การจดจำรูปแบบตัวพิมพ์โน้ตดนตรี

MUSICAL NOTE RECOGNITION

นาย วิรัตน์ สิงหกำ
นาย ธวัชชัย บูรณะวัฒนาศิลป์
นาย อภิเศก สร้อยน้ำทิพย์

ปริญญานิพนธ์เล่มนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต
สาขาวิศวกรรมการวัดคุม
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2542

ปีการศึกษา 2542

การจดจำรูปแบบตัวพิมพ์โน้ตดนตรี

Musical Note Recognition

นาย วิรัตน์ สิงหกำ

นาย ธวัชชัย บูรณะวัฒนาศิลป์

นาย อภิเศก สร้อยน้ำทิพย์

อาจารย์ที่ปรึกษา

รศ. เกษตร์ ศิริสันติสัมฤทธิ์

ปริญญานิพนธ์ปีการศึกษา 2542

ภาควิชาเทคโนโลยีการวัดคุมทางอุตสาหกรรม

สาขาวิชาวิศวกรรมการวัดคุม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การจดจำรูปแบบตัวพิมพ์โน้ตดนตรี

Musical Note Recognition

ผู้จัดทำ

1. นาย วิรัตน์ สิงหกำ รหัสประจำตัว 40012100
2. นาย ธวัชชัย บูรณะวัฒนาศิลป์ รหัสประจำตัว 40013407
3. นาย อภิเศก สร้อยน้ำทิพย์ รหัสประจำตัว 40013438

.....อาจารย์ที่ปรึกษา

(รศ. เกษตร์ ศิริสันติสัมฤทธิ์)

หัวข้อปริญญานิพนธ์	:	การจดจำรูปแบบตัวพิมพ์โน้ตดนตรี
โดย	:	นาย วิรัตน์ สิงหกำ
		นาย ธวัชชัย บูรณะวัฒนาศิลป์
		นาย อภิเศก สร้อยน้ำทิพย์
ภาควิชา	:	เทคโนโลยีการวัดคุมทางอุตสาหกรรม
อาจารย์ที่ปรึกษา	:	รศ. เกษตร์ ศิริสันติสัมฤทธิ์
ปีการศึกษา	:	2542

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้ ได้รายงานถึงวิธีการจดจำรูปแบบตัวพิมพ์โน้ตดนตรีโดยคอมพิวเตอร์ ซึ่งคอมพิวเตอร์จะรับภาพตัวพิมพ์โน้ตดนตรีจากเครื่องสแกนเนอร์แล้วเก็บเป็นไฟล์รูปภาพ Bitmap และใช้วิธีการประมวลผลภาพกับการจดจำรูปแบบมาวิเคราะห์ภาพตัวพิมพ์โน้ตดนตรีเพื่อพิจารณาถึงโครงร่างของตัวโน้ต ด้วยการแยกโน้ตออกจากบรรทัด 5 เส้นจากนั้นทำการแยกโน้ตแต่ละตัวออกจากกัน โดยวิธีการตีกรอบตัวโน้ตแต่ละตัวข้อมูลภาพของตัวโน้ตจะนำไปทำให้หลือแต่โครงร่างเพื่อนำโครงร่างดังกล่าวไปหาลักษณะเด่นของตัวโน้ตอันได้แก่ จำนวนและตำแหน่งของจุดตัด จุดแยก และจุดปลาย ของตัวโน้ตแต่ละตัว แล้วตีความหมายให้ทราบถึงระดับเสียงและความยาวเสียงของตัวโน้ต จากนั้นนำไปสร้างเป็นฐานข้อมูลเพื่อแปลงให้เป็นรหัสมิดี้ส่งไปควบคุมเครื่องคนตรีประเภทมิดี้ ต่อไป

Project Report Title	:	Musical Note Recognition
By	:	Mr. Wirat Singhakam
		Mr. Tawatchai Buranawattanasin
		Mr. Abhisake Soynamthip
Department	:	Industrial Instrument Technology
Project Report	:	Assoc. Prof. Kaset Sirisantisamrid
Academic	:	1999

Abstract

The thesis reports the process of recognition of printed musical note by computer. The computer will receive the picture of note from scanner and save as Bitmap file format and then use image processing and pattern recognition to analyze the printed musical note. The method is used to analyze the structure of musical note by extraction the staff line then separate each of musical note is framing in order to find it's meaning. The image's data has to be reduced by thinning method to find the characteristic of musical note such as number and position of cross point, branch point and end point of musical note and the interpreted into pitch and duration of musical note. After that, Those would be used to make database for convert to MIDI code in order to control MIDI instrument.

สารบัญ

บทที่1
บทนำ

ในบทนี้จะกล่าวถึงความเป็นมางานของงานวิจัย เหตุผลที่มาของงานวิจัย วัตถุประสงค์ที่ต้องการรวมไปถึงประโยชน์ที่คาดว่าจะได้รับจากงานวิจัย สุดท้ายจะบอกถึงขอบเขตและผลของงานวิจัยนี้

1.1 ความเป็นมาและปัญหา

1.1.1 ความเป็นมา

ในสภาพการณ์ปัจจุบันนั้น เทคโนโลยีต่างๆได้เข้ามามีบทบาทในชีวิตประจำวันเป็นอย่างมากและยังก้าวหน้าอย่างรวดเร็ว เพื่อตอบสนองความต้องการของมนุษย์ อย่างเช่น ดนตรีซึ่งเป็นศิลปะอันประณีตบรรจงของมนุษย์ ในอันที่จะสรรสร้าง หรือถ่ายทอดจินตนาการ ความรู้สึกนึกคิดออกมาเป็นเสียงดนตรีอันไพเราะสู่ผู้ฟัง จากความประณีตบรรจงอันนี้เป็นสิ่งสร้างสรรอันหนึ่ง ที่ทำให้มนุษย์ในยุคปัจจุบันนี้พยายามใช้เทคโนโลยีในการศึกษาค้นคว้า พัฒนา เกี่ยวกับดนตรี จากอดีตจนถึงปัจจุบันและต่อไปยังอนาคต โน้ตดนตรียังคงมีให้เห็นอยู่ทั่วไปเนื่องมาจากโน้ตดนตรียังคงเป็นมาตราฐานที่นักดนตรีใช้ในการนำโน้ตดนตรีไปเล่นเพลงตามโน้ตนั้นๆ แต่จากความต้องการที่จะให้เทคโนโลยีเข้ามามีบทบาททางด้านนี้ จึงมีการคิดค้นให้คอมพิวเตอร์สามารถเล่นบทเพลงตามโน้ตดนตรีได้ โดยผ่านอุปกรณ์ MIDI การ์ด ซึ่งเป็นอุปกรณ์ที่เพิ่มเติมเข้ามาประกอบกับเครื่องคอมพิวเตอร์

เนื่องมาจากงานวิจัยนี้เป็นผลงานที่มีผู้ทำมาแล้ว แต่ยังไม่สมบูรณ์ด้วยเหตุที่สามารถเช็คตัวโน้ตได้เพียงบางส่วนเท่านั้น งานวิจัยฉบับนี้จึงได้ทำการพัฒนาหาวิธีที่เหมาะสมเพื่อตรวจเช็คตัวโน้ตให้ได้มากขึ้น

1.1.2 ปัญหาจากงานวิจัยที่ผ่านมา

1. ไม่สามารถตรวจเช็คตัวโน้ตได้ทั้งหมด
2. ไม่สามารถตรวจเช็คตัวโน้ตที่เป็นกลุ่มได้ (ตัวเขบิ็ตติดกัน)

รูปที่ 1.1 ตัวโน้ตที่เป็นกลุ่ม

1.2 วัตถุประสงค์ของปริญญานิพนธ์

1. เพื่อศึกษาทฤษฎีต่างๆ เกี่ยวกับการประมวลผลภาพ และนำทฤษฎีต่างๆ เกี่ยวกับการประมวลผลภาพมาใช้กับงานวิจัย พิจารณาเลือกว่าควรจะใช้วิธีการใดเพื่อให้เหมาะสมกับงานวิจัยที่กำลังศึกษา หรือนำเอาทฤษฎีที่มีอยู่มาประยุกต์ใช้เพื่อความเหมาะสมต่อไป
2. เพื่อศึกษาการใช้งานโปรแกรม(Visual Basic)และสามารถเลือกใช้งาน Control, Application wizard ต่างๆ ที่มีอยู่ในโปรแกรม (Visual Basic) ได้อย่างเหมาะสม
3. เพื่อศึกษาเรื่องโน้ตดนตรี เข้าใจสัญลักษณ์และความหมายของโน้ตดนตรีแต่ละตัว เพื่อให้ทำงานวิจัยได้ถูกต้องตรงตามสัญลักษณ์และความหมายของโน้ตดนตรี
4. นำทฤษฎีต่างๆ เกี่ยวกับการประมวลผลภาพมาพัฒนางานวิจัยด้านอื่นๆ
5. ศึกษาการทำงานของ MIDI การ์ด การติดต่อสื่อสารกับ MIDI การ์ด การต่อ input output และ การสร้างไฟล์ MIDI (*.MID)

1.3 ประโยชน์ที่ได้รับ

1. ทำให้คอมพิวเตอร์สามารถรู้จำตัวโน้ตดนตรีได้
2. สามารถใช้งาน MIDI การ์ดได้
3. บุคคลทั่วไปที่มีแผ่นโน้ตดนตรี สามารถนำโปรแกรมของงานวิจัยนี้ไปใช้เพื่อฟังเพลงจากโน้ตดนตรีนั้นได้

1.4 ขอบเขตของงานวิจัย

สามารถนำภาพตัวพิมพ์โน้ตดนตรี ผ่านเครื่องสแกนเนอร์เพื่อเก็บเป็นไฟล์ *.BMP แล้วนำไฟล์ที่ได้ไปใช้ในขั้นตอนการจดจำรูปตัวพิมพ์โน้ตดนตรีเพื่อแปลงเป็นไฟล์ *.MID และสามารถเล่นไฟล์ *.MID ที่ได้ ตามโน้ตดนตรีที่ได้จากขั้นตอนการจดจำรูปตัวพิมพ์โน้ตดนตรี

บทที่ 2
ระบบมิดี้ และ โครงสร้างไฟล์มิดี้

เทคโนโลยีที่ได้รับความสนใจอย่างมากอีกชนิดหนึ่ง คือ MIDI (Musical Instrument Digital Interface) ซึ่งเป็นมาตราฐานการสื่อสารข้อมูลแบบอนุกรมเพื่อใช้สำหรับเครื่องดนตรีอิเล็กทรอนิกส์ ที่มาจากบริษัทผู้ผลิตหลายบริษัท สามารถติดต่อสื่อสารกันได้ และเพื่อให้เครื่องคอมพิวเตอร์สามารถติดต่อสื่อสารกับเครื่องดนตรีอิเล็กทรอนิกส์เหล่านั้นได้อีกด้วย ทำให้การใช้งานมีความสะดวกในการแต่งเพลงมากขึ้น หรือการใช้งานในห้องบันทึกเสียงต่างๆ และสามารถนำเพลงมาแก้ไขได้สะดวกมากขึ้น

2.1 ระบบมิดี้

2.1.1 มิดี้ คือ อะไร

มิดี้เป็นมาตราฐานการสื่อสารข้อมูล แบบอนุกรม ใช้สื่อสารระหว่างเครื่องดนตรีอิเล็กทรอนิกส์กับเครื่องคอมพิวเตอร์โดยข้อมูลที่ส่งไปนั้นเป็นข้อมูลแบบดิจิตอล เครื่องดนตรีเหล่านี้ได้แก่ คีย์บอร์ด เครื่องเป่าอิเล็กทรอนิกส์ กีตาร์ มิดี้ดรัมแมชีน และอุปกรณ์ที่บ่งบอกไว้ว่าสามารถสื่อสารข้อมูลด้วยระบบมิดี้ได้

เราจะเรียกอุปกรณ์หรือเครื่องดนตรีที่สามารถสื่อสารทางมิดี้ว่า midi device หรือ อุปกรณ์มิดี้ โดยใช้มิดี้เป็นตัวกลางในการสื่อสารกับอุปกรณ์มิดี้เหล่านั้น ทำให้การแต่งเพลงหรือการบันทึกเสียงง่ายขึ้นมากและที่สำคัญคือ ราคาที่ถูกมากเมื่อเทียบกับสตูดิโอที่ไม่ใช้ระบบมิดี้

2.1.2 ระบบมิดี้

ระบบมิดี้จะประกอบด้วยส่วนอินพุท คือ คีย์บอร์ด ซึ่งเป็นส่วนที่เราจะเล่นดนตรีลงไป โดยขณะที่เราเล่นโน้ตต่างๆนั้นข้อมูลของโน้ตจะถูกแปลงเป็นข้อมูลของมิดี้ที่แสดงคีย์ที่ถูกกด ส่งไปให้ midi in ของส่วน master ซึ่งปกติจะเป็นเครื่องซีเควนเซอร์ หรือ ไมโครคอมพิวเตอร์ ทำหน้าที่ควบคุมหน่วยกำเนิดเสียง (sound module หรือ tone gen.) แต่ละชิ้นให้เล่นเป็นโน้ตต่างๆกันเพื่อประสานเสียงกันเป็นเสียงดนตรี โดยแต่ละชิ้นก็จะแทนเสียงเครื่องดนตรีแต่ละประเภท ข้อมูลที่ตัว master ส่งไปนั้นก็จะได้จากการบันทึก (record) ข้อมูลที่นักดนตรีเล่นผ่านเข้ามาทาง midi in เก็บไว้ในหน่วยความจำก่อน คล้ายๆกับการอัดเทปทั่วไปแต่ข้อมูลที่บันทึกเป็นข้อมูลมิดี้แทน ไม่ใช่เป็นเสียงเพลง ดังนั้นการแก้ไขข้อมูลดังกล่าวในกรณีที่นักดนตรีเล่นพลาด ย่อมทำได้ง่ายกว่าเพราะเป็นข้อมูลแบบ ดิจิตอล เราจะเห็นว่าการใช้ระบบมิดี้มาช่วยในการทำงานทางดนตรีนั้น เราไม่จำเป็นต้องมีนักดนตรีหลายคนเพื่อเล่นดนตรีในแต่ละชิ้นเราสามารถใช้ master แทน ซึ่งนักดนตรีคนเดียวก็สามารถ

ทำเพลงได้อย่างสบาย โดยให้ตัว master เล่นเครื่องดนตรีชิ้นอื่นๆแทนนักดนตรี ซึ่งความจริงก็คือเราบันทึกข้อมูลของเครื่องดนตรีแต่ละชิ้นๆ เก็บไว้ใน ส่วน master ก่อนนั้นเอง

2.1.3 ประเภทของอุปกรณ์มิดี้

อุปกรณ์มิดี้ทุกชิ้นจะมีสิ่งหนึ่งที่เหมือนกัน คือ มีไมโครโปรเซสเซอร์เป็นหัวใจของการทำงาน แต่เราก็ยังแบ่งอุปกรณ์มิดี้ได้เป็น 2 พวกใหญ่ๆคือ

1. **midi instrument** พวกนี้มีหน้าตาเหมือนเครื่องดนตรีโดยทั่วไป แต่ติดตั้งระบบมิดี้เข้าไปด้วยเช่น ซินธิไซเซอร์ กีตาร์มิดี้ แซกโซโฟนไฟฟ้า กลองไฟฟ้า ปกติแล้วในตัวของมันเองจะมีหน่วยกำเนิดเสียงอยู่ด้วยหรือพูดง่ายๆ คือ มันสามารถให้เสียงดนตรีออกมาได้ ถ้ามันไม่สามารถให้เสียงดนตรีออกมาได้เราจะเรียกมันว่า คอนโทรลเลอร์ เพราะตัวมันเองมีหน้าตาเป็นเครื่องดนตรีแต่ไม่สามารถให้เสียงดนตรีได้ ต้องส่งข้อมูลมิดี้ไปควบคุมหน่วยกำเนิดเสียงตัวอื่นให้ออกเสียงดนตรีแทน

2. **non - midi instrument** พวกนี้หน้าตาของมันจะไม่มีความเป็นเครื่องดนตรีเลย แต่จะมีปุ่มฟังก์ชันเต็มไปหมดและอาจจะมีจอแสดงผลเล็กๆไปจนถึงจอLCD ขนาดใหญ่ที่เราสามารถใช้พูลดาวน์เมนูได้ อุปกรณ์พวกนี้ได้แก่ ซีเควนเซอร์ ดรัมแมชีน เป็นต้น หน้าที่โดยรวมของมัน คือ การประมวลผลเกี่ยวกับข้อมูล ทางมิดี้ทั้งหมด และควบคุมอุปกรณ์มิดี้ที่มีหน่วยกำเนิดเสียงให้เล่นเป็นเพลงออกมา จะว่าไปแล้วอุปกรณ์เหล่านี้ก็เปรียบเสมือน conductor ที่คอยดูแลให้นักดนตรีแต่ละคนเล่นดนตรีนั่นเอง

2.1.4 ภายในตัวมิดี้

การสื่อสารข้อมูลของมิดี้เป็นการสื่อสารแบบอะซิงโครนัสด้วยความเร็ว 31250 บิตต่อวินาที ประกอบด้วย start bit และ stop bit อย่างละ 1 บิต และมีขนาดของข้อมูล 8 บิต รวมเป็น 10 บิต สายที่ใช้เป็นสายตีเกลียว (twisted-pair) มีชิลด์ (shield) พันรอบ ขนาดของความยาวสายไม่ควรเกิน 1.5 เมตร ขั้วต่อ (connector) แบบ 5 pin DIN ตัวผู้ มิดี้จะใช้เพียง pin 4, 5 เป็นสายสัญญาณ pin ที่ 2 จะต่ออยู่กับกราวด์(GND) สำหรับพอร์ตของมิดี้ที่ติดกับอุปกรณ์มิดี้ก็จะเป็น DIN ตัวเมีย สามารถส่งพอร์ตได้ 3 ประเภทคือ MIDI IN, MIDI OUT และ MIDI THRU โดย MIDI IN จะเป็นพอร์ตที่รับข้อมูลจากอุปกรณ์ภายนอก MIDI OUT จะเป็นพอร์ตที่ส่งข้อมูลออกในขณะที่มีการเล่น ส่วน MIDI THRU จะเป็นพอร์ตที่ส่งข้อมูลออกจากตัวอุปกรณ์แต่เป็นข้อมูลที่เหมือนกับได้รับจาก MIDI IN, ดังนั้น MIDI THUR จึงเปรียบเสมือนปลั๊ก 3 ตา ที่มีไว้สำหรับส่งผ่านข้อมูลจาก master ไปยังอุปกรณ์ตัวอื่นนั่นเอง เราจะเห็นว่าข้อมูลมิดี้ที่ส่งจาก master เข้าสู่อุปกรณ์ตัวแรกแล้ว ข้อมูลชุดเดิมก็จะส่งต่อไปยังอุปกรณ์ตัวต่อไปที่มีข้อมูลเหมือนกันทุกประการแต่อุปกรณ์แต่ละตัวจะตอบสนองเฉพาะข้อมูลบางชุดเท่านั้น โดยข้อมูลที่ส่งไปแต่ละชุดนั้น (package) จะประกอบด้วยจำนวนไบต์ที่แตกต่างกัน (ชุดข้อมูลที่ส่งไปนี้จะมีไบต์ แสดงสถานะหรือในภาษาคอมพิวเตอร์เรียกว่า heading) เป็นตัวแสดงว่า ชุดข้อมูลดังกล่าวเป็นของอุปกรณ์ตัวใด โดยในระบบมิดี้ ตัว master จะควบคุมอุปกรณ์มิดี้ได้ 16 แชนเนล(channel) หรือ 16 ช่อง อุปกรณ์แต่ละตัวก็จะถูกกำหนดให้ตอบสนองเฉพาะแชนเนลใด

แชนเนลหนึ่งใน 16 แชนเนล เราสามารถเพิ่มอุปกรณ์มิดี้ในระบบให้มากกว่า 16 ตัว ได้ โดยการกำหนดให้อุปกรณ์บางตัวตอบสนองในแชนเนลที่เหมือนกัน แต่อุปกรณ์คู่ดังกล่าวก็จะทำงานตามข้อมูลที่ได้รับจากมิดี้เหมือนกันทุกประการ เช่น ทั้งคู่จะโน้ต โด เร มี พร้อมๆกัน และในทางตรงกันข้ามเราสามารถลดจำนวนอุปกรณ์มิดี้ในระบบให้น้อยกว่า 16 ตัวได้ โดยใช้ อุปกรณ์มิดี้ที่สามารถประพฤติตัวคล้ายกับเป็นอุปกรณ์หลายๆตัวได้(เช่น ซินธิไซเซอร์) ภายในจะมีหน่วยกำเนิดเสียงหลายๆตัว (module) แล้วผู้ใช้ก็จะกำหนดให้แต่ละโมดูลตอบสนองในแต่ละแชนเนลเท่านั้นลักษณะความสามารถอันนี้เราเรียกว่า Multitimbral

2.1.5 รูปแบบของการส่งข้อมูลมิดี้

การส่งข้อมูลในระบบมิดี้จะเป็นการส่งในลักษณะ "ชุดข้อมูล" (package) โดยในไบต์แรกของแต่ละชุดข้อมูล (heading) ซึ่งในมิดี้เราเรียกว่า ไบต์แสดงสถานะจะมี MSB เป็น 1 ไบต์ข้อมูลจะมี MSB เป็น 0

MSB 1 บิต	บิตข้อมูล 7 บิต

MSB เป็น 0 คือ ไบต์แสดงข้อมูล

MSB เป็น 1 คือ ไบต์แสดงสถานะ

ดั้งนั้นเราจะสรุปได้ว่า MSB ที่ส่งในระบบมิดี้จะเป็นตัวที่ชี้ว่าข้อมูลตัวนั้นเป็นไบต์แสดงสถานะ หรือไบต์ข้อมูล ทำให้สามารถใช้ได้เพียง 7 บิตที่เหลือ มาแสดงความหมายต่างๆทางดนตรีได้เท่านั้น โดยที่ไบต์แสดงสถานะเป็นตัวที่แสดงว่าไบต์ข้อมูลที่ตามมาเป็นประเภทของไบต์แสดงสถานะใดซึ่งมี 2 ประเภทคือ ข้อมูลแสดงแชนเนล (channel message) และข้อมูลระบบ (system message)

2.1.6 ข้อมูลแสดงแชนเนล

ไบต์แสดงสถานะที่เป็นประเภทข้อมูลแสดงแชนเนล จะส่งตรงไปให้กับแชนเนลใดแชนเนลหนึ่งในระบบเท่านั้น โดยที่ไบต์แสดงสถานะประเภทนี้จะบ่งบอกถึงแชนเนลที่ส่งไปในบิตที่ 0 – 3 ถ้าอุปกรณ์มิดี้ที่เป็นสลาฟ(slave) ตัวใดมีแชนเนลตรงกับแชนเนลที่ส่งไป สลาฟตัวนั้นจึงจะตอบสนองต่อไบต์ข้อมูลที่ตามมา ซึ่งข้อมูลแสดงแชนเนลจะแบ่งออกได้เป็น ข้อมูลเสียง(voice message) กับ ข้อมูลโหมด (mode message)

ข้อมูลเสียง (Channel voice message)

1. Note On, Note Off จะเกี่ยวข้องถึงการเล่นโน้ตต่างๆบนอุปกรณ์มิดี้
2. Control Change ข้อมูลชุดนี้จะเป็นการควบคุมเกี่ยวกับตัวควบคุมเสียงบางประเภท เช่น ความดัง (volume)
3. Pitch Bend Change เกี่ยวข้องกับการเอื้อนเสียงของแชนเนล
4. Program Change จะใช้ในการเปลี่ยนเสียงของแชนเนลนั้นๆ
5. Polyphonic Key ข้อมูลชุดนี้จะส่งเมื่อมีการขยี้คีย์หลังจากที่เราเล่นโน้ตไปแล้ว

6. Channel Pressure ข้อมูลนี้จะเกี่ยวข้องกับการขยี้บนคีย์เช่นกันแต่จะมีผลต่อเสียงทั้งแชนเนลไม่เฉพาะเจาะจงเฉพาะคีย์อย่างในข้อ 5

ข้อมูลโหมด (Channel mode message)

เป็นการควบคุมโหมดในการทำงานของแต่ละแชนเนลในระบบ โดยจะมีโหมดการทำงานดังนี้

1. Poly / Mono คือ การเซ็ตให้อุปกรณ์มิดี้สามารถเปล่งเสียงโน้ตในแบบ Poly หรือ Mono โดยที่แบบ Mono จะเปล่งเสียงได้ที่ละ 1 โน้ตเท่านั้น ส่วนแบบ Poly จะเปล่งเสียงได้หลายโน้ตพร้อมๆ กัน
2. Omni On /Off เป็นการบังคับให้แชนเนลนั้นตอบสนองเฉพาะข้อมูลที่ตรงกับแชนเนลของตน หรือตอบ สนองในทุกๆแชนเนล ถ้า On ตอบสนองในทุกแชนเนลโดยปกติ ถ้านำอุปกรณ์มิดี้มาต่อกันเป็นระบบ แล้วโหมดที่ใช้งานคือ Poly :Omni OFF

2.1.7 ข้อมูลระบบ (System message)

ข้อมูลประเภทนี้จะส่งไปโดยไม่เจาะจงแชนเนลใดแชนเนลหนึ่งในระบบ ทุกๆแชนเนลจะตอบรับข้อมูลเหล่านี้ แบ่งออกได้เป็น

1. Common Message ข้อมูลชุดนี้จะเกี่ยวข้องกับการเลือกเพลง การเลือกจังหวะ การเลือกตำแหน่งของเพลงที่จะเล่น และการบังคับให้อุปกรณ์ทางอนาล็อกปรับแต่งระบบเสียงของตัวเอง (tune request)
2. System Real Time จะเกี่ยวข้องโดยตรงกับการสั่งเริ่มเล่นเพลง (start) การหยุด (stop) และข้อมูลที่สำคัญคือ system clock ที่เปรียบเสมือน clock ของเพลงที่ทำให้เพลงเดินเร็วหรือช้า ถ้าความถี่ในการส่ง system clock มาก เพลงก็จะเดินเร็วขึ้น

2.2 โครงสร้างไฟล์มิดี้

ไฟล์รหัสมิดี้จะมีลักษะเป็นส่วนๆ โดยในแต่ละส่วนจะประกอบไปด้วยตัวอักษร 4 ตัวและตามด้วยข้อมูลขนาด 32 บิต ซึ่งสามารถแบ่งออกเป็น 2 ประเภท คือ Header chunk และ track chunk ไฟล์มิดี้จะเริ่มด้วย header chunk และตามด้วย track chunk เสมอมีลักษณะดังนี้

<Header chunk> <track chunk> <track chunk> <track chunk> <track chunk>.....

2.2.1 Header Chunk

Header chunk คือ เป็นส่วนการเริ่มต้นของไฟล์มิดี้เสมอและมีเพียง 1 Header chunk เท่านั้นซึ่งมีส่วนประกอบและรายละเอียดมีดังนี้

<Header chunk> = <chunk type> <length of header data> < header data >

โดยที่

<chunk type> เป็นตัวอักษร 4 ตัว ที่บอกว่าเป็น Header chunk หรือ track chunk ถ้าเป็น Header chunk จะเป็นตัวอักษร "MThd" มีขนาด 4 ไบต์ ในรหัส ASCII มีค่าเป็น 'M=4D' 'T=54' 'h=68' 'd=64'

<length of header data> เป็นส่วนที่บอกความยาวของ header data ที่ตามหลัง chunk type ว่ามีขนาดเท่าใด มีขนาดในการเก็บ 4 ไบต์ โดยที่จะมีค่าเท่ากับ 00 00 00 06 เนื่องจาก header data มีขนาด 6 ไบต์เสมอ

< header data> ประกอบด้วยส่วนย่อยอีก 3 ส่วน คือ

< header data> = <format> <number of tracks> <delta-time>

<format> เป็นส่วนที่บอกชนิดของ midi track มีขนาด 2 ไบต์ มี 3 ชนิด คือ

1. 00 00 - single track
2. 00 01 - multiple track, Synchronous
3. 00 02 - multiple track, Asynchronous

<number of tracks> เป็นส่วนที่บอกจำนวนของ track chunk ในไฟล์มิดี้ มีขนาด 2 ไบต์

<delta-time> เป็นส่วนที่บอกค่าเวลาของ delta-time ต่อ Quarter note มีขนาด 2 ไบต์

หมายเหตุ delta-time คือ ค่าเวลาเหตุการณ์ของคำสั่ง หรือก็คือ ค่าเวลาของโน้ตที่เล่น
Quarter note คือ ค่าความยาวเสียงของโน้ตตัวดำ หรือ ¼ ของ โน้ตตัวกลม

2.2.2 Track chunk

Track chunk คือ เป็นส่วนที่เก็บข้อมูลคำสั่งต่างๆ ของไฟล์มิดี้ มีได้หลาย Track chunk ซึ่งมีรายละเอียดดังนี้

<Track chunk> = <chunk type> <length of track data> <track data>

โดยที่

<chunk type> เป็นตัวอักษร 4 ตัว ที่บอกว่าเป็น Header chunk หรือ track chunk ถ้าเป็น track chunk จะเป็นตัวอักษร "MTrk" มีขนาด 4 ไบต์ ในรหัส ASCII มีค่าเป็น 'M=4D' 'T=54' 'r=72' 'k=6B'

<length of track data> เป็นส่วนที่บอกความยาวของ track data ที่ตามหลัง chunk type ว่ามีความยาวเท่าใด มีขนาดในการเก็บ 4 ไบต์

<track data> เป็นส่วนที่เก็บข้อมูลคำสั่งที่สั่งให้ midi ทำงานตามต้องการ ขนาดของข้อมูลขึ้นอยู่กับเหตุการณ์ของคำสั่งทั้งหมดที่มี ลักษณะดังนี้

<track data> = <MTrk event> + <MTrk event> + ...

