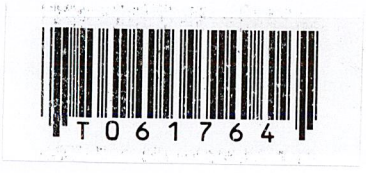
โปรแกรมตั้งเสียงดนตรีและบันทึกตัวโน้ตอัตโนมัติ

MUSICAL TUNING AND AUTOMATIC RECORDING SOFTWARE



โดย

นาย สถิต ไพโรจน์

อาจารย์ที่ปรึกษา

รศ. สมศักดิ์ มิตะถา

อ. เจริญ วงษ์ชุ่มเย็น

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2547

โปรแกรมตั้งเสียงดนตรีและบันทึกตัวโน้ตอัตโนมัติ

MUSICAL TUNING AND AUTOMATIC RECORDING SOFTWARE

นาย สถิต ไพโรจน์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2547

ปริญญานิพนธ์ ปีการศึกษา 2547

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง โปรแกรมตั้งเสียงดนตรีและบันทึกตัวโน้ตอัตโนมัติ

MUSICAL TUNING AND AUTOMATIC RECORDING SOFTWARE

คณะผู้จัดทำ นาย สถิต ไพโรจน์ รหัสประจำตัว 45015386

..... อาจารย์ที่ปรึกษา

(รศ. สมศักดิ์ มิตะถา)

..... อาจารย์ที่ปรึกษา

(อ. เจริญ วงษ์ชุ่มเย็น)

โปรแกรมตั้งเสียงดนตรีและบันทึกตัวโน้ตอัตโนมัติ

นาย สถิต ไพโรจน์ 45015386
รศ. สมศักดิ์ มิตะถา อาจารย์ที่ปรึกษา
อ. เจริญ วงษ์ชุ่มเย็น อาจารย์ที่ปรึกษา
ปีการศึกษา 2547

บทคัดย่อ

วิวัฒนาการของเทคโนโลยีต่างๆล้วนมีบทบาทที่สำคัญต่อชีวิตประจำวันของมนุษย์เรา ทุกอย่างย่อมมีการพัฒนาให้มีความสะดวกสบายในการใช้งานมากขึ้น แน่นอน ดนตรี ก็เป็นหนึ่งในหลายๆสิ่งเหล่านั้นที่มีการพัฒนามาอย่างต่อเนื่อง ซึ่งเมื่อเรากล่าวถึงดนตรีแล้ว ย่อมหลีกเลี่ยงไม่ได้ที่จะพูดถึงโน้ตดนตรี อันเป็นส่วนสำคัญที่จะทำให้เราสามารถสร้างสรรค์เสียงเพลง หรือแทนเสียงเพลงเหล่านั้นออกมาเป็นสัญลักษณ์ และนี่เองที่เป็นเหตุผลสำคัญของโครงงานในครั้งนี้ ที่จะได้นำเอาเทคนิคและวิธีการต่างๆมาประยุกต์ใช้ เพื่อเป็นทางเลือกหนึ่งในการให้ความสะดวกสบายเกี่ยวกับเสียงดนตรี

โครงงานนี้เป็นการพัฒนาโปรแกรมและฮาร์ดแวร์สำหรับประมวลผลเสียงโน้ตดนตรีที่รับเข้ามาแล้วแปลงให้อยู่ในรูปของเลขฐานสิบหก โดยการทำงานของโปรแกรมและฮาร์ดแวร์โดยรวมนั้นจะแบ่งออกเป็น 3 ส่วนหลักๆคือ ส่วนการรับเสียงโน้ตดนตรี, ส่วนการประมวลผล และสุดท้ายส่วนแสดงผล ซึ่งการทำงานนั้นจะเริ่มจากการรับเสียงของเครื่องดนตรีเข้ามาทางฮาร์ทแวร์ที่ได้จัดทำขึ้น ต่อมาจะถูกทำการแปลงความถี่เสียงนั้นให้เป็นสัญญาณดิจิตอลแล้วแปลงให้เป็นเลขฐานสิบหก จากนั้นข้อมูลจะถูกส่งไปยังส่วนการประมวลผล เพื่อวิเคราะห์ตัวโน้ตและสัญลักษณ์ต่างๆ สุดท้ายข้อมูลที่ได้นี้จะถูกนำไปเขียนลงบรรทัดห้าเส้นนั่นเอง นอกจากนี้จะมีการนำเสนอให้เห็นถึงเทคนิคต่างๆที่ได้ทำการศึกษาและทดลองมาเหตุใดจึงเลือกเอาเทคนิคนั้นๆมาใช้ รวมทั้งผลการทดสอบโปรแกรมและฮาร์ดแวร์ที่ได้จัดทำขึ้นมา

