ปริญญานิพนธ์ปีการศึกษา 2537

ภาควิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบมิดี้

ผู้จัดทำ

1. นายภาณุ ตรงเมธีรัตน์
2. นายรัฐ ไข่แก้ว

อาจารย์ที่ปรึกษา

(ร.ศ. ดร. มนัส สังวรศิลป์)

ระบบมิดี้

นายภาณุ ตรงเมธีรัตน์
นายรัฐ ไข่แก้ว

รศ.ดร.มนัส สังวรศิลป์อาจารย์ที่ปรึกษา
ปีการศึกษา 2537

บทคัดย่อ

ด้วย รายงานฉบับนี้ เรียบเรียงขึ้นจากผลงานที่ได้จากการศึกษาและทดลองสร้างวงจรตัวกลางมิดี้(MIDI Interface) ซึ่งเป็นเครื่องมือที่ใช้เชื่อมต่อระหว่างระบบคอมพิวเตอร์และเครื่องดนตรี ที่สามารถเข้าใจมาตรฐานมิดี้ (Potocal) ให้สามารถติดต่อสื่อสารถึงกันได้ มาตรฐานมิดี้กำหนดให้การรับ - ส่งข้อมูลอนุกรมอะซิงโครนัส (Serial Asyncronous) ด้วยความเร็ว 31250 บิทต่อ วินาที ผ่านทางมิดี้พอร์ท (Port) ดังนั้นจึงใช้ไอซีเบอร์ 8251 เป็นตัวกลางในการรับส่งข้อมูล และมีแหล่งกำเนิดสัญญาณนาฬิกาขนาด 2 เมกกะเฮิร์ต

การใช้งานตัวกลางมิดี้จำเป็นต้องมีโปรแกรมควบคุมตามแต่วัตถุประสงค์ ซึ่งตามโครงงานนี้ต้องการนำคอมพิวเตอร์มาทำหน้าที่เป็นซีเควนเซอร์ (Sequencer) โดยทำหน้าที่ควบคุมซินธีไซเซอร์ให้ออกเสียงตามที่บันทึก รวมถึงการบันทึกและแก้ไขข้อมูลในขณะบันทึกและหลังบันทึก

MIDI System

Panu Thongmateerathana

Ratha Kaikeaw

Associate Professor Dr. Manus Sungvorasilapa

1994

Abstract

This thesis is about a studying and working on an experiment of MIDI interfacing medium. MIDI interfacing medium is a tool for interfacing between computers and musical instruments. MIDI is a kind of potocal for communicating between component which can understand this potocal. Standard of transmitting and receiving data is in a serial asynchronous mode with a speed of 3125 bits per second by a MIDI port. In this project we used IC8251 to be UART for transmitting and receiving data with a clock 's frequency of 2 MHz.

When using MIDI , we must set an objetive control program for each task. In this project,computer is used as sequencer. Computer is used to control a synthesizer, either playback or record. And computer can used to record and correct data while recording or after recording too.

สารบัญ

บทที่ 1
บทนำ(Introduction)

ปัจจุบันคอมพิวเตอร์ได้เข้ามาเป็นส่วนหนึ่งของชีวิตประจำวันของเรามากขึ้นเรื่อย ๆ ไม่ว่าในด้านการศึกษา ด้านธุรกิจ ด้านอุตสาหกรรม หรือ แม้กระทั่งในเรื่องของศิลปะทางด้านดนตรี

คอมพิวเตอร์เริ่มเข้ามาเกี่ยวพันธ์กับศาสตร์ทางด้านดนตรี ในระยะที่มีการสร้างเครื่องดนตรีพวกดรัมแมทชีน (DRUMMACHINE) ซึ่งทำหน้าที่เป็นผู้ตีกลอง โดยในยุคแรกยังคงเป็นเพียงวงจรนับ กับ วงจรตรรกธรรมดา ต่อมาในปี 1980 บริษัทผู้ผลิตซินธิไซเซอร์ (SYNTHESIZER) ได้มีการนำ ไมโครโปรเซทเซอร์ (MICROPROCESSOR) มาเป็นตัวอ่านลิ่มคีย์ แล้วนำข้อมูลส่งให้ ออสซิลเลเตอร์ (OSCILLATOR) สร้างเป็นความถี่ตามโน๊ตดนตรีที่กด นับจากนั้นเป็นต้นมาได้มีการพัฒนาประสิทธิภาพของเครื่องดนตรีอย่างต่อเนื่อง บนพื้นฐานของระบบคอมพิวเตอร์ จนกระทั่งในปี 1983 ได้มีการนำข้อมูลการอ่านลิ่มคีย์ ส่งออกจากตัวเครื่องดนตรีด้วย โดยบริษัทผู้ผลิตเครื่องดนตรีได้รวมตัวกันจัดตั้งมาตรฐานในการส่งข้อมูลนี้ ในชื่อว่ามิดี้ (MIDI MUSICAL INSTRUMENT DIGITAL INTERFACING)

นับจากวันนั้นจนถึงวันนี้ ระบบมิดี้ ได้สร้างประโยชน์ต่องานทางด้านดนตรีอย่างสูง ระบบ มิดี้ทำให้มีการสื่อสารกันระหว่างเครื่องดนตรีกับเครื่องดนตรี และเครื่องดนตรีกับ ซีเควนเซอร์ (SCQUENCER) ซึ่งเป็นอุปกรณ์บันทึกข้อมูลมิดี้ไปจนถึงการสื่อสารระหว่างเครื่องดนตรีกับไมโครคอมพิวเตอร์

สำหรับโครงงานนี้เป็นการนำเอา ไมโครคอมพิวเตอร์ตระกูล IBM-PC มาใช้ติดต่อสื่อสารข้อมูลกับเครื่องดนตรี

ประโยชน์ที่ได้จากโครงงานนี้

-จะทำให้เราสามารถควบคุมเครื่องดนตรี ให้เล่นเพลงให้กับเราตามที่เราต้องการได้ ทำให้นักดนตรีสามารถแต่งเพลง และสร้างงานทางด้านดนตรีได้ รวดเร็วและมีประสิทธิภาพ กว่าการใช้ความสามารถที่มีอยู่ในตัวนักดนตรีเอง และนอกจากนั้นยังมีประโยชน์ต่อผู้ฝึกหัดเล่นดนตรี

ใหม่รวมไปถึงผู้ที่สนใจ เพราะเราสามารถนำข้อมูลเพลงจากนักดนตรีที่มีความสามารถมาศึกษาได้

-ทำให้การแต่งเพลงสามารถทำได้โดยง่ายกว่าเก่ามาก ซึ่งผู้แต่งก็ไม่จำเป็นที่ต้องมีความสามารถทางด้านดนตรีสูงและไม่จำเป็นที่จะต้องเล่นดนตรีเป็นหลาย ๆ ประเภทในการแต่งเพลง หนึ่ง ๆ

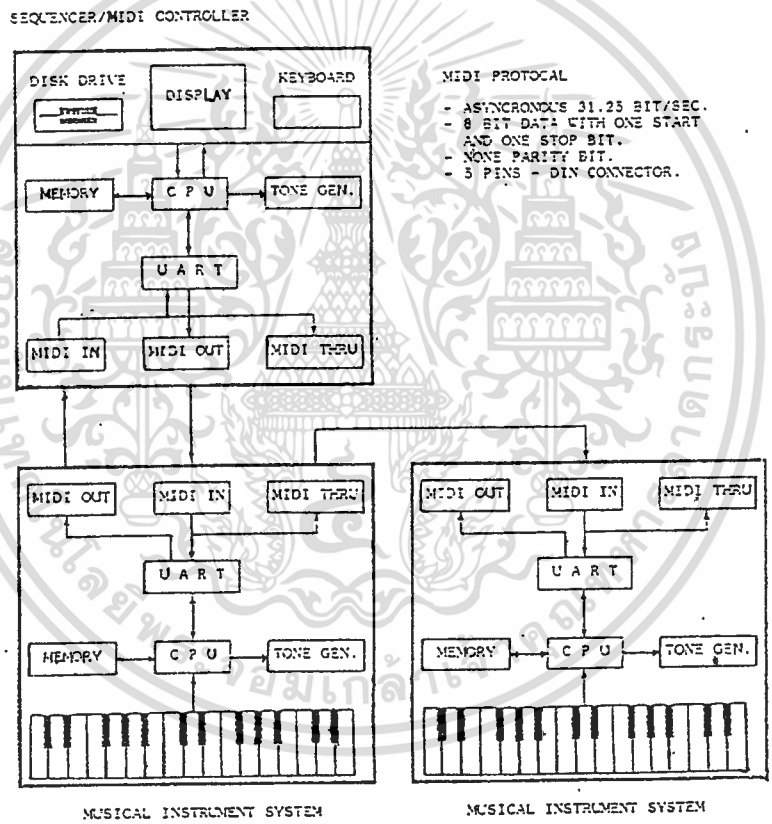
-ทำให้ประหยัดเครื่องดนตรีและอุปกรณ์ ลงไปมากเพราะ ไมโครคอมพิวเตอร์ (MICROCOMPUTER) ตัวเดียว สามารถทำหน้าที่แทนอุปกรณ์ได้หลายชนิด

-เป็นพื้นฐานในการศึกษาถึง ระบบมิดี้ (MIDI) เนื่องจากระบบนี้ยังไม่แพร่หลายเท่าไรนักสำหรับในประเทศไทยและอาจเป็นพื้นฐานในการพัฒนาเครื่องดนตรีอื่น ๆ โดยเฉพาะดนตรีไทยให้สามารถนำมาประยุกต์เล่นภายในระบบนี้ได้ ซึ่งเป็นระบบที่เป็นมาตรฐานทั่วโลก

บทที่ 2
หลักการ และ ข้อกำหนดในระบบมิดี้

หลักการ

มิดี้ (MIDI) ย่อมาจาก Musical Instrument Digital Interface เป็นระบบมาตรฐานสำหรับการสื่อสารข้อมูลระหว่างเครื่องดนตรี โดยจะมีการสื่อสารกันแบบ อซิงโครนัส (asynchronous) ซึ่งเขียนแผนภาพแสดงการทำงานได้ดังนี้



รูปที่ 2.1 แผนภาพแสดงโครงสร้างของเครื่องดนตรี

จากรูป อุปกรณ์ควบคุม (controlling instrument) เช่น คอมพิวเตอร์สามารถควบคุมการทำงานของเครื่องดนตรีได้โดยคอมพิวเตอร์จะส่งรหัสมิดี้ไปยังคีย์บอร์ดตัวที่ 1 และใช้ออพโต้ไอโ

ซเลเตอร์(optoisolator)เป็นตัวคัปปลิ้งสัญญานส่วน UART จะเปลี่ยนข้อมูลแบบขนานให้เป็นแบบอนุกรม (และเปลี่ยนจากแบบอนุกรมให้เป็นแบบขนานด้วย)

ข้อมูลที่ได้จะเข้ามายัง ส่วนควบคุมภายในเครื่องดนตรี (micro controller) พอตีความแล้วสั่งให้คีย์บอร์ดทำงานตามรหัสมิดี้ที่รับได้

ข้อมูลจากออพโต้ไอโซเลเตอร์ส่วนหนึ่งจะผ่านไปยังบัฟเฟอร์แล้วส่งออกไปเรียกว่า มิดี้ธรู (MIDI thru) ซึ่งก็คือการลอกแบบข้อมูลที่ได้รับมานั่นเอง ข้อมูลส่วนนี้จะถูกส่งไปให้คีย์บอร์ด 2 จะเห็นได้ว่า คีย์บอร์ดตัวแรกกับตัวที่สองจะติดต่อกันได้ โดยคีย์บอร์ดตัวแรกจะควบคุมคีย์บอร์ดตัวที่สอง ซึ่งอาศัยข้อมูลที่ได้มาจากคอมพิวเตอร์

ถ้าไม่อาศัยข้อมูลจากคอมพิวเตอร์ เมื่อมีการเล่น คีย์บอร์ด 1 (ใช้คนเล่น) ส่วนควบคุมภายในคีย์บอร์ด 1 จะส่งรหัสมิดี้ออกไป และคอมพิวเตอร์จะได้รับข้อมูลนี้ แล้วนำมาทำขบวนการต่างๆ ได้เช่น เก็บข้อมูลนี้ไว้ในดิสก์ ในลักษณะนี้จะคล้ายกับเทปบันทึกเสียงแต่แทนที่จะเก็บเป็นสัญญาณเสียง คอมพิวเตอร์จะเก็บในรูปของรหัสมิดี้ และเมื่อต้องการเล่นซ้ำ (play back) ทางคอมพิวเตอร์ก็จะส่งรหัสมิดี้นี้กลับไปยังเครื่องดนตรี และเครื่องดนตรีก็จะทำงาน (เล่น) ตามรหัสมิดี้เดิม ซึ่งเป็นการสร้างเสียงขึ้นใหม่ (regenerate) นั่นเอง

จะเห็นว่าเมื่อเครื่องดนตรีสามารถสื่อสารกันภายใต้มาตรฐานเดียวกันแล้วเราสามารถนำไปประยุกต์ใช้งานได้อย่างกว้างขวาง เช่น

1. ควบคุมเครื่องดนตรีได้ทีละหลาย ๆ เครื่อง : เนื่องจากเครื่องดนตรีทุกเครื่องสื่อสารด้วยภาษาเดียวกัน

2.สามารถส่งรหัสมิดี้ไปควบคุมกลองอิเล็คทรอนิก (Drum machine setups) เพื่อให้มีจังหวะตรงกับเครื่องดนตรีอื่น ๆ (synchronized) ได้

3. อุปกรณ์แต่งเสียง (sound effect) ต่างๆ สามารถควบคุมได้ด้วยรหัสมิดี้

4. ใช้ในการสร้างเสียง (voice quality) ของเครื่องดนตรีประเภท ซินธิไซเซอร์ (voice editing software)

5. ข้อมูลจากเครื่องดนตรีเครื่องหนึ่งอาจนำไปใช้กับอีกเครื่องได้ (Data transfer)

6.ถ้าใช้การเก็บรหัสมิดี้ แทนที่จะเก็บเป็นสัญญาณเสียง เราสามารถเล่นซ้ำได้ดังที่กล่าวไปแล้ว ซึ่งจะคล้ายกับการบันทึกเสียง (Tepeless recording)

7.ประโยชน์จากการสื่อสาร เช่นสามารถส่งข้อความมิดี้ผ่านโมเด็ม (modem) ได้ ดังนั้นแม้จะอยู่กันคนละประเทศ นักดนตรีก็ยังสามารถส่งดนตรีของเขาไปให้ห้องอัดเสียงได้ทางโทรศัพท์ โดยไม่มีความเพี้ยนของเสียงดนตรีเลยเนื่องจากส่งเป็นโค๊ด

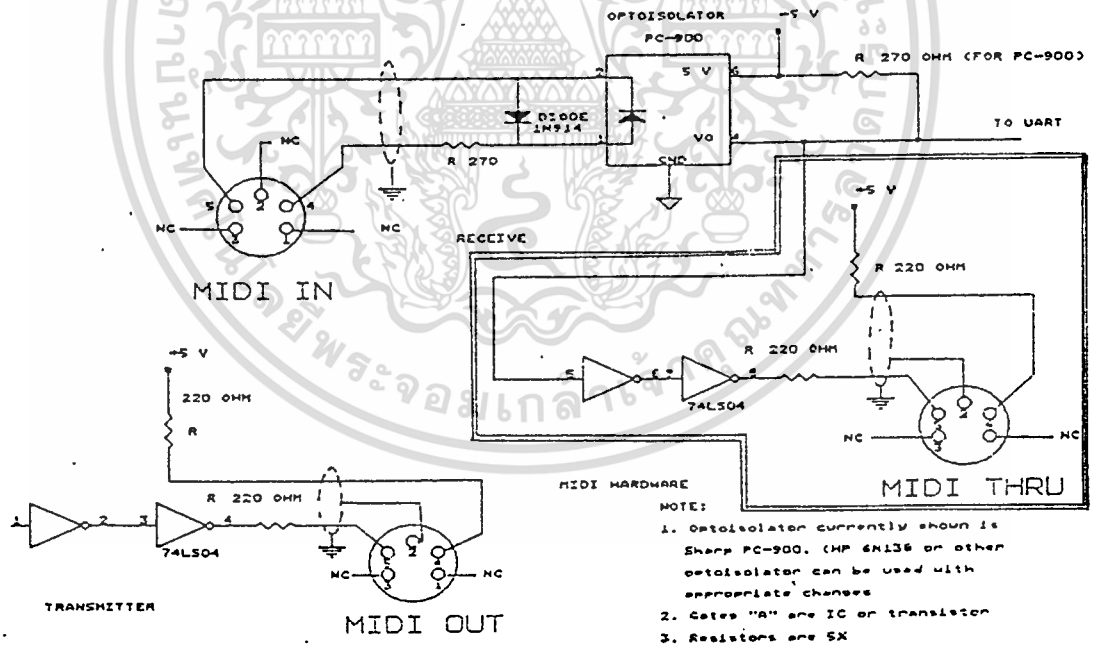
สำหรับเครื่องดนตรีที่ไม่ใช้ไฟฟ้า (acoustic) ก็สามารถใช้พ่วงกับเครื่องดนตรีอื่นด้วย ระบบมิดี้ได้ถ้า

เราสามารถทำให้มันส่งข้อมูลออกไปตามข้อกำหนดของระบบมิดี้

ข้อกำหนดของระบบมิดี้

ข้อกำหนดโดยทั่วไป

มิดี้เป็นมาตรฐานที่จะให้ ซินธิไซเซอร์ (synthesizer), ซีเควนเซอร์ (sequencer)คอมพิวเตอร์, อุปกรณ์สร้างจังหวะ ฯลฯ สามารถติดต่อสื่อสารกันได้



รูปที่ 2.2 แผนภาพแสดงข้อกำหนดในการเชื่อมต่อ

อุปกรณ์ที่ใช้ระบบมิดี้ มักจะรับ - ส่ง ข้อความระบบมิดี้ได้ โดยทางด้านรับจะรับข้อความ (message)ที่เป็นรหัสมิดี้แล้วทำงานตามความหมายของข้อความนั้น อุปกรณ์ทางฮาร์ดแวร์ที่สำคัญ

ประกอบด้วยออพโต้ไอโซเลเตอร์, Universal Asynchronous Receiver-Transmitter(UART) ส่วนทางด้านส่งก็จะส่งรหัสมิดี้ออกไปโดยมี UART และ ตัวขับสัญญาณ (line driver) เป็นองค์ประกอบสำคัญ

