การเปลี่ยนระดับเสียงเป็นสัญญาณมิดี้ควบคุมโดยไมโครคอมพิวเตอร์ IBM PC

นายมาโนชญ์ กาญจนเพ็ญ

โครงงานนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรบัณฑิต

ภาควิชาฟิสิกส์ประยุกต์

คณะวิทยาศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2452

2542

PITCH TO MIDI CONVERTER CONTROL BY MICROCOMPUTER

Mr. Manorth Kanjanapen

A Special project , Submitted in Partial Fulfillment of the

Requirement , for the Degree of Bachelor of Science

Department of Applied Physics

Faculty of Science

King Mongkut's Institute of Technology Ladkrabang

1999

หัวข้อโครงงานพิเศษ	**การเปลี่ยนระดับเสียงเป็นสัญญาณมิดี้ ควบคุมโดยไมโครคอมพิวเตอร์ IBM PC**
โดย	**นายมาโนชญ์ กาญจนเพ็ญ**
ภาควิชา	**ฟิสิกส์ประยุกต์**
อาจารย์ที่ปรึกษา	**ดร. จิติ หนูแก้ว**
	ผศ.ดร. เสน่ห์ เอกะวิภาค

ภาควิชาฟิสิกส์ประยุกต์ คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อนุมัติให้โครงงานพิเศษฉบับนี้ เป็นส่วนหนึ่งของการศึกษา ตามหลักสูตรวิทยาศาสตร์บัณฑิต

____________________หัวหน้าภาควิชาฟิสิกส์ประยุกต์

(รศ. สุรพล รักวิจัย)

คณะกรรมการโครงงานพิเศษ

____________________กรรมการ

(ดร. จิติ หนูแก้ว)

____________________กรรมการ

(ผศ.ดร. เสน่ห์ เอกะวิภาค)

หัวข้อโครงงานพิเศษ	การเปลี่ยนระดับเสียงเป็นสัญญาณมิดี้ควบคุมโดย ไมโครคอมพิวเตอร์ IBM PC
นักศึกษา	นายมาโนชญ์ กาญจนเพ็ญ
อาจารย์ที่ปรึกษา	ดร. จิติ หนูแก้ว
	ผศ.ดร.เสน่ห์ เอกะวิภาต
ภาควิชา	ฟิสิกส์ประยุกต์
ปีการศึกษา	2542

บทคัดย่อ

โครงงานพิเศษนี้ เป็นการพัฒนาการเปลี่ยนระดับความถี่เสียงไปเป็นข้อมูลมิดี้ โดยใช้ความถี่ของรูปคลื่นสัญญาณเสียงที่อยู่ในย่านที่มนุษย์รับฟังได้ แล้วทำการเปลี่ยนให้เป็นข้อมูลตัวโน้ตทางมิดี้ ซึ่งในส่วนของการควบคุมการทำงานนั้นใช้ไมโครคอมพิวเตอร์ IBM PC และใช้โปรแกรมควบคุมที่เขียนด้วยภาษา Visual Basic สำหรับในส่วนของ มิดี้ และโปรแกรม Matlab เพื่อใช้ในการวิเคราะห์คลื่นเสียง

Special Project Title	Pitch to MIDI converter control by microcomputer IBM PC
Name	Mr. Manorth Kanjanapen
Special Project Advisor	Dr, Jiti Nukeaw
	Asst. Prof Dr. Sanay Akavipat
Department	Applied Physics , King Mongkut's Institude of Technology
Academic year	1999

Abstract

This special project is one of developed software in MIDI field - Musical Instrument Digital Interface. Allowed wave to midi data conversion, This software is applied for a signal at audio frequency (20 Hz–20 KHz) .The operation is used PC for controlling via portable keyboard ,also named as "Synthesizer" and cooperated with visual basic. Finally, MATLAB is utilized for audio - signal .

กิติกรรมประกาศ

โครงงานพิเศษนี้สามารถเสร็จสมบูรณ์ได้ความช่วยเหลือ และความอนุเคราะห์จากบุคคลต่าง ๆ ดังนี้

คุณพ่อคุณแม่	ผู้คอยให้กำลังใจ และให้ความอุปการะในทุก ๆ ด้านจนสำเร็จการศึกษา
ดร. จิติ หนูแก้ว	ผู้ที่ให้อิสรภาพในการค้นคว้าและทำโครงงานพิเศษนี้จนสำเร็จ
น้อง บี พระจันทร์สีน้ำเงิน	ผู้ที่ทำให้รู้สึกว่าทุกวันนี้ยังมีชีวิตอยู่และมีความสุขกับการมีชีวิต
คุณ แสงชัย สีประโคน	ผู้ที่ช่วยเหลือในการทำบุญสะเคาะเคราะห์
เพื่อนๆทุกคน	ที่คอยให้กำลังใจ และช่วยเหลือในทุกๆอย่าง
สมาคมมิดี้นานาชาติ	ผู้ก่อตั้งและกำหนดมาตรฐานของมิดี้
คณะกรรมการทุกท่าน	ที่ช่วยตรวจรายงานฉบับนี้
ภาควิชาฟิสิกส์ประยุกต์	ให้ยืมอุปกรณ์และเครื่องมือต่าง ๆ

มาโนชญ์ กาญจนเพ็ญ

สารบัญ

หน้า

หน้า

สารบัญรูป

หน้า

หน้า

บทที่ 1
บทนำ

ดนตรีไฟฟ้า หรือ Electronic Music นั้นเป็นศิลปะอีกแขนงหนึ่ง ซึ่งเป็นจุดเชื่อมประสานระหว่างศาสตร์แห่งศิลป์ กับ เทคโนโลยีทางด้านอิเล็กทรอนิกส์ และไมโครคอมพิวเตอร์ เราสามารถนำมันมาใช้ในการสร้างสรรค์งานดนตรีในแนวทางต่าง ๆ และนับวันจะยิ่งมีบทบาทมากขึ้น MIDI (Musical Instrument Digital Interface) เป็นส่วนหนึ่งของวิวัฒนาการทางด้านดนตรีไฟฟ้านี้ อันมีวัตถุประสงค์เพื่อให้เป็นมาตราฐาน ในการสื่อสารระหว่างเครื่องดนตรีไฟฟ้าทั่วโลก MIDI สามารถช่วยลดขั้นตอนในการทำงานทางด้านดนตรีให้สั้น และยังช่วยลดค่าใช้จ่ายได้มากอีกด้วย อีกทั้งยังช่วยเพิ่มคุณภาพของเสียงดนครี ซึ่งเป็นผลมาจากการเชื่อมโยงเสียงจาก เครื่องดนตรีไฟฟ้านานาชนิดเข้าด้วยกัน

Synthesizer เป็นเครื่องดนตรีไฟฟ้าประเภทคีย์บอร์ด ซึ่งมีความสามารถในการสังเคราะห์เสียงต่าง ๆ ได้มากมาย และเป็นเครื่องดนตรีที่มีความสัมพันธ์กับ MIDI มากที่สุด Synthesizer ทุกรุ่นในปัจจุบันจะติดตั้งระบบ MIDI ไว้เพื่อสามารถควบคุมหรือสื่อสารกับ Synthesizer ตัวอื่นๆ หรือแม้แต่กับเครื่องไมโครคอมพิวเตอร์ที่สามารถสื่อสารในระบบมิดี้ได้

ในช่วงหนึ่งของการพัฒนาทาง MIDI ได้มีความพยายามที่จะนำเสียงจากเครื่องดนตรีจริง เช่น กีตาร์ ทรัมเป็ต หรือแม้แต่เสียงร้องของมนุษย์มาใช้ในการควบคุม Synthesizer ให้เล่นตามท่วงทำนองที่เกิดจากเสียงของเครื่องดนตรีเหล่านั้น เพราะสามารถทำให้นักดนตรี หรือบุคคลที่มีจินตนาการทางด้านดนตรี เล่นหรือควบคุม Synthesizer สร้างสรรค์ทำนองหรือบทเพลงได้อย่างอิสระอันเนื่องมาจาก Synthesizer เป็นเครื่องดนตรีซึ่งมีความสามารถในการสร้างเสียงต่าง ๆ ไม่เพียงแต่เสียงที่มีอยู่จริงในธรรมชาติเท่านั้น แม้แต่เสียงที่เกิดจากจินตนาการก็สามารถสร้างขึ้นมาได้หากมีความรู้ เกี่ยวกับโครงสร้างของเสียงเพียงพอ จากคุณสมบัติพิเศษข้อนี้เองทำให้เราสามารถสร้างเสียงใดก็ได้จากการเล่นเครื่องดนตรีที่เราสามารถเล่นได้เครื่องเดียว โดยอาศัยความรู้ในการสื่อสารกับ Synthesizer ของ MIDI นั่นเอง มาถึงจุดนี้เราจำเป็นต้องอาศัยอุปกรณ์ที่มีความสามารถในการเปลี่ยนท่วงทำนองทางด้านดนตรี ซึ่งอยู่ในรูปของระดับความถี่เสียงของโน้ตทางดนตรีให้อยู่ในรูปของข้อมูลทาง MIDI โดยอาศัยหลักการที่ว่า " ระดับความถี่ของคลื่นที่มีคาบเวลาที่แน่นอน (periodic wave) ใด ๆ จะหาได้จากความถี่มูลฐานของคลื่นนั้น "

1.1 วัตถุประสงค์ในการจัดทำโครงงานพิเศษ

1. เพื่อศึกษาการควบคุมเครื่องดนตรีไฟฟ้าในระบบมิดี้ด้วยเสียงดนตรี
2. เพื่อศึกษาการประยุกต์ใช้งานไมโครคอมพิวเตอร์ทางด้านดนตรี
3. เพื่อนำความรู้ที่ได้รับจากการฝึกงานมาประยุกต์ใช้

1.2 ขั้นตอนการดำเนินงาน

1. ศึกษาการทำงานของระบบควบคุมและประมวลผลด้วยไมโครคอมพิวเตอร์
2. ศึกษาวิธีการแปลงระดับความถี่เสียงให้อยู่ในรูปสัญญาณควบคุม
3. ศึกษามาตราฐานและระบบของมิดี้
4. ทำการส่งระดับความถี่ที่รับเข้ามาไปเป็นข้อมูลในระบบของมิดี้โดยใช้โปรแกรม MATLAB และ Visual Basic มาทำการวิเคราะห์ ควบคุม และทำการประมวลผลโดยใช้ไมโครคอมพิวเตอร์

1.3 ประโยชน์ที่ได้รับจากการทำโครงงานพิเศษนี้

1. สามารถนำแนวความคิดทางฟิสิกส์และเทคโนโลยีทางไมโครคอมพิวเตอร์ มาพิสูจน์ให้เห็นถึงกระบวนการในการแปลง รูปของสัญญาณเสียง เพื่อพัฒนามาใช้สำหรับงานดนตรี
2. สามารถนำไปใช้ในงานแสดงดนตรีได้โดยตรง
3. สามารถนำไปใช้ร่วมกับซอฟแวร์ ทางดนตรี เพื่อช่วยในการประพันธ์เพลงได้
4. สามารถช่วยให้ผู้ที่ขาดทักษะการเล่นดนตรี เล่นหรือสร้างงานดนตรีดั่งใจปรารถนาได้
5. หากโครงงานนี้สำเร็จตามที่คาดหวังจะสามารถใช้ในการแกะเพลงที่มีโครงสร้างทางดนตรีที่มีความซับซ้อนไม่มากได้ เช่น ดนตรีที่บรรเลงโดย piano หรือ music box เป็นต้น

บทที่ 2
หลักการสื่อสารข้อมูลในระบบมิดี้

2.1 ความหมายของมิดี้

มิดี้หรือ MIDI ย่อมาจากคำว่า Musical Instrument Digital Interface เป็นระบบมาตราฐานของการสื่อสารข้อมูลระหว่างเครื่องดนตรีไฟฟ้า วัตถุประสงค์หลักของระบบมิดี้ก็คือ การทำให้เครื่องดนตรีไฟฟ้าที่ต่างรุ่นกัน หรือต่างผู้ผลิตสามารถทำงานร่วมกันได้ เพื่อให้เกิดระบบการสร้างดนตรีที่มีประสิทธิภาพ มิดี้เป็นระบบที่พัฒนามาจากแนวความคิดที่จะเชื่อมโยงการทำงานของเครื่องดนตรีไฟฟ้าประเภท Synthesizer ของ Dave Smith แห่งบริษัท Sequential circuit Incorporation ซึ่งภายหลังได้รับการปรับปรุงจนเป็นมาตราฐานภายใต้ชื่อ MIDI 1.0 โดยสมาคมมิดี้นานาชาติ (International MIDI Association หรือ IMA) ข้อกำหนดต่างๆ ของ MIDI 1.0 ได้รับการยอมรับจากบริษัทผู้ผลิตเครื่องดนตรีไฟฟ้าทั่วโลก และได้ถูกนำไปติดตั้งให้กับเครื่องดนตรีไฟฟ้าและเครื่องดนตรีต่าง ๆ มากมายจนทำให้เครื่องดนตรีเหล่านั้น สามารถสื่อสารกันได้ทางมิดี้ ดังนั้น มิดี้ ก็เปรียบเสมือนภาษาสากลของเครื่องดนตรีไฟฟ้านั่นเอง

สัญญาณทางมิดี้นั้นจะอยู่ในรูปของข้อความทางดิจิตอล ที่มีความหมายในเชิงควบคุมไมโครโปรเซสเซอร์ของเครื่องดนตรีไฟฟ้าที่มีระบบมิดี้สามารถตีความได้ ภายใต้ข้อกำหนดเหล่านี้จะทรรจุคำสั่งและข้อมูลที่สามารถทำให้เครื่องดนตรีไฟฟ้าเครื่องหนึ่งสามารถปฏิบัติงานตามข้อความที่ได้รับจากเครื่องดนตรีไฟฟ้าอีกเครื่องหนึ่งได้

2.2 รูปแบบการสื่อสารในระบบมิดี้

มิดี้ที่ใช้การสื่อสารแบบอะซิงโครนัสคล้ายกับมาตรฐาน RS-232 หนึ่งคำของมิดี้มีขนาด 10 บิต ประกอบไปด้วยบิตเริ่มต้น (start bit) 1 บิต , บิตข้อมูล 8 บิต และตามด้วยบิตหยุด (stop bit) อีก 1 บิต ดังแสดงดังรูป แต่ละบิตของคำทางมิดี้จะมีคาบเวลา 32 ไมโครวินาที โดยบิตเริ่มต้นจะมีลอจิกเป็น 0 ตามด้วยบิตข้อมูลซึ่งเรียงจากบิตที่มีนัยสำคัญต่ำสุด (LSB) ไปจนถึงบิตที่มีนัยสำคัญสูงสุด (MSB) จากนั้นก็จะเป็นบิตหยุด ซึ่งเป็นบิตหยุด ซึ่งมีลอจิกเป็น 1

Start Bit	Do	D1	D2	D3	D4	D5	D6	D7	Stop Bit

รูปที่ 2.1 รูปแบบของข้อมูลมิดี้มาตรฐานที่มีความเร็ว 31.25 kbps

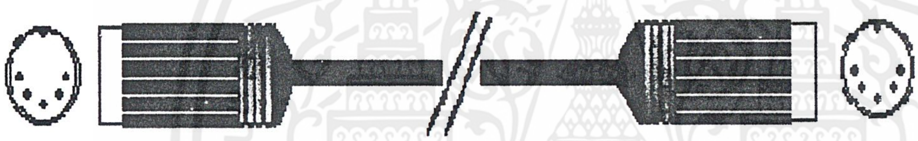
2.3 Cable, Port **และการต่อแบบ** Daisy Chain

2.3.1 Cable

- 5 Pin DIN Connector
- แต่ละ PIN มีหน้าที่ต่างกัน

โดย จะมี 2 PIN ที่ไม่ได้ใช้งาน ส่วนอีก 3 PIN จะทำหน้าที่ในการรับคำสั่ง MIDI , ground และใช้ในการป้อนไฟ 5 V และ 0 V เพื่อใช้ในการแสดงข้อมูล binary (on และ off)

- ความยาวต้องไม่เกิน 15 เมตร หรือ 50 ฟุต
- Opto- Isolator



รูปที่ 2.2 MIDI Cable

2.3.2 Port

- In
- Out
- Thru

2.3.3 **การต่อแบบ** Daisy Chain

- ต่อได้ไม่เกิน 3 Slave จาก 1 Master

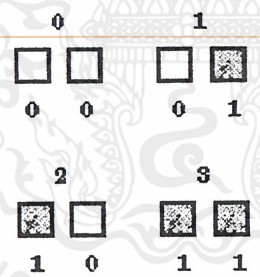
ก่อนอื่น ควรจะมีความเข้าใจเกี่ยวกับข้อมูลในระบบ ดิจิตอล ซึ่งจะได้แสดงถึงตัวเลขในระบบ decimal ,binary และ hexadecimal ซึ่งจะช่วยให้เกิดความเข้าใจในการสื่อสารของระบบ MIDI ได้ดียิ่งขึ้น

2.4 **ระบบ** decimal , binary & hexadacimal

เลขฐานสิบ หรือในระบบ decimal จะประกอบไปด้วยตัวเลขทั้งหมด 10 ตัวเลขคือ 0,1,2,3,4,5,6,7,8 และ 9 โดยตัวเลขในหลักต่อไปจะเป็นการรวมกันของตัวเลขสองหลัก ซึ่งในหลักที่สองจะคูณเข้าไปด้วย 10 โดย 10 จะมีค่า (1x10)+0 และจะมีค่าสูงสุดที่ 99 ส่วนในหลักที่สามจะทำการคูณด้วย 100 และหลักต่อไปก็จะทำการคูณด้วย 1000 , 10000 ไปเรื่อย ๆ ตัวอย่างเช่น 6594 จะคำนวณ ได้เป็น
(6x1000)+(5x100)+(9x10)+(4x1)

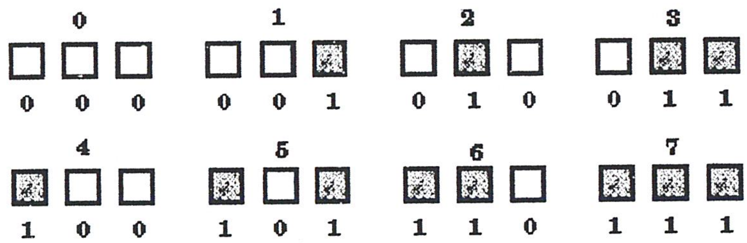
การทำงานของ computer ในระบบดิจิตอลจะมีตัวเลขเพียง 0 และ 1 ตัวเลขทุกค่าในระบบ decimal จะสามารถเขียนโดยตัวเลข 0 และ 1 ในระบบของ binary ได้ ถ้าหากมีเพียงหนึ่งหลัก ก็จะมีค่าที่เป็นไปได้สองค่า

ซึ่งถ้ามีสองหลักก็จะมีค่าความเป็นไปได้อยู่ทั้งหมดสี่ค่า



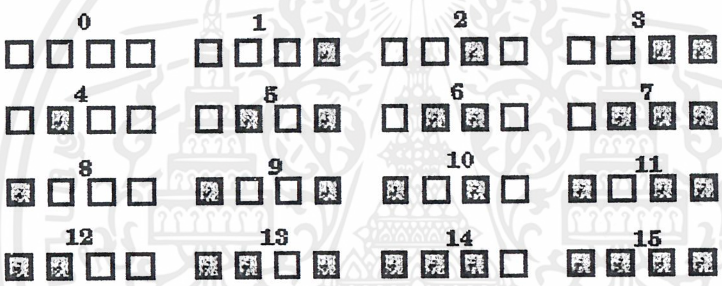
โดยตัวเลขที่อยู่ข้างบนจะแสดงเลขฐานสิบ และตัวเลขข้างล่างของแต่ละรูปแสดงการนับในระบบฐานสอง ในระบบฐานสองนั้น หลักแรก (แสดงโดยกล่องทางด้านขวามือ) ค่า ของฐานสิบที่แสดงออกมาจะเป็นค่า 0 หรือ 1 ในขณะที่หลักที่สอง (แสดงโดยกล่องทางด้านซ้ายมือ) ค่าของฐานสิบที่แสดงออกมาจะเป็นค่า 0 หรือ 2 การคูณของระบบตัวเลขฐานสิบจะเป็น 1, 10 , 100, 1000, ... ส่วนการคูณของระบบฐานสองนั้นจะเป็น 1 , 2, 4, 8 , 16

ถ้าหากมีค่าสามหลักหรือมีกล่องสามกล่อง จะมีความเป็นไปได้อยู่แปดค่า

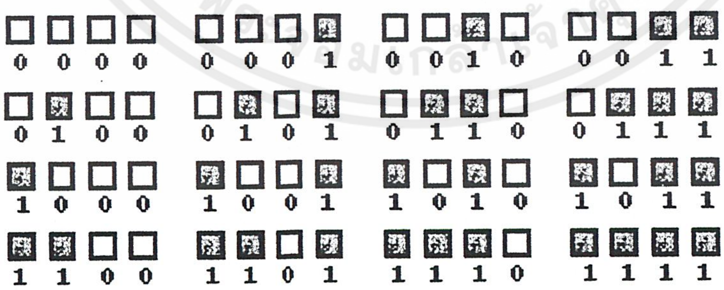


ค่าของหลักแรกจะแสดงค่า 0 หรือ 1 หลักที่สองจะแสดงค่า 2 หรือ 0 และหลักสุดท้ายจะมีค่า 4 หรือ 0

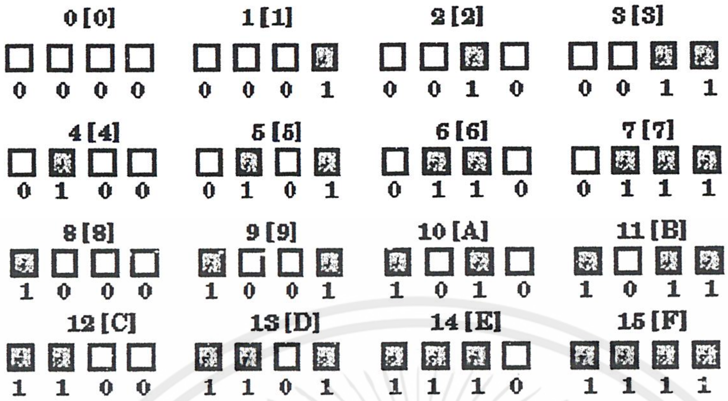
และถ้าหากมีสี่หลักหรือมีอยู่สี่กล่องจะมีค่าความเป็นไปได้อยู่สิบหกค่าด้วยกันดังนี้



รูปข้างล่างนี้จะเป็นรูปแบบเดียวกับรูปที่แสดงก่อนหน้านี้ แต่จะเป็นการแสดงค่าของตัวเลขระบบของฐานสอง



ท้ายสุดจะเป็นการแสดงความสัมพันธ์กันระหว่าง ระบบของเลขฐานสิบ เลขฐานสอง และฐานสิบหก ในรูปเดียวกัน



เลขฐานสิบหก จะมีตัวเลขที่แสดงอยู่สิบหกค่า คือ 0,1,2,3,4,5,6,7,8,9 เลข 10 จะมีอยู่สองหลักดังนั้นในฐานสิบหก จะเขียนแทนด้วยตัวอักษร A ตัวเลข 11 แทนด้วย B 12 แทนด้วย C 13 แทนด้วย D 14 แทนด้วย E และ 15 แทนด้วย F

ตัวเลข 16 ของเลขฐานสิบ จะมีค่าเป็น 10 ในระบบเลขฐานสิบหก (1x16) + 0 17 ในเลขฐานสิบ จะมีค่าเป็น 11 ในระบบเลขฐานสิบหก เลข 9C ของฐานสิบหกจะมีค่า (9x16) + 12 = 156 ในระบบเลขฐานสิบ

multiple of each digit	128	64	32	16	8	4	2	1
binary counting system	1	0	1	1	0	1	1	0

10110110 ของระบบฐานสองจะมีค่าเป็น 182 ของเลขฐานสิบ คือ (128x1) + (64x0) + (32x1) + (16x1) + (8x0) + (4x1) + (2x1) + (1x0)

multiple of each digit	256	16	1
hexadecimal counting system	1	A	F

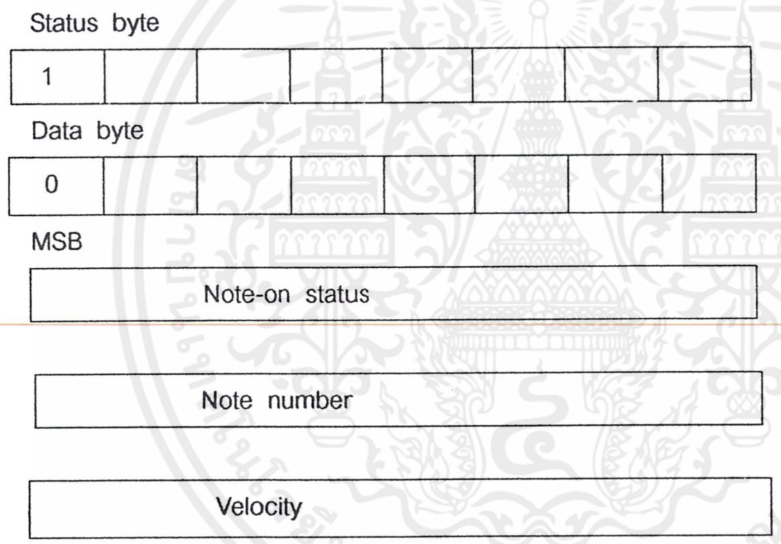
1AF ของเลขฐานสิบหก จะมีค่าเป็น 431 ของเลขฐานสิบ คือ (256x1) + (16x10) + (1x15)