<MTrk event> เป็นส่วนคำสั่งของ midi ต่างๆ ประกอบด้วยส่วนย่อย 2 ส่วน คือ

<MTrk event> = <delta-time> <event>

1. <delta-time> เป็นส่วนที่บอกค่าเวลาของเหตุการณ์ นั้นๆ มีขนาด 1-2 ไบต์
2. <event> เป็นส่วนคำสั่งของเหตุการณ์ที่สั่งให้ midi ทำงาน ขนาดขึ้นอยู่กับคำสั่งนั้นๆ ซึ่งมี 2 ประเภท คือ
   - <midi event> คือ ส่วนที่บอกสถานะที่กำลังทำงาน (running status) สำหรับข้อความเสียง และโหมด ในแชนเนลที่ต้องการ
   - <meta-event> คือ ส่วนที่บอกข้อมูลระบบ จะไม่มีการกำหนดแชนเนล และทุกแชนเนลต้องตอบสนองข้อมูลของระบบนี้

2.2.2.1 คำสั่งข้อมูลแสดงแชนเนล (MIDI event command)

ไบต์คำสั่งมีขนาด 1 ไบต์ซึ่งแบ่งเป็น 2 ส่วน คือ 4 บิตบนจะเป็นส่วนคำสั่ง ส่วน 4 บิตล่างเป็นส่วนของหมายเลขแชนเนล(channel number)ที่จะกำหนดให้ทำงาน ซึ่งมีทั้งหมด 16 แชนเนล และตามด้วยข้อมูลของคำสั่งดังแสดงตามตารางที่ 2.1 โดยที่ x คือ หมายเลขแชนเนล (channel number)

<midi event> = < คำสั่งและหมายเลขแชนเนล > < ข้อมูล >

ตัวอย่างเช่น 90 3C 64 โดยที่ ไบต์แรกคือ 80 เป็นคำสั่ง Note On ที่แชนเนล 1 ไบต์ที่ 2 คือ 3C เป็นหมายเลขโน้ตดังตารางที่ 2.3 เลขฐานสิบคือ 60 (เสียง C octave ที่ 5) ไบต์ที่ 3 คือ 64 เป็นน้ำหนักในการเล่นโน้ตเป็นเลขฐานสิบคือ 100

คำสั่ง (HEX.)	ข้อมูล (Data)	รายละเอียด (Comment)
8x	nn vv nn = หมายเลขโน้ต vv = น้ำหนักในการเล่น (velocity)	ช่วงเลิกเล่นโน้ต (Note off)
9x	nn vv nn = หมายเลขโน้ต vv = น้ำหนักในการเล่น (velocity)	ช่วงเล่นโน้ต (Note on)
Ax	nn vv nn = หมายเลขโน้ต vv = น้ำหนักในการเล่น (velocity)	เพิ่มความดันของคีย์อีกหลังจากที่กดคีย์นั้นไปแล้ว (Polyphonic Key pressure)

Bx	cc vv cc = หมายเลขตัวควบคุม vv = หมายเลขของ ตัวควบคุมตัวใหม่	ควบคุมการเปลี่ยนตัวควบคุมที่ใช้ในการปรับแต่งเสียง
Cx	pp pp = หมายเลขเครื่องดนตรี	เป็นการกำหนดชนิดของเครื่องดนตรี
Dx	Cc cc = หมายเลขแชนเนล	กำหนดความดันเฉลี่ยของทุกๆ คีย์ของแชนเนล
Ex	bb tt bb = bottom(least sig.) 7 bit of value tt = top (most sig.) 7 bit of value	การเอื้อนเสียง (Pitch wheel change)

ตารางที่ 2.1 คำสั่งข้อมูลแสดงแชนเนล

2.2.2.2 **คำสั่งข้อมูลระบบ** (Meta-event command)

เป็นคำสั่งเกี่ยวกับข้อมูลระบบ และทุกแชนเนลต้องตอบสนอง ซึ่งมีรูปแบบของคำสั่งดังนี้

<meta-event> = { FF <command> <length of data> <data> }

ทุกๆ meta-event ต้องเริ่มด้วยค่า FF เสมอดังรูปแบบข้างต้นแล้วตามด้วยคำสั่งต่อด้วยความยาวของข้อมูล และสุดท้ายข้อมูลของคำสั่ง ดังตารางที่ 2.2 ตัวอย่างเช่น FF 2F 00 เป็นคำสั่งในการจบ Track ของแต่ละ Track chunk ไบต์แรก คือ FF เป็นการเริ่มต้นของคำสั่งข้อมูลระบบ ไบต์ที่สอง คือ 2F เป็นคำสั่งดังตารางที่ 2.2 ส่วนไบต์ที่สามคือ 00 เป็นข้อมูลของคำสั่ง

คำสั่ง (HEX.)	ข้อมูล (Data)	รายละเอียด (Comment)
00	nn ssss nn = 02 (length of 2-byte number sequence number) ssss = Sequence	เซ็ตลำดับหมายเลขของ Track
01	nn tt ..... nn = ความยาวของตัวอักษร tt = ตัวอักษร	ข้อความเกี่ยวกับชื่อของ Track
02	nn tt ..... nn = ความยาวของตัวอักษร tt = ตัวอักษร	ข้อความเกี่ยวกับลิขสิทธิ์

03	nn tt ..... nn = ความยาวของตัวอักษร tt = ตัวอักษร	ข้อความบอกชนิดของ Track
04	nn tt ..... nn = ความยาวของตัวอักษร tt = ตัวอักษร	ข้อความบอกชนิดของเครื่องดนตรีที่เล่น
05	nn tt ..... nn = ความยาวของตัวอักษร tt = ตัวอักษร	ข้อความบอกทำนองของเพลง
06	nn tt ..... nn = ความยาวของตัวอักษร tt = ตัวอักษร	ข้อความระบุตำแหน่งที่จะเล่น
07	nn tt ..... nn = ความยาวของตัวอักษร tt = ตัวอักษร	ข้อความแนะนำเหตุการณ์ของเพลง
2F	00	เป็นการจบ Track chuck
51	03 tt tt tt tt tt tt = ไมโครวินาที ต่อ Quarter note	กำหนดความเร็วของเพลง (Tempo)
58	04 nn dd cc bb nn = ตัวเลขของจังหวะห้อง เช่น 4/4 dd = denominator of time sig.2=quarter 3 = eighth, etc. cc = number of ticks in metronomeclick bb = number of 32nd note to the quarter note	กำหนดจังหวะของห้องเพลง (Time signature)
59	02 sf mi sf = sharps / flats(-7=7 flats, 0 = key of C,7=7 sharps) mi = major / minor (0=major, 1=minor)	กำหนดคีย์เสียงของเพลง (Key signature)

ตารางที่ 2.2 คำสั่งข้อมูลระบบ

<u>การเล่นโน้ตของมิดี้</u>

ในการกำหนดจังหวะมิดี้จะมีการส่งสัญญาณนาฬิกาออกไปด้วยอัตรา 24 ลูก ต่อ โน้ตตัวดำหนึ่งตัว(ส่งออกไปอย่างต่อเนื่อง) เพื่อเป็นจังหวะอ้างอิงของระบบ และในการกำหนดความเร็วของ

สัญญาณนาฬิกาต่อ 24 ลูก(tempo) จะกำหนดโดยใน 1 นาทีจะให้มีการส่งกี่ครั้ง ในการส่ง 1 ครั้ง(24 ลูก) จะเท่ากับโน้ตตัวดำหนึ่งตัวแล้วนำมาคำนวณหาค่าเวลาในการเล่นโน้ตตัวดำหนึ่งตัว เช่น เพลงที่มี Tempo = 120 จะมีค่าเวลาในการเล่นโน้ตตัวดำหนึ่งตัวเท่ากับ 60 วินาที / 120 = 500,000 ไมโครวินาที

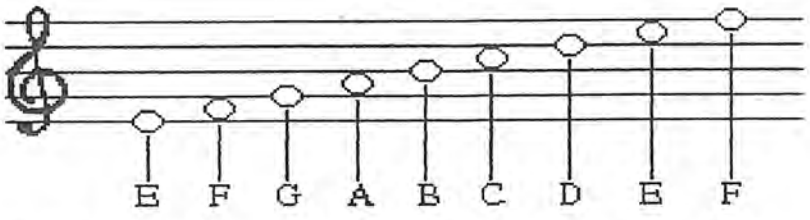
ระดับเสียงของโน้ตจะมีการกำหนดหมายเลข(note number) มีค่าได้ตั้งแต่ 0-127 โดย 0 จะเป็นโน้ตที่ต่ำที่สุด 127 เป็นโน้ตที่เสียงสูงที่สุด แต่โดยทั่วไปเครื่องดนตรีจะรับรู้โน้ตได้น้อยกว่า เช่น ซินธิไซเซอร์ ที่มี 5 ออคเตฟ จะรับรู้โน้ตได้ ตั้งแต่ 36-96 หมายเลขโน้ตของโดกลาง(middle c) คือ 60 ดังตารางที่ 2.3 น้ำหนักในการเล่นโน้ตหรือความเร็วที่กดคีย์ คือ ถ้ากดเร็วจะได้เสียงที่ดัง และ ถ้ากดช้าจะได้เสียงที่เบาเช่นกัน ความเร็วของคีย์มีค่าตั้งแต่ 0-127 โดยความเร็ว = 127 จะเป็นเสียงที่ดังที่สุด ส่วนความเร็ว = 0 จะไม่มีเสียงออกมาสำหรับเครื่องดนตรีที่ไม่มีการตอบสนนองต่อความเร็วจะกำหนดให้มีค่าเท่ากับ 64

Note Numbers / Octave	C | C# | D | D# | E | F | F# | G | G# | A | A# | B
0 ||	0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11
1 ||	12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23
2 ||	24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35
3 ||	36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47
4 ||	48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59
5 ||	60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71
6 ||	72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 | 81 | 82 | 83
7 ||	84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95
8 ||	96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 | 104 | 105 | 106 | 107
9 ||	108 | 109 | 110 | 111 | 112 | 113 | 114 | 115 | 116 | 117 | 118 | 119
10 ||	120 | 121 | 122 | 123 | 124 | 125 | 126 | 127 |

ออคเตฟ(Octave) หมายถึง ช่วงของระดับเสียงของโน้ต โด(C) ถึง ที(B) ในแต่ละระดับเสียง

ตารางที่ 2.3 หมายเลขโน้ต(note number)

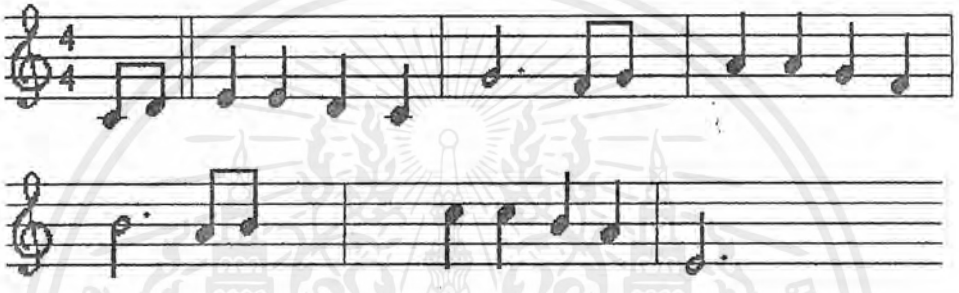
ในการเทียบระดับเสียงเราจะกำหนดหมายเลขของระดับเสียงโน้ต โดยจะเทียบกับกุญแจซอลในบรรทัด 5 เส้น เส้นล่างสุดจะเป็นเสียง เร (E) ให้เป็นหมายเลขโน้ตในออกเตฟที่ 5 มีค่าเท่ากับ 64 ซึ่งเป็นค่าที่คิดว่าเหมาะสมและไล่ระดับไปตามตารางที่ 2.3 ดังรูปที่ 2.1



มีค่าเท่ากับ 64 65 67 69 71 72 74 76 78

รูปที่ 2.1 ตัวอย่างการเทียบระดับเสียงของมิดี้กับบรรทัด 5 เส้น

ตัวอย่าง ภาพของโน้ตที่แปลงเป็นไฟล์มิดี้มีรายละเอียดดังนี้ ซึ่งแสดงเป็นเลขฐาน 16



Header chunk (MThd) ความยาว Header data Format 1 2 Track delta-Time Track chunk (MTrd) ความยาว Track data

4D 54 68 64 00 00 00 06 00 01 00 02 00 60 4D 54 72 6B 00 00 00 13

Time signature Tempo จบแทรก (End Track) Track chunk (MTrd) ความยาว Track data

00 FF 58 04 04 02 18 08 00 FF 51 03 07 A1 20 00 FF 2F 00 4D 54 72 6B 00 00 00 8D

เซ็ทเครื่องดนตรี คำสั่ง Note on ที่ Ch. 1 Note Off Note On Note Off Note On Note Off Note On

00 C0 21 00 90 3C 40 30 3C 00 00 3E 40 30 3E 00 00 40 40 70 40 00 00 40 4

Note Off Note On Note Off Note On Note Off Note On Note Off

60 40 00 00 3E 40 60 3E 00 00 3C 40 60 3C 00 00 43 40 82 68 43 00

Note On Note Off Note On Note Off Note On Note Off Note On Note Off

00 41 40 30 41 00 00 43 40 30 43 00 00 45 40 60 45 00 00 45 40 60 45 0

Note On Note Off Note On Note Off Note On Note Off Note On

00 43 40 60 43 00 00 41 40 60 41 00 00 47 40 82 68 47 00 00 45 40

Note Off Note On Note Off Note On Note Off Note On Note Off Note On

30 45 00 00 47 40 30 47 00 00 48 40 60 48 00 00 48 40 60 48 00 00 47 4

Note Off Note On Note Off Note Off Note On จบแทรก (End Track)

60 47 00 00 45 40 60 45 00 00 40 40 82 68 40 00 00 FF 2F 00

4D 54 68 64 MThd (รหัส ASCII) chunk type ของ header chunk

00 00 00 06 ความยาวของ header data เท่ากับ 6 ไบต์

00 01 00 02 00 78 header data: 00 01 = format 1; 00 02 = จำนวน 2 track;

00 78 = delta-time หรือ ก็คือความยาวเสียงของโน้ตตัวดำ

เท่ากับ 120 (แปลงจากฐาน 16)

4D 54 72 6B MTrk แทรกแรก chunk type ของ track chunk

00 00 00 13 ความยาวของ track data เท่ากับ 19 ไบต์ (แปลงจากฐาน 16)

track data

delta-time	event	comment
00	FF 58 04 04 02 18 08	FF 58 = time-signature ;04 = ความยาว 4 ไบต์ 04 = 4/4; 02 = quarter note; 18 = 24 midi clock/click; 08 = 8 ของ 32nd note/24 midi clock
00	FF 51 03 07 A1 20	FF 51=tempo ; 03=ความยาว 3 ไบต์ ; 07 A1 20 = เวลาในการเล่นโน้ตตัวดำหนึ่งตัว 500,000 m.Sec / quarter note(แปลงจากฐาน 16)
00	FF 2F 00	FF 2F 00 = จบ track chunk

4D 54 72 6B MTrk แทรกที่สอง chunk type ของ track chunk

00 00 00 8D ความยาวของ track data เท่ากับ 141 ไบต์(แปลงจากฐาน 16)

track data

delta-time	event	comment
00	C0 21	C0 = เซ็ตเครื่องดนตรีที่ แชนเนล 1 ; 21= เป็นหมายเลขเครื่องดนตรีที่มีค่าเท่ากับ 33 (แปลงจากฐาน 16)
00	90 3C 40	90 = แชนเนล 1 note on; 3C =ระดับเสียงของโน้ตโด(60) จากตารางที่ 2.3 ; 40 = ความหนักของเสียงเท่ากับ 64 (แปลงจากฐาน 16)
30	3C 00	30 = ค่าเวลาของ note on เท่ากันค่าของเขบ็ต 1 ชั้น เท่ากับ 48 (แปลงจากฐาน 16); 3C = ระดับเสียงของโน้ตโด(60) จากตารางที่ 2.3 00 = หยุดเล่นโน้ต(โด) ความหนักของเสียงเท่ากับ 0
00	3E 40	3E=ระดับเสียงของโน้ต(เร); 40= ความหนักของเสียงเท่ากับ 64 (แปลงจากฐาน 16)
30	3E 00	3C = ค่าเวลาของ note on เท่ากันค่าของเขบ็ต 1 ชั้น

เท่ากับ 48 (แปลงจากฐาน 16);

3E =ระดับเสียงของโน้ตเร(62) จากตารางที่ 2.3

00 = หยุดเล่นโน้ต(เร) ความหนักของเสียงเท่ากับ 0

00 FF 2F 00 FF 2F 00 = จบ track chunk

วิธีการสร้างไฟล์มิดี้

การสร้างไฟล์มิดี้ในโครงงานนี้เป็นไปตามมาตราฐานของโครงสร้างไฟล์มิดี้แต่มีข้อมูลของโครงสร้างของมิดี้บางส่วนไม่สามารถกำหนดได้แน่นอนเช่น Tempo ทางผู้จัดทำโครงงานนี้จึงเลือกค่าที่คิดว่าเหมาะสม วิธีการสร้างมีดังต่อไปนี้

1. สร้าง Header chunk ประกอบด้วย

4D 54 68 64 MThd (รหัส ASCII) chunk type ของ header chunk ต้องเริ่มด้วยเลขนี้เท่านั้นเป็นมาตราฐานของมิดี้

00 00 00 06 ความยาวของ header data เท่ากับ 6 ไบต์ เสมอ

00 01 00 02 00 78 header data:

00 01 = format 1 กำหนดชนิดของแทรกดังที่อธิบายข้างต้น

00 02 = ประกอบด้วย Track chunk 2 แทรก

00 60 = delta-time หรือ ก็คือ อัตราความยาวเสียงเทียบกับโน้ตตัวดำเท่ากับ 96 (แปลงจากฐาน 16) และนำค่านี้เป็นตัวอ้างอิงกับอัตราความยาวเสียงของโน้ตตัวอื่น เช่น ตัวเขบ็ต 1 ชั้น มีค่าเท่ากับ 48 (ครึ่งหนึ่งของตัวดำ) ส่วนตัวอื่นมีค่าดังตารางที่ 2.4

ตัวโน้ต	ค่าความยาวของเสียง
ตัวกลม	384
ตัวขาว	192
ตัวดำ	96
ตัวเขบ็ต 1 ชั้น	48
ตัวเขบ็ต 2 ชั้น	24
ตัวเขบ็ต 3 ชั้น	12

ตารางที่ 2.4 แสดงค่าความยาวเสียงของตัวโน้ตและตัวหยุด

2. สร้าง Track chunk ซึ่งจำนวนของแทรกจะมีกี่แทรกก็ได้ ในที่นี้เราจะใช้ 2 แทรก โดยที่ส่วนใหญ่แทรกแรกจะเป็นการกำหนดคุณสมบัติของเพลงเช่น Tempo Time-Signature เป็นต้น ส่วนแทรกต่อๆ ไปจะเป็นแทรกของข้อมูลโน้ต มีรายละเอียดดังนี้

4D 54 72 6B MTrk แทรกแรก chunk type ของ track chunk ต้องเริ่มด้วยเลขนี้เท่านั้นเป็นมาตราฐานของมิดี้ในการบอกเริ่มแทรก

00 00 00 13 ความยาวของ track data เท่ากับ 19 ไบต์ (แปลงจากฐาน 16)

track data ประกอบด้วยส่วนของเหตุการณ์ <MTrk event> + <MTrk event> + ... ซึ่งได้บอกรายละเอียดไว้ข้างต้น

delta-time	event	comment
00	FF 58 04 04 02 18 08	FF 58 = time-signature ;04 = ความยาว 4 ไบต์ 04 = 4/4; 02 = quarter note; 18 = 24 midi clock/click; 08 = 8 ของ 32nd note/24 midi clock
00	FF 51 03 07 A1 20	FF 51= คำสั่ง tempo ; 03=ความยาว 3 ไบต์ ; 07 A1 20 = เวลาในการเล่นโน้ตตัวดำหนึ่งตัว 500,000 m.Sec / quarter note(แปลงจากฐาน 16) ค่านี้สามารถให้ผู้ใช้โปรแกรมเลือกค่าที่ต้องการได้ในช่อง Tempo
00	FF 2F 00	FF 2F 00 = คำสั่งในการจบ track chunk ต้องมีทุกแทรก

4D 54 72 6B MTrk แทรกที่สอง chunk type ของ track chunk ต้องเริ่มด้วยเลขนี้เท่านั้นเป็นมาตราฐานของมิดี้ในการบอกเริ่มแทรก

00 00 00 8D ความยาวของ track data เท่ากับ 141 ไบต์(แปลงจากฐาน 16) ขึ้นอยู่กับจำนวนข้อมูลของโน้ต

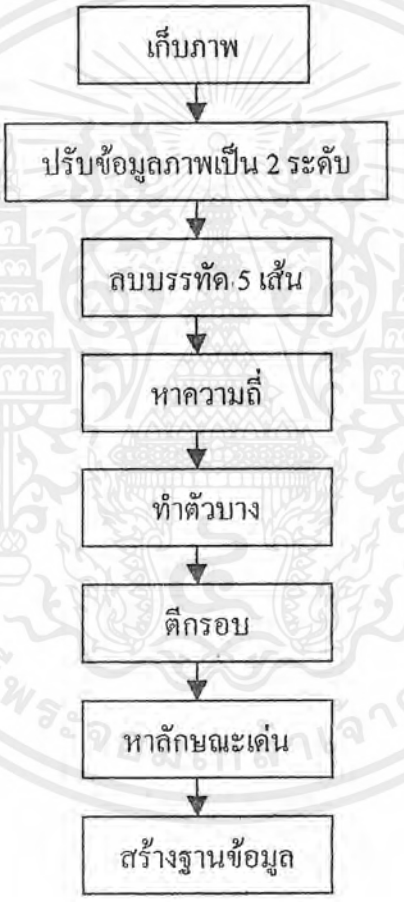
track data ประกอบด้วยส่วนของเหตุการณ์ <MTrk event> + <MTrk event> + ... ซึ่งได้บอกรายละเอียดไว้ข้างต้น

delta-time	event	comment
00	C0 21	C0 = เซ็ตชนิดของเครื่องดนตรีที่ออกที่แชนเนล 1 ซึ่งมีทั้งหมด 16 แชนเนลในการเซ็ตของโครงงานนี้ เราเลือกแชนเนล 1 จะเซ็ตไว้ช่วงแรกก่อนเริ่มข้อมูลโน้ต: 21= เป็นหมายเลขของเครื่องดนตรีที่มีค่าเท่ากับ 33

		(แปลงจากฐาน 16) ดูรายละเอียดได้ที่ภาคผนวก ข.
00	90 3C 40	90 = คำสั่ง note on ที่แชนเนล 1 ต้องมีก่อนเริ่มเล่นโน้ต ; 3C =ระดับเสียงของโน้ตโด(60) จากตารางที่ 2.3 ; 40 = ความหนักของเสียงเท่ากับ 64 (แปลงจากฐาน 16)
30	3C 00	30 = ค่าเวลาของ note on เท่ากันค่าของเขบ็ต 1 ชั้น เท่ากับ 48 (แปลงจากฐาน 16); ตามตารางที่ 2.4 3C = ระดับเสียงของโน้ตโด(60) จากตารางที่ 2.3 00 = ความหนักของเสียงเท่ากับ 0
00	3E 40	3E=ระดับเสียงของโน้ต(เร); 40= ความหนักของเสียงเท่ากับ 64 (แปลงจากฐาน 16)
30	3E 00	30 = ค่าเวลาของ note on เท่ากันค่าของเขบ็ต 1 ชั้น เท่ากับ 48 (แปลงจากฐาน 16);
00	FF 2F 00	FF 2F 00 = คำสั่งในการจบ track chunk ต้องมีทุก แทรก

บทที่ 3
การออกแบบและอัลกอริทึมการจดจำรูปแบบตัวพิมพ์โน้ตดนตรี

ในบทนี้จะกล่าวถึงการทำงานและขั้นตอนของการจดจำรูปแบบตัวพิมพ์โน้ตดนตรี และทฤษฎีที่ใช้กับขั้นตอนนั้นๆ ซึ่งขั้นตอนการทำงานจะประกอบไปด้วย การเก็บภาพ การปรับข้อมูลภาพให้เป็น 2 ระดับ การแยกบรรทัด 5 เส้น หาความถี่ในแนวแกนนอนและแนวแกนตั้ง การทำโครงกระดูก ตีกรอบตัวโน้ต การรู้จำตัวสัญลักษณ์และตัวโน้ตดนตรี สร้างฐานข้อมูล จนกระทั่งได้เป็นไฟล์ *.MID



รูปที่ 3.1 ขั้นตอนการทำการจดจำรูปแบบตัวพิมพ์โน้ต

3.1 ขั้นตอนการจดจำรูปแบบตัวพิมพ์โน้ต

ขั้นตอนที่ 1 การเก็บภาพ

การเก็บภาพจะเป็นการเก็บข้อมูลของภาพตัวพิมพ์โน้ตเข้ามา โดยใช้สแกนเนอร์รับภาพเข้ามาเก็บไว้ ในแผ่นดิสก์หรือ Frame memory ก็ได้ ภาพที่ใช้เป็นภาพ bitmap ขนาดไม่เกิน 1500 x 1500 พิกเซล

ขั้นตอนที่ 2 ปรับข้อมูลภาพให้เป็น 2 ระดับ

ข้อมูลที่อยู่ใน Frame memory หรือในแผ่นดิสก์จะถูกถ่ายลงในหน่วยความจำ จากนั้นจะถูกทำการแปลงให้มีค่าเพียง 2 ระดับ โดยทำการตัดค่า threshold ที่เหมาะสมจะทำให้ภาพที่ได้มีความชัดเจนและง่ายต่อการวิเคราะห์

ขั้นตอนที่ 3 การลบบรรทัด 5 เส้น

ภาพตัวพิมพ์โน้ตจะมีบรรทัด 5 เส้นเพื่อเป็นตัวบอกระดับเสียงของตัวโน้ต ในขั้นตอนนี้เส้นบรรทัดทั้ง 5 เส้นจะถูกกำจัดออกโดยใช้หลักการการลบเส้นที่มีความยาวเป็นเส้น ตรงที่ต่อเนื่องกันในแนวนอนและยาวมากที่สุด

ขั้นตอนที่ 4 หาความถี่ในแนวแกนตั้ง

การหาความถี่ของตัวโน้ตในแนวแกนตั้ง เพื่อที่จะหาขอบเขตของตัวโน้ตแต่ละตัวนอกจากนี้ ยังนำเอาความถี่ในแนวแกนตั้งมาพิจารณาแยกโน้ตที่เป็นชุดออกจากกัน (โน้ตเขบ็ต) และเป็นส่วนหนึ่งในการหากรอบ

ขั้นตอนที่ 5 การทำให้บาง

การทำให้บางเป็นการลดขนาดของข้อมูลภาพที่ผ่านการลบบรรทัด 5 เส้นแล้ว เพื่อให้เหลือแต่โครงของภาพโดยใช้วิธีการทำ Thinning

ขั้นตอนที่ 6 ตีกรอบตัวโน้ต

หลังจากขั้นตอนการหาความถี่ในแนวแกนตั้งก็จะได้ช่วงของตัวโน้ตแต่ละตัวแล้ว นำเอาความถี่ในแนวแกนตั้งของตัวโน้ตแต่ละตัว มาเป็นขอบเขตในการหาความถี่ในแนวแกนนอนเพื่อหากรอบของตัวโน้ต

ขั้นตอนที่ 7 การรู้จำตัวสัญลักษณ์และตัวโน้ตดนตรี

ขั้นตอนนี้จะทำการสแกนข้อมูลของภาพแต่ละตัวโน้ตที่ได้มาจากขั้นตอนที่ผ่านมา เพื่อหา ลักษณะเด่นของตัวโน้ตแต่ละตัวทำการจัดแบ่งและตีความหมายของตัวโน้ตแต่ละตัว

ขั้นตอนที่ 8 สร้างฐานข้อมูล

นำเอาผลการรู้จำมาเก็บไว้ในรูปของฐานข้อมูล เพื่อนำข้อมูลที่ได้ไปเปลี่ยนให้เป็นรหัสควบคุมของมิดี้การ์ด

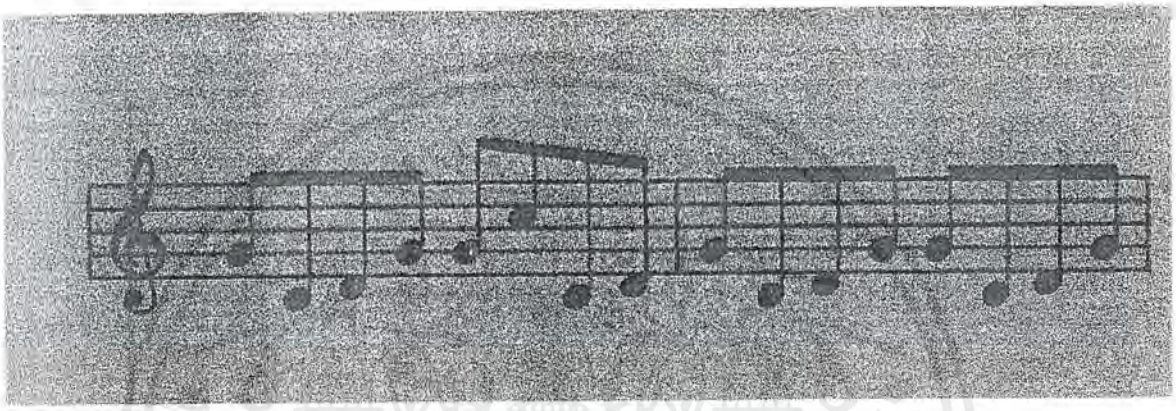
3.2 การเก็บภาพ

จะเป็นการเก็บข้อมูลของภาพตัวพิมพ์โน้ตเข้ามา โดยใช้สแกนเนอร์ เข้ามาไว้ในแผ่นดิสก์ หรือ Frame memory ก็ได้ ภาพที่ใช้เป็นภาพ bitmap ขนาดไม่เกิน 1500 x 1500 พิกเซล