ฮาร์ดแวร์

สื่อสารแบบอซิงโครนัส โดยใช้ความเร็วในการรับ - ส่งที่ 31.25 kBaud มีบิทเริ่ม_และบิทหยุด (start bit, stop bit) อย่างละ 1 บิท, บิทข้อมูลมี 8 บิท รวมทั้งหมด 10 บิท ต่อข้อมูล 1 ไบท์ ซึ่งใช้เวลาส่ง 320 ไมโครวินาที มีวงจรเป็นรูป MIDI HARDWARE

จากรูป กระแสในลูป = 1.5 mA โดยที่ โลจิก 0 จะมีกระแสไหล ทางด้านรับควรมีออพโต้ไอโซเลเตอร์เป็นตัวคัปปลิ้งสัญญาณ ซึ่งเป็นชนิดทำงานที่ความเร็วสูงเช่น Sharp PC - 900 เวลาในการสวิทช์ ควรน้อยกว่า 2 ไมโครวินาที

อุปกรณ์สำหรับเชื่อมต่อ (connector)

เป็นแบบ DIN 5 ขา (แจ็คกลม) ตัวเมีย ขาที่ใช้คือขา 5,4 ส่วนขา 2 ต่อกราวนด์ นอกจากนั้นไม่ใช้ปล่อยไว้เฉย ๆ

ความยาวของสาย (MIDI cable) ไม่เกิน 15 เมตร หรือ 50 ฟุต และควรมีการพันเกลียว (twist) เพื่อลดสัญญาณรบกวน

ชื่อของคอนเนคเตอร์ : MIDI IN (ข้อมูลเข้า) , MIDI OUT (ข้อมูลออก) และMIDI THRU ซึ่งเป็นข้อมูลขาออกที่ได้มาจากการก๊อปปี้ข้อมูล มิดี้ อิน โดยตรง

รูปแบบของข้อมูล (data format)

แต่ละข้อความในระบบมิดี้ มักมีหลายไบท์โดยประกอบด้วยไบท์สถานะ (staus byte) ตามด้วยไบท์ข้อมูล (data byte) อีก 1 หรือ 2 ไบท์ ยกเว้นข้อความเกี่ยวกับเวลา (real time -messaage) และข้อความพิเศษ (exclusive message) ซึ่งจะกล่าวถึงภายหลัง

ประเภทของข้อความจะแบ่งออกเป็น 2 ประเภทใหญ่ ๆ คือ แชนแนล กับ ระบบ

ข้อความเกี่ยวกับแชนแนล

ในไบท์สถานะจะใช้ 4 บิท เป็นตัวกำหนดแชนแนลของไบท์สถานะ (สามารถกำหนดได้ 16 แชนแนล)ทางด้านรับจะตอบสนองเฉพาะไบท์สถานะที่มีแชนแนลตรงกับแชนแนลของตนเท่านั้น ถ้าไม่ตรงกันจะไม่สนใจ

ข้อความเกี่ยวกับแชนแนลมี 2 ประเภท

Voice - ใช้ควบคุมเสียง ข้อความเกี่ยวกับเรื่องเสียงจะถูกส่งผ่านทางแชนแนลเสียง(voice channel)

Mode - ใช้กำหนดลักษณะการตอบสนองต่อข้อความเกี่ยวกับเรื่องเสียงของเครื่องดนตรี โดยจะส่งมาทาง แชนแนลพื้นฐาน (Basic channel) ของเครื่องดนตรี

ข้อความเกี่ยวกับระบบ

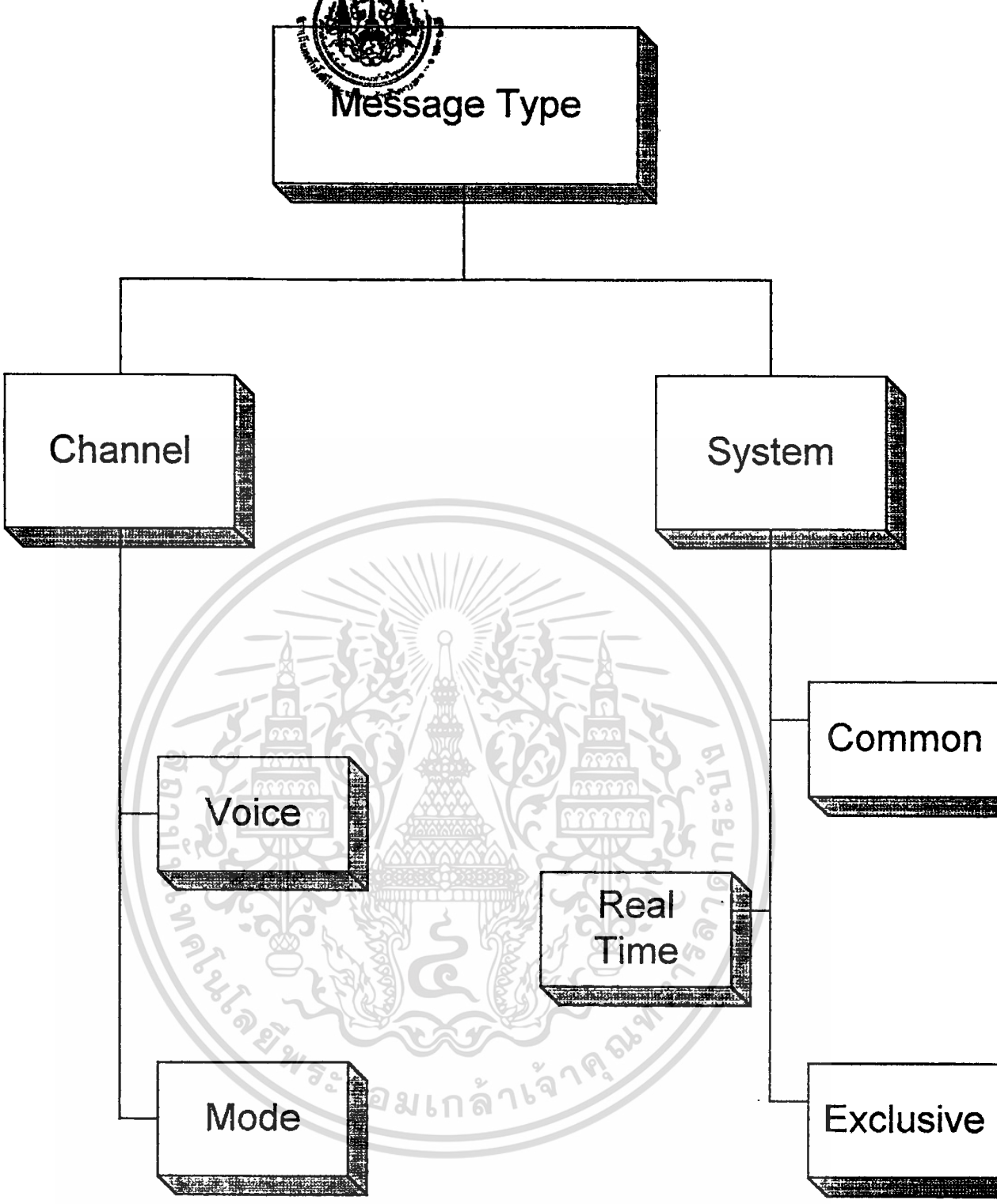
ข้อความเกี่ยวกับระบบจะไม่มีการกำหนดแชนแนล ดังนั้นไม่ว่าด้านรับจะเป็นแชนแนลไหนก็จะตอบสนองต่อข้อความเกี่ยวกับระบบเสมอ

ข้อความเกี่ยวกับระบบมี 3 แบบ

1. ข้อความทั่วไป (common message) เป็นข้อความสำหรับทุกอุปกรณ์ในระบบ
2. ข้อความเกี่ยวกับเวลา จะมีแต่ไบท์สถานะเท่านั้น ไม่มีไบท์ข้อมูล สำหรับข้อความประเภทนี้จะส่งเมื่อไหร่ก็ได้ แม้แต่ส่งคั่นระหว่างไบท์ของข้อความอื่น
3. ข้อความพิเศษ (exclusive message) มีหลายไบท์ข้อมูล และจะจบด้วยไบท์ที่เรียกว่า End Of Exclusive (EOX) หรือไบท์สถานะอื่น ๆ เป็นข้อความที่ใช้เฉพาะกับอุปกรณ์ของผู้ผลิตที่กำหนดไว้ในรหัสประจำเครื่องของแต่ละบริษัทผู้ผลิตหรือ Identification Codes (ID codes) เท่านั้น ถ้าอุปกรณ์ทางด้านรับมีรหัสประจำเครื่องไม่ตรงกับข้อความพิเศษที่ได้ มันจะไม่สนใจไบท์ข้อมูลที่ตามมา ผู้ผลิตแต่ละรายสามารถกำหนดรูปแบบ ความหมายข้อมูลของตนเองได้ โดยส่งเป็นข้อความพิเศษ ที่มีรหัสประจำเครื่องของตน (ซึ่งเป็นรหัสเดียวกันกับทางด้านรับ)

ประเภทของข้อมูล (data type)

ไบท์สถานะ มี 8 บิท โดยมีบิทที่มีนัยสำคัญสูงสุด (MSB) ถูกเซ็ท (เป็น 1)ไบท์สถานะจะเป็นตัวบอกถึงประเภทของข้อความ และบอกให้รู้ว่าไบท์ข้อมูลที่ตามมาจะเอาไปใช้เพื่ออะไร



รูปที่ 1.3 แสดงถึงชนิดต่างๆ ของ Message

สถานะที่กำลังทำงาน (running status) สำหรับข้อความเกี่ยวกับเสียง และโหมด เท่านั้น เมื่อด้านรับได้รับไบท์สถานะและทำขบวนการต่างๆ ด้านรับจะยังคงอยู่ในสถานะนั้นจนกว่าจะได้รับไบท์สถานะใหม่ที่ไม่เหมือนเดิม ดังนั้น ถ้ามีการส่งไบท์สถานะเดิมซ้ำมาอีกครั้ง อาจทำเมื่อต้องการแก้ไบท์ข้อมูลที่ผิดโดยส่งใหม่ ภายใต้สถานะที่กำลังทำงานข้อความที่สมบูรณ์ต้องประกอบด้วยไบท์ข้อมูลที่ถูกต้องตามข้อกำหนด

สถานะที่กำลังทำงานจะหยุดเมื่อมีไบท์สถานะใหม่เข้ามา เว้นแต่ ถ้ามีข้อความเกี่ยวกับเวลาเข้ามามันจะขัดจังหวะสถานะที่กำลังทำงานแต่เพียงชั่วคราวเท่านั้น

สถานะที่ไม่สามารถทำงานได้ (Unimplement Status) ทางด้านรับจะไม่สนใจไบท์สถานะที่มันไม่สามารถทำงานตามได้ และจะไม่สนใจไบท์ข้อมูลที่ตามมา

สถานะที่ไม่มีการกำหนดความหมาย (Undefined Status) จะไม่มีการใช้ไบท์สถานะที่ไม่มีความหมาย ควรระวังในขณะเปิดเครื่อง หรือ ปิดเครื่อง อาจส่งข้อมูลผิดพลาดได้

ไบท์ข้อมูล จะถูกส่งตามหลังไบท์สถานะ โดยมีไบท์ข้อมูลประมาณ 1-2 ไบท์ (ยกเว้นข้อความเกี่ยวกับเวลา) ไบท์ข้อมูลมี 8 บิท. โดยบิทที่มีนัยสำคัญต่ำสุดถูกรีเซ็ท (เป็น 0) จำนวนและขอบเขต (ค่าที่มีได้)ของไบท์ข้อมูล จะเป็นไปตามที่ไบท์สถานะกำหนด ในแต่ละไบท์สถานะจะส่งไบท์ข้อมูลจำนวนที่ถูกต้องตามไปเสมอ

การตอบสนองต่อข้อความจะมีได้ก็ต่อเมื่อด้านรับได้รับไบท์ข้อมูลตามต้องการ แล้วทางด้านรับจะไม่สนใจกับไบท์ข้อมูลที่ตามหลังไบท์สถานะที่มันไม่รู้จัก

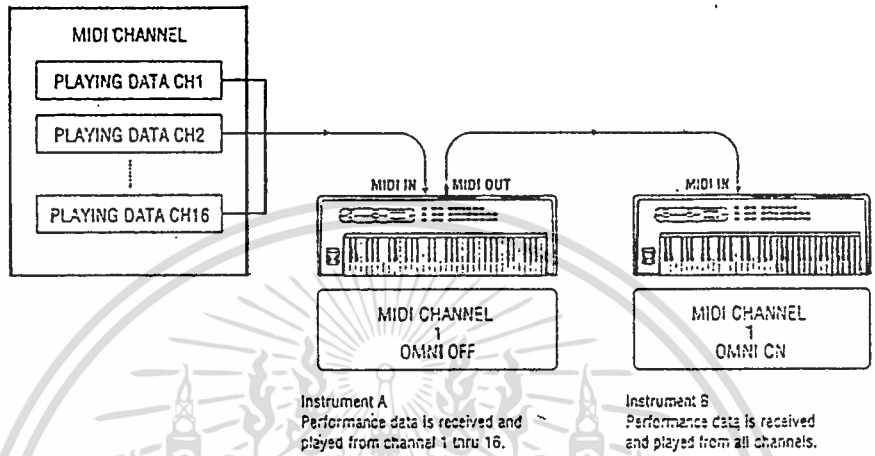
โหมดต่าง ๆ ในแต่ละแชนแนล (Channel Mode)

ซินธิไซเซอร์จะประกอบด้วย ส่วนที่สร้างเสียง (sound generator) เสียงที่เกิดขึ้นเรียกว่า voice การกำหนดเสียงเป็นขบวนการเกี่ยวกับข้อมูลพวกการเล่นโน๊ต (note on)กับการเลิกเล่นโน๊ต (note off) จากคีย์บอร์ดซึ่งเสียงที่มีโน๊ตต่างๆ จะถูกเล่นได้ในจังหวะที่ถูกต้อง

ข้อความเกี่ยวกับโหมด (mode message) ใช้กำหนดลักษณะการสร้างเสียงของเครื่องดนตรีประเภท ซินธิไซเซอร์ ได้แก่ Omni (On/off),Poly และ Mono

Mono on จะเป็นการกำหนดให้เสียงในแชนแนลเสียงมีได้ 1 เสียงเท่านั้นในเวลาเดียวกัน(Mono phonic)

Mono off (Poly on) ซินธิไซเซอร์ที่อยู่ในโหมดนี้สามารถสร้างเสียงได้พร้อมๆกันที่ละหลาย ๆเสียง (Poly phonic) ซึ่งทั้งนี้จะขึ้นอยู่กับความสามารถในการสร้างเสียงของตัวซินธิไซเซอร์ด้วย



รูปที่ 2.4 แผนภาพแสดงการทำงานในโหมดต่างๆ

สำหรับ Omni ถ้า Omni on ตัวรับจะสามารถรับข้อความเกี่ยวกับเสียงได้จากทุกแชนแนลเสียงโดยไม่มีความต่างกันของแชนแนล ถ้า Omni off ด้านรับจะรับเฉพาะข้อความเกี่ยวกับเสียงที่มีแชนแนลตรงกับของตนเท่านั้น

ทางด้านรับที่ถูกกำหนดเป็นแชนแนลพื้นฐานจะมีลักษณะการทำงานได้ 4 โหมดดังนี้

โหมด 1. Omni On , Poly

โหมด 2. Omni On , Mono

โหมด 3. Omni off, Poly

โหมด 4. Omni off, Mono

ทางด้านส่งและรับจะทำงานได้ทีละ 1 โหมดเท่านั้นในเวลาเดียวกันและตามปกติจะส่งและรับในโหมดเดียวกัน

ด้านรับจะรับรู้ข้อความเกี่ยวกับโหมดทางแชนแนลพื้นฐานเท่านั้น ส่วนข้อความเกี่ยวกับเสียงอาจได้รับทางแชนแนลพื้นฐานหรือแชนแนลอื่น ๆ ก็ได้ซึ่งเรียกรวมกันว่าแชนแนลเสียง

โดยทั่วไปมักกำหนดให้แชนแนลแรก (0) เป็นแชนแนลพื้นฐาน

ความหมายของข้อความต่าง ๆ ในระบบมิดี้

1. ข้อความที่เกี่ยวกับเสียงในแต่ละแชนแนล (channel voice message)

การเล่นโน๊ต (note on)

เป็นข้อความที่บอกให้รู้ถึงการเล่นโน๊ต (เช่นการเอานิ้วกดไปที่คีย์บอร์ด) มี 3 ไบท์

1. ไบท์สถานะคือ 1001 nnnn (ฐานสอง) โดยที่ nnnn คือหมายเลขแชนแนล (0-15)

2. ไบท์ข้อมูลแรกจะบอกให้รู้ว่าโน๊ตไหนที่ถูกกด (note number)มีค่าได้ตั้งแต่ 0-127โดย 0 จะเป็นโน๊ตที่ต่ำที่สุด 127 เป็นโน๊ตที่สูงที่สุด แต่โดยทั่วไปเครื่องดนตรีจะรับรู้โน๊ตได้น้อยกว่า เช่น ซินธิไซเซอร์ ที่มี 5 ออคเตฟ จะรับรู้หมายเลขโน๊ตได้ ตั้งแต่ 36 - 96 หมายเลขโน๊ตของโดกลาง (middle C) คือ 60

3. ไบท์ข้อมูลต่อมาจะบอกถึงน้ำหนักในการเล่นหรือความเร็วที่กดคีย์(key velocity)เหมือนกับการเล่นเปียโน ถ้ากดคีย์เปียโนด้วยความเร็วจะได้เสียงที่ดัง และ ถ้ากดช้าก็จะได้เสียงที่เบาเช่นกัน ความเร็วของคีย์มีค่าได้ตั้งแต่ 0 - 127 โดยความเร็ว = 127 จะเป็นเสียงที่ดังที่สุด ส่วนความเร็ว = 0 จะไม่มีเสียงออกมา (note off) ซึ่งมีความหมายเหมือนการยกนิ้วขึ้นจากคีย์ สำหรับเครื่องดนตรีที่ไม่มีการตอบสนองต่อความเร็วของคีย์จะกำหนดให้ข้อมูลส่วนนี้มีค่าเท่ากับ 64

การเลิกเล่นโน๊ต (note off)

เป็นข้อความที่บอกให้รู้ถึงการเลิกสร้างเสียง (เช่นเกิดขึ้นในขณะที่ยกนิ้วขึ้นจากคีย์) มี 3 ไบท์

