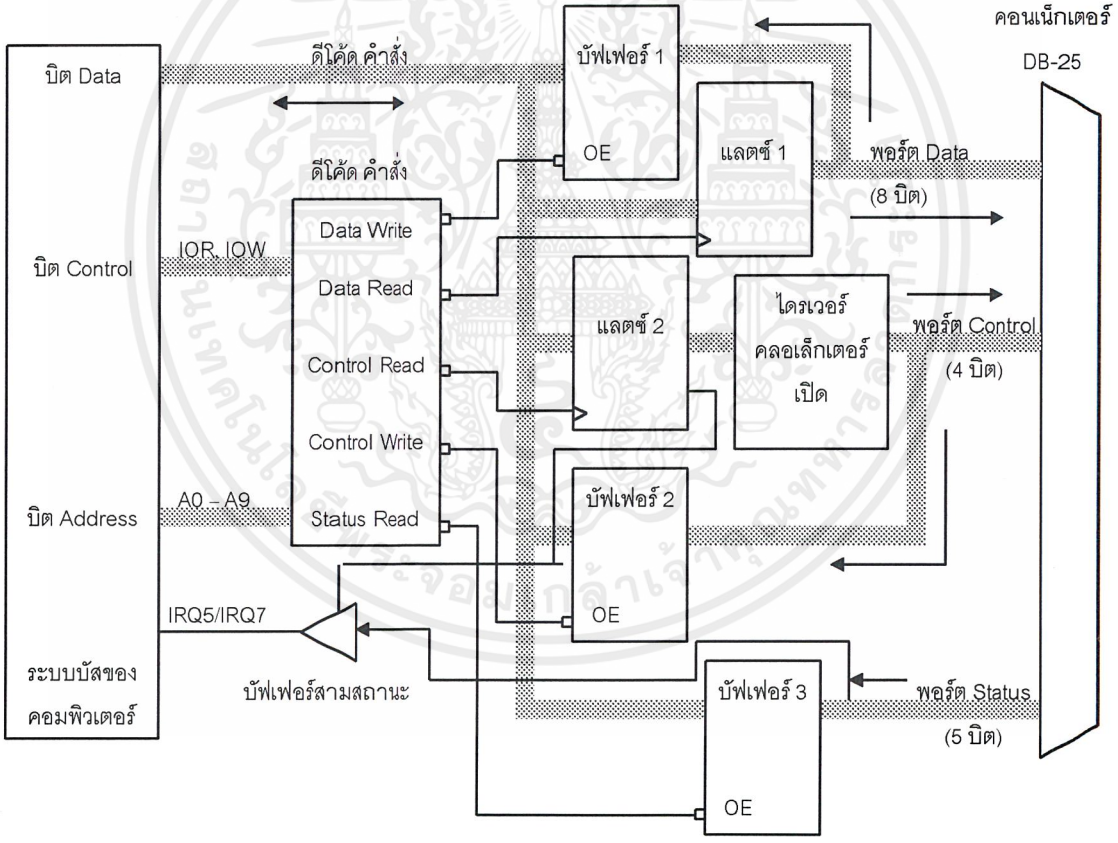
ตารางที่ 2.13 แสดงหน้าที่และการทำงานของตำแหน่งขาต่างๆ ของพอร์ตขนาน

DB-25	รีจิสเตอร์	ทิศทาง	ตำแหน่งบิต	ชื่อขาสัญญาณ	หน้าที่การทำงาน
1	Control	Out	-C0	NSTROBE	แอกทีฟ “0” ส่งค่าออกไปเพื่อบอกว่าที่ขา Data มีข้อมูลแล้ว
2 – 9	Data	Out	D1 - D8	DATA1 – DATA8	ส่งข้อมูล
10	Status	In	S6	NACK	เป็นพัลส์ลอจิก “0” ที่ส่งมาจากอุปกรณ์ภายนอกเมื่อได้รับข้อมูล
11	Status	In	-S7	BUSY	อุปกรณ์ภายนอกไม่พร้อม
12	Status	In	S5	PE	เครื่องพิมพ์แจ้งกระดาษหมด
13	Status	In	S4	SELECT	แจ้งว่าเครื่องพิมพ์ต่ออยู่
14	Control	Out	-C1	AUTO FEED	สั่งเครื่องพิมพ์ให้เลื่อนบรรทัด
15	Status	In	S3	NERROR	อุปกรณ์ภายนอกมีปัญหา เช่น มีข้อผิดพลาดจากการพิมพ์
16	Control	Out	C2	NINIT	รีเซตเครื่องพิมพ์โดยลอจิก “0”
17	Control	Out	-C3	NSELECT – IN	ส่งสัญญาณไปยังเครื่องพิมพ์ว่า ต้องการเลือกเครื่องพิมพ์เครื่องนี้
18 – 25	-	-	-	GND	สายกราวด์

โดยปกติพอร์ตขนานออกแบบมาให้มีสายสัญญาณอยู่ทั้งหมด 17 เส้น โดยที่สายสัญญาณเหล่านั้น จะมี รีจิสเตอร์ ควบคุมการทำงานอยู่ 3 ตัวคือ

1. พอร์ตเอาต์พุต สำหรับสัญญาณข้อมูล 8 เส้น มีรีจิสเตอร์เดต้า (Data) ควบคุม
2. พอร์ตอินพุต สำหรับการอ่านค่าสถานะต่างๆ จากภายนอกมีอยู่ด้วยกัน 5 เส้น ใช้รีจิสเตอร์สเตตัส ในการควบคุม
3. พอร์ตเอาต์พุต สำหรับส่งสัญญาณควบคุมไปยังอุปกรณ์ภายนอก มีอยู่ด้วยกัน 4 เส้น ใช้รีจิสเตอร์คอนโทรล ในการควบคุม

ระบบบัสของคอมพิวเตอร์สำหรับติดต่อกับพอร์ตขนาน สัญญาณเอาต์พุตจากพอร์ตขนานจะถูกส่งไปยังคอนเน็กเตอร์ แบบ DB-25 สำหรับคอมพิวเตอร์ส่วนใหญ่ในปัจจุบันพอร์ตขนานจะมีมาพร้อมกับเมนบอร์ด และยังคงสนับสนุนการทำงานของพอร์ตขนานในรูปแบบมาตรฐาน (Standard Parallel Port : SPP) อยู่ด้วย



รูปที่ 2.8 แสดงระบบบัสภายในของพอร์ตขนาน

จากรูปที่ 2.8 ซึ่งเทียบการทำงานโดยทั่วไปของการเชื่อมต่อผ่านการ์ดที่เสียบลงในสล็อตของคอมพิวเตอร์แล้ว พอร์ตขนานจะมีลักษณะใกล้เคียงกัน โดยการติดต่อกับพอร์ตขนานจะต้องมีการอ้างแอดเดรส ตำแหน่งแอดเดรสที่ใช้อ้างถึงตำแหน่ง A0 – A9 และใช้ขา IOR และ IOW สำหรับเป็นตัวเลือกว่าต้องการอ่านหรือเขียนรีจิสเตอร์ตัวใด จากการดีโค้ดแอดเดรส A0 – A9 นี้เองทำให้ได้สัญญาณ ออกมาเพื่อไปควบคุม หรืออีนาเบิลวงจรบัฟเฟอร์ต่างๆ ดังนี้

Data Write สัญญาณอีนาเบิลสำหรับนำข้อมูลที่อยู่ในบัสข้อมูลไปออกที่ขาเดต้าของพอร์ตขนาน

Data Read สัญญาณอีนาเบิลสำหรับอ่านข้อมูลจากขาเดต้า ของพอร์ตขนานมาเก็บไว้ในบัสข้อมูล

Control Write สัญญาณอีนาเบิลสำหรับนำข้อมูลที่อยู่ในบัสข้อมูลไปออกที่ขาคอนโทรล ของพอร์ตขนาน สำหรับพอร์ตนี้นอกจากจะส่งข้อมูลออกไปยังพอร์ตขนานแล้ว ยังทำหน้าที่อีนาเบิล การอินเตอร์รัป ของการเปลี่ยนแปลงสัญญาณที่พอร์ตสเตตัส อีกด้วย

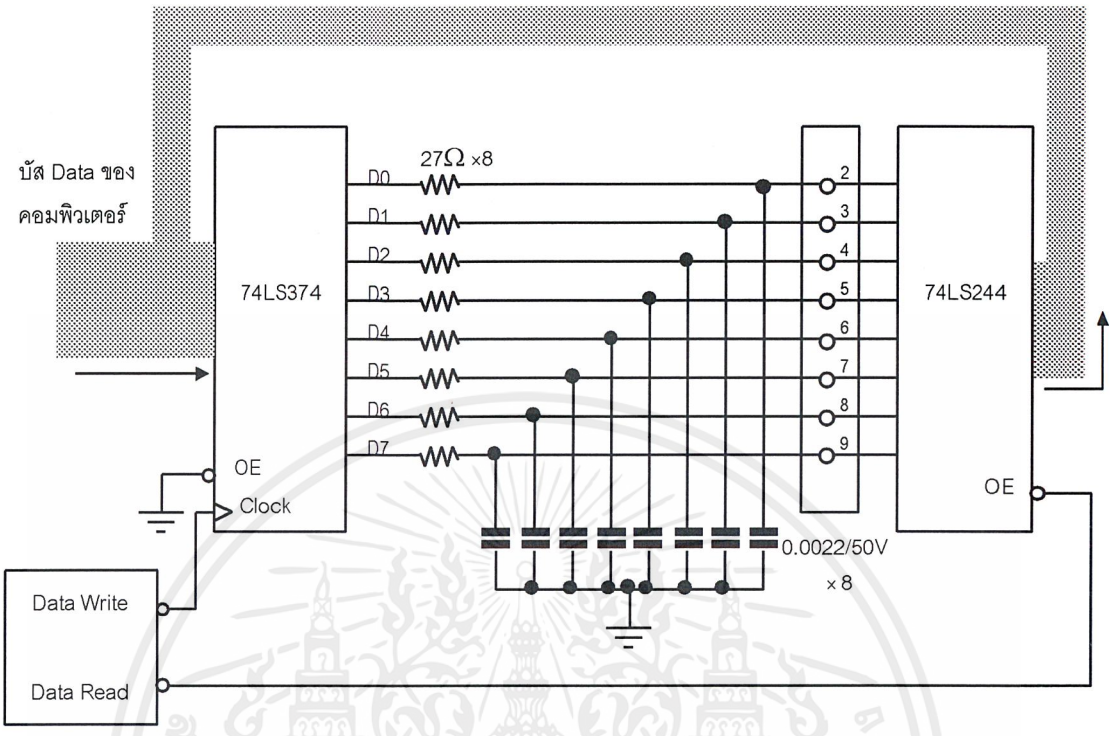
Control Read สัญญาณอีนาเบิลสำหรับอ่านข้อมูลจากขาคอนโทรล มาเก็บไว้ในบัสข้อมูล

Status Read สัญญาณอีนาเบิลสำหรับอ่านข้อมูลจากขาพอร์ตสเตตัส มาเก็บไว้ในบัสข้อมูล

ตารางที่ 2.14 แสดงแอดเดรสของพอร์ตขนาน

ชื่อพอร์ต	ตำแหน่ง LPT1 :		ตำแหน่ง LPT2 :		ตำแหน่ง LPT3 :	
	ฐานสิบ	ฐานสิบหก	ฐานสิบ	ฐานสิบหก	ฐานสิบ	ฐานสิบหก
DATA	888	378H	956	3BCH	632	278H
STATUS	889	379H	957	3BDH	633	279H
CONTROL	890	37AH	958	3BEH	634	27AH

3) พอร์ตเดต้า



รูปที่ 2.9 แสดงวงจรภายในของพอร์ตเดต้า

จากรูปที่ 2.9 แสดงให้เห็นว่า พอร์ตเดต้า ประกอบไปด้วยบัฟเฟอร์ 1 ตัว และ ไอซีแลตช์ (74LS374)อีก 1 ตัว เมื่อคอมพิวเตอร์ต้องการส่งข้อมูลไปยังเครื่องพิมพ์ คอมพิวเตอร์จะเขียนข้อมูลไปยังไอซีแลตช์1 ทั้ง 8 บิต เอาต์พุตของไอซีแลตช์1 คือ D0 – D7 ซึ่งเอาต์พุตนี้จะไปปรากฏอยู่ที่พอร์ตขนานในตำแหน่งขา 2 ถึงขา 9 และที่ขาเอาต์พุตนี้สัญญาณเดต้าจะส่งกลับไปเป็นอินพุตของบัฟเฟอร์1 ด้วยทำให้คอมพิวเตอร์สามารถอ่านค่าสถานะปัจจุบันที่เกิดขึ้นกับพอร์ตเดต้าได้