3.3 การปรับข้อมูลภาพเป็น 2 ระดับ

3.3.1 ภาพระดับสีเทา

ข้อมูลภาพที่รับเข้ามาจะมีสีถึง 16.7 ล้านสี (2 ยกกำลัง 24 บิต) จะแบ่งเป็น แดง(Red) 1 byte เขียว(Green) 1 byte และ น้ำเงิน(Blue) 1 byte จะใช้สีแดง(Red) เพียงสีเดียวเพื่อทำเป็นภาพระดับเทาเนื่องจากมีความเหมาะสมกว่าใช้สีอื่น เมื่อนำค่าสีแดงไปทำฮิสโตแกรมเพื่อตัดค่าเธร็ชโฮลด์ ภาพ 2 ระดับที่ได้จะเป็นไปตามที่ต้องการคือ ไม่ทำให้ตัวโน้ตดนตรีขาด และ noise เกิดขึ้นน้อยกว่าการใช้สีอื่น



รูปที่ 3.2 ภาพระดับสีเทาที่ใช้สีแดง(Red)

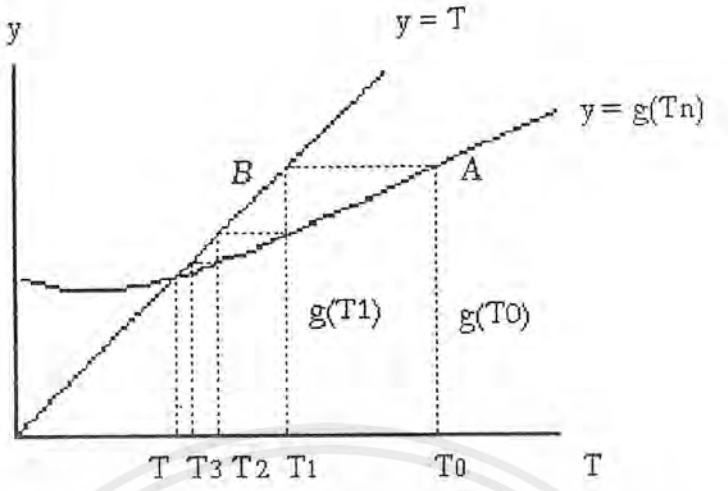
3.3.2 การหาค่าเธร็ชโฮลด์

Reddi et al. ได้พัฒนาวิธีการค้นหาแบบรวดเร็ว สำหรับการตัดค่า threshold ของภาพซึ่งได้อธิบายไว้ใหม่ในรูปของสมการคณิตศาสตร์ ต่อไปนี้

$$T_{k+1} = \frac{1}{2}\left[\frac{\sum_{i=0}^{T_K} n(i)i}{\sum_{i=0}^{Tk} n(i)} + \frac{\sum_{i=T_{K+1}}^{N} n(i)i}{\sum_{i=T_{K+1}}^{N} n(i)}\right] \qquad (3.1)$$

โดยที่ $n(i)$ คือ จำนวนรวมทั้งหมดของ gray-level ที่มีค่าเท่ากับ i

T_{K+1} , T_K คือ ค่า threshold ของการรวนซ้ำที่ $k+1$ และ k ตามลำดับ



รูปที่ 3.3 แสดงการหาค่าเธร็ชโฮลด์

$g(Tn)$ คือ ค่าเธร็ชโฮลด์ที่หาได้จากสมการที่ 3.1

สำหรับการหาค่าเธร็ชโฮลด์ดังแสดงในรูปที่ 3.3 นั้นจะกำหนดให้ค่าเธร็ชโฮลด์ให้มีค่าเริ่มแรกเท่ากับ$T0$ เมื่อแทนค่า $T0$ ใน $g(Tn)$ จะได้ $g(T0)$ การ projection ในแนวดิ่ง ของ $g(T0)$ บนเส้น $y = T$ ได้จุด B เมื่อลากมาตัดแกนเธร็ชโฮลด์ ก็จะได้ค่าเธร็ชโฮลด์ค่าใหม่ คือ $T1$ เราสามารถหาค่าเธร็ชโฮลด์ตัวต่อๆไปได้โดยวิธีเดียวกัน คือนำค่าเธร็ชโฮลด์ค่าใหม่แทนใน $g(Tn)$ แล้วทำการ projection ในแนวดิ่งบนเส้น $y = T$ ก็จะได้ค่าเธร็ชโฮลด์ค่าต่อไป $(T2, T3, ...)$

ค่า threshold T จะเป็นเลขจำนวนเต็มที่มีค่าระหว่าง 0 ถึง 255 สำหรับรูปภาพ 8-bit gray level image เพราะว่าด้านขวามือของสมการ (3.1) ได้ค่าเลขเป็นจำนวนจริง แล้วนำมาปัดเศษให้เป็นเลขจำนวนเต็มที่ใกล้เคียงหลังจากการวนรอบทุกครั้ง ค่าการลู่เข้า (convergence) จะได้มาเมื่อ

$$T_{K+1} = T_k \qquad (3.2)$$

แต่การใช้งานจริงในโครงงานโปรแกรมจะหยุดการวนรอบเมื่อ $T_{k+1} - T_k < 0.01$ เนื่องจากเป็นไปไม่ได้ที่จะได้ค่า $T_{k+1} = T_k$

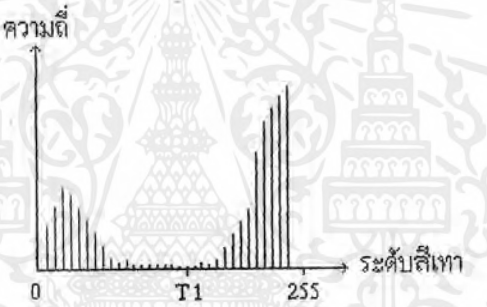
วิธีการหาค่าเธร็ชโฮลด์ มีดังต่อไปนี้

1. นำข้อมูลภาพที่เป็นระดับสีเทามาหาค่าความถี่สะสมของแต่ละระดับ ตั้งแต่ระดับ 0-255 ว่าแต่ละระดับสีเทามีค่าความถี่กี่พิกเซล เพื่อที่จะนำไปหาค่าเธร็ชโฮลด์ ตามสมการที่ 3.1 แสดงผลของค่าความถี่สะสมของระดับสีเทาตั้งแต่ 0-255 ได้โดยใช้ฮิสโตแกรม
2. กำหนดค่าเธร็ชโฮลด์ ครั้งแรก (T_k) ให้มีค่าสูงๆ แล้วแทนค่าตามสมการที่ 3.1 ก็จะได้ค่าเธร็ชโฮลด์ ครั้งต่อไป (T_{k+1})

3. พิจารณาค่า $T_{k+1} - T_k < 0.01$
4. ถ้าค่า $T_{k+1} - T_k < 0.01$ ไม่เป็นจริงก็ให้ค่า T_{k+1} เป็นค่า T_k แล้วนำค่า T_k ไปแทนในสมการ 3.1 อีกครั้งเพื่อหาค่า T_{k+1} ครั้งใหม่ แล้วพิจารณาค่า $T_{k+1} - T_k < 0.01$ อีกครั้ง
5. ถ้าค่า $T_{k+1} - T_k < 0.01$ เป็นจริง ค่า T_{k+1} ที่ได้เป็นค่าเธร็ชโฮลด์ แต่ถ้าไม่ใช่ให้ทำตามข้อ 4. จนกว่าค่า $T_{k+1} - T_k < 0.01$ เป็นจริง

หลังจากได้เธร็ชโฮลด์แล้วยังไม่สามารถนำไปใช้ได้ เนื่องจากภาพสแกนตัวพิมพ์โน้ตดนตรีค่อนข้างมีสีขาวมากกว่าสีดำมาก การตัดเธร็ชโฮลด์ที่ได้นี้จะทำให้ภาพตัวโน้ตบางส่วนขาดหายไป จึงต้องมีการนำช่วงตั้งแต่ค่าเธร็ชโฮลด์ครั้งแรก$T1$ ที่ได้ถึงค่า 255 ไปหาค่าเธร็ชโฮลด์ใหม่อีกครั้งมีวิธีการดังนี้

1. กำหนดให้ค่าเธร็ชโฮลด์ ที่ได้จากการหาครั้งแรกเท่ากับ $T1$



รูปที่ 3.3 กราฟฮีสโตแกรม

2. นำข้อมูลของกราฟฮิสโตแกรม ตั้งแต่ระดับ $T1$ ถึง 255 มาใช้ในการหาค่าเธร็ชโฮลด์ครั้งที่ 2
3. แทนค่าของสมการ 3.3 ด้วยข้อมูลจากข้อที่ 2. ดังต่อไปนี้

$$T_{k+1} = \frac{1}{2}\left[\frac{\sum_{i=T1}^{T_K} n(i)i}{\sum_{i=T1}^{Tk} n(i)} + \frac{\sum_{i=T_{K+1}}^{N} n(i)i}{\sum_{i=T_{K+1}}^{N} n(i)}\right] \qquad (3.3)$$

4. แล้วกำหนดค่าเธร็ชโฮลด์ในการหารอบแรกเท่ากับ (T_k) ให้มีค่าสูงๆ แล้วแทนค่าตามสมการที่ 3.3 ก็จะได้ค่าเธร็ชโฮลด์ ครั้งต่อไป (T_{k+1})
5. พิจารณาค่า $T_{k+1} - T_k < 0.01$
6. ถ้าค่า $T_{k+1} - T_k < 0.01$ ไม่เป็นจริงก็ให้ค่า T_{k+1} เป็นค่า T_k แล้วนำค่า T_k ไปแทนใน สมการ 3.3 อีกครั้งเพื่อหาค่า T_{k+1} ครั้งใหม่ แล้วพิจารณาค่า $T_{k+1} - T_k < 0.01$ อีกครั้ง

7. ถ้าค่า $T_{k+1} - T_k < 0.01$ เป็นจริง ค่า T_{k+1} ที่ได้เป็นค่าเธร็ชโฮลด์ แต่ถ้าไม่ใช่ให้ทำตามข้อ 4. จนกว่าค่า $T_{k+1} - T_k < 0.01$ เป็นจริง

ผลของการหาค่าเธร็ชโฮลด์ครั้งที่ 2 เมื่อนำไปใช้แล้วปรากฏว่าภาพตัวโน้ตที่ได้ไม่ขาด แต่เนื่องจากภาพที่ได้มี noise มาก อาจทำให้เกิดปัญหาขึ้นในการรู้จำได้เพราะ noise อาจจะมีลักษณะเช่นเดียวกับโน้ตได้ จึงนำค่าเธร็ชโฮลด์ ครั้งแรก(T_1) และค่าเธร็ชโฮลด์ครั้งที่สอง (T_2) มาคำนวณหาค่าเธร็ชโฮลด์ ที่เหมาะสมโดยใช้สมการดังนี้

$$T = T_1 + \left[\frac{T_2 - T_1}{4}\right] \qquad (3.4)$$

T คือ ค่าเธร็ชโฮลด์ ที่ต้องการ

T_1 คือ ค่าเธร็ชโฮลด์ ครั้งที่ 1

T_2 คือ ค่าเธร็ชโฮลด์ ครั้งที่ 2

ผลจากนำค่าเธร็ชโฮลด์ที่ได้จากสมการที่ 3.4 ไปใช้กับรูปสแกนตัวพิมพ์โน้ตดนตรีภาพตัวโน้ตที่ได้ไม่ขาดและ noise ที่เกิดขึ้นน้อยลง สรุปขั้นตอนการเธร็ชโฮลด์ มีดังนี้

ขั้นตอนที่ 1 หาค่าความถี่สะสมของแต่ละระดับสีเทาตั้งแต่ระดับ 0–255 แสดงโดยกราฟฮิสโตแกรม

ขั้นตอนที่ 2 นำช่วงระดับสีเทาตั้งแต่ 0 – 255 ของกราฟฮิสโตแกรมที่ได้จากการหาค่าความถี่สะสมของระดับสีเทา ไปหาค่าเธร็ชโฮลด์จะได้ค่าเท่ากับ T_1

ขั้นตอนที่ 3 นำช่วงระดับสีเทาตั้งแต่ T_1 – 255 ของกราฟฮิสโตแกรมที่ได้จากการหาค่าความถี่สะสมของระดับสีเทา ไปหาค่าเธร็ชโฮลด์จะได้ค่าเท่ากับ T_2

ขั้นตอนที่ 4 ค่าเธร็ชโฮลด์ ที่นำไปใช้ หาได้โดยแทนค่าตามสมการต่อไปนี้

$$T = T_1 + \left[\frac{T_2 - T_1}{4}\right]$$

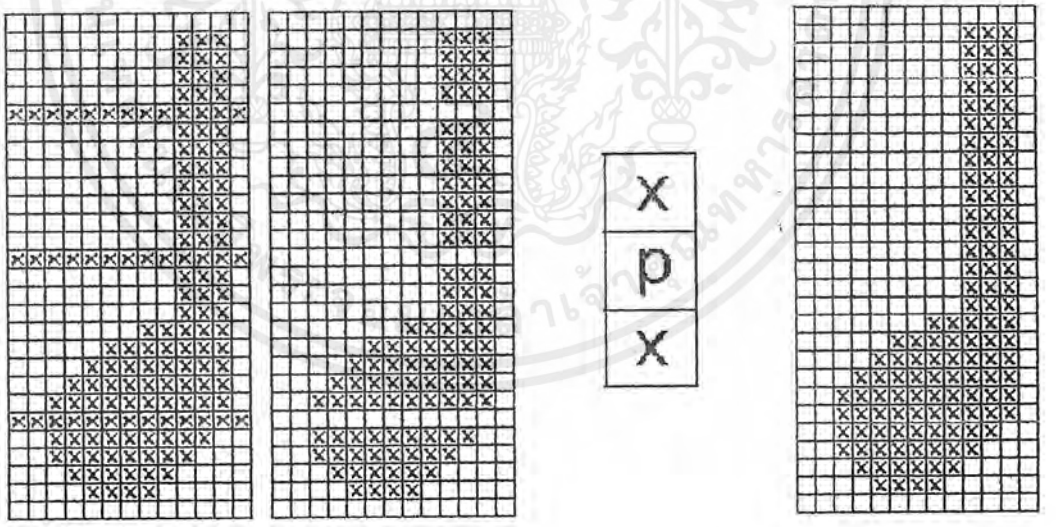
3.4 การลบบรรทัด 5 เส้น

หลังจากทำภาพ 2 ระดับ ขั้นตอนต่อไปคือ การแยกตัวโน้ตออกจากบรรทัด 5 เส้น โดยลบบรรทัด 5 เส้นออกจากภาพ วิธีการหาบรรทัด 5 เส้นคือ หาเส้นตรงเส้นที่ยาวที่สุดในแนวนอน โดยเก็บค่าความถี่ของจุดดำในแต่ละแถวไว้ แต่เนื่องจากบรรทัด 5 เส้นอาจมีความยาวไม่เท่ากัน อันเป็นผลมาจากการสแกนภาพตัวพิมพ์โน้ต จึงควรตรวจสอบว่าเส้นใดมีค่าความถี่เกิน 80 % ของเส้นที่ยาวที่สุดให้ถือว่าเป็นบรรทัด 5 เส้น แล้วเก็บตำแหน่งเอาไว้

รูปที่ 3.4 ภาพสแกนโน้ตดนตรีที่มี noise และ เส้นน้อย

ตัวพิมพ์โน้ตดนตรีบางแผ่น จะมีเส้นน้อยอยู่ด้วยซึ่งเราจะต้องลบเส้นน้อยออก วิธีการลบเส้นน้อยคือ ส่วนใหญ่ระยะห่างระหว่างบรรทัด 5 เส้น กับเส้นน้อยทั้งบนและล่างจะมีระยะห่างที่เท่ากัน จึงใช้การหาระยะห่างระหว่างเส้นของบรรทัด 5 เส้นแทน แล้วนำระยะห่างที่ได้นี้บวกเพิ่ม 4 ครั้ง จากเส้นที่อยู่ล่างสุดของบรรทัด 5 เส้น และลบออก 4 ครั้งจากเส้นที่อยู่บนสุดของบรรทัด 5 เส้น เมื่อได้ตำแหน่งของเส้นบรรทัด 5 เส้นและตำแหน่งเส้นน้อยแล้วจากนั้นลบบรรทัด 5 เส้นและเส้นน้อยออกทั้งหมด

วิธีการลบบรรทัด 5 เส้นและเส้นน้อยออก จะลบบรรทัด 5 เส้นและเส้นน้อยที่ได้เก็บตำแหน่งไว้แล้วทั้งหมด แต่เนื่องจากเส้นของบรรทัด 5 เส้นและเส้นน้อยที่ผ่านการทำภาพ 2 ระดับ ภาพบางช่วงของเส้นจะมี noise อยู่จึงต้องลบเพิ่มอีกข้างละ 1 พิกเซล ทั้งด้านบนและด้านล่างของเส้นเพื่อลบ noise ออก



(a) การลบบรรทัด 5 เส้นออก (b) การคืนส่วนของตัวโน้ต

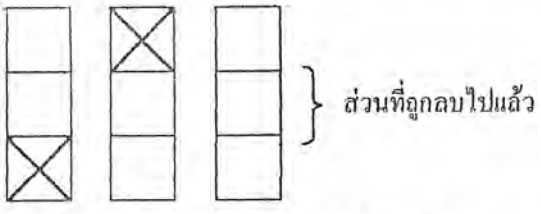
รูปที่ 3.5 แสดงการลบบรรทัด 5 เส้นและการคืนส่วนของตัวโน้ตที่ลบ

จะเห็นได้ว่าการลบบรรทัด 5 เส้นจะทำให้ภาพตัวโน้ตบางส่วนขาดหายไป จึงต้องคืนส่วนของโน้ตที่ถูกลบไป การคืนภาพเราจะพิจารณาจาก พิกเซลด้านบน และ พิกเซลด้านล่างของตำแหน่งที่ลบว่าเป็นสีดำทั้งคู่หรือไม่ ถ้าเป็นสีดำทั้งข้างบนและข้างล่างก็จะคืนส่วนที่ลบทั้งหมด ถ้าพิกเซลด้านใดด้านหนึ่งเป็นสีขาวก็ไม่ต้องคืนส่วนที่ลบไป

สรุปขั้นตอนการลบบรรทัด 5 เส้นมีดังนี้

1. หาค่าความถี่ในแนวแกนนอนของภาพเพื่อหาความถี่สูงสุดของจุดดำ
2. พิจารณาจากความถี่ที่ได้จากข้อ 1. ว่าแถวใดมีความถี่เกิน 80 % ของค่าความถี่สูงสุด จะกำหนดให้เป็นบรรทัด 5 เส้น และเก็บตำแหน่งนั้นเอาไว้
3. หาเส้นน้อย โดยหาระยะระหว่างเส้นของบรรทัด 5 เส้นก่อน เมื่อได้ระยะของบรรทัด 5 เส้นแล้วทำการบวกเพิ่มจากบรรทัด 5 เส้นจากเส้นล่างสุดไป 4 ครั้ง และลบจากเส้นบนสุดไป 4 ครั้ง กำหนดให้เป็นเส้นน้อยและเก็บตำแหน่งไว้
4. ลบบรรทัด 5 เส้นและเส้นน้อยออกทั้งหมด โดยจะลบเพิ่มทั้งด้านบนและด้านล่างของบรรทัด 5 เส้นและเส้นน้อยอีก 1 พิกเซล เพื่อลบ noise ที่เกิดกับบรรทัด 5 เส้นและเส้นน้อย
5. หลังจากลบบรรทัด 5 เส้นและเส้นน้อยออกแล้วจะเห็นว่าตัวโน้ตบางส่วนขาด จึงต้องทำการคืนภาพโน้ตที่ขาดไป โดยพิจารณาจาก พิกเซลด้านบน และ พิกเซลด้านล่างของตำแหน่งที่ลบว่าเป็นสีดำทั้งคู่หรือไม่ ถ้าเป็นสีดำทั้งข้างบนและข้างล่างก็จะคืนส่วนที่ลบทั้งหมด ถ้าพิกเซลด้านใดด้านหนึ่งเป็นสีขาวก็ไม่คืนส่วนที่ลบไป

(a) รูปแสดงลักษณะที่ต้องมีการคืนภาพ

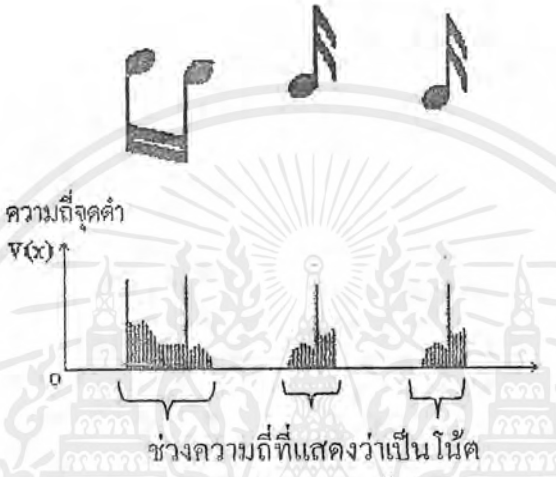


(b) รูปแสดงลักษณะที่ไม่ต้องมีการคืนภาพ

รูปที่ 3.6 แสดงลักษณะการคืนภาพ

3.5 การหาความถี่ในแนวตั้ง

หลังจากได้ภาพตัวโน้ตที่ผ่านการลบบรรทัด 5 เส้นแล้ว จะเหลือแต่เพียงตัวโน้ตเท่านั้น ภาพที่ได้นี้จะนำมาทำการหาความถี่ในแนวแกนตั้ง จะนำความถี่ที่ได้มาพิจารณาหาช่วงของตัวโน้ตแต่ละตัว เพื่อนำไปหากรอบ นอกจากนี้ยังนำไปใช้พิจารณาคุณสมบัติของโน้ตว่าโน้ตที่พบเป็น โน้ตที่เป็นชุด (โน้ตเขบ็ตติดกัน) หรือ โน้ตเดี่ยวที่มีขา (ตัวขาว, ตัวดำ, ตัวเขบ็ตเดี่ยว) หรือโน้ตที่ไม่มีขา (ตัวกลม, ตัวหยุดชนิดต่างๆ) : V(x) คือ Vertical projections



รูปที่ 3.7 ช่วงความถี่แนวแกนตั้งที่แสดงว่าเป็นโน้ต

(a) โน้ตชุด

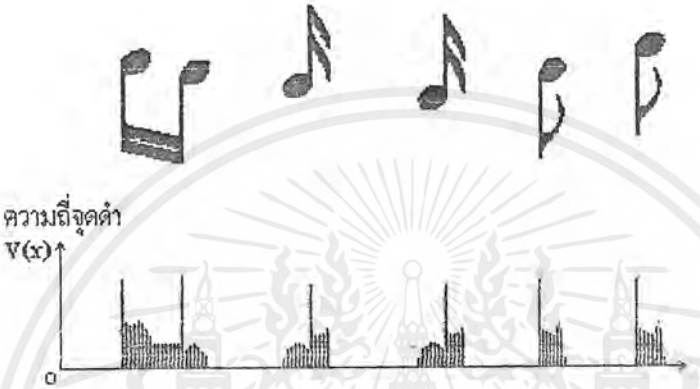
(b) โน้ตเดี่ยวมีขา

(c) โน้ตเดี่ยวไม่มีขา

รูปที่ 3.8 โน้ตในลักษณะต่างๆ

การหาความถี่ในแนวตั้ง

เมื่อได้ภาพที่ทำการลบบรรทัด 5 เส้นแล้วจะนำภาพเฉพาะช่วงของบรรทัด 5 เส้นและเส้นน้อยที่อยู่ระหว่างบรรทัด 5 เส้นชุดนั้นๆ มาทำการหาความถี่ในแนวตั้งจนครบทุก Column ของภาพ ซึ่งช่วงที่พบตัวโน้ตจะเกิดความถี่ของจุดดำขึ้น และช่วงที่ไม่มีตัวโน้ตจะไม่เกิดความถี่ของจุดดำ ดังนั้นเราสามารถทราบช่วงความกว้าง และความถี่ของตัวโน้ตในแนวแกนตั้งได้ดังรูปที่ 3.9

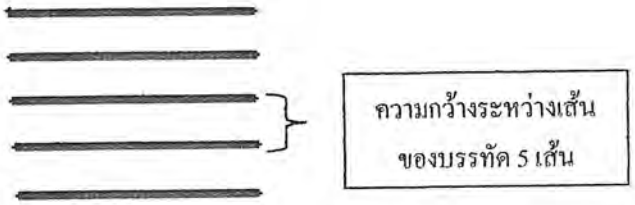


รูปที่ 3.9 ความถี่ของตัวโน้ตในแนวตั้ง

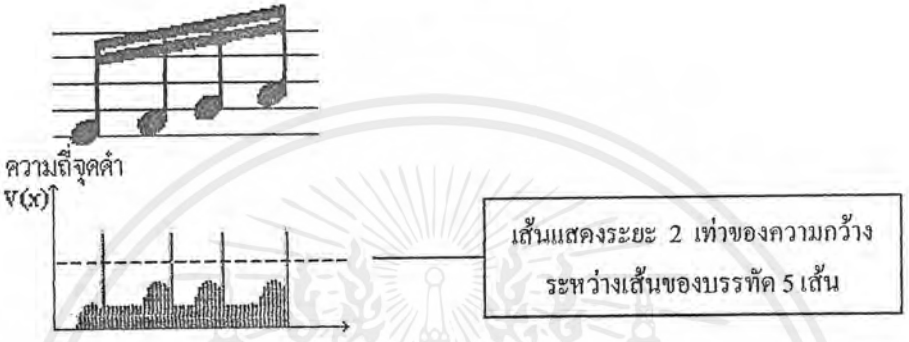
3.5.1 การพิจารณาคุณสมบัติของโน้ต (โน้ตชุด, โน้ตเดี่ยวที่มีขา, โน้ตเดี่ยวที่ไม่มีขา)

นำเอาช่วงความถี่ในแนวแกนตั้งของโน้ตแต่ละตัวที่หาได้มาพิจารณาหาว่าโน้ตแต่ละตัวมีคุณสมบัติเป็นเช่นไร โดยแบ่งคุณสมบัติของโน้ตเป็นดังต่อไปนี้

คุณสมบัติของโน้ต	ลักษณะของความถี่ (ผลของช่วงความถี่ที่ได้ของโน้ตแต่ละตัว)
1. โน้ตชุด	จะมีความถี่เกิน 2 เท่าของความกว้างระหว่างเส้นของบรรทัด 5 เส้น มากกว่า 1 ครั้ง
2. โน้ตเดี่ยวที่มีขา	จะมีความถี่เกิน 2 เท่าของความกว้างระหว่างเส้นของบรรทัด 5 เส้น เท่ากับ 1 ครั้ง
3. โน้ตเดี่ยวที่ไม่มีขา	จะมีความถี่ไม่ เกิน 2 เท่าของความกว้างระหว่างเส้นของบรรทัด 5 เส้น



รูปที่ 3.10 บรรทัด 5 เส้น



ก) โน้ตเดี่ยวมีขาที่มีความถี่เกินระยะมากกว่า 1 ครั้ง

ข) โน้ตเดี่ยวมีขาที่มีความถี่เกินระยะ 1 ครั้ง

ค) โน้ตเดี่ยวไม่มีขาที่มีความถี่ไม่เกินระยะ

รูปที่ 3.11 รูปโน้ตในลักษณะต่างๆ

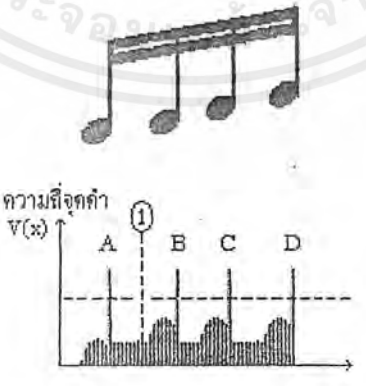
3.5.2 การแยกโน้ตที่เป็นชุด

เมื่อหาความถี่ในแนวแกนตั้งแล้ว พบว่าช่วงโน้ตนั้นมีคุณสมบัติเป็นโน้ตชุด เราจะแยกโน้ตชุดนั้นออกจากกัน มีวิธีดังต่อไปนี้

1. พิจารณาโน้ตชุด ว่ามีค่าความถี่ที่เกินระยะความกว้าง 2 ช่องระหว่างเส้นของบรรทัด 5 เส้นกี่ครั้ง ให้ถือว่าจำนวนโน้ตที่แยกได้จากโน้ตชุด มีจำนวนตัวโน้ตเท่ากับจำนวนครั้งของความถี่ที่เกินระยะ ดังรูปที่ 3.12 มีจุดที่เกินระยะ อยู่ 4 ครั้ง