1. ไบท์สถานะ คือ 1000nnnn (ฐานสอง) โดยที่ nnnn คือ หมายเลขแชนแนล

2. ไบท์ข้อมูลแรกบอกให้รู้ว่าโน๊ตตัวไหนที่จะให้เลิกสร้างเสียง (key number)

3.ความเร็วที่ปล่อยคีย์ (key off (release) velocity) เป็นไบท์ข้อมูลที่บอกถึงความเร็วในการปล่อยคีย์ ในกรณีที่อุปกรณ์ไม่มีความเร็วของคีย์จะให้ไบท์นี้มีค่าเป็น 64

เมื่อเครื่องดนตรีได้รับข้อความเกี่ยวกับการเล่นโน๊ต มันจะผลิตเสียงค้างอยู่อย่างนั้นจนกว่าจะได้รับข้อความเกี่ยวกับการเลิกเล่นโน๊ตของโน๊ตตัวนั้นมันถึงจะหยุดทำเสียง ดังนั้นเมื่อมีการส่งข้อความเกี่ยวกับการเล่นโน๊ตออกไปแล้ว จะส่งข้อความเกี่ยวกับการเลิกเล่นโน๊ตตามมาด้วยทุกครั้ง มิฉะนั้นจะเกิดเสียงค้าง

การกำหนดความดันของคีย์แบบโพลีโฟนิค (Polyphonic Key pressure)

คีย์บอร์ดในระบบยมิดี้บางเครื่องไม่เพียงแต่ตอบสนองความเร็วของคีย์ได้เท่านั้นยังตอบสนองน้ำหนัก (ความดัน) ของนิ้วที่กดลงไปอีกหลังจากที่กดคีย์ลงไปสุดแล้ว ประกอบด้วย 3 ไบท์

1. ไบท์สถานะ คือ 1010nnnn

2. หมายเลขของโน๊ต

3. ความดันของคีย์ (key pressure value) มีค่าตั้งแต่ 0 - 127 โดย 0 หมายถึง ไม่มีความหมายดันเลย และ 127 มีความดันสูงสุด

ประโยชน์ของโพลีโฟนิคคีย์เพรชเชอร์ คือจะให้ความรู้สึกของเครื่องเป่า เช่น ทรัมเป็ดเมื่อดังเสียงของซินธิไซเซอร์เป็นเสียงทรัมเป็ด การกดคีย์ลงไปอีกหลัง จากสุดคีย์แล้วจะมีลักษณะเหมือนการที่เราเป่าลมเข้าไปในทรัมเป็ดอีก

คีย์บอร์ดมีโพลิโฟนิคคีย์เพรชเชอร์จะมีระบบที่ซับซ้อน และมีราคาแพงมาก ซึ่ง จะหาคีย์บอร์ดที่มีระบบนี้ได้น้อยราย

การกำหนดความดันรวม (Overall Pressure)

ต่างกับโพลิโฟนิคคีย์เพรชเชอร์ตรงที่ไม่สามารถบอกความดันแยกกันของแต่ละคีย์ที่ถูกกดได้ แต่จะบอกเป็นความดันเฉลี่ยของทุก ๆ คีย์ในเวลานั้น ๆ มีอยู่ด้วยกัน 2 ไบท์

1. ไบท์ข้อมูลคือ 1101nnnn (nnnn คือ หมายเลขแชนแนล)

2. ความดันของแชนแนล (channel pressure value) มีค่าตั้งแต่ 0 - 127 โดยที่ 0 จะไม่มีความดัน และ 127 มีความดันมากที่สุด

ข้อมูลเกี่ยวกับการควบคุม (Contol Change)

ใช้ในการปรับแต่งเสียง สำหรับข้อมูลเกี่ยวกับการควบคุมในปัจจุบันยังไม่มี มาตรฐานที่แน่นอนระหว่างแต่ละบริษัทผู้ผลิตมี 3 ไบท์

1. ไบท์สถานะ คือ 1011nnnn

2.หมายเลขตัวควบคุม (controller number) มีค่าตั้งแต่ 0 - 127 ดูความหมายได้จาก MIDI Implementation Sheet ของแต่ละบริษัทผู้ผลิต

3. ค่าที่จะให้แก่ตัวควบคุม (controller value)

การเอื้อนเสียง (Pitch Bend)

เป็นลักษณะของการเอื้อนเสียงจากสูงไปต่ำหรือจากต่ำไปสูงโดยที่ไม่ต้อง เปลี่ยนตำแหน่งของคีย์ที่กด การเอื้อนเสียงมี 2 ไบท์

1. ไบท์สถานะ คือ 1110nnnn

2. ค่าที่จะให้เอื้อน (pitch bend value)

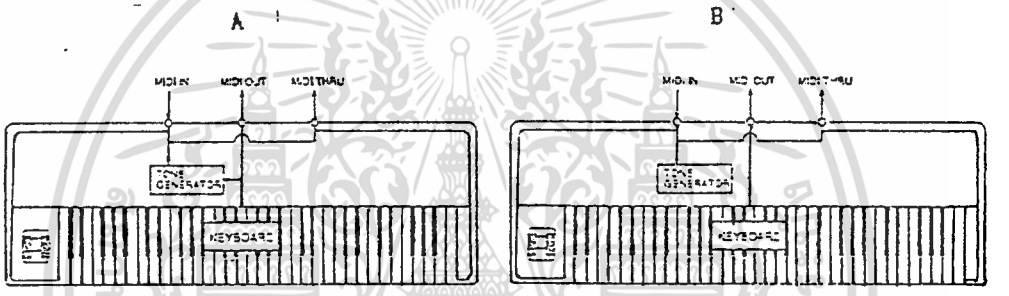
ข้อมูลเกี่ยวกับโปรแกรมเสียง (Program Change) หรือบางทีเรียกว่า Program Select จะบอกให้รู้ถึงการใช้โปรแกรมเสียงต่าง ๆ ในเครื่องดนตรี มี 2 ไบท์

1. ไบท์สถานะคือ 1100nnnn

2. โปรแกรมเสียงที่ใช้ (selected program number) มีได้ตั้งแต่ 0 ถึง 127

2. ข้อมูลเกี่ยวกับลักษณะการทำงานในแต่ละแชนแนล(Channel Mode Messages)

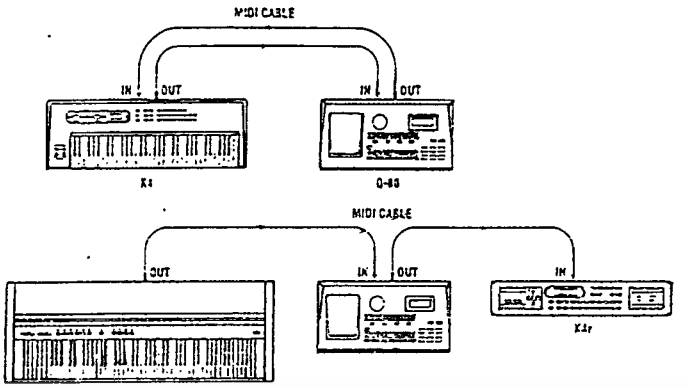
เป็นข้อความเกี่ยวกับโหมดในการทำงาน เป็นส่วนหนึ่งของข้อความเกี่ยวกับการควบคุมคีย์บอร์ดแบบ โลคอล/รีโหมด (Local/Remote Keyboard Control) ใช้ในการพ่วงเครื่องดนตรีเข้าด้วยกันโดยควบคุมจากเครื่องเดียว



รูปที่ 2.5 แผนภาพแสดงทางเดินของสัญญาณ

จากรูป (a) ถ้าเป็นการควบคุมแบบโลคอล (Local Control On) ข้อมูลทางด้านเสียง (เช่น การเล่นโน๊ต) จากคีย์บอร์ดจะเข้าไปยังส่วนสร้างเสียงและข้อมูลเดียวกันนี้จะออกไปทางมิดีเอ๊าท์ด้วย (ข้อมูลจากมิดี้เอ๊าท์ใช้ส่งให้กับอุปกรณ์อื่น) ดังนั้นในการพ่วงซินธิไซเซอร์เข้าด้วยกันดังรูปที่แสดงการเชื่อมต่อเมื่อเล่น ซินธิไซเซอร์ # 1 จะมีเสียงออกมาจากทั้ง ซินธิไซเซอร์ #1 และ ซินธิไซเซอร์ #2 (ซินธิไซเซอร์ #2 เล่นตามข้อมูลที่ส่งมาจาก ซินธิไซเซอร์ #1)

ในกรณีที่เป็น การควบคุมแบบรีโหมด ((Local control off) (รูป(b)) ข้อมูลจากคีย์บอร์ดจะส่งไปที่มิดี้เอ๊าท์โดยตรง (ไม่ส่งให้ส่วนสร้างเสียง) ดังนั้นในรูปแสดงการเชื่อมต่อถ้าอยู่ในลักษณะการควบคุมแบบรีโหมด เมื่อเล่นที่ซินธิไซเซอร์ #1 จะมีเสียงออกมาจาก ซินธิไซเซอร์ #2 เท่านั้น



รูปที่ 2.6 แผนภาพแสดงการทำงานการเชื่อมต่ออุปกรณ์มิดี้

ข้อมูลเกี่ยวกับการควบคุมคีย์บอร์ดแบบ โลคอล/รีโหมด (มี 3 ไบท์)

1.ไบท์สถานะ คือ 1011nnnn (ซึ่งเป็นไบท์สถานะของข้อความเกี่ยวกับการควบคุม) nnnn คือ แชนแนลพื้นฐาน

2. หมายเลขตัวควบคุม = 122

3. ถ้าเป็นแบบรีโหมดจะให้ไบท์นี้เป็น 0, ถ้าเป็นแบบโลคอลจะเป็น 127

การเลิกเล่นโน๊ตทุกตัว (All Note Off)

เป็นการหยุดการการสร้างเสียง (หยุดทั้งแชนแนลโดยไม่สนว่าเป็นโน๊ตตัวไหน) มี 3 ไบท์

1.ไบท์สถานะคือ 1011nnnn (ซึ่งเป็นไบท์สถานะของข้อความเกี่ยวกับการควบคุม) nnnn คือ แชนแนลพื้นฐาน

2. หมายเลขตัวควบคุม = 123

3. เป็นไบท์ที่ไม่มีความหมายอะไร ใส่ไปให้ครบ 3 ไบท์เท่านั้น (dummy byte) เนื่องจากข้อความเกี่ยวกับการควบคุมมี ความยาว 3 ไบท์

การเลือกโหมด Omni/Poly/Mono

ความหมายของแต่ละโหมดต่างๆได้อธิบายไปแล้วในตอนต้น ทุกครั้งที่เปลี่ยนโหมดจะเลิก_เล่นโน๊ตทุกตัว รูปแบบของข้อความนี้ก็คล้าย ๆ กับข้อความเกี่ยวกับการควบคุมอื่น ๆ ประกอบด้วย

1. ไบท์สถานะคือ1011nnnn (nnnn เป็นแชนแนลพื้นฐาน)

2. หมายเลขตัวควบคุมของแต่ละโหมด เป็นดังนี้

Omni Off = 124

Omni On = 125

Mono On = 126

Poly On = 127

3. ไบท์ที่ 3 จะให้เป็น 0 สำหรับ ตัวควบคุมหมายเลข 123,124,125,127 ในกรณีของโหมดโมโน (126) ไบท์จะเป็นการกำหนดแชนแนลที่จะให้เป็นโมโน

3. **ข้อมูลทั่วไปของระบบ** (System Common Message) (สำหรับทุกแชนแนล)

เนื่องจากข้อมูลประเภทนี้เป็นข้อมูลของระบบ ดังนั้นจะไม่มีการกำหนดแชนแนล นั่นคือทุกแชนแนลต้องตอบสนองต่อข้อมูลทั่วไปของระบบ

การชี้ตำแหน่งของเพลง (Song Position Pointer)

บอกให้รู้ถึงตำแหน่งของเพลงที่กำลังเล่นอยู่ การบอกตำแหน่งจะบอกว่าเพลงได้ดำเนินไปแล้กี่บิท (MIDI beat) บอกได้สูงสุด 16, 384 บีท โดย 1 บีทมีค่าเท่ากับโน๊ตเบ็ทสองชั้นหนึ่งตัว

การเลือกเพลง (Song Select)

คล้ายๆ กับการเลือกโปรแกรมเสียง แต่แทนที่จะเป็นโปรแกรมเสียงจะเป็นโปรแกรมรูปแบบ (pattern) สั้น ๆ ของเพลงแทน (ความยาวไม่กี่ห้อง) โดยมากจะใช้ในกลองอีเลคโทรนิคแต่ละเพลงก็คือแต่ละจังหวะ เช่น เพลงที่หนึ่ง (song 1) เป็น จังหวะร็อค และเพลงที่สอง (song_2) เป็นจังหวะวอลซ์

4. **ข้อความพิเศษของระบบ** (System Exclusive) : ได้อธิบายแล้วในตอนต้น

5. **ข้อความเกี่ยวกับเวลาของระบบ** (System Real Time Message) (ทุกแชนแนล)

เป็นข้อความที่ทำให้เครื่องดนตรีต่าง ๆ เล่นอยู่ในจังหวะเดียวกัน (Synchronized) เช่นในการพ่วงซินธิไซเซอร์ กับ กลองอิเล็คโทรนิคเข้าด้วยกันจำเป็นอย่างยิ่งที่อุปกรณ์ทั้งสองนี้จะต้องทำงานอยู่ในจังหวะเดียวกัน ไม่เช่นนั้นก็คงจะฟังไม่รู้เรื่องถ้าปล่อยให้ต่างฝ่ายต่างเล่น

การรีเซ็ทระบบ (System Reset)

เป็นข้อความที่บอกให้อุปกรณ์ระบบมิดี้ทุกชิ้นทำการรีเซ็ทตัวเอง (กลับไปอยู่ในสภาพแรกหลังจากเปิดไฟเข้าเครื่อง)

สัญญาณนาฬิกา (Timing Clock)

ในการกำหนดจังหวะจะมีการส่งสัญญาณนาฬิกาออกไปด้วยอัตรา 24 ลูก ต่อ โน๊ตตัวดำหนึ่งตัว (ส่งออกไปอย่างต่อเนื่อง) เพื่อให้เป็นจังหวะอ้างอิงในระบบควรใช้สัญญาณนาฬิกาเดียวกัน (มีอุปกรณ์ส่งสัญญาณนาฬิกาเพียงตัวเดียว)

การกลับไปที่จุดเริ่มต้น (Start From First Measure)

เป็นข้อความที่บอกให้ซีเควนเซอร์ และกลองอิเลคโทรนิคกลับไปที่จุดเริ่มต้นของเพลง แล้วเริ่มเล่นเพลงในทันทีที่ได้รับสัญญาณนาฬิกา นั่นคือจะบอกให้แต่ละอุปกรณ์เริ่มเล่นพร้อม ๆ กัน

การหยุด (Stop)

ข้อความนี้จะบอกให้หยุดการกระทำทุกอย่างที่มีผลมาจากการกำหนดจังหวะ (timing sensative)

การเริ่มเล่นต่อ (Continue Start)

บอกให้อุปกรณ์ทำการเล่นต่อหลังตำแหน่งเพลงที่ถูกหยุดครั้งสุดท้าย

การตรวจสอบสภาวะการทำงาน (Active Sensing)

เป็นข้อความที่ใช้แก้ปัญหาตัวส่งกับตัวรับถูกตัดขาดออกจากกัน (เช่นสายหลุด) ซึ่งจะก่อให้เกิดความผิดพลาด เช่น ส่งข้อความให้เล่นโน๊ตออกไปแล้วเกิดสายหลุด เมื่อทางด้านส่งส่งข้อความให้เลิกเล่นโน๊ตตามไป ก็จะไปไม่ถึงด้านรับ ทำให้เกิดเสียงค้างที่เครื่องดนตรีด้านรับ ปัญหานี้จะเกิดความเสียหายมากในการแสดงสด

แอคทีฟเซ็นซิ่ง จะทำงานโดยการส่งข้อความที่เรียกว่าแอคทีฟเซ็นซิ่ง ออกไปเรื่อย ๆ เมื่อด้านรับได้รับ แอคทีฟเซ็นซิ่ง แล้ว ถ้าภายในเวลาสั้น ๆ ที่กำหนดไม่ได้รับข้อความนี้เข้ามาอีก มันจะหยุดการสร้างเสียง

ที่กล่าวมาแล้วทั้งหมดเป็นความหมายของข้อความในระบบมิดี้ต่างๆแต่ โดยทั่วไปเครื่องดนตรีแต่ละชิ้นจะไม่สามารถตอบสนองต่อข้อความเหล่านี้ได้ทุกข้อความ ซึ่งความสามารถในการตอบสนองต่อข้อความต่าง ๆ ของแต่ละอุปกรณ์ จะหาดูได้ใน มิดี้ อิมพลีเมนท์เทชั่น ชีต (Midi Implementation Sheet) ของอุปกรณ์นั้นๆ

บทที่ 3
การสร้างและการใช้งานตัวกลางมีดี้

โครงงานนี้แบ่งออกเป็นสองส่วน คือ การสร้างตัวกลางมีดี้ และ เขียนโปรแกรมใช้งานตัวกลางมีดี้ที่สร้างขึ้นมา

การเลือกอุปกรณ์ในการรับ-ส่งข้อมูลแบบอนุกรมasynchronous
ซึ่งโดยทั่วไปสิ่งที่อุปกรณ์ที่ใช้กับระบบมีดี้ควรจะมีคือ

1. พอร์ทมีดี้เข้า (MIDI IN Port) ใช้รับข้อมูลมีดี้
2. พอร์ทมีดี้ออก (MIDI OUT Port) ใช้ในการสส่งข้อมูล
3. พอร์ทมีดี้ผ่าน (MIDI THRU Port) ใช้ในการทวนสัญญาณข้อมูล

การติดต่อแบบอนุกรมในระบบมีดี้ มีความเร็วในการรับ-ส่งข้อมูลประมาณ 31,250 บิทต่อวินาที (ผิดพลาดไม่เกิน 1 เปอร์เซ็น) ซึ่งสามารถใช้ความเร็วขนาดนี้ได้ง่ายโดยใช้แหล่งกำเนิดความถี่ 2MHz ข้อมูลประกอบด้วย 1 บิทเริ่มต้น (1 Start Bit) 8 บิทข้อมูล (8 Data Bits) และ 1 บิท สิ้นสุด (1 Stop Bit)