จงจำไว้เสมอว่า ข้อมูลในระบบ MIDI นั้นจะมีการส่งโดยใช้ระบบเลขฐานสอง โดยเมื่อมีความเข้าใจในระบบของเลขฐานสอง ก็จะสามารถดูข้อมูลของ MIDI นั้น ๆ และมีความเข้าใจในการส่งข้อมูลไปยัง MIDI cable ได้อย่างถูกต้อง

10010000,00110111,01111011 = 90 , 37 , 7B โดยจะสามารถแปลข้อมูลออกมาได้เป็น note on , MIDI channel 1 , เล่นโน้ตตัวที่ 55 โดยมี velocity เป็น 123

2.5 รูปแบบของข้อมูล

สำหรับการสื่อสารด้วยข้อความทางมิดี้นั้น ในแต่ละข้อความจะประกอบไปด้วย (status byte) อันเป็นส่วนคำสั่งทางมิดี้ และตามด้วยไบท์ข้อมูล (data byte) ซึ่งใช้เป็นพารามิเตอร์ของคำสั่งนั้นอีก 1 หรือ 2 ไบท์ โดยจะแสดงตัวอย่าง ของรูปแบบดังกล่าวโดยไบท์ที่ 1 จะเป็นคำสั่งบอกให้เล่นโน้ต ส่วนไบท์ที่ 2 จะบอกว่าเล่นโน้ตตัวใด และสำหรับไบท์ที่ 3 จะบอกถึงความดังของตัวโน้ต ซึ่งประมวลผลจากความเร็วในการกดคีย์



รูปที่ 2.3 รูปแบบของข้อมูลในระบบมิดี้

การเล่นโน้ต (note on)

ข้อความจะถูกส่งออกไป เมื่อมีการกดคีย์บอร์ดของซินธิไซเซอร์ ประกอบไปด้วยข้อมูล 3 ไบท์ ได้แก่

ไบท์ที่ 1 เป็นรหัสคำสั่งและหมายเลขของช่องสัญญาณของโน้ตที่ถูกเล่น

ไบท์ที่ 2 เป็นหมายเลขของโน้ต หรือ รหัสมิดี้ของโน้ตที่ถูกเล่น มีค่าตั้งแต่ 0 ถึง 127 โดยที่ 0 จะเป็นโน้ตตัวต่ำสุด 127 จะเป็นโน้ตตัวที่สูงสุด และ 60 จะเป็นโน้ตตัวกลาง (middle c)

ไบท์ที่ 3 เป็นความเร็วในการกดคีย์ มีค่าตั้งแต่ 0 ถึง 127 โดย 0 หมายถึง ไม่มีการกดคีย์บอร์ด และ 127 จะหมายถึงการกดคีย์บอร์ดด้วยความเร็วสูงสุด สำหรับคีย์บอร์ดที่ไม่มีการตรวจจับความเร็วจะถูกกำหนดให้เป็น 64 ความเร็วในการกดคีย์ ซึ่งขึ้นอยู่กับการเปลี่ยนแปลงในการเล่น จะเป็นตัวกำหนดระดับความดังโดยรวมของเสียง

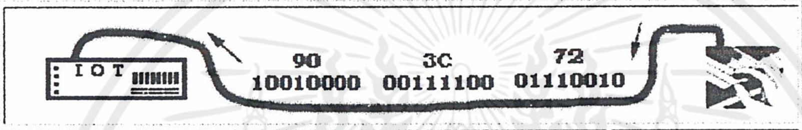
ดังนั้นถ้าต้องการจะให้เล่นโน้ตตัว middle c โดยมี full velocity จะทำโดย

status byte : 1001nnnn (nnnn represents a MIDI channel number)

data byte : 00111100 (binary code for 60 = middle c)

data byte : 01111111 (binary code for 127 = full velocity)

หรือ



Binary code	**10010000**	**00111100**	**01110010**
Hexadecimal	9 0	3 C	7 2

9 0	**Status Byte Note On ที่ MIDI channel 1.**
3 C	**Data Byte,** เป็นการสั่งให้ตัวโน้ตที่ **60** แสดงค่าโดยมีอัตราอยู่ในช่วง **0** ถึง **127 ,** ซึ่งก็คือค่าของ **Middle C.**
7 2	เป็นข้อมูลของ **key velocity** คือความเร็วของการกด โดยข้อมูลจะมีช่วง **0** ถึง**127** โดยเมื่อค่าของ **velocity** ค่าความดังของเสียงก็จะมีค่าเพิ่มขึ้นด้วย

2.6 ชนิดของข้อมูลที่ใช้ในการส่งผ่านไปเป็นสัญญาณ MIDI

จะมีชนิดของข้อมูลอยู่ 4 ชนิด คือ

- Channel Message
- System Exclusive
- System Common
- System Real -Time

<u>Channel Messages</u>

MIDI Message	Status Byte	Data Byte	Data Byte
Note Off	8n	Note Number	Velocity
Note On	9n	Note Number	Velocity
Polyphonic Aftertouch	An	Note Number	Pressure
Control Change	Bn	Control Number	Data Information
Control Change	Bn	01 Modulation Wheel	Data
Control Change	Bn	02 Breath Controller	Data
Control Change	Bn	04 Foot Controller	Data
Control Change	Bn	05 Portamento Time	Data
Control Change	Bn	06 Data Entry Slider	Data
Control Change	Bn	07 Main Volume	Data
Control Change	Bn	08 Balance Controller	Data
Control Change	Bn	0A Pan Controller	Data
Control Change	Bn	0B Expression Controller	Data

Control Change	Bn	40 Sustain Pedal (Damper)	00: Off 7F: On
Control Change	Bn	41 Portamento	00: Off 7F: On
Control Change	Bn	42 Sostenuto	00: Off 7F: On
Control Change	Bn	43 Soft Pedal	00: Off 7F: On
Control Change	Bn	45 Hold 2	00: Off 7F: On
Control Change	Bn	5B External Effects Depth	00: Off 7F: On
Control Change	Bn	5C Tremelo Depth	00: Off 7F: On
Control Change	Bn	5D Chorus Depth	00: Off 7F: On
Control Change	Bn	5E Celeste (Detune) Depth	00: Off 7F: On
Control Change	Bn	5F Phaser Depth	00: Off 7F: On
Control Change	Bn	79 Reset All Controllers	Data to reinitialize
Control Change	Bn	7A Local Control	00: Off 7F: On
Control Change	Bn	7B All Note Off	00
Control Change	Bn	7C Omni Off	00
Control Change	Bn	7D Omni On	00
Control Change	Bn	7E Mono On	00-0A Number of Channels
Control Change	Bn	7F Poly On	00
Program Change	Cn	Program Number 00 to 7F	No second data byte needed.
Channel Aftertouch	Dn	Note Number	Pressure Value
Pitch Wheel	En	Least Significant Bit	Most Significant Bit

System Exclusive

MIDI Message	Status Byte	Data Byte	Data Byte
System Exclusive	F0	Manufacture ID Code	Manufacture Data
System Exclusive Universal	F0	7E Sample Dump	Data on sample, bits, length, rate, and loop points
System Exclusive Universal	F0	7F MIDI Time Code	Data on Complete Message
System Exclusive Universal	F0	MIDI Machine Control	Data commands
System Exclusive Universal	F0	MIDI Show Control	Data commands
System Exclusive Universal	F0	Standard MIDI Song File	Data for Song File

System Common

MIDI Message	Status Byte	Data Byte	Data Byte
MIDI Time Code Quarter Frame	F1	Frame Number Data	No second data byte. F1 status byte is repeated with data byte.
Song Position Pointer	F2	Data on number of SSP 6 Bytes everyQuarterNote	Data on number of SSP 6 Bytes everyQuarterNote
Song Select	F3	Song Number	No second data byte.
Tune Request	F6	Data to change tuning	Data to change tuning
End of Exclusive	F7	No data bytes	No data bytes

System Real-Time			
MIDI Message	**Status Byte**	**Data Byte**	**Data Byte**
TimingClock MIDISync	F8	No data bytes	No data bytes. 24 Status bytes per quarter note.
Start	FA	No data bytes	No data bytes
Continue	FB	No data bytes	No data bytes
Stop	FC	No data bytes	No data bytes
Active Sensing	FE	No data bytes	No data bytes
System Reset	FF	Sets instrument to default	No data bytes

1. ข้อความควบคุมช่องสัญญาณ (Channel Message)

เครื่องดนตรีในระบบ MIDI สามารถติดต่อสื่อสารกับเครื่องดนตรีไฟฟ้า เครื่องดนตรีอื่น ๆ ได้หลายเครื่องพร้อมกัน ข้อความทาง MIDI จึงจะมีส่วนที่ใช้บอกว่าจะทำการติดต่อกับอุปกรณ์ตัวใด MIDI นั้นจะใช้ 4 บิตล่าง ของไบท์สถานะเป็น ตัวกำหนดช่องสัญญาณของอุปกรณ์ที่ต้องการติดต่อด้วย เพราะฉะนั้นจึงสามารถกำหนดได้ 16 ช่องสัญญาณ ข้อความนี้จะได้รับการตอบสนองเฉพาะไบท์สถานะที่มีช่องสัญญาณตรงกันกับช่องสัญญาณที่ด้านรับเท่านั้น ข้อความที่เกี่ยวข้องกับช่องสัญญาณมี 2 ประเภทคือ

1 . ข้อความควบคุมเสียง (Voice Message) ใช้ควบคุมเสียงของเครื่องดนตรีไฟฟ้า ข้อความเกี่ยวกับเสียงจะถูกส่งผ่านทางช่องสัญญาณเสียง (Voice Channel)

2 . ข้อความควบคุมโหมดการทำงาน (Mode Message) ใช้กำหนดลักษณะการตอบสนองของเครื่องดนตรี ต่อข้อความเกี่ยวกับเสียง หรือ เป็นการกำหนดโหมดการทำงานของระบบนั่นเองโดยจะส่งผ่านทางช่องสัญญาณพื้นฐาน (Basic Channel)

2 . System Exclusive มีหลายไบท์ข้อมูลและจะจบด้วยไบท์ที่เรียกว่า End of Exclusive (EOX) หรือไบท์สถานะอื่น ๆ เป็นข้อความที่ใช้เฉพาะกับอุปกรณ์ของผู้ผลิตนั้น ๆ ที่กำหนดไว้ในรหัสประจำเครื่อง (Identification Codes , ID codes) ของแต่ละบริษัทผู้ผลิตเท่านั้น ถ้าอุปกรณ์ทางด้านรับ มีรหัสประจำตัวเครื่องที่ไม่ตรงกันกับข้อความพิเศษที่ได้ มันจะไม่สนใจไบท์ข้อมูลที่ตามมา ผู้ผลิตแต่ละรายสามารถกำหนดรูปแบบความหมายข้อมูลของตนเองได้ โดยส่งเป็น ข้อความพิเศษที่มีรหัสประจำตัวเครื่องของตน (ซึ่งเป็นรหัสเดียวกันกับทางด้านรับ)

3 . System Common เป็นข้อความที่ใช้สำหรับอุปกรณ์ทุก ๆ ตัวในระบบ

4 . System Real – Time จะมีเฉพาะไบท์สถานะเท่านั้น ไม่มีไบท์ข้อมูล ข้อความนี้จะถูกส่งออกไปเมื่อไรก็ได้

2.7 General MIDI

เป็นมาตราฐานของ 128 เสียงเครื่องดนตรี นำมาจัดเป็นหมวดหมู่ โดยจัดแบ่งเป็น 16 ชนิดเครื่องดนตรี และในแต่ละกลุ่มก็จะจำแนกออกได้อีกเป็น 8 ชนิด ซึ่งชนิดเสียงของเครื่องประกอบจังหวะจะจัดไว้เฉพาะที่ Channel 10

Piano	**Bass**	**Reed**	**Synth F/X**
Chromatic Perc.	**Strings**	**Pipe**	**Ethnic**
Organ	**Ensemble**	**Synth Lead**	**Percussive**
Guitar	**Brass**	**Synth Pad**	**Sound F/X**

2.7.1 General MIDI Instrument List

Piano			
1.Acoustic Piano	2.BrtAcou Piano	3.ElecGrand Piano	4.Honky Tonk Piano
5.Elec.Piano 1	6.Elec.Piano 2	7.Harsichord	8.Clavichord
Chromatic Percussion			
9.Celesta	10.Glockenspiel	11.Music Box	12.Vibraphone
13.Marimba	14.Xylophone	15.Tubular Bells	16.Dulcimer
Organ			
17.Drawbar Organ	18.Perc. Organ	19.Rock Organ	20.Church Organ
21.Reed Organ	22.Accordian	23.Harmonica	24.Tango Accordian
Guitar			
25.Acoustic Guitar	26.SteelAcous. Guitar	27.El.Jazz Guitar	28.Electric Guitar
29.El. Muted Guitar	30.Overdriven Guitar	31.Distortion Guitar	32.Guitar Harmonic

Bass			
33.Acoustic Bass	34.El.Bass Finger	35.El.Bass Pick	36.Fretless Bass
37.Slap Bass 1	38.Slap Bass 2	39.Synth Bass 1	40.Synth Bass 2
Strings			
41.Violin	42. Viola	43.Cello	44.Contra Bass
45.Tremelo Strings	46.Pizz. Strings	47.Orch. Strings	48.Timpani
Ensemble			
49.String Ens.1	50.String Ens.2	51.Synth.Strings 1	52.Synth.Strings 2
53.Choir Aahs	54. Voice Oohs	55. Synth Voice	56.Orchestra Hit
Brass			
57.Trumpet	58.Trombone	59.Tuba	60.Muted Trumpet
61.French Horn	62.Brass Section	63.Synth Brass 1	64.Synth Brass 2
Reed			
65.Soprano Sax	66.Alto Sax	67.Tenor Sax	68.Baritone Sax
69. Oboe	70.English Horn	71.Bassoon	72.Clarinet

Pipe			
73.Piccolo	74.Flute	75.Recorder	76.Pan Flute
77.Blown Bottle	78.Shakuhachi	79.Whistle	80.Ocarina
Synth Lead			
81.Lead1 Square	82.Lead2 Sawtooth	83.Lead3 Calliope	84.Lead4 Chiff
85.Lead5 Charang	86.Lead6 Voice	87.Lead7 Fifths	88.Lead8 Bass Ld
Synth Pad			
89.Pad1 New Age	90.Pad2 Warm	91.Pad3 Polysynth	92.Pad4 Choir
93.Pad5 Bowed	94.Pad6 Metallic	95.Pad7 Halo	96.Pad8 Sweep
Synth F/X			
97.FX1 Rain	98.FX2 Soundtrack	99.FX3 Crystal	100.FX4 Atmosphere
101.FX5 Brightness	102.FX6 Goblins	103.FX7 Echoes	104.FX8 Sci-Fi
Ethnic			
105.Sitar	106.Banjo	107.Shamisen	108.Koto
109.Kalimba	110. Bagpipe	111. Fiddle	112. Shanai
Percussive			
113.TinkerBell	114.Agogo	115.SteelDrums	116.Woodblock
117.TaikoDrum	118.Melodic Tom	119.SynthDrum	120.Reverse Cymbal
Sound F/X			
121.Guitar Fret Noise	122. Breath Noise	123.Seashore	124.BirdTweet
125.Telephone	126.Helicopter	127.Applause	128.Gunshot

2.7.2 General MIDI Percussion Key Map

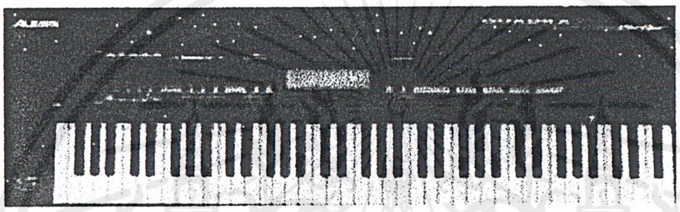
ที่ Channel 10 จะจัดไว้จำเพาะเสียงกลอง ซึ่งจะจำแนกได้ดังนี้

Key#	Drum Sound	Key#	Drum Sound
35	Acoustic Bass Drum	59	Ride Cymbal 2
36	Bass Drum 1	60	Hi Bongo
37	Side Stick	61	Low Bongo
38	Acoustic Snare	62	Mute Hi Conga
39	Hand Clap	63	Open Hi Conga
40	Electric Snare	64	Low Conga
41	Low Floor Tom	65	High Timbale
42	Closed Hi Hat	66	Low Timbale
43	High Floor Tom	67	High Agogo
44	Pedal Hi-Hat	68	Low Agogo
45	Low Tom	69	Cabasa
46	Open Hi-Hat	70	Maracas
47	Low-Mid Tom	71	Short Whistle
48	Hi Mid Tom	72	Long Whistle
49	Crash Cymbal 1	73	Short Guiro
50	High Tom	74	Long Guiro
51	Ride Cymbal 1	75	Claves
52	Chinese Cymbal	76	Hi Wood Block
53	Ride Bell	77	Low Wood Block
54	Tambourine	78	Mute Cuica
55	Splash Cymbal	79	Open Cuica
56	Cowbell	80	Mute Triangle
57	Crash Cymbal 2	81	Open Triangle
58	Vibraslap		

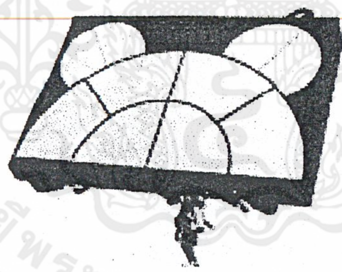
2.8 MIDI Controller

MIDI Controller จะเป็นตัวส่งผ่าน หรือสามารถปฏิบัติตามคำสั่งที่มีความเกี่ยวเนื่องกับ MIDI ได้ ซึ่งโดยส่วนใหญ่แล้วตัว Controller จะเป็น Keyboard เสียส่วนใหญ่ และโดยปกติแล้ว keyboard เหล่านี้ จะมี weighted key จะเป็นตัวที่ก่อให้เกิด ความ รู้สึก และมีการตอบสนองเพื่อให้เหมือนกันกับ Piano และ keyboard controller ส่วนใหญ่จะมี 76 หรือ 88 คีย์

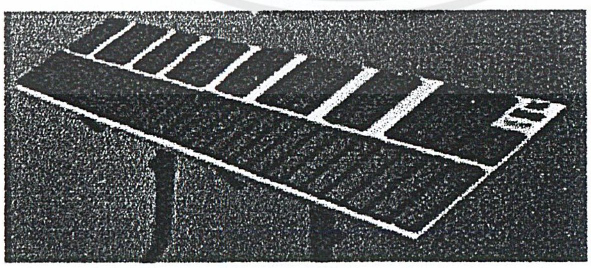
แต่นอกจาก keyboard แล้วก็ยังมี controller แบบอื่นอีก เช่น drumpads , keyboard mallets , wind controllers , guitar และ bass controllers



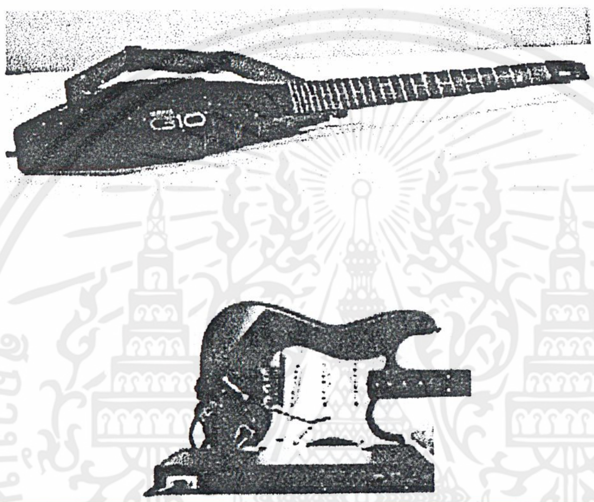
รูปที่ 2.4 **แสดง** Synthesizers **ที่มี** keyboard controllers **รวม อยู่ด้วย**
เสียงที่เกิดขึ้นจะเป็นเสียงที่เกิดจาก Synthesizers Modules ซึ่งจะไม่ปรากฎใน keyboard



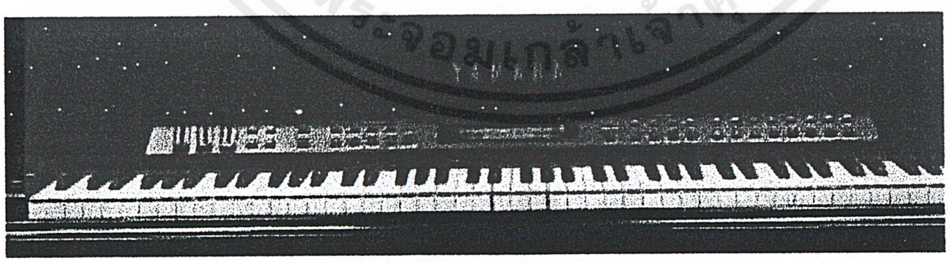
รูปที่ 2.5 Drum – Pad Controller



รูปที่ 2.6 MIDI Percussion Controller



รูปที่ 2.7 MIDI Guitar Controller



รูปที่ 2.8 Yamaha MIDI Grand

2.9 ข้อความควบคุมเสียงของช่องสัญญาณ (Channels Voice Message)

ข้อความเหล่านี้ จะประกอบไปด้วยข้อมูลข่าวสารที่ใช้ในการควบคุม Voice ของ synthesizer ให้ทำงานตามระบบควบคุมต่าง ๆ ที่ติดต่อกับผู้เล่น เพื่อให้ synthesizer นั้นสามารถเปล่งเสียงตามลักษณะของการเล่น ของผู้เล่น ซึ่งจะทำให้เสียงดนตรีที่ได้มีคุณลักษณะที่สมจริง ระบบควบคุมต่าง ๆ ได้แก่ แป้นคีย์บอร์ด , วงล้อควบคุมการสั่นของเสียง (Modulation Wheel) และระบบควบคุมการหายใจ (Breath controller) เป็นต้น ระบบต่าง ๆเหล่านี้ จะมีวงจรในการตรวจวัดปริมาณการเปลี่ยนแปลงทางกายภาพ เพื่อเปลี่ยนไปเป็นข้อมูลในการควบคุม ซึ่งใช้ประกอบเป็นข้อความสำหรับแต่ละช่องสัญญาณ ตามแต่จะกำหนด

2.9.1 การเล่นโน้ต (Note on)

ข้อความจะถูกส่งออกไป เมื่อมีการกดคีย์บอร์ดของ synthesizer ประกอบด้วยข้อมูล 3 ไบท์ ได้แก่

ไบท์ที่ 1 เป็นรหัสคำสั่งและหมายเลขช่องสัญญาณของโน้ตที่ถูกเล่น

ไบท์ที่ 2 เป็นหมายเลขของโน้ตหรือรหัส MIDI ของโน้ตที่ถูกเล่น มีค่าตั้งแต่ 0 ถึง 127 โดย 0 จะเป็นตัวโน้ตตัวที่ต่ำสุด 127 จะเป็นตัวโน้ตตัวที่สูงสุด และ 60 จะเป็นตัวโน้ต โด กลาง (middle C)

ไบท์ที่ 3 เป็นความเร็วในการกดคีย์ มีค่าตั้งแต่ 0 ถึง 127 โดย 0 จะหมายถึงไม่มีการกดคีย์บอร์ด และ 127 จะหมายถึงการกดด้วยความเร็วสูงสุด สำหรับคีย์บอร์ดที่ไม่มีการตรวจจับความเร็วจะถูกกำหนดเป็น 64 ความเร็วในการกดคีย์ ซึ่งขึ้นกับการเปลี่ยนแปลงในการเล่น จะเป็นตัวกำหนดระดับความดังโดยรวมของเสียง

1	0 0 1	Channel

0	Note number

0	velocity

รูปที่ 2.9 แสดงข้อความการเล่นโน้ต (Note on)

2.9.2 การหยุดโน้ต (Note off)

ข้อความที่ถูกส่งออกไป เมื่อเราทำการยกนิ้วขึ้นจากคีย์บอร์ดที่ถูกกดอยู่ประกอบด้วยข้อมูล 3 ไบท์ ได้แก่