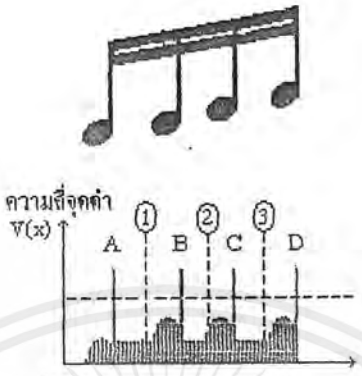
รูปที่ 3.12 แสดงโน้ตชุดที่มีเส้นความถี่ที่เกิน 2 ช่อง
ระหว่างเส้นของบรรทัด 5 เส้น 4 เส้น

2. พิจารณาดังรูปที่ 3.13 นำจุดที่มีค่าความถี่ที่เกินเส้นระยะ 2 เท่าของความกว้างระหว่างเส้นของบรรทัด 5 เส้น จุดแรกคือ จุด A กับจุดที่สองคือ จุด B มาทำการลบกัน จุด B – จุด A แล้วหารด้วย 2 ก็จะได้จุดที่แบ่งโน้ตตัวแรกกับโน้ตตัวที่สอง ก็คือ จุด 1 ดังรูปที่ 3.13



รูปที่ 3.13 แสดงการแบ่งโน้ต

3. ทำตามข้อที่ 2. กับจุด B และ จุด C จนแยกโน้ตครบทุกตัว โดย นำจุด C – B แล้วหาร 2 ก็จะได้จุดแบ่งโน้ตที่จุด 2 และนำจุด D – C แล้วหาร 2 ก็จะได้จุดแบ่งโน้ตที่จุด 3 ดังรูปที่ 3.14



รูปที่ 3.14 แสดงการแบ่งโน้ตส่วนที่เหลือ

ผลของการแบ่งตัวโน้ตแล้วจะมีลักษณะดังรูป

รูปที่ 3.15 แสดงรูปโน้ตหลังจากการแบ่ง

3.6 การหาโครงกระดูก (Skeleton)

วิธีการหาโครงร่างมีหลายวิธีการ และแต่ละวิธีก็ใช้งานแตกต่างกัน ซึ่งเราต้องเอาวิธีใดวิธีหนึ่งมาใช้ในโครงงานนี้ ในโครงงานนี้ใช้วิธีการของ Zhang และ Suen ซึ่งได้พัฒนาขึ้นในปี ค.ศ. 1984 ใช้สำหรับ Thinning binary regions หมายถึงภาพที่มีลักษณะที่บาง และข้อมูลที่เก็บจะมีลักษณะเป็น binary คือ จุดที่เป็นภาพจะมีค่าเป็น 1 และจุดที่เป็นฉากหลังจะมีค่าเป็น 0 ซึ่งวิธีการของ Skeleton นี้จะทำให้เราได้โครงร่างที่มีความหนา 1 พิกเซลอยู่ที่กึ่งกลางของภาพ

วิธีการมี 2 ขั้นตอน ซึ่งจะใช้การ Contour point คือ การพิจารณาจุดรอบข้างทั้ง 8 จุด (ดังรูป 3.16 จุดรอบข้างคือจุด P1 – P8) โดย Contour point เป็นจุดที่มีค่าเป็น 1 และมีจุดรอบๆอย่างน้อย 1 จุดที่มีค่าเป็น 0 ในการตรวจสอบจุดภาพทำโดยนำเมตริกซ์ขนาด 3x3 ไปวางครอบจุดที่ต้องการตรวจสอบ(P0) แล้วใช้จุดที่อยู่รอบๆจุดที่ต้องการตรวจสอบ (P1 - P8) เป็นตัวพิจารณา ดังรูปที่ 3.16 เป็นเมตริกซ์ขนาด 3x3 ที่นำไปครอบจุดที่ต้องการตรวจสอบ

P8	P1	P2
P7	P0	P3
P6	P5	P4

รูปที่ 3.16 เมตริกซ์ขนาด 3x3

<u>ขั้นตอนของการตรวจสอบจุดของภาพ</u>

ขั้นตอนที่ 1

เมื่อเราใช้เมตริกซ์ ขนาด 3x3 ไปครอบจุดที่ต้องการตรวจสอบและจุดนั้นจะถูกลบเมื่อเงื่อนไขต่อไปนี้ เป็นจริง

$$\left.\begin{array}{ll} \text{(a)} & 2 \le N(P0) \le 6 \\ \text{(b)} & S(P0) = 1 \\ \text{(c)} & P1 * P3 * P5 = 0 \\ \text{(d)} & P3 * P5 * P7 = 0 \end{array}\right\} \ldots\ldots\ldots\ldots (3.5)$$

โดยที่ N(P0) คือ จำนวนจุดรอบ P0 ที่มีค่าเป็น 1 เราจะได้

$$N(P0) = P1 + P2 + \ldots + P7 + P8 \qquad (3.6)$$

และ S(P0) คือ จำนวนการเปลี่ยนแปลงจาก 0 ไปเป็น 1 ในลำดับของ P1, P2, ..., P8 ตัวอย่างในรูปที่ 3.17 N(P0) = 4 และ S(P0) = 3

0	0	1
1	P0	0
1	0	1

รูปที่ 3.17 แสดงการตรวจสอบเงื่อนไข

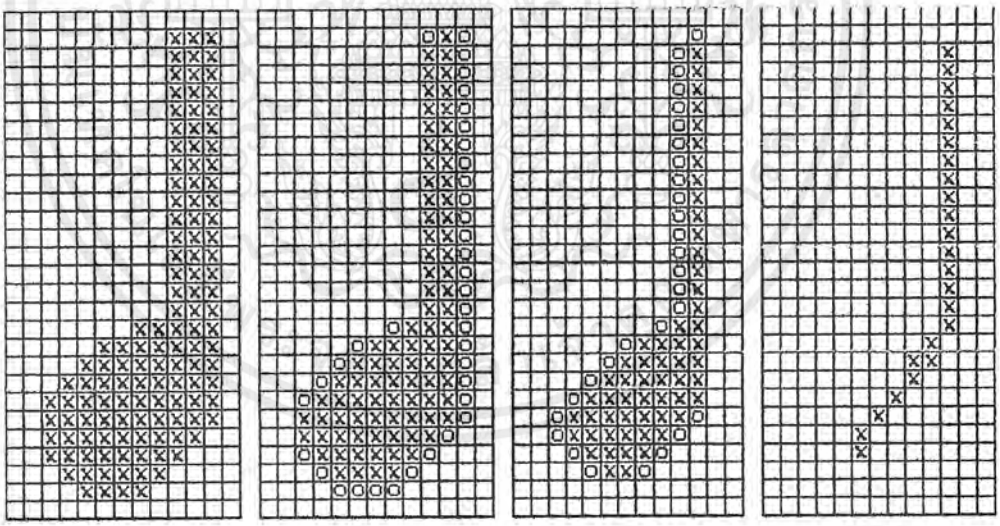
ขั้นตอนที่ 2

สำหรับขั้นตอนนี้ จะได้เงื่อนไข (a) และ (b) ในการตรวจสอบจะเหมือนขั้นตอนแรก แต่เงื่อนไข (c) และ (d) จะถูกเปลี่ยนไป

$$\left.\begin{array}{l} \text{(a)}\ \ 2 \le N(P0) \le 6 \\ \text{(b)}\ \ S(P0) = 1 \\ \text{(c)}\ \ P1 * P3 * P7 = 0 \\ \text{(d)}\ \ P1 * P5 * P7 = 0 \end{array}\right\} \quad \ldots\ldots\ldots\ldots \quad (3.7)$$

วิธีการ

จากขั้นตอนที่ 1 เราจะตรวจสอบทุกจุดในภาพ เพื่อหาจุดที่มีเงื่อนไขเป็นจริงตาม (a) ถึง (d) เมื่อพบจุดที่ตรงตามเงื่อนไข ก็ทำสัญลักษณ์ (mark) ที่จุดนั้นก่อน ตรวจสอบจนกว่าจะครบทุกจุดในภาพ จึงทำการลบจุดที่ทำสัญลักษณ์ไว้ทั้งหมด จากนั้นทำการตรวจสอบใหม่ทั้งภาพตามเงื่อนไขในขั้นตอนที่ 2 และลบจุดที่ทำสัญลักษณ์ไว้ทั้งหมดเช่นกัน ทำซ้ำในขั้นตอนที่ 1 และ 2 ใหม่จนกว่าจะไม่มีจุดที่ตรงตามเงื่อนไขใดๆเลย ซึ่งก็จะได้โครงร่างของภาพที่มีลักษณะเป็นเส้นขนาดเท่ากับ 1 พิกเซลที่อยู่กึ่งกลางของภาพ ดังรูปที่ 3.18



(a) ภาพตัวโน้ตก่อนทำให้บาง

(b) ผลจากขั้นตอนที่ 1 ในการประมวลผลรอบแรก

(c) ผลจากขั้นตอนที่ 2 ในการประมวลผลรอบแรก

(d) ตัวโน้ตที่ได้จากการประมวลผลรอบสุดท้าย

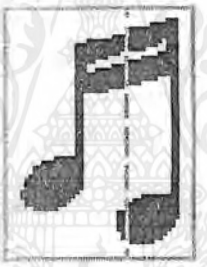
รูปที่ 3.18 แสดงการหาโครงกระดูก

3.7 การตีกรอบตัวโน้ต

ก่อนที่จะทำขั้นตอนการตีกรอบเราจะต้องหาความถี่ในแนวแกนตั้งและแนวนอนก่อน ส่วนความถี่ในแนวแกนตั้งได้กล่าวไว้แล้วในหัวข้อ 3.4 ในที่นี้จะกล่าวเฉพาะการหาความถี่ในแนวแกนนอน

การหาความถี่ในแนวนอน

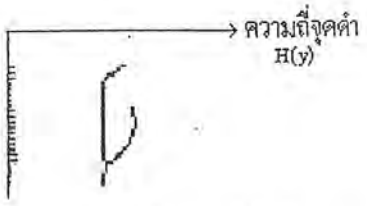
เมื่อทราบระยะความกว้างของโน้ตแต่ละตัวที่ได้จากการหาความถี่ในแนวตั้งแล้ว จะนำมาหาความถี่ในแนวนอน ตัวโน้ตที่จะนำมาหาความถี่ในแนวแกนนอนจะต้องเป็นตัวโน้ตที่ผ่านการทำโครงกระดูกแล้ว เหตุที่ต้องทำตัวบางก่อนการหาความถี่ในแนวนอนเนื่องจากโน้ตชุดที่เป็นลักษณะดังรูปที่ 3.19 เมื่อแยกโน้ตชุดออกจากกันแล้วจะมีบางส่วนของโน้ตตัวที่สองมาอยู่ในขอบเขตของโน้ตตัวแรก เมื่อนำไปหาความถี่ในแนวตั้งเลยโดยยังไม่ได้ทำตัวบางจะมีลักษณะดังรูปที่ 3.20 เมื่อนำช่วงขอบเขตที่ได้จากการหาความถี่ในแนวตั้งไปหากรอบของโน้ต กรอบที่ได้จะมีขนาดใหญ่กว่าโน้ตจริงมาก และเมื่อนำไปหาลักษณะเด่นของโน้ตจะทำให้เกิดการผิดพลาดได้



รูปที่ 3. 19 แสดงโน้ตชุดที่มีปัญหา

รูปที่ 3.20 กรอบจะใหญ่กว่าขนาดจริง

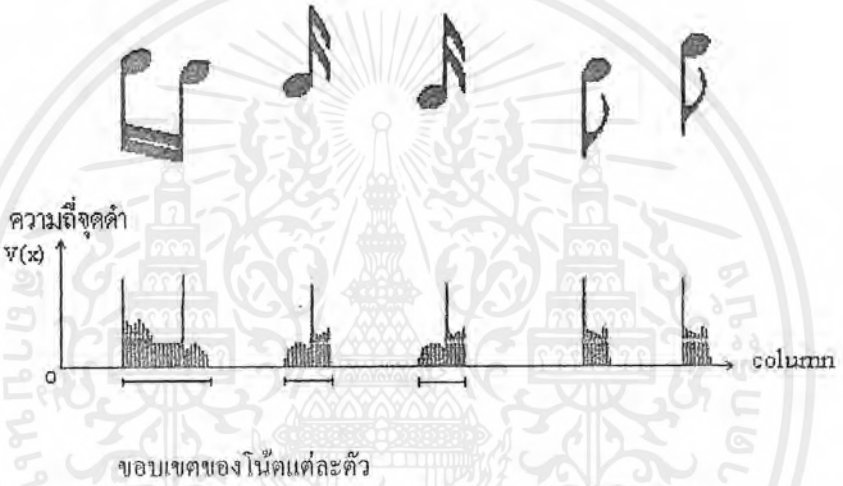
วิธีการหาความถี่ในแนวนอน โดยนำระยะความกว้างของตัวโน้ตที่หาได้จากการหาความถี่ในแนวแกนตั้งของโน้ตแต่ละตัว มาทำการหาความถี่ของจุดดำในแนวนอนจนครบทุกแถว (Row) ของช่วงบรรทัด 5 เส้นที่พิจารณา ซึ่งช่วงที่พบจุดดำก็คือ ขนาดความสูงของตัวโน้ต ทำเช่นนี้กับโน้ตทุกตัวที่ได้จากการหาความถี่ในแนวตั้ง ดังรูปที่ 3.21



รูปที่ 3.21 ความถี่ของตัวโน้ตในแนวนอน

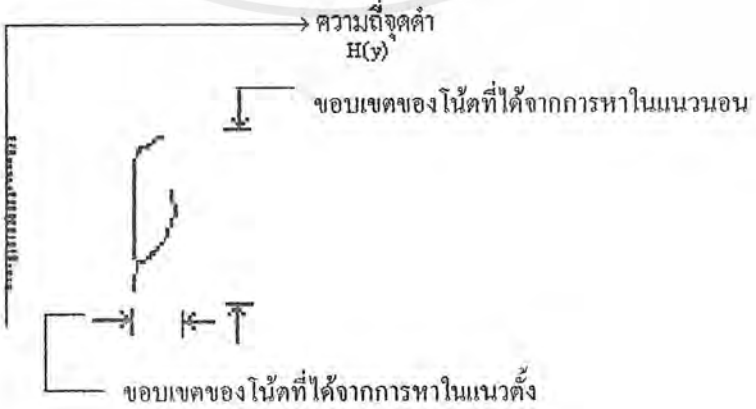
ผลจากการหาความถี่ในแนวตั้งและแนวนอน ทำให้สามารถหาขอบเขตของโน้ตได้ จึงนำมาใช้ในการหากรอบของโน้ตแต่ละตัวเพื่อที่จะหาลักษณะเด่นของโน้ตต่อไป โดยมีขั้นตอนดังนี้

ขั้นตอนที่ 1 หาช่วงความถี่ในแนวแกนตั้งเพื่อหาช่วงที่เป็นตัวโน้ตแต่ละตัว



รูปที่ 3.22 หาขอบเขตของโน้ตในแนวแกนตั้งแต่ละตัว

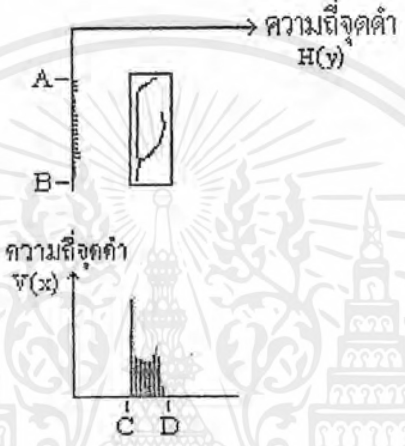
ขั้นตอนที่ 2 หาช่วงความถี่ในแนวแกนนอน โดยนำช่วงความถี่ในแนวแกนตั้งของโน้ตตัวนั้นมาเป็นขอบเขตในการหาความถี่ในแนวนอนและภาพตัวโน้ตที่จะหาความถี่ในแนวแกนนอนต้องผ่านการทำโครงกระดูกมาแล้ว



รูปที่ 3.23 ขอบเขตของโน้ตในแนวแกนนอนและ
ภาพต้องผ่านการทำโครงกระดูกมาแล้ว

ขั้นตอนที่ 3

- ผลจากการหาความถี่ในแนวแกนตั้ง ช่วงที่มีความถี่ของจุดดำ เป็นความกว้างของตัวโน้ต คือ จุด C ถึง จุด D
- ผลจากการหาความถี่ในแนวแกนนอน ช่วงที่มีความถี่ของจุดดำ เป็นความสูงของตัวโน้ต คือ จุด A ถึง จุด B
- ในการตีกรอบตัวโน้ตเราต้องเผื่อจากจุดเดิมคือ จุด A B C และ D ข้างละ 1 พิกเซล เพื่อไม่ให้ตัวโน้ตติดกับกรอบ ดังรูปที่ 3.24



รูปที่ 3.24 การหากรอบตัวโน้ต

3.8 กาหาลักษณะเด่นของตัวโน้ต

ก่อนที่เราจะหาลักษณะเด่นของโน้ตได้นั้น ตัวโน้ตต้องผ่านขั้นตอนการทำ Skeleton มาแล้วทั้งนี้ก็เพื่อให้เหลือเฉพาะโครงร่างของโน้ตเท่านั้น โดยโครงร่างของโน้ตแต่ละตัวจะมีลักษณะเด่นที่ไม่เหมือนกันซึ่งลักษณะเด่นของตัวโน้ต คือ จุดตัด, จุดแยก และจุดปลาย วิธีการหาลักษณะเด่นทำโดยการใช้หน้าต่าง 3x3 ครอบจุดที่ต้องการตรวจสอบในโครงร่างของตัวโน้ตนั้น ว่าจุดที่อยู่กลางกรอบหน้าต่างมีลักษณะเด่นเป็นอย่างไร ซึ่งหน้าต่างที่จะใช้เป็นเครื่องมือในการหาลักษณะเด่นของตัวโน้ตแสดงดังรูปที่ 3.25

X4	**X3**	**X2**
X5	**X0**	**X1**
X6	**X7**	**X8**

รูปที่ 3.25 หน้าต่าง 3x3

ในการหาลักษณะเด่น จะใช้จุด X0 เป็นจุดที่จะบอกว่ามีลักษณะเด่นเป็นอย่างไร โดยพิจารณาจุดที่อยู่ล้อมรอบจุด X0 ทั้ง 8 จุด ว่ามีเงื่อนไขเป็นอย่างไร ในการหา จุดตัด, จุดแยก และจุดปลายทำได้ดังนี้

จุดตัด เป็นจุดที่ส่วนประกอบรอบๆมีการเปลี่ยนแปลงจากค่า 1 ไปเป็น 0 อยู่ 4 ช่วง

จุดแยก เป็นจุดที่ส่วนประกอบรอบๆมีการเปลี่ยนแปลงจากค่า 1 ไปเป็น 0 อยู่ 3 ช่วง

จุดปลาย เป็นจุดที่ส่วนประกอบรอบๆมีการเปลี่ยนแปลงจากค่า 1 ไปเป็น 0 อยู่ 1 ช่วง

ตัวอย่าง จุดตัด, จุดแยก และจุดปลาย แสดงดังรูป

1	0	1
0	1	0
0	1	0

0	0	0
1	1	1
0	1	0

รูปตัวอย่างของจุดแยก

1	0	1
0	1	0
1	0	1

0	1	0
1	1	1
0	1	0

รูปตัวอย่างของจุดตัด

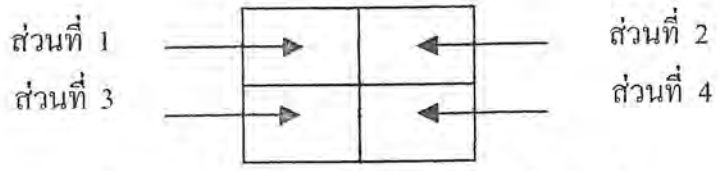
0	1	0
0	1	0
0	0	0

0	0	1
0	1	0
0	0	0

รูปตัวอย่างของจุดปลาย

3.9 การแบ่งกลุ่มของตัวโน้ต

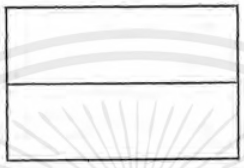
จากการหาลักษณะเด่นของตัวโน้ต ทำให้สามารถหาจำนวนและตำแหน่งของจุดตัด จุดแยก จุดปลายของตัวโน้ตแต่ละตัวเพื่อที่จะนำลักษณะเด่นของตัวโน้ตแต่ละตัวมาทำการจดจำรูปแบบโดยตัวโน้ตแต่ละตัวจะแบ่งออกเป็น 4 ส่วนดังรูป



รูปที่ 3.26 การแบ่งส่วนของตัวโน้ตออกเป็น 4 ส่วน

วิธีการแบ่งกรอบของตัวโน้ตแต่ละตัวออกเป็น 4 ส่วนคือ

1. การแบ่งกรอบตัวโน้ตในแนวนอน จะแบ่งครึ่งของความสูงของกรอบที่หาได้จากขั้นตอนข้างต้น



รูปที่ 3.27 การแบ่งกรอบแนวนอน

2. การแบ่งกรอบตัวโน้ตในแนวตั้ง จะแบ่งได้ 2 ลักษณะตามคุณสมบัติของโน้ตในหัวข้อ 3.5.1 การพิจารณาคุณสมบัติของตัวโน้ต คือ
   - โน้ตที่ไม่มีขา (โน้ตที่มีความถี่ไม่เกินระยะ 2 เท่าของความกว้างระหว่างเส้นของบรรทัด 5 เส้น) เช่น ตัวกลม ตัวหยุดชนิดต่างๆ จะแบ่งครึ่งความกว้างของกรอบ

รูปที่ 3.28 การแบ่งกรอบตามแนวตั้งของโน้ตที่ไม่มีขา

   - โน้ตที่มีขา (โน้ตที่มีความถี่เกินระยะ 2 เท่าของความกว้างระหว่างเส้นของบรรทัด 5 เส้น) เช่น ตัวดำ ตัวขาว ตัวเขบ็ต จะแบ่งโดยนำตำแหน่งของขา(ตำแหน่งของโน้ตที่มีความถี่ไม่เกินระยะ 2 เท่าของความกว้างระหว่างเส้นของบรรทัด 5 เส้น) โน้ตที่ได้ แล้วขยับไปทางขวา 4 พิกเซล ดังรูปที่ 3.29

รูปที่ 3.29 การแบ่งกรอบตามแนวตั้งของโน้ตที่มีขา

แต่มีปัญหากับโน้ตที่มีขาบางตัวคือ เมื่อขยับจากตำแหน่งของขาโน้ตไปทางขวา 4 พิกเซล แล้วไม่สามารถแบ่งได้เนื่องจากสุดกรอบของโน้ต ก็ให้ขยับมาทางซ้าย 4 พิกเซล แทน ดังรูปที่ 3.30

รูปที่ 3.30 การแบ่งกรอบตามแนวตั้งของโน้ตที่มีขา

เมื่อแบ่งตัวโน้ตออกเป็น 4 ส่วนทำให้เราสามารถทราบจำนวนของจุดตัด จุดแยก จุดปลายที่อยู่ในแต่ละส่วนของตัวโน้ตแต่ละตัวก็จะสามารถแบ่งกลุ่มตัวโน้ตได้ โดยจะแบ่งกลุ่มของตัวโน้ตตามคุณสมบัติของตัวโน้ต (โน้ตชุด, โน้ตเดี่ยวที่มีขา, โน้ตเดี่ยวที่ไม่มีขา) ก่อนแล้วจึงแบ่งโน้ตตามลักษณะเด่น (จุดตัด, จุดแยก, จุดปลาย ของแต่ละช่อง) อีกครั้ง แบ่งออกได้เป็น 3 กลุ่มดังนี้

1. กลุ่มของตัวโน้ตเดี่ยวที่ไม่มีขาโน้ต ตัวอย่างเช่น

รูปโน้ต	ช่องที่	จุดตัด	จุดแยก	จุดปลาย	มี pixel
 ตัวกลม	1	0	0	0	1
	2	0	0	0	1
	3	0	0	0	1
	4	0	0	0	1

รูปโน้ต	ช่องที่	จุดตัด	จุดแยก	จุดปลาย	มี pixel
 ตัวชาร์ป	1	0	1	1	1
	2	0	1	1	1
	3	0	1	1	1
	4	0	1	1	1
รูปโน้ต	ช่องที่	จุดตัด	จุดแยก	จุดปลาย	มี pixel
 ตัวแฟลต	1	0	0	1	1
	2	0	0	0	0
	3	0	1	0	1
	4	0	0	0	1

2. กลุ่มของตัวโน้ตเดี่ยวที่มีขาโน้ต ตัวอย่างเช่น

รูปโน้ต	ช่องที่	จุดตัด	จุดแยก	จุดปลาย	มี pixel
 ตัวขาวหงาย	1	0	0	0	0
	2	0	0	1	1
	3	0	0	0	1
	4	0	1	0	1
รูปโน้ต	ช่องที่	จุดตัด	จุดแยก	จุดปลาย	มี pixel
 ตัวขาวคว่ำ	1	0	1	0	1
	2	0	0	0	1
	3	0	0	1	1
	4	0	0	0	0
รูปโน้ต	ช่องที่	จุดตัด	จุดแยก	จุดปลาย	มี pixel
 ตัวดำหงาย	1	0	0	0	0
	2	0	0	1	1
	3	0	0	1	1
	4	0	0	0	1

3. กลุ่มของตัวโน้ตชุด ตัวอย่างเช่น

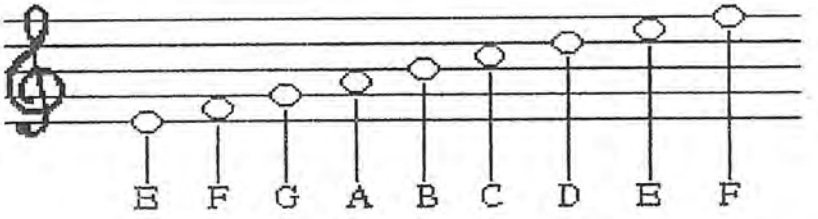
รูปโน้ต	ช่องที่	จุดตัด	จุดแยก	จุดปลาย	มี pixel
 ตัวเขบ็ต 1 ชั้นคว่ำ	1	0	0	0	1
	2	0	0	1	1
	3	0	0	0	1
	4	0	0	1	1
รูปโน้ต	ช่องที่	จุดตัด	จุดแยก	จุดปลาย	มี pixel
 ตัวเขบ็ต 1 ชั้นคว่ำ	1	0	0	0	1
	2	0	0	1	1
	3	0	0	1	1
	4	0	0	0	0
รูปโน้ต  ตัวเขบ็ต 1 ชั้นคว่ำ	ช่องที่	จุดตัด	จุดแยก	จุดปลาย	มี pixel
	1	0	0	0	1
	2	0	0	0	1
	3	0	1	1	1
	4	0	0	1	1

3.10 การหาตำแหน่งของตัวโน้ตบนบรรทัด 5 เส้น

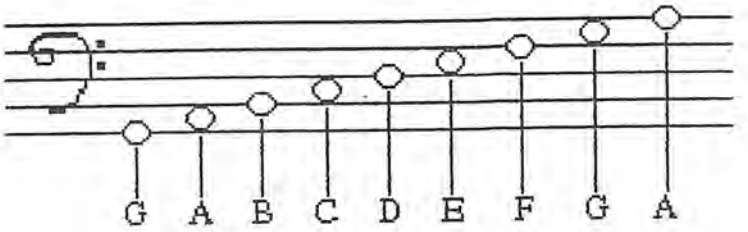
เมื่อเราทราบว่าตัวโน้ตแต่ละตัวเป็นโน้ตชนิดใด ก็จะทำให้รู้ความสั้น – ยาวของเสียง แต่เรายังไม่ทราบว่าเสียงนั้นมีระดับเสียงเป็นอย่างไร ระดับเสียงของตัวโน้ต หมายถึง ความสูง – ต่ำของระดับเสียง ซึ่งมีชื่อเรียกเรียงตามลำดับจากเสียงต่ำไปเสียงสูง รวม 7 ชื่อดังนี้

ชื่อระดับเสียง โด เร มี ฟา ซอล ลา ที

ชื่อตัวอักษร C D E F G A B



รูปที่ 3.30 แสดงระดับเสียงของกุญแจซอลของตัวโน้ตบนบรรทัด 5 เส้น



รูปที่ 3.31 แสดงระดับเสียงของกุญแจฟาของตัวโน้ตบนบรรทัด 5 เส้น

ระดับเสียงของตัวโน้ตแต่ละตัวสามารถหาได้จากตำแหน่งของตัวโน้ตบนบรรทัด 5 เส้น จากขั้นตอนการแยกตัวโน้ตออกจากบรรทัด 5 เส้น เราได้เก็บตำแหน่งของบรรทัด 5 เส้นเอาไว้ และจากขั้นตอนการตีกรอบแยกตัวโน้ตแต่ละตัวออกจากกันเราได้เก็บตำแหน่งของตัวโน้ตแต่ละตัวเอาไว้เช่นกัน เมื่อนำเอาตำแหน่งของตัวโน้ตไปเปรียบเทียบกับบรรทัด 5 เส้น ก็จะทำให้ทราบระดับเสียงของตัวโน้ตแต่ละตัว

ในการเอาตำแหน่งของตัวโน้ตไปเปรียบเทียบกับตำแหน่งของบรรทัด 5 เส้นจะเอาตำแหน่งตรงกลางของหัวโน้ตมาพิจารณาหาระดับเสียงของโน้ต