การทำงานของวงจรนี้ใช้กระแสขนาด 5 มิลลิแอมแปร์ (5 mA) เนื่องจากตัวรับข้อมูลเป็นแบบ Opto-isolate ดั้งนั้นไม่ควรใช้กระแสเกิน 5 mA สายที่ใช้ในการเชื่อมต่อระหว่างตัวกลางมีดี้กับอุปกรณีอื่นไม่ควรเกิน 50 ฟุต และที่ปลายด้านหนึ่งมีplugตัวผู้ขนาด 5 ขา เพื่อใช้เสียบเข้ากับเครื่องดนตรี

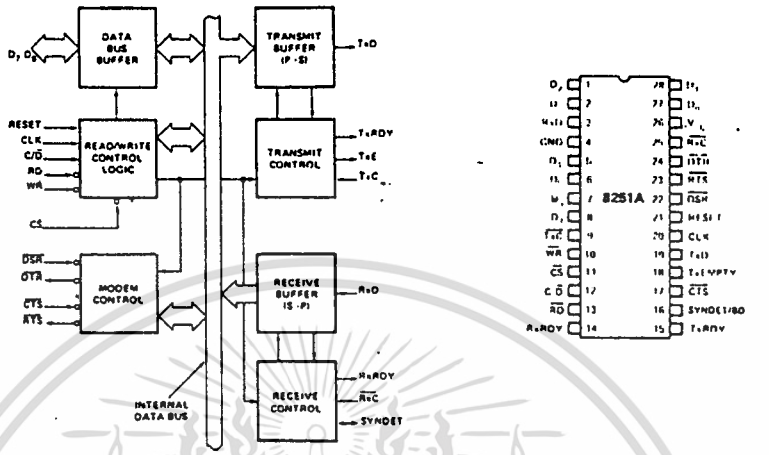
เนื่องจากตัวกลางมีดี้มีการส่งข้อมูลแบบอนุกรมจึงต้องใช้ยูอาร์ท (UART) ในการเปลี่ยนโอนข้อมูล (transfer) ในที่นี้เลือกใช้ IC8251

การจะเก็บค่าข้อมูลต่างๆนั้น เราจำเป็นต้องมีตัวนับเวลา (timer) จึงได้เลือก programable timer ของ Intel เบอร์ 8253 ตัว timer นี้แบ่งเป็น counter 3 ชุดด้วยกัน แต่เราจะใช้เพียง 2 ชุด เพราะฉะนั้นวงจรตัวกลางมีดี้จึงประกอบด้วยส่วนสำคัญ 3 ส่วนคือ

1. ส่วนของการเชื่อมระหว่างมีดี้และเครื่องดนตรี
2. ส่วนของวงจรสำหรับเชื่อมต่อ 8251 กับ computer
3. ส่วนของ timer และแหล่งกำเนิดความถี่

หลักการทำงานทั่วไปของ 5251 ในวงจร

เป็นอุปกรณ์รับ-ส่งข้อมูลระหว่าง Microcomputer กับอุปกรณ์ Input/Output แบบอนุกรม โดยทำงานได้ 2 ระบบ คือ Synchronous และ Asynchronous



รูปที่ 3.1 แสดงการทำงานของ IC8251

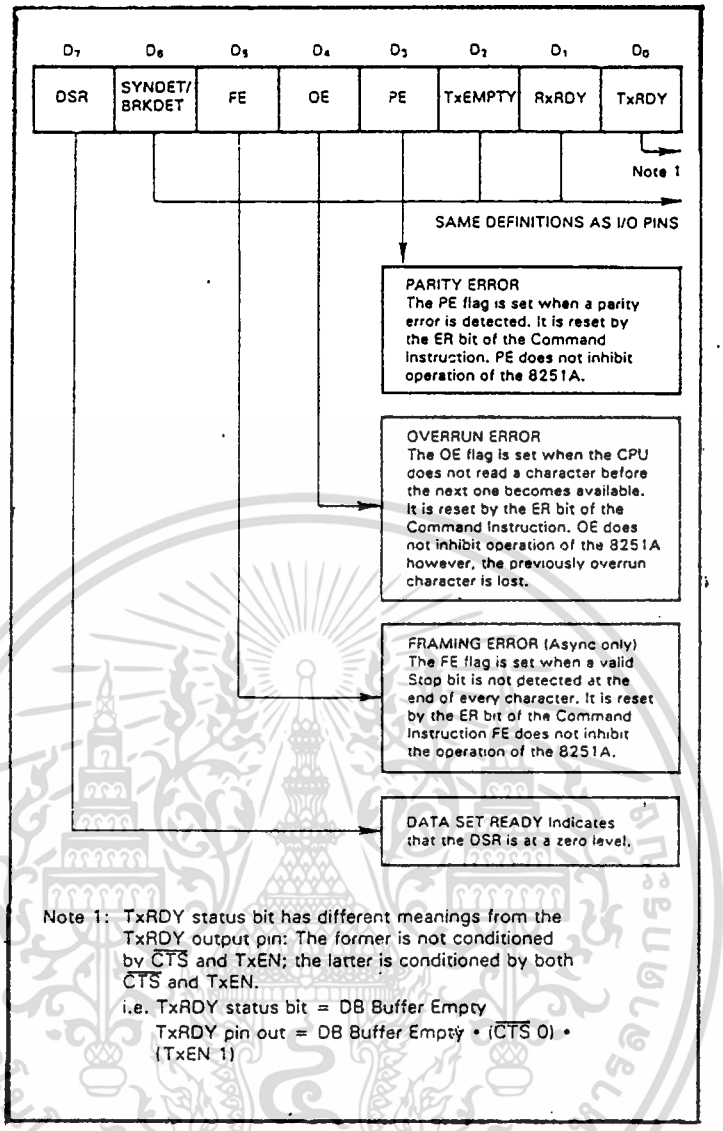
จาก Block diagram ของ 8251 จะเห็นว่าประกอบไปด้วยช่องรับ-ส่งข้อมูลอย่างละช่อง โดยแต่ละช่องจะมี buffer ไว้สำหรับพักข้อมูล และแต่ละช่องจะมีหน่วยควบคุม (Control Unit) ไว้คอยควบคุมอีกทีนึง buffer ของแต่ละช่องรับ-ส่งข้อมูลก็จะติดต่อกับ buffer ของข้อมูลที่ใช้รับ-ส่งกับ CPU อีกทีนึงโดยมีหน่วยตรรกะควบคุมการอ่านเขียน (read/write control logic) เป็นตัวควบคุม

การโปรแกรม 8251

Mode การทำงานของ 8251 สามารถควบคุมโดยใช้ software ซึ่งต้องเขียนเอาไว้ก่อน เพื่อจัดลักษณะการทำงานไว้สำหรับการใช้งาน โดยมีขั้นตอนดังนี้

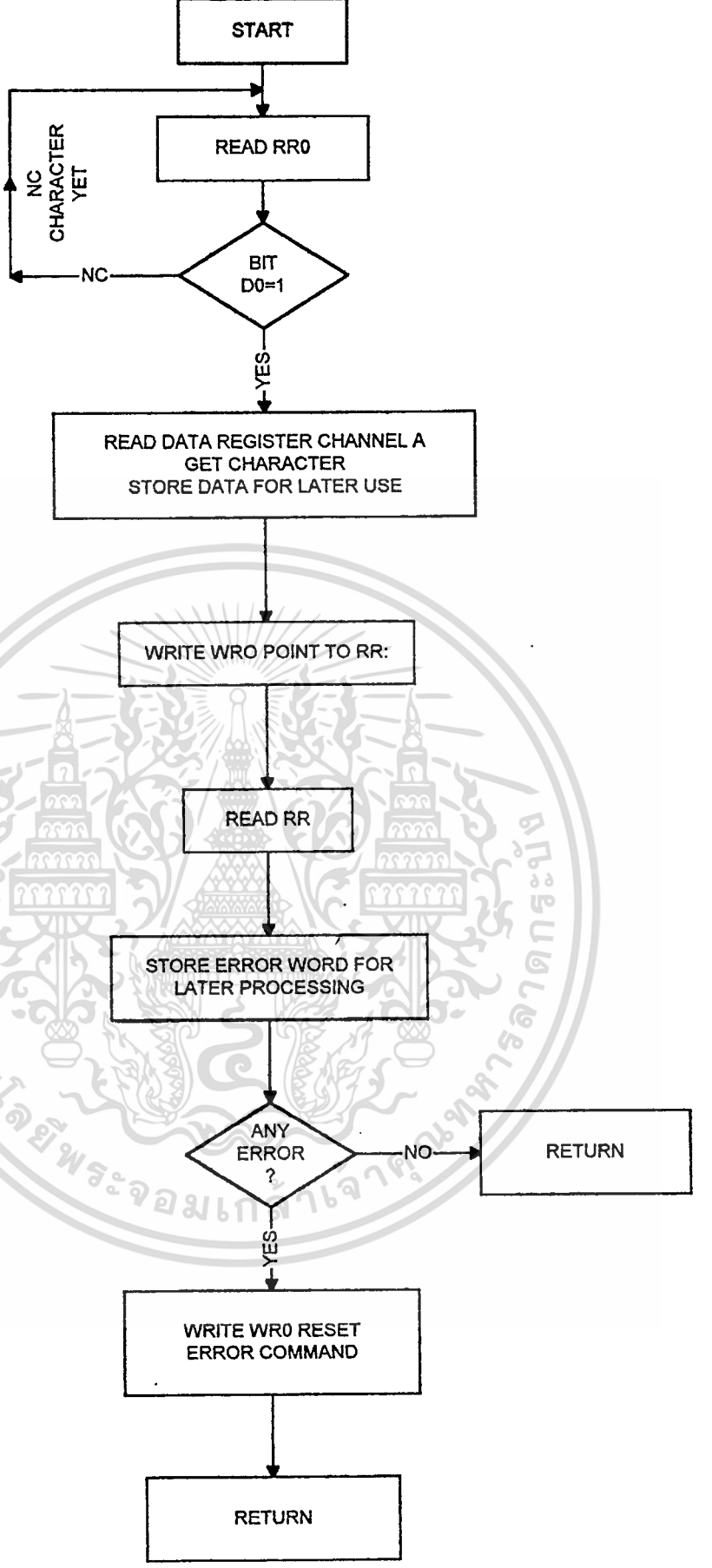
1. ให้สัญญาณ RESET เป็นการเริ่มต้นค่าใดๆก่อน
2. ทำคำสั่งเลือกขบวนการทำงาน (Mode Select)
3. ทำคำสั่ง Command

ในการส่งข้อมูล computer ต้องตรวจดู status register ของ 8251 เสียก่อนว่าพร้อมที่จะรับ-ส่งข้อมูลหรือไม่



รูปที่ 3.2 แสดงถึงการทำงานของ Status Register ของ 8251

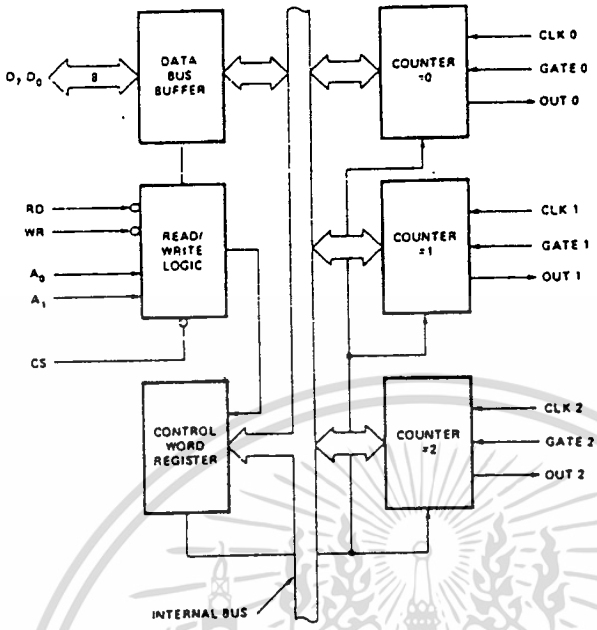
เราสามารถเขียน Block diagram ขั้นตอนการตรวจสอบ status register และการรับ-ส่งข้อมูลได้ดังรูปที่ 3.3



รูปที่ 3.3 แสดง Block diagram การรับ-ส่งข้อมูล

จาก Block Diagram ในรูปที่ 4.3 นั้นเราสามารถอธิบายได้ดังนี้คือ ในตอนแรกอ่านค่า RR0 เข้ามาแล้วทำการตรวจสอบดูว่าที่ bit D0 = 1 หรือไม่ ถ้าไม่ก็ให้กลับไปอ่านค่า RR0 เข้ามาใหม่ แต่ถ้า bit D0 = 1 ก็ให้อ่าน data register จาก channel A และเก็บข้อมูลต่างๆ ไว้เพื่อนำไปใช้ประโยชน์ต่อไป ต่อไปก็กำหนด ให้ WR0 ชี้ไปที่ RR อ่านค่า RR เก็บค่าที่ผิดไว้เพื่อการประมวลผลต่อไป ต่อจากนั้นก็ตรวจดูว่ามีค่าผิดหรือไม่ ถ้าไม่มีก็จบการทำงาน แต่ถ้ามีก็ให้เขียนคำสั่งที่ชื่อว่า WR0 reset error command เพื่อทำการล้าง WR0 และเริ่มค่าใหม่ เสร็จการทำงาน

หลักการทำงานทั่วไปของ 8253 ที่ใช้ในวงจร



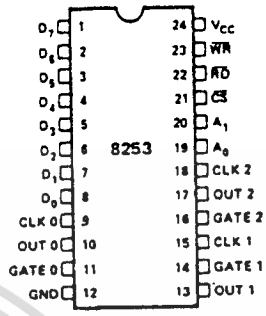


Figure 2. Pin Configuration

รูป 3.4 Block diagram แสดงการทำงานของ 8253

ส่วนของบัฟเฟอร์ทางเดินข้อมูล (Data Bus Buffer) เป็นที่พักข้อมูลเข้าและออกระหว่าง Microcomputer กับ register ของ 8253 ส่วนลอจิกการอ่านและเขียน (Read/Write Logic) ใช้สำหรับควบคุมการอ่านและเขียนของ register ของตัวนับเวลา และส่วนสุดท้ายที่จะกล่าวถึงเป็นส่วนที่เก็บข้อความที่ถูกโปรแกรมไว้ (Control Word Register)

รายละเอียดต่างๆของ register ภายใน 8253

1. รีจีสเตอร์ควบคุมข้อความ (Control Word Register) ใช้ควบคุมการทำงานแบบต่างและเลือกใช้วิธีการนับเวลาว่าจะให้นับแบบเลขฐานสองหรือแบบ BCD (Binary Code Decimal) ก่อนจะใช้งานต้องโปรแกรมข้อมูลให้กับรีจีสเตอร์นี้เสียก่อน เพื่อกำหนดการทำงานของตัวนับเวลา

2. ตัวนับเวลา 0 (ชุดที่ 1), ตัวนับเวลา 1 (ชุดที่ 2), และตัวนับเวลา 2 (ชุดที่ 3) แต่ละตัวนับเวลามี 16บิท ทำหน้าที่นับลง ซึ่งสามารถนับได้แบบเลขฐานสองหรือแบบ BCD ก็ได้ โดยที่การอ่านข้อมูลภายในตัวนับเวลา การเลือกชุดนับเวลาทำได้โดยการกำหนดการเลือกไว้ในบิทที่ 7 และ 8 ของรีจีสเตอร์ควบคุมข้อความ

ลักษณะการทำงานของบิทต่าง ๆในรีจีสเตอร์ mode control

บิท	D7	D6	D5	D4	D3	D2	D1	D0
หน้าที่	SC1	SC0	RL1	RL0	M2	M1	M0	BCD

(3.5 ก)

SC1	SC0	แชนแนลที่ถูกเลือก
0	0	0
0	1	1
1	0	2
1	1	-

(3.3ข)

RL1	RL0	
0	0	ทำการแลทช์ค่าในรีจิสเตอร์-เคาน์เตอร์
0	1	อ่าน/เขียนเฉพาะข้อมูลเฉพาะใน 8 บิทล่าง(Least-significant Bit)
1	0	อ่าน/เขียนเฉพาะข้อมูลเฉพาะใน 8 บิทบน(Most-significant Bit)
1	1	อ่าน/เขียนเฉพาะข้อมูลทั้ง 16 บิท โดยเริ่มจาก 8 บิทล่างก่อน จากนั้นจึงอ่าน/เขียนข้อมูลใน 8 บิทบน

(3.3ค)

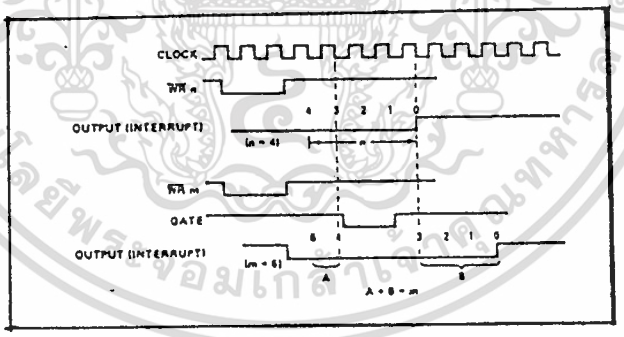
M2	M1	M0	โหมดการทำงาน
0	0	0	โหมด 0 : Interrupt on Terminal Count
0	0	1	โหมด 1 : Programmable One-Shot
0	1	0	โหมด 2 : Rate Generator
0	1	1	โหมด 3 : Square Wave Generator
1	0	0	โหมด 4 : Software Triggered Strobe
1	0	1	โหมด 5 : Hardware Triggered Strobe

(3.3ง)

รูปที่ 3.3 (ก-ง) แสดงถึงการทำงานของบิทต่างๆของรีจีสเตอร์

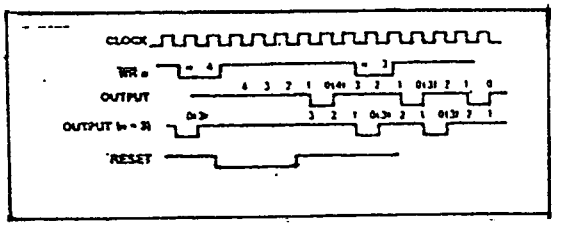
ในวงจรตัวกลางมีดี้เลือกใช้เฉพาะ โหมด 0 และ โหมด 2 ซึ่งมีรายละเอียดดังนี้