เมื่อคอมพิวเตอร์ส่งข้อมูล ข้อมูลจะถูกส่งมาจากบัสข้อมูลของคอมพิวเตอร์ผ่านให้กับไอซี 74LS374 ซึ่งเป็นไอซีแลตช์ข้อมูล และเมื่อต้องการให้ข้อมูลปรากฏที่เอาต์พุต คอมพิวเตอร์จะส่งสัญญาณ Data Write ออกไปที่ขา Clock ของ 74LS374 เอาต์พุตจาก 74LS374 จะถูกกรองด้วยวงจร RC ซึ่งประกอบด้วยตัวต้านทานค่า 27 Ω และตัวเก็บประจุ 0.0022 μF เพื่อให้ช่วงเวลาที่เปลี่ยนจากลอจิก “0” เป็นลอจิก “1” หรือจากลอจิก “1” เป็นลอจิก “0” เป็นไปอย่างช้าๆ เนื่องจากการเปลี่ยนแรงดันอย่างรวดเร็วทำให้เกิดสัญญาณรบกวนเหนี่ยวนำข้ามไปยังข้อมูลบิตอื่นๆ ได้ ทำให้ข้อมูลที่ส่งออกไปมีข้อผิดพลาด จากค่าตัวต้านทานและตัวเก็บประจุในวงจรทำให้เกิดการหน่วงเวลาไปประมาณ 60 นาโนวินาที

และจากวงจรดังรูปที่ 2.10 ทำให้เอาต์พุตของพอร์ตเดต้ามีคุณสมบัติ ดังนี้

- กระแสซิงค์สูงสุด 24 mA
- กระแสซอร์สสูงสุด 2.6 mA
- ระดับแรงดันของลอจิก "1" ต่ำสุดเท่ากับ 2.4 V
- ระดับแรงดันสูงสุดสำหรับลอจิก "0" เท่ากับ 0.5 V

สำหรับบัฟเฟอร์ที่ใช้ในการอ่านข้อมูลกลับ ได้แก่เบอร์ 74LS244 ซึ่งเมื่อต้องการอ่านค่าคอมพิวเตอร์จะส่งสัญญาณ Data Read ออกมาเพื่ออีนาเบิลไอซี 74LS244 สำหรับพอร์ตขนานแบบมาตรฐาน (SPP) พอร์ต Data จะต้องใช้เพื่อการส่งค่าออกเอาต์พุตเท่านั้น แต่สำหรับพอร์ตขนานที่มีการสื่อสารสองทิศทาง (Bidirectional Parallel Port) สามารถอ่านค่าจากพอร์ต Data ได้ด้วย แต่ก่อนที่จะอ่านค่าต้องป้อนข้อมูลเอาต์พุตให้มีค่าลอจิก "1" ทั้งหมดก่อน และเนื่องจากระบบสนใจการส่งข้อมูลทางพอร์ตเดต้า (LPT1) จึงขอกล่าวถึงเฉพาะในส่วนนี้เท่านั้น

4) การเชื่อมต่อกับบอร์ดเชื่อมต่อพอร์ตขนาน (P-Board)

เริ่มจากการต่อบอร์ดเข้ากับพอร์ตขนานของคอมพิวเตอร์ทางคอนเน็กเตอร์แบบ DB-25 ตัวเมีย หลังจากนั้นบอร์ดจะสามารถรับข้อมูลที่ถูกส่งจากคอมพิวเตอร์ด้วยคำสั่ง Out ได้

2.9.2 การใช้งานบอร์ด LED มอนิเตอร์ 16 ช่อง (EX-01)

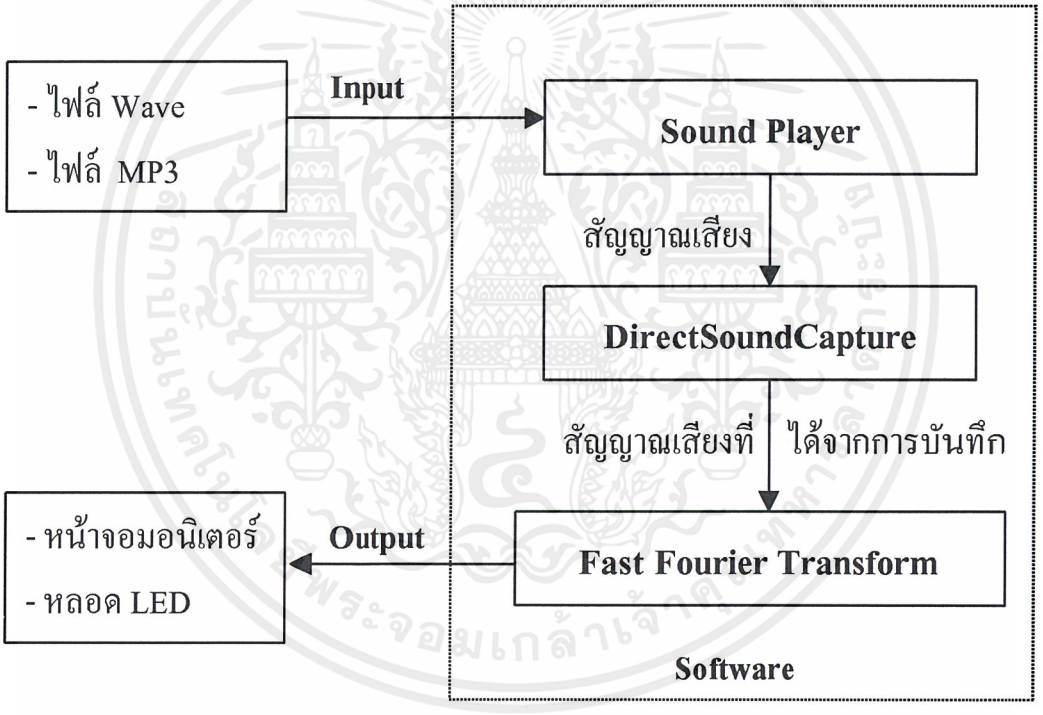
บอร์ด EX-01 นี้สามารถเชื่อมต่อกับ P-Board และได้รับไฟเลี้ยงจากระบบบัส โดยผ่านคอนเน็กเตอร์ P-BUS และ DATA BUS บนบอร์ด LED ที่มีอยู่ทั้งหมด 16 ดวง แบ่งออกเป็น 2 ชุด ชุดละ 8 ดวงโดยได้รับการสั่งงานจากไอซีบัฟเฟอร์ ซึ่งจะแสดงข้อมูลที่ถูกส่งมาจาก P-Board โดยคำสั่ง Out และอาจเป็นข้อมูลเดต้า 8 บิตทาง DATA BUS ที่ 1, 2 และบิตคอนโทรลอีก 4 บิต ถ้าใช้ P-BUS แต่เนื่องจากไม่สามารต่อหลายบัสเข้าพร้อมกันได้ จึงไม่สามารถแสดงข้อมูลด้วยหลอด LED ทั้ง 16 หลอด แต่ถ้าเป็น 12 หลอด คือ เดต้า 8 บิต รวมกับคอนโทรลอีก 4 บิตทาง P- BUS สามารถทำได้ ส่วนในระบบจะใช้เพียงแค่ 8 หลอด คือ เดต้า 8 บิตเท่านั้น เพื่อความสวยงาม เนื่องจาก P-BUS จะแสดงผลหลอด LED ที่อยู่ห่างกัน

บทที่ 3
การออกแบบ และการพัฒนาโปรแกรม

ระบบควบคุมหลอดไฟตามจังหวะเสียงนี้ สามารถแบ่งออกเป็นส่วนต่างๆ เป็น 4 ส่วนคือ

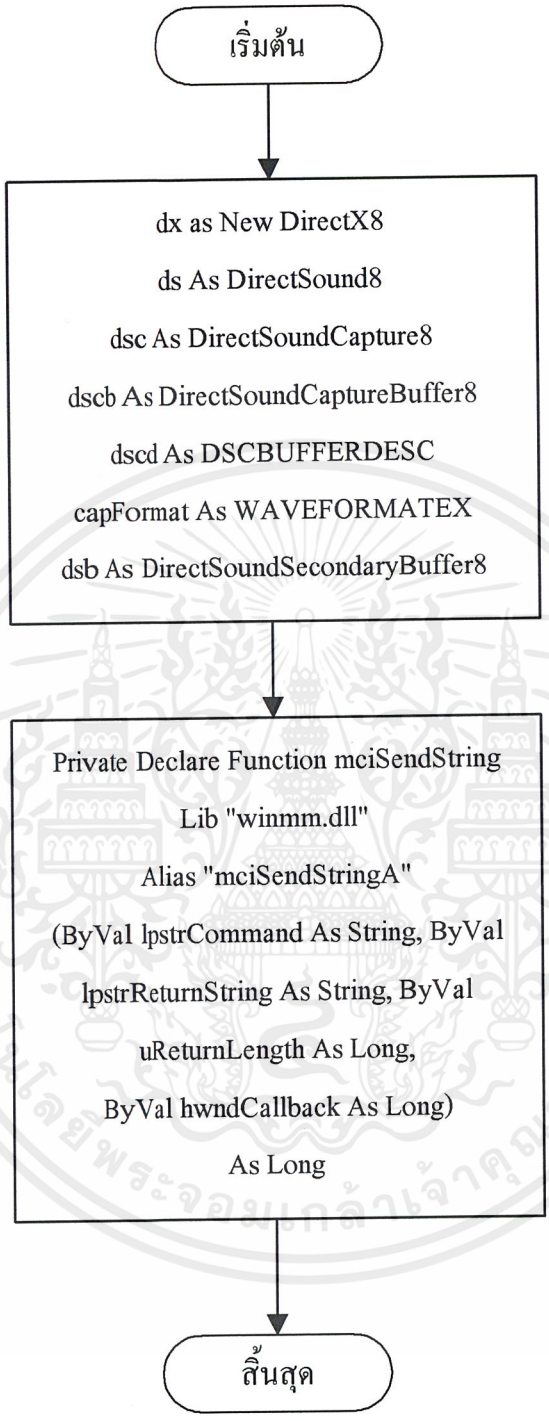
1. ส่วนของการเล่นไฟล์เสียง (Sound Player)
2. ส่วนของการบันทึกเสียงลงบัฟเฟอร์ (Capture Sound)
3. ส่วนของการวิเคราะห์ข้อมูลโดยใช้การแปลงฟูเรียร์อย่างเร็ว (Fast Fourier Transform)
4. ส่วนของการแสดงผลทางอุปกรณ์ภายนอกผ่านทางพอร์ตขนาน

โดยการทำงานของระบบควบคุมหลอดไฟตามจังหวะเสียงนี้จะเป็นไปตาม รูปที่ 3.1 แสดงการทำงานทั้งหมดของระบบ



รูปที่ 3.1 แสดงการทำงานทั้งหมดของระบบ

สำหรับรูปที่ 3.2 จะแสดงถึงการกำหนดค่าเริ่มต้น และตัวแปรต่างๆ ที่ต้องใช้ในส่วนของการเล่นไฟล์เสียงของ Visual Basic 6.0 และ DirectX 8.0 จากขั้นตอนที่ได้พูดถึงไว้ข้างต้นต่อไป



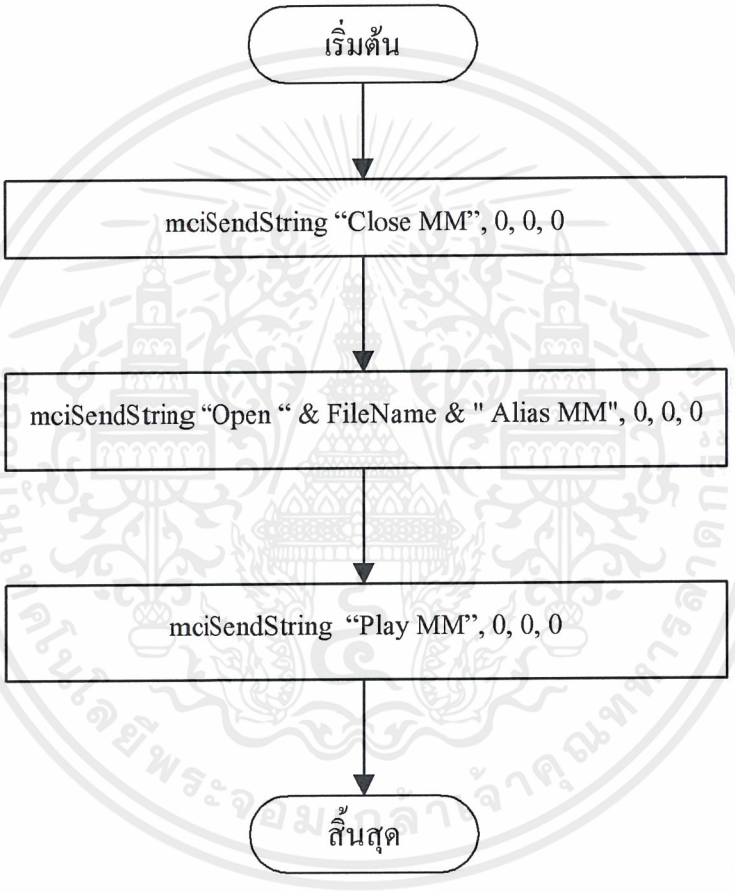
รูปที่ 3.2 แสดงการประกาศค่าตัวแปร และอ๊อบเจ็คต่างๆ ที่ต้องใช้ในการเล่นไฟล์เสียง

3.1 ส่วนของการเล่นไฟล์เสียง (Sound Player)

การเล่นไฟล์เสียงในระบบแบ่งออกเป็นการเล่นไฟล์ MP3 กับไฟล์ Wave โดยมีวิธีการดังนี้

1) การเล่นไฟล์ MP3

หลังจากที่ได้ประกาศใช้ฟังก์ชัน mciSendString ของลิบรารี WinMM.dll ในส่วนของการประกาศ แล้วจะทำการเรียกใช้ฟังก์ชัน mciSendString แล้วตามด้วยอากิวเมนท์ต่างๆ ดังนี้ lpstrCommand As String, lpstrReturnString As String, uReturnLength As Long, hwndCallback As Long ตามลำดับ ซึ่งลำดับการทำงานแสดงในรูปที่ 3.3