รูปที่ 3.32 แสดงระดับเสียงของตัวโน้ตบนบรรทัด 5 เส้น

การหากุญแจเสียงของบรรทัด 5 เส้น

จากต้นบรรทัด 5 เส้นทุกชุดจะมีกุญแจเสียงบอกอยู่ โดยจะพิจารณากุญแจเสียงจากโน้ตตัวที่ 2 (เนื่องจากตัวแรกจะเป็นเส้นกั้นห้อง) ถ้าโน้ตตัวที่ 2 มีความถี่ในแนวตั้งเกินความกว้างของบรรทัด 5 เส้น จะกำหนดให้บรรทัด 5 เส้นนั้นเป็นระดับเสียงของกุญแจซอล ถ้าไม่เกินความกว้างของบรรทัด 5 เส้น จะกำหนดให้บรรทัด 5 เส้นนั้นเป็นระดับเสียงของกุญแจฟา

3.11 การสร้างไฟล์รหัสมิดี้ (*.MID)

จากขั้นตอนการประมวลภาพและการจดจำรูปแบบตัวพิมพ์โน้ตดนตรี ทำให้เราทราบจังหวะและระดับเสียงของตัวโน้ตแต่ละตัว เพื่อนำไปเข้ารหัสสร้างเป็นไฟล์มาตรฐานของมิดี้ (*.MID) แต่ในที่นี้ ก่อนที่จะนำข้อมูลของจังหวะและระดับเสียงของโน้ตไปแปลงเป็นไฟล์มิดี้ เราต้องทำเป็นไฟล์รหัสข้อมูลเพื่อการตรวจเช็คความถูกต้องก่อนที่จะมีการแปลงเป็นไฟล์มิดี้ต่อไป โดยเรากำหนดให้ไฟล์รหัสข้อมูลของเราเป็นไฟล์ที่มีนามสกุลเป็น SCC (*.SCC) เป็นเลขฐาน 16 ซึ่งตัวโน้ตแต่ละตัวจะใช้ขนาดในการเก็บ 2 ไบต์ โดยแต่ละบิตมีความหมายดังนี้

High byte	Low byte			
8-15	5-7	4	3	0-2

บิตที่ 0 – 2 คือ บอกจังหวะของโน้ต

000 = ตัวกลม

001 = ตัวขวา

010 = ตัวดำ

011 = ตัวเขบ็ต 1 ชั้น

100 = ตัวเขบ็ต 2 ชั้น

101 = ตัวเขบ็ต 3 ชั้น

บิตที่ 3 คือ บอกว่ามีจุดตามหลังตัวโน้ตหรือไม่

0 = ไม่มีจุดตามหลัง

1 = มีจุดตามหลัง

บิตที่ 4 คือ บอกว่าเป็นตัวหยุดหรือไม่

0 = ไม่เป็นตัวหยุด

1 = เป็นตัวหยุด

บิตที่ 5 - 7 คือ เส้นโยงเสียง

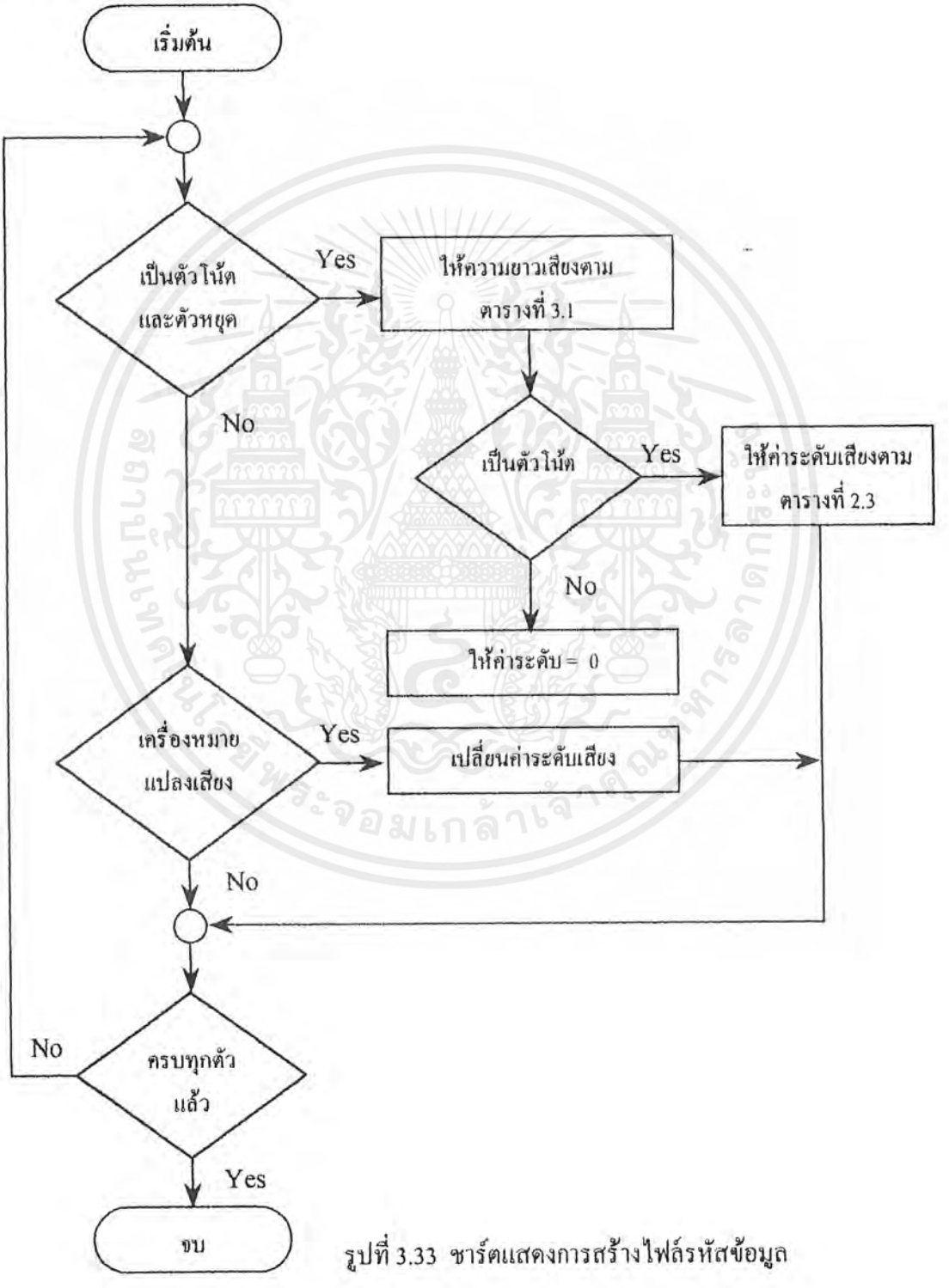
บิตที่ 8 – 15 คือ บอกระดับ เสียงของตัวโน้ตดังตารางที่ 2.3 เช่น โด = 60 (Hex. = 3C)

เมื่อเราได้ไฟล์รหัสข้อมูลตัวโน้ต (*.SCC) ที่เก็บระดับเสียงและความยาวเสียงของตัวโน้ตแต่ละตัวแล้ว เมื่อต้องการติดต่อกับอุปกรณ์ประเภทมิดี้ เราต้องทำการแปลงไฟล์รหัสข้อมูลตัว

ตัวโน้ต	ค่าความยาวของเสียง
ตัวกลม	384
ตัวขาว	192
ตัวดำ	96
ตัวเขบ็ต 1 ชั้น	48
ตัวเขบ็ต 2 ชั้น	24
ตัวเขบ็ต 3 ชั้น	12

ตารางที่ 3.1 แสดงค่าความยาวเสียงของตัวโน้ตและตัวหยุด

โน้ตให้เป็นไฟล์รหัสมิดี้ซึ่งได้กล่าวถึงโครงสร้างไฟล์มิดี้ไว้แล้วในหัวข้อ 2.2 ตัวอย่างเช่น ไฟล์รหัสข้อมูล 3C 02 ไบต์แรกคือ 3C เป็นเลขฐานสิบเท่ากับ 60 เมื่อไปเปรียบเทียบกับตารางที่ 2.3 ได้ระดับเสียงเท่ากับ C (โด) ไบต์ที่สองคือ 02 หมายถึงโน้ตตัวดำ และจะกำหนดให้มีค่าความยาวเสียงตามตารางที่ 3.1 ได้เท่ากับ 100 เมื่อได้ระดับเสียงและความยาวเสียงของโน้ตแล้วก็นำไปใส่ใน Track data ของไฟล์มิดี้ต่อไป



รูปที่ 3.33 ชาร์ตแสดงการสร้างไฟล์รหัสข้อมูล

ขั้นตอนการทำให้ข้อมูลที่ได้ให้เป็นไฟล์รหัสข้อมูลและไฟล์รหัสมิดี้

รูปที่ 3.34 ตัวอย่างโน้ตในการสร้างไฟล์รหัสข้อมูล

จากภาพโน้ตที่ผ่านขั้นตอนการจดจำรูปแบบตัวพิมพ์โน้ตดนตรี ได้โน้ตดังต่อไปนี้

1. ตัวเขบิ็ต 1 ชั้น ระดับเสียง โด
2. ตัวเขบิ็ต 1 ชั้น ระดับเสียง เร
3. ตัวดำ ระดับเสียง มี
4. ตัวขาว ระดับเสียง ซอล

สร้างเป็นไฟล์รหัสข้อมูลได้ดังนี้

ตัวที่ **1** ตัวเขบิ็ต 1 ชั้น ระดับเสียง โด แปลงเป็นไฟล์รหัสข้อมูลได้ 3C 03

- High byte = 3C คือ ตำแหน่งของตัวโน้ตที่อยู่ตรงบรรทัด 5 เส้นนี้ มี ระดับเสียง โด เมื่อเทียบกับตาราง 2.2 มีค่าตามรหัสมิดี้ เท่ากับ 3C
- Low byte = 03 คือตัวเขบิ็ต 1 ชั้น แสดงเป็นเลขฐานสอง ได้ 0000 0011 (ส่วนที่ขีดเส้นใต้หมายถึงตัวเขบิ็ต 1 ชั้น)

ตัวที่ **2** ตัวเขบิ็ต 1 ชั้น ระดับเสียง เร แปลงเป็นไฟล์รหัสข้อมูลได้ 3D 03

- High byte = 3D คือ ตำแหน่งของตัวโน้ตที่อยู่ตรงบรรทัด 5 เส้นนี้ มี ระดับเสียง เร เมื่อเทียบกับตาราง 2.2 มีค่าตามรหัสมิดี้ เท่ากับ 3D
- Low byte = 03 คือตัวเขบิ็ต 1 ชั้น แสดงเป็นเลขฐานสอง ได้ 0000 0011 (ส่วนที่ขีดเส้นใต้หมายถึงตัวเขบิ็ต 1 ชั้น)

ตัวที่ **3** ตัวดำ ระดับเสียง มี แปลงเป็นไฟล์รหัสข้อมูลได้ 40 02

- High byte = 40 คือ ตำแหน่งของตัวโน้ตที่อยู่ตรงบรรทัด 5 เส้นนี้ มี ระดับเสียง มี เมื่อเทียบกับตาราง 2.2 มีค่าตามรหัสมิดี้ เท่ากับ 40
- Low byte = 02 คือ แสดงเป็นเลขฐานสอง ได้ 0000 0010 (ส่วนที่ขีดเส้นใต้หมายถึง ตัวดำ)

ตัวที่ **4** ตัวขาว ระดับเสียง ซอล แปลงเป็นไฟล์รหัสข้อมูลได้ 43 01

- High byte = 43 คือ ตำแหน่งของตัวโน้ตที่อยู่ตรงบรรทัด 5 เส้นนี้ มี ระดับเสียง ซอล เมื่อเทียบกับตาราง 2.2 มีค่าตามรหัสมิดี้ เท่ากับ 43
- Low byte = 02 คือ แสดงเป็นเลขฐานสอง ได้ 0000 0001 (ส่วนที่ขีดเส้นใต้หมายถึง ตัวขาว)

เมื่อสร้างเป็นไฟล์รหัสข้อมูลได้ทั้งหมดเป็น **3C 03 3E 03 40 02 43 01**

การแปลงไฟล์รหัสข้อมูลให้เป็นไฟล์มิดี้

การแปลงไฟล์รหัสข้อมูลให้เป็นไฟล์มิดี้ จะนำข้อมูลของไฟล์รหัสข้อมูลมาใส่ในส่วนของ Track data ของ Track chunk ที่ 2 ของไฟล์มิดี้เท่านั้น ส่วนที่เป็น Header Chunk และ Track chunk ที่ 1 ดูรายละเอียดการใส่ค่าที่ต่างๆในหัวข้อ 2.2

ส่วนของ header chunk และ track chuk แรก

```
4D 54 68 64 00 00 00 06 00 01 00 02 00 78 4D 54
72 6B 00 00 00 13 00 FF 58 04 04 02 18 08 00 FF
51 03 07 A1 20 00 FF 2F 00 4D 54 72 6B 00 00 00
8D 00 C0 21
```

ส่วนของ track chunk 2 เป็นข้อมูลที่ได้จากไฟล์รหัสข้อมูล

```
            00 90 3C 40 30 3C 00 00 3E 40 30 3E 00
00 40 40 60 40 00 00 43 40 C0 43 00 00 FF 2F 00
```

จะนำไฟล์รหัสข้อมูลมาเป็นข้อมูล Note On และ Note Off ของTrack chunk ที่ 2 ของไฟล์มิดี้ รายละเอียดของ Note On ,Note Off ประกอบไปด้วย 3 ส่วน คือ delta-time , ระดับเสียงที่เล่น (จากตาราง 2.2) , ความหนักของโน้ต ตัวอย่างเช่น

Note On = 00 3C 40 หมายถึง เล่นโน้ตโด (3C) ด้วยความหนัก 40 ทันที(00)

Note Off = 30 3C 00 หมายถึง เล่นโน้ต โด (3C) ด้วยความหนัก 00 หลังจากเวลาผ่านไป ครึ่งหนึ่งของ 1 จังหวะ(30 เทียบจากตาราง 3.1 เป็นตัวเขบ็ต 1 ชั้น)

ขั้นตอนการแปลงไฟล์รหัสข้อมูลให้เป็นไฟล์มิดี้

1. อ่านข้อมูลไฟล์รหัสข้อมูลมาทีละ 2 ไบต์จากตัวอย่างแรก 3C 03
2. สามารถเขียนเป็นข้อมูล Note On และ Note Off ได้ดังนี้

   Note On ได้ 00 3C 40

   00 เริ่มเล่นโน้ตทันที

   3C ได้จากไฟล์รหัสข้อมูลไบต์แรก เป็นโน้ตโด

   40 เป็นค่าความหนักของการเล่นโน้ต เนื่องจากโน้ตที่ได้(03)ไม่ใช้ตัวหยุด(ดูได้จากโครงสร้างไฟล์รหัสข้อมูล)ถ้าเป็นตัวหยุด ความหนักเท่ากับ 00

Note Off ได้ 30 3C 00

30 เริ่มเล่นโน้ตหลังจากเวลาผ่านไป ครึ่งหนึ่งของ 1 จังหวะ (ตามตาราง 3.1 ค่านี้จะได้เทียบกับชนิดของข้อมูลตัวโน้ตเช่นตัวเขบ็ต 1 ชั้นจะได้เท่ากับ 30)

3C ได้จากไฟล์รหัสข้อมูลไบต์แรก เป็นโน้ตโด

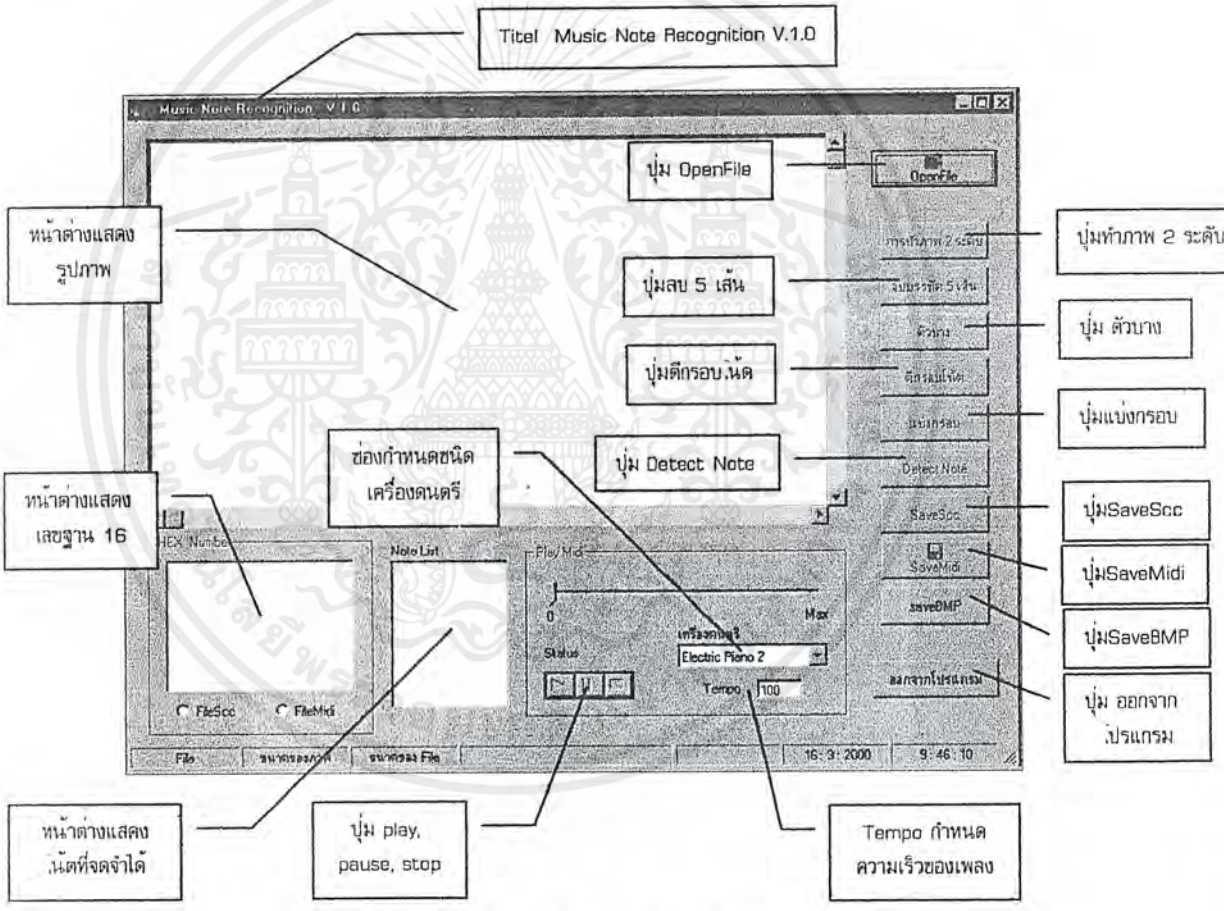
00 เป็นค่าความหนักของการเล่นโน้ต เนื่องจากเป็นNote Off จึงให้ความหนักเท่ากับ 0 (ไม่ต้องออกเสียง)

3. อ่านข้อมูล 2 ไบต์ แล้วเริ่มทำตามข้อ 2 ต่อจนกว่าจะหมดข้อมูลของไฟล์รหัสข้อมูล

บทที่ 4
ผลการทดลอง

ในบทนี้ได้แสดงถึงส่วนติดต่อกับผู้ใช้ บอกถึงหน้าที่การทำงานของปุ่มแต่ละตัว หน้าต่างแสดงผลของรูปภาพ หน้าต่างแสดงเลขฐานสิบหกของไฟล์มิดี้ นอกจากนี้ยังได้บอกถึงการทำงานแต่ละขั้นตอนพร้อมทั้งภาพการทำงานในขั้นตอนนั้นๆด้วย

4.1 ส่วนที่ติดต่อกับผู้ใช้



รูปที่ 4.1 ภาพแสดงส่วนติดต่อกับผู้ใช้

โปรแกรมในส่วนติดต่อกับผู้ใช้ประกอบด้วยส่วนต่างๆดังต่อไปนี้

-ปุ่มการทำงาน

1. ปุ่ม Open File มีหน้าที่ แสดงรายชื่อไฟล์ที่มีส่วนขยายเป็น .BMP ทั้งหมดที่มีอยู่ใน Directory ปัจจุบัน จากนั้นจึงใส่ชื่อไฟล์ที่ต้องการเปิด

2. ปุ่ม การทำภาพ 2 ระดับ มีหน้าที่ ทำภาพที่เราเปิดขึ้นมาให้เป็นภาพ 2 ระดับสี คือ สีดำและสีขาว
3. ปุ่ม การลบบรรทัด 5 เส้น มีหน้าที่ หาเส้นบรรทัด 5 เส้นและเส้นน้อยที่มีในภาพทั้งหมด และทำการลบบรรทัด 5 เส้นและเส้นน้อยทั้งหมดด้วย
4. ปุ่ม ตัวบาง มีหน้าที่ ทำตัวโน้ตทั้งหมดในภาพให้เป็นโครงกระดูก
5. ปุ่ม ตีกรอบโน้ต มีหน้าที่ ตีกรอบตัวโน้ตทุกตัว
6. ปุ่ม แบ่งกรอบ มีหน้าที่ แบ่งกรอบของตัวโน้ตแต่ละตัวให้มี 4 ส่วน
7. ปุ่ม Detect Note มีหน้าที่ หาลักษณะเด่นของโน้ตแต่ละตัว เพื่อบอกว่าโน้ตที่ได้เป็นโน้ตชนิดใด
8. ปุ่ม Save Scc มีหน้าที่ บันทึกตัวโน้ตทั้งหมดที่พบให้เป็นไฟล์รหัสข้อมูล (*.SCC)
9. ปุ่ม Save Midi มีหน้าที่ แปลงไฟล์รหัสข้อมูล (*.SCC) ให้เป็นไฟล์มิดี้ (*.MID)
10. ปุ่ม save BMP มีหน้าที่ บันทึกภาพที่แสดงอยู่ที่หน้าต่างแสดงรูปภาพ ให้เป็นไฟล์ที่มีส่วนขยายเป็น *.BMP เพื่อเก็บภาพการทำงานแต่ละขั้นตอนได้
11. ปุ่ม ออกจากโปรแกรม มีหน้าที่ ปิดการทำงานของโปรแกรม

-<u>หน้าต่างการทำงาน</u>

1. หน้าต่างแสดงรูปภาพ มีหน้าที่ แสดงรูปภาพหลังจากการทำงานในแต่ละขั้นตอน
2. หน้าต่าง HEX Number มีหน้าที่ แสดงข้อมูลไฟล์มิดี้ และไฟล์ SCC เป็นเลขฐานสิบหก
3. หน้าต่าง Note List มีหน้าที่แสดงรายชื่อโน้ตที่พบในขั้นตอนการ Detect note เป็นภาษาไทย

-**<u>Play Midi</u>** เป็นส่วนของการทำงานของไฟล์มิดี้ เพื่อให้เกิดเป็นเสียงดนตรีออกทาง มิดี้การ์ด

- เครื่องดนตรี เป็นส่วนของการกำหนดชนิดของเครื่องดนตรีในไฟล์มิดี้
- Tempo เป็นส่วนของการกำหนดความเร็วในการเล่นของไฟล์มิดี้

-**<u>Status Bar</u>** เป็นส่วนที่บอกรายละเอียดเกี่ยวกับ ไฟล์รูปภาพตัวพิมพ์โน้ตดนตรี ได้แก่ ชื่อไฟล์ ขนาดของไฟล์ รายละเอียดเกี่ยวกับ วัน เดือน ปี เวลา ขณะที่กำลังทำงานอยู่

4.2 <u>ขั้นตอนการทำงาน</u>

ขั้นตอนที่ 1 กดปุ่ม OpenFile เพื่อเรียกไฟล์รูปภาพที่ต้องการ

ขั้นตอนที่ 2 กดปุ่ม การทำภาพเป็น 2 ระดับ ภาพที่เราเปิดขึ้นมาจะเป็น 2 ระดับสี

ขั้นตอนที่ 3 กดปุ่ม การหาระยะเส้น จะหาบรรทัด 5 เส้นและเส้นน้อยทั้งหมดและลบบรรทัด 5 เส้นและเส้นน้อยออกทั้งหมด

ขั้นตอนที่ 4 กดปุ่ม ตัวบาง โน้ตทั้งหมดจะเป็นโครงกระดูก

ขั้นตอนที่ 5 กดปุ่ม ตีกรอบ จะตีกรอบโน้ตที่เป็นโครงกระดูกทุกตัว

ขั้นตอนที่ 6 กดปุ่ม แบ่งกรอบ กรอบของตัวโน้ตทุกตัวออกเป็น 4 ส่วน

ขั้นตอนที่ 7 กดปุ่ม Detect Note เพื่อหาลักษณะเด่นของโน้ตแต่ละตัว และ บอกเป็นรายชื่อโน้ตตรงหน้าต่างNote List

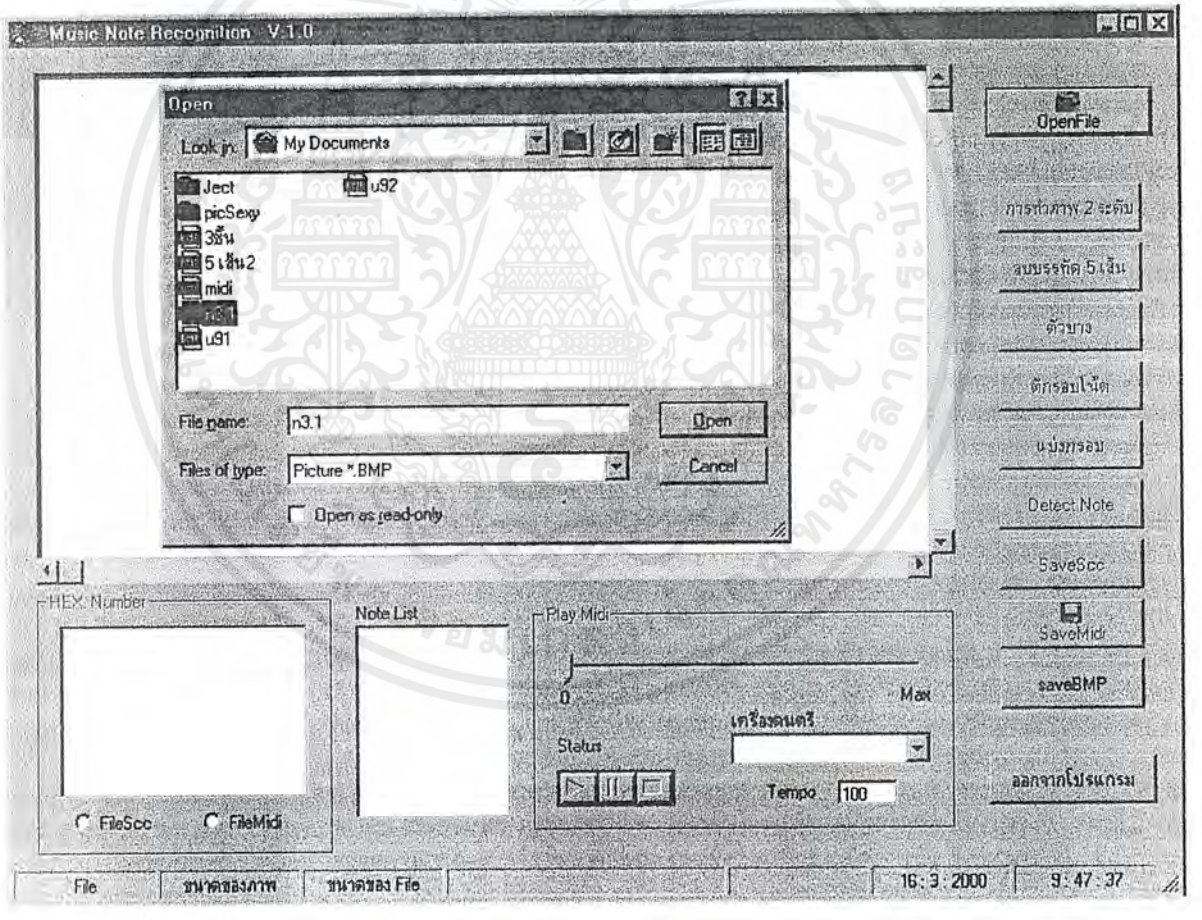
ขั้นตอนที่ 8 กดปุ่ม SaveScc จะบันทึกข้อมูลตัวโน้ตที่หาได้เป็นไฟล์รหัสข้อมูล (*.SCC)

ขั้นตอนที่ 9 กดปุ่ม SaveMidi จะแปลงไฟล์รหัสข้อมูล (*.SCC) เป็นไฟล์มิดี้ (*.MID)