ไบท์ที่ 1 เป็นรหัสคำสั่งและหมายเลขช่องสัญญาณของโน้ตที่จะหยุดเล่น

ไบท์ที่ 2 เป็นหมายเลยหรือรหัส MIDI ของโน้ตตัวที่จะหยุดเล่น

ไบท์ที่ 3 เป็นความเร็วในการปล่อยคีย์ มีลักษณะเดียวกับความเร็วในการกดคีย์ ข้อมูลนี้จะบอกได้ว่าเราปล่อยคีย์เร็วเพียงใด ซึ่งจะทำให้ช่วงเวลาของการปล่อยโน้ต ถูกควบคุมได้จากรูปแบบการเล่น

1	0 0 0	Channel

0	Note number

0	Release velocity

รูปที่ 2.10 แสดงข้อความการหยุดโน้ต (Note off)

2.9.3 **การกำหนดแรงกดคีย์แบบประสานเสียง** (Polyphonic Key Pressure)

นอกจากการตอบสนองจากความเร็วในการกดคีย์ MIDI ยังได้กำหนดให้มีการตรวจสอบแรงกดของคีย์บอร์ดอีกด้วย เมื่อคีย์บอร์ดถูกกด แรงของการกดคีย์ในแต่ละคีย์ที่ถูกกดจะทำให้ได้ข้อมูล 3 ไบท์ เป็นข้อมูลเฉพาะในแต่ละคีย์ อันประกอบด้วย

ไบท์ที่ 1 เป็นหมายเลขช่องสัญญาณที่ข้อมูลแรงกดคีย์นั้นจะถูกส่ง

ไบท์ที่ 2 เป็นหมายเลขของตัวโน้ตที่ทำให้เกิดแรงกดคีย์นั้น ซึ่งมีค่าอยู่ระหว่าง 0-127 โดยที่ 0 เป็นโน้ตตัวที่ต่ำที่สุด และ 127 เป็นโน้ตตัวที่สูงที่สุด ส่วน 60 จะเป็นโน้ต โด กลาง (middle C)

ไบท์ที่ 3 เป็นค่าแรงกดของคีย์ มีค่าอยู่ระหว่าง 0 –127 การตรวจสอบแรงกดแบบนี้เป็นแบบที่ต้องติดตั้งตัวตรวจวัดแรงกดกับทุก ๆ คีย์ของแผงคีย์บอร์ด จึงเป็นวิธีที่ใช้ต้นทุนสูง ดังนั้น synthesizer ส่วนใหญ่จะไม่ใช้วิธีนี้แต่จะใช้วิธีการตรวจสอบแรงกดโดยรวมของคีย์แทน ซึ่งเป็นหัวข้อที่เราจะกล่าวต่อไป

1	0 1 0	Channel

0	key number

0	Pressure value

รูปที่ 2.11 **แสดงข้อความกำหนดแรงกดคีย์แบบประสานเสียง** (Polyphinic Key Pressure)

2.9.4 **การกำหนดแรงกดรวม** (Overall Pressure **หรือ** Channel Aftertouch)

การตรวจสอบแรงกดด้วยวิธีนี้ เป็นการกำหนดแรงกดคีย์ของช่องสัญญาณ กล่าวคือ ใช้การเฉลี่ยค่าแรงกดคีย์ของคีย์ทั้งหมดที่ถูกกด ให้เป็นข้อมูลแรงกดคีย์ของช่องสัญญาณที่กำหนด ข้อความนี้ประกอบด้วยข้อมูล 2 ไบท์

ไบท์ที่ 1 เป็นรหัสคำสั่งและหมายเลขของช่องสัญญาณที่ข้อมูลแรงกดคีย์นั้นจะถูกส่ง

ไบท์ที่ 2 เป็นค่าแรงกดคีย์ของช่องสัญญาณ ซึ่งมีค่าระหว่าง 0 -127

การกำหนดแรงกดคีย์ทั้งสองแบบนี้ มักรู้จักกันในชื่อของ After – Touch อันเป็นหนึ่งในสอง**ของการตรวจสอบการสัมผัส** (Touch sensitivity) **ของคีย์บอร์ดซึ่งประกอบด้วยการตรวจสอบความเร็ว** (Velocity sensitivity) และการตรวจสอบแรงกด (Pressure sensitivity) ระบบดังกล่าวเป็นส่วนสำคัญอย่างยิ่ง ในการที่จะทำให้เครื่องดนตรีไฟฟ้า เล่นเสียงที่ให้อารมณ์และความรู้สึกเช่นเดียวกันกับเครื่องดนตรีจริง

1	1 0 1	Channel

0	Pressure value

รูปที่ 2.12 แสดงข้อความการกำหนดแรงรวม (Overall Pressure หรือ Channel After-Touch)

2.9.5 **การเปลี่ยนแปลงการควบคุม** (Control Change)

เป็นการส่งข้อมูลสำหรับตัวควบคุม (Control) ต่าง ๆ ในระบบ ข้อมูลเหล่านี้อาจใช้ในการควบคุมการเล่นหรือใช้สำหรับการแก้ไขเปลี่ยนแปลงข้อมูลใด ๆ ซึ่งขึ้นอยู่กับหน้าที่ของตัวควบคุมเหล่านั้น MIDI ได้กำหนดให้มีตัวควบคุมได้ถึง 128 ตัว (0-127) ข้อความนี้ประกอบด้วยข้อมูล 3 ไบท์ ได้แก่

ไบท์ที่ 1 เป็นรหัสคำสั่งและหมายเลขช่องสัญญาณที่ข้อมูลควบคุมถูกส่ง

ไบท์ที่ 2 เป็นหมายเลขของตัวควบคุม (controller number) มีค่าตั้งแต่ 0 ถึง 127

ไบท์ที่ 3 เป็นค่าข้อมูลของตัวควบคุมนั้น ๆ ตัวควบคุมหมายเลข 122-127 นั้นจะสงวนไว้ใช้สำหรับ Control Mode Message

1	0 1 1	Channel

0	Control number

0	value

รูปที่ 2.13 แสดงข้อความเปลี่ยนแปลงการควบคุม (Control Change)

2.9.6 **การเปลี่ยนแปลงโปรแกรมเสียง** (Program Change)

ในระบบ MIDI นั้น เราสามารถควบคุมการเปลี่ยนแปลงโปรแกรมของเสียง จากเสียงหนึ่งไปสู่อีกเสียงหนึ่งได้ โดยทำการส่งข้อความ ขอเปลี่ยนแปลงโปรแกรมไปยังเครื่องดนตรีตัวที่ต้องการ ข้อความการเปลี่ยนแปลงโปรแกรม จะประกอบไปด้วยข้อมูล 2 ไบท์ได้แก่

ไบท์ที่ 1 เป็นรหัสคำสั่งและหมายเลขช่องสัญญาณของอุปกรณ์ตัวที่ต้องการให้มีการเปลี่ยนแปลง

ไบท์ที่ 2 เป็นหมายเลขของโปรแกรมที่ถูกเลือก มีค่าตั้งแต่ 0 ถึง 127

1	1 0 0	Channel

0	Program number

รูปที่ 2.14 แสดงข้อความเปลี่ยนแปลงโปรแกรมเสียง (Program Change)

2.9.7 **การเปลี่ยนระดับความถี่เสียงอย่างต่อเนื่อง** (Pitch Bend)

เพื่อที่จะเลียนแบบลักษณะการเปลี่ยนแปลงเสียง ของตัวโน้ตบนเครื่องดนตรีจริงบางชนิด เช่น การเล่นโน้ต โดยวิธีการดันสายของกีตาร์ , การใช้คันโยกของทรอมโบน เป็นต้น เสียงของโน้ตดนตรีที่เกิดขึ้นจากวิธีการเหล่านี้ จะมีการเปลี่ยนแปลงอย่างต่อเนื่องจากตัวโน้ตหนึ่งไปสู่อีกตัวโน้ตหนึ่ง MIDI ได้กำหนดข้อความที่ใช้ตอบสนองการกำเนิดเสียงในลักษณะดังกล่าวไว้ ซึ่งประกอบด้วยข้อมูล 3 ไบท์ โดยที่

ไบท์ที่ 1 เป็นรหัสคำสั่งและหมายเลขช่องสัญญาณของข้อมูลควบคุมที่จะถูกส่ง

ไบท์ที่ 2 เป็นค่าการเปลี่ยนแปลงระดับความถี่เสียงอย่างต่อเนื่อง โดยที่ไบท์ที่ 2 เป็นไบท์ที่มีนัยสำคัญต่ำ

ไบท์ที่ 3 ส่วนไบท์ที่ 3 นั้นเป็นไบท์ที่มีนัยสำคัญสูง และโดยทั่วไปแล้วมักจะใช้เฉพาะไบท์ที่มีนัยสำคัญสูง จึงมีการเปลี่ยนได้จาก 0-127 โดยที่ค่า 64 จะหมายถึง ไม่มีการเปลี่ยนแปลง ดังนั้นค่าที่มากกว่าหรือน้อยกว่า 64 จึงมีผลทำให้โน้ตตัวที่เล่นอยู่นั้นมีเสียงสูงต่ำ ลงตามลำดับ ส่วนไบท์ที่ 2 นั้นจะถูกส่งด้วยค่า 0

ตามที่กล่าวมาเป็นการทำงานในแบบความละเอียดต่ำ (Low resolution) สำหรับการทำงานในแบบความละเอียดสูง (High resolution) นั้นจะใช้ไบท์ที่ 2 ด้วย แต่ไม่ค่อยได้รับความนิยมมากนัก โดยทั่วไปแล้ว pitch bend value นี้จะถูกควบคุมจากวงล้อ (Wheel) หรือ (Joystick)

1	1 1 0	Channel

0	Least significant byte

0	Most significant byte

รูปที่ 2.15 แสดงข้อความเปลี่ยนแปลงระดับเสียงอย่างต่อเนื่อง (Pitch Bend)

2.9.9 การหยุดเล่นโน้ตทุกตัว (All note off)

เป็นการหยุดเล่นโน้ตทุกตัวที่เล่นอยู่ในขณะนั้น ประกอบด้วยข้อมูล 3 ไบท์ ได้แก่

ไบท์ที่ 1 เป็นรหัสคำสั่งและหมายเลขช่องสัญญาณที่ข้อมูลการควยคุมจะถูกส่ง

ไบท์ที่ 2 เป็นหมายเลขตัวควบคุม ในที่นี้ คือ 123

ไบท์ที่ 3 เป็นไบท์ที่ไม่มีความหมายอะไร เป็นการใส่ข้อมูล 3 ไบท์ เท่านั้น เนื่องจากข้อความเกี่ยวกับการควบคุมมีขนาด 3 ไบท์

2.9.10 การเลือกโหมดทำงาน (Omni / Poly / Mono Select)

เราจะอธิบายถึงโหมดการทำงานเหล่านี้แล้ว ในการส่งข้อความเปลี่ยนแปลงโหมดนั้น จะเป็นการหยุดการเล่นตัวโน้ตทุกตัว (All Note off) ลง ข้อความนี้ประกอบด้วย

ไบท์ที่ 1 เป็นรหัสคำสั่งและหมายเลขช่องสัญญาณที่ข้อมูลการควบคุมถูกส่ง

ไบท์ที่ 2 เป็นหมายเลขของตัวควบคุมในแต่ละโหมด มีดังนี้

Omni Off	=	124
Omni On	=	125
Mono On	=	126
Poly On	=	127

ไบท์ที่ 3 สำหรับตัวควบคุมหมายเลข 123 , 124 , 125 , และ 127 ไบท์นี้จะเป็น 0 แต่ในกรณีของตัวควบคุมหมายเลข 126 (Mono) ไบท์นี้จะเป็นการกำหนดหมายเลขช่องสัญญาณที่จะให้เป็น Mono

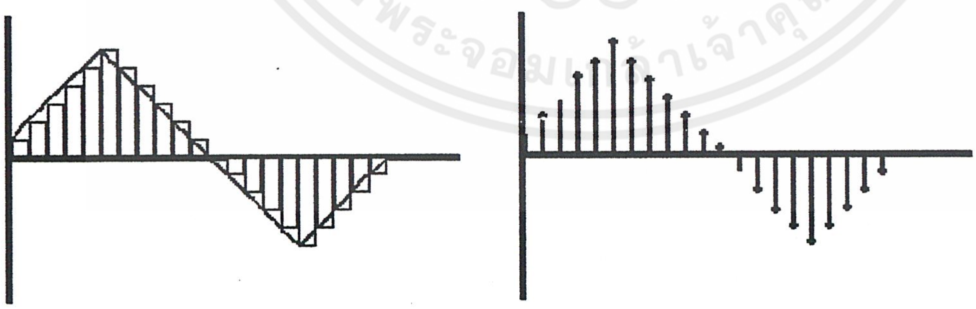
2.10 Audio vs. MIDI Flies

สัญญาณ Analog จะแปลงมาจาก Acoustic signal ซึ่งเกิดจากการเคลื่อนที่ของโมเลกุลในตัวกลางของอากาศ โดยสัญญาณ voltage จะเปลี่ยนขึ้นลงตามการสั่นของเส้นลวดไฟฟ้า โดย voltage ในเส้นลวดจะเป็น analog ซึ่งเป็นแบบเดียวกับการออกแบบให้เครื่องดนตรีเกิดเป็นเสียงต่าง ๆ พลังงานของเสียงจะเกิดขึ้นจากการกระทบกันของอนุภาคของอากาศโดยมีทั้งส่วนอัดและส่วนขยาย การเคลื่อนที่จะเกิดขึ้นอย่างรวดเร็ว ซึ่งจะมีค่าเท่ากันกับต้นกำเนิดของคลื่นเสียง เสียงสูงต่ำที่ผ่านเข้ามาจะถูกรับด้วยไมโครโฟน โดยมี diaphragm เป็นตัวตอบรับกับการเคลื่อนที่ของอากาศ ซึ่งที่จุดนี้เราจะใช้สิ่งที่เรียกว่า transducer เพราะที่จุดนี้พลังงานจะเปลี่ยนจาก acoustic signal ไปเป็นสัญญาณไฟฟ้า โดยจะแสดงออกมาใน waveform อันเดียวกัน โดยสัญญาณ voltage จะถูกส่งต่อไปยัง amplifier และไปยัง ลำโพง ซึ่งจะเปลี่ยนกลับไปเป็นสัญญาณ acoustic โดยรับจากระบบของ auditory

สัญญาณ audio จะถูกนำมาแสดงลงในหน่วยความจำของ digital ด้วยระบบเลขฐานสอง chip computer ADC (Analog to Digital Converter) จะถูกนำมาใช้ในการเปลี่ยนสัญญาณ analog ไปเป็นสัญญาณ.ในระบบ digital โดยในกระบวนการนี้เราเรียกว่าการ sampling ซึ่งเป็นการเปลี่ยนแปลงอันน่าทึ่งของโลกแห่งเสียงเลยทีเดียว

รูปที่ 2.16 รูปแบบสัญญาณ audio

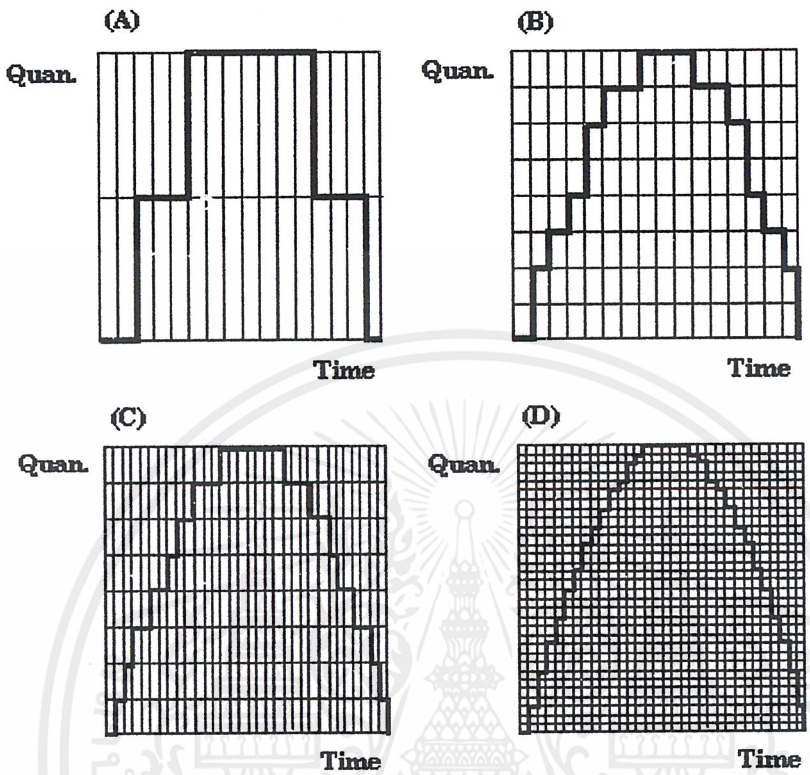
โดยมีรูปแบบของกระบวนการดังนี้



รูปที่ 2.17 กระบวนการ sampling rate

โดยถ้าหากมีการเปลี่ยน sampling rate คาบเวลาหรือช่องว่างระหว่าง sample ก็จะเปลี่ยนไป ดังตัวอย่างข้างล่าง

2.11 Different Quantization and Sampling Rates



รูปที่ 2.18 Sampling rate และ Quantization

รูป A และ B จะมีค่าของ sampling rate ที่เหมือนกัน แต่ในรูป B จะมีผลของการ quantization ที่ดีกว่า ส่วนในรูป C & D จะมีค่า sampling rate ที่มีค่าเป็นทวีคูณ และในรูป D จะมีค่าของ bit ที่เพิ่มขึ้น

บทที่ 3
แนวคิดและหลักการทำงาน

จากเนื้อหาที่ก่อนหน้านี้เราได้แนวคิดเบื้องต้น เกี่ยวกับพื้นฐานของ MIDI และการแปลงระดับความถี่เสียงไปเป็นข้อมูล MIDI หรือ pitch to MIDI convertor ไปบ้างแล้ว สำหรับในเนื้อหาต่อไปจะกล่าวถึงรายละเอียด ต่าง ๆ ที่ใช้ในการทำโครงงานพิเศษนี้ต่อไป

3.1 ระดับความถี่เสียงและการได้ยิน

เสียงที่ถูกรับโดยหูและถูกประมวลผลโดยระบบการฟัง ซึ่งเชื่อมต่อจากหูชั้นนอกจนถึงเยื่อหุ้มสมอง หูจะกระทำตัวเสมือนทรานสดิวเซอร์ (Transducer) ซึ่งแปลงการเคลื่อนที่เชิงกลของรูปแบบการสั่นไปเป็นการกระตุ้นทางประสาท โดยเมื่อเสียงเดินทางมาถึงหู มันจะเดินทางลงไปตามช่องภายนอก (External canal) ไปสู่แก้วหู ซึ่งจะส่งการสั่นนั้นไปสู่กระดูก 3 ชิ้นที่เรียงต่อกัน อันประกอบด้วย กระดูกรูปค้อน (Hammer) กระดูกรูปทั่ง (Anvil) และกระดูกรูปโกลน กระดูกรูปโกลนจะส่งการเคลื่อนไหวนั้นไปสู่เนื้อเยื่อที่เรียกว่า oval window ซึ่งเป็นเหตุให้คลื่นซึ่งกำลังเดินทางแพร่ไปในของไหลที่อยู่ในส่วนที่เรียกว่า Cochlea ให้ไปกระตุ้นขนที่อยู่บนเนื้อเยื่อที่เรียกว่า Basilar membrane เส้นประสาทที่เชื่อมต่อกับขนเหล่านี้ไปสู่สมอง เมื่อมีการสั่นแบบคลื่นซายน์ที่ความถี่ซึ่งอยู่ในช่วงที่สามารถรับฟังได้เข้ามาถึงหูชั้นใน มันจะทำการกระตุ้นปลายประสาทที่อยู่บน Basilar membrane เป็นสัดส่วนกับความถี่ของเสียงนั้น ความรู้สึกถึงความดังถูกกำหนดจากปริมาณพลังงานซึ่งรับได้โดยหู

ระดับความถี่เสียง เป็นพารามิเตอร์ที่ตอบสนองตามความถี่ มันเป็นความรู้สึกหลัก ซึ่งเป็นผลมาจากการกระตุ้นปลายประสาทบน Basilar membrane

ขอบเขตโดยประมาณของความถี่ที่มนุษย์สามารถรับฟังได้อยู่ระหว่าง 20 - 20,000 Hz ความถี่ตั้งแต่ 20 จนถึง 4000 Hz ประกอบด้วยย่านที่ไวต่อการได้ยินและไวต่อการเปลี่ยนแปลงความถี่ที่สุด ย่านความถี่สูงส่วนที่เหลือนั้นจะติดต่อเพียงหนึ่งในสาม การแยกแยะความถี่สำหรับคลื่นซายน์เป็นสัดส่วนที่เลวลงที่ความถี่ต่ำ และจะยิ่งยากขึ้น ถ้าช่วงเวลาที่ได้รับสั้นลง

ความสัมพันธ์ระหว่างความสามารถในการรับความถี่นั้นจะไม่เป็นเชิงเส้น การกระตุ้นบน basilar membrane จะปรากฏเป็นจุดที่มีลักษณะเกือบจะเป็นสัดส่วนกับล็อกการิทึ่ม ของความถี่ สำหรับความถี่สูงกว่า เมื่อความถี่เป็น 2 เท่า ระยะทางระหว่างจุดที่ถูกกระตุ้นบน Basilar membrane จะเปลี่ยนไปเท่ากับระยะทางคงที่โดยประมาณค่าหนึ่ง ดังนั้นผู้ฟังจึงสามารถเปรียบเทียบเสียงได้จากพื้นฐานของคาบทางดนตรี ซึ่งแบ่งแยกมันออกจากกัน นั้นคือใช้อัตราส่วนของความถี่ของมันเอง ตัวอย่างเช่น ช่วงของระดับความถี่เสียงที่ห่างกัน 1 ออกเตฟ จะมีอัตราส่วนของความถี่เป็น 2 : 1 เมื่อเราได้ยินมันในลักษณะความต่อเนื่องทางความถี่ มันก็จะให้ความรู้สึกที่เหมือนกัน

ความถี่ของโน้ต A3 (220 เฮิร์ต) นั่นเป็นครึ่งหนึ่งของความถี่ของโน้ตตัวที่สูงกว่ามัน 1 ออกเตฟ ซึ่งก็คือ A4 (440 เฮิร์ต) และความถี่ของ A4 ก็เป็นครึ่งหนึ่งของความถี่ของ A5 (880 เฮิร์ต) เช่นกัน ผลต่างของความถี่ระหว่างโน้ต 2 ตัวใดๆ ที่อยู่ติดกันจะไม่เท่ากัน นั่นหมายความว่าเราไม่สามารถหาความถี่ของโน้ตตัวหนึ่งจากความถี่ของโน้ตตัวที่อยู่ก่อนหน้ามัน ด้วยการบวกค่าคงที่ใด ๆ ได้ เนื่องจากค่าในหนึ่งออกเตฟความถี่ของโน้ตแต่ตัวจะเป็นอัตราส่วนกับความถี่ของโน้ตตัวแรกในบันไดเสียง (Unison)

ในทางด้านดนตรีนั้น บันไดโครมาติก (Chromatic scale) จะประกอบไปด้วยตัวโน้ตทั้งหมด 12 ตัว Temperament นั้นก็คือ ความถี่ที่ถูกต้องของโน้ตดนตรีทั้ง 12 ตัวนั่นเอง

3.2 ความถี่ และ ระดับเสียง

ในสมัยก่อนนั้น ตัวโน้ตดนตรีตัวเดียวกันจะมีค่าความถี่ไม่เหมือนกัน ใน เครื่องดนตรีที่ต่างกัน ตัวอย่างเช่น organ Silbermann ในโบสถ์ Strassburg ของประเทศเยอรมัน (1713) จะมีระดับเสียงของโน้ต a = 393 ในขณะที่ organ Schnitger ใน S.Jacobi ที่ Hamburg (1688) จะปรับเสียง a=489 ซึ่งมีค่าสูงกว่าประมาณสี่ semitone ในทางตอนเหนือของเยอรมัน Pretorius ให้เสียง a = 567 ซึ่งมีค่าสูงกว่าถึงหก semitone โดยที่เครื่องดนตรีทั่วไป จะใช้วัสดุที่ให้ระดับเสียงที่สูงกว่าเครื่องดนตรีที่ใช้ในทางศาสนา ดังนั้น ดนตรีที่เล่นในห้องโถง จึงมีระดับเสียงที่แตกต่างจากระดับเสียงดนตรีในโบสถ์

ส่วนในประเทศอังกฤษ organ ของวิทยาลัย Trinity ใน Cambridge ตั้งเสียง a = 395 ในขณะที่ Organ Berhardt Schmidt ใน โบสถ์ Durham และ ในห้องสวดของกษัตริย์ มีค่า a = 474 ซึ่งมีค่ามากกว่า organ Trinity ถึงสาม semitone