1. การทำงานของโหมด 0 ตัวนับเวลาจะเริ่มนับเวลาลงเมื่อขาออก (Out) มีค่าลอจิกเป็น 1 ค่าเริ่มต้นการนับและรูปแบบการนับ (แบบเลขฐานสองหรือแบบ BCD) จะถูกโปรแกรมไว้ก่อน ตัวนับเวลาจะนับด้วยความถี่ของสัญญาณนาฬิกาที่ป้อนเข้ามา เมื่อนับลงจนถึงค่า แล้ว ขาออกจะมีลอจิกเป็น 1 อีก ซึ่งค่าเริ่มต้นอาจใช้ค่าเดิมหรือเปลี่ยนค่าใหม่ก็ได้



รูปที่ 3.4 Timing Diagram ของ Mode 0

2. การทำงานของโหมด 2 นี้ 8253 จะถูกใช้เป็นตัวนับเวลาที่ทำหน้าที่ให้ค่าลอจิกเป็น 0 และ 1 สลับกันไปด้วยอัตราความถี่เท่ากับสัญญาณนาฬิกาที่เข้ามา หารด้วยค่าๆหนึ่งซึ่งทำให้ความกว้างของลอจิก 0 มีค่าเท่ากับ 1 คาบของสัญญาณนาฬิกาที่เข้ามา ส่วนความกว้างของลอจิก 1 มีค่าเท่ากับคาบของสัญญาณนาฬิกาที่เข้ามาคูณด้วยเลขจำนวนนั้น



รูปที่ 3.5 Timing Diagram ของ Mode 2

การกำหนดรายละเอียดต่าง ๆในวงจร

สำหรับเครื่อง Microcomputer ชนิด IBM Compatible มักจะใช้ข้อมูลขนาด 9 บิทในการกำหนดแอดเดรส (Address) ของอุปกรณ์อินพุท-เอาพุท ซึ่งในโครงงานนี้เราจะอ้างอิงตามนั้นเช่นกัน

8251 จะประกอบไปด้วย พอร์ทควบคุม (Control Port) และ พอร์ทข้อมูล (Data Port) โดยกำหนด

Control Port I กำหนด address ไว้ที่ตำแหน่ง 2B1H

Data Port I กำหนด address ไว้ที่ตำแหน่ง 2B0H

8253 จะประกอบไปด้วย พอร์ทตัวนับ (Counter Port) และ พอร์ทควบคุม (Control Port) โดยกำหนดไว้ดังนี้

Counter Port 0,1,2 I กำหนด address ไว้ที่ตำแหน่ง 40H, 41H,และ 42H ตามลำดับ

Control Port I กำหนด address ไว้ที่ตำแหน่ง 43H

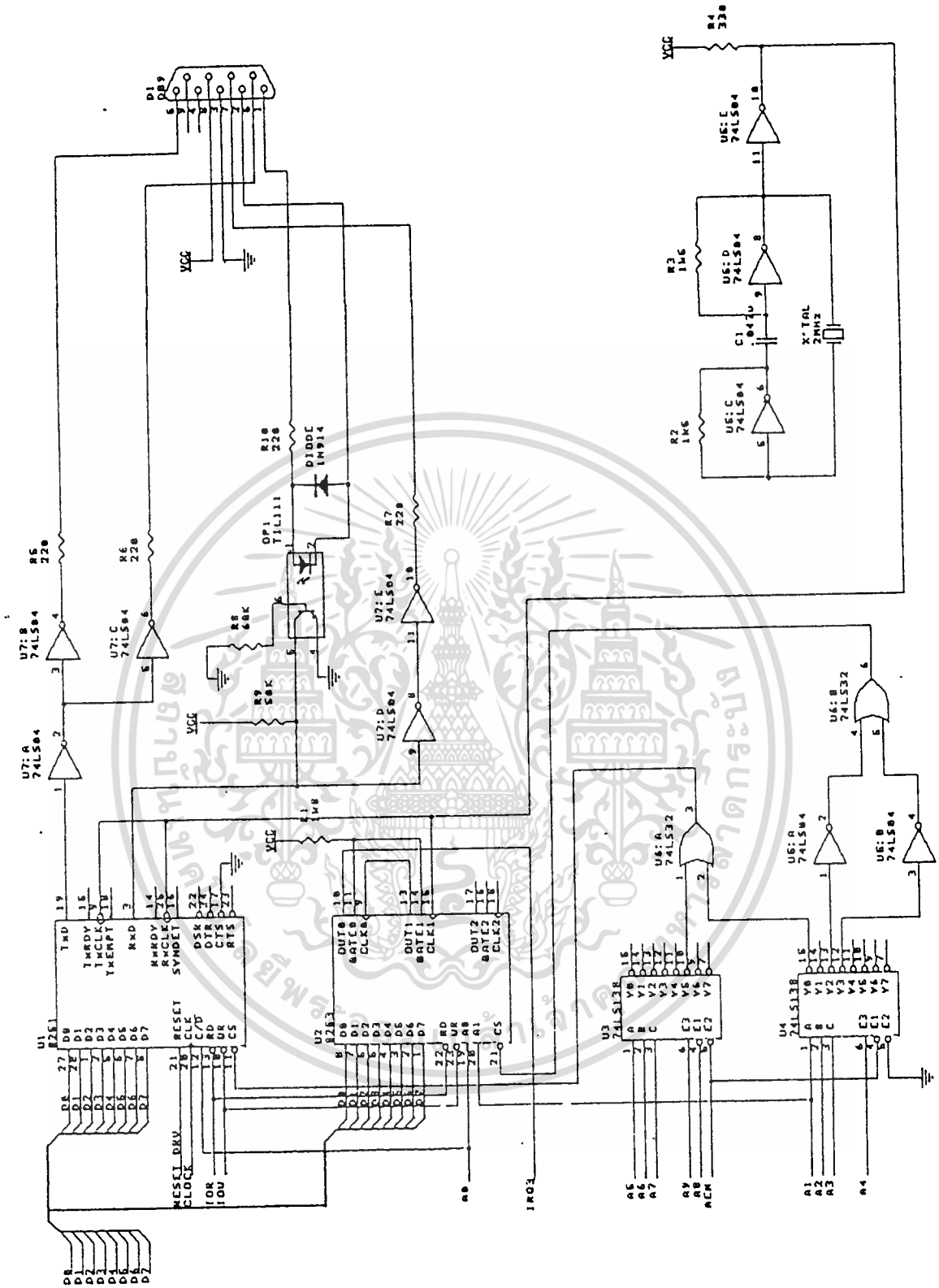
สำหรับตัวรับข้อมูล (RxD) ของ 8251 จะทำการเชื่อมกับทางเดินข้อมูล โดยผ่าน Optoisolater TIL111 สัญญาณมีดี้ที่เข้ามา จะถูกป้องกันโดยไดโอด IN914 และถูก buffer โดยอินเวอร์เตอร์ (Inverter)

และสำหรับตัวส่งข้อมูล (TxD) จะส่งสัญญาณผ่านอินเวอร์เตอร์ (Inverter) โดยมี Inverter เป็น buffer ให้ ข้อมูลที่ผ่าน Inverter เหล่านี้จะถูกส่งเป็นแบบอนุกรมไปเข้าตัวต้านทานขนาด 220 โอห์ม เพื่อทำการเข้าคู่อย่างอิสระกับทางเดินตัวกลางมีดี้

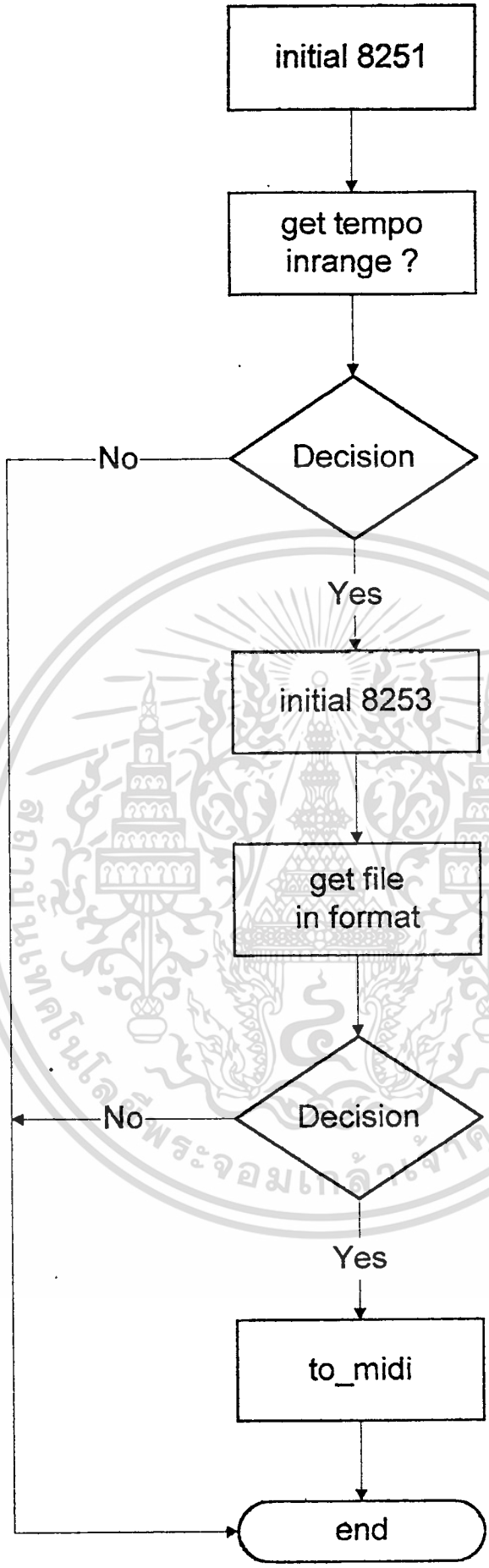
ตัวนับเวลาแต่ละชุดของ 8253จะต้องมีลอจิก 1 ไปกระตุ้น (Trig) เพื่อเริ่มการทำงาน โดย counter 0 จะได้รับสัญญาณนาฬิกาจากวงจรกำเนิดความถี่ (Crystal Oscillator) ขนาด 2 MHz แล้ว

ทำการหารความถี่ลงไปเป็นระดับ 24 จังหวะต่อควอเตอร์โน้ต เพื่อใช้เป็นสัญญาณนาฬิกาสำหรับตัว counter 1 ตัว counter 1 จะเป็นตัวเชื่อมระหว่างsoftwareที่ใช้ในการบันทึกเพลงและการทำงานของตัวกลางมีดี้ ให้สามารถสอดคล้องเล่นเป็นเพลงได้อย่างสมบูรณ์

ตัว Interface card แสดงไว้ในรูปที่ 3.6 และรูปที่ 3.7 แสดง Block diagram ขั้นตอนการทำงานของ Program ของ Interface card



รูปที่ 3.6 แสดงถึงวงจร Interface card



รูปที่ 3.7 Block diagram แสดงการ Program รับส่งข้อมูล

บทที่ 4
การทดลองและผลการทดลอง

การเล่น

การที่จะส่งข้อมูลให้ถูกต้องตามเวลาที่กำหนดจำเป็นต้องมีจังหวะ (Timer) เอาไว้อ้างอิงซึ่งเป็นฮาร์ดแวร์ สำหรับการทดลองนี้จะใช้ 8253 ซึ่งเป็น Programmable Interval Timer ซึ่งมีอยู่แล้วใน IBM-PC โดยใช้สัญญาณคล๊อคจากแชนแนล 0

ใน IBM-PC คล๊อกที่ป้อนให้ 8253 จะใช้คล๊อกของระบบหารด้วย 4 นั่นคือ 4.772727/4 = 1.1931817 MHz

การทำงานของ SEQ.EXE ที่แสดงใน flow chart ในรูปที่ 4.1 สามารถอธิบายได้ต่อไปนี้ SEQ.EXE เป็นโปรแกรมใช้ในการเขียนแบบ step time โดยการแสดงผลแบบ Tablature ของ keyboard ซึ่งประกอบด้วย module ต่างๆ ดังนี้

Insnote.c เป็น module หลักใช้ในการใส่ตัวโน้ตในห้องเพลง

Desk.c เป็น module ซึ่งใช้ในการวาด Desktop และเทียบตำแหน่งของโน้ต

Mouse.c เป็น module ซึ่งรวบรวม function เกี่ยวกับ mouse และ cursor ต่างๆ

Testlink.c เป็น module ของ function ในการจัดการเกี่ยวกับข้อมูล Linklist การ save และ load ข้อมูลจาก disk

Paint.c ใช้ในการ load ข้อมูลจาก Linklist แล้วแสดงผลในหน้าจอซึ่งจะแสดงตำแหน่งของโน้ตและ doration

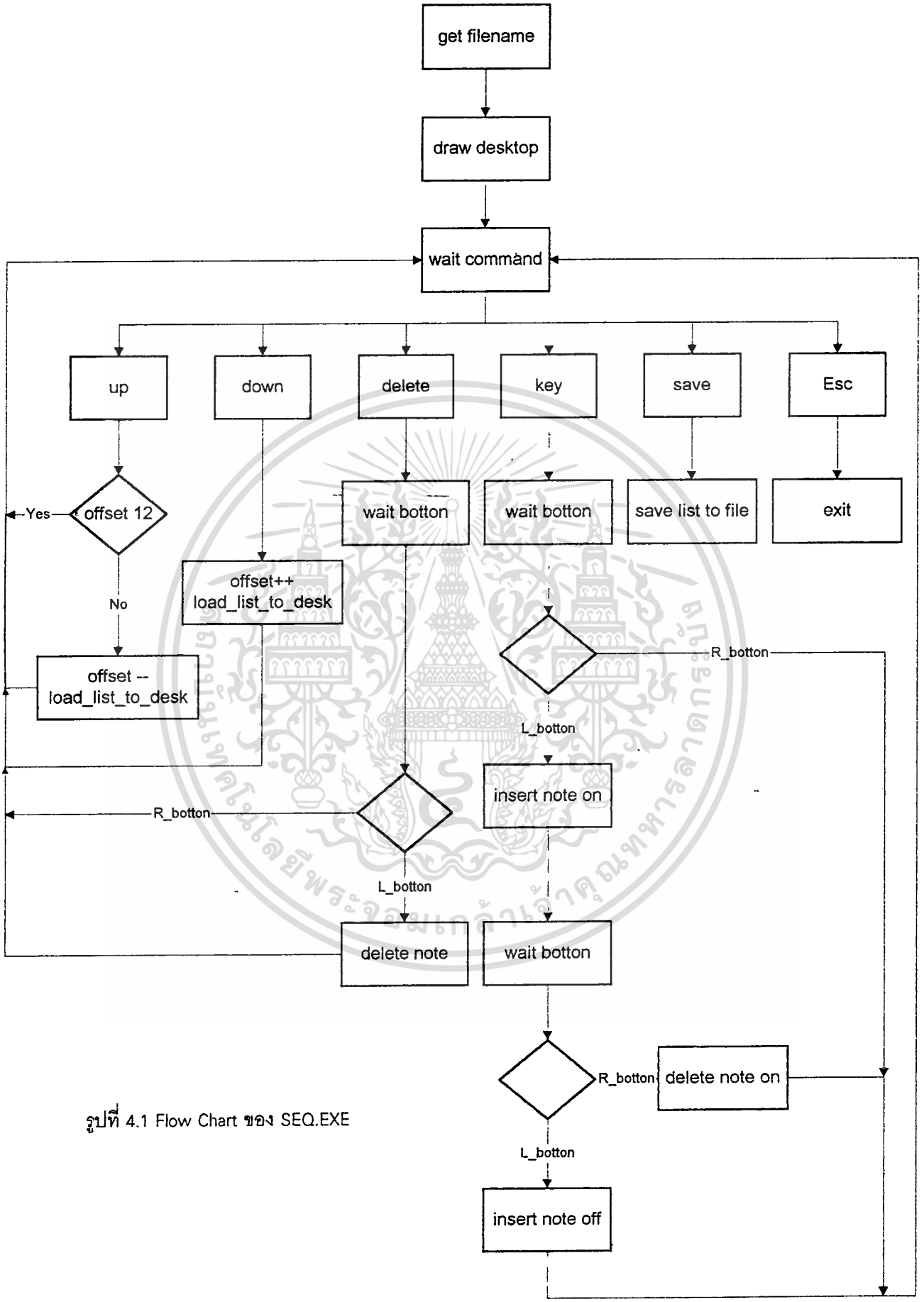
PLAY.EXE เป็นโปรแกรมใช้ในการเล่นเพลงที่มีการบันทึกไว้ส่งผ่านไปยังเครื่องดนตรี ประกอบด้วย 2 module ซึ่งแสดงในรูป 4.2

Systimer เป็น module หลักประกอบด้วย function ในการ set 8251, set system 8253 tame โดย

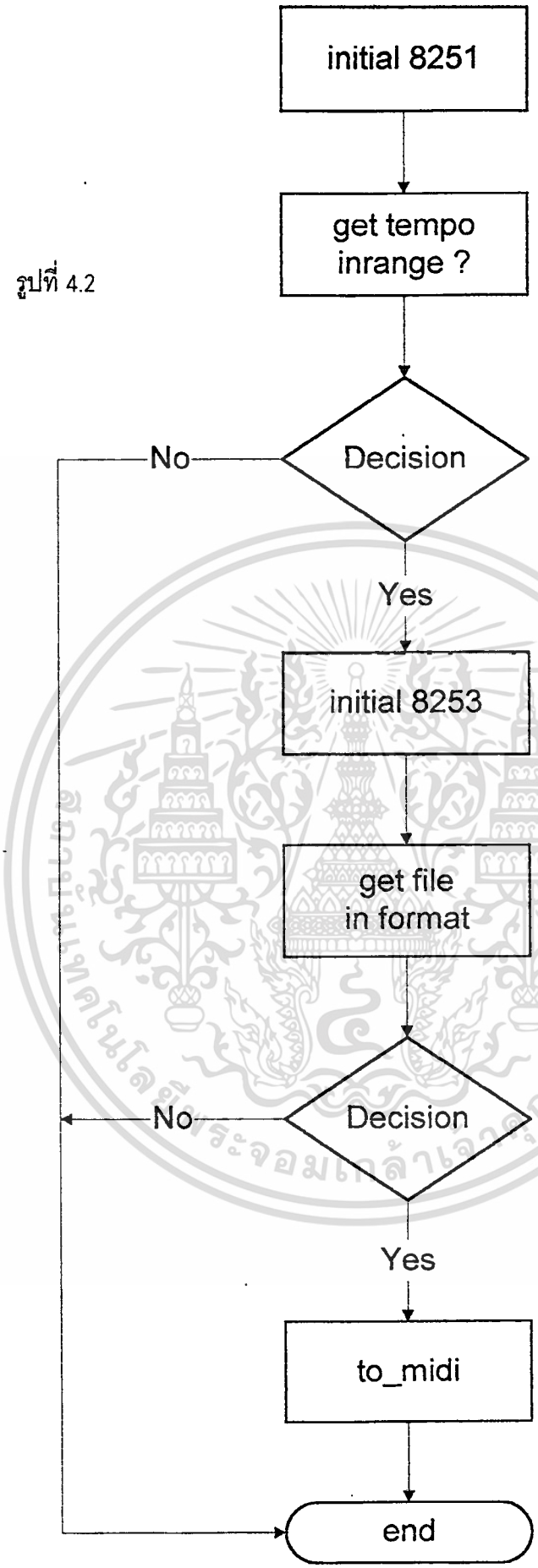
รับค่า tempo จาก user, load file และ tomode คือการส่งข้อมูลไปยังเครื่องดนตรีตามค่าเวลาของแต่ละตัวโน้ต