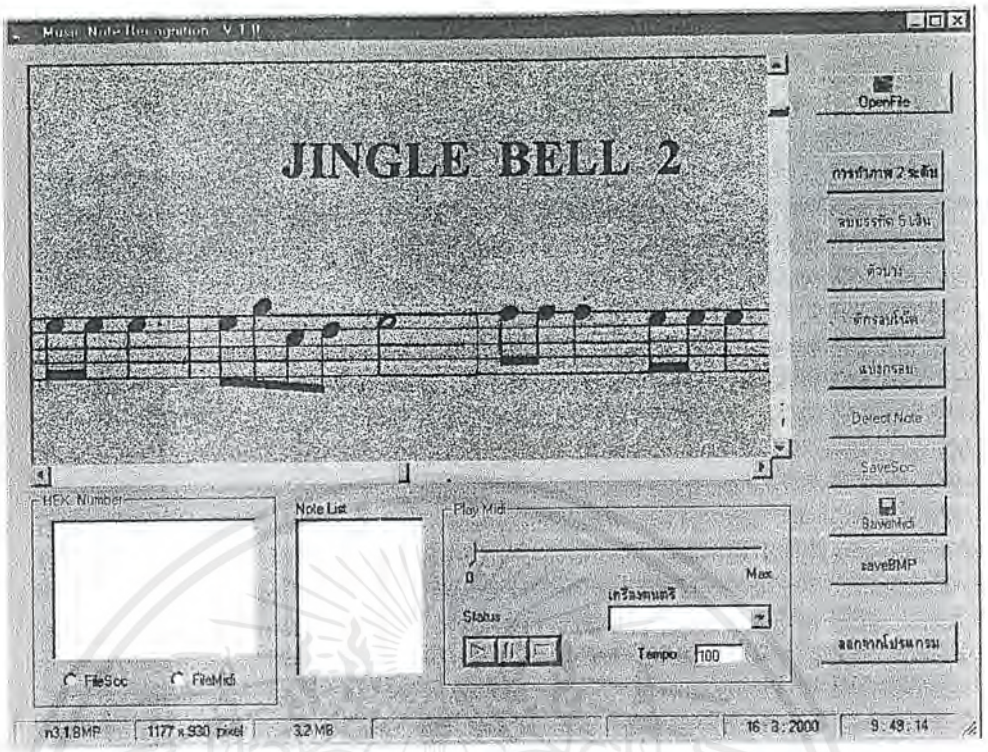
4.3 ผลการทดลองการวิเคราะห์ตัวโน้ต

ผลการทดลองการวิเคราะห์ตัวโน้ตแสดงรูปการทำงานแต่ละขั้นตอนได้ดังต่อไปนี้

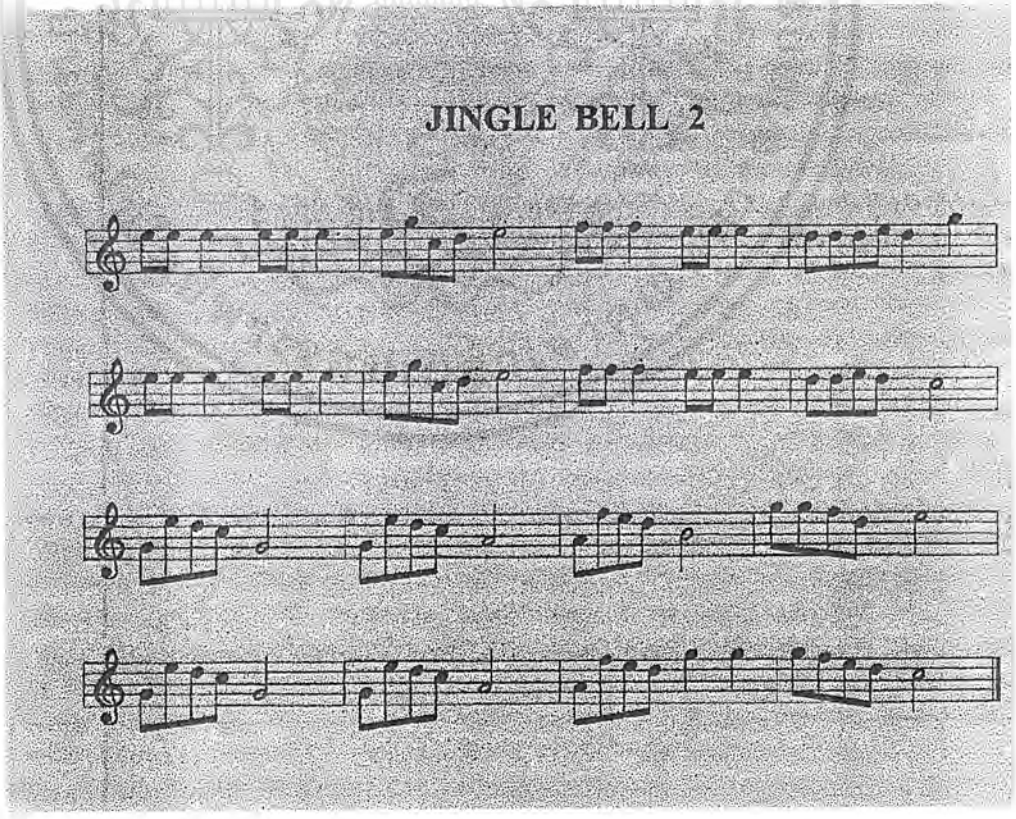
ขั้นตอนที่ 1 กดปุ่ม OpenFile เพื่อเรียกไฟล์รูปภาพที่ต้องการ จะมี Dialog เพื่อเลือกไฟล์ภาพที่ต้องการ



รูปที่ 4.2 การเลือกชื่อไฟล์ภาพตัวโน้ตที่จะทำการวิเคราะห์

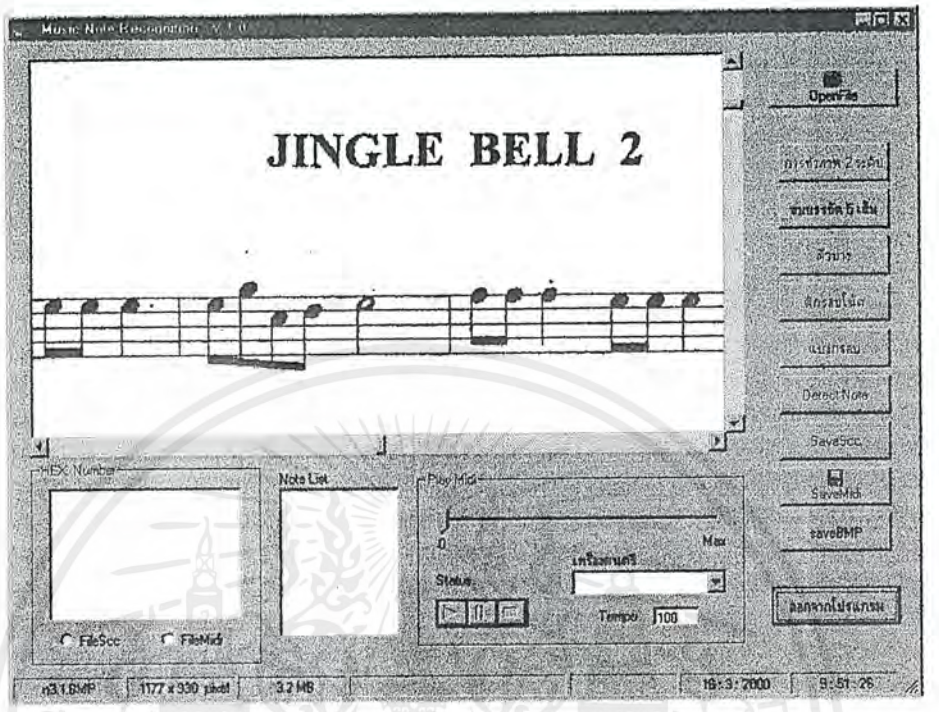


รูปที่ 4.3 ขั้นตอนหลังจากเปิดภาพ

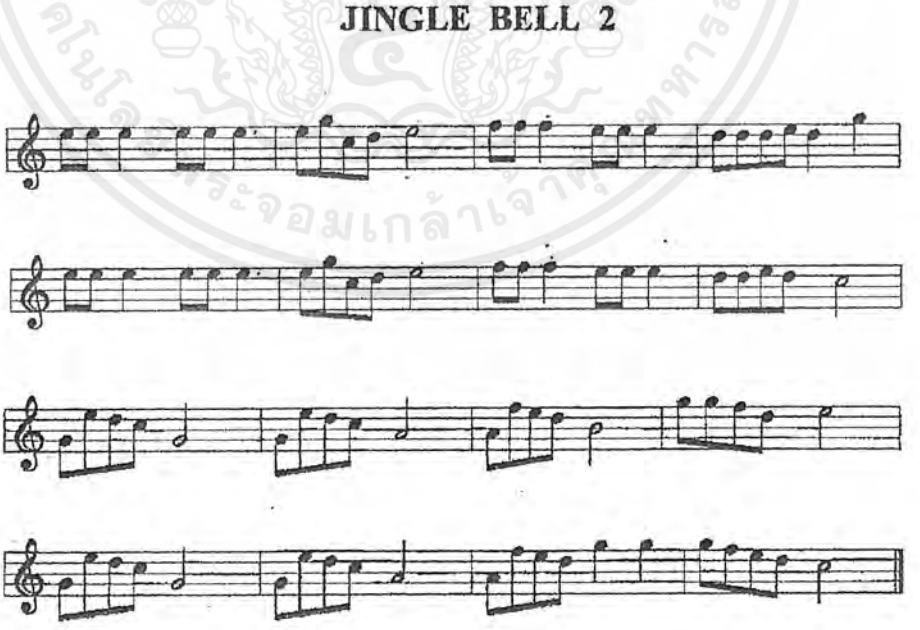


รูปที่ 4.4 ภาพตัวโน้ตที่จะทำการวิเคราะห์

ขั้นตอนที่ 2 กดปุ่ม การทำภาพเป็น 2 ระดับ ในหน้าต่างแสดงภาพจะเป็น 2 ระดับสี ดังรูปที่ 4.6 โดยวิธีการดังหัวข้อ 3.3

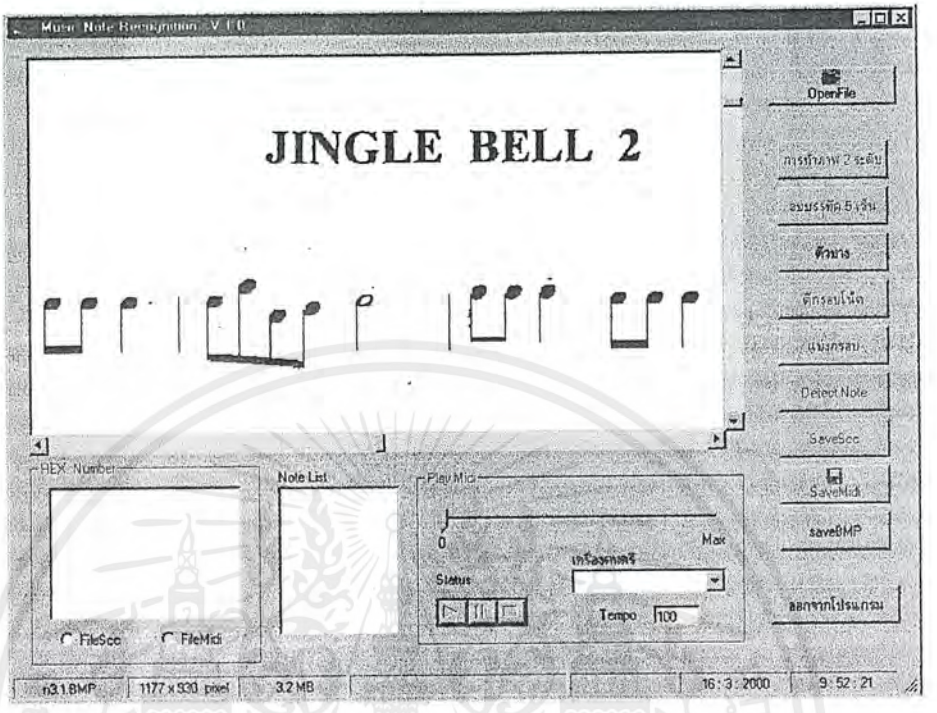


รูปที่ 4.5 ขั้นตอนการทำภาพ 2 ระดับ

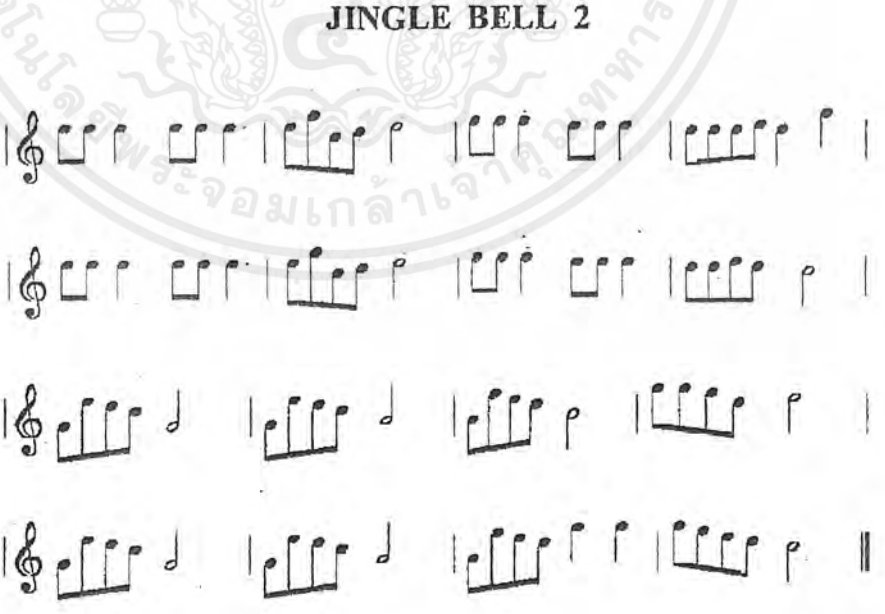


รูปที่ 4.6 ภาพที่ได้หลังขั้นตอนการทำภาพ 2 ระดับ

ขั้นตอนที่ 3 กดปุ่ม การหาระยะเส้น จะหาบรรทัด 5 เส้นและเส้นน้อยทั้งหมดและลบบรรทัด 5 เส้นและเส้นน้อยออกทั้งหมด โดยวิธีการดังหัวข้อที่ 3.4

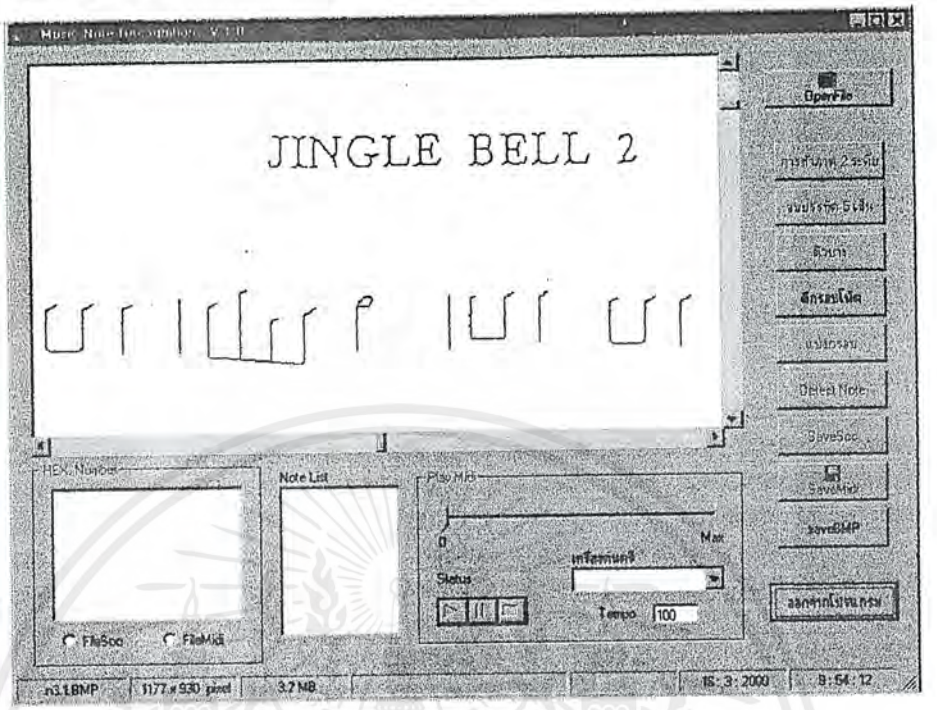


รูปที่ 4.7 ขั้นตอนการลบบรรทัด 5 เส้น

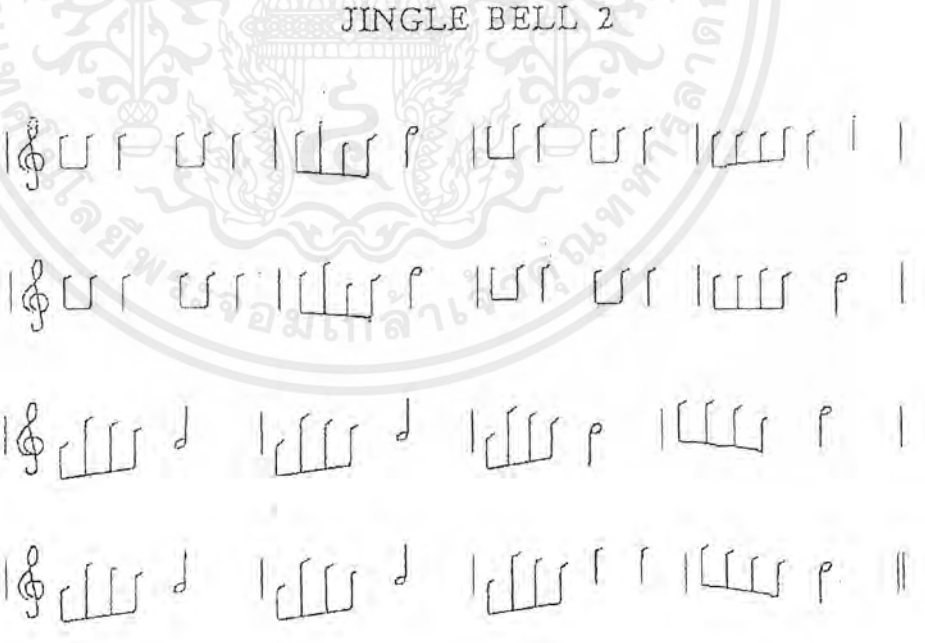


รูปที่ 4.8 ภาพที่ได้ลบบรรทัด 5 เส้นออกแล้ว

ขั้นตอนที่ 4 กดปุ่ม ตัวบาง โน้ตทั้งหมดจะเป็นโครงกระดูก ตามวิธีการดังหัวข้อ 3.6

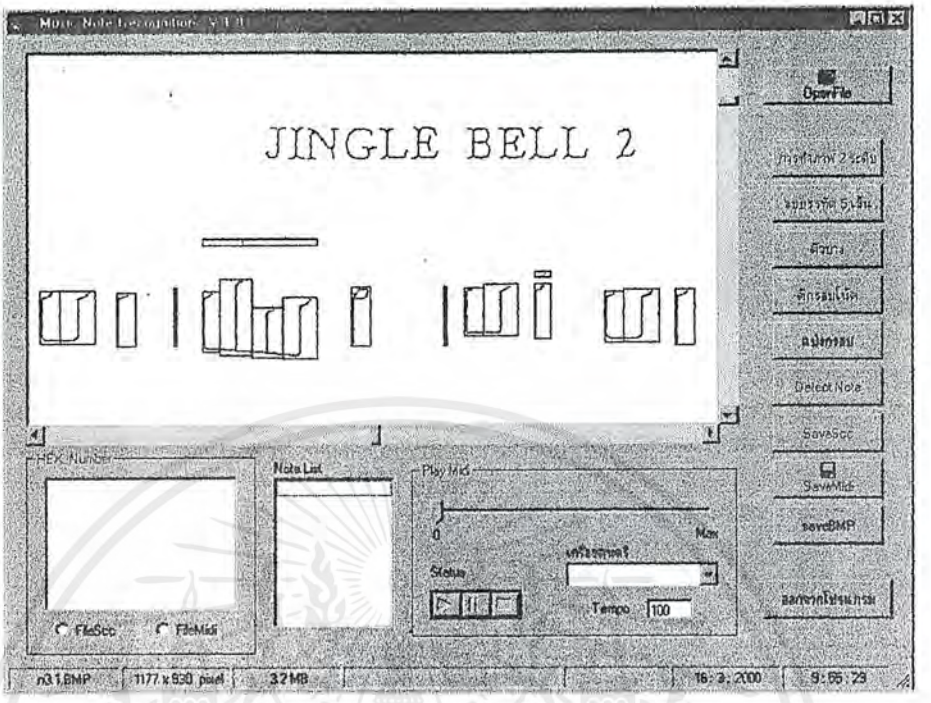


รูปที่ 4.9 ขั้นตอนการทำตัวบาง

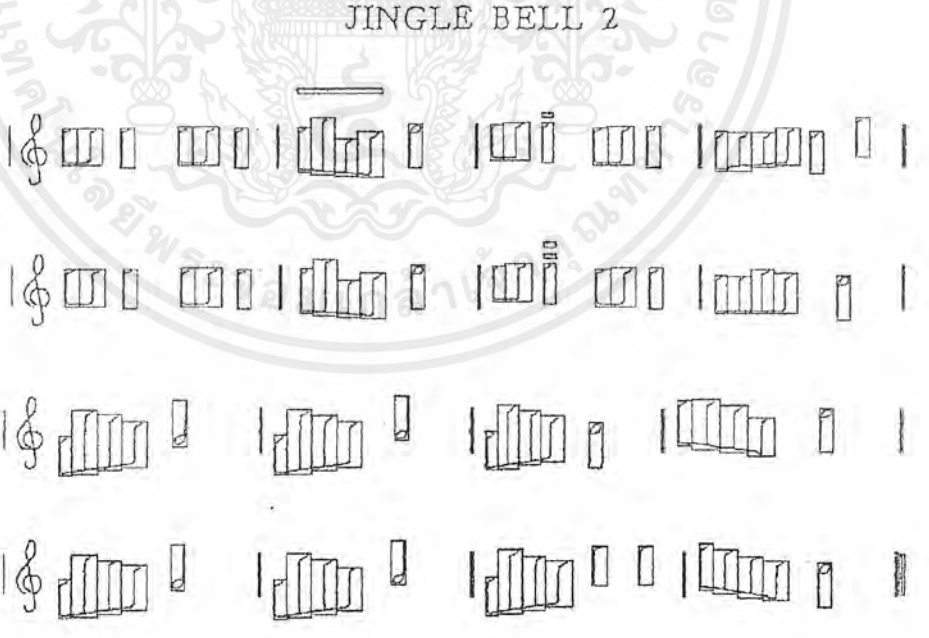


รูปที่ 4.10 ภาพที่ได้จากการทำตัวบาง

ขั้นตอนที่ 5 กดปุ่ม ตีกรอบ จะตีกรอบโน้ตที่เป็นโครงกระดูกทุกตัว โดยวิธีดังหัวข้อ 3.7

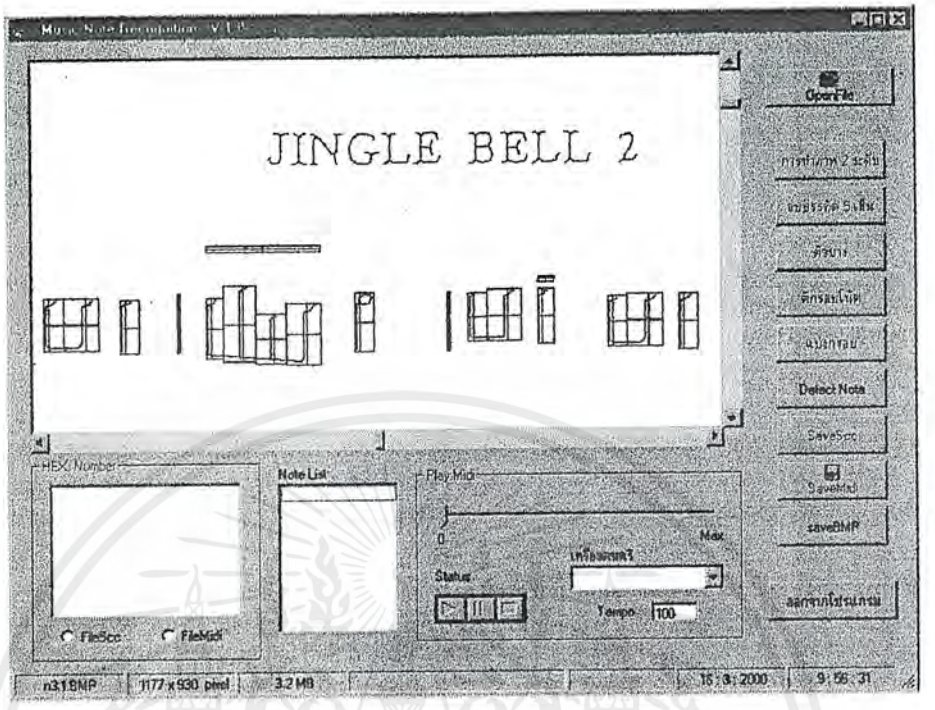


รูปที่ 4.11 ขั้นตอนการตีกรอบตัวโน้ต

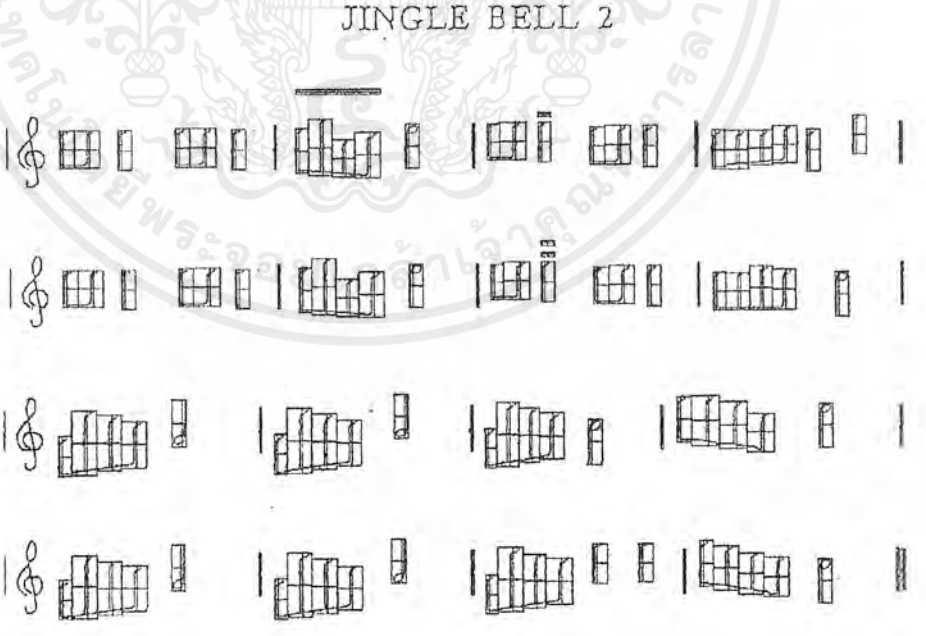


รูปที่ 4.12 ภาพที่ได้จากการตีกรอบตัวโน้ต

ขั้นตอนที่ 6 กดปุ่ม แบ่งกรอบ กรอบของตัวโน้ตทุกตัวออกเป็น 4 ส่วน

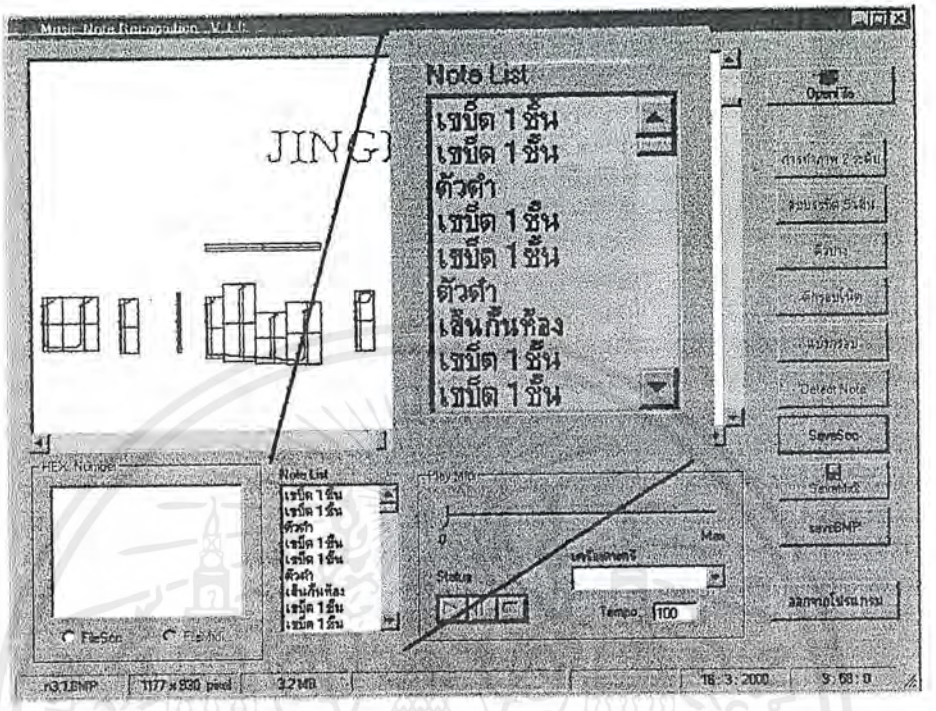


รูปที่ 4.13 ขั้นตอนการแบ่งกรอบตัวโน้ต



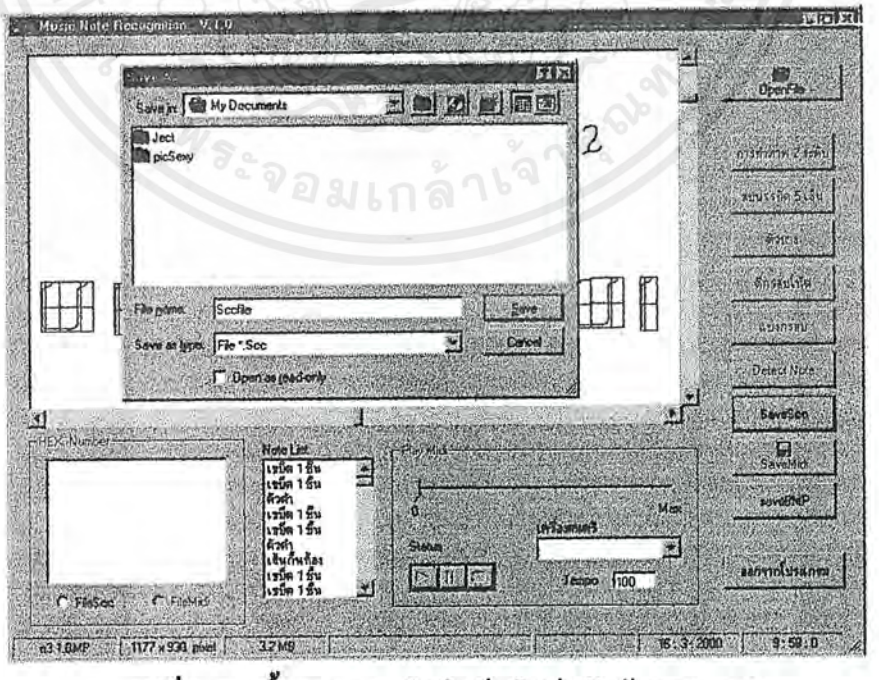
รูปที่ 4.14 ภาพที่ได้จากการแบ่งกรอบตัวโน้ต