ในช่วงต้นของศตวรรษที่ สิบแปด ได้มีการทำให้เครื่องดนตรีต่างมีระดับเสียงที่เป็นรูปแบบเดียวกัน แต่ก็ยังคงอยู่ในช่าง 415 ถึง 430 สำหรับตัวโน้ต a และยังคงอยู่เช่นนั้นอีกประมาณเกือบร้อยปี จนกระทั่งเหล่านักดนตรีต้องการเสียงที่สดใสและมีโทนเสียงที่สูงขึ้น ก็ได้เริ่มมีการเพิ่มความถี่เสียงขึ้นอีก ในปี 1879 Covent Garden Orchestra ได้เล่นโน้ตตัว a ที่มีค่าเป็น 450 ในขณะเดียวกัน Concert Pitch ในอเมริกามีค่า a = 461.6

ในปี 1859 ทางประเทศฝรั่งเศษ ได้เสนอให้มีการกำหนดมาตรฐานของระดับตัวโน้ต a เป็น 435 แต่ทางด้านอเมริกาได้ใช้ a = 440 เป็นมาตรฐาน และในอังกฤษจะนิยมใช้ในการตั้งเสียงจากโน้ต C โดย c' มีค่า 522 ดังนั้น a จึงมีค่าเป็น 438.9

ในปี 1939 ได้มีการร่วมประชุมกันที่กรุงลอนดอน และได้มีความเห็นให้ใช้ a = 440 และให้เป็นมาตรฐานใช้กันทั่วโลก

เราสามาหาความห่างของตัวโน้ตจาก c ถึง c# ได้โดยการเพิ่มขึ้นของความถี่ที่มีค่าประมาณ 6 เปอร์เซ็นต์ และในทุก ๆ semitone ก็จะมีลักษณะเช่นเดียวกัน แต่จะใช้ค่า 6 เปอร์เซ็นต์ เลยทีเดียวก็ไม่ถูกต้องนัก เพราะการเพิ่มขึ้นของในหนึ่ง octave จะมีช่วงห่างระหว่างโน้ตอยู่ 12 ช่วง ดังนั้นจะมีค่าเท่ากับ 1.06 x 1.06 x 1.06 .... ทั้งหมดสิบสองครั้ง ซึ่งในทางคณิตศาสตร์จะเขียนได้เป็น $(1.06)^{12}$ และจะมีค่าเท่ากับ 2.0122 ซึ่งไม่ตรงกับค่า 2 ที่เราต้องการ

ดังนั้น ค่าของความถี่ที่เพิ่มขึ้นที่ถูกต้องจริง ๆ ควรเป็น 1.05946 โดยจะแสดงอัตราส่วนได้ดังนี้

กำหนดให้ c = 1

$c\# = 1.05946$

$d = (1.05946)^2 = 1.1225$

$d\# = (1.05946)^3 = 1.1892$

$e = (1.05946)^4 = 1.2599$

$f = (1.05946)^5 = 1.3348$

$f\# = (1.05946)^6 = 1.4142$

$g = (1.05946)^7 = 1.4983$

$g\# = (1.05946)^8 = 1.5874$

$a = (1.05946)^9 = 1.6818$

$a\# = (1.05946)^{10} = 1.7818$

$b = (1.05946)^{11} = 1.8877$

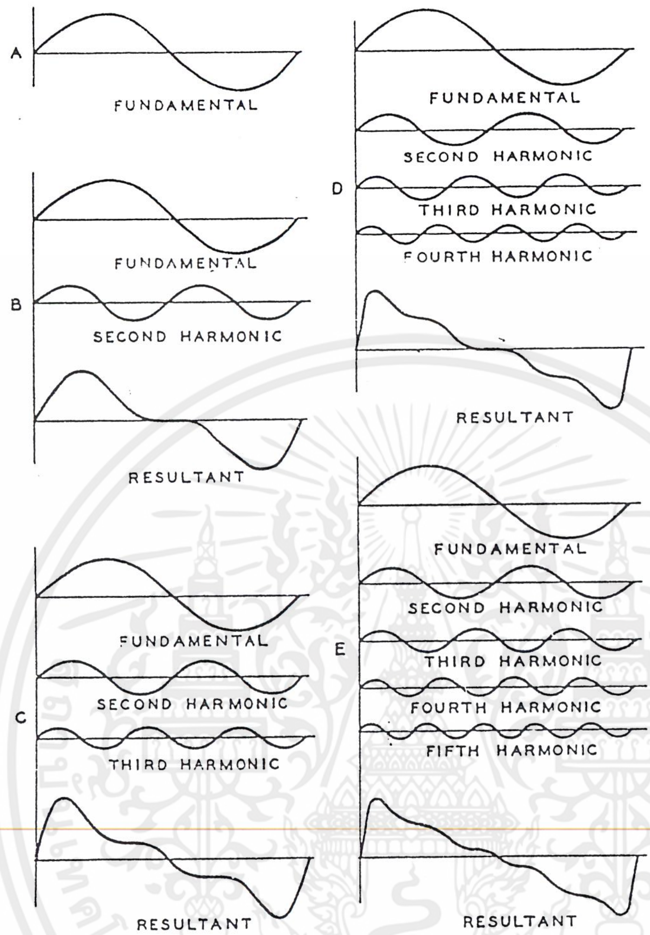
$c' = (1.05946)^{12} = 2.0000$

และโน้ตตัวต่อไปคือ c # ก็จะมีความถี่เท่ากับ $(1.05946)^{13}$ หรือมีค่าเท่ากับ 2×1.05946

การวิเคราะห์ทางฟูเรียร์และองค์ประกอบทางฟูเรียร์

เสียงที่เราได้ยินอยู่ในธรรมชาตินั้น ส่วนใหญ่เป็นคลื่นเชิงซ้อน (complex wave) ซึ่งประกอบไปด้วยคลื่นชายน์หลาย ๆ คลื่นรวมกันอยู่ คลื่นซายน์ที่มีความถี่เป็นค่าเริ่มต้นของคลื่นซายน์ทั้งหมด ที่มีความสัมพันธ์เชิงความถี่กันอยู่ เรียกว่า คลื่นมูลฐาน (Fundamental) ส่วนคลื่นซายน์อื่น ๆ นั้นเรียกว่า ฮาร์โมนิค (Harmonic) ฮาร์โมนิคเหล่านี้จะมีความถี่เป็นจำนวนเท่าทวีคูณกับความถี่มูลฐาน (Fundamental frequency) โดยใช้เลขจำนวนเต็มสำหรับบอกว่า ฮาร์โมนิค นั้นมีความถี่เป็นกี่เท่าของความถี่มูลฐาน ยกตัวอย่างเช่น คลื่นมูลฐานมีความถี่ 100 เฮิร์ต ฮาร์โมนิคที่สองจะมีความถี่เป็น 200 เฮิร์ต ฮาร์โมนิคที่สามก็จะมีความถี่เป็น 300 เฮิร์ต เป็นเช่นนี้เรื่อยไปตามลำดับ คลืนเชิงซ้อนแต่ละคลื่นจะมีจำนวนองค์ประกอบทางฮาร์โมนิคไม่เท่ากัน อันดับที่ขององค์ประกอบทางฮาร์โมนิคที่ประกอบขึ้นมาเป็นคลื่นเชิงซ้อนรูปหนึ่งขนาดของแอมปลิจูด และ เฟสของมันจะเป็นตัวกำหนดรูปคลื่น (waveform) ของคลื่นเชิงซ้อน

เพื่อความเข้าใจ เราขอยกตัวอย่างเพื่อให้ชัดเจนยิ่งขึ้น จากรูป 3.1 โดยรูปคลื่นมูลฐานจะแสดงได้ดังรูป A การรวมกันของคลื่นมูลฐานกับฮาร์โมนิคที่สอง หรือโอเวอร์โทนอันดับแรกแสดงได้ดังได้ดังรูป B รูปคลื่นที่เป็นการรวมกันของคลื่นมูลฐานกับฮาร์โมนิคที่สอง และสาม หรือ หรือโอเวอร์โทนที่หนึ่งและสองปสดงได้ดังรูป C รูปคลื่นที่เป็นการรวมกันของคลื่นมูลฐานกับฮาร์โมนิคที่ สอง สาม และ สี่ แสดงได้ดังรูป D รูปคลื่นที่เป็นการรวมกันของคลื่นมูลฐานกับฮาร์โมนิคที่ สอง สาม สี่ และ ห้า แสดงได้ดังรูป E และถ้าเราทำต่อไปโดยใช้ค่าต่าง ๆ เหล่านี้เป็นจำนวนมากประกอบกับค่าของแอมปลิจูด และ เฟส ที่เหมาะ สมผลลัพธ์ที่ได้ก็จะเป็นรูปฟันเลื่อย ที่สมบูรณ์มากยิ่งขึ้น



รูปที่ 3.1 คลื่นมูลฐาน และการรวมกันของคลื่นมูลฐานกับ ฮาร์โมนิคต่าง ๆ

รูป E ที่ได้แสดงไว้ข้างต้น เราสามารถแสดงเป็นสมการทางคณิตศาสตร์ได้เป็น

$$P_1 = \sin \omega t$$

$$P_2 = 1/2 \sin 2\omega t$$

$$P_3 = 1/3 \sin 3\omega t$$

$$P_4 = 1/4 \sin 4\omega t$$

$$P_5 = 1/5 \sin 5\omega t$$

โดยที่

P_1 = คลื่นมูลฐาน

P_2 = ฮาร์โมนิคที่สอง

P_3 = ฮาร์โมนิคที่สาม

P_5 = ฮาร์โมนิคที่สี่

P_6 = ฮาร์โมนิคที่ห้า

$\omega = 2\pi f$

f = ความถี่

t = เวลา

โดยที่ผลลัพธ์ของคลื่นสามารถเขียนได้เป็น

$$P_R = P_1 + P_2 + P_3 + P_4 + P_5$$

$$P_R = \sin \omega t + 1/2 \sin 2\omega t + 1/3 \sin 3 \omega t + 1/4 \sin 4 \omega t + 1/5 \sin 5 \omega t$$

โครงสร้างของคลื่นเสียงที่เกิดจากเครื่องดนตรี สามารถแสดงให้เห็นได้ดังภาพที่ 3.2 ตัวอย่างเช่น รูปที่แสดงความถี่ที่เป็นฮาร์โมนิคที่สอง ซึ่งก็คือมีความถี่มูลฐานสองครั้ง ค่าแอมปลิจูดก็จะมีค่าเป็นครึ่งหนึ่ง ของอันแรก เช่นเดียวกันความถี่ของฮาร์โมนิคที่สาม ซึ่งก็คือมีความถี่มูลฐานสามครั้ง จะมีค่าแอมปลิจูดลดลงมาเหลือ หนึ่งส่วนสามของแอมปลิจูดจากความถี่มูลฐาน

การรวมกันของคลื่นซายน์ในสามคู่อันดับแรกที่เป็นจำนวนคี่ซึ่งแสดงได้ดังรูป B นั้นสามารถเขียนได้ดังนี้

$$P_1 = \sin \omega t$$

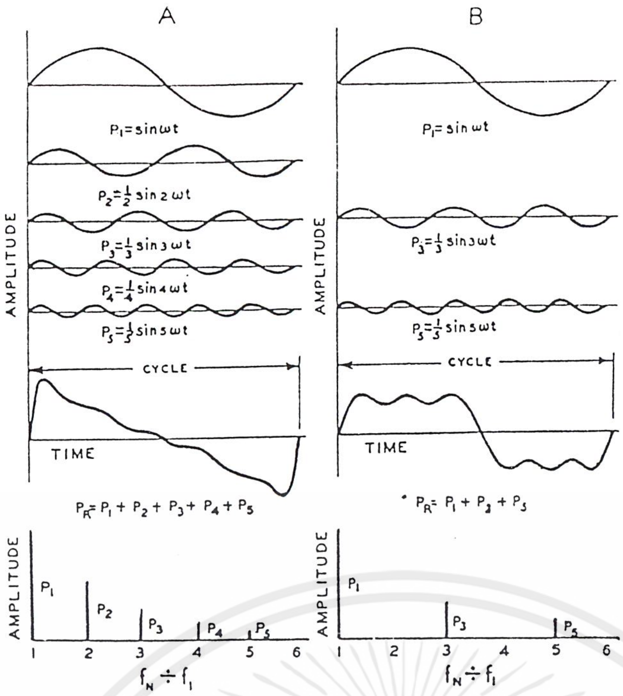
$$P_3 = 1/3 \sin 3\omega t$$

$$P_5 = 1/5 \sin 5\omega t$$

โดย $P_R = P_1 + P_3 + P_5$

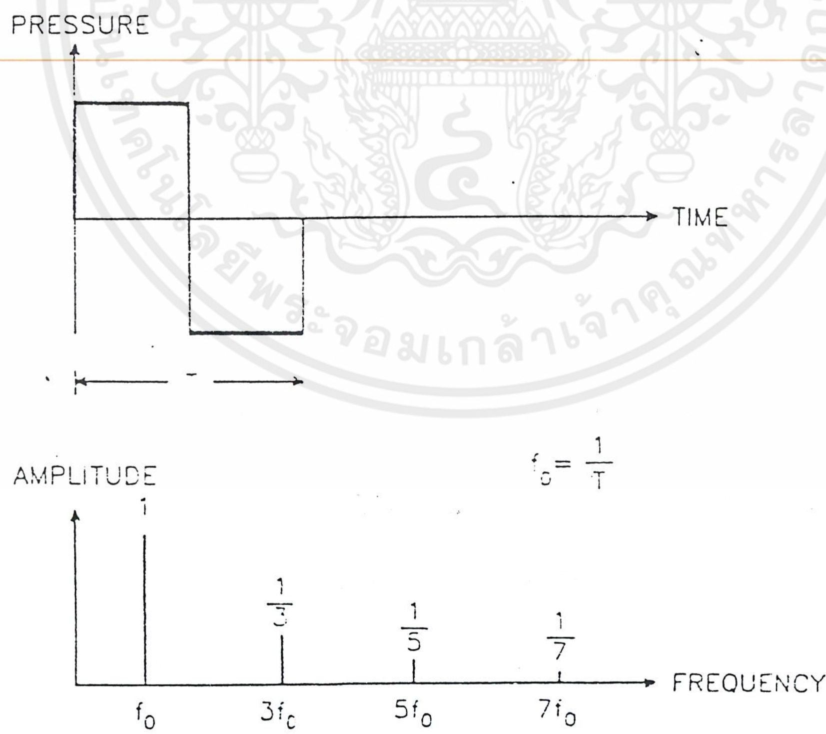
$$P_R = \sin \omega t + 1/3 \sin 3\omega t + 1/5 \sin 5\omega t$$

จะสังเกตได้ว่ารูป 3.2 A จะมีลักษณะเป็นรูปฟันเลื่อย ส่วนรูป 3.2 B จะเป็นแบบสี่เหลี่ยม

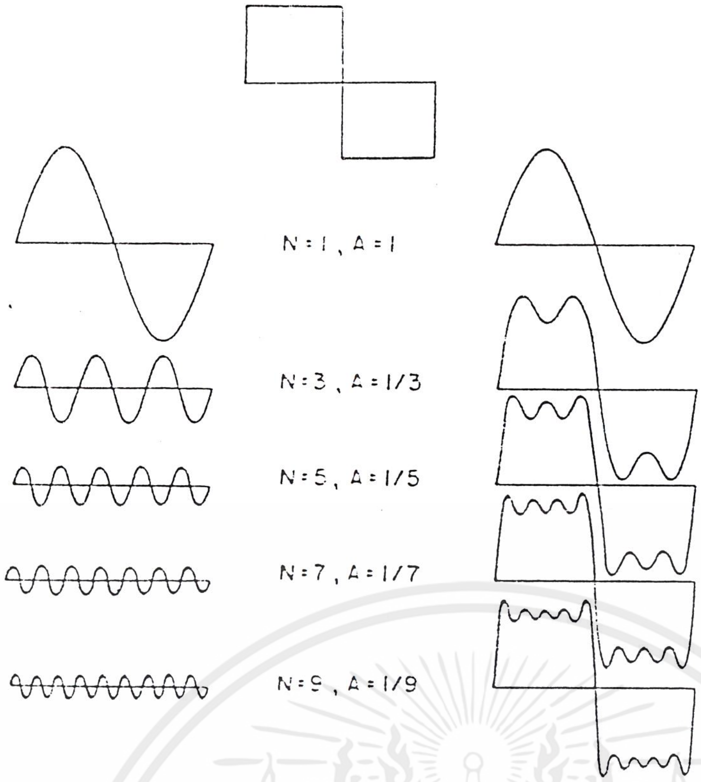


รูปที่ 3.2 คลื่นรวมที่มีลักษณะเป็นฟันเลื่อยและแบบสี่เหลี่ยม

เราสามารถสังเคราะห์คลื่นสี่เหลี่ยมจัตุรัสขึ้นได้จากคลื่นซายน์ความถี่หนึ่ง โดยการเพิ่มองค์ประกอบทางฮาร์โมนิคที่มีอันดับเลขคี่ ซึ่งมีขนาดแอมปลิจูดและเฟสของแต่ละฮาร์โมนิคเหมาะสมตามเงื่อนไขข้างต้นเข้าไป ดังแสดงในรูป 3.3 จะเห็นได้ว่า ยิ่งจำนวนขององค์ประกอบทางฮาร์โมนิคยิ่งเพิ่มขึ้น คลื่นเชิงซ้อนที่ได้ยิ่งใกล้เคียงกับคลื่นสี่เหลี่ยมจัตุรัสมากขึ้น



รูปที่ 3.3 คลื่นสี่เหลี่ยมจัตุรัส และองค์ประกอบทางฮาร์โมนิค



รูปที่3.4 แสดงการสังเคราะห์ทางฟูเรียร์ของคลื่นสี่เหลี่ยมจัตุรัส

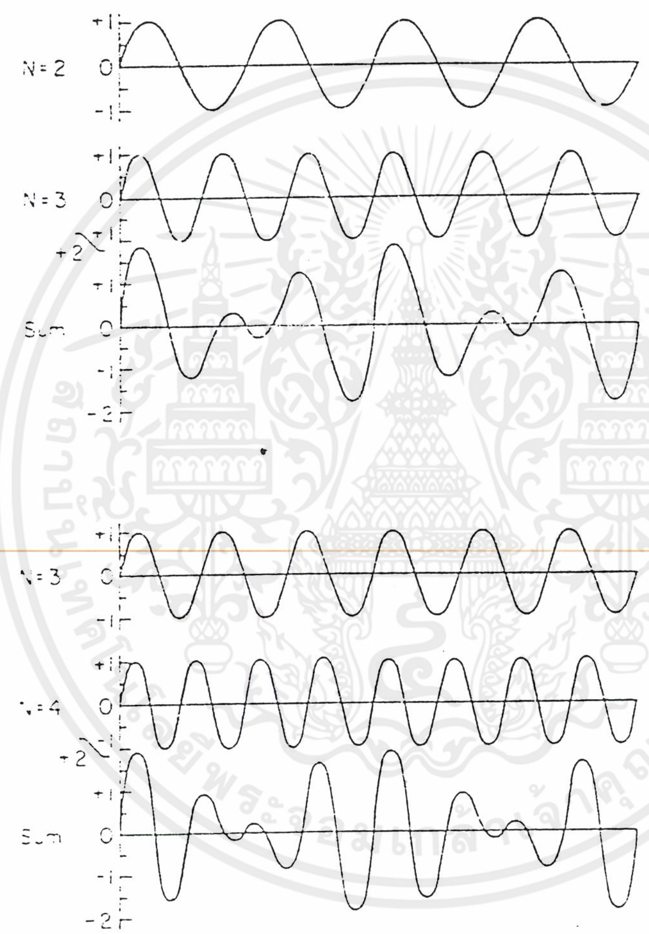
อันที่จริง ในเสียงจริง ๆ ตามธรรมชาติ เราอาจจะไม่พบคลื่นจัตุรุสที่มีลักษณะสมบูรณ์ดังรูป คลื่นรูปสี่เหลี่ยมที่ใช้นั้นเป็นส่วนหนึ่งในจำนวนคลื่นเชิงซ้อนมาตรฐาน (Standard complex wave) ที่ใช้อธิบายการวิเคราะห์ และสังเคราะห์คลื่นเชิงซ้อน คลื่นเชิงซ้อนมาตรฐานเหล่านี้ประกอบไปด้วย คลื่นซายน์ (sine) , คลื่นสามเหลี่ยม (Triangular) , คลื่นสี่เหลี่ยมจัตุรัส (square) , คลื่นฟันเลื่อย (Sawtooth) และพัลส์ต่อเนื่อง (Pulse train) ความสัมพันธ์ทางแอมปลิจูดของฮาร์โมนิคต่างๆ ของคลื่นเหล่านี้จะเป็นไป ตามตารางที่ 3.1

ตารางที่ 3.1 ความสัมพันธ์ทางแอมปลิจูดของฮาร์โมนิคของคลื่นมาตรฐาน

Wave	Harmonic amplitudes										
	$N =$ 1,	2,	3,	4,	5,	6,	7,	8,	9,	10	. . .
Sine	1,	0,	0,	0,	0,	0,	0,	0,	0,	0,	. . . , $N = 1$ only
Triangle	1,	0,	$\frac{1}{9}$,	0,	$\frac{1}{25}$,	0,	$\frac{1}{49}$,	0,	$\frac{1}{81}$,	0,	. . . , $\frac{1}{N^2}$ for odd N
Square	1,	0,	$\frac{1}{3}$,	0,	$\frac{1}{5}$,	0,	$\frac{1}{7}$,	0,	$\frac{1}{9}$,	0,	. . . , $\frac{1}{N}$ for odd N
Sawtooth (ramp)	1,	$\frac{1}{2}$,	$\frac{1}{3}$,	$\frac{1}{4}$,	$\frac{1}{5}$,	$\frac{1}{6}$,	$\frac{1}{7}$,	$\frac{1}{8}$,	$\frac{1}{9}$,	$\frac{1}{10}$,	. . . , $\frac{1}{N}$ for all N
Pulse train	1,	1,	1,	1,	1,	1,	1,	1,	1,	1,	. . . , equal for all N

จากที่กล่าวมาแล้วข้างต้น เราพอสรุปได้ว่า คลื่นที่มีคาบที่แน่นอนใด ๆ นั้นสามารถสังเคราะห์ขึ้นได้จากความถี่มูลฐานค่าหนึ่ง และฮาร์โมนิคที่มีแอมปลิจูดและเฟสที่เหมาะสม โดยที่ความถี่ของคลื่นเชิงซ้อนที่เกิดขึ้นนั้นก็คือ ความถี่มูลฐานของมันนั่นเอง

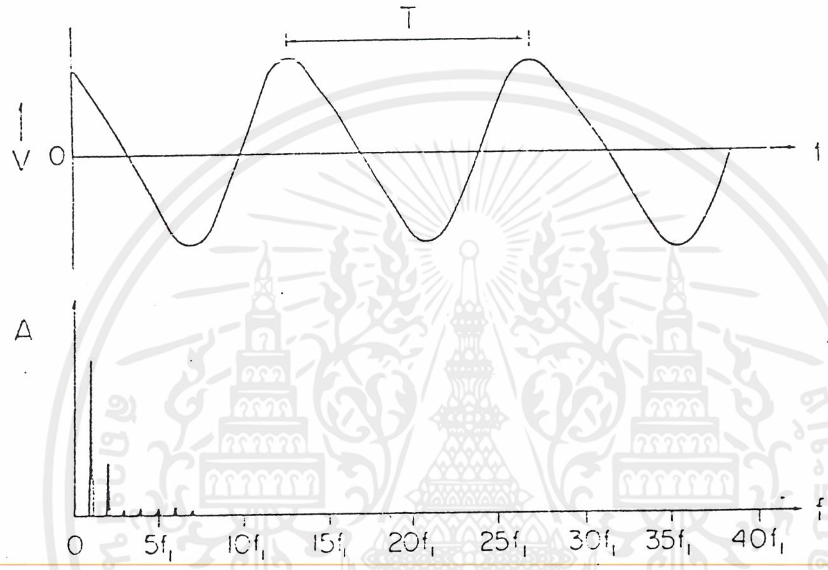
ลองพิจารณารูปที่ 3.5 บน จะเห็นคลื่นเชิงซ้อนซึ่งเกิดจากการรวมกันของฮาร์โมนิคอันดับที่ 2 และ 3 ส่วนรูปล่าง จะเป็นคลื่นเชิงซ้อนที่เกิดจากการรวมกันของฮาร์โมนิคอันดับที่ 3 และ 4 ของความถี่มูลฐานเดียวกับรูปบน เราเห็นรูปร่างของคลื่นเชิงซ้อนที่เกิดในทั้งสองกรณีจะมีลักษณะที่แตกต่างกัน แต่คาบของคลื่นเชิงซ้อนทั้งสองรูปนั้นจะเท่ากัน และเท่ากับคาบของคลื่นมูลฐานของมันทั้งสองด้วย นั่นแสดงว่าคลื่นเชิงซ้อนใดๆ จะมีความถี่เท่ากับความถี่มูลฐานของมัน ถึงแม้ว่าจะไม่มีคลื่นมูลฐานนั้นในโครงสร้างทางฮาร์โมนิคของมันก็ตาม