รูปที่ 3.3 แสดงการทำงานของส่วนของการเล่นไฟล์เสียง (Sound Player)

2) การเล่นไฟล์ Wave

โดยใช้ DirectSoundSecondaryBuffer8 ของ DirectSound8 ในลิบรารี DirectX 8.0 ก่อนอื่นจะทำการสร้างบัฟเฟอร์สำหรับไฟล์เสียงที่ต้องการให้กับ DirectSoundBuffer จากนั้นจึงใช้เมธทอดของ DirectSoundBuffer ควบคุมการทำงานไม่ว่าจะเล่นหรือหยุดเล่นไฟล์เสียงในบัฟเฟอร์นั้น

3.2 ส่วนของการเก็บบันทึกโดย DirectSoundCapture

มีขั้นตอนดังนี้ (รายละเอียดของการเรียกใช้ฟังก์ชัน DirectSoundCapture รวบรวมไว้ที่ภาคผนวก ก)

3.2.1 สร้างอ๊อบเจกของ DirectX8 :

อ๊อบเจกของ DirectX8 มีไว้เพื่อเป็นตัวแทนของ DirectX8 การจะใช้งานส่วนใดก็ตามของ DirectX8 จะต้องประกาศใช้อ๊อบเจกในคลาส DirectX8นี้เสมอ การสร้างอ๊อบเจกของ DirectX8 จะเกิดตั้งแต่ตอนที่ประกาศตัวแปร ซึ่งแตกต่างจาก อ๊อบเจกอื่นๆ ที่จะต้องประกาศเป็นตัวแปรไว้ก่อน แล้วจึงเรียกใช้ เมทธอดของคลาสแม่สร้างขึ้นอีกครั้งในโปรแกรม ทั้งนี้เพราะว่า ออบเจกของ DirectX8 นั้น เป็นอ๊อบเจกของคลาสแม่ที่ไม่มีคลาสอื่นเป็นคลาสแม่อีกแล้วนั่นเอง การสร้างอ๊อบเจก DirectX8 ใช้คำสั่งประกาศ dx As New DirectX8 โดย dx เป็นชื่ออ๊อบเจกของคลาส DirectX8 จะเห็นได้ว่า ใช้คำสั่ง New เป็นการกำหนดค่าให้กับ dx ทันที

3.2.2 สร้างอ๊อบเจกของ DirectSoundCapture :

อ๊อบเจกของ DirectSoundCapture คือตัวแทนอุปกรณ์ที่ใช้ในการบันทึกเสียง มีหน้าที่ในการสร้างบัฟเฟอร์ที่จะใช้ในการบันทึกเสียง สามารถกำหนดว่าจะใช้อุปกรณ์ใดในการทำงานได้ ในที่นี้ส่งค่า vbNullString ซึ่งหมายถึง ใช้อุปกรณ์บันทึกเสียงที่เป็น Default ของระบบ คำสั่งที่ใช้ในการสร้างคือ

```
Set dsc = dx.DirectSoundCaptureCreate(vbNullString)
```

3.2.3 รวบรวมข้อมูลของ capFormat ที่มีประเภทเป็น WAVEFORMATX อ๊อบเจก capFormat จะใช้เป็นรายละเอียดหนึ่งในการสร้างบัฟเฟอร์บันทึก หมายถึงรูปแบบของสัญญาณเสียงบนบัฟเฟอร์ที่จะสร้าง โดยในโปรแกรม จะสร้าง capFormat ด้วยฟังก์ชัน CreateWaveFormat() มีพารามิเตอร์ 3 ค่า คือ ความถี่สุ่มตัวอย่าง จำนวนช่องสัญญาณ และจำนวนบิตต่อหนึ่งตัวอย่าง รายละเอียดอื่นๆใน capFormat จะถูกคำนวณขึ้นจากพารามิเตอร์ที่ใส่ไปให้กับฟังก์ชัน ในงานนี้ใช้ค่าความถี่สุ่ม 44100 Hz จำนวน 1 ช่องสัญญาณ และ 16 บิตต่อหนึ่งตัวอย่าง คือ

```
capFormat = CreateWaveFormat ( 44100 , 1 , 16 )
```

3.2.4 รวบรวมข้อมูลของ dscd : dscd มีประเภทเป็น DSCBUFFERDESC คือเป็นอ๊อบเจกที่ใช้ เก็บรายละเอียดที่จะใช้สร้างบัฟเฟอร์ในการบันทึก ประกอบด้วย

3.2.4.1 fxFormat นำมาจากค่าของ capFormat

3.2.4.2 lBufferbyte ขนาดของบัฟเฟอร์ ใช้วิธีกำหนดจาก AveragePerSec ใน fxFormat เช่น ต้องการให้บัฟเฟอร์ยาว 10 วินาทีก็กำหนดเป็น AveragePerSec * 10 ในที่นี้ กำหนดให้บัฟเฟอร์มี ความยาว 1 วินาที

3.2.4.3 lFlag ค่าแฟลก : กำหนดเป็น DSCBCAPS_WAVEMAPPED คือในกรณีที่เป็นรูปแบบสัญญาณที่อุปกรณ์ไม่สนับสนุนก็จะใช้เป็น WaveMapper ของ Win32 แทน

3.2.5 ทำการสร้างบัฟเฟอร์บันทึก : บัฟเฟอร์บันทึกเป็นตัวแทนของพื้นที่ที่ใช้เก็บข้อมูลสัญญาณเสียง โดยมากการทำการเก็บบันทึกส่วนใหญ่จะเป็นเมทธอดของบัฟเฟอร์บันทึกนี้

คำสั่งที่ใช้ในการสร้างบัฟเฟอร์บันทึกคือ

dscb = dsc.CreateCaptureBuffer(dscd)

โดยประกาศ dscb As directSoundCaptureBuffer บัฟเฟอร์ dscb ที่ได้จะมีคุณสมบัติ ตามที่ระบุไว้ใน dscd (DirectSoundCaptureBuffer Description)

3.2.6 เมื่อสร้างบัฟเฟอร์บันทึกแล้วก็พร้อมที่จะใช้งานได้ทันที เมทธอดที่สำคัญของบัฟเฟอร์ได้แก่

3.2.6.1 Start : dscb.Start จะเป็นการเริ่มต้นเก็บบันทึกสัญญาณเสียง โดยเริ่มที่ต้นบัฟเฟอร์เสมอ ดังนั้นเมื่อบันทึกใหม่สัญญาณเก่าจะถูกทับไปหมดเสมอ จะหยุดเมื่อเรียกใช้เมทธอด Stop เท่านั้น

3.2.6.2 Stop : dscb.Stop บัฟเฟอร์ที่เก็บบันทึกอยู่จะหยุดการเก็บบันทึก

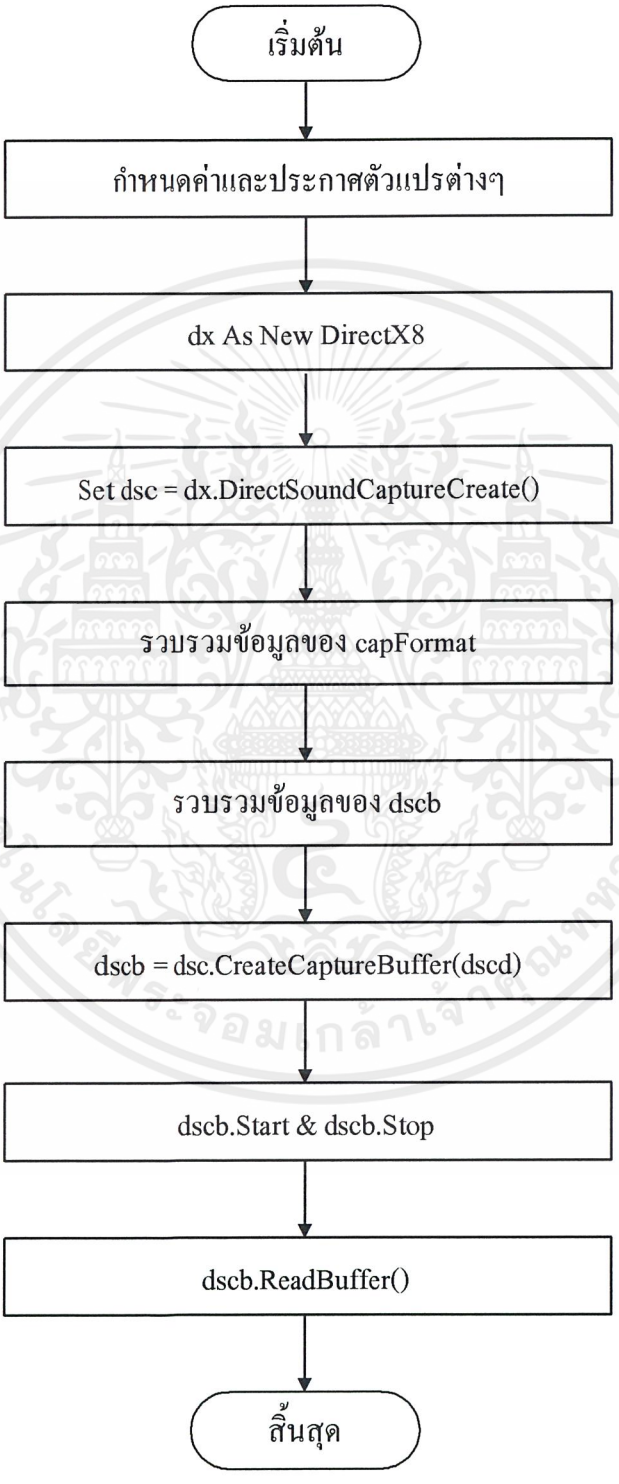
3.2.6.3 ReadBuffer : จะอ่านค่าสัญญาณบนบัฟเฟอร์ลงบนบัฟเฟอร์ของโปรแกรม ซึ่งก็คืออาร์เรย์นั่นเอง ใช้พารามิเตอร์ 4 ค่า ได้แก่ ไบต์แรกที่จะเริ่มอ่าน , จำนวนไบต์ที่จะอ่าน บัฟเฟอร์ที่ใช้รับข้อมูล , แฟลก (กำหนดเป็น Default)

dscb.ReadBuffer (0 , 512, ByteBuffer , Default)

- ไบต์ที่อ่านเริ่มที่ 0 คือต้นบัฟเฟอร์
- จำนวนไบต์ที่อ่านคือ 512 ไบต์ เนื่องจากสัญญาณเป็น 16 บิตดังนั้นตัวกอย่าง 1 ตัวใช้เนื้อที่ 2 ไบต์
- ดังนั้น ByteBuffer ก็ต้องเป็นอาร์เรย์ของ Integer ซึ่งจะได้สัญญาณมา 256 ตัว

เมทธอด ReadBuffer เป็นตัวกลางสำคัญที่ทำให้สามารถนำสัญญาณมาวิเคราะห์ได้โดยการบันทึกและอ่านลงสู่บัฟเฟอร์นั้นจะทำเป็นช่วงเวลาสั้นๆ เพื่อนำข้อมูลที่ได้ไปแสดงผลเป็นชุดๆ โดยส่งต่อไปที่กระบวนการ

FFT การเก็บสัญญาณด้วย DirectSoundCapture และการทำ FFT สืบเนื่องไปจนการแสดงผลนั้น จะทำต่อเนื่องกันไปในขณะที่มีการเล่นไฟล์เพลงอยู่เท่านั้น และการเก็บสัญญาณเพื่อคำนวณและแสดงผลจะทำไปจนกว่าเพลงจะจบหรือมีการสั่งหยุดเล่นไฟล์เสียง หรือไม่ก็ปิดโปรแกรม



รูปที่ 3.4 แสดงการทำงานของส่วนของการบันทึกสัญญาณเสียงด้วย DirectSoundCaptureBuffer8

3.3 ส่วนของการวิเคราะห์ข้อมูลโดยใช้การแปลงฟูเรียร์อย่างเร็ว (Fast Fourier Transform)

เนื่องจากข้อมูลสัญญาณเสียงที่ได้มาจากอาร์เรย์ที่ทำการเก็บบันทึกสัญญาณเสียงที่ได้จากบัฟเฟอร์ของ DirectSound นั้นเป็นข้อมูลที่ อยู่ในรูปของโดเมนที่เป็นค่าของเวลา (Time Domain) เราจึงใช้ทฤษฎีการแปลงฟูเรียร์อย่างเร็ว (Fast Fourier Transform : FFT) เพื่อแปลงชุดของข้อมูลนั้นให้อยู่ในรูปของโดเมนที่เป็นค่าของความถี่แทน (Frequency Domain) ซึ่งจะทำให้การวิเคราะห์ข้อมูลนั้นสามารถกระทำได้โดยง่ายและสะดวกยิ่งขึ้น