ขั้นตอนที่ 7 กดปุ่ม Detect Note เพื่อหาลักษณะเด่นของโน้ตแต่ละตัว และ บอกเป็นรายชื่อโน้ตที่รู้จำได้ที่หน้าต่าง Note List

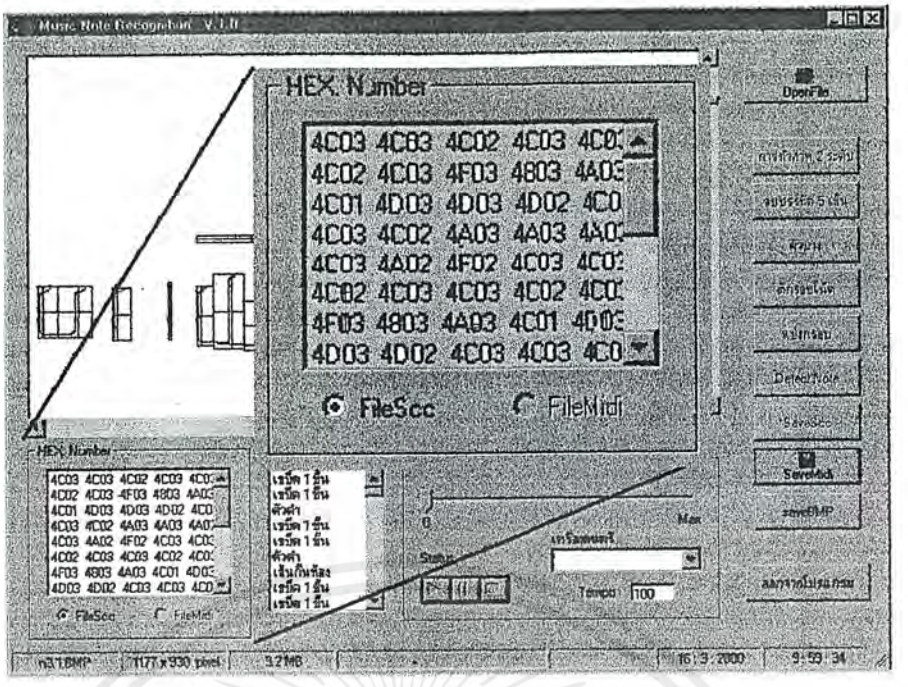


รูปที่ 4.15 ขั้นตอนการรู้จำตัวโน้ต

ขั้นตอนที่ 8 กดปุ่ม SaveScc จะบันทึกข้อมูลตัวโน้ตที่หาได้เป็นไฟล์รหัสข้อมูล (*.SCC)

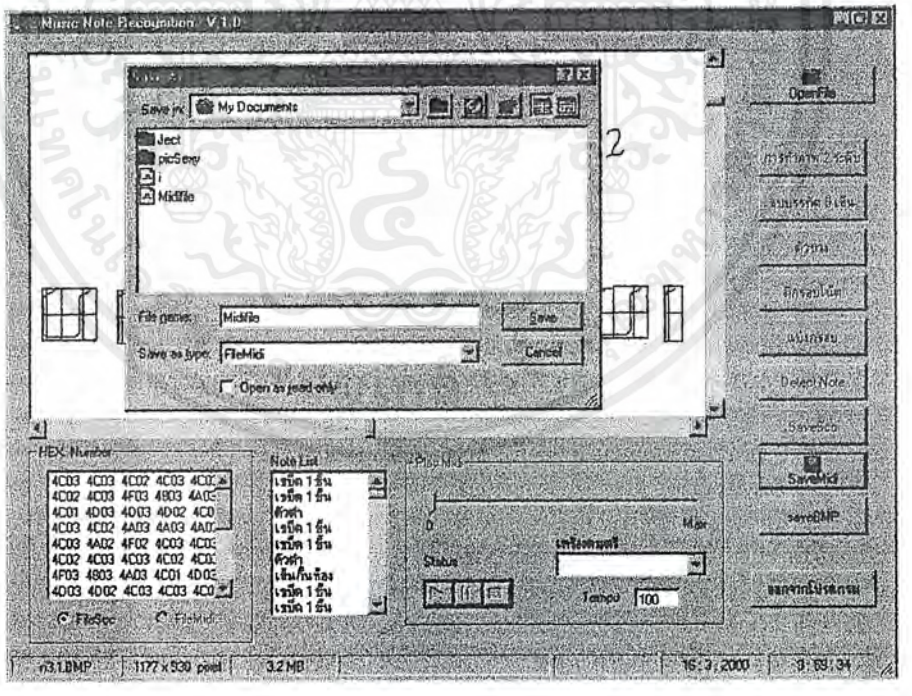


รูปที่ 4.16 ขั้นตอนการบันทึกเป็นไฟล์รหัสข้อมูล(*.scc)

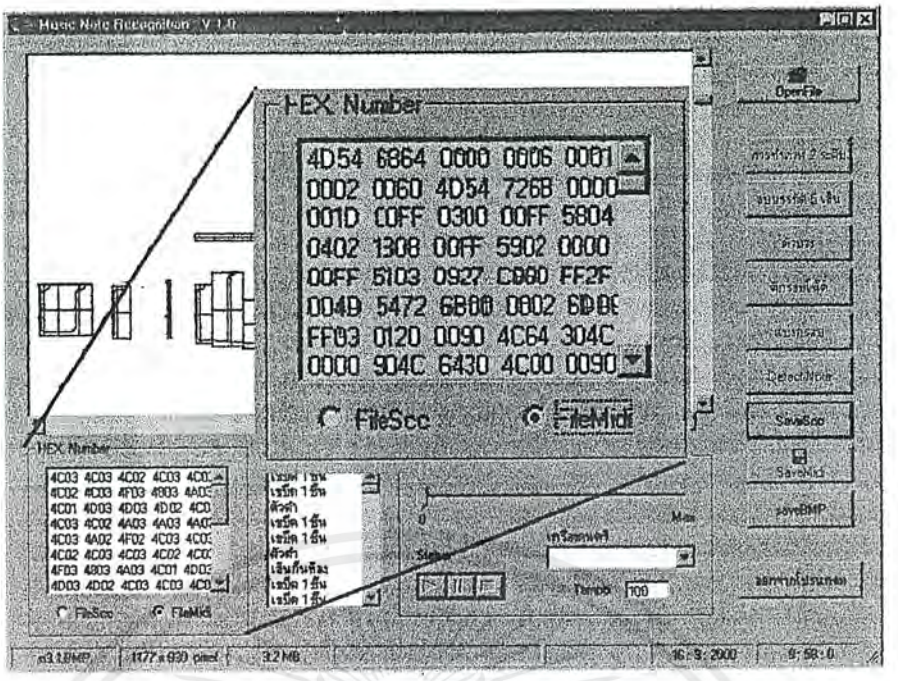


รูปที่ 4.17 ภาพแสดงรหัสข้อมูล(*.scc)

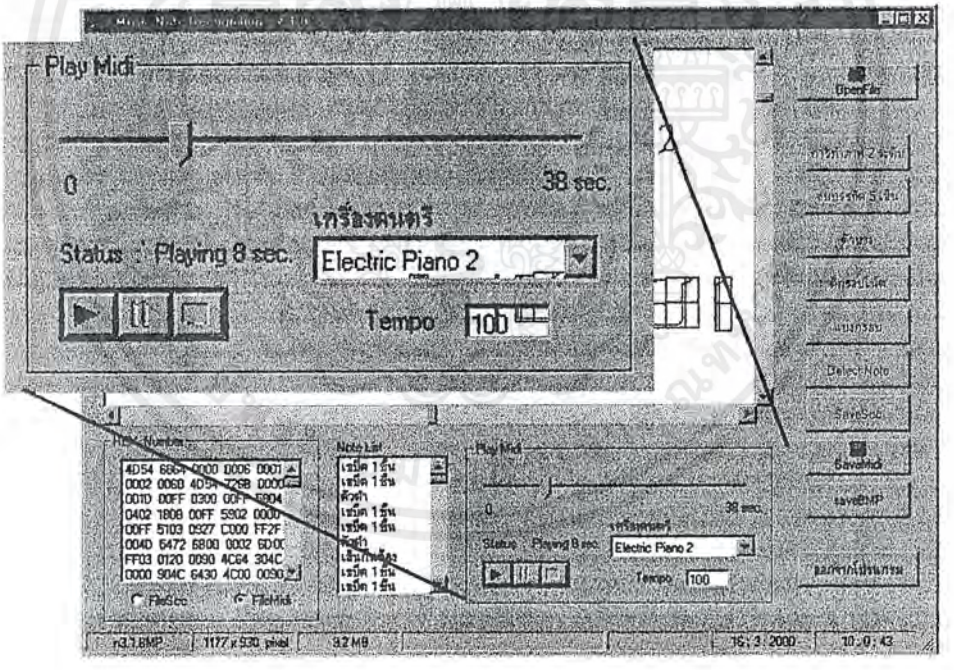
ขั้นตอนที่ 9 กดปุ่ม SaveMidi จะแปลงไฟล์รหัสข้อมูล (*.SCC) เป็นไฟล์มิดี้ (*.MID)



รูปที่ 4.18 ขั้นตอนการบักทึกไฟล์รหัสข้อมูลเป็นไฟล์มิดี้(*.mid)

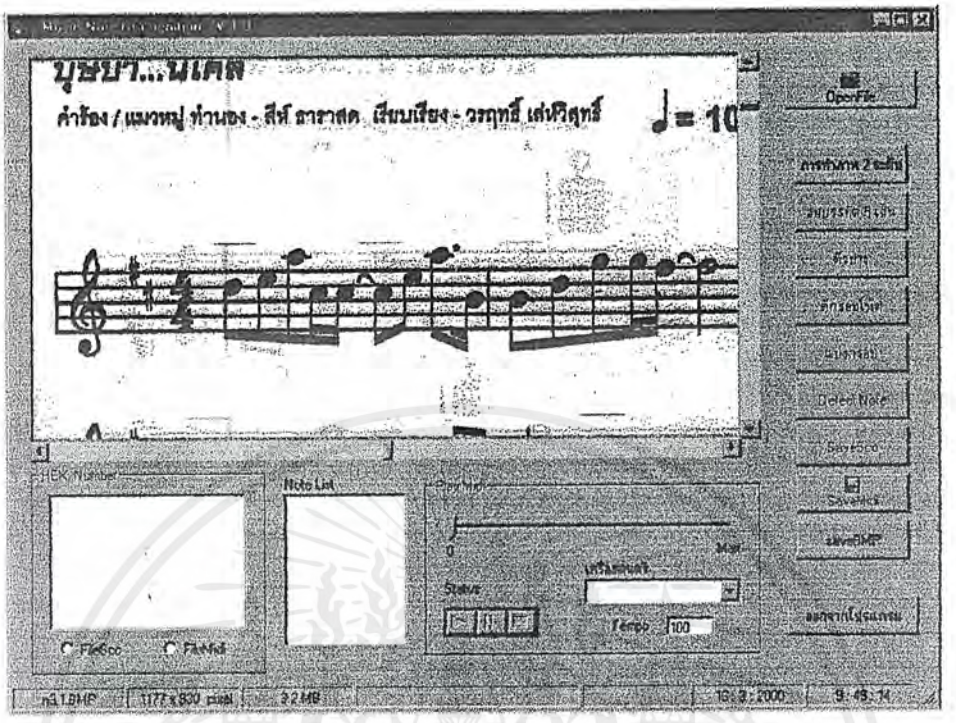


รูปที่ 4.19 ภาพแสดงรหัสข้อมูลมิดี้(*.mid)

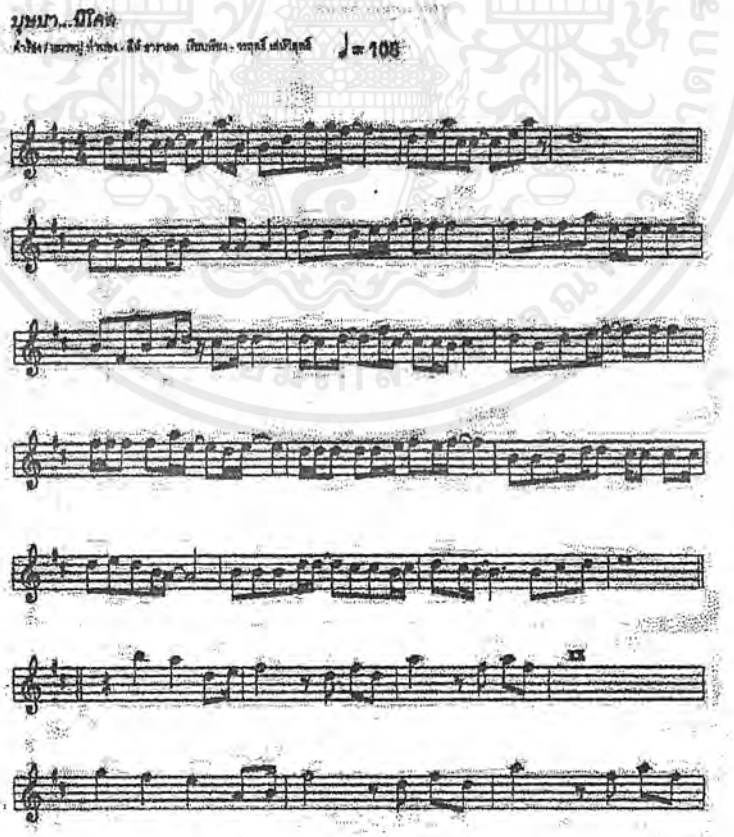


รูปที่ 4.20 ขั้นตอนการเล่นเพลง

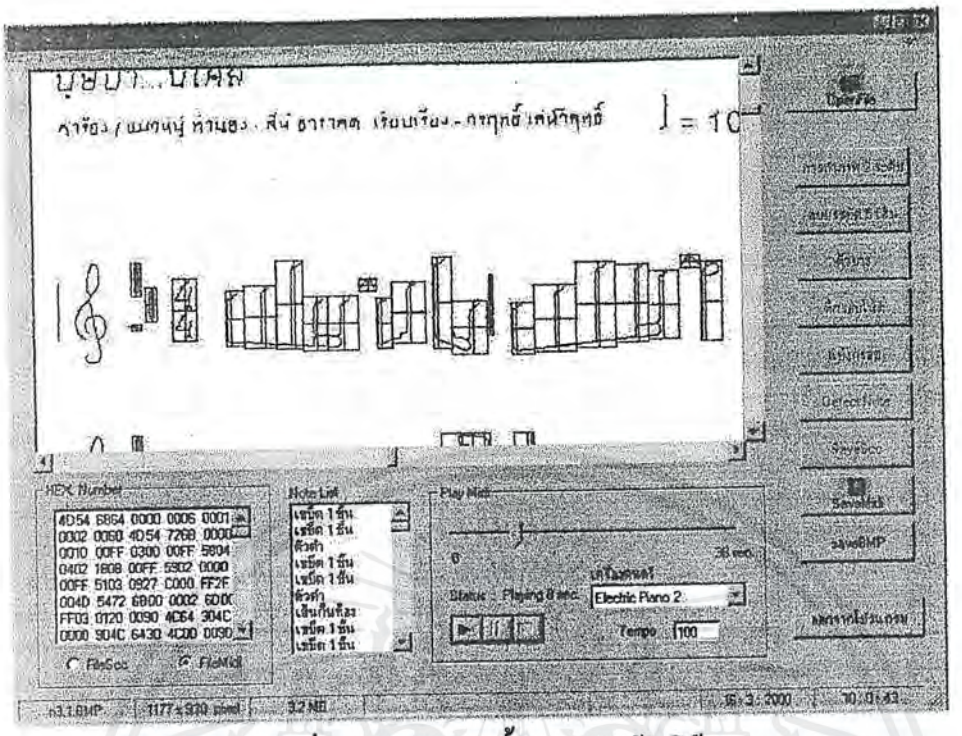
ตัวอย่างเพลง



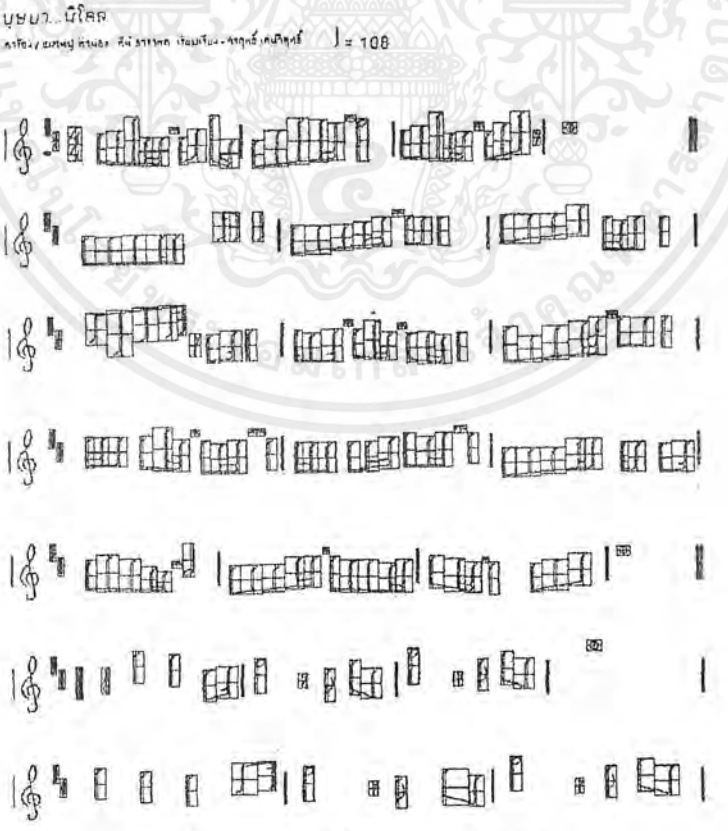
รูปที่ 4.21 ขั้นตอนหลังจากเปิดภาพโน้ตเพลง บุษบา



รูปที่ 4.22 ภาพโน้ตเพลง บุษบา



รูปที่ 4.23 หลังผ่านขั้นตอนการรู้จำโน้ต



รูปที่ 4.24 ภาพเมื่อผ่านการรู้จำโน้ต

```
03 4A 03 4C 03 51 04 49 24 49 43 49 03 4C 03 51
04 47 03 47 03 4A 03 4F 04 4F 24 4E 41 4E 03 4A
03 4C 03 51 04 49 24 49 43 49 03 4C 03 51 0B 00
00 4A 03 47 03 47 03 47 04 47 03 47 04 45 03 45
02 45 03 4A 03 4A 03 4A 04 4C 24 4E 43 4E 03 4E
02 4E 03 4E 03 4E 03 4E 03 51 04 4C 04 4A 03 4C
02 4C 03 47 03 41 03 47 04 4A 04 4A 0C 00 03 4A
04 4A 02 4A 04 4A 03 49 24 4A 43 4A 04 4E 24 49
43 49 04 49 04 47 02 49 03 4A 03 47 03 4A 04 4C
24 4E 43 4E 03 4E 02 4E 04 4F 04 4F 03 4F 03 4F
04 51 24 4C 43 4C 04 4A 24 4C 42 4C 04 4A 04 4A
03 4A 03 4A 04 4A 04 4C 03 4E 04 4C 24 4E 42 4E
03 47 03 47 03 47 04 4A 03 4A 04 49 03 49 03 49
04 49 03 4A 03 4C 03 4A 04 47 24 45 41 45 03 47
03 47 03 47 04 4A 24 4A 43 4A 03 49 03 49 04 47
04 49 03 4A 04 49 24 47 42 47 03 47 03 49 03 4A
00 4C 0A 00 02 53 02 51 03 4A 04 4C 01 4E 0B 00
03 4A 03 4E 03 4A 01 51 0B 00 03 4E 03 51 03 4E
00 53 02 4F 02 4F 02 4C 03 45 04 47 01 4E 0B 00
03 4A 03 4E 03 4A 01 51 0B 00 03 4E 03 51 03 4E
```

รูปที่ 4.25 แสดงรหัสข้อมูล(*.SCC) ของเพลงบุษบา

```
4D 54 68 64 00 00 00 06 00 01 00 02 00 60 4D 54
72 6B 00 00 00 1D 00 FF 03 00 00 FF 58 04 04 02
18 08 00 FF 59 02 00 00 00 FF 51 03 09 27 C0 00
FF 2F 00 4D 54 72 6B 00 00 03 A7 00 C0 0B 00 90
4A 64 30 4A 00 00 4C 64 30 4C 00 00 51 64 30 51
00 00 49 64 18 49 00 00 49 64 04 49 64 28 49 00
1C 49 00 00 4C 64 30 4C 00 00 51 64 30 51 00 00
47 64 18 47 00 00 47 64 30 47 00 00 4A 64 30 4A
00 00 4F 64 30 4F 00 00 4F 64 18 4F 00 00 4E 64
04 4E 64 28 4E 00 81 2C 4E 00 00 4A 64 30 4A 00
00 4C 64 30 4C 00 00 51 64 30 51 00 00 49 64 18
49 00 00 49 64 04 49 64 28 49 00 1C 49 00 00 4C
64 30 4C 00 00 51 64 30 51 00 30 4A 64 83 00 4A
00 00 47 64 30 47 00 00 47 64 30 47 00 00 47 64
30 47 00 00 47 64 18 47 00 00 47 64 30 47 00 00
45 64 18 45 00 00 45 64 30 45 00 00 45 64 60 45
00 00 4A 64 30 4A 00 00 4A 64 30 4A 00 00 4A 64
30 4A 00 00 4C 64 18 4C 00 00 4E 64 04 4E 64 28
4E 00 1C 4E 00 00 4E 64 30 4E 00 00 4E 64 60 4E
00 00 4E 64 30 4E 00 00 4E 64 30 4E 00 00 4E 64
30 4E 00 00 51 64 30 51 00 00 4C 64 18 4C . . .
```

รูปที่ 4.26 แสดงรหัสมิดี้(*.MID) ของเพลงบุษบา

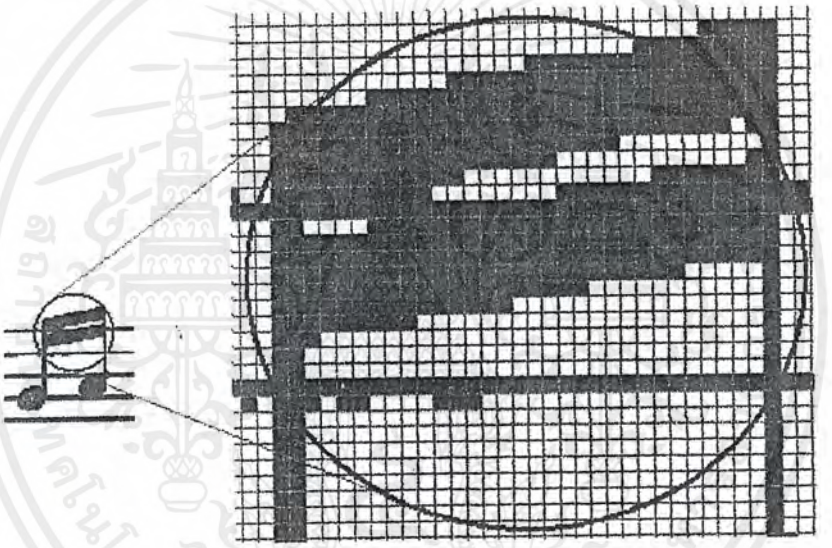
บทที่ 5
สรุปและวิจารณ์ผลการทดลอง

ในบทนี้จะกล่าวถึงปัญหา ที่เกิดจากงานวิจัย สาเหตุของปัญหานั้นๆ และท้ายที่สุดจะกล่าวถึงแนวทางการพัฒนาต่อไปสำหรับงานวิจัยนี้

5.1 ปัญหาที่พบ

จากการวิเคราะห์รูปภาพตัวพิมพ์โน้ตที่ได้จากการสแกน ปัญหาที่เกิดขึ้นกับตัวโน้ตมีลักษณะดังนี้

1. โน้ตเขบ็ต 2 ชั้นที่บรรทัด 5 เส้นลากผ่านช่องเขบ็ตนั้น



รูปที่ 5.1 โน้ตเขบ็ต 2 ชั้นที่บรรทัด 5 เส้นลากผ่านช่องเขบ็ต

จากรูปที่ 5.1 ปัญหาที่เกิดขึ้นคือหลังจากผ่านขั้นตอนการลบเส้นแล้ว รูปที่ได้จะมีลักษณะเป็นดังรูปที่ 5.2 ซึ่งจะมีส่วนเกินเกิดขึ้น

รูปที่ 5.2 โน้ตผ่านการลบเส้น

เมื่อนำรูปที่ 5.2 ไปทำตัวบาง และทำการแยกโน้ตแต่ละตัวออกจากกันตามขั้นตอนคุณสมบัติโน้ตชุด

รูปที่ 5.3 โน้ตผ่านการทำตัวบาง

รูปที่ 5.4 โน้ตที่มีคุณสมบัติเป็นโน้ตชุดผ่านการแยก

จะเห็นได้ว่าหลังจากการแยกโน้ตชุดออกจากกัน เพื่อที่จะไปหาลักษณะเด่นของโน้ตแต่ละตัว โน้ตตัวแรกที่ถูกแยกออกมาจากโน้ตชุดจะมีลักษณะไม่ตรงตามฐานข้อมูลที่มี ทำให้ไม่สามารถวิเคราห์ได้ถูกต้อง

ก. โน้ตที่ได้ ข. โน้ตที่มีในฐานข้อมูล

รูปที่ 5.5

2. โน้ตตัวหยุดเขบ็ตที่อยู่บนบรรทัด 5 เส้นในลักษณะดังรูปที่ 5.6

รูปที่ 5.6 โน้ตตัวหยุดเขบ็ตที่มีปัญหา

ภาคผนวก ก.

ทฤษฎีโน้ตดนตรี

<u>ทฤษฎีโน้ตดนตรี</u>

สัญลักษณ์โน้ตดนตรีประกอบด้วยกลุ่มตัวโน้ต 2 กลุ่ม คือ

1. กลุ่มของตัวโน้ตที่บอกจังหวะเสียง และโน้ตตัวหยุด จะมีระดับความยาวของเสียงต่างๆกัน ดังนี้

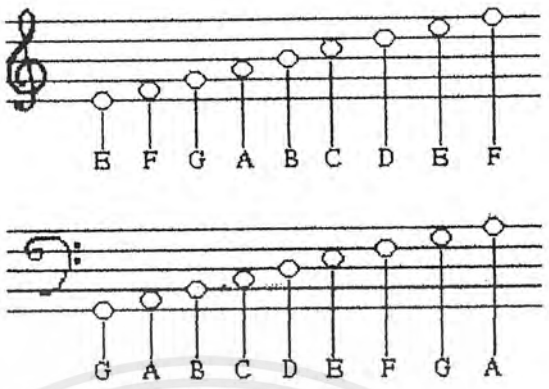
	ตัวโน้ต	ตัวหยุด	ค่าจังหวะ
ตัวกลม			4
ตัวขาว			2
ตัวดำ			1
ตัวเขบ็ต 1 ชั้น			1/2
ตัวเขบ็ต 2 ชั้น			1/4
ตัวเขบ็ต 3 ชั้น			1/8

2. กลุ่มตัวโน้ตอื่นๆ

	ตัวโน้ต	หน้าที่
ตัวแฟลต	b	ลดระดับของเสียงครึ่งเสียง
ตัวชาร์ป	#	เพิ่มระดับของเสียงครึ่งเสียง
กุญแจซอล		บอกให้รู้ว่าบนเส้นที่ 2 มีชื่อว่า G ซอล
กุญแจฟา		บอกให้รู้ว่าบนเส้นที่ 4 มีชื่อว่า F ฟา
ตัวโยงเสียง		รวมเสียงของโน้ตให้ยาวนานขึ้น
เส้นกั้นห้อง		เพื่อแบ่งจังหวะตามที่กำหนด
จุด		จะอยู่หลังตัวโน้ตเพื่อเพิ่มความยาวเสียงของตัวโน้ตไปอีกครึ่งเสียง

ระดับเสียงและชื่อทางดนตรี

การจัดเรียงระดับเสียงจากต่ำไปหาสูง 7 เสียงดังนี้ คือ C D E F G A B (โด เร มี ฟา ซอล ลา ที) โดยใช้สัญลักษณ์ตัวโน้ตต่างๆ เขียนลงบรรทัด 5 เส้น ระดับของเสียงบนบรรทัด 5 เส้น เป็นดังนี้



รูปที่ ก.1 แสดงระดับเสียงของโน้ต

The MIDI 1.0 Specification*

Introduction

MIDI is the acronym for Musical Instrument Digital Interface.