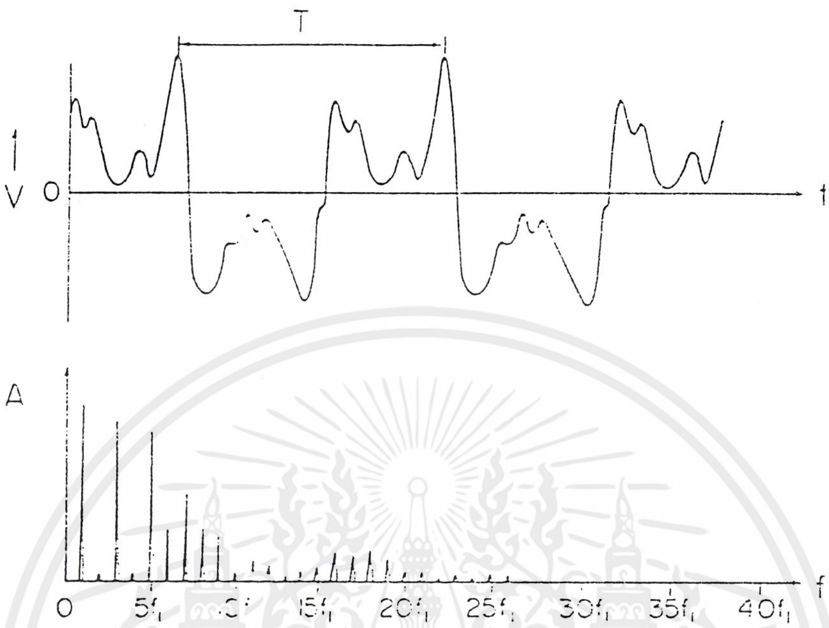
รูปที่ 3.5 การสังเคราะห์ทางฟูเรียร์ของคลื่นเชิงซ้อน 2 รูป ซึ่งได้จากการรวมกันของคลื่นฮาร์โมนิคที่มีความถี่มูลฐานเดียวกัน และแอมปลิจูดเท่ากันแต่ต่างอันดับกัน

เราสามารถวิเคราะห์หาองค์ประกอบทางฮาร์โมนิคของคลื่นเชิงซ้อนใดๆ ได้ด้วยการวิเคราะห์ทางฟูเรียร์ จากวิธีการดังกล่าวเราสามารถสรุปได้ว่าระดับความถี่เสียงที่เกิดจากคลื่นที่มีคาบเวลาที่แน่นอนใด ๆ สามารถหาได้จากความถี่มูลฐานของตัวมัน เอง ตัวอย่าง ขององค์ประกอบทางฮาร์โมนิคของเสียงจากเครื่องที่ได้จากการวิเคราะห์ฟูเรียร์ ได้แสดงไว้ในรูป 3.6 และ 3.7

จากข้อสรุปดังกล่าว ทำให้เราได้แนวความคิดในการวิเคราะห์หาระดับความถี่เสียงจากเครื่องดนตรี โดยการหาความถี่มูลฐานของเสียงนั้น



รูปที่3.6 รูปคลื่นและองค์ประกอบทางฮาร์โมนิคตของ โน้ต C ซึ่งมีความถี่ 523.25 เฮิร์ต ของเสียง alto recorder



รูปที่ 3.7 รูปคลื่นและองค์ประกอบทางฮาร์โมนิคของ โน้ต Bb ซึ่งมีความถี่ 493.88 เฮิร์ต ของเสียง violin

บทที่ 4
ระบบควบคุมและประมวลผล

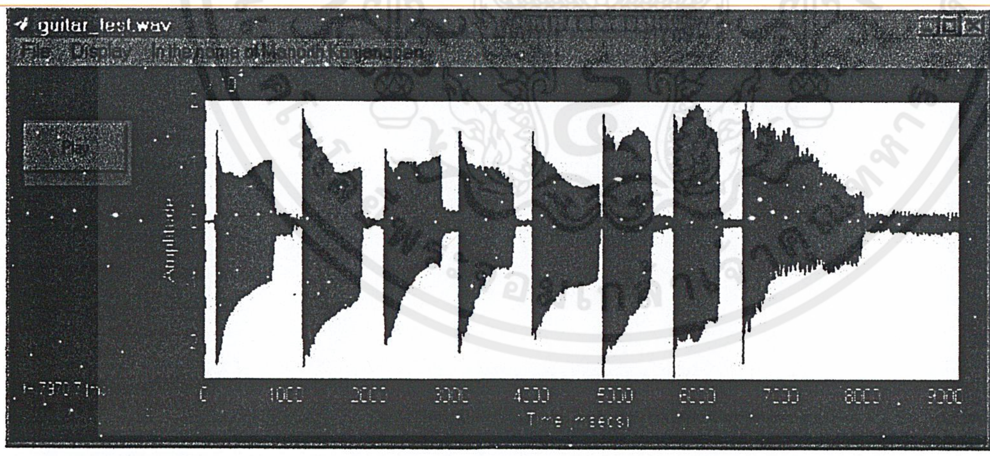
สำหรับในโครงงานพิเศษนี้ เรามีความจำเป็นในการใช้ไมโครคอมพิวเตอร์ในการควบคุมและประมวลผล อีกทั้งการนำระบบไมโครคอมพิวเตอร์มาใช้นั้นจะช่วยลดความยุ่งยากซับซ้อนในการออกแบบระบบฮาร์ดแวร์ เราจะใช้การเขียนโปรแกรม MATLAB เพื่อใช้ในการควบคุมและวิเคราะห์คลื่นเสียง จุดประสงค์หลักก็คือต้องการหาค่าค่าความถี่ของคลื่นที่รับเข้ามา เพื่อนำไปเปรียบเทียบกับโน้ตดนตรี แล้วส่งค่าไปเป็นข้อมูล MIDI เพื่อใช้ในการติดต่อกับ Synthesizer โดยในส่วนที่เกี่ยวข้องกับข้อมูล MIDI จะใช้การเขียนโปรแกรม visual basic ในการควบคุม

4.1 การวิเคราะห์คลื่นเสียงโดยโปรแกรม MATLAB

เราจะทำการวิเคราะห์คลื่นเสียงที่รับเข้ามาเป็น *.wav โดยไม่จำเป็นว่าจะเป็นเสียงเครื่องดนตรีชนิดต่าง ๆ เสมอไป อาจเป็นเสียงคนพูด หรือเสียงร้องเพลงก็ได้

<u>ขั้นตอนที่ 1</u> แสดงรูปคลื่นสัญญาณเสียง

ตัวอย่างของรูปแบบสัญญาณเสียงที่รับเข้ามาสามารถแสดงได้ดังรูป 4.1



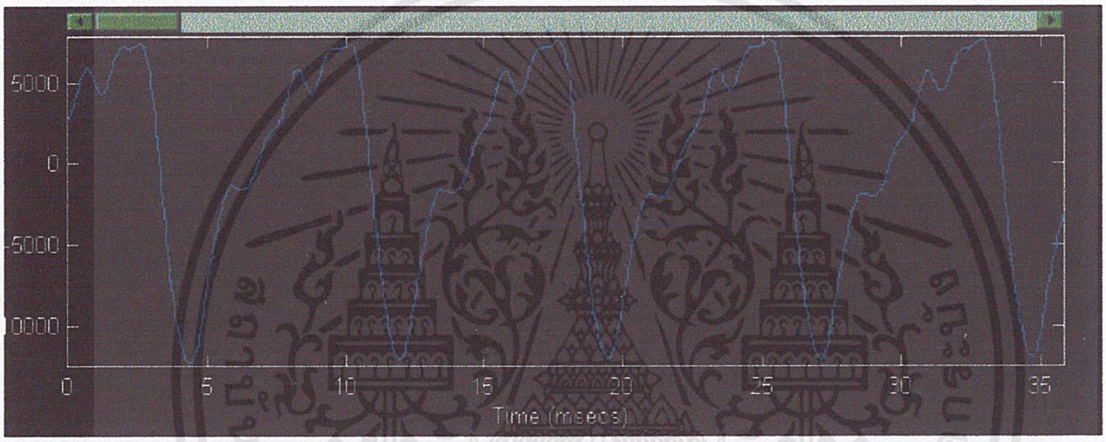
รูปที่ 4.1 แสดงคลื่นเสียงสัญญาณ guitar ที่มีการดีด 8 ครั้ง ต่อเนื่องกัน

<u>ขั้นตอนที่ 2</u> ทำการวิเคราะห์สัญญาณเสียงเพื่อหาค่าความถี่ และค่าเวลาจริงของสัญญาณ

จากรูปที่ 4.1 เราจะเห็นได้ว่า คลื่นสัญญาณที่รับเข้า แบ่งออกเป็น 8 ช่วงคลื่นสัญญาณ ดังนั้นในการวิเคราะห์ จะต้องทำการวิเคราะห์ 8 ครั้ง เพื่อหาค่าความถี่ของคลื่นสัญญาณเสียงแต่ละสัญญาณ

2.1 ทำการวิเคราะห์คลื่นสัญญาณที่ 1

รูปที่ 4.2 เป็นรูปที่แสดงคลื่นสัญญาณที่ 1 โดยทำการขยายรูปคลื่นเพื่อหาค่าความถี่ที่ถูกต้อง



รูปที่ 4.2 แสดงช่วงคลื่นสัญญาณแรกที่ถูกขยาย

จากรูปนี้เมื่อทำการขยายไปอีก จะได้คาบเวลาของคลื่นนี้มีค่าเท่ากับ 7.5 msecs

แล้วนำไปทำการหาค่าความถี่จาก

ความถี่ (เฮิร์ต) = 1 / คาบเวลา (sec)

$= 1 / 7.5 \times 10^{-3}$

$= 133.3333$ Hz

จากนั้น ก็นำความถี่ที่ได้นี้ ไปเทียบกับความถี่ มาตรฐาน ซึ่งจะพบว่ามีค่าใกล้เคียงกันกับตัวโน้ต C3

และในที่นี้ เพื่อให้ทราบรายละเอียดยิ่งขึ้นจึงนำข้อมูลที่ได้มานี้ไปใช้กับฟังก์ชัน frq2midi ที่เขียนขึ้นมา โดยฟังก์ชันตัวนี้จะนำค่าความถี่ที่ได้มาไปเทียบเคียงอย่างอัตโนมัติ และมีความแม่นยำมากยิ่งขึ้น โดยเมื่อเรียกฟังก์ชันนี้ขึ้นมาจะแสดงผลดังนี้คือ

[n,t]=frq2midi(133.3333)

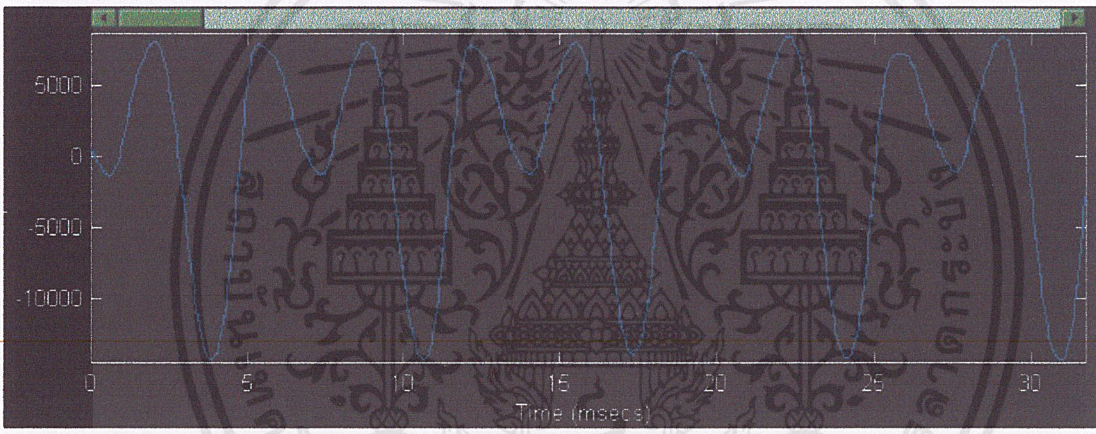
n = 48.3304

t =C3

โดยที่ n ค่าเทียบกับตัวโน้ต c กลาง หรือ c4 ซึ่งมีค่าเป็น 60 และจากบทเบื้องต้นเราทราบแล้วว่า c3 อยู่ต่ำกว่า c4 อยู่หนึ่ง octave โดยใน 1 octave จะมีอยู่ 12ช่วงดังนั้น ตัวโน้ต c 3 จึงมีค่า เป็น 48 ซึ่งตัวเลข 48.3304 จะมีค่าใกล้เคียงกับค่านี้จึงจัดให้อยู่ในระดับเสียงของโน้ต c 3

2.2 ทำการวิเคราะห์สัญญาณเสียงช่วงที่ 2

สัญญาณเสียงช่วงที่สองเมื่อนำมาทำการขยายจะแสดงได้ดังรูป 4.3



รูปที่ 4.3 แสดงสัญญาณคลื่นช่วงที่สองที่ถูกทำการขยาย

เช่นเดียวกันเมื่อทำการขยายเพิ่มขึ้นเราก็จะสามารถหาคาบเวลาของคลื่นเสียงนี้ได้เป็น 6.9 msec

ทำการหาความถี่ได้เป็น $1 / 6.9 \times 10^{-3} = 144.9275$ Hz

นำไปทำการเทียบความถี่ได้เป็น

[n,t]=frq2midi(144.9275)

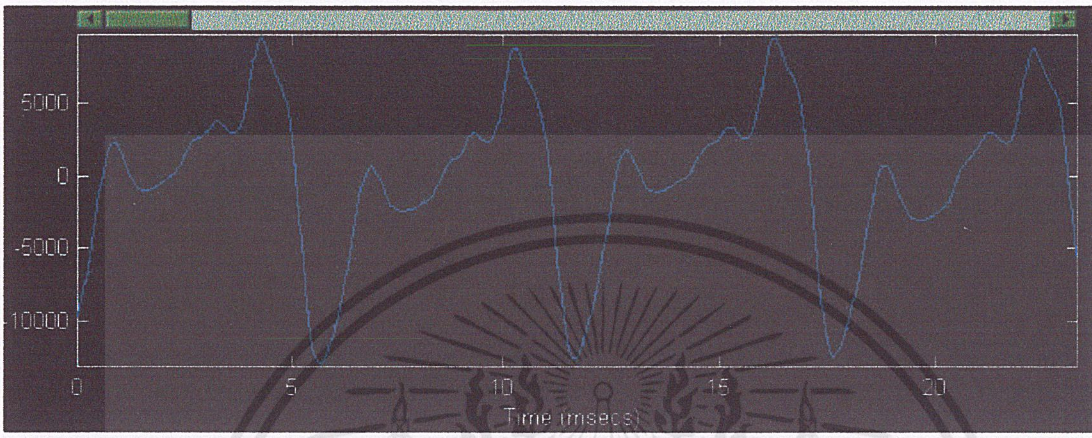
n = 49.7739

t =D3

ค่าของความถี่ที่ได้จะใกล้เคียงกับ ตัวโน้ต D3 ซึ่งตัวโน้ต D3 จะมีค่าเป็น 50 เมื่อเทียบกับโน้ต C กลาง

2.3 ทำการวิเคราะห์ช่างความถี่ที่ 3

เมื่อนำเอาช่างความถี่ที่ 3 มาทำการขยายจะเป็นดังรูป 4.4



รูปที่ 4.4 แสดงรูปคลื่นความถี่ช่วงที่สามเมื่อนำมาทำการขยาย

และเมื่อทำการขยายไปอีกจะได้ค่าคาบเวลาเป็น 6 msec

จากนั้นนำมาคำนวณหาความถี่ = $1 / 6 \times 10^{-3}$ = 166.6666 Hz

แล้วนำไปเทียบหาค่าของตัวโน้ตจะได้เป็น

[n,t]=frq2midi(166.6666)

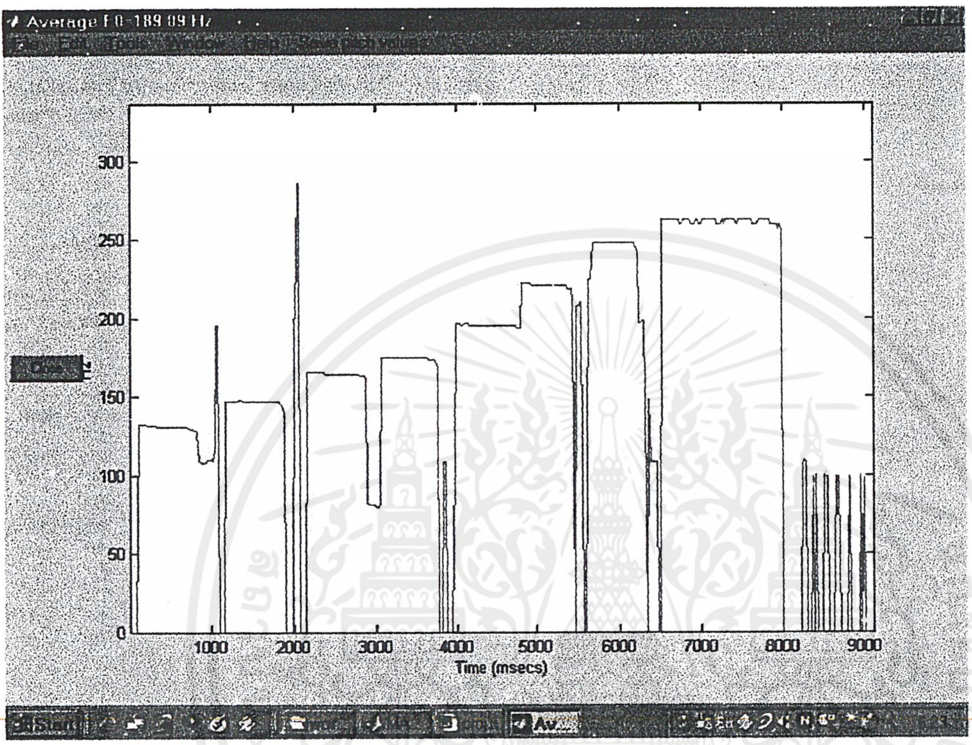
n =52.1935

t =E3

ตัวโน้ตที่ได้จะมีค่าใกล้เคียงกับ ตัวโน้ต E3 ซึ่งมีค่าเป็น 52 เมื่อเทียบกับโน้ต C กลาง

จากนั้นก็ทำการตรวจสอบค่าของช่วงความถี่ ที่มีทั้งหมด 8 ช่วงไปจนครบตามตัวอย่างที่ได้แสดงไว้ข้างต้น ซึ่งจะพบว่า ช่วงความถี่ทั้ง 8 ช่วงมีค่าของตัวโน้ตเป็น C3 ,D3 , E3 , F3 , G3 , A3 , B3 และ C4 ตามลำดับ

ซึ่งสามารถแสดงภาพของ *.wav ที่รับเข้ามาในรูปของ ความถี่กับ เวลาได้ดังรูป ที่ 4.5



รูปที่ 4.5 แสดงความสัมพันธ์ระหว่างความถี่กับคาบเวลาของคลื่นที่ทำการวิเคราะห์

4.2 กระบวนการในการส่งข้อมูล MIDI ออกทาง MIDI Port ของ Soundcard

จากเบื้องต้นเราได้รู้ถึงหลักการเบื้องต้นของ MIDI มาบ้างแล้ว ต่อไปจะเป็นการนำหลักการเหล่านั้นมาใช้ โดยในที่นี้จะใช้โปรแกรมควบคุมที่เขียนด้วย visual basic

ในส่วนของโปรแกรมจะมีฟังก์ชัน และโครงสร้างที่เกี่ยวเนื่องกับ MIDI มากมายโดยจำแนกเป็นกลุ่ม ๆ ได้ดังนี้

Allocating และ Managing Buffers

MIDIHDR

midiInAddBuffer

midiInPrepareHeader

midiInUnprepareHeader

midiOutPrepareHeader

midiOutUnprepareHeader

Callback Functions

MidiInProc

MidiOutProc

Device Capabilities

MIDIINCAPS

midiInGetDevCaps

midiInGetID

midiInGetNumDevs

MIDIOUTCAPS

midiOutGetDevCaps

midiOutGetID

midiOutGetNumDevs

MIDISTRMBUFFVER

Error Processing

midiInGetErrorText

midiOutGetErrorText

MIM_ERROR

MIM_LONGERROR

MM_MIM_ERROR

MM_MIM_LONGERROR

Managing MIDI Streams

midiStreamClose

midiStreamOpen

midiStreamOut

midiStreamPause

midiStreamPosition

midiStreamProperty

midiStreamRestart

midiStreamStop

Opening และ Closing Devices

midiInClose

midiInOpen

midiOutClose

midiOutOpen

MIM_CLOSE

MIM_OPEN

MM_MIM_CLOSE

MM_MIM_OPEN

MM_MOM_CLOSE

MM_MOM_OPEN

MOM_CLOSE

MOM_OPEN

Output Devices

KEYARRAY

midiOutCacheDrumPatches

midiOutCachePatches

midiOutGetVolume

midiOutSetVolume

PATCHARRAY

Playing a Message or Messages

MEVT_EVENTPARM

MEVT_EVENTTYPE

MIDIEVENT

midiOutLongMsg

midiOutReset

midiOutShortMsg

midiStreamOut

midiStreamPause

midiStreamRestart

midiStreamStop

MM_MOM_DONE

MM_MOM_POSITIONCB

Recording

midiConnect

midiDisconnect

midiInReset

midiInStart

midiInStop

MIDIPROPTEMPO

MIDIPROPTIMEDIV

MIM_DATA

MIM_LONGDATA

MIM_MOREDATA

MM_MIM_DATA

MM_MIM_MOREDATA

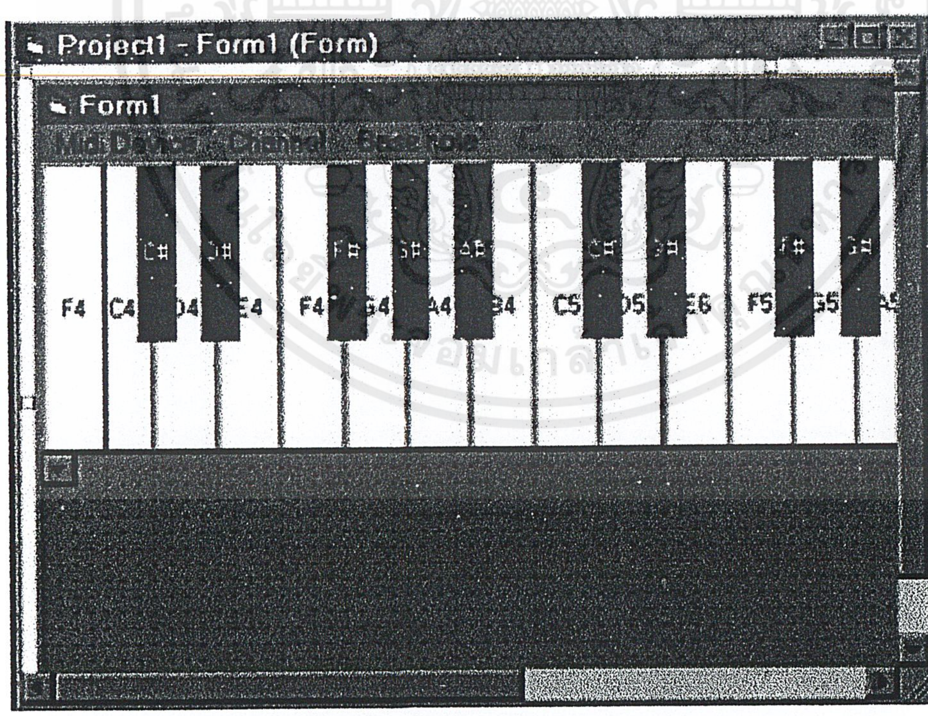
MM_MIM_LONGDATA

Sending Messages to Devices

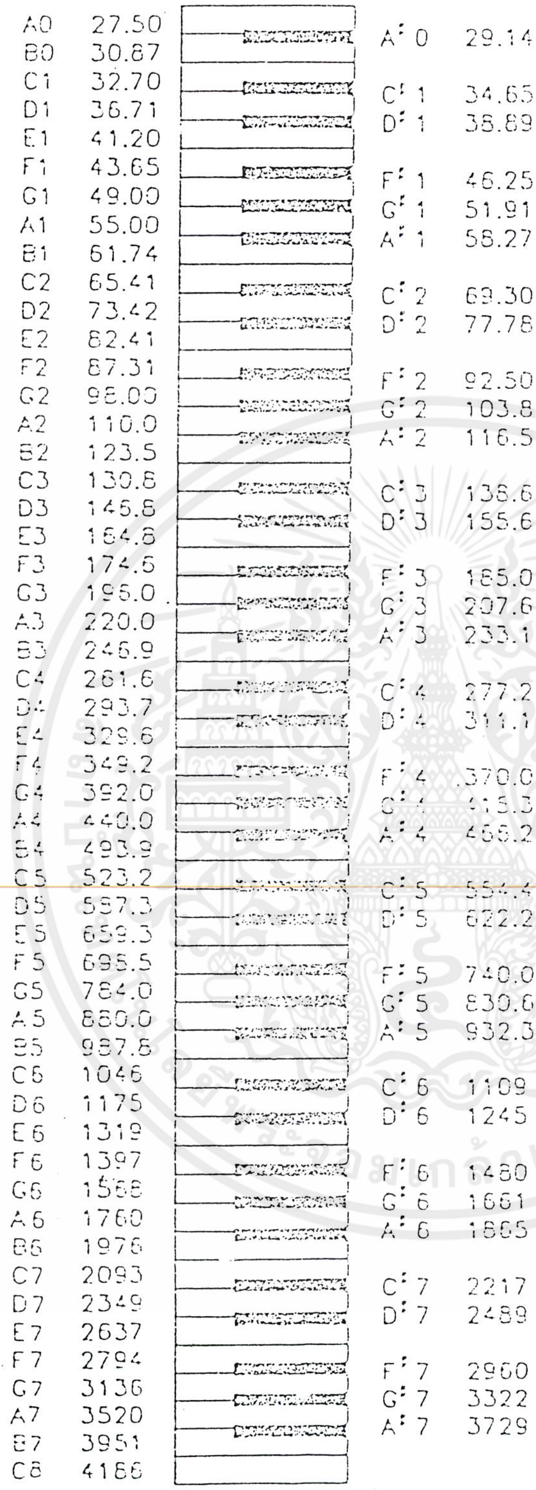
midiInMessage

midiOutMessage

และในการใช้งานเราจำเป็นที่จะต้องเรียกใช้ Library winmm.dll



รูปที่ 4.6 โปรแกรมที่เขียนขึ้นเพื่อโน้ตค่าต่าง ๆ ในสัญญาณของ MIDI



รูปที่ 4.7 ความถี่ของระดับเสียงเปียโนในระบบ Equal Temperament

ตัวอย่างของ Code ที่ใช้ในการเรียกฟังก์ชันที่ต้องการใช้งานของโปรแกรมนี้

```
Declare Function midiOutGetNumDevs Lib "winmm" () As Integer
Declare Function midiOutGetDevCaps Lib "winmm.dll" Alias "midiOutGetDevCapsA" (ByVal
uDeviceID As Long, lpCaps As MIDIOUTCAPS, ByVal uSize As Long) As Long
Declare Function midiOutClose Lib "winmm.dll" (ByVal hMidiOut As Long) As Long
Declare Function MIDIOutOpen Lib "winmm.dll" Alias "midiOutOpen" (lphMidiOut As Long, ByVal
uDeviceID As Long, ByVal dwCallback As Long, ByVal dwInstance As Long, ByVal dwFlags As
Long) As Long
Declare Function midiOutShortMsg Lib "winmm.dll" (ByVal hMidiOut As Long, ByVal dwMsg As Long)
As Long
```