ในส่วนของการแปลงฟูเรียร์อย่างเร็วนี้ก็จะมีขั้นตอนการทำงานตามลำดับต่อไปนี้

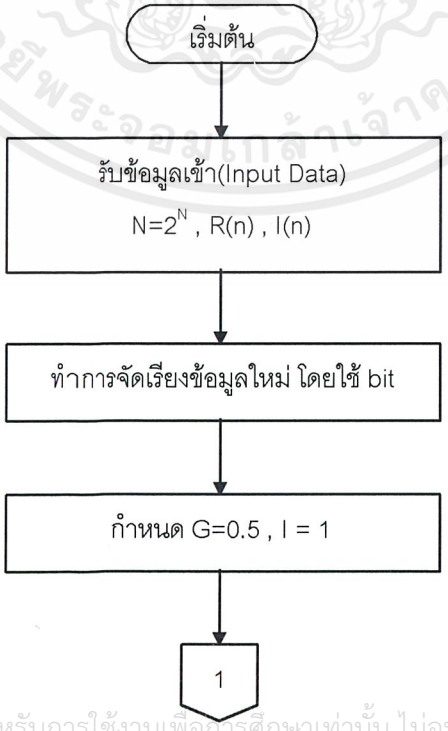
3.3.1 รับข้อมูลสัญญาณอินพุทจากอาร์เรย์ที่ได้รับข้อมูลจากบัฟเฟอร์ที่ใช้ทำการบันทึกเก็บไว้ ของ DirectX โดยจะใช้ข้อมูลจำนวน 256 ตัว

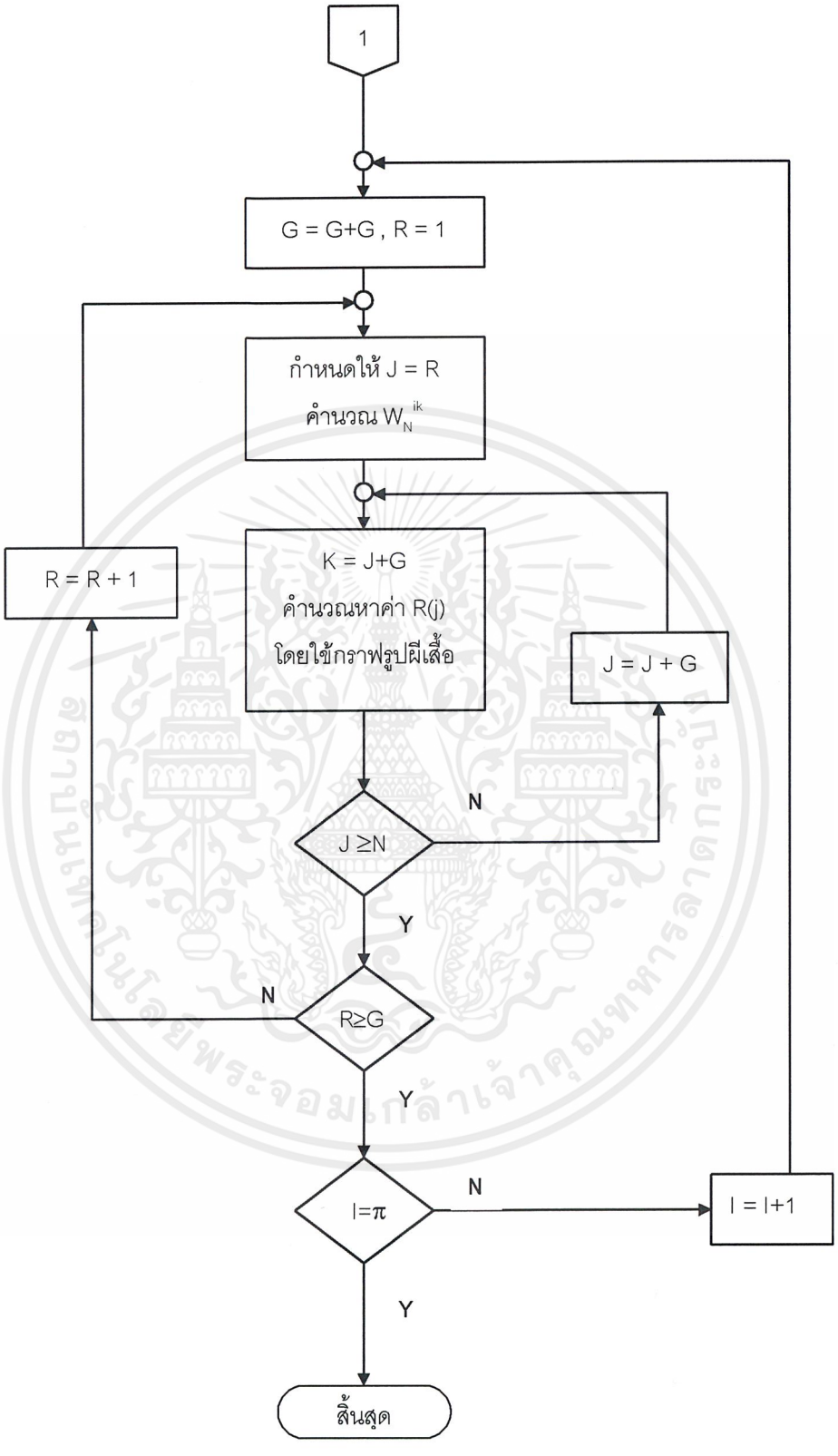
3.3.2 ทำการจัดเรียงข้อมูลใหม่โดยใช้ bit reversal

3.3.3 คำนวณหาค่าข้อมูลที่ m+1 คือ

1) $x_{m+1}(p) = x_m(p) + W_N x_m(p)$
2) $x_{m+1}(q) = x_m(p) - W_N x_m(q)$
3) นำค่า $x_{m+1}(p)$ ไปเก็บไว้ที่ $x_m(p)$
4) นำค่า $x_{m+1}(q)$ ไปเก็บไว้ที่ $x_m(q)$
5) กลับไปขั้นตอนที่ 1) ทำการคำนวณจนค่า m+1 = n โดยค่า m เริ่มจาก 0 และ r = f(m,p,q)

3.3.4 จะได้ค่าของข้อมูลสัญญาณที่ผ่านการแปลงฟูเรียร์อย่างเร็ว เก็บในรูปของอาร์เรย์จำนวน 1024 ตัว





รูปที่ 3.5 แสดงขั้นตอนการแปลงฟูเรียร์อย่างเร็ว

3.4 ส่วนของการแสดงผล

ในส่วนนี้เราจะแบ่งการแสดงผลออกเป็น 2 ส่วนคือ

3.4.1 การแสดงผลออกทางจอภาพ

การแสดงผลทางจอภาพทำโดยใช้ อ็อบเจก Line ในโปรแกรม Visual Basic สร้างตารางกราฟฟิกขึ้น และเส้นที่ใช้แทนขนาดของความถี่ก็เป็นเส้น Line เช่นเดียวกัน

สำหรับรายละเอียดของการแสดงผลทางจอภาพ คือ

- ทำการแสดงผลเส้นสเปกตรัมทั้ง 256 เส้นที่ได้จากการทำ FFT
- ความสูงมากที่สุดของเส้นสเปกตรัมคือ 4000 หน่วย
- ขนาดของเส้นสเปกตรัมจะถูกย่อขนาดให้พอดีกับตารางกราฟฟิก ประมาณ 1 ใน 20
- ทำการกำหนดค่าที่ย่อแล้วให้กับค่า Y1 ของเส้นที่ใช้แสดงผลใดๆ ซึ่งต้องมีการกลับค่าเล็กน้อยเนื่องจากในจอภาพนั้นค่า Y ที่ต่ำกว่าจะอยู่สูงกว่า
- ตารางกราฟฟิกที่ใช้มีขนาด 8000 * 4000 หน่วย
- เส้นที่อยู่ซ้ายสุดจะเป็นเส้นแรกในชุดของผลลัพธ์ของ FFT
- การแสดงผลจะเป็นจังหวะเดียวกับที่มีการบันทึกสัญญาณเสียงแล้วทำ FFT ในแต่ละรอบเป็นเวลาประมาณ 20 – 50 milliSecond แต่ไม่สั้นกว่า 10 millisecond

3.4.2 การแสดงผลออกทางหลอด LED ผ่านทางพอร์ตขนาน

เนื่องจากการเขียนโปรแกรมด้วย Visual Basic 6.0 ไม่มีคำสั่งในการติดต่อกับพอร์ตโดยตรง เหมือนในบางภาษา ดังนั้นเพื่อให้สามารถติดต่อกับพอร์ตขนานได้จึงจำเป็นต้องเพิ่มโปรแกรมบางตัวเข้าไป โดยโปรแกรมดังกล่าวจะอยู่ในรูปของไฟล์ DLL (Dynamic Link Library) ซึ่งก็คือไฟล์ inpout32.dll (เพิ่มเข้าไปในโฟลเดอร์ System ของ Windows) แล้วทำการประกาศเรียกใช้ฟังก์ชัน โดยประกาศได้ดังนี้

```
#If Win32 Then
'Declare Inp and Out for port I/O
Public Declare Function Inp Lib "inpout32.dll" _
Alias "Inp32" (ByVal PortAddress As Integer) As Integer
Public Declare Sub Out Lib "inpout32.dll" _
Alias "Out32" (ByVal PortAddress As Integer, ByVal Value As Integer)
#Else
Declare Function Inp Lib "InpOut.DLL" (ByVal Port%) As Integer
Declare Sub Out Lib "InpOut.DLL" (ByVal Port%, ByVal Value%)
#End If
```

รูปที่ 3.6 แสดงขั้นตอนการประกาศเพื่อเรียกใช้ฟังก์ชันควบคุมหลอด LED

เราจะสามารถใช้คำสั่ง Inp และ Out เพื่อติดต่อกับพอร์ตขนานได้ แล้วเราจะสามารถติดต่อกับบอร์ด P-Board ซึ่งเป็นบอร์ดเชื่อมต่อพอร์ตขนานได้ จากนั้น P-Board จะติดต่อกับบอร์ด EX-01 ซึ่งเป็นบอร์ด LED มอนิเตอร์ ซึ่งเราต้องการแสดงผลแอมปลิจูดของเส้นสเปกตรัมที่ความถี่ต่างๆ (60, 170, 310, 600, 1k, 3k, 6k, 12k, 14k, 16k Hz) ซึ่งได้จากการคำนวณออกทางบอร์ดนี้ โดยเราจะติดต่อผ่านทาง พอร์ตข้อมูล (Data) ของ LPT1 ซึ่ง จะควบคุมหลอดไฟทั้งหมด 8 หลอด หลังจากนั้นเราจะนำข้อมูลที่ได้จากการแปลง FFT ในความถี่ต่างๆ ตามที่ต้องการมาแปลงเป็นระดับ 0-8 ระดับ (ตามจำนวนหลอดไฟ LED) แล้วส่งค่าไปยังพอร์ตขนานเพื่อแสดงผล โดย จะใช้คำสั่ง

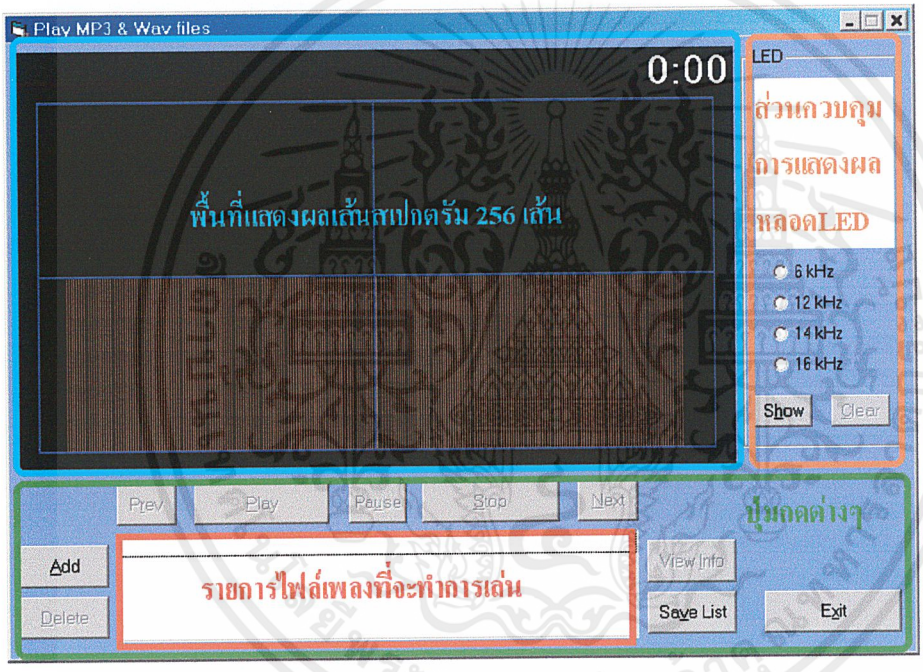
Out &H378, &H(ค่าของระดับ 9 ระดับ ในรูปเลขฐาน 16)

บทที่ 4
การใช้งานโปรแกรม

อินเตอร์เฟสของโปรแกรมจะถูกแบ่งออกเป็น 4 ส่วนหลัก คือ

1. พื้นที่แสดงผลเส้นสเปกตรัม 256 เส้น
2. ปุ่มควบคุมการทำงานต่างๆ
3. รายการไฟล์เพลงที่จะทำการเล่น
4. ส่วนควบคุมการแสดงผลหลอด LED