MIDI enables synthesizers, sequencers, home computers, rhythm machines, etc. to be interconnected through a standard interface.

Each MIDI-equipped instrument usually contains a receiver and a transmitter. Some instruments may contain only a receiver or transmitter. The receiver receives messages in MIDI format and executes MIDI commands. It consists of an opto-isolator, Universal Asynchronous Receiver/Transmitter (UART), and other hardware needed to perform the intended functions. The transmitter originates messages in MIDI format, and transmits them by way of a UART and line driver.

The MIDI standard hardware and data format are defined in this specification.

Conventions

Status and Data bytes given in Tables I through VI are given in binary.

Numbers followed by an "H" are in hexadecimal.

All other numbers are in decimal.

*MIDI Manufacturers Association, 5316 West 57th Street, Los Angeles, CA. 90056

Hardware

The interface operates at 31.25 (± 1%) Kbaud, asynchronous, with a start bit, 8 data bits (D0 to D7), and a stop bit. This makes a total of 10 bits for a period of 320 microseconds per serial byte.

Circuit: 5 mA current loop type. Logical 0 is current ON. One output shall drive one and only one input. The receiver shall be opto-isolated and require less than 5 mA to turn on. Sharp PC-900 and HP 6N138 opto-isolators have been found acceptable. Other high-speed opto-isolators may be satisfactory. Rise and fall times should be less than 2 microseconds.

Connectors: DIN 5 pin (180 degree) female panel mount receptacle. An example is the SWITCHCRAFT 57 GB5F. The connectors shall be labelled "MIDI IN" and "MIDI OUT." Note that pins 1 and 3 are not used, and should be left unconnected in the receiver and transmitter.

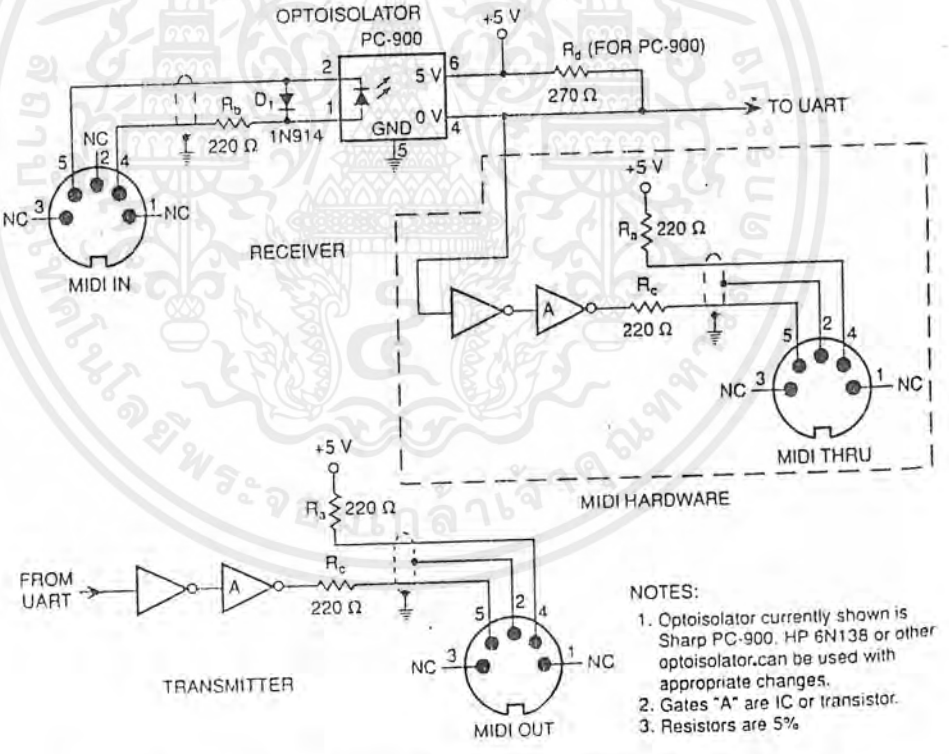


Fig. A-1. Standard hardware configuration for MIDI in, out, and thru ports.

Cables shall have a maximum length of fifty feet (15 meters), and shall be terminated on each end by a corresponding 5-pin DIN male plug, such as the SWITCHCRAFT 05GM5M. The cable shall be shielded twisted pair, with the shield connected to pin 2 at both ends.

A "MIDI THRU" output may be provided if needed, which provides a direct copy of data coming in MIDI IN. For very long chain lengths (more than three instruments), higher-speed opto-isolators must be used to avoid additive rise/fall time errors which affect pulse width duty cycle.

Data Format

All MIDI communication is achieved through multi-byte "messages" consisting of one Status byte followed by one or two Data bytes, except Real-Time and Exclusive messages (see below).

Message Types

Messages are divided into two main categories: Channel and System.

Channel

Channel messages contain a four-bit number in the Status byte which address the message specifically to one of sixteen channels. These messages are thereby intended for any units in a system whose channel number matches the channel number encoded into the Status byte.

There are two types of Channel messages: Voice and Mode.

Voice–To control the instrument's voices, Voice messages are sent over the Voice Channels.

Mode–To define the instrument's response to Voice messages, Mode messages are sent over the instrument's Basic Channel.

System

System messages are not encoded with channel numbers.

There are three types of System messages: Common, Real-Time, and Exclusive.

Common–Common messages are intended for all units in a system.

Real-Time–Real-Time messages are intended for all units in a system.

They contain Status bytes only—no Data bytes. Real-Time messages may be sent at any time—even between bytes of a message which has a different status. In such cases the Real-Time message is either ignored or acted upon, after which the receiving process resumes under the previous status.

Exclusive–Exclusive messages can contain any number of Data bytes, and are terminated by an End of Exclusive (EOX) or any other Status byte. These messages include a Manufacturer's Identification (ID) code. If the receiver does not recognize the ID code, it should ignore the ensuing data.

So that other users can fully access MIDI instruments, manufacturers should publish the format of data following their ID code. Only the manufacturer can update the format following their ID.

Data Types

Status Bytes

Status bytes are eight-bit binary numbers in which the Most Significant Bit (MSB) is set (binary 1). Status bytes serve to identify the message type; that is, the purpose of the Data bytes which follow the Status byte.

Except for Real-Time messages, new Status bytes will always command the receiver to adopt their status, even if the new Status is received before the last message was completed.

Running Status—For Voice and Mode messages only, when a Status byte is received and processed, the receiver will remain in that status until a different Status byte is received. Therefore, if the same Status byte would be repeated, it may (optionally) be omitted so that only the correct number of Data bytes need be sent. Under Running Status, then, a complete message need only consist of specified Data bytes sent in the specified order.

The Running Status feature is especially useful for communicating long strings of Note On/Off messages, where "Note On with Velocity of 0" is used for Note Off. (A separate Note Off Status byte is also available.)

Running Status will be stopped when any other Status byte intervenes, except that Real-Time messages will only interrupt the Running Status temporarily.

Unimplemented Status—Any status bytes received for functions which the receiver has not implemented should be ignored, and subsequent data bytes ignored.

Undefined Status—Undefined Status bytes must not be used. Care should be taken to prevent illegal messages from being sent during power-up or power-down. If undefined Status bytes are received, they should be ignored, as should subsequent Data bytes.

Data Bytes

Following the Status byte, there are (except for Real-Time messages) one or two Data bytes which carry the content of the message. Data bytes are eight-bit binary numbers in which the MSB is reset (binary 0). The number and range of Data bytes which must follow each Status byte are specified in the tables which follow. For each Status byte the correct number of Data bytes must always be sent. Inside the receiver, action on the message should wait until all Data bytes required under the current status are received. Receivers should ignore Data bytes which have not been properly preceded by a valid Status byte (with the exception of "Running Status," above).

Channel Modes

Synthesizers contain sound generation elements called voices. Voice assignment is the algorithmic process of routing Note On/Off data from the keyboard to the voices so that the musical notes are correctly played with accurate timing.

When MIDI is implemented, the relationship between the sixteen available MIDI channels and the synthesizer's voice assignment must be defined. Several Mode messages are available for this purpose (see Table III). They are Omni (On/Off), Poly, and Mono. Poly and Mono are mutually exclusive, i.e., Poly Select disables Mono, and vice versa. Omni, when on, enables the receiver to receive Voice messages in all Voice Channels without discrimination. When Omni is off, the receiver will accept Voice messages from only the selected Voice Channel(s). Mono, when on, restricts the assignment of Voices to just one voice per Voice Channel (Monophonic.) When Mono is off (=Poly On), any number of voices may be allocated by the Receiver's normal voice assignment algorithm (Polyphonic).

For a receiver assigned to Basic Channel "N," the four possible modes arising from the two Mode messages are:

Mode	Omni		
1	On	Poly	Voice messages are received from all Voice Channels and assigned to voices polyphonically.
2	On	Mono	Voice messages are received from all Voice Channels, and control only one voice, monophonically.
3	Off	Poly	Voice messages are received in Voice Channel N only, and are assigned to voices polyphonically.
4	Off	Mono	Voice messages are received in Voice Channels N thru N+M−1, and assigned monophonically to voices 1 thru M, respectively. The number of voices M is specified by the third byte of the Mono Mode Message.

Four modes are applied to transmitters (also assigned to Basic Channel N). Transmitters with no channel selection capability will normally transmit on Basic Channel 1 (N=0).

Mode	Omni		
1	On	Poly	All voice messages are transmitted in Channel N.
2	On	Mono	Voice messages for one voice are sent in Channel N.
3	Off	Poly	Voice messages for all voices are sent in Channel N.
4	Off	Mono	Voice messages for voices 1 thru M are transmitted in Voice Channels N thru N+M−1, respectively. (Single voice per channel.)

A MIDI receiver or transmitter can operate under one and only one mode at a time. Usually the receiver and transmitter will be in the same mode. If a mode cannot be honored by the receiver, it may ignore the message (and any subsequent data bytes), or it may switch to an alternate mode (usually Mode 1, Omni On/Poly).

Mode messages will be recognized by a receiver only when sent in the Basic Channel to which the receiver has been assigned, regardless of the current mode. Voice messages may be received in the Basic Channel and in other channels (which are all called Voice Channels), which are related specifically to the Basic Channel by the rules above, depending on which mode has been selected.

A MIDI receiver may be assigned to one or more Basic Channels by default or by user control. For example, an eight-voice synthesizer might be assigned to Basic Channel 1 on power-up. The user could then switch the instrument to be configured as two four-voice synthesizers, each assigned to its own Basic Channel. Separate Mode messages would then be sent to each four-voice synthesizer, just as if they were physically separate instruments.

Power-Up Default Conditions

On power-up all instruments should default to Mode # 1. Except for Note On/Off Status, all Voice messages should be disabled. Spurious or undefined transmissions must be suppressed.

Table I. Summary of Status Bytes

Status D7— D0	# of Data Bytes	Description
		Channel Voice Messages
1000nnnn	2	Note Off event
1001nnnn	2	Note On event (velocity=0: Note Off)
1010nnnn	2	Polyphonic key pressure/after touch
1011nnnn	2	Control change
1100nnnn	1	Program change
1101nnnn	1	Channel pressure/after touch
1110nnnn	2	Pitch bend change
		Channel Mode Messages
1011nnnn	2	Selects Channel Mode
		System Messages
11110000	*****	System Exclusive
11110sss	0 to 2	System Common
11111ttt	0	System Real Time

Notes:

nnnn: N-1, where N = Channel #,
i.e., 0000 is Channel 1.
0001 is Channel 2.
.
.
.
1111 is Channel 16.

*****: 0iiiiiii, data, ..., EOX

iiiiiii: Identification

sss: 1 to 7

ttt: 0 to 7

Table II. Channel Voice Messages

Status	Data Bytes	Description
1000nnnn	0kkkkkkk 0vvvvvvv	Note Off (see notes 1-4) vvvvvvv: note off velocity
1001nnnn	0kkkkkkk 0vvvvvvv	Note On (see notes 1-4) vvvvvvv - 0: velocity vvvvvvv = 0: note off
1010nnnn	0kkkkkkk 0vvvvvvv	Polyphonic Key Pressure (After-Touch) vvvvvvv: pressure value
1011nnnn	0ccccccc 0vvvvvvv	Control Change ccccccc: control # (0-121) (see notes 5-8) vvvvvvv: control value ccccccc = 122 thru 127: Reserved. (See Table III)
1100nnnn	0ppppppp	Program Change ppppppp: program number (0-127)
1101nnnn	0vvvvvvv	Channel Pressure (After-Touch) vvvvvvv: pressure value
1110nnnn	0vvvvvvv 0vvvvvvv	Pitch Bend Change LSB (see note 10) Pitch Bend Change MSB

Notes:

1. nnnn: Voice Channel # (1-16, coded as defined in Table I notes)

2. kkkkkkk: note # (0 - 127)
   kkkkkkk = 60: Middle C of keyboard

0	12	24	36	48	60	72	84	96	108	120	127
		ac	c	c	c	c	c	c	c		
		|——— piano range ———|									

3. vvvvvvv: key velocity
   A logarithmic scale would be advisable.

0	1			64				127
off	ppp	pp	p	mp	mf	f	fff	fff

   vvvvvvv = 64: in case of no velocity sensors
   vvvvvvv = 0: Note Off, with velocity = 64

4. Any Note On message sent should be balanced by sending a Note Off message for that note in that channel at some later time.

5. cccccc: control number

cccccc	Description
0	Continuous Controller 0 MSB
1	Continuous Controller 1 MSB (MODULATION BENDER)
2	Continuous Controller 2 MSB
3	Continuous Controller 3 MSB
4-31	Continuous Controllers 4-31 MSB
32	Continuous Controller 0 LSB
33	Continuous Controller 1 LSB (MODULATION BENDER)
34	Continuous Controller 2 LSB
35	Continuous Controller 3 LSB
36-63	Continuous Controllers 4-31 LSB
64-95	Switches (On/Off)
96-121	Undefined
122-127	Reserved for Channel Mode messages (see Table III).

6. All controllers are specifically defined by agreement of the MIDI Manu facturers Association (MMA) and the Japan MIDI Standards Committee (JMSC). Manufacturers can request through the MMA or JMSC that logical controllers be assigned to physical ones as necessary. The controller allocation table must be provided in the user's operation manual.

7. Continuous controllers are divided into Most Significant and Least Signili cant Bytes. If only seven bits of resolution are needed for any particular controllers, only the MSB is sent. It is not necessary to send the LSB. If more resolution is needed, then both are sent, first the MSB, then the LSB. If only the LSB has changed in value, the LSB may be sent without re-sending the MSB.

8. vvvvvv: control value (MSB)
(for controllers)

0 ——————————————— 127
min max

(for switches)

0 - - - - - - - - - - - 127
off on

Numbers 1 through 126, inclusive, are ignored.

9. Any messages (e.g., Note On), which are sent successively under the same status, can be sent without a Status byte until a different Status byte is needed.

10. Sensitivity of the pitch bender is selected in the receiver. Center position value (no pitch change) is 2000H, which would be transmitted EnH-00H-40H.

Table III. Channel Mode Messages

Status	Data Bytes	Description
1011nnnn	0ccccccc 0vvvvvvv	Mode Messages ccccccc = 122: Local Control vvvvvvv = 0, Local Control Off vvvvvvv = 127, Local Control On ccccccc = 123: All Notes Off vvvvvvv = 0 ccccccc = 124: Omni Mode Off (All Notes Off) vvvvvvv = 0 ccccccc = 125: Omni Mode On (All Notes Off) vvvvvvv = 0 ccccccc = 126: Mono Mode On (Poly Mode Off) (All Notes Off) vvvvvvv = M, where M is the number of channels. vvvvvvv = 0, the number of channels equals the number of voices in the receiver. ccccccc = 127: Poly Mode On (Mono Mode Off) vvvvvvv = 0 (All Notes Off)

Notes:

1. nnnn: Basic Channel # (1-16, coded as defined in Table I)

2. Messages 123 thru 127 function as All Notes Off messages. They will turn off all voices controlled by the assigned Basic Channel. Except for message 123, All Notes Off, they should not be sent periodically, but only for a specific purpose. In no case should they be used in lieu of Note Off commands to turn off notes which have been previously turned on. Therefore any All Notes Off command (123-127) may be ignored by receiver with no possibility of notes staying on, since any Note On command must have a corresponding specific Note Off command.

3. Control Change #122. Local Control, is optionally used to interrupt the internal control path between the keyboard, for example, and the sound-generating circuitry. If 0 (Local Off message) is received, the path is disconnected: the keyboard data goes only to MIDI and the sound-generating circuitry is controlled only by incoming MIDI data. If a 7FH (Local On message) is received, normal operation is restored.

4. The third byte of "Mono" specifies the number of channels in which Monophonic Voice messages are to be sent. This number, "M", is a number between 1 and 16. The channel(s) being used, then, will be the current

Basic Channel (=N) thru N+M−1 up to a maximum of 16. If M=0, this is a special case directing the receiver to assign all its voices, one per channel, from the Basic Channel N through 16.

Table IV. System Common Messages

Status	Data Bytes	Description
11110001		Undefined
11110010		Song Position Pointer
	0lllllll	lllllll: (Least significant)
	0hhhhhhh	hhhhhhh: (Most significant)
11110011	0sssssss	Song Select
		sssssss: Song #
11110100		Undefined
11110101		Undefined
11110110	none	Tune Request
11110111	none	EOX: "End of System Exclusive" flag

1. Song Position Pointer: Is an internal register which holds the number of MIDI beats (1 beat = 6 MIDI clocks) since the start of the song. Normally it is set to 0 when the START switch is pressed, which starts sequence playback. It then increments with every sixth MIDI clock receipt, until STOP is pressed. If CONTINUE is pressed, it continues to increment. It can be arbitrarily preset (to a resolution of 1 beat) by the SONG POSITION POINTER message.
2. Song Select: Specifies which song or sequence is to be played upon receipt of a Start (Real-Time) message.
3. Tune Request: Used with analog synthesizers to request them to tune their oscillators.
4. EOX: Used as a flag to indicate the end of a System Exclusive transmission (see Table VI).

Table V. System Real Time Messages

Status	Data Bytes	Description
11111000		Timing Clock
11111001		Undefined
11111010		Start
11111011		Continue
11111100		Stop
11111101		Undefined
11111110		Active Sensing
11111111		System Reset

Notes:

1. The System Real Time messages are for synchronizing all of the system in real time.

2. The System Real Time messages can be sent at any time. Any messages which consist of two or more bytes may be split to insert Real Time messages.

3. Timing clock (F8H)
   The system is synchronized with this clock, which is sent at a rate of 24 clocks/quarter note.

4. Start (from the beginning of song) (FAH)
   This byte is immediately sent when the PLAY switch on the master (e.g., sequencer or rhythm unit) is pressed.

5. Continue (FBH)
   This is sent when the CONTINUE switch is hit. A sequence will continue at the time of the next clock.

6. Stop (FCH)
   This byte is immediately sent when the STOP switch is hit. It will stop the sequence.

7. Active Sensing (FEH)
   Use of this message is optional, for either receivers or transmitters. This is a "dummy" Status byte that is sent every 300 ms (max), whenever there is no other activity on MIDI. The receiver will operate normally if it never receives FEH. Otherwise, if FEH is ever received, the receiver will expect to receive FEH or a transmission of any type every 300 ms (max). If a period of 300 ms passes with no activity, the receiver will turn off the voices and return to normal operation.

8. System Reset (FFH)
   This message initializes all of the system to the condition of just having turned on power. The system Reset message should be used sparingly, preferably under manual command only. In particular, it should not be sent automatically on power up.

Table VI. System Exclusive Messages

Status	Data Bytes	Description
11110000		Bulk dump etc.
	0iiiiiii	iiiiiii: identification
	.	
	(0*******)	
	.	Any number of bytes may be sent here, for any purpose, as long as they all have a zero in the most significant bit.
	(0*******)	
	.	
	11110111	EOX: "End of System Exclusive"

Notes:

1. iiiiiii: identification ID (0-127)
2. All bytes between the System Exclusive Status byte and EOX or the next Status byte must have zeroes in the MSB.
3. The ID number can be obtained from the MMA or JMSC.
4. In no case should other Status or Data bytes (except Real-Time) be interleaved with System Exclusive, regardless of whether or not the ID code

ภาคผนวก ค.

The MIDI Implementation Chart

Function ...		Transmitted	Recognized	Remarks
Basic Channel	Default	1 – 16	1 – 16	memorized
	Changed	1 – 16	1 – 16	
Mode	Default	Mode 3, 4	Mode 3	memorized
	Messages	OMNI OFF, MONO POLY	×	
	Altered	**********		
Note Number	True Voice	0 – 127	0 – 127	
		**********	12 – 108	
Velocity	Note ON	○ v = 1 – 127	○ v = 1 – 127	
	Note OFF	× 9n v = 0	×	
After Touch	Key's	×	×	
	Ch's	×	×	
Pitch Bender		○	○ 0 – 24 semitone	
Control Change	1	○	○	Modulation
	2 – 5	×	×	
	6	**	**	Data Entry MSB
	7	○	○	Volume
	8 – 15	×	×	
	16	×	○	General Purpose Control-1
	17 – 37	×	×	
	38	**	×	Data Entry LSB
	39 – 63	×	×	
	64	×	○	Hold 1
	65 – 80	×	×	
	81	×	○	General Purpose Control-1
	82 – 99	×	×	
	100 – 101	** (0)	** (0)	RPC LSB, MSB
	102 – 120	×	×	
	121	○	○	Reset All Controllers
Prog Change	True #	○ 0 – 127	○ 0 – 127.	
		**********	0 – 127	
System Exclusive		○	○	
System Common	Song Pos	×	×	
	Song Sel	×	×	
	Tune	×	×	
System Real Time	Clock	×	×	
	Commands	×	×	
Aux Message	Local ON/OFF	×	×	
	All Notes OFF	×	○	
	Active Sense	○	○	
	Reset	×	×	
Notes		* Control Change messages from 0 to 95 which are recognized through Control channel are transmitted throgh all the channels which are used in Branches. However, General Purpose Control -1 and General Purpose Control – 6 are converted into the same functions as the FC-100 EV-5 assign and the FC-100 Switch assign in the System Setup, and are transmitted. ** RPC = Registered Parameter Control Number RPC # 0 : Bender Range The value of parameter is to be determined by entering data.		

Mode 1 : OMNI ON, POLY Mode 2 : OMNI ON, MONO ○ : Yes
Mode 3 : OMNI OFF, POLY Mode 4 : OMNI OFF, MONO × : No

Fig. B-1. Example of a MIDI Implementation Chart.

Despite efforts at standardization, slight inconsistencies within the chart's specifications allow for variations in the symbols, abbreviations, spelling, etc. that can be used by different manufacturers. The following guidelines provide a basic understanding of these differences.

- In general, the symbol O is used to indicate that a MIDI function *is* implemented, while an X is used to show that the function is *not* implemented. However, some charts may use an X to equal a *yes* and an O to equal a *no*. This will usually be indicated within a key at the lower right-hand corner of the chart.
- OX or "*" is used to indicate a selectable function. Further information on the range or type of selectability will be placed within the remarks column.
- MIDI modes are listed as follows:

  Mode 1 (omni on, poly)
  Mode 2 (omni on, mono)
  Mode 3 (omni off, poly)
  Mode 4 (omni off, mono)

  These will often be listed at the bottom of the chart. Occasionally abbreviations of these modes (i.e., omni on/off, omni on or poly) may be used by a manufacturer.

Detailed Explanation of the Chart

The following is a detailed explanation of the various functions and their related categories that are found within the chart.

Header

The *header* provides the user with the model number, brief description, date, and version number of the device.

Basic Channel

Basic channel indicates which MIDI channels are used by the device to transmit and receive data. The subheadings for this function are ***default*** and ***changed***.

- *Default*: This indicates which MIDI channel is in use when the device is first turned on.
- *Changed*: This indicates which of the MIDI channels can be addressed after the device is first turned on.

Mode

Mode indicates which of the MIDI modes may be used by the device. The subheadings for this function are default, messages, and altered.

- *Default*: This indicates which of the four MIDI modes is active when the device is first turned on.
- *Messages*: This describes which of the four MIDI modes can be transmitted or recognized by the device.
- *Altered*: This refers to mode messages which cannot be recognized by the device. It may be followed by a description of the mode that the device automatically enters into upon receiving a request message for an unavailable mode.

Note Number

The transmitted *note number* indicates the range of MIDI note numbers that are transmitted by a device. The maximum possible range spans from 0–127, while 21–108 corresponds to the 88 keys of an extended keyboard controller. Should the note number be greater than the actual number of keys on a keyboard device, a key transposition feature is indicated.

The recognized note number indicates the range of MIDI note numbers that can be recognized by a device. MIDI notes that are out of this range shall be ignored by this device. A second note number range, known as *true voice*, indicates the number of notes the device can actually play. Recognized notes that are out of the actual voice range are transposed up or down in octaves until they fall within this range.

Velocity

This category indicates whether the device is capable of transmitting or receiving attack- and release-velocity messages. The subheadings for this function are *note on* and *note off*.

- *Note on*: This indicates if the device is capable of transmitting and responding to variable-velocity (attack) messages. Not all dynamically controllable devices respond to the full velocity range (1–127). Some devices, such as drum machines, respond to a finite number of velocity steps.

- *Note off*: Indicates whether the device is capable of transmitting and responding to variable release velocity messages. Many devices use a message (note-on velocity = 0) to indicate a note-off condition. This is often indicated in the chart by *9NH v=0* or *89n 00*, which is the hexadecimal equivalent for this message.

After Touch

After touch indicates how pressure data is transmitted or received. The subheadings for this function are *key's* and *ch's*.

- *Key's*: This indicates if the device will transmit or receive independent polyphonic-pressure messages for each key.
- *Ch's*: Indicates whether the device is capable of transmitting or receiving channel-pressure changes (a common after-touch mode, providing one pressure value for an entire MIDI channel).

Pitch Bender

Pitch bender indicates if the device is capable of transmitting or receiving pitch-bend information. If so, the remarks column will often give information as to the pitch bend range and resolution.

Control Change

Control change indicates whether the device is capable of transmitting or receiving continuous-controller messages. The chart will often list which of these messages are supported in addition to providing a detailed breakdown of their parameters within the remarks column.

Program Change

This category indicates if the device is capable of transmitting or receiving *program-change messages*. *True #* indicates the message numbers that are actually supported by the device's program-change buttons.

System Exclusive

This indicates if the device is capable of transmitting and receiving *system exclusive data*. The remarks column will often give general information as to which type of SysEx data is supported. However, more detailed data will generally be provided within the device's manual.

System Common

This indicates whether the device is capable of transmitting or receiving the different types of *system common messages*, such as SPP, MIDI time code, song select, and tune-request messages.

System Real Time

This category indicates whether the device can transmit or receive *system real-time messages*. The subheadings for this function are *clock* and *commands*.

- *Clock*: This refers to the device's ability to receive or transmit MIDI clock messages. A device that can transmit MIDI clock may be used to provide master timing information within a MIDI system, while a device capable of receiving clock data may only be slaved to other MIDI devices.
- *Commands*: This indicates whether the device is capable of transmitting or responding to start, stop, and continue messages.

Aux Messages

This indicates if a device is capable of transmitting or receiving local control-on/off, all notes-off, active-sensing, and system-reset messages.

Notes

This area is used by the manufacturer to comment on any function or implementation particular to the specific MIDI device.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้คงไม่สามารถดำเนินลุล่วงไปได้ หากปราศจากการสนับสนุนจากบุคคลที่มีพระคุณยิ่งเหล่านี้ ที่คอยให้คำปรึกษาและช่วยเหลือแก้ไขปัญหาต่างๆด้วยดีตลอดมา

ขอขอบคุณ รศ. เกษตร์ ศิริสันติสัมฤทธิ์ อาจารย์ที่ให้คำปรึกษาเป็นอย่างดี

ขอขอบคุณ พี่นพนันทน์ (ขออภัยที่ไม่ทราบนามสกุล) ให้คำแนะนำด้านการเขียนโปรแกรม

ขอขอบคุณ พี่ปริวัฒน์ (ขออภัยที่ไม่ทราบนามสกุล) ให้คำแนะนำด้านดนตรี

ขอขอบคุณ เพื่อนเทพ, เพื่อนนุ้ย, เพื่อนอูม ที่คอยให้กำลังใจและด้านอื่นๆ

ขอขอบคุณ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ที่ให้ความรู้ต่างๆ

สุดท้าย ขอขอบคุณภาควิชาเทคโนโลยีการวัดคุมฯ ที่เอื้อเฟื้อเครื่องมือและอุปกรณ์ต่างๆในการทำโครงงานนี้ให้สำเร็จลุล่วงไปด้วยดี

บรรณานุกรม

1. ซัคเซส มีเดีย , " Microsoft Visual Basic Version 5.0 " , บริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน) , 2541
2. สุทธิศักดิ์ พงศ์ธนาพานิช ,"Visual Basic Professional 5.0" ,บริษัท ซีเอ็ดยูเคชั่น จำกัด(มหาชน) , 2541
3. ชัชวาล ศุภเกษม , " Microsoft Visual Basic Version 6.0 ภาคปฏิบัติ ", บริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน) , 2542
4. นิรัตน์ บุญชูมณี, ภิญโญ เหล่ามงคลชัย, " การจดจำรูปแบบตัวพิมพ์โน้ต " , ปริญญานิพนธ์ ปี 2537
5. พิชัย ปรัชญานุสรณ์ , " ทฤษฎีดนตรี " , พับลิคบิสเนสพริ้นท์ , 2528