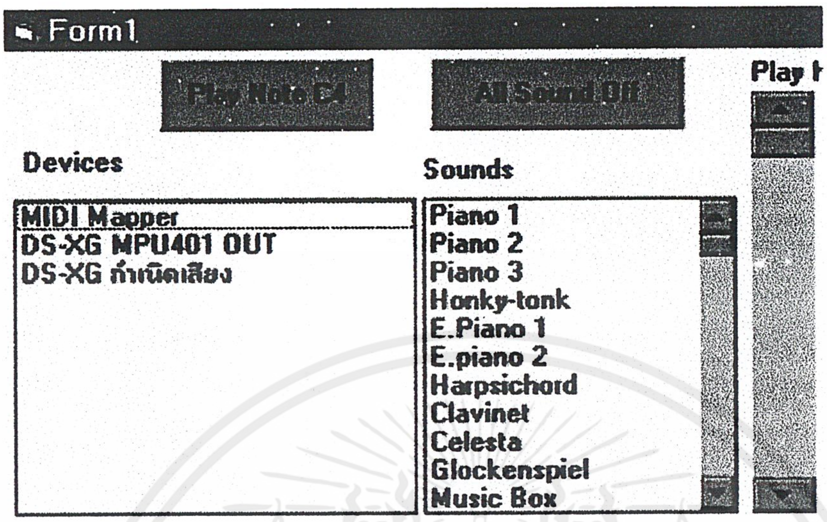
ตัวอย่าง code ในส่วนที่ใช้ในการส่งโน้ต

```
Private Sub StartNote(Index As Integer)
  If (Index = INVALID_NOTE) Then
    Exit Sub
  End If
  If (key(Index).Value = 1) Then
    Exit Sub
  End If
  key(Index).Value = 1
  midimsg = &H90 + ((baseNote + Index) * &H100) + (volume * &H10000) + channel
  midiOutShortMsg hmidi, midimsg
End Sub
```

ตัวอย่าง code ในส่วนที่ใช้ในการหยุดโน้ต

```
Private Sub StopNote(Index As Integer)
  If (Index = INVALID_NOTE) Then
    Exit Sub
  End If
  key(Index).Value = 0
  midimsg = &H80 + ((baseNote + Index) * &H100) + channel
  midiOutShortMsg hmidi, midimsg
End Sub
```

ต่อไปเป็นตัวอย่างของอีกหนึ่งโปรแกรมที่แสดงตัวอย่างการเปลี่ยนแปลงโปรแกรมเสียง



รูปที่ 4.8 ตัวอย่างโปรแกรมในการเรียกใช้ Control Change ของ MIDI

ตัวอย่างของ code ที่เกี่ยวข้อง มีดังนี้

```
Sub control_change(ch As Integer, ccnr As Integer, ByVal v As Integer)
  Call midi_outshort(&HB0 + ch, ccnr, v)
End Sub
 Sub program_change(ch As Integer, cc0nr As Integer, ByVal pnr As Integer)
  Call control_change(ch, 0, cc0nr)
  Call midi_outshort(&HC0 + ch, pnr, 0)
End Sub
```

และจากโปรแกรมนี้จะมีการใช้ฟังก์ชัน all sound off
ซึ่ง ตัว code ของโปรแกรมจะมีลักษณะดังนี้

```
Sub all_sounds_off()
Dim channel As Integer

  For channel = 0 To 15
    Call midi_outshort(&HB0 + channel, &H78, 0)
    Call midi_outshort(&HB0 + channel, &H7B, 0)
  Next
End Sub
```

โดยจะเห็นได้ว่า จะมีการเรียกใช้ ฟังก์ชันของ MIDI controller ไปที่ ตัวเลขของ controler 78 และ 7B เพื่อที่จะเรียกฟังก์ชัน All sounds off และ All note off ตามลำดับ

บทที่ 5
การใช้งานและข้อเสนอแนะ

การแปลงระดับความถี่เสียงไปเป็นข้อมูลมิดี้ Pitch – to – MIDI converter ซึ่งสร้างขึ้นจากแนวความคิดทางฟิสิกส์ ผนวกเข้ากับเทคโนโลยีทางไมโครคอมพิวเตอร์ ได้บรรลุเป้าหมายในขั้นต้น นั่นคือ การพิสูจน์ให้เห็นถึงกระบวนการแปลงรูปสัญญาณอีกวิธีหนึ่ง อันเป็นวิธีที่เราได้พัฒนาขึ้นมาใช้สำหรับงานด้านดนตรี

5.1 การใช้งานโปรแกรมการแปลงระดับความถี่เสียงไปเป็นข้อมูลมิดี้

เริ่มต้นด้วยการทำการบันทึกเสียงหรือระดับความถี่เสียงที่เราต้องการแปลง เป็น *.wav โดยมีโปรแกรมมากมายที่สามารถใช้ในการบันทึก ยกตัวอย่าง เช่น Sound recorder ที่มีอยู่ใน windows เป็นต้น จากนั้นก็ทำการวิเคราะห์ค่าความถี่เพื่อนำไปเปรียบเทียบว่าเป็นตัวโน้ตตัวไหนในทางดนตรี โดยการใช้โปรแกรมที่เขียนขึ้นโดยใช้โปรแกรม MATLAB โดย เมื่อเปิดโปรแกรมขึ้นมาให้พิมพ์คำดังต่อไปนี้

manorth และตามด้วยชื่อ .wav ที่เราได้ทำการบันทึกไว้

จากนั้นโปรแกรมก็จะทำการเปิดหน้าต่างขึ้นมาโดยเราจะเห็นรูปของสัญญาณคลื่นที่เราได้ทำการบันทึกไว้ โดยจะเป็นความสัมพันธ์ระหว่าง Amplitude กับ คาบเวลา (msec)

จากนั้นเราก็ทำการแสดงผลให้อยู่ในรูปของความถี่ กับคาบเวลา โดยฟังก์ชันนี้ จะอยู่ในหมวดของ Display เมื่อเราคลิกที่ส่วนนี้ ทางโปรแกรมก็จะทำการแปลงข้อมูลให้เรียบร้อย ซึ่งในตอนที่เราก็จะสามารถทราบความถี่ของ แต่ระดับความถี่ที่รับเข้ามา และเวลาจริงที่ใช้ในการบันทึก

ต่อไปก็ทำการหาค่าของตัวโน้ตโดยเรียกฟังก์ชัน frq2midi (การแปลงความถี่ไปเป็นข้อมูล MIDI) โดยมีขั้นตอนในการใช้งานดังนี้

[n,t] = frq2midi(f)

โดยในที่นี้เราทำการป้อนค่าความถี่แทนลงไปที่ตัวแปร f จากนั้นโปรแกรมก็จะทำการ run ผลออกมาโดยที่ จะ ให้ค่า n คือ ค่าของตัวโน้ตเมื่อเทียบกับ ตัวโน้ต C กลางซึ่งมีค่าเป็น 60 และค่า t เป็นค่าของตัวโน้ตจะถูกส่งออกมาเป็นข้อมูลในระบบมิดี้

จากนั้นเราก็ทำการเก็บข้อมูลที่ได้มาโดยเฉพาะ ตัวโน้ตซึ่งเป็นค่าของแต่ละความถี่ ค่าเวลาจริงที่ใช้แล้วทำการส่งออกเป็นข้อมูล MIDI โดยใช้ โปรแกรม Visual Basic โดยการส่งข้อมูลนี้เราสามารถทำการแก้ไขในรายละเอียดปลีกย่อยได้อีก เช่น เราสามารถปรับค่าความดัง เปลี่ยนเสียงเป็นเสียงอื่นได้อีก โดยการใช้ ค่าเสียงที่มีอยู่ใน GM ซึ่งเป็นฟังก์ชันหนึ่งของ MIDI

5.2 ข้อเสนอแนะ

จากที่กล่าวมาข้างต้นจะเห็นได้ว่า เราต้องใช้ 2 โปรแกรมทำงานร่วมกัน ซึ่งทำให้เกิดการยุ่งยากต่อการใช้งาน ดังนั้นในการพัฒนาต่อไปควรจะเขียนโปรแกรมที่ สามารถนำเอา source code ของทั้งสองโปรแกรมมาใช้งานร่วมกันได้ ซึ่งในความคิดของผู้จัดทำโครงงานพิเศษ คิดว่าควรจะใช้โปรแกรม Visual C++ เพราะว่า โปรแกรม MATLAB สามารถทำการแปลง code ของโปรแกรมไปเป็น code ทางภาษา C ได้ และในขณะเดียวกัน source code ของ โปรแกรม Visual Basic ก็มีการใช้งานในฟังก์ชันที่มีความเกี่ยวเนื่องกับฟังก์ชันของภาษา Visual C++ ดังนั้นหากเขียนโปรแกรมขึ้นมาใหม่ตามที่กล่าวมาข้างต้นก็จะทำให้สามารถ run โปรแกรมได้เพียงครั้งเดียว คือ โปรแกรมสามารถทำการ รับค่า *.wav แล้วส่งค่าเป็นข้อมูล MIDIได้ในทันทีทันใด

ภาคผนวก ก.

VISUAL BASIC PROGRAM

```
Option Explicit
Public Const MAXPNAMELEN = 32           ' Maximum product name length
Global Const MAXERRORLENGTH = 128
' Error values for functions used in this sample. See the function for more information
Public Const MMSYSERR_BASE = 0
Public Const MMSYSERR_BADDEVICEID = (MMSYSERR_BASE + 2)     ' device ID out of range
Public Const MMSYSERR_INVALPARAM = (MMSYSERR_BASE + 11)     ' invalid parameter passed
Public Const MMSYSERR_NODRIVER = (MMSYSERR_BASE + 6)        ' no device driver present
Public Const MMSYSERR_NOMEM = (MMSYSERR_BASE + 7)           ' memory allocation error

Public Const MMSYSERR_INVALHANDLE = (MMSYSERR_BASE + 5)     ' device handle is invalid
Public Const MIDIERR_BASE = 64
Public Const MIDIERR_STILLPLAYING = (MIDIERR_BASE + 1)      ' still something playing
Public Const MIDIERR_NOTREADY = (MIDIERR_BASE + 3)          ' hardware is still busy
Public Const MIDIERR_BADOPENMODE = (MIDIERR_BASE + 6)       ' operation unsupported w/
open mode

'User-defined variable the stores information about the MIDI output device.
Type MIDIOUTCAPS
   wMid As Integer              ' Manufacturer identifier of the device driver for the MIDI output device
                                ' For a list of identifiers, see the Manufacturer Indentifier topic in the
                                ' Multimedia Reference of the Platform SDK.

   wPid As Integer              ' Product Identifier Product of the MIDI output device. For a list of
                                ' product identifiers, see the Product Identifiers topic in the Multimedia
                                ' Reference of the Platform SDK.

   vDriverVersion As Long       ' Version number of the device driver for the MIDI output device.
                                ' The high-order byte is the major version number, and the low-order byte is
                                ' the minor version number.

   szPname As String * MAXPNAMELEN  ' Product name in a null-terminated string.

   wTechnology As Integer       ' One of the following that describes the MIDI output device:
                                '    MOD_FMSYNTH-The device is an FM synthesizer.
```

' MOD_MAPPER-The device is the Microsoft MIDI mapper.

' MOD_MIDIPORT-The device is a MIDI hardware port.

' MOD_SQSYNTH-The device is a square wave synthesizer.

' MOD_SYNTH-The device is a synthesizer.

wVoices As Integer ' Number of voices supported by an internal synthesizer device. If the
' device is a port, this member is not meaningful and is set to 0.

wNotes As Integer ' Maximum number of simultaneous notes that can be played by an
internal
' synthesizer device. If the device is a port, this member is not meaningful
' and is set to 0.

wChannelMask As Integer ' Channels that an internal synthesizer device responds to, where
the least
' significant bit refers to channel 0 and the most significant bit to channel
' 15. Port devices that transmit on all channels set this member to 0xFFFF.

dwSupport As Long ' One of the following describes the optional functionality supported
by
' the device:
' MIDICAPS_CACHE-Supports patch caching.
' MIDICAPS_LRVOLUME-Supports separate left and right volume control.
' MIDICAPS_STREAM-Provides direct support for the midiStreamOut
function.
' MIDICAPS_VOLUME-Supports volume control.
'
' If a device supports volume changes, the MIDICAPS_VOLUME flag will be
set
' for the dwSupport member. If a device supports separate volume changes
on
' the left and right channels, both the MIDICAPS_VOLUME and the
' MIDICAPS_LRVOLUME flags will be set for this member.

End Type

Type MIDIHDR

```
  lpData As String           ' pointer to locked data block
  dwBufferLength As Long      ' length of data in data block
  dwBytesRecorded As Long      ' used for input only
  dwUser As Long             ' for client's use
  dwFlags As Long             ' assorted flags (see defines)
  lpNext As Long            ' reserved for driver
  reserved As Long            ' reserved for driver
End Type
Declare Function midiOutGetNumDevs Lib "winmm" () As Integer
' This function retrieves the number of MIDI output devices present in the system.
' The function returns the number of MIDI output devices. A zero return value means
' there are no MIDI devices in the system.

Declare Function midiOutGetDevCaps Lib "winmm.dll" Alias "midiOutGetDevCapsA" (ByVal
uDeviceID As Long, lpCaps As MIDIOUTCAPS, ByVal uSize As Long) As Long
' This function queries a specified MIDI output device to determine its capabilities.
' The function requires the following parameters;
'    uDeviceID-    unsigned integer variable identifying of the MIDI output device. The
'              device identifier specified by this parameter varies from zero to one
'              less than the number of devices present. This parameter can also be a
'              properly cast device handle.
'    lpMidiOutCaps- address of a MIDIOUTCAPS structure. This structure is filled with
'              information about the capabilities of the device.
'    cbMidiOutCaps- the size, in bytes, of the MIDIOUTCAPS structure. Use the Len
'              function with the MIDIOUTCAPS variable as the argument to get
'              this value.
'
' The function returns MMSYSERR_NOERROR if successful or one of the following error values:
'    MMSYSERR_BADDEVICEID    The specified device identifier is out of range.
'    MMSYSERR_INVALPARAM     The specified pointer or structure is invalid.
'    MMSYSERR_NODRIVER       The driver is not installed.
'    MMSYSERR_NOMEM          The system is unable to load mapper string description.

Declare Function midiOutClose Lib "winmm.dll" (ByVal hMidiOut As Long) As Long
' The function closes the specified MIDI output device. The function requires a
```

' handle to the MIDI output device. If the function is successful, the handle is no
' longer valid after the call to this function. A successful function call returns
' MMSYSERR_NOERROR.

' A failure returns one of the following:
'   MIDIERR_STILLPLAYING Buffers are still in the queue.
'   MMSYSERR_INVALHANDLE The specified device handle is invalid.
'   MMSYSERR_NOMEM   The system is unable to load mapper string description.

Declare Function MIDIOutOpen Lib "winmm.dll" Alias "midiOutOpen" (lphMidiOut As Long, ByVal uDeviceID As Long, ByVal dwCallback As Long, ByVal dwInstance As Long, ByVal dwFlags As Long) As Long
' The function opens a MIDI output device for playback. The function requires the
' following parameters
'   lphmo-   Address of an HMIDIOUT handle. This location is filled with a
'   handle identifying the opened MIDI output device. The handle
'   is used to identify the device in calls to other MIDI output
'   functions.
'   uDeviceID-   Identifier of the MIDI output device that is to be opened.
'   dwCallback-   Address of a callback function, an event handle, a thread
'   identifier, or a handle of a window or thread called during
'   MIDI playback to process messages related to the progress of
'   the playback. If no callback is desired, set this value to 0.
'   dwCallbackInstance- User instance data passed to the callback. Set this value to 0.
'   dwFlags-Callback flag for opening the device. Set this value to 0.
'

' The function returns MMSYSERR_NOERROR if successful or one of the following error values:
'   MIDIERR_NODEVICE-   No MIDI port was found. This error occurs only when the mapper is opened.
'   MMSYSERR_ALLOCATED-   The specified resource is already allocated.
'   MMSYSERR_BADDEVICEID- The specified device identifier is out of range.
'   MMSYSERR_INVALPARAM- The specified pointer or structure is invalid.
'   MMSYSERR_NOMEM-   The system is unable to allocate or lock memory.

```
Declare Function midiOutShortMsg Lib "winmm.dll" (ByVal hMidiOut As Long, ByVal dwMsg As Long)
As Long
' This function sends a short MIDI message to the specified MIDI output device. The function
' requires the handle to the MIDI output device and a message is packed into a doubleword
' value with the first byte of the message in the low-order byte. See the code sample for
' how to create this value.
'
' The function returns MMSYSERR_NOERROR if successful or one of the following error values:
'    MIDIERR_BADOPENMODE-  The application sent a message without a status byte to a stream
handle.
'    MIDIERR_NOTREADY-     The hardware is busy with other data.
'    MMSYSERR_INVALHANDLE- The specified device handle is invalid.
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
Private Declare Function mciGetErrorString Lib "winmm.dll" Alias "mciGetErrorStringA" (ByVal dwError
As Long, ByVal lpstrBuffer As String, ByVal uLength As Long) As Long
'Private Declare Function midiOutGetNumDevs Lib "winmm" () As Integer
'Private Declare Function midiOutGetDevCaps Lib "winmm.dll" Alias "midiOutGetDevCapsA" (ByVal
uDeviceID As Long, lpCaps As MIDIOUTCAPS, ByVal uSize As Long) As Long
Private Declare Function GetPrivateProfileString Lib "kernel32" Alias "GetPrivateProfileStringA" (ByVal
lpApplicationName As String, ByVal lpKeyName As Any, ByVal lpDefault As String, ByVal
lpReturnedString As String, ByVal nSize As Long, ByVal lpFileName As String) As Long
Private Declare Function WritePrivateProfileString Lib "kernel32" Alias "WritePrivateProfileStringA"
(ByVal lpApplicationName As String, ByVal lpKeyName As Any, ByVal lpString As Any, ByVal
lpFileName As String) As Long
Private Declare Function GetProfileString Lib "kernel32" Alias "GetProfileStringA" (ByVal lpAppName
As String, ByVal lpKeyName As String, ByVal lpDefault As String, ByVal lpReturnedString As String,
ByVal nSize As Long) As Long
Private Declare Function WriteProfileString Lib "kernel32" Alias "WriteProfileStringA" (ByVal
lpszSection As String, ByVal lpszKeyName As String, ByVal lpszString As String) As Long
Private Declare Function midiOutGetVolume Lib "winmm.dll" (ByVal uDeviceID As Long, lpdwVolume
As Long) As Long
Private Declare Function midiOutSetVolume Lib "winmm.dll" (ByVal uDeviceID As Long, ByVal
dwVolume As Long) As Long
```

```
Private Declare Function midiOutGetErrorText Lib "winmm.dll" Alias "midiOutGetErrorTextA" (ByVal err
As Long, ByVal lpText As String, ByVal uSize As Long) As Long
'Private Declare Function MIDIOutOpen Lib "winmm.dll" Alias "midiOutOpen" (lphMidiOut As Long,
ByVal uDeviceID As Long, ByVal dwCallback As Long, ByVal dwInstance As Long, ByVal dwFlags
As Long) As Long
'Private Declare Function midiOutClose Lib "winmm.dll" (ByVal hMidiOut As Long) As Long
Private Declare Function midiOutPrepareHeader Lib "winmm.dll" (ByVal hMidiOut As Long,
lpMidiOutHdr As MIDIHDR, ByVal uSize As Long) As Long
Private Declare Function midiOutUnprepareHeader Lib "winmm.dll" (ByVal hMidiOut As Long,
lpMidiOutHdr As MIDIHDR, ByVal uSize As Long) As Long
'Private Declare Function midiOutShortMsg Lib "winmm.dll" (ByVal hMidiOut As Long, ByVal dwMsg
As Long) As Long
Private Declare Function midiOutLongMsg Lib "winmm.dll" (ByVal hMidiOut As Long, lpMidiOutHdr
As MIDIHDR, ByVal uSize As Long) As Long
```

```
Option Explicit
Const INVALID_NOTE = -1     ' Code for keyboard keys that we don't handle
Dim m_hmidiout As Long
Dim numDevices As Long      ' number of midi output devices
Dim curDevice As Long       ' current midi device
Dim hmidi As Long           ' midi output handle
Dim rc As Long              ' return code
Dim midimsg As Long         ' midi output message buffer
Dim channel As Integer      ' midi output channel
Dim volume As Integer       ' midi volume
Dim baseNote As Integer     ' the first note on our "piano"

Private Sub base_Click()
Dim s As String
  Dim i As Integer
  s = InputBox("Enter the new base note for the keyboard (0 - 111)", "Base note", CStr(baseNote))
  If IsNumeric(s) Then
    i = CInt(s)
    If (i >= 0 And i < 112) Then
      baseNote = i
      End If
      End If
End Sub

Private Sub Chan_Click(Index As Integer)
Chan(channel).Checked = False
  channel = Index
  Chan(channel).Checked = True
End Sub

Private Sub device_Click(Index As Integer)
 device(curDevice + 1).Checked = False
  device(Index).Checked = True
  curDevice = Index - 1
```

```
  rc = midiOutClose(hmidi)
  rc = MIDIOutOpen(hmidi, curDevice, 0, 0, 0)
  If (rc <> 0) Then
    MsgBox "Couldn't open midi out, rc = " & rc
  End If
End Sub

' If user presses a keyboard key, start the corresponding midi note
Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
  StartNote NoteFromKey(KeyCode)
End Sub

' If user lifts a keyboard key, stop the corresponding midi note
Private Sub Form_KeyUp(KeyCode As Integer, Shift As Integer)
  StopNote NoteFromKey(KeyCode)
End Sub
Private Sub Form_Load()
  Dim i As Long
  Dim caps As MIDIOUTCAPS

  ' Set the first device as midi mapper
  device(0).Caption = "MIDI Mapper"
  device(0).Visible = True
  device(0).Enabled = True

  ' Get the rest of the midi devices
  numDevices = midiOutGetNumDevs()
  For i = 0 To (numDevices - 1)
    midiOutGetDevCaps i, caps, Len(caps)
    device(i + 1).Caption = caps.szPname
    device(i + 1).Visible = True
    device(i + 1).Enabled = True
  Next

  ' Select the MIDI Mapper as the default device
```

```
  device_Click (0)

  ' Set the default channel
  channel = 0
  Chan(channel).Checked = True

  ' Set the base note
  baseNote = 60

  ' Set volume range
  volume = 127
  vol.Min = 127
  vol.Max = 0
  vol.Value = volume

End Sub

Private Sub Form_Unload(Cancel As Integer)
  ' Close current midi device
  rc = midiOutClose(hmidi)
End Sub
' Start a note when user click on it
Private Sub key_MouseDown(Index As Integer, Button As Integer, Shift As Integer, x As Single, Y As
Single)
  StartNote (Index)
End Sub

' Stop the note when user lifts the mouse button
Private Sub key_MouseUp(Index As Integer, Button As Integer, Shift As Integer, x As Single, Y As
Single)
  StopNote (Index)
End Sub
Private Sub StartNote(Index As Integer)
  If (Index = INVALID_NOTE) Then
    Exit Sub
```

```
  End If
  If (key(Index).Value = 1) Then
    Exit Sub
  End If
  key(Index).Value = 1
  midimsg = &H90 + ((baseNote + Index) * &H100) + (volume * &H10000) + channel
  midiOutShortMsg hmidi, midimsg
End Sub

' Raise the button and send midi stop event
Private Sub StopNote(Index As Integer)
  If (Index = INVALID_NOTE) Then
    Exit Sub
  End If
  key(Index).Value = 0
  midimsg = &H80 + ((baseNote + Index) * &H100) + channel
  midiOutShortMsg hmidi, midimsg
End Sub

' Get the note corresponding to a keyboard key
Private Function NoteFromKey(key As Integer)
  NoteFromKey = INVALID_NOTE
  Select Case key
  Case vbKeyC4
    NoteFromKey = 0
  Case vbKeyC#
    NoteFromKey = 1
  Case vbKeyD4
    NoteFromKey = 2
  Case vbKeyD#
    NoteFromKey = 3
  Case vbKeyE4
    NoteFromKey = 4
  Case vbKeyF4
    NoteFromKey = 5
```

```
  Case vbKeyF#
    NoteFromKey = 6
  Case vbKeyG4
    NoteFromKey = 7
  Case vbKeyG#
    NoteFromKey = 8
  Case vbKeyA4
    NoteFromKey = 9
  Case vbKeyA#
    NoteFromKey = 10
  Case vbKeyB
    NoteFromKey = 11
  Case vbKeyC5
    NoteFromKey = 12
  Case vbKeyC#
    NoteFromKey = 13
  Case vbKeyD5
    NoteFromKey = 14
  Case vbKeyD#
    NoteFromKey = 15
  Case vbKeyE5
    NoteFromKey = 16
  End Select

End Function

Private Sub vol_Change()
volume = vol.Value
End Sub
```

ภาคผนวก ข.