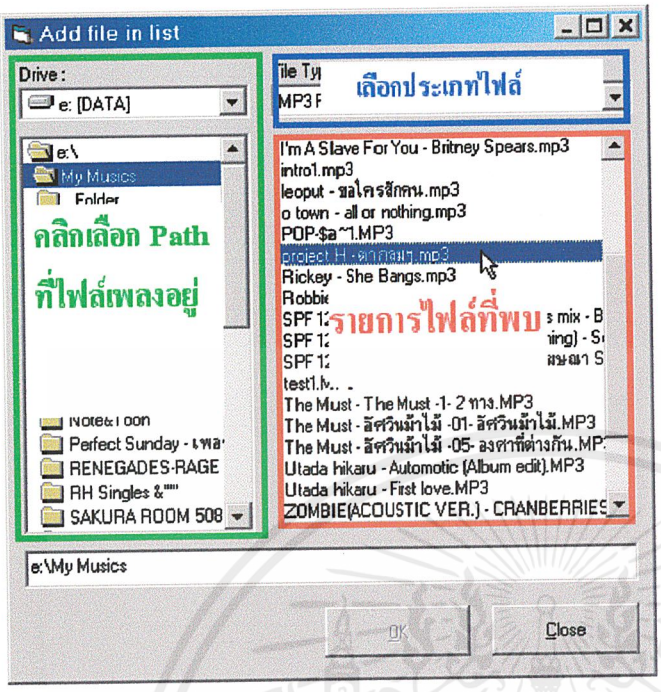
ซึ่งอินเตอร์เฟสของโปรแกรมนั้นแสดงให้เห็นดังรูปที่ 4.1



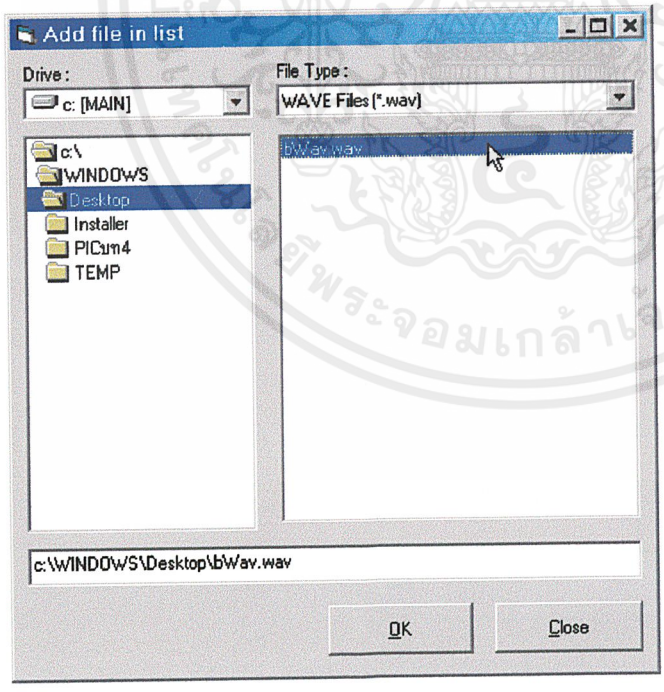
รูปที่ 4.1 อินเทอร์เฟสของโปรแกรม

4.1 การจัดการรายการเพลงที่จะทำการเล่น

4.1.1 การเพิ่มเพลงเข้าสู่รายการ : คลิกปุ่ม Add จะปรากฏหน้าต่างดังรูป 4.2 และรูปที่ 4.3 ผู้ใช้จะต้องทำการเลือกไฟล์ที่ต้องการจะเล่น โดยที่สามารถทำการเลือกไฟล์ที่ต้องการจะเล่นได้ 2 ประเภทคือไฟล์ Wave และไฟล์ Mp3 เพิ่มเข้าไปในรายการไฟล์เพลงที่จะทำการเล่น

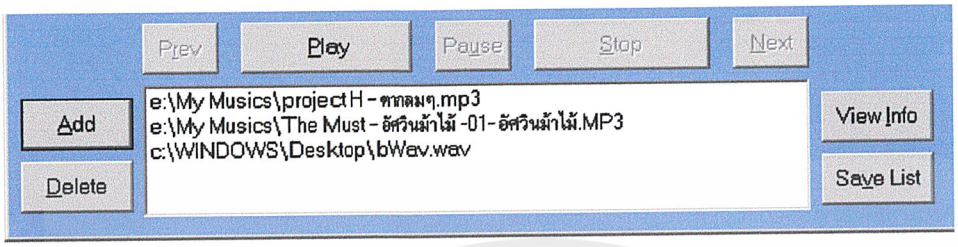


รูปที่ 4.2 แสดงหน้าต่างการเพิ่มรายการ – เลือกแสดงไฟล์ประเภท MP3



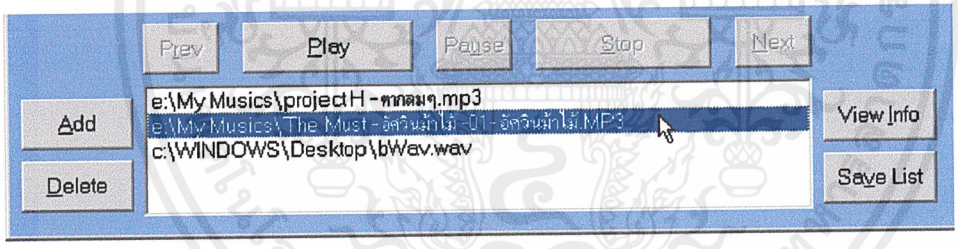
รูปที่ 4.3 แสดงหน้าต่างการเพิ่มรายการ– เลือกแสดงไฟล์ประเภท WAVE

โดยทำการคลิกที่ File Type เพื่อเลือกว่าจะทำการแสดงไฟล์ประเภทใด และเลือก Path ที่ต้องการแสดง ไฟล์เพลงที่พบทั้งหมดใน Path นั้นจะแสดงในลิสต์บ๊อก เลือกไฟล์ใดไฟล์หนึ่งที่พบแล้วคลิกปุ่ม OK ไฟล์ที่เลือกจะถูกเพิ่มต่อท้ายเข้าไปในรายการเพลงดังรูปที่ 4.4

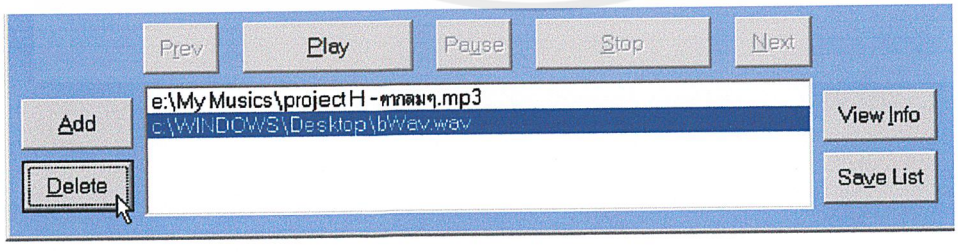


รูปที่ 4.4 แสดงรายการเพลงที่ทำการเพิ่มไฟล์เพลงแล้ว

4.1.2 การลบเพลงออกจากรายการเพลง : คลิกเลือกเพลงให้เกิดแถบสีที่เพลงที่ต้องการลบออกจากรายการ ดังรูปที่ 4.5 จากนั้นคลิกปุ่ม Delete เพลงนั้นจะถูกลบออกจากรายการและเพลงอื่นที่ต่อจากเพลงนั้นจะขยับเลื่อนขึ้นมาแทนที่ ดังรูปที่ 4.6



รูปที่ 4.5 แสดงการเลือกเพลงที่จะลบออกจากรายการ

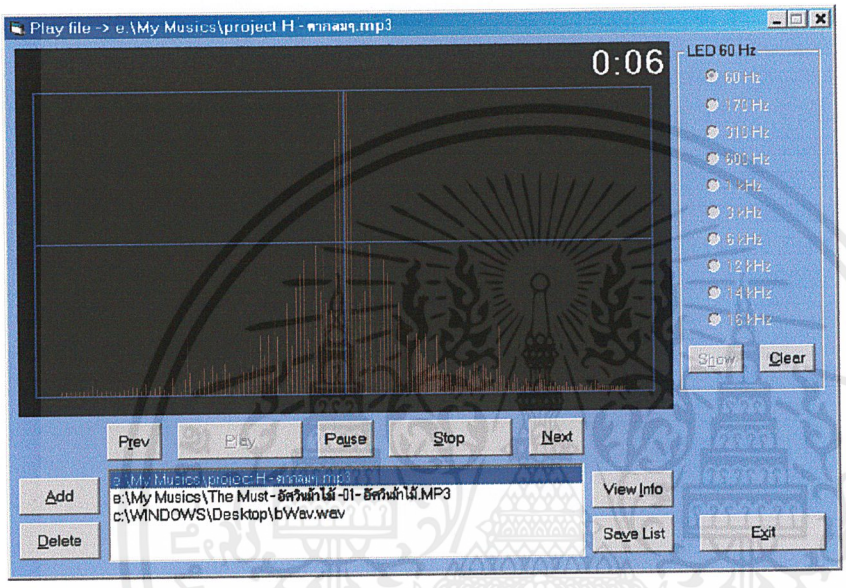


รูปที่ 4.6 แสดงผลที่เกิดหลังการลบเพลงออกจากรายการ

4.2 การควบคุมการเล่นเพลงในรายการ

4.2.1 การเริ่มการเล่นเพลง : ทำได้ 3 วิธีคือ

- ดับเบิ้ลคลิกที่เพลงในรายการ
- คลิกที่เพลงในรายการให้เกิดแถบสี แล้วคลิกปุ่ม Play
- คลิกที่เพลงในรายการให้เกิดแถบสี แล้วกดแป้นคีย์บอร์ด Alt + P (กด Alt ค้างไว้ แล้วกด P)



รูปที่ 4.7 แสดงการเล่นเพลงด้วยการคลิกปุ่ม Play

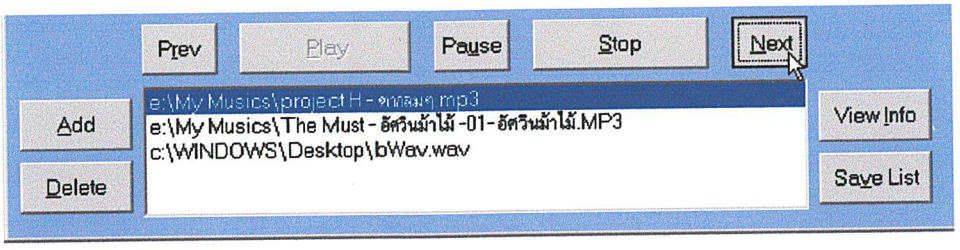
4.2.2 การหยุดเล่นเพลงที่กำลังเล่นอยู่ : ทำได้ 2 วิธีคือ

- คลิกปุ่ม Stop
- กดแป้นคีย์บอร์ด Alt + S

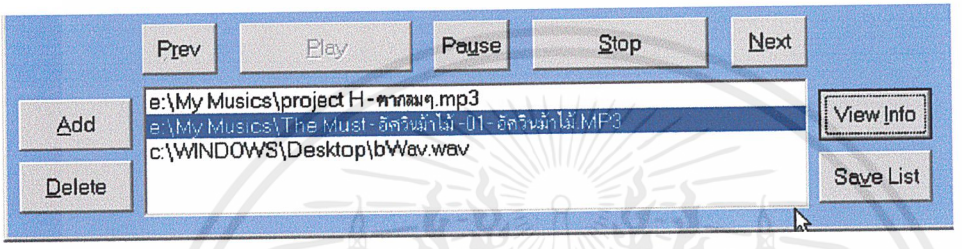
4.2.3 การข้ามไปเล่นเพลงถัดไปในรายการ : เมื่อเล่นเพลงใดๆอยู่สามารถข้ามไปเล่นเพลงถัดไปได้ทันทีทำได้โดย

- คลิกปุ่ม Next
- กดแป้นคีย์บอร์ด Alt + N

แถบสีจะเลื่อนไปสู่เพลงถัดไปและจะทำการเล่น ถ้าเพลงที่เล่นอยู่เป็นเพลงสุดท้ายของรายการแล้ว จะกลับไปทำการเล่นเพลงแรกของรายการ



รูปที่ 4.8 แสดงภาพก่อนการข้ามเล่นเพลงถัดไป

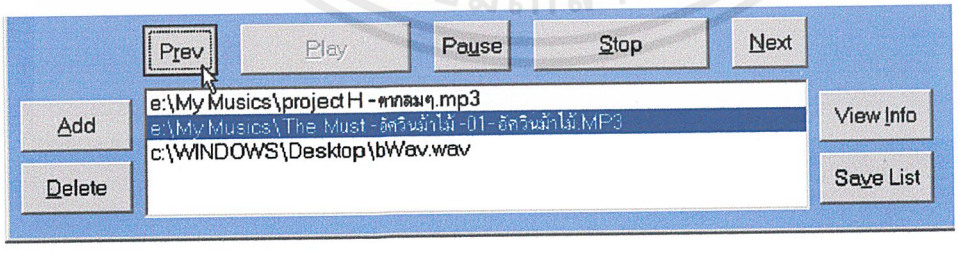


รูปที่ 4.9 แสดงภาพหลังการข้ามเล่นเพลงถัดไป

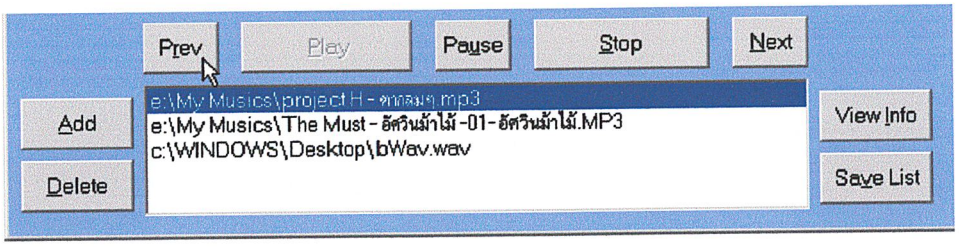
4.2.4 การย้อนกลับไปเล่นเพลงก่อนหน้า : ทำได้ขณะเล่นเพลงใดๆเช่นเดียวกับการข้ามไปเล่นเพลงถัดไปทำได้โดย