TL.EXE เป็น module ซึ่งจัดการเกี่ยวกับ Fole, Linklist และการ serch list

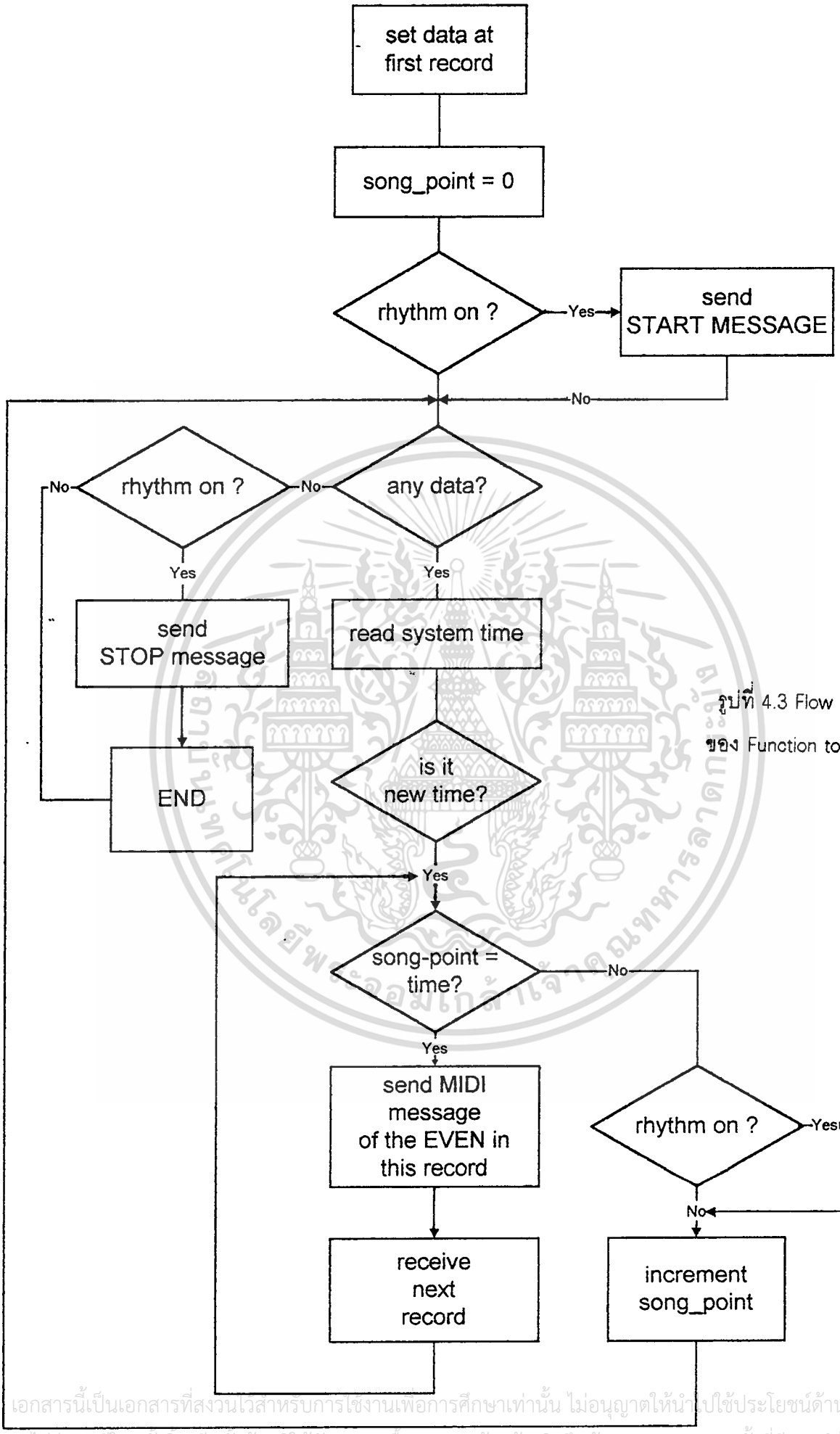
Flow chart แสดงส่วนของ Function to_midi อยู่ในรูป 4.3



รูปที่ 4.1 Flow Chart ของ SEQ.EXE



รูปที่ 4.2



รูปที่ 4.3 Flow Chart ของ Function to_midi

บทที่ 5
สรุป

การทดลองนี้เป็นการทดลองในส่วนของตัวกลางมิดี้ (MIDI Interface) ซึ่งเป็นการทดลองการส่งข้อมูลผ่าน Interface Card ซึ่งในการใช้งานของตัวระบบมิดี้ยังไม่สมบูรณ์

ในส่วนของตัวกลาง อาจปรับปรุงในส่วนของ UART โดยการใช้เบอร์อื่นๆ เช่น 8250 , MC6850 หรือZ80-SIO ซึ่งมีอัตราการรับส่งข้อมูลดีขึ้น โดย 8250 นั้นจะมีข้อดีในแง่ของการ Interface กับ IBM-PCเนื่องจากมีการออกแบบที่ Compatibleกัน (เพราะเป็นของบริษัท Intel) และมีขา Interrupt ซึ่งทำให้สามารถเขียนโปรแกรม สนับสนุนในการตรวจสอบการรับ-ส่งข้อมูลซึ่งทำให้การบันทึก (เขียน) ข้อมูลจากการเล่น (Real-Time) จริงได้ โดยผ่านทาง ชิป 8259 ซึ่งเป็นตัวจัดการเกี่ยวกับการ Interrupt ของ IBM-PC โดยไม่ต้องคอยให้ตัวโปรแกรมคอยตรวจสอบ แต่ให้ทำงานตามสัญญาณ Interrupt เมื่อมีข้อมูล

แนวทางในการพัฒนาต่อไปคือการเขียนโปรแกรมในส่วนของซีเควนเซอร์ เพื่อใช้ในการเขียนเพลงแก้ไขเพลง และในส่วนอื่นๆ เช่น ดรัมแมทชีน(drum machine) ซึ่งเป็นตัวให้จังหวะของเพลง โดยสามารถปรับแต่งข้อมูลโดยการใช้ Tool ที่สร้างขึ้นซึ่งทำให้ข้อมูลดีขึ้นตามต้องการได้