MATLAB PROGRAM

```
function manorth(infile,Srate1)


global hl hr doit
global fed shW
global filename
global fno Srate n_Secs filename AXISLOC  nLPC HDRSIZE
global nChannels fftSize filterWeights UpperFreq LowFreq ChanpUp
global center LPCSpec SpecUp Dur DurUp filterA filterB  S1 S0
global HDRSIZE cAxes En Be TIME centSMSP Asmsp Bsmsp OVRL WAV1 CLR
global TWOFILES wav agcsc MAX_AM upFreq upFreq2 FIX_SCALE
global SHOW_CRS SHOW_CHN ftype ftype2 bpsa bpsa2 PREEMPH LPC_ONLY
global lpcParam fft_par NAR_BAND SPEC_EXP FILT_TYPE VOL_MAX
global VOL_NORM
global sli n_samples FFT_SET XFFT
global tpc boc
global smp frq TOP
global preUp  defUp w64Up w128Up w256Up w512Up narUp
global crsUp chnUp lpcUp chnlpUp fbrd fnar
global ovlpfilt SET_X_AXIS LD_LABELS



if (nargin < 1)
 [pth,fname] = dlgopen('open','*.ils;*.wav');
 if ((~isstr(fname)) | ~min(size(fname))), return; end
 filename=[pth,fname];
else
  filename=infile;
end
pos = get(0, 'screensize'); % get the screensize
sWi = pos(3);
sHe = pos(4);

WIDTH   =round(0.9375*sWi);
HEIGHT  =round(0.43*sHe) ;
LEFT    =round(0.025*sWi);
BOTTOM  =round(sHe-HEIGHT-40);

LPCSpec = 1;             % if 0 display FFT, else LPC spectrum
TIME=1;                  % if 0 display spectrogram, else time
waveform
WAV1=0;                  % If 1, then it is a .wav file with 8
bits/sample
CLR=1;                   % If 1 display spectrogram in color, else in
gray scale
TWOFILES=0;              % If 1, then display two files
wav(1)=0; wav(2)=0;      % Used in case of dual-displays for 1-byte
samples, as in WAV
upFreq=1.0;              % Upper frequency (percentage of Srate/2) in
spectrogram
upFreq2=1.0;             % Upper frequency in spectrogram
(1.0=Srate/2)
FIX_SCALE=-1;            % if > 0, then channel y-axis is always 0-15
dB
SHOW_CRS=1;              % if 1, show cursor lines, else dont
SHOW_CHN=1;              % if 1, show channel output/LPC display
PREEMPH=1;               % if 1, do pre-emphasis when computing the
spectrogram
```

```
LPC_ONLY=1;               % if 1, only the LPC specrtrum is displayed
lpcParam(1)=1;            % if 1, use hamming window in LPC analysis,
else use rectangular
lpcParam(2)=0;            % if 1, first-order pre-emphasis in LPC
analysis, else dont
         % NOTE: this pre-emphasis is done in addition to the CIS
pre-emphasis
lpcParam(3)=-1;           % if 1, enhance spectral peaks in LPC
analysis
fft_par(1)=1;             % if 1, use lines when plotting FFT, else use
pickets
fft_par(2)=1;             % if 1 use hamming window, else rectangular
NAR_BAND=0;               % if 1, display narrowband spectrograms
SPEC_EXP=0.25;    % Used in spectrogram display (root compression)
FILT_TYPE='broad';   % the filter type
VOL_MAX=0;      % used for controling the volume
VOL_NORM=1;     % if 1, then volume is normalized
FFT_SET=0;      % if 1, use user defined FFT size
XFFT=0;         % same as above
SET_X_AXIS=0;
ovlpfilt=0;
LD_LABELS=0;
TOP=0;

fp = fopen(filename,'r');

if fp <=0
   disp('ERROR! File not found..')
   return;
end

ftype='short'; bpsa=2;     % bytes per sample
ftype2='short'; bpsa2=1;

ind1=find(filename == '.');
if length(ind1)>1, ind=ind1(length(ind1)); else, ind=ind1; end;
ext = lower(filename(ind+1:length(filename)));



[HDRSIZE, xSrate, bpsa, ftype] =  gethdr(fp,ext);


if xSrate==0, return;
else Srate=xSrate; end;

if strcmp(ftype,'ascii')
 x=fscanf(fp,'%f',inf);
else
 x  = fread(fp,inf,ftype);
end




fclose(fp);

if Srate<6000 | Srate>45000 & nargin<2
h=warndlg('Sampling rate not in the range: 10,000 < F < 45,000 .
Setting it to  10,000 Hz.','WARNING!');
```

```
  disp('Warning! Sampling rate not in the range: 6,000 < F <
45,000');
  disp('...Setting it to the default value of 10,000 Hz.');
  Srate=10000;
end


x= x - mean(x);  %----------remove the DC bias----

if (nargin==2)
 Srate  = str2num(Srate1);
 if Srate<10000 | Srate>45000
   error('Invalid sampling frequency specified: 10,000<F<45,000');
 end
end

MAX_AM=2048; % This allows 12-bit resolution

mx=max(x);
agcsc=MAX_AM/mx;


n_samples = length(x);
n_Secs    = n_samples/Srate;
Dur=10.0;   % Duration in msec of window
S1=n_samples;
S0=0;
Be=S0;
En=S1;
OVRL=1;  % if 1 then hold off plots, else hold on plots in 'pllpc'

fprintf('Samp.Freq: %d Hz,  num.samples: %d (%4.2f
secs)\n',Srate,n_samples,n_Secs);

fno =  figure('Units', 'Pixels', 'Position', [LEFT BOTTOM WIDTH
HEIGHT],...
   'WindowButtonDownFcn','mclick',...
   'Resize','on','Name',filename,'NumberTitle','Off',...
   'Menubar','None','WindowButtonMotionFcn','showpt',...
   'KeyPressFcn','getkb','Color','k');

%------------ deterime the dimensions of the axis ------------

le=round(0.2*WIDTH);
bo=round(0.174*HEIGHT);
wi=round(0.773*WIDTH);
he=round(0.739*HEIGHT);

AXISLOC = [le bo wi he];
cAxes = axes('Units','Pixels','Position',AXISLOC);


axes(cAxes);



   Et=1000*n_samples/Srate;
   xax=0:1000/Srate:(Et-1000/Srate);

   plot(xax,x,'y')
```

```
   xlabel('Time (msecs)');
   ylabel('Amplitude');
   set(gca,'Color','k'); set(gca,'Xcolor','w');
set(gca,'YColor','w');
   %set(gca,'Units','points','FontSize',9);
   if min(x)<-1000 | mx >1000
     axis([0 Et min(x)-200 max(x)+200]);
   else
     axis([0 Et min(x) max(x)]);
   end

   %set(gca,'Color','k');
   set(gca,'XColor','w');
   set(gca,'YColor','w');
xywh = get(fno, 'Position');
axi=AXISLOC;

% Buttons.
left = 10;
wide = 80;
top  = xywh(4) - 10;
high = 22;
high=22;
if 9*(22+8) > xywh(4), high=17; end;
inc  = high + 8;
%---------- Display the slider and the push-buttons------------
sli = uicontrol('Style','slider','min',0,'max',1000','Callback',...
   'getslide','Position',[axi(1) axi(2)+axi(4)+2 axi(3) 12]);



Zin = uicontrol('Style', 'PushButton', 'Callback','zoomi(''in'')',
...
    'HorizontalAlign','center', 'String', 'Zoom In',...
    'Position', [left top-high wide high]);

top = top - inc;
Zout = uicontrol('Style', 'PushButton', 'Callback','zoomi(''out'')',
...
    'HorizontalAlign','center', 'String', 'Zoom Out',...
    'Position', [left top-high wide high]);
if Srate>12000
  nLPC=14;
else
 nLPC=12; % initialize LPC order
end
top = top - inc-20;
uicontrol('Style','Frame','Position',[left top-high-10 wide+5
high+30],...
   'BackgroundColor','b');

uicontrol('Style','text','Position',[left+wide/3 top 40 high-
3],'BackGroundColor','b',...
   'HorizontalAlignment','left','ForeGroundColor','w','String','Play'
);

plUp = uicontrol('Style', 'PushButton', 'Callback', 'playf(''all'')',
...
     'HorizontalAlign','center', 'String', 'all',...
     'Position', [left top-high wide high]);
```

```
%---Draw the squares in case its TWOFILES
wwi=xywh(3); whe=xywh(4);
tpc=uicontrol('Style','text','Position',[wwi-10 2*whe/3+10 10
10],'String',' ','BackGroundColor',[0 0 0]);
boc=uicontrol('Style','text','Position',[wwi-10 whe/3-10 10
10],'String',' ','BackGroundColor',[0 0 0]);


%----Draw the time and freq numbers----------
smp=uicontrol('Style','text','Position',[10 30 wide+10
15],'BackGroundColor',[0 0 0],...
   'HorizontalAlignment','left');
frq=uicontrol('Style','text','Position',[10 10 wide+10
15],'BackGroundColor',[0 0 0],...
   'HorizontalAlignment','left');


%
%--------------------------MENUS-----------------------------------
%
ff=uimenu('Label','File');

   uimenu(ff,'Label','Exit','CallBack','quitall','Separator','on');



fd=uimenu('Label','Display');


   %uimenu(fd,'Label','Energy Plot','Callback','engy');
   fdf0=uimenu(fd,'Label','F0 contour');
        uimenu(fdf0,'Label','Autocorrelation
approach','Callback','estf0(''autocor'')');



%-----------Initialize handles to cursor lines ------------

np=3; Ylim=get(gca,'YLim');
hl=line('Xdata',[np np],'Ydata',Ylim,'Linestyle','-',...
      'color',[0 0 0],'Erasemode','xor');

hr=line('Xdata',[np np],'Ydata',Ylim,'Linestyle','--',...
      'color',[0 0 0],'Erasemode','xor');
doit=0;
```

```
function crsrline(np,Ylim,button)



global TWOFILES TOP  hl hr doit
global htop hbot lastClick hwi frst1 frst0 doit0 doit1


if strcmp(button,'left')  % -- left mouse button was clicked
   if TWOFILES==1

      if TOP==1
          if doit1==1 | frst0==1
             hwi(1,1)=line('Xdata',[np np],'Ydata',Ylim,...%
'Linestyle','--',...
                'color','r','Erasemode','xor');
             hwi(1,2)=line('Xdata',[np+4
np+4],'Ydata',Ylim,...%'Linestyle','--',...
                'color',[0 0 0],'Erasemode','xor');
             frst0=0;
            doit1=0;
         else
            set(hwi(1,1),'Xdata',[np np],'Ydata',Ylim,'color','r');
         end
      else

         if doit0==1 | frst1==1
             hwi(2,1)=line('Xdata',[np np],'Ydata',Ylim,...%
'Linestyle','--',...
                'color','r','Erasemode','xor');
             hwi(2,2)=line('Xdata',[np+4
np+4],'Ydata',Ylim,...%'Linestyle','--',...
                'color',[0 0 0],'Erasemode','xor');
             frst1=0;
             doit0=0;
         else
            set(hwi(2,1),'Xdata',[np np],'Ydata',Ylim,'color','r');
         end
      end
   else

     invalid=0;
     eval('findobj(hl);','invalid=1;');
     if doit==1 | invalid==1
      hl=line('Xdata',[np np],'Ydata',Ylim,...% 'Linestyle','--',...
       'color','r','Erasemode','xor');
      hr=line('Xdata',[np+4 np+4],'Ydata',Ylim,'Linestyle','--',...
       'color',[0 0 0],'Erasemode','xor');

     else
      set(hl,'Xdata',[np np],'Ydata',Ylim,'color','r');
     end
   end

elseif strcmp(button,'right') %---rigt button was clicked

   if TWOFILES==1
      if TOP==1
          if doit1==1 | frst0==1
```

```
            hwi(1,1)=line('Xdata',[np+4 np+4],'Ydata',Ylim,...%
'Linestyle','--',...
                'color',[0 0 0],'Erasemode','xor');
            hwi(1,2)=line('Xdata',[np np],'Ydata',Ylim,...
%'Linestyle','--',...
                'color','m','Erasemode','xor');
           frst0=0; doit1=0;
        else
           set(hwi(1,2),'Xdata',[np np],'Ydata',Ylim,'color','m');
        end
     else

        if doit0==1 | frst1==1
            hwi(2,1)=line('Xdata',[np+4 np+4],'Ydata',Ylim,...%
'Linestyle','--',...
                'color',[0 0 0],'Erasemode','xor');
            hwi(2,2)=line('Xdata',[np np],'Ydata',Ylim,...
%'Linestyle','--',...
                'color','m','Erasemode','xor');
           frst1=0; doit0=0;
        else
           set(hwi(2,2),'Xdata',[np np],'Ydata',Ylim,'color','m');
        end
     end
   else

   invalid=0;
   eval('findobj(hr);','invalid=1;');
   if doit==1 | invalid==1
     hl=line('Xdata',[np+4 np+4],'Ydata',Ylim,'Linestyle','-',...
         'color',[0 0 0],'Erasemode','xor');
    hr=line('Xdata',[np np],'Ydata',Ylim,'Linestyle','--',...
      'color','m','Erasemode','xor');
   else
     set(hr,'Xdata',[np np],'Ydata',Ylim,'color','m');
   end
      end
     end
```

```
function engy


global nChannels filename filterA filterB Srate eFig
global HDRSIZE S0 S1 ratePps En Be bl al WAV1 n_Secs En2 Be2
global filename2 TWOFILES n_Secs2 Srate2 TOP HDRSIZE2
global ftype ftype2 bpsa bpsa2

pos = get(0, 'screensize'); % get the screensize
sWi = pos(3);
sHe = pos(4);



rate=200;

WIDTH   =round(0.9375*sWi);
HEIGHT = round(0.5*sHe);
LEFT    =round(0.025*sWi);
BOTTOM  =20;

if TWOFILES==1 & TOP==1
 fname=filename2;
 hdr=HDRSIZE2;
 n_samples=En2-Be2; %round(n_Secs2*Srate2)
 strt=Be2;
 offSet=strt*bpsa2+hdr;
 ftp=ftype2;
else
 fname=filename;
 hdr=HDRSIZE;
 n_samples=En-Be; %round(n_Secs*Srate)
 strt=Be;
 offSet=strt*bpsa+hdr;
 ftp=ftype;
end

fp = fopen(fname,'r');

if fp <=0
   disp('ERROR! File not found..')
   return;
end



   st = fseek(fp,offSet,'bof');
        x=zeros(1,n_samples);
   x = fread(fp,n_samples,ftp);

   fclose(fp);
%---- Do some error checking on the signal level ---
meen=mean(x);
x= x - meen; %----------remove the DC bias---


nmF='Energy Plot';
```

```
if (isempty(eFig))
   eFig =  figure('Units', 'Pixels', 'Position', [LEFT BOTTOM WIDTH
HEIGHT],...
   'Pointer','crosshair',...
   'Resize','on','Name',nmF,'NumberTitle','Off');

else
   figure(eFig);
   set(eFig,'Name',nmF);
end



fRate=round(Srate/rate);  %--- Number of  samples in each frame

nFrames=floor(n_samples/fRate);


zcnt=zeros(1,nFrames);
ampl=zeros(1,nFrames);
k=1;
for t=1:nFrames
      yin=x(k:k+fRate-1);
      eng(t)=10*log10(norm(yin,2));
      zc=0;
      x2=zeros(fRate,1);
           x2(1:fRate-1)=yin(2:fRate);
           zc=length(find((yin>0 & x2<0) | (yin<0 & x2>0))); %-- Zero
Crossings
           zcnt(t)=zc;
      %f0=pitch(yin);
      %pit(t)=f0;

k=k+fRate;

end


%--- Plot energy, zero crossings, and waveform -----------
subplot(2,1,1),plot(eng);
set(gca,'Xlim',[0 nFrames-1]);
ylabel('dB');
%subplot(3,1,2),plot(pit);
%set(gca,'Xlim',[0 nFrames-1]);
%ylabel('#zc');
subplot(2,1,2),plot(x);
set(gca,'Xlim',[0 n_samples-1]);
ylabel('Ampl.');
xlabel('Samples');
```

```
function estf0(type,action)


global nChannels filename Srate eFig
global HDRSIZE S0 S1 ratePps En Be  n_Secs En2 Be2
global filename2 TWOFILES n_Secs2 Srate2 TOP HDRSIZE2
global bpsa bpsa2 ftype ftype2 bf0 af0 SrateChange


if nargin==1, action='plot'; end;

pos = get(0, 'screensize'); % get the screensize
sWi = pos(3);
sHe = pos(4);


WIDTH   =round(0.9375*sWi);
HEIGHT = round(0.5*sHe);
LEFT    =round(0.025*sWi);
BOTTOM  =20;

if TWOFILES==1 & TOP==1
 fname=filename2;
 hdr=HDRSIZE2;
 n_samples=En2-Be2;
 strt=Be2;
 sr=Srate2;
 offSet = strt*bpsa2+hdr;
 ftp=ftype2;
else
 fname=filename;
 hdr=HDRSIZE;
 n_samples=En-Be; %round(n_Secs*Srate)
 strt=Be;
 sr=Srate;
 offSet = strt*bpsa+hdr;
 ftp=ftype;
end

fp = fopen(fname,'r');

if fp <=0
   disp('ERROR! File not found..')
   return;
end


   st = fseek(fp,offSet,'bof');
        x=zeros(1,n_samples);
   x = fread(fp,n_samples,ftp);

   fclose(fp);

%-----select a method for pitch detection ----------
%
if strcmp(type,'cepstrum')
    CEPSTRUM=1;
 else
   CEPSTRUM=0;
   if isempty(bf0) | SrateChange==1
      [bf0,af0]=butter(4,900/(sr/2));
```

```
   end
 end;


%---- Do some error checking on the signal level ---
meen=mean(x);
x= x - meen; %----------remove the DC bias---


nmF='F0 Contour Plot';


if isempty(eFig)
   eFig =  figure('Units', 'Pixels', 'Position', [LEFT BOTTOM WIDTH
HEIGHT],...
   'Pointer','crosshair',...
   'Resize','on','Name',nmF,'NumberTitle','Off');
   crM=1;
else
   figure(eFig);
   set(eFig,'Name',nmF);
   crM=0;
end


updRate=floor(20*sr/1000);  %-- Update every 20 msec
if CEPSTRUM==1
 fRate=floor(40*sr/1000);  % -- Use a 40 msec segment
else
 fRate=floor(30*sr/1000);  % -- Use a 30 msec segment
end
nFrames=floor(n_samples/updRate)-1;

% --- Do some error checking in the length of the region selected ---
------
%
if nFrames<1
  errordlg('Region selected was too small..','ERROR in pitch
estimation','on');
  delete(eFig);
  eFig=[];
  return;
end
k=1;
f01=zeros(1,nFrames);
f0=zeros(1,nFrames);

m=1;
avgF0=0;
for t=1:nFrames
      yin=x(k:k+fRate-1);

      if CEPSTRUM==1
         a=pitch(fRate,sr,yin); % Use the cepstrum method
      else
            a=pitchaut(fRate,sr,yin); % Use the autocorr. method
      end
      f0(t)=a;
      if t>2 & nFrames>3 %--do some median filtering
        z=f0(t-2:t);
        md=median(z);
```

```
        f01(t-2)=md;
        if md > 0
          avgF0=avgF0+md;
          m=m+1;
         end
      elseif nFrames<=3
        f01(t)=a;
        avgF0=avgF0+a;
        m=m+1;
      end


k=k+updRate;

end


if m==1, avgF0=0; else, avgF0=avgF0/(m-1); end;

str= sprintf('Average F0=%5.2f Hz',avgF0);

set(eFig,'Name',str);

%---------------Save F0 values in an ASCII file ---------------
%
if strcmp(action,'save')

  [pth,f0name] = dlgopen('save',['*.pit']);
  if ((~isstr(f0name)) | ~min(size(f0name))), return; end
  fname1=[pth,f0name];
  fpout=fopen(fname1,'w');
  upd=round(1000*updRate/sr);

  fprintf(fpout,'t(msec)  F0(Hz)\n-------+------\n');
  k=1;
  for t=upd/2:upd:nFrames*upd
   fprintf(fpout,'%4.2f     %5.2f\n',t,f01(k));
  k=k+1;
  end

  fclose(fpout);
end


%-------------- Plot F0 contour  --------------------------------
%
xt=1:nFrames;
xt=20*xt;

if ~isempty(xt) & nFrames>1
  plot(xt,f01)
  axis([xt(1) xt(nFrames) 0 max(f01)+50]);
else
  if nFrames==1, plot(f01,'o'); else, plot(f01); end;
end

ylabel('Hz');
xlabel('Time (msecs)');
```

```
%--------------Create the 'close' button' ---------------

xywh = get(eFig, 'Position');

% Buttons.
left = 6;
wide = 60;
top  = xywh(4) - 10;
high = 22;
inc  = high + 8;

uicontrol('Style', 'PushB', 'Callback', 'closem(''efig'')', ...
          'HorizontalAlign','center', 'String', 'Close', ...
          'Position', [left top-high wide high]);

if crM==1
 if strcmp(type,'cepstrum'), str='estf0(''cepstrum'',''save'')';
 else str='estf0(''autoc'',''save'')'; end;
 uimenu('Label','Save pitch values','Callback',str);
end




function [n,t]=frq2midi(f)

n=(69+12*log(abs(f)/440)/log(2));
if nargout > 1
  m=round(n(:));
  o=floor(m/12)-1;
  m=m-12*o+6*sign(f(:))-5;
  a=('CDDEEFGGAABBCCDDEFFGGAAB')';
  b=(' - -  - - -  # #  # # # ')';
  t=setstr([a(m) mod(o,10)+'0' b(m)]);

end
```

```
function [HDRSIZE, Srate1, bps, filet] =  gethdr(fp,ext)


bps=2; filet='short';  % bytes per samples

if strcmp(ext,'ils') %---- ILS ---
   HDRSIZE=512;   % Header size in bytes
   [xx, cnt] = fread(fp,HDRSIZE/2,'short');
   if cnt < HDRSIZE/2
     fprintf('ERROR in reading file %s (possibly a zero-bytes
file)\n',filename);
     return;
   end
   Srate1 = xx(62);

elseif strcmp(ext,'adf') %-- ADF format -CSRE ----
   HDRSIZE=512;
   csre=fread(fp,8,'char');
   setstr(csre);
   numsam=fread(fp,1,'int32');
   xx=fread(fp,5,'short'); bits=xx(4); cd=xx(5);
   srt=fread(fp,1,'float');
   Srate1=1000*srt;
   fseek(fp,HDRSIZE,'bof');

elseif strcmp(ext,'adc')  %-- old TIMIT format----
   xx=fread(fp,3,'short');  % xx(1)= hdrsize in words
              % xx(2)= version
              % xx(3)= number of channels
   if xx(2)>10 | xx(3)>3
     error('ERROR reading TIMIT header..');