- คลิกปุ่ม Prev
- กดแป้นคีย์บอร์ด Alt + R

แถบสีจะเลื่อนไปสู่เพลงก่อนหน้าและจะทำการเล่น ถ้าเพลงที่เล่นอยู่ เป็นเพลงแรกของรายการแล้ว จะไปทำการเล่นเพลงท้ายสุดของรายการ



รูปที่ 4.10 แสดงภาพก่อนการย้อนกลับไปเล่นเพลงก่อนหน้า

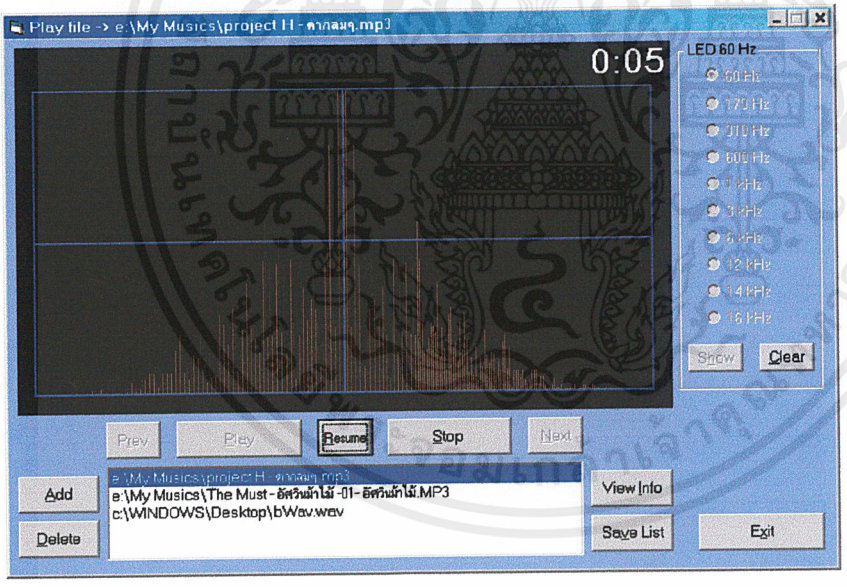


รูปที่ 4.11 แสดงภาพหลังการย้อนกลับไปเล่นเพลงก่อนหน้า

4.2.5 การหยุดเล่นเพลงชั่วคราว : หยุดเล่นเพลงที่กำลังเล่นไว้ชั่วคราว ทำได้โดย

- คลิก Pause
- กดแป้นคีย์บอร์ด Alt + U

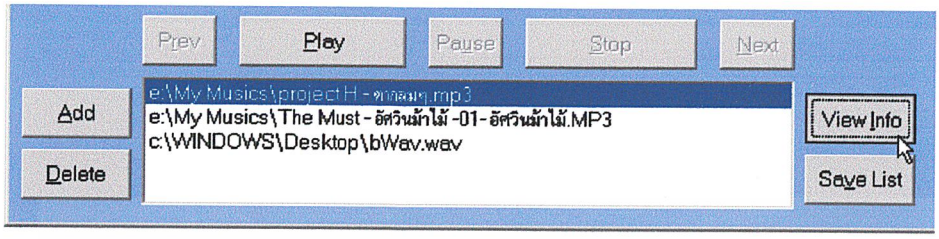
ปุ่มกดจะเปลี่ยนชื่อเป็น Resume คลิกที่ปุ่มอีกครั้งจะทำการเล่นต่อหรือกดแป้นคีย์บอร์ด Alt + R ก็ได้



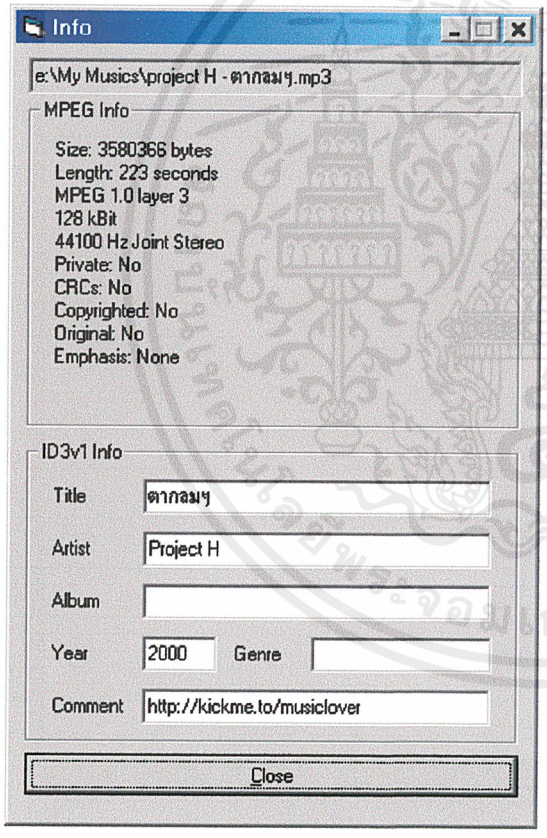
รูปที่ 4.12 แสดงการหยุดเล่นเพลงชั่วคราว

4.3 การดูข้อมูลไฟล์ MP3

ทำได้โดยเลือกเพลง MP3 ในรายการแล้วคลิกปุ่ม View Info หรือกดแป้นคีย์บอร์ด Alt + I ดังรูป 4.13 และจะปรากฏหน้าต่างแสดงข้อมูลดังรูปที่ 4.14



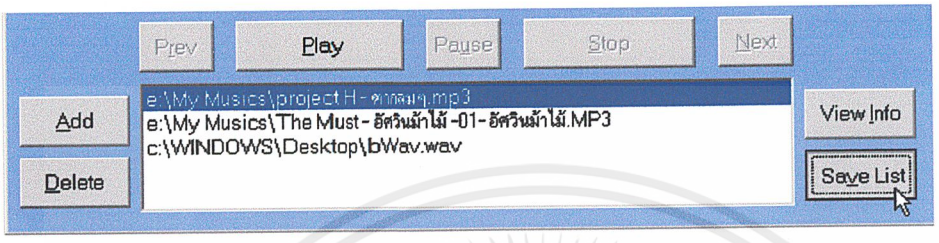
รูปที่ 4.13 แสดงการคลิกดูข้อมูลไฟล์ MP3



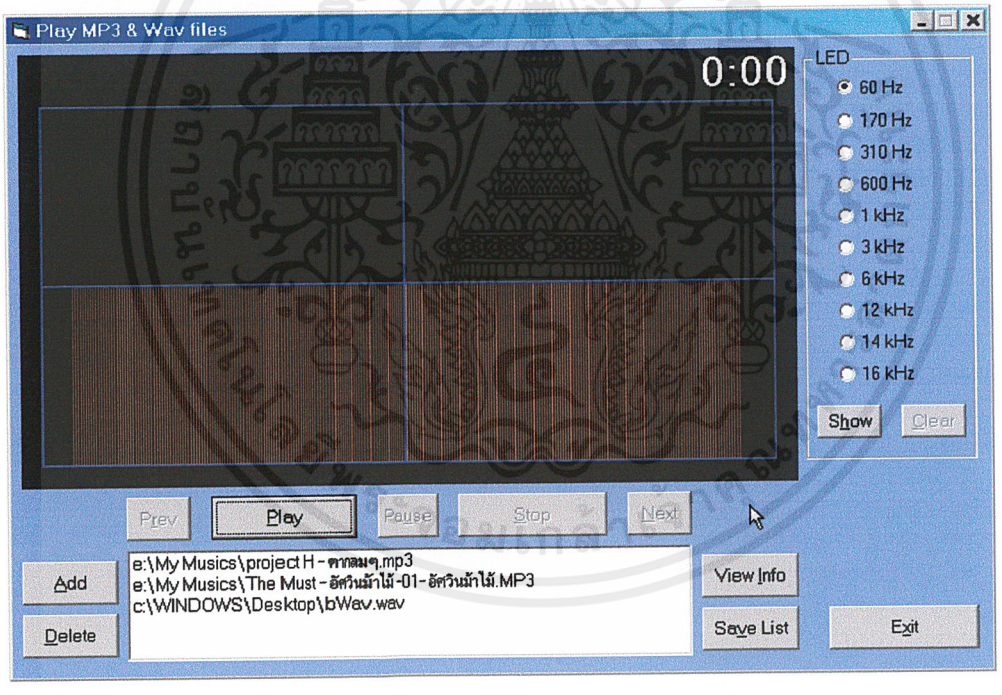
รูปที่ 4.14 แสดงหน้าต่างข้อมูลไฟล์ MP3

4.4 การเก็บรายการเพลงลงไฟล์

สามารถเก็บรายการเพลงปัจจุบันซึ่งจะอ่านขึ้นโดยอัตโนมัติเมื่อเปิดโปรแกรมครั้งต่อไป ทำได้โดย คลิก Save List หรือกดแป้นคีย์บอร์ด Alt + V ไฟล์ที่ใช้เก็บรายการชื่อ list.txt ไฟล์เป็นแบบ Sequetial



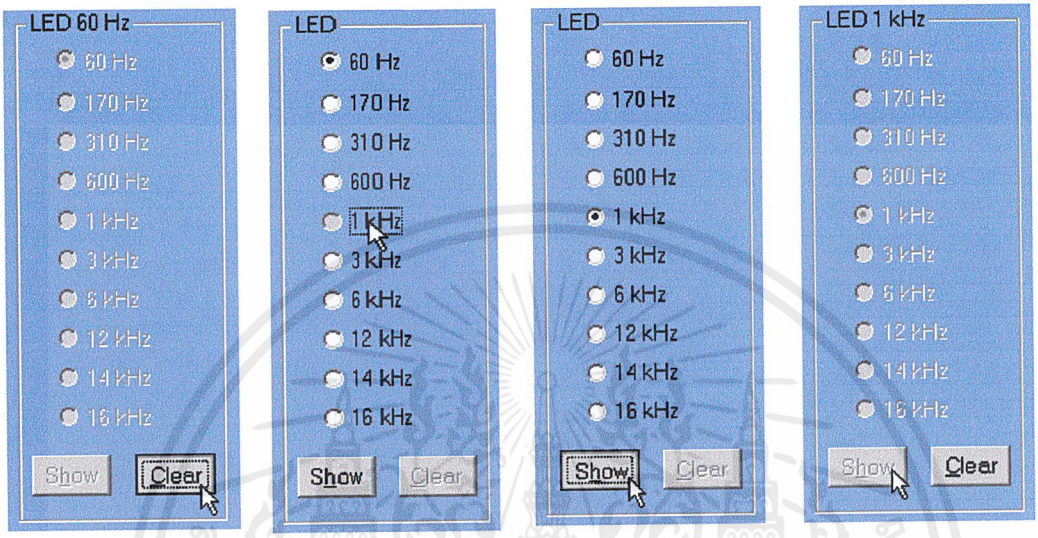
รูปที่ 4.15 แสดงการคลิกคำสั่งเก็บรายการเพลงลงไฟล์



รูปที่ 4.16 แสดงการอ่านรายการที่เก็บไว้ เมื่อเรียกโปรแกรมอีกครั้ง

4.5 การเลือกเส้นความถี่ที่จะแสดงผลทางหลอด LED

การแสดงผลทางหลอด LED นั้นสามารถทำการแสดงได้ครั้งละ 1 เส้นความถี่ สามารถเลือกให้หลอด LED ทำการแสดงผลในเส้นความถี่ที่ต้องการให้แสดงผลได้ โดยมีความถี่ที่ให้เลือกดังนี้ 60 , 170 , 310 , 600 , 1000 , 3000 , 6000 , 12000 , 14000 , 16000 Hz ดังรูป 4.17



A B C D

รูปที่ 4.17 A : แสดงการหยุดการแสดงผลเพื่อเลือกความถี่ใหม่ที่จะแสดงผล

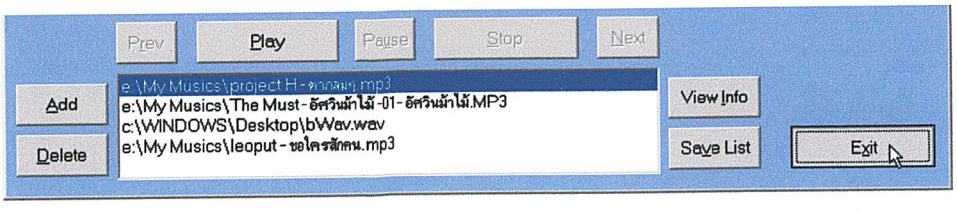
B : แสดงการเลือกความถี่ใหม่ที่ต้องการ

C : แสดงการคลิกเริ่มแสดงผลด้วยความถี่ใหม่

D : แสดงการเลือกความถี่ใหม่เสร็จสิ้น ที่ด้านบนแสดงความถี่ปัจจุบันที่แสดงอยู่

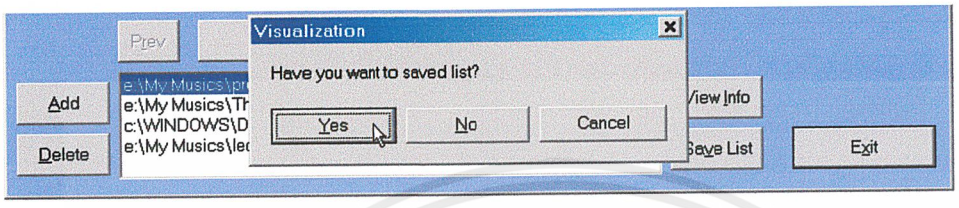
4.6 การออกจากโปรแกรม

การออกจากโปรแกรมนั้นสามารถกระทำได้ 2 วิธีคือ คลิกปุ่ม Exit หรือกดแป้นคีย์บอร์ด Alt + X เพื่อออกจากโปรแกรม ดังรูป 4.18