ภาคผนวก

```
//
//                SYSTIMER.C
#include <stdio.h>
#include <conio.h>
#include <time.h>
#include <dos.h>

#include "midi.h"

int tempo, _rhythm, song_point, loop, def_ch;
long last_tick, tick;
extern char midi_file[15];
node *_info;
clock_t clk;

void set_8253(int speed), init8251(void), cursor
(int row, int col);
void interrupt (*oldint)(void);
void return_vect(void);
void disp(void), send_out(), to_midi();

void main()
{
     init8251();
     printf("Enter tempo ");
     scanf("%d",&tempo);
     set_8253(tempo);
     printf("enter filename ");
     scanf("%15s",midi_file);
     load(midi_file);
     printf("press any key to begin");
     _rhythm=1;
     getch();
     to_midi();
     clearnode();
     return_vect();
}

void set_8253(int speed)
{
     unsigned  time;
     time = (1193180*60/(speed*24));
     outportb(0x43, 0x36);
     outportb(0x40, time % 256);
```

```
    outportb(0x40, time / 256);
}

void return_vect(void)
{
    outportb(0x40, 65536 % 256);
    outportb(0x40, 65536 / 256);
}

void to_midi()
{
     _info=headnode;
     song_point = last_tick = 0;
     if(_rhythm)
      START;
     while(_info){
      do{
           clk=clock();
      }while(clk==last_tick);
      last_tick=clk;
     POLL:if(song_point==_info->time){
           send_out(_info);
           _info=_info->next;
           goto POLL;
      }
      if(_rhythm)
           CLOCK;
      song_point++;
     }
}

void send_out(node *mass)
{
     check; outportb(data_port, mass->command);
     check; outportb(data_port, mass->note);
     check; outportb(data_port, mass->data);
}

void _note_on(unsigned note, unsigned velocity,
unsigned channel)
{
     check; outportb(data_port, note_on+channel);
     check; outportb(data_port, note);
     check; outportb(data_port, velocity);
```

```
}

void _all_note_off(int channel)
{
     check; outportb(data_port,
all_note_off+channel);
     check; outportb(data_port, 0);
}

void _stop(void)
{
     int i;
     if(_rhythm)
      STOP;
     for(i=0;i<15;i++)
      _all_note_off(i);
}

void hide_cursor(void)
{
  cursor(26,1);
}

void cursor(int row, int col)
{
     union REGS ireg;

     ireg.h.ah = 0x02;
     ireg.h.bh = getvpage();
     ireg.h.dh = row - 1;
     ireg.h.dl = col - 1;

     int86(0x10, &ireg, &ireg);
}

int getvpage(void)
{
   union REGS ireg;

   ireg.h.ah = 0x0f;                    //
Function 0x0f

   int86(0x10, &ireg, &ireg);
   return (ireg.h.bh);
```

```
}

void init8251(void)
{
     outportb(control_port, 0x4f);// async 64
xBdrate factor
     outportb(control_port, 0x15);// 8 data bit 2
stop bit
     outportb(control_port, 0x55);// command word
To reset 8251
     outportb(control_port, 0x4f);// async 64
xBdrate factor
     outportb(control_port, 0x15);
}//
//                TL.C
#include <stdio.h>
#include <malloc.h>
#include <conio.h>

typedef struct singlelist{
          unsigned int time;
          unsigned int command;
          unsigned int note;
          unsigned int data;
          struct singlelist  *prior;
          struct singlelist  *next;
     }node;
typedef struct list{
          unsigned int time;
          unsigned int command;
          unsigned int note;
          unsigned int data;
     }dim;
extern struct singlelist  *headnode, *tailnode;
struct singlelist  *walknode , *newnode, *
nextnode;

void deletenode(node *before);
node *search(node *ser);
node *new_node(void);
void clearnode(void);
void ins_dat(void);
void disp_dat(void);
void save(char file[]);
```

```
void load(char file[]);
char midi_file[15];

node *new_node(void){

      return(node *)malloc(sizeof(node));
}


void delnode(node *p){

      free((void *)p);
}

void clearnode(void){

      walknode = headnode;
      newnode = walknode->next;
      while(walknode != NULL){
       free(walknode);
       walknode = newnode;
       newnode = newnode->next;
      }
}

void deletenode(node *before){

      node *after;
      if(before->next != NULL){
       after = before->next->next;
       delnode(before->next);
       before->next = after;
       after->prior = before;
      }
}

node *search(node *ser){

      node *tempnode, *testnode;
      testnode = headnode;
      while((testnode->time < ser->time) &&
(testnode != NULL))
       testnode=testnode->next;
      return testnode;
```

```
}

void disp_dat(void){

        walknode = headnode;
        do{
          printf("  Time=%u  ",walknode->time);
          walknode = walknode->next;
        }while(walknode != NULL);
}

void save(char file[]){

        register int t;
        node *info;
        FILE *fp;

        if((fp=fopen(file, "wb")) == NULL){
          printf("Cannot open file\n");
          exit(1);
        }
        printf("\nsaving file");

        info = headnode;
        while(info){
          fwrite(info, sizeof(node), 1, fp);
          info = info->next;
        }
        fclose(fp);
}

void load(char file[]){

        register int t, i;
        node *info, *temp=NULL;
        FILE *fp;
        if((fp=fopen(file, "rb")) == NULL){
          printf("Cannot open file\n");
          exit(1);
        }
        clearnode();

        while(headnode){          //clear all list
          info = headnode->next;
```

```
            free(info);
            headnode=info;
          }

          printf("\nloading file\n");

          headnode=(node *)malloc(sizeof(node));
          if(!headnode){
            printf("Out of memory\n");
            return;
          }
          info=headnode;
          while(!feof(fp)){
            if(1 != fread(info, sizeof(node), 1, fp))
              break;
            info->next = (node *)malloc(sizeof(node))
;
            if(!info->next){
              printf("Out of memory\n");
              return;
            }
            info->prior=temp;
            temp=info;
            info=info->next;
          }
          temp->next=NULL;
          tailnode=temp;

          headnode->prior=NULL;
          fclose(fp);
}

void in_list(unsigned time_o, unsigned com_m,
unsigned data_1,
             unsigned data_2)
{

          if(headnode==NULL){
            headnode = new_node();
            newnode = headnode;

            newnode->time=time_o;
            newnode->command=com_m;
            newnode->note=data_1;
```

```
        newnode->data=data_2;

        newnode->next=NULL;
        newnode->prior=NULL;
        tailnode=newnode;
      }
      else{
        newnode = new_node();

        newnode->time=time_o;
        newnode->command=com_m;
        newnode->note=data_1;
        newnode->data=data_2;

        walknode = search(newnode);
        if(walknode!=NULL){
         if(walknode->prior==NULL)
             headnode=newnode;
         else
             walknode->prior->next = newnode;
         newnode->prior = walknode->prior;
         walknode->prior = newnode;
         newnode->next = walknode;
        }
        else{
         newnode->next = NULL;
         newnode->prior=tailnode;
         tailnode->next=newnode;
         tailnode=newnode;
        }
      }
      gotoxy(1,1);
      printf("%5u %3u %5u",newnode->time,newnode-
>command, newnode->note);
      hide_cursor();
}

void del_list(unsigned time_o, unsigned com_m,
unsigned data_1,
          unsigned data_2)
{
      walknode = headnode;
      while((walknode->time!=time_o)&&(walknode->
command!=com_m)
```

```
          &&(walknode->note!=data_1)&&(walknode->
data!=data_2)){

         walknode=walknode->next;
        }
        walknode->prior->next=walknode->next;
        walknode->next->prior=walknode->prior;
        free(walknode);

}□
```

```
//
//                    INSNOTE.C
#define MAIN 1
#include <stdio.h>
#include <conio.h>
#include "win.h"
#include "def.h"

unsigned note, vel, offset, channel;
int row, col, note_pos;
unsigned int key;
int button;
char *let[6];
int n=0;
node *begin, *end, *cur;
extern char midi_file[15];

void select_note(int col, char *let[]);
void ins_note(void);
void wait_on(int button);
void p_note(),print_off();
void edit_ins(void);

void main()
{
     channel=0;
     vel=64;
     printf("enter file to work\n");
     scanf("%15s",midi_file);
     headnode = tailnode = NULL;
     headnode->prior= NULL;
     headnode->next = NULL;              //
     edit_ins();
     clearnode();
}

void select_note(int col2, char *let2[])
{
     setb;
     gotoxy(1, 25);
     cprintf("select note from keybord");
     hide_cursor();
     note=0;
```

```
      note_pos = col2;
      note = comp_table(col2, let2);
      gotoxy(75, 25);
      setb;
      cprintf(*let2);
      hide_cursor();

      if(note!=0){
       mouse_hrange((note_pos-1)*8, (note_pos-1)
*8+7 );
       mouse_vrange(7*8, 22*8);
      }
}

void ins_note(void)
{
      int a, d, m, h, i, v, x, y, tes, temp,
temp1, row_e, col_e;
      char *b;
      unsigned time_on, time_off, on_ch, off_ch,
on_set, of_set;
      setb;
      gotoxy(1, 25);
      cprintf("Insert note in measure ");
      hide_cursor();
      setd;
      tes=0;
      if(note==0)
       return;
      do{


       if(mouse_press(&temp, &temp, LBUTTON)==
LBUTTON){
                                    read_mouse(&row_e,
&col_e, &button);
            wait_on(LBUTTON);
            x = col_e;
            y = row_e;
            gotoxy(x, y);
            cprintf("▒");
            on_set=offset;
            of_set=on_set;
```

```
            time_on=48*(on_set-1)+3*(y-8); //
insert begin note list
            on_ch=note_on+channel;
            off_ch=note_off+channel;
            in_list(time_on, on_ch, note, vel);
            hide_cursor();
            i=y;
            mouse_cursor_off();
            do{
             mouse_counters(&v, &h);
             if(v >= sense){
              setd;
              gotoxy(x, i);
              cprintf("▒");
              hide_cursor();
              if(i<23)
                   i++;
              else{
                   of_set++;
                   offset=of_set;
                   print_off();
//                 load_list_to_desk(offset);
                   i = f_row;
              }
             }

             if(v <= -sense){
              setd;
              gotoxy(x, i);
              cprintf("_");
              hide_cursor();
              if(of_set==on_set){
                   if(i>(y+1))
                     i--;
              }
              else{
                   if(i>f_row)
                     i--;
                   else{
                     of_set--;
                     offset=of_set;
                     print_off();
//                   setd;
```

```
//                      draw_measure();
//                      load_list_to_desk(offset);
                        if(of_set==on_set)
                          d=y;
                        else
                          d=f_row;
                        for(a=d;a<=l_row;a++){
                          setd;
                          gotoxy(x, a);
                          cprintf("▒");
                        }
                        hide_cursor();
//                      load_list_to_desk(offset);
                        i=l_row;
                      }
                    }
                  }

                  if(mouse_press(&temp, &temp,
LBUTTON) == LBUTTON){
                    wait_on(LBUTTON);
                    temp1 = 1;
                    mouse_cursor_on();
                    time_off=48*(of_set-1)+3*(i-8);

                    in_list(time_off, off_ch, note,
vel);
                  }
                  if(mouse_press(&temp, &temp,
RBUTTON) == RBUTTON){
                    wait_on(RBUTTON);
                    temp1 = 2;
                    mouse_cursor_on();
                    del_list(time_on, on_ch, note, vel)
;
                  }
                }while((temp1!=1)&&(temp1!=2));
                if(temp1==2){
                  offset=on_set;
                  print_off();
                  setd;
                  for(i=y; i<=23;i++){
                    gotoxy(x,i);
                    cprintf("_");
```

```
                }
//              load_list_to_desk(on_set);
              }
              mouse_hrange(0, 640);
              mouse_vrange(0, 200);
              tes=1;
        }
      }while(tes!=1);
      mouse_hrange(0, 640);
      mouse_vrange(0, 200);
}

void wait_on(int button)
{
      int bit;
      if(button == LBUTTON)
        while(mouse_press(&bit, &bit, LBUTTON));
      else
        while(mouse_press(&bit, &bit, RBUTTON));
}

void p_note()
{
      setb;
      note=0;
      select_note(col, let);
      ins_note();
      setb;
      gotoxy(75,25);
      cprintf("   ");
      hide_cursor();
      note = 0;
}

void edit_ins(void)
{
      int  row1, col1, temp;
      unsigned char c;
      clrscr();
      testmouse();
      mouse_here = mouse_status(&button);
      offset=1;
      c_scroll(2, 1, 80, 23, 0, 7, 16);
      setd;
```

```
        draw_desk();
        c_scroll(25, 1, 80, 1, 1, 7, 48);
        setb;
        gotoxy(70, 25);
        cprintf("Note ");
        gotoxy(35, 25);
        cprintf("offset ");
        gotoxy(42, 25);
        cprintf("%4u",offset);
        if(mouse_here)
         mouse_cursor_on();
        hide_cursor();

        do{
         do{
             if(kbhit())
             c=getch();
             if(c==27)
             break;
         }while(mouse_press(&temp, &temp, LBUTTON)!=
LBUTTON);
        wait_on(LBUTTON);
        read_mouse(&row, &col, &button);
        if(up){
             if(offset!=1){
              offset--;
              print_off();
              setd;
              draw_measure();
//            load_list_to_desk(offset);
             }
        }
        if(down){
             offset++;
             print_off();
             setd;
             draw_measure();
//           load_list_to_desk(offset);
        }
        if(ins){
             p_note();
        }
        if(menu){
             gotoxy(1, 25);
```

```
                setb;
                cprintf("saving                ");
                save(midi_file);
                cprintf("                      ");
                hide_cursor();
          }

        }while(c!=27);
        mouse_hrange(0, 640);
        mouse_vrange(0, 200);
        mouse_cursor_off();
}

void print_off()
{
        setb;
        gotoxy(42, 25);
        cprintf("%4d",offset);
        hide_cursor();
}
//
//                DESK.C
#include <stdio.h>
#include <stdlib.h>
#include <dos.h>
#include <conio.h>

void init_vmode();
void draw_block(int x, int y, int num);
void draw_box( int row, int col, int wide, int
high );
void draw_measure(void);
void scroll(int row, int col, int wide, int deep,
int num, int function);
void save_video(int startx,int endx,int
starty,int endy,unsigned char *buf_ptr);
void restore_video();
void draw_desk(void);

int vmode;
char far *vidmem;
/*
void main()
{
```

```
        unsigned  char *pt;
        init_vmode();
        clrscr();
        draw_desk();
        pt = (unsigned char *)malloc((70)*(14));
        gotoxy(6,8);
        getch();
        save_video(5, 76, 7, 24, pt);
        scroll(8,6,69,14,0,6);
        getch();
        scroll(8,6,69,14,0,7);
        restore_video(5, 76, 7, 24, pt);
        gotoxy(6,8);
        getch();
        free(pt);
        clrscr();
} */

void draw_key(int x, int y)
{
        gotoxy(x,y);cprintf(" ▌▌ | ▌▌▌ | ");
y++;
        gotoxy(x,y);cprintf(" ▌▌ | ▌▌▌ | ");
         y++;
        gotoxy(x,y);cprintf(" | | | | | | ");
         y++;
        gotoxy(x,y);cprintf("└┴┴┴┴┴┴┘ ");
}

void draw_left(int x, int y)
{
        gotoxy(x, y);cprintf("│");y++;
        gotoxy(x, y);cprintf("│");y++;
        gotoxy(x, y);cprintf("│");y++;
        gotoxy(x, y);cprintf("└");
}

void draw_right(int x, int y)
{
        gotoxy(x, y);cprintf(" │");y++;
        gotoxy(x, y);cprintf(" │");y++;
        gotoxy(x, y);cprintf(" │");y++;
        gotoxy(x, y);cprintf("─┘");
}
```

```
void draw_block(int x, int y, int num)
{
      int i;
      textbackground(15);
      textcolor(0);
      draw_left(x, y);x++;
      for(i=0;i<num;i++){
       draw_key(x, y);
       x = x+14;
      }
      draw_right(x, y);
      textcolor(15);
      textbackground(9);
}

void draw_box(int row, int col, int wide, int
high)
{
      int i,x;
      for(x=0;x<3;x++){
       for(i=col+1;i<wide+1;i++){
            gotoxy(i,7+x*8);
            cprintf("_");
       }
      }
      gotoxy(col, row);
      cprintf("╒");
      gotoxy((col+wide),row);
      cprintf("╕");
      for(i=(row+1);i<(row+high);i++){
       gotoxy((col+wide),i);
       cprintf("│");
       gotoxy((col+wide)-3,i);
       cprintf("│");
      }

      for(i=(col+1);i<(col+wide);i++){
       gotoxy(i,row);
       cprintf("=");
      }
      gotoxy(col,(row+high));
      cprintf("└");
      for(i=(col+1);i<(col+wide);i++){
```

```
        gotoxy(i,(row+high));
        cprintf("─");
      }
      gotoxy((col+wide),(row+high));
      cprintf("┘");
      for(i=(row+1);i<(row+high);i++){
        gotoxy(col,i);
        cprintf("│");
        gotoxy(col+4,i);
        cprintf("│");
      }
      gotoxy(col+4,row+high);
      cprintf("┴");
      gotoxy((col+wide)-3,row+high);
      cprintf("┴");
      textcolor(1);
      textbackground(15);
      gotoxy(79, 7);
      cprintf("-");
      gotoxy(79, 23);
      cprintf("");
}

void draw_measure(void)
{
      int i;
      for(i=0;i<16;i++){
        gotoxy(6,i+8);
        cprintf("_____ _______ _____ _______ _____
_______ ");
        cprintf("_____ _______ _____ _______ _");
      }
      for(i=1; i<=2; i++){
        gotoxy(6,7+i*8);

cprintf("_____________________________________________
_");
        cprintf("_____________________________________");
      }
      hide_cursor();
}

void scroll(int row, int col, int wide, int deep,
int num, int function)
```

```
{
   union REGS ireg;

   ireg.h.ah = function;
   ireg.h.al = num;
   ireg.h.ch = row - 1;          /* Index from (0,
0) */
   ireg.h.cl = col - 1;
   ireg.h.dh = row + deep;
   ireg.h.dl = col + wide;
   ireg.h.bh = 0;                /* attribute byte
*/

   int86(0x10, &ireg, &ireg);
}

void save_video(int startx,int endx,int
starty,int endy,unsigned char *buf_ptr)
{
     register int i, j;
     char far *v, far *t;
     v = vidmem;
     for(i=starty; i<endy; i++)
      for(j=startx; j<endx; j++){
           t = v + (i*160) + (j*2);
           *buf_ptr++ = *t++;
           *buf_ptr++ = *t;
      //   *(t-1) = '';     /* delete character */
      }
}

void restore_video( startx, endx, starty, endy,
buf_ptr)
int  startx, endx, starty, endy;
unsigned char *buf_ptr;
{
     register int i,j;
     char far *v, far *t;
     v = vidmem;
     t = v;
     for(i=starty; i<endy; i++)
      for(j=startx; j<endx; j++){
           v = t;
           v += (i*160)+(j*2);
```

```
                *v++ = *buf_ptr++;
                *v = *buf_ptr++;
        }
}

video_mode()
{
     union REGS r;

     r.h.ah = 15;
     return int86(0x10, &r, &r) & 255;
}

void draw_desk(void)
{
     draw_box(2, 1, 79, 22);
     draw_block(5, 3, 5);
     draw_measure();
}

void init_vmode(void)
{
     vmode = video_mode();
     if((vmode!=2)&&(vmode!=3)&&(vmode!=7)){
      printf("video must be in 80 column text
mode");
       exit(1);
     }
     if(vmode==7) vidmem = (char far *) 0
xB0000000;
     else vidmem = (char far *) 0xB8000000;
}

int comp_table(int col, char *note[])  // for
convert column position to note value */
{
     int x;
     switch(col){
          case  6 :  x=36 ; *note = "2C" ; break;
          case  7 :  x=37 ; *note = "2C#"; break;
          case  8 :  x=38 ; *note = "2D" ; break;
          case  9 :  x=39 ; *note = "2D#"; break;
          case 10 :  x=40 ; *note = "2E" ; break;
          case 12 :  x=41 ; *note = "2F" ; break;
```

```
case 13 :  x=42 ; *note = "2F#"; break;
case 14 :  x=43 ; *note = "2G" ; break;
case 15 :  x=44 ; *note = "2G#"; break;
case 16 :  x=45 ; *note = "2A" ; break;
case 17 :  x=46 ; *note = "2A#"; break;
case 18 :  x=47 ; *note = "2B" ; break;
case 20 :  x=48 ; *note = "3C" ; break;
case 21 :  x=49 ; *note = "3C#"; break;
case 22 :  x=50 ; *note = "3D" ; break;
case 23 :  x=51 ; *note = "3D#"; break;
case 24 :  x=52 ; *note = "3E" ; break;
case 26 :  x=53 ; *note = "3F" ; break;
case 27 :  x=54 ; *note = "3F#"; break;
case 28 :  x=55 ; *note = "3G" ; break;
case 29 :  x=56 ; *note = "3G#"; break;
case 30 :  x=57 ; *note = "3A" ; break;
case 31 :  x=58 ; *note = "3A#"; break;
case 32 :  x=59 ; *note = "3B" ; break;
case 34 :  x=60 ; *note = "4C" ; break;
case 35 :  x=61 ; *note = "4C#"; break;
case 36 :  x=62 ; *note = "4D" ; break;
case 37 :  x=63 ; *note = "4D#"; break;
case 38 :  x=64 ; *note = "4E" ; break;
case 40 :  x=65 ; *note = "4F" ; break;
case 41 :  x=66 ; *note = "4F#"; break;
case 42 :  x=67 ; *note = "4G" ; break;
case 43 :  x=68 ; *note = "4G#"; break;
case 44 :  x=69 ; *note = "4A" ; break;
case 45 :  x=70 ; *note = "4A#"; break;
case 46 :  x=71 ; *note = "4B" ; break;
case 48 :  x=72 ; *note = "5C" ; break;
case 49 :  x=73 ; *note = "5C#"; break;
case 50 :  x=74 ; *note = "5D" ; break;
case 51 :  x=75 ; *note = "5D#"; break;
case 52 :  x=76 ; *note = "5E" ; break;
case 54 :  x=77 ; *note = "5F" ; break;
case 55 :  x=78 ; *note = "5F#"; break;
case 56 :  x=79 ; *note = "5G" ; break;
case 57 :  x=80 ; *note = "5G#"; break;
case 58 :  x=81 ; *note = "5A" ; break;
case 59 :  x=82 ; *note = "5A#"; break;
case 60 :  x=83 ; *note = "5B" ; break;
case 62 :  x=84 ; *note = "6C" ; break;
case 63 :  x=85 ; *note = "6C#"; break;
```

```
            case 64 :  x=86 ; *note = "6D" ; break;
            case 65 :  x=87 ; *note = "6D#"; break;
            case 66 :  x=88 ; *note = "6E" ; break;
            case 68 :  x=89 ; *note = "6F" ; break;
            case 69 :  x=90 ; *note = "6F#"; break;
            case 70 :  x=91 ; *note = "6G" ; break;
            case 71 :  x=92 ; *note = "6G#"; break;
            case 72 :  x=93 ; *note = "6A" ; break;
            case 73 :  x=94 ; *note = "6A#"; break;
            case 74 :  x=95 ; *note = "6B" ; break;
            case 76 :  x=96 ; *note = "7C" ; break;
            default :  x=0  ; break;

     }
     return x;
}

void c_scroll(int row, int col, int wide, int
deep, int num, int f, int color)
{
   union REGS ireg;

   row--;                                  /* Index
from (1, 1)  */
   col--;
   ireg.h.ah = f;
   ireg.h.al = num;
   ireg.h.ch = row;
   ireg.h.cl = col;
   ireg.h.dh = row + deep - 1;
   ireg.h.dl = col + wide;
   ireg.h.bh = color;

   int86(0x10, &ireg, &ireg);
}



//
//                PAINT.C
#include <stdio.h>
#include <conio.h>
```

```
#define EXTERN 1
#include "def.h"

typedef struct{
            unsigned  time;
            unsigned  comm;
            unsigned  data;
            char  status;
     }sample;

//node *headnode;
unsigned begin_time, end_time;
//extern node *begin, *end, *cur;

void paint(),search_b_e(),list_2_desk(),
load_list_to_desk();
int note_to_col();

void load_list_to_desk(unsigned _off_set)
{
     search_b_e();
     list_2_desk(_off_set);
}

void search_b_e(void)
{
     node *tnode, *wnode;                    // "
b_node==NUll " indicate that
     begin_time=48*(offset-1);               //   no
data list in that offset
     end_time=48*offset-3;
     wnode=headnode;
     if(wnode==NULL)
      begin=wnode;
     else{
      while((wnode->time<begin_time) && (wnode!=
NULL))
            wnode=wnode->next;
      if((wnode->time<=end_time)&&(wnode->time>=