   end
   HDRSIZE=xx(1)*2; % header size in bytes
   if HDRSIZE<0 | HDRSIZE>20
      error('ERROR reading TIMIT header..');
   end
   srt=fread(fp,1,'short'); % sampling period in 1/4 microsecs
   nsam=fread(fp,1,'int32'); % number of samples
   if srt>0
     Srate1=4*10^6/srt;
   else
    error('ERROR reading TIMIT header..');
   end


 elseif strcmp(ext,'wav')  % -- MS Windows WAV files or TIMIT files
   xx=fread(fp,4,'char');
   riff=(setstr(xx))';
      if strcmp(riff,'RIFF') ~= 1    % maybe a SPHERE file
        fseek(fp,0,'bof');
        nist=fscanf(fp,'%s',1);
        if strcmp(nist(1:4),'NIST') ==0
          error('Error reading voice file header, not a MS Windows
WAV nor TIMIT ..');
          return;
        end

        nn=fscanf(fp,'%d',1);
        for i=1:5
          X=fscanf(fp,'%s %s %s',3);
        end
```

```
        X=fscanf(fp,'%s %s',2);
        Srate1=fscanf(fp,'%d',1);

        for i=1:2
          X=fscanf(fp,'%s %s %s',3);
        end

        X=fscanf(fp,'%s %s',2);
        bps=fscanf(fp,'%d',1);

        fseek(fp,1024,'bof'); % skip the header
        HDRSIZE=1024;    % Header Size.
   else                 %------ maybe a MS windows .wav file
        numsa=fread(fp,1,'int32');  % length in bytes of rest of file
        xx=fread(fp,7,'short');
        chan=fread(fp,1,'short');
        Srate1=fread(fp,1,'int32');  % -- Get the sampling frequency-
------
        datasec=fread(fp,2,'short');
        bytesamp=fread(fp,1,'short');
        if bytesamp==1, bps=1; filet='char'; end
        bitssamp=fread(fp,1,'short');
        xx=fread(fp,4,'char');
        if strcmp('data',setstr(xx)')==1 |
strcmp('fact',setstr(xx)')==1
            if strcmp('fact',setstr(xx)')==1,
nsam=fread(fp,3,'int32'); end;

         else
           fprintf('ERROR! Not a valid wav file. Missing data.\n');
        end
        nsam=fread(fp,1,'int32'); % Number of samples in bytes
        HDRSIZE=ftell(fp);

 end


elseif strcmp(ext,'voc') %-------- Creative Lab's VOC files-------

   cint=[1 2^8]';
   xx=fread(fp,20,'char');
   vf=setstr(xx');
   if strcmp(vf(1:19),'Creative Voice File')==0
     disp('ERROR! File is not a proper VOC file');
     return;
   end
   xoff=fread(fp,1,'short');
   x1=fread(fp,2,'uchar'); % frmt=sum(x1 .* cint)
   x1=fread(fp,2,'uchar'); % vid=sum(x1 .* cint)
   blk_type=fread(fp,1,'uchar');
   x3=fread(fp,3,'uchar');
   blen=sum(x3 .* [cint; 2^16])-2;
   if blk_type==1  % Voice data
     tconst=fread(fp,1,'uchar');
     Srate1=10^6/(256-tconst);
     pck=fread(fp,1,'uchar');
     if pck ~=0, disp('WARNING! Voice data is not in 8-bit sample
format'); end;
     fread(fp,1,'uchar');
```

```
   else
     Srate1=11125;
     disp('WARNING! Not a voice file.');
   end
   HDRSIZE=ftell(fp);
   WAV=1; bps=1; filet='char';
else


   fprintf('ERROR! Unknown file extension..\n');
    Srate1=0;
    convert; % convert file format

 end




function mclick


global fno Srate n_Secs AXISLOC h1T h2T S0 cAxes S1
global Be En TIME smspEc cAxes TWOFILES TOP Be2 En2
global Srate2 n_Secs2 tpc boc smp frq hl hr doit hll
global upFreq upFreq2 htop hbot lastClick hwi frst1 frst0
global doit0 doit1 SHOW_CRS SHOW_CHN posi

xv = get(fno,'CurrentPoint');  % get cursor coordinates in pixels
xp = xv(1);
yp = xv(2);


if TWOFILES==1
   set(fno,'Units','Normal');
   xv2 = get(fno,'CurrentPoint');
   set(fno,'Units','Pixels');
   if xv2(2)>0.5
      TOP=1;
      set(tpc,'String',' ','BackGroundColor','y');
      set(boc,'String',' ','BackGroundColor',[0 0 0]);
   else
      TOP=0;
      set(boc,'String',' ','BackGroundColor','y');
      set(tpc,'String',' ','BackGroundColor',[0 0 0]);
   end
   if isempty(posi), posi=zeros(2,2); end;
end

typ = get(fno,'SelectionType');   % get mouse button (normal= left,
             % extend=middle, alt=right)


FigXY  = get(fno,'Position');

%
% The offset Be is added to account for zoomed displays
%
if TWOFILES==1
  if TOP==1
   Sample = Be2+round((xp-AXISLOC(1))*(En2-Be2)/AXISLOC(3));
   yoffs=0.55*FigXY(4);
```

```
   zhei =0.34*FigXY(4);
   Freq   = round((yp-yoffs)*Srate2*0.5*upFreq2/zhei);
   stime = Sample*1000/Srate2;
   b= n_Secs2*Srate2;
   hsra=0.5*Srate2;


  else
   Sample = Be+round((xp-AXISLOC(1))*(En-Be)/AXISLOC(3));
   yoffs=0.11*FigXY(4);
   zhei =0.34*FigXY(4);
   Freq   = round((yp-yoffs)*Srate*0.5*upFreq/zhei);
   stime = Sample*1000/Srate;
   b= n_Secs*Srate;
   hsra=0.5*Srate;
  end

else
  Sample = Be+round((xp-AXISLOC(1))*(En-Be)/AXISLOC(3));
  Freq   = round((yp-AXISLOC(2))*Srate*0.5*upFreq/AXISLOC(4));
  b= n_Secs*Srate;
  hsra=0.5*Srate;
  stime = Sample*1000/Srate;
end

%---- Check to see if cursor is within window box -------

IN_WINDOW=0;
if TWOFILES==1 & TOP==1
 if ( Sample >= Be2 & Freq > 0.0 & Freq <= hsra & Sample < En2)
   if Sample < b
    IN_WINDOW=1;
   end
 end
else
 if ( Sample >= Be & Freq > 0.0 & Freq <= hsra & Sample < En)
   if Sample < b
    IN_WINDOW=1;
   end
 end
end

if  IN_WINDOW == 1


     tStr = sprintf('t= %d (ms)',round(stime));
     fStr = sprintf('f= %d Hz',Freq);

     top=FigXY(2)+100;
     left=FigXY(1)+10;


  %---------------- Draw the cursor lines-------------------

    if SHOW_CRS==1
      if TWOFILES==0, axes(cAxes); end

    np=(Sample-Be)*1000/Srate;

    if TWOFILES==1
```

```
    if TOP==1
    lastClick=1;
    axes(htop);
         Ylim=get(gca,'Ylim');
    np=(Sample-Be2)*1000/Srate2;

    else
    lastClick=0;
    axes(hbot);
    Ylim=get(gca,'Ylim');
    end
   if lastClick==1  & TOP==0, frst1=1; end;
   if lastClick==0  & TOP==1, frst0=1; end;
   if isempty(frst0), frst0=1; end;
   if  isempty(frst1), frst1=1; end;
 else
    Ylim=get(gca,'YLim');
 end

   if (strcmp(typ,'normal')==1) % ==== Left button was pushed ===
 if TWOFILES==1
   if TOP==1
     if  doit1==1, posi(1,2)=En2; end;
     S1=posi(1,2); posi(1,1)=Sample;
   else
     if doit0==1,  posi(2,2)=En; end;
     S1=posi(2,2); posi(2,1)=Sample;
   end
 end
 crsrline(np,Ylim,'left');  % Draw the cursor line

   elseif (strcmp(typ,'extend')==1) %===== Right Button ======
 if TWOFILES==1
   if TOP==1
     if  doit1==1,  posi(1,1)=Be2; end;
     S0=posi(1,1);  posi(1,2)=Sample;
   else
     if doit0==1,  posi(2,1)=Be; end;
     S0=posi(2,1); posi(2,2)=Sample;
   end
 end
 crsrline(np,Ylim,'right');

   end

end%------------------end of IF show cursor lines-----

 %-----------plot the sample number and/or freq -----
 set(smp,'String',tStr,'ForegroundColor','w','BackgroundColor',[0 0
0]);

   if (TIME==0)

 set(frq,'String',fStr,'ForegroundColor','w','BackgroundColor',[0 0
0]);
   end

  if (strcmp(typ,'normal')==1 | strcmp(typ,'alt')==1)
    S0 = Sample;
      if SHOW_CHN==1, pllpc(Sample); end;
      if ~isempty(smspEc)
```

```
      if smspEc==1
       smspspec;
      end
        end

    else %---- Right button was pressed (in SUN its middle)
    S1 = Sample;
    end


end

doit=0;




function [f0] = pitchaut(len,sr,xin)

global bf0 af0



xin=filter(bf0,af0,xin);  % LPF at 900 Hz

i13=len/3;
maxi1=max(abs(xin(1:i13)));

i23=2*len/3;
maxi2=max(abs(xin(i23:len)));

if maxi1>maxi2, CL=0.68*maxi2; else CL= 0.68*maxi1; end;

%----------Center clip waveform, and compute the autocorrelation  ---
--------------------

clip=zeros(len,1);
ind1=find(xin>=CL);
clip(ind1)=xin(ind1)-CL;

ind2=find(xin <= -CL);
clip(ind2)=xin(ind2)+CL;

engy=norm(clip,2)^2;

RR=xcorr(clip);
m=len;


%-------Find the max autocorrelation in the range 60 <= f <= 320 Hz -
-----------
%
LF=floor(sr/320);
HF=floor(sr/60);

Rxx=abs(RR(m+LF:m+HF));
[rmax, imax]= max(Rxx);

imax=imax+LF;
```

```
f0=sr/imax;


%------------ Check max RR against V/UV threshold -------------------
---------
silence=0.4*engy;


if (rmax > silence)  & (f0 > 60) & (f0 <=320)
 f0=sr/imax;
else % -- its unvoiced segment ---------
 f0=0;
end
```

```
function playf(cmd)

%

global filename Srate
global HDRSIZE S0 S1  Be En WAV1 Srate2 HDRSIZE2 Be2 En2
global TWOFILES TOP filename2 wav
global bpsa bpsa2 ftype ftype2 VOL_MAX VOL_NORM


onebyte=WAV1;

   if TWOFILES==1

      if strcmp(cmd,'sel') %-- play only a selected region
            if S1>S0
               n_samples=S1-S0;
         else
               errordlg('Invalid selection','ERROR in play','on');
               return;
             end
            bes=S0;
      else %--play the whole file
            if TOP==1, n_samples=En2-Be2; bes=Be2;
         else        n_samples=En-Be; bes=Be;
         end
      end

      if TOP==1
        offSet=bes*bpsa2+HDRSIZE2;
        fname=filename2;
        ftp=ftype2;
      else
        offSet=bes*bpsa+HDRSIZE;
        fname=filename;
        ftp=ftype;
      end
```

```
   fp = fopen(fname,'r');

   if fp <=0
     disp('ERROR! File not found..')
     return;
   end


   st = fseek(fp,offSet,'bof');
   if st<0, disp('FAILURE in reading'); return; end;
        x=zeros(1,n_samples);
   [x,cnt] = fread(fp,n_samples,ftp);
   if cnt ~= n_samples,
     disp('ERROR in reading in playf.m');
   end

   fclose(fp);
   else  % ---------- single window -------------------
   fp = fopen(filename,'r');

   if fp <=0
     disp('ERROR! File not found..')
     return;
   end

   if strcmp(cmd,'sel')
      if S1>S0
        n_samples=S1-S0;
      else
        errordlg('Invalid selection','ERROR in play','on');
        fclose(fp);
        return;
      end
      bes=S0;
   else %--play the whole file or whole window
      n_samples=En-Be;
      bes=Be;
   end

   offSet = bes*bpsa+HDRSIZE;
   st = fseek(fp,offSet,'bof');
        x=zeros(1,n_samples);
   x = fread(fp,n_samples,ftype);

   fclose(fp);

  end


%----------remove the DC bias----
meen=mean(x);
x= x - meen;

if VOL_NORM==1 % normalize volume - autoscale

  mx=max(abs(x));
  mx=mx*(1+VOL_MAX);

elseif VOL_NORM==2 % have the option to scale
```

```
  maxx=max(abs(x));
  mx=32600;
  new_max=VOL_MAX*mx;
  x=x*new_max/maxx;

elseif VOL_NORM==3 % -- absolute scale

  mx=32600;

end

v=[-mx-10 mx+10];


if TWOFILES==1
   if TOP==1, sr=Srate2; else sr=Srate; end;
else
 sr=Srate;
end


%---Play the speech file -----

saxis(v);
sound(x,sr);



function showpt



global smp


if ~exist('smp'), return; end;

global fno Srate n_Secs AXISLOC
global Be En TIME  TWOFILES TOP Be2 En2
global Srate2 n_Secs2 tpc boc  frq
global upFreq upFreq2


xv = get(fno,'CurrentPoint');  % get cursor coordinates in pixels
xp = xv(1);
yp = xv(2);

FigXY  = get(fno,'Position');

%
% The offset Be is added to account for zoomed displays
%
in_win=0;

if TWOFILES==1
   set(fno,'Units','Normal');
   xv2 = get(fno,'CurrentPoint');
   set(fno,'Units','Pixels');
   if xv2(2)>0.5
      top_win=1;
```

```
   else
      top_win=0;
   end
end


if TWOFILES==1
  if top_win==1
   Sample = Be2+round((xp-AXISLOC(1))*(En2-Be2)/AXISLOC(3));
   yoffs=0.55*FigXY(4);
   zhei =0.34*FigXY(4);
   Freq   = round((yp-yoffs)*Srate2*0.5*upFreq2/zhei);
   stime = Sample*1000/Srate2;
   b= n_Secs2*Srate2;
   hsra=0.5*upFreq2*Srate2;
   if Sample>=0 & Sample<=b, in_win=1; end;

  else
   Sample = Be+round((xp-AXISLOC(1))*(En-Be)/AXISLOC(3));
   yoffs=0.11*FigXY(4);
   zhei =0.34*FigXY(4);
   Freq   = round((yp-yoffs)*Srate*0.5*upFreq/zhei);
   stime = Sample*1000/Srate;
   b= n_Secs*Srate;
   hsra=0.5*upFreq*Srate;
   if Sample>=0 & Sample<=b, in_win=1; end;
  end

else
  Sample = Be+round((xp-AXISLOC(1))*(En-Be)/AXISLOC(3));
  Freq   = round((yp-AXISLOC(2))*Srate*0.5*upFreq/AXISLOC(4));
  b= n_Secs*Srate;
  hsra=0.5*upFreq*Srate;
  stime = Sample*1000/Srate;

  if Sample>=0 & Sample<=b, in_win=1; end;
end



 if in_win==1

     tStr = sprintf('t= %4.1f (ms)',(stime));
     fStr = sprintf('f= %d Hz',Freq);


      %-----------plot the sample number and/or freq -----
     set(smp,'String',tStr,'ForegroundColor','w','BackgroundColor',[0
0 0]);
     if TIME==0
    if Freq>0 & Freq<=hsra

set(frq,'String',fStr,'ForegroundColor','w','BackgroundColor',[0 0
0]);
    end
     end


end
```

```
function zoomi(action)


global S0 S1 filename filename2 Srate fno fftSize n_Secs
global HDRSIZE cAxes En Be TIME WAV1 CLR TWOFILES En2 Be2 TOP
global HDRSIZE2 n_Secs2 Srate2 sli wav
global hl hr doit specto1 upFreq upFreq2
global hbot htop doit0 doit1 frst0 frst1
global bpsa bpsa2 ftype ftype2 LD_LABELS

onebyte=0;

if (S1 > S0) | strcmp(action,'out')==1
   figure(fno);
   if TWOFILES==0, axes(cAxes); end
   doit=1; %used in mclick.m
   if TWOFILES==1 & TOP==1
     En2=S1; Be2=S0;
   else
     En=S1; Be=S0;
   end


   if strcmp(action,'out')==1
       set(sli,'Value',0);
       if TWOFILES==1 & TOP==1
      S0=0;
      S1=round(n_Secs2*Srate2);
      En2=S1;
      Be2=S0;
      frst0=1; doit1=1;
        else
      S0=0;
      S1=round(n_Secs*Srate);
      En=S1;
      Be=S0;
      doit0=1; % used in mouse-click routine
      frst1=1;
        end
   end

   if LD_LABELS==1  % --- in case labels have been drawn ------------
-----

      if strcmp(action,'in'),      label('loadzm');
      elseif strcmp(action,'out'), label('load');
      end;
   end

   if TWOFILES==1 & TOP==1
     fname=filename2;
     offSet=S0*bpsa2+HDRSIZE2;
     frst0=1; doit1=1;
     ftp=ftype2;
    else
     fname=filename;
     offSet=S0*bpsa+HDRSIZE;
     doit0=1; frst1=1;
     ftp=ftype;
    end
```

```
fp = fopen(fname,'r');
if fp<=0
  disp('Could not open file in zocmi.m....');
  return;
end


nSamples=S1-S0 ;  % get the number of samples to read

st = fseek(fp,offSet,'bof');
     inp=zeros(1,nSamples);
[inp,cnt] = fread(fp,nSamples,ftp);
if cnt ~= nSamples,
  fclose(fp);
  errordlg('ERROR reading file ..','ERROR','on');
  return;
end
fclose(fp);


%----------- Remove the DC bias------------
meen=mean(inp);
inp=inp-meen;
if norm(inp,2) < 1.0e-7
  if meen < 1.0e-3
   meen=1.0e-3;
      end
      inp=inp+meen;
end


if (TIME)

  if TWOFILES==1
   if TOP==1
         Et=1000*nSamples/Srate2;
      %if isempty(htop)
        subplot(2,1,1),
        set(gca,'Position',[0.2 0.55 0.775 0.34]);

        htop=gca;
      %else
      %  axes(htop);
      %end
      plot(0:1000/Srate2:(Et-1000/Srate2),inp,'y');
      ylabel('Amplitude');%,'FontSize',9);
      %set(gca,'Units','points','FontSize',9);
      %set(gca,'FontSize',9);
      axis([0 Et min(inp)-200 max(inp)+200]);
      set(gca,'Xcolor','w'); set(gca,'Ycolor','w');
      set(gca,'Color','k');
   else


      Et=1000*nSamples/Srate;
      %if isempty(hbot)
        subplot(2,1,2),
        set(gca,'Position',[0.2 0.11 0.775 0.34]);
        hbot=gca;
      %else
```

```
        % axes(hbot);
       %end

       plot(0:1000/Srate:(Et-1000/Srate),inp,'y');
       xlabel('Time (msecs)');%,'FontSize',9);
       ylabel('Amplitude');%,'FontSize',9);
       %set(gca,'Units','points','FontSize',9);
       %set(gca,'FontSize',9);
       axis([0 Et min(inp)-200 max(inp)+200]);
       set(gca,'Xcolor','w'); set(gca,'Ycolor','w');
       set(gca,'Color','k');
    end

   else

    Et=1000*nSamples/Srate;
    xax=0:1000/Srate:(Et-1000/Srate);
    plot(xax,inp)

    axis([0 Et min(inp)-200 max(inp)+200]);
    xlabel('Time (msecs)');%,'FontSize',9);
    ylabel('Amplitude'); %'FontSize',9);
    set(gca,'Xcolor','w'); set(gca,'Ycolor','w');
    set(gca,'Color','k');
    %set(gca,'Units','points','FontSize',9);
    end
   else % ========== Spectrogram=================
yLab = [0.5*Srate*upFreq/1000 0];
ex=(En-Be)*1000/Srate;
xLab = [0 ex];

%--- define colormap

if CLR == 0
   nMap=length(colormap);
   x1=gray(nMap);
   x4=flipud(x1);
   colormap(x4);

else
   colormap('jet');
end
if nSamples<1000
  ovr=32;
  ns=4;
elseif nSamples>1000 & nSamples<4000
  ovr=10;
  ns=3;
else
  ovr=4;
  ns=3;
end
if TWOFILES==1
  if TOP==1
   yLab = [0.5*Srate2*upFreq2/1000 0];
   ex=(En2-Be2)*1000/Srate2;
   xLab = [0 ex];
   %if isempty(htop)
     subplot(2,1,1), cla reset; hold off;
     set(gca,'Position',[0.2 0.55 0.775 0.34]);
```

```
        htop=gca;
      %else
       % axes(htop);
      %end

      cIm = image(xLab,yLab,spectrg(inp,128,ovr,ns,upFreq2));
              axis xy
      ylabel('Freq. (KHz)');
      set(gca,'Xcolor','w'); set(gca,'Ycolor','w');
      get(gca,'YDir');
      %set(gca,'Units','points','FontSize',9);
    else
      %if isempty(hbot)
         subplot(2,1,2), cla reset; hold off;
         set(gca,'Position',[0.2 0.11 0.775 0.34]);
         hbot=gca;
      %else
       %  axes(hbot);
      %end

      cIm = image(xLab,yLab,spectrg(inp,128,ovr,ns,upFreq));
      axis xy
      xlabel('Time (msecs)');%;,'FontSize',9);
      ylabel('Freq. (KHz)');%,'FontSize',9);
      %set(gca,'Units','points','FontSize',9);
      set(gca,'Xcolor','w'); set(gca,'Ycolor','w');

    end
  else %=========== Single display =====================


       specto1=spectrg(inp,128,ovr,ns,upFreq);

       cIm = image(xLab,yLab,specto1);
       axis xy
       xlabel('Time (msecs)');%,'FontSize',9);
       ylabel('Freq. (KHz)');%,'FontSize',9);
       %set(gca,'Units','points','FontSize',9);
       set(gca,'Xcolor','w'); set(gca,'Ycolor','w');

  end

   end


else
   errordlg('Invalid selection. Right mark is less than left
mark','ERROR in zoom','on');

end
```

เอกสารอ้างอิง

1. Craig Anderton , MIDI for Musicians . (Howard Jones ed.) Amsco publications, New York , 1986.
2. Dominic Milano , MIND OVER MIDI , GPI Publications , Califomia, 1987
3. Richard E.Berg , David G.Stork , The Physics of Sound , Prentice-Hall International Editions , New Jersey , 1982.
4. Charles Dodge ,Thomas A. Jerse , COMPUTER MUSIC SYNTHESIS , COMPOSITION, AND PERFORMANCE , Schirmer Books A Division of Mcmillan , Inc . NEW YORK, 1985.

ประวัติผู้เขียน

นายมาโนชญ์ กาญจนเพ็ญ เกิดวันที่ 30 มิถุนายน พ.ศ. 2521 สำเร็จการศึกษาระดับมัธยมตอนต้นจากโรงเรียนศรีปทุมพิทยาคาร และ ระดับมัธยมศึกษาตอนปลายจากโรงเรียนเบ็ญจะมะมหาราช แล้วเข้าศึกษาต่อในระดับปริญญาตรีสาขาฟิสิกส์ประยุกต์ คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง และสำเร็จการศึกษาในปีการศึกษา 2542