รูปที่ 4.18 แสดงการคลิกปุ่มเพื่อออกจากโปรแกรม

ในกรณีที่รายการเพลงมีการเปลี่ยนแปลงจะปรากฏหน้าต่างดังรูป 4.19 เพื่อถามว่าต้องการเก็บรายการไฟล์เพลงที่ต้องการจะเล่น ก่อนออกจากโปรแกรมหรือไม่ คลิก Yes เพื่อทำการเก็บรายการและต้องการออกจากโปรแกรม คลิก No เมื่อไม่ต้องการเก็บรายการเพลงปัจจุบันและออกจากโปรแกรม และคลิก Cancel เพื่อกลับไปใช้งานโปรแกรม ดังแสดงในรูป 4.19



รูปที่ 4.19 แสดงหน้าต่างถามการเก็บรายการก่อนออกจากโปรแกรม

บทที่ 5
สรุปและข้อเสนอแนะ

5.1 บทสรุป

จากโปรแกรมควบคุมหลอดไฟตามจังหวะของเสียง เราสามารถสรุปได้ดังนี้

5.1.1 โปรแกรมควบคุมหลอดไฟตามจังหวะของเสียงนี้สามารถทำการเล่นไฟล์เสียงประเภท ไฟล์ MP3 และ ไฟล์ Wave ได้

5.1.2 ความสามารถควบคุมการเล่นไฟล์เสียงที่มีได้แก่ การเริ่มเล่นไฟล์เพลงที่ถูกเลือกในรายการ, การหยุดเล่นไฟล์เพลงที่เล่นอยู่, การข้ามไปเล่นไฟล์เพลงที่อยู่ก่อนของไฟล์ที่เล่นอยู่ในปัจจุบัน, การข้ามไปเล่นไฟล์เพลงที่อยู่ถัดไปของไฟล์ที่เล่นอยู่ในปัจจุบัน, การหยุดเล่นชั่วคราวและเล่นต่อไป

5.1.3 มีรายการไฟล์ที่จะทำการเล่นแบบคิว ซึ่งสามารถเพิ่มไฟล์เพลงลงในรายการและลบออกจากรายการได้ รายการที่เลือกไว้สามารถทำการเก็บไว้ได้โดยในการเปิดโปรแกรมครั้งหน้า รายการไฟล์เพลงจะเป็นรายการที่ได้ทำการเก็บไว้ล่าสุด สามารถดูรายละเอียดของไฟล์เพลงที่อยู่ในรายการได้

5.1.4 โปรแกรมสามารถทำการแสดงผลทางหน้าจอ ได้เป็นเส้นความถี่ 256 เส้น และสำหรับการแสดงผลทางหลอด LED เลือกแสดงผล 1 ความถี่จาก 10 ความถี่ ได้แก่ ความถี่ 60 , 170 , 310 , 600 , 1000 , 3000 , 6000 , 12000 , 14000 และ 16000 Hz แสดงผลเป็น 8 ระดับ

5.2 ข้อจำกัดในการใช้งานโปรแกรม

โปรแกรมควบคุมหลอดไฟตามจังหวะของเสียงนี้ มีข้อจำกัดทางด้านฮาร์ดแวร์ และ ซอฟท์-แวร์ ดังต่อไปนี้

5.2.1 โปรแกรมทำงานบนเครื่องพีซีสแตนอโลน (PC Stand Alone)

5.2.2 ใช้ได้กับระบบปฏิบัติการ Windows 9X, Me เท่านั้น

5.2.3 ภายในไดเร็คทอรี System ของ Windows จะต้องมีไฟล์ WinMM.dll, Kernel32.dll

5.3 บทการวิจารณ์และแนวทางการพัฒนา

5.3.1 เนื่องจากโปรแกรมนี้ไม่สามารถทำการเล่นไฟล์เสียงประเภท ไฟล์ MIDI ได้ ดังนั้นควรจะมีการพัฒนาโปรแกรมให้สามารถเล่นไฟล์เสียงได้ในทุกๆประเภท

5.3.2 เนื่องจากการแสดงผลทางหน้าจอ เราใช้เส้น Line หลายๆเส้น ทำให้การทำงานของระบบช้าลงไป อาจจะใช้วิธีอื่นๆในการแสดงผลทางหน้าจอ เพื่อที่จะไม่ส่งผลกระทบกับระบบมากนัก

5.3.3 โปรแกรมนี้จะขึ้นกับการ์ดเสียง เนื่องจากคุณภาพของสัญญาณเสียงที่จะถูกวิเคราะห์นั้นขึ้นอยู่กับการกำหนดคุณภาพของบัฟเฟอร์ที่ใช้ซึ่งขึ้นกับการ์ดเสียงโดยตรง กล่าวคือ ถ้าการ์ดเสียงไม่มีคุณภาพในการบันทึกเสียงที่ดีพอ การแสดงผลก็จะเกิดข้อผิดพลาดขึ้น

5.3.4 เนื่องมาจากบอร์ดแสดงผล LED ที่ใช้สามารถแสดงผลได้เพียงแค่ 8 ช่องเท่านั้นจึงไม่สามารถทำการแสดงผลในทุกความถี่ที่ต้องการพร้อมๆกันได้ ดังนั้นถ้ามีการใช้บอร์ดแสดงผลที่สามารถแสดงผล LED ได้มากกว่านี้ก็จะทำให้ระบบสามารถทำงานได้ดีขึ้น

ดังนั้นเพื่อประโยชน์ในการพัฒนาต่อไป จึงแนะนำให้ทำการทดลองกับบอร์ด LED ที่สามารถแสดงผลได้มากกว่านี้ และควรจะพัฒนาระบบควบคุมหลอดไฟตามจังหวะเสียงนี้ให้สามารถทำงานได้กับไฟล์เสียงในทุกๆประเภท

ภาคผนวก

ภาคผนวก ก.
การติดตั้งโปรแกรม

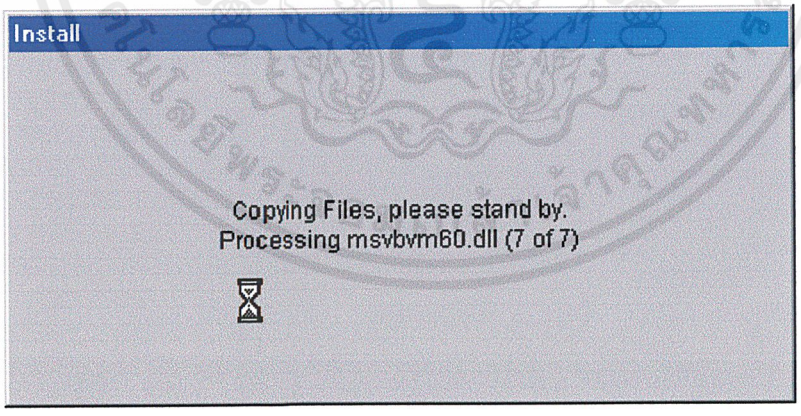
ภายใน CD-ROM ติดตั้งโปรแกรมประกอบด้วยไฟล์ Setup.exe, Setup.LST, TCLS.zip และโฟลเดอร์ Support

ขั้นตอนในการติดตั้งโปรแกรม

1. ใส่ CD-ROM ใน Drive CD
2. เปิด My Computer เข้าไปใน Drive CD ที่มีแผ่นอยู่

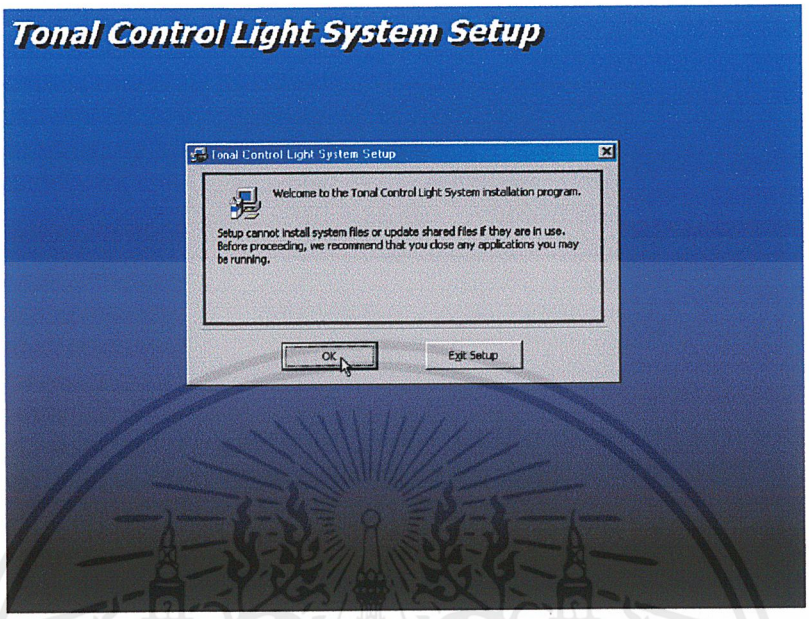
รูปที่ ก-1 แสดงไฟล์ที่ใช้ในการติดตั้งโปรแกรม

3. เรียกไฟล์ Setup.exe โปรแกรมติดตั้งจะปรากฏขึ้นดังรูป ก-2



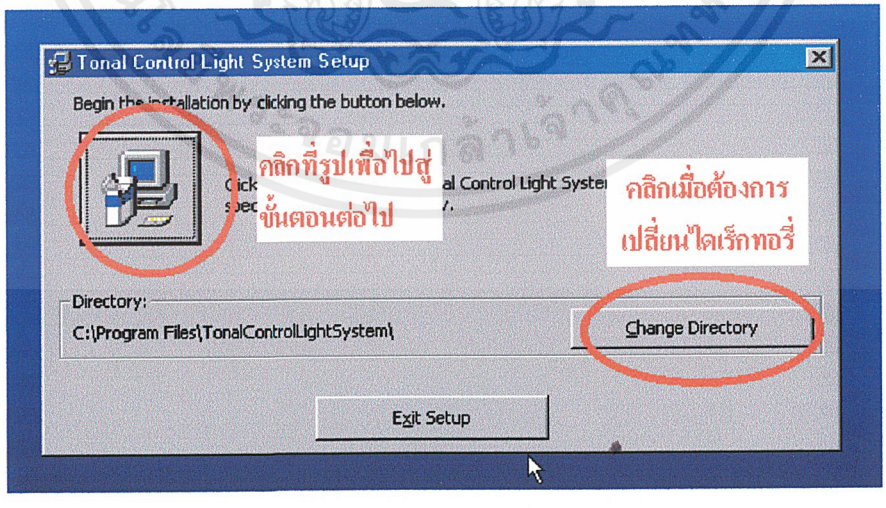
รูปที่ ก-2 แสดงการเริ่มต้นของการติดตั้งโปรแกรม

4. เลือก OK ที่หน้าต่างแรกดังรูป ก-3



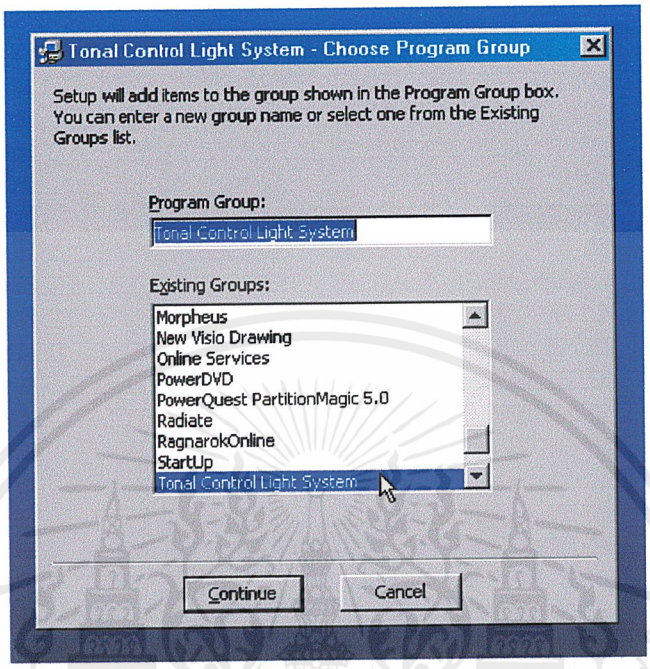
รูปที่ ก-3 แสดงหน้าต่างแรกของโปรแกรมติดตั้ง

5. ปรากฏหน้าต่าง Tonal Control Light System Setup ดังรูป ก-4 คลิกที่ Change Directory ถ้าต้องการเปลี่ยนพาทที่จะติดตั้งโปรแกรม จากนั้นคลิกที่ปุ่มรูปเครื่องคอมพิวเตอร์เพื่อไปสู่ขั้นตอนต่อไป