begin_time))
            begin=wnode;
      else
            begin=NULL;
     }
```

```
      wnode=headnode;
      if(wnode==NULL)
       end=NULL;
      else{
       while((wnode->time <= end_time) && (wnode!=
NULL)){
            tnode=wnode;
            wnode=wnode->next;
       }
       if((tnode->time<=end_time)&&(tnode->time>=
begin_time))
            end=tnode;
       else
            end=NULL;
      }
}

void list_2_desk(unsigned off_set)          //
find bigin and end row for
{                                          // draw
current offset
      sample *conv[50];
      int i, j, m;
      unsigned b_time, e_time, note_t;
      if((begin==NULL)&&(end==NULL)){
       setd;
       draw_measure();
      }
      if((begin!=NULL)&&(end!=NULL)){
       cur=begin;
       m=0;
       do{                                    // load
current list to array
            conv[m]->time=cur->time;
            conv[m]->comm=cur->command;
            conv[m]->data=cur->note;
            conv[m]->status='u';
            cur=cur->next;
            m++;
       }while(cur!=end->next);
       m--;                        // m is max number of
list
       i=0;
       while(i<=m){
```

```
            if(conv[i]->status!='n'){
             if(conv[i]->comm/16==note_off/16){
              b_time=begin_time;
              e_time=conv[i]->time;
              note_t=conv[i]->data;
              conv[i]->status='n';
             }
             if(conv[i]->comm/16==note_on/16){
              b_time=conv[i]->time;     // set
note on
              note_t=conv[i]->data;
              conv[i]->status='n';
              j=i;
              while(j<=m){              //search note
off
                  if(conv[j]->status!='n'){
                   if(conv[j]->data==note_t){
                     e_time=conv[j]->time;
                     conv[j]->status='n';
                   }
                  }
                  j++;
              }
              if(e_time>end_time)
                  e_time=end_time;
             }
             paint(note_t, b_time, e_time,
off_set);
            }
            i++;
       }
     }
}

void paint(unsigned note_v, unsigned be_time,
unsigned en_time, unsigned off_set)
{                              // draw current offset
     int b_row, e_row, note_c, x;
     b_row=((be_time-(48*(off_set-1)))/3)+8;
     e_row=((en_time-(48*(off_set-1)))/3)+8;
     note_c=note_to_col(note_v);
     setd;
     for(x=b_row;x<=e_row;x++){
      gotoxy(note_c, x);
```

```
        cprintf("▒");
      }
      hide_cursor();
}

int note_to_col(int note)     // for convert note
value to column position
{
     int x;
     switch(note)
     {
          case 36 :  x=6  ;  break;
          case 37 :  x=7  ;  break;
          case 38 :  x=8  ;  break;
          case 39 :  x=9  ;  break;
          case 40 :  x=10 ;  break;
          case 41 :  x=12 ;  break;
          case 42 :  x=13 ;  break;
          case 43 :  x=14 ;  break;
          case 44 :  x=15 ;  break;
          case 45 :  x=16 ;  break;
          case 46 :  x=17 ;  break;
          case 47 :  x=18 ;  break;
          case 48 :  x=20 ;  break;
          case 49 :  x=21 ;  break;
          case 50 :  x=22 ;  break;
          case 51 :  x=23 ;  break;
          case 52 :  x=24 ;  break;
          case 53 :  x=26 ;  break;
          case 54 :  x=27 ;  break;
          case 55 :  x=28 ;  break;
          case 56 :  x=29 ;  break;
          case 57 :  x=30 ;  break;
          case 58 :  x=31 ;  break;
          case 59 :  x=32 ;  break;
          case 60 :  x=34 ;  break;
          case 61 :  x=35 ;  break;
          case 62 :  x=36 ;  break;
          case 63 :  x=37 ;  break;
          case 64 :  x=38 ;  break;
          case 65 :  x=40 ;  break;
          case 66 :  x=41 ;  break;
          case 67 :  x=42 ;  break;
          case 68 :  x=43 ;  break;
```

```
            case 69 :   x=44 ;  break;
            case 70 :   x=45 ;  break;
            case 71 :   x=46 ;  break;
            case 72 :   x=48 ;  break;
            case 73 :   x=49 ;  break;
            case 74 :   x=50 ;  break;
            case 75 :   x=51 ;  break;
            case 76 :   x=52 ;  break;
            case 77 :   x=54 ;  break;
            case 78 :   x=55 ;  break;
            case 79 :   x=56 ;  break;
            case 80 :   x=57 ;  break;
            case 81 :   x=58 ;  break;
            case 82 :   x=59 ;  break;
            case 83 :   x=60 ;  break;
            case 84 :   x=62 ;  break;
            case 85 :   x=63 ;  break;
            case 86 :   x=64 ;  break;
            case 87 :   x=65 ;  break;
            case 88 :   x=66 ;  break;
            case 89 :   x=68 ;  break;
            case 90 :   x=69 ;  break;
            case 91 :   x=70 ;  break;
            case 92 :   x=71 ;  break;
            case 93 :   x=72 ;  break;
            case 94 :   x=73 ;  break;
            case 95 :   x=74 ;  break;
            case 96 :   x=76 ;  break;
            default :   x=0  ;  break;

    }
    return x;
}

//
//                  TESTLINK.C
#include <stdio.h>
#include <malloc.h>
#include <conio.h>

typedef struct singlelist{
            unsigned int time;
            unsigned int command;
            unsigned int note;
```

```
            unsigned int data;
            struct singlelist  *prior;
            struct singlelist  *next;
    }node;
typedef struct list{
            unsigned int time;
            unsigned int command;
            unsigned int note;
            unsigned int data;
    }dim;
extern struct singlelist  *headnode, *tailnode;
struct singlelist  *walknode , *newnode, *
nextnode;

void deletenode(node *before);
//void io_dat(node *nodet);
node *search(node *ser);
node *new_node(void);
void clearnode(void);
void ins_dat(void);
void disp_dat(void);
void save(char file[]);
void load(char file[]);
char midi_file[15];

/*void main()
{
      char midi_file[15];
      int choise,i;
      headnode=tailnode=NULL;
      headnode->prior = NULL;
      headnode->next = NULL;              //
      printf("\n%u",headnode->time);
      do{
       printf("\n1.Inseart node\n");
       printf("2.Viwe time in sequence\n");
       printf("3.Save list\n");
       printf("4.Load list\n");
       printf("5.End and clead all List\n");
       choise = getch();
       switch(choise){
        case '1' : ins_dat(); break;
        case '2' : disp_dat(); break;
        case '3' : printf("enter file name : ");
```

```
                scanf("%15s",midi_file);
              //     gets(midi_file);
                save(midi_file); break;
        case '4' : printf("enter file name : ");
              //     gets(midi_file);
                scanf("%15s",midi_file);
                load(midi_file);

                break;
        case '5' : clearnode(); exit(1);
       }
     }while(choise != 5);
}  */

node *new_node(void){

     return(node *)malloc(sizeof(node));
}


void delnode(node *p){

     free((void *)p);
}
/*
void io_dat(node *nodet){

     nodet->time=time;
     nodet->command=_ch;
     nodet->note=note;
     nodet->data=vel;
     unsigned int dat;
     printf("\nInput Time:");
     scanf("%u",&dat);
     nodet->time=dat;
     printf("\nInput Command:");
     scanf("%u",nodet->command);
     printf("\nInput Note:");
     scanf("%u",nodet->note);
     printf("\nInput Data:");
     scanf("%u",nodet->data);
} */

void clearnode(void){
```

```
        walknode = headnode;
        newnode = walknode->next;
        while(walknode != NULL){
          free(walknode);
          walknode = newnode;
          newnode = newnode->next;
        }
}

void deletenode(node *before){

        node *after;
        if(before->next != NULL){
          after = before->next->next;
          delnode(before->next);
          before->next = after;
          after->prior = before;
        }
}

node *search(node *ser){

        node *tempnode, *testnode;
        testnode = headnode;
        while((testnode->time < ser->time) &&
(testnode != NULL))
          testnode=testnode->next;
        return testnode;
}
/*
void ins_dat(void){

        if(headnode==NULL){
          headnode = new_node();
          newnode = headnode;
          io_dat(newnode);
          newnode->next=NULL;
          newnode->prior=NULL;
          tailnode=newnode;
          printf("\n%u",newnode->time);
        }
        else{
          newnode = new_node();
```

```
            io_dat(newnode);
            printf("\n%u",newnode->time);
            walknode = search(newnode);
            if(walknode!=NULL){
             if(walknode->prior==NULL)
                  headnode=newnode;
             else
                  walknode->prior->next = newnode;
             newnode->prior = walknode->prior;
             walknode->prior = newnode;
             newnode->next = walknode;
            }
            else{
             newnode->next = NULL;
             newnode->prior=tailnode;
             tailnode->next=newnode;
             tailnode=newnode;
            }
           }
} */

void disp_dat(void){

      walknode = headnode;
      do{
       printf("  Time=%u  ",walknode->time);
       walknode = walknode->next;
      }while(walknode != NULL);
}

void save(char file[]){

      register int t;
      node *info;
      FILE *fp;

      if((fp=fopen(file, "wb")) == NULL){
        printf("Cannot open file\n");
        exit(1);
      }

      info = headnode;
      while(info){
        fwrite(info, sizeof(node), 1, fp);
```

```
        info = info->next;
      }
      fclose(fp);
}

void load(char file[]){

      register int t, i;
      node *info, *temp=NULL;
      FILE *fp;
      if((fp=fopen(file, "rb")) == NULL){
        printf("Cannot open file\n");
        exit(1);
      }
      clearnode();

      while(headnode){          //clear all list
        info = headnode->next;
        free(info);
        headnode=info;
      }

      printf("\nloading file\n");

      headnode=(node *)malloc(sizeof(node));
      if(!headnode){
        printf("Out of memory\n");
        return;
      }
      info=headnode;
      while(!feof(fp)){
        if(1 != fread(info, sizeof(node), 1, fp))
         break;
        info->next = (node *)malloc(sizeof(node))
;
        if(!info->next){
          printf("Out of memory\n");
          return;
        }
        info->prior=temp;
        temp=info;
        info=info->next;
      }
      temp->next=NULL;
```

```
        tailnode=temp;

        headnode->prior=NULL;
        fclose(fp);
}


void in_list(unsigned time_o, unsigned com_m,
unsigned data_1,
          unsigned data_2)
{

        if(headnode==NULL){
         headnode = new_node();
         newnode = headnode;

         newnode->time=time_o;
         newnode->command=com_m;
         newnode->note=data_1;
         newnode->data=data_2;

         newnode->next=NULL;
         newnode->prior=NULL;
         tailnode=newnode;
        }
        else{
         newnode = new_node();

         newnode->time=time_o;
         newnode->command=com_m;
         newnode->note=data_1;
         newnode->data=data_2;

         walknode = search(newnode);
         if(walknode!=NULL){
          if(walknode->prior==NULL)
               headnode=newnode;
          else
               walknode->prior->next = newnode;
          newnode->prior = walknode->prior;
          walknode->prior = newnode;
          newnode->next = walknode;
         }
         else{
```

```
            newnode->next = NULL;
            newnode->prior=tailnode;
            tailnode->next=newnode;
            tailnode=newnode;
          }
        }
        gotoxy(1,1);
        printf("%5u %3u %5u",newnode->time,newnode-
>command, newnode->note);
        hide_cursor();
}

void del_list(unsigned time_o, unsigned com_m,
unsigned data_1,
            unsigned data_2)
{
        walknode = headnode;
        while((walknode->time!=time_o)&&(walknode->
command!=com_m)
            &&(walknode->note!=data_1)&&(walknode->
data!=data_2)){

         walknode=walknode->next;
        }
        walknode->prior->next=walknode->next;
        walknode->next->prior=walknode->prior;
        free(walknode);

}//
//                MOUSE.C
#include <conio.h>

#define  EXTERN 1
#include "menu.h"

void mouse_counters(int *row, int *col);
void cursor(int row, int col);
void read_mouse(int *row, int *col, int *button);
void mouse_cursor_on(void);
void hide_cursor(void);
void mouse_vrange(int min, int max);
void mouse_hrange(int min, int max);

void testmouse()
```

```
{
     int button;
/*   int i, row, col, s, press, button, v, h;*/
     if(!mouse_status(&button))
     {
          printf("Mouse not install ,program
abort!");
          exit(1);
     }
/*   gotoxy(1, 1);
     printf("Enter mouse sensitivity  ");
     scanf("%d",&s);
     gotoxy(1,1);
     printf("press left button and hold down");
     do{
          press = mouse_press(&row, &col, 1);
     }while(press!=1);
     gotoxy(10,1);
     printf("▒");
     printf("left button now press ");
     printf("release button to quit ");
     mouse_cursor_on();
     i=1;
     do{
          mouse_counters(&v, &h);
          if(v >= s)
          {
               gotoxy(10,i);
               printf("▒");
               if(i<25)
                 i++;
          }
          if(v <= -s)
          {
               gotoxy(10,i);
               printf("_");
               if(i>1)
                i--;
          }
     }while(mouse_press(&row, &col, 1)==1);
     printf("no button press\n");
     do{
          press = mouse_press(&row, &col, 2);
     }while(press!=2);
```

```
                */
}
int mouse_status(int *buttons)
{
   union REGS ireg;

   ireg.x.ax = 0x00;                    /* Function
0x00--mouse status   */
   int86(0x33, &ireg, &ireg);           /* bx is
number of mouse buttons */
   *buttons = ireg.x.bx;
   return ireg.x.ax;
}


void mouse_cursor_on(void)
{
   union REGS ireg;

   ireg.x.ax = 0x01;                    /*
Toggle cursor on          */
   int86(0x33, &ireg, &ireg);

}



void mouse_cursor_off(void)
{
   union REGS ireg;

   ireg.x.ax = 0x02;                    /*
Toggle cursor off         */
   int86(0x33, &ireg, &ireg);
}


void read_mouse(int *row, int *col, int *button)
{
   union REGS ireg;

   ireg.x.ax = 0x03;                    /* Read mouse
position and status */
```

```
   int86(0x33, &ireg, &ireg);
   *button = ireg.x.bx;
   *col = ireg.x.cx/8 + 1;
   *row = ireg.x.dx/8 + 1;

}


int mouse_press(int *row, int *col, int button)
{
   union REGS ireg;

   ireg.x.ax = 0x05;                /* Function to
get mouse data          */
   ireg.x.bx = button;              /* Update only
on this button press */
   int86(0x33, &ireg, &ireg);
   *col = ireg.x.cx/8 + 1;
   *row = ireg.x.dx/8 + 1;
   return (ireg.x.ax);              /* Return
button status           */
}


int mouse_release(int *row, int *col, int button)
{
   union REGS ireg;

   ireg.x.ax = 0x06;                        /*
Function to get mouse data */
   ireg.x.bx = button;                      /* The
button to inspect       */
   int86(0x33, &ireg, &ireg);
   *col = ireg.x.cx;
   *row = ireg.x.dx;
   return (ireg.x.ax);                      /* Return
button status        */
}


void mouse_vrange(int min, int max)
{
   union REGS ireg;
```

```
   ireg.x.ax = 0x08;                    /* Function
to set vertical range */
   ireg.x.cx = min;
   ireg.x.dx = max;
   int86(0x33, &ireg, &ireg);
}


void mouse_hrange(int min, int max)
{
   union REGS ireg;

   ireg.x.ax = 0x07;                  /* Function to
set horizontal range */
   ireg.x.cx = min;
   ireg.x.dx = max;
   int86(0x33, &ireg, &ireg);
}


void set_mouse(int row, int col)
{
   union REGS ireg;

   ireg.x.ax = 0x04;                     /* Function
to set mouse position */
   ireg.x.cx = col;
   ireg.x.dx = row;
   int86(0x33, &ireg, &ireg);
}

void mouse_cursor_g(void)
{
   static unsigned int mask[] = {
      0x3ff, 0x1ff, 0x0fff, 0x7ff, 0x3ff, 0x1ff,
0xff, 0x7f,
      0x3f, 0x1f, 0x1ff, 0x10ff, 0x30ff, 0xf87f,
0xf87f, 0xfc3f,
      0x00, 0x4000, 0x6000, 0x7000, 0x7800, 0
x7c00, 0x7e00, 0x7f00,
      0x7f80, 0x78c0, 0x7c00, 0x4600, 0x0600, 0
x0300, 0x300, 0x0180};
   union REGS ireg;
```

```
   struct SREGS s;                  /* CAUTION: MUST
BE "SREG" FOR ECO-C88 */

   segread(&s);                                  /* Read
segment reg          */
   s.es = s.ds;                                  /* Load
es with dseg         */
   ireg.x.ax = 0x09;                             /* Set
graphics cursor      */
   ireg.x.bx = 0x01;                             /*
Column hot spot           */
   ireg.x.cx = 0x01;                             /* Row
hot spot                  */
   ireg.x.dx = (unsigned) mask;                  /*
Pointer to mask           */
   int86x(0x33, &ireg, &ireg, &s);               /* Set
all registers             */
}


void mouse_cursor_t(int type, int screen, int
mask)
{
   union REGS ireg;
   struct SREGS s;                  /* CAUTION: MUST
BE "SREG" FOR ECO-C88 */

   segread(&s);                                  /* Read
segment reg          */
   s.es = s.ds;                                  /* Load
es with dseg         */
   ireg.x.ax = 0x0a;                             /* Set
text cursor          */
   ireg.x.bx = type;                             /* Type
of cursor            */
   ireg.x.cx = (unsigned) screen;                /*
Screen mask               */
   ireg.x.dx = (unsigned) mask;                  /*
Cursor mask               */
   int86x(0x33, &ireg, &ireg, &s);               /* Set
all registers        */
}
```

```
void mouse_counters(int *row, int *col)
{
   union REGS ireg;

   ireg.x.ax = 0x0b;                    /* Function
to read motion counts */
   int86(0x33, &ireg, &ireg);
   *col = ireg.x.cx;
   *row = ireg.x.dx;
}

void hide_cursor(void)
{
  cursor(26,1);
}

void cursor(int row, int col)
{
     union REGS ireg;

     ireg.h.ah = 0x02;
     ireg.h.bh = getvpage();
     ireg.h.dh = row - 1;
     ireg.h.dl = col - 1;

     int86(0x10, &ireg, &ireg);
}


int getvpage(void)
{
   union REGS ireg;

   ireg.h.ah = 0x0f;
/* Function 0x0f */

   int86(0x10, &ireg, &ireg);
   return (ireg.h.bh);
}
/*
void wait_on(int button)
{
     if(button == LBUTTON)
     while(mouse_press());
```

```
    if(button == RBUTTON)
     while(mouse_press());
} */

#include <stdlib.h>                    /* Needed for
the memchr() function */

unsigned int read_key(int button)
{
   char press, *tptr;
   int dhm, vmove, hmove, temp;
   hide_cursor();
   while (TRUE) {
      if (mouse_here) {                          /* If
there is a mouse          */
     press = mouse_press(&temp, &temp, button);
     if (press == button) {
        return ENTER;
     }
     mouse_counters(&vmove, &hmove); /* Get
relative movement      */
     dhm += hmove;
     if (dhm > delta) {                          /* Enough to
move it right    */
        return RARROW;
     }
     if (dhm < -delta) {                         /* How about
left?            */
        return LARROW;
     }
      }
      temp = _kbhit();                            /* Poll
the keyboard          */
      if (temp == 0) {                           /* No
key                      */
     continue;
      }
      if (temp == ENTER)                         /*
Pressed Enter key?        */
     return ENTER;

      if (temp < ASCII_LIMIT) {                  /*
Single character?         */
```

```
      tptr = memchr(cptr, tolower(temp),
menucount);
      if (tptr) {
         last = tptr - cptr;          /* Found a
match              */
         return ENTER;
      }
       }
       return temp - EXTEND_OFF;          /*
Extended keycode?          */
   }
   return 0;                              /*
Should never be here       */
}


#include <dos.h>

#define ZEROFLAG  0x40

int _kbhit(void)
{
   int zflag;
   union REGS ireg;

   ireg.h.ah = 0x06;                  /* Function 6
*/
   ireg.h.dl = 0xff;                  /* We want to
read it, not output */
   zflag = intdos(&ireg, &ireg);

   if ( (zflag & ZEROFLAG) == 0) {   /* A
character and 0?              */
      if (ireg.h.al == 0) {           /* Extended
keycode?              */
      ireg.h.ah = 0x06;
          ireg.h.dl = 0xff;
          intdos(&ireg, &ireg);
          return (ireg.h.al + 0x100);
 }
 return ireg.h.al;                    /* An ASCII
character             */
   }
```

```
   return 0;                              /* No
character                        */
}


unsigned int read_key_all(int *row, int *col)
{
   char press, *tptr;
   int dhm, dvm, vmove, hmove, temp, tbutton;

   hide_cursor();
                        /* Newer machines fast enough
to read  */
                        /* multiple presses when user
wants    */
                        /* only one. This next loop
clears out */
                        /* false presses.
*/

   if (mouse_here) {
      while (TRUE) {
      read_mouse(row, col, &tbutton);
      if (tbutton == 0)
        break;
      }
   }
   dvm = dhm = 0;

   while (TRUE) {
      if (mouse_here) {                    /* If
there is a mouse       */
     read_mouse(row, col, &tbutton);

     if (tbutton == LBUTTON) {
        return ENTER;                     /* User made
a choice         */
     }
     if (tbutton == RBUTTON) {
        return ESCAPE;                    /* Return
without choice     */
     }

          /* Could put MBUTTON code here */
```

```
      mouse_counters(&vmove, &hmove); /* Get
relative movement      */
      dhm += hmove;                   /* Get
horizontal movement    */
      dvm += vmove;                   /* Get
vertical movement      */


      if (dhm > DELTA) {
         return RARROW;
      }
      if (dhm < -DELTA) {
         return LARROW;
      }
      if (dvm > DELTA) {
         return DARROW;
      }
      if (dvm < -DELTA) {
         return UARROW;
      }
       }
                          /* _kbhit() changes:
*/
                          /* LIST0305.C changed to
avoid  */
                          /* conflict with compilers
that */
                          /* have a kbhit() function
*/
                          /* Note leading underscore
*/

       temp = _kbhit();                /* Poll the
keyboard              */

       if (temp == 0) {                /* No key
press                  */
      continue;
       }
       if (temp == ENTER) {            /* Pressed
Enter key?             */
      return ENTER;
       }
```

```
        if (temp == ESCAPE) {
       return ESCAPE;
        }
        if (temp < 127) {                    /* Pressed
ASCII key?            */
           last = temp;
           return ENTER;
        }
        last = 0;
        return temp - EXTEND_OFF;            /* Extended
keycode?              */
    }
}
```

บรรณานุกรม

1. ธานินทร์ ถาวรศาสนวงศ์, ทินกร ดุ๊ก ,"**การอินเทอร์เฟส IBM/PC**", ฟิสิกส์เซ็นเตอร์, 138 หน้า, 2536

2. ชูชัย ธนสารตั้งเจริญ, ทินกร ดุ๊ก, " **การใช้งาน Z-80** ", ฟิสิกส์เซ็นเตอร์, 180 หน้า, 2533

3. Craig anderton, " MIDI for Musicians ", Amsco Publiccations, 150 p. ,1986.

4. John Uffenback, " Microcomputer and Microprocessor", Prentice-Hall, 690 p., 1991.

5. Richard Evers, "MIDI - Musical Instrument Digital Interface", McGraw Hill, 200 p, 1987.

กิตติกรรมประกาศ

ขอขอบคุณ

- รศ.ดร.มนัส สังวรณ์ศิลป์ ที่ให้การสนับสนุนการทำ MIDI PROJECT
- นายอาคม บุษยกุล ที่ให้คำแนะนำ
- นายนิธิ ใจแก้ว เอื้อเฟื้ออุปกรณ์
- นางสาวปณการญจน์ สัตยพงศ์ฐิติ ที่ให้กำลังใจ