รูปที่ ก-4 แสดงหน้าต่าง Tonal Control Light System Setup

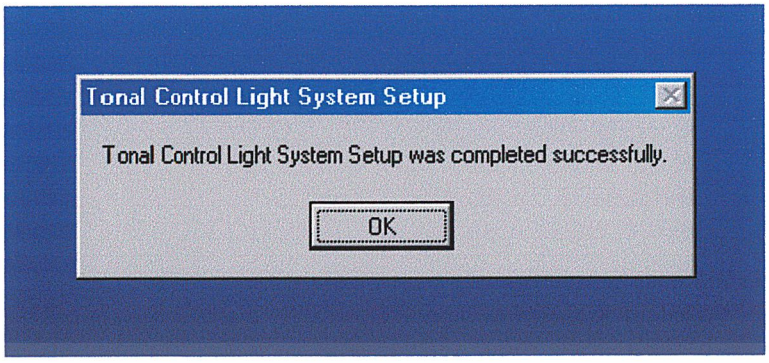
6. ต่อมาที่หน้าต่าง Choose Program Group ทำการเลือกกลุ่มของโปรแกรม พิมพ์เพื่อเปลี่ยนชื่อกลุ่มใหม่ หรือเลือกจากกลุ่มที่มีอยู่แล้ว แนะนำให้ใช้ชื่อที่กำหนดให้ คลิก Continue เพื่อไปสู่ขั้นตอนต่อไป



รูปที่ ก-5 แสดงหน้าต่างการเลือกกลุ่มโปรแกรม

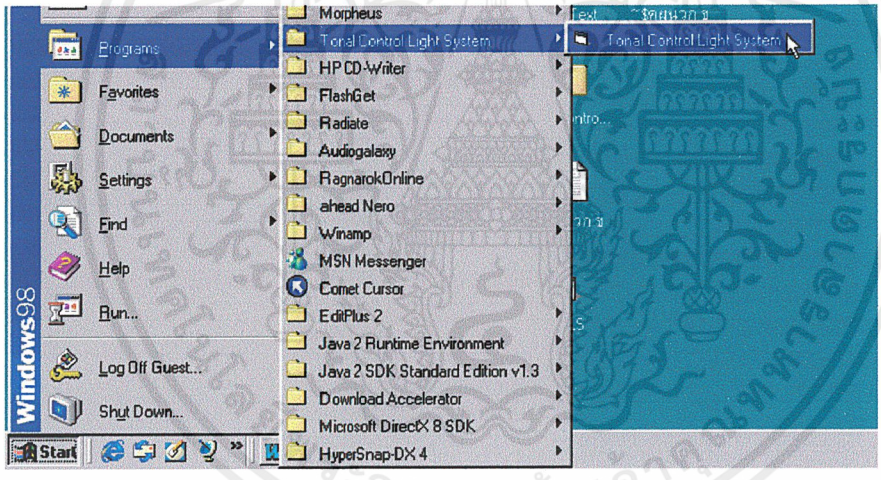
7. โปรแกรมติดตั้งจะเริ่มการโอนถ่ายข้อมูลดังรูป ก-6 ถ้าไม่มีสิ่งผิดปกติเมื่อเสร็จสิ้นจะปรากฏหน้าต่างแจ้งว่าการติดตั้งสมบูรณ์แล้ว ดังรูป ก-7 คลิก OK

รูปที่ ก-6 แสดงสภาพขณะโอนถ่ายข้อมูลต่างๆของโปรแกรม



รูปที่ ก-7 แสดงหน้าต่างแจ้งการติดตั้งโปรแกรมเสร็จสมบูรณ์

8. จบขั้นตอนของการติดตั้งโปรแกรม Tonal Control Light System สามารถเรียกใช้โปรแกรมได้ที่ Start > Program > Tonal Control Light System > Tonal Control Light System ดังรูป ก-8



รูปที่ ก-8 แสดงการเรียกใช้โปรแกรมจาก Start Menu

<u>หมายเหตุ</u> – ในกรณีที่ในซิสเต็มพาทของวินโดว์ ไม่มีไฟล์ WinMM.dll กับ Kernel32.dll ต้องทำการคัดลอกไฟล์ดังกล่าว จากพาทที่ทำการติดตั้งโปรแกรม ไปไว้ในซิสเต็มพาทของวินโดว์ก่อน โปรแกรมจึงจะทำงานได้

ภาคผนวก ข.
การใช้คำสั่งต่างๆ

เนื่องจากลิบรารีที่ใช้ในระบบมีฟังก์ชันการทำงานจำนวนมาก และส่วนใหญ่ไม่ได้นำมาใช้ในระบบ จึงจะกล่าวถึงเฉพาะในส่วนที่มีการใช้งานในระบบเท่านั้น โดยในที่นี้จะอ้างอิงคำสั่งต่างๆ ที่ใช้ใน Visual Basic 6.0 เป็นหลัก

คำสั่งการใช้งาน DirectX 8.0

ต้องมีไฟล์ DX8V8.dll อยู่ภายใน System path ของวินโดว์

ตารางที่ ข-1 แสดงเมธทอดคำสั่งที่เกี่ยวกับ DirectX 8.0

ออบเจก	เมธทอด	อธิบาย
DirectX8	DirectSoundCaptureCreate	สร้างออบเจก DirectSoundCapture8
	DirectSoundCreate	ทำการสร้างออบเจก DirectSound8
DSC	CreateCaptureBuffer	สร้าง DirectSoundCaptureBuffer8
DS	CreateBufferFromFile	สร้าง DirectSoundSecondaryBuffer จากไฟล์
	CreateSoundBufferFromFile (*filepath*, *DSbufDesc*)	สร้าง SoundBuffer สำหรับไฟล์จากตัวแปร *filepath* ให้กับ DSB โดย DS ดังลักษณะที่กำหนดในตัวแปร *DsbufDesc*
DSCB	Start	เริ่มบันทึกเสียง
	Stop	หยุดการบันทึกเสียง
	ReadBuffer	อ่านข้อมูลจากบัฟเฟอร์ลงอาร์เรย์
DSB	Play	เริ่มเล่นไฟล์เสียงใน DSB
	Stop	หยุดเล่นไฟล์เสียงใน DSB

* DSC คือ DirectSoundCapture8

* DS คือ DirectSound8

* DSCB คือ DirectSoundCaptureBuffer8

* DSB คือ DirectSoundSecondaryBuffer8

รายละเอียดเกี่ยวกับคำสั่งหรือเมธทอดอื่นๆ สามารถดูได้เพิ่มเติมจาก DirectX Documentation (Visual Basic)

คำสั่งการใช้งาน mciSendString

ต้องมีไฟล์ WinMM.dll อยู่ภายใน System path ของวินโดว์ ซึ่งเป็นลิบราลีที่เก็บฟังก์ชัน mciSendString อยู่และมีรูปแบบการประกาศเรียกใช้ใน Visual Basic 6.0 ดังรูป

```
Private Declare Function mciSendString Lib "winmm.dll" Alias "mciSendStringA"
                        (ByVal lpstrCommand As String, _
                        ByVal lpstrReturnString As String, _
                        ByVal uReturnLength As Long, _
                        ByVal hwndCallback As Long) _
                        As Long
```

รูปที่ ข-1 แสดงการประกาศเรียกใช้ mciSendString

จากรูปที่ ข-1 สามารถอธิบายตัวแปรต่างๆ ได้ดังนี้

- lpstrCommand คือ ชุดของตัวอักษรซึ่งเป็นคำสั่งที่ต้องการให้ฟังก์ชัน mciSendString ทำงาน ซึ่งต้องเป็นชุดของตัวอักษรเฉพาะที่มีการกำหนดไว้เท่านั้น ในการเรียกใช้ไฟล์เสียง หรือไฟล์ข้อมูลใดๆ ก็ตามในฟังก์ชัน mciSendString จะต้องทำการเปิดไฟล์ก่อน และควรปิดไฟล์เพื่อเมื่อเลิกใช้งานไฟล์ดังกล่าว

- lpstrReturnString คือ ชุดของตัวอักษรที่ใช้รับค่าข้อมูล หรือรายละเอียดซึ่งจะถูกส่งกลับในกรณีที่ไม่ต้องการข้อมูลดังกล่าว อาจตั้งให้ค่านี้เป็นนัลได้

- uReturnLength คือ ขนาด ของชุดตัวอักษรของพารามิเตอร์ lpstrReturnString

- hwndCallback คือ แฮนเดิลที่วินโดว์ ใช้ส่งค่ากลับ

ตารางที่ ข-2 แสดงการส่งอากิวเมนท์ต่างๆ ของฟังก์ชัน mciSendString

คำสั่ง	การทำงาน
- MciSendString "Open " & *filepath* & " Alias MM", 0, 0, 0	เปิดไฟล์เสียง MP3 ที่แอดเดรส *filepath* เช่น "C:\song.mp3"
- MciSendString "Close MM", 0, 0, 0	ปิดไฟล์เสียงที่เปิดอยู่แล้ว
- MciSendString "Play MM", 0, 0, 0	เล่นไฟล์เสียงที่เปิดอยู่
- MciSendString "Pause MM", 0, 0, 0	หยุดไฟล์เสียงที่เปิดอยู่ชั่วคราว
- MciSendString "Stop MM", 0, 0, 0	หยุดไฟล์เสียงที่เปิดอยู่เพื่อเริ่มต้นใหม่

คำสั่งการใช้งาน Out

ต้องมีไฟล์ inpout.dll, inpout32.dll อยู่ภายใน System path ของวินโดว์ และมีรูปแบบการเรียกใช้งานดังรูปที่ ข-2

```
#If Win32 Then
        Public Declare Sub Out Lib "InpOut32.dll" Alias "Out32" _
                                (ByVal PortAddress As Integer, _
                                ByVal Value As Integer)
#Else
        Public Declare Sub Out Lib "InpOut.DLL" _
                                (ByVal Port%, _
                                ByVal Value%)
#End If
```

รูปที่ ข-2 แสดงการเรียกใช้งานฟังก์ชัน Out

จากรูปที่ ข-2 แสดงว่า ถ้าเป็น Win32 ต้องมีไฟล์ InpOut32.dll และถ้าไม่ใช่ Win32 หรือก็คือ Windows 95 แล้วต้องมีไฟล์ InpOut.dll หลังจากการประกาศเรียกใช้ลิบรารีดังกล่าวแล้ว จะสามารถเรียกใช้ฟังก์ชัน Out (PortAddress, Value) ส่งข้อมูลออกทางบอร์ดเชื่อมต่อพอร์ตขนาน หรือ P-Board ได้โดย PortAddress คือ ตำแหน่งแอดเดรสของพอร์ต และ Value คือค่าที่ต้องการส่งออกทางพอร์ตดังกล่าว ซึ่งค่าที่ส่งออกไปนี้จะแสดงผลออกทางบอร์ด LED มอนิเตอร์ 16 ช่อง หรือ EX-01 สามารถแสดงได้ดังตารางที่ ข-3

ตารางที่ ข-3 แสดงการส่งอากิวเมนท์ต่างๆ ของฟังก์ชัน Out

คำสั่ง	การทำงาน
- Out &H378, &H0	หลอด LED ดับหมดทุกดวง
- Out &H378, &H1	หลอด LED ติด 1 ดวง
- Out &H378, &H3	หลอด LED ติด 2 ดวง
- Out &H378, &H7	หลอด LED ติด 3 ดวง
- Out &H378, &HF	หลอด LED ติด 4 ดวง
- Out &H378, &H1F	หลอด LED ติด 5 ดวง
- Out &H378, &H3F	หลอด LED ติด 6 ดวง
- Out &H378, &H7F	หลอด LED ติด 7 ดวง
- Out &H378, &HFF	หลอด LED ติดหมดทุกดวง

หมายเหตุ - H378 คือ แอดเดรสของพอร์ตเดต้าของ LPT1

บรรณานุกรม

กิตติ ภักดีวัฒนะกุล และจำลอง ครูอุตสาหะ. 2543. **VisualBasic 6 ฉบับโปรแกรมเมอร์.** พิมพ์ครั้งที่ 7. กรุงเทพฯ : บริษัท เคทีพี คอมพ์ แอนด์ คอนซัลท์ จำกัด

คู่มือการเชื่อมต่อคอมพิวเตอร์กับอุปกรณ์ภายนอก : Innovative Experiment Co.,Ltd.

พงษ์ศักดิ์ วิสูตรกาญจนชัย และสมชาติ รุ่งเรืองสรการ. 2524. **การแปลงฟูเรียร์อย่างเร็ว (The Fast Fourier Transform).** งานวิจัยของคณะครุศาสตร์อุตสาหกรรมและวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้า พระนครเหนือ

Gabriel Bouvigne. 2001. **MP3'Tech.** [Online]. Avaliable. http://www.mp3-tech.org

ISO 11172-3. **MPEG Audio.** [Online]. Avaliable. http://www.iso.ch

Microsoft Corporation 1995-2000. **Microsoft DirectX 8.0 (Visual Basic).** [CD-Rom] .

Microsoft Corporation 1991-2000. **MSDN LIBRALY.** [Online]. Avaliable. http://www.microsoft.com/msdn/

Scot Hacker. 2000. **MP3 The Definitive Guide** : O'REILLY