MUSICAL TUNING AND AUTOMATIC RECORDING SOFTWARE

Mr. Sathit Pairoch	45015386
ASSOC.PROF. Somsak Mitatha	Advisor
Mr. Charoen Vongchumyen	Advisor
Academic Year 2004	

ABSTRACT

Recently, Technology has become a part of life and it has been applied to satisfy human such as musical technology. Because musical note is still the standard for musicians, so we think of technology that applied by pattern recognition to analyze the structure of printed musical note. After that the computer can recognize musical note and play music from these data correctly.

The purpose of the project is to develop a program that transforms received notes into hexadecimal. The method of this program is provided into 3 main part that are input of notes, processing and output. It will begin with receiving the sound of instrument through the provided hardware. Then the sound will be changed the frequency of sound into digital signal and then change it into hexadecimal. Then the information will be send to processing part to analyses notes and other symbol. Finally ,this information will be written in a line. Moreover, it will present the other technique that have been studied and experimented why we chose to use those technique including the testing result of the program.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สามารถสำเร็จลุล่วงได้เป็นอย่างดี ด้วยคำแนะนำ คำปรึกษา และได้รับการดูแลจากหลายๆฝ่ายด้วยกัน แน่นอนว่าแรงผลักดันสำคัญอันทำให้ปริญญานิพนธ์นี้สามารถสำเร็จลุล่วงได้นั่น ก็ด้วยความกรุณาจากท่านอาจารย์ที่ปรึกษา รศ.สมศักดิ์ มิตะถา และ อ. เจริญ วงษ์ชุ่มเย็น ที่คอยประสิทธิประสาทวิชาความรู้ ให้ความเอาใจใส่ ห่วงใย และให้ความช่วยเหลือเสมอมา ซึ่งข้าพเจ้ารู้สึกทราบซึ้งในความอนุเคราะห์และขอขอบพระคุณเป็นอย่างสูง

ขอขอบพระคุณภาควิชาวิศวกรรมคอมพิวเตอร์ที่คอยจัดเตรียมสิ่งอำนวยความสะดวกต่างๆ ไม่ว่าจะเป็นห้องวิจัย อินเทอร์เน็ตความเร็วสูงสำหรับค้นคว้าหาข้อมูล หรือความอนุเคราะห์อื่นๆ เพื่อให้การดำเนินงานนั้นเป็นไปอย่างต่อเนื่อง ซึ่งท้ายที่สุดก็ได้เป็นปริญญานิพนธ์นี้ออกมา

ขอขอบคุณรุ่นพี่ที่คอยให้คำปรึกษา เพื่อนๆที่คอยให้กำลังใจและเคียงข้างกันเสมอมา ไม่ว่าวันเวลาจะผ่านไปนานเท่าไรสิ่งหนึ่งที่ยังคงมีอยู่และจะยังคงมีไว้ตราบนานเท่านานคือมิตรภาพระหว่างเรา และที่เห็นทีจะขาดเสียมิได้ ขอขอบคุณห้องฮาร์ดแวร์อันเปรียบเสมือนบ้านอีกหลังหนึ่งของเรา

สุดท้ายนี้ทุกสิ่งทุกอย่างที่กล่าวมาคงไม่อาจเกิดขึ้น หากข้าพเจ้าไม่ได้มายืนอยู่ตรงจุดนี้ ขอกราบขอบพระคุณบุพการีผู้ให้กำเนิด ที่คอยเลี้ยงดู อบรมสั่งสอน ที่สำคัญได้ให้โอกาสทางการศึกษาแก่ข้าพเจ้าอันทำให้มีวันนี้ วันที่เราประสบความสำเร็จ และความภาคภูมิใจเป็นของเราทุกคน

คุณค่าและประโยชน์อันพึงมีจากปริญญานิพนธ์ฉบับนี้ ผู้วิจัยขอมอบแด่ผู้มีพระคุณทุกท่าน

สถิต ไพโรจน์

สารบัญ

สารบัญ(ต่อ)

สารบัญ(ต่อ)

สารบัญตาราง

สารบัญภาพ

สารบัญภาพ(ต่อ)

หน้าที่

รูปที่

บทที่ 1
บทนำ

1.1 หลักการและเหตุผล

เนื่องมาจากวิวัฒนาการทางเทคโนโลยีที่ก้าวหน้าอย่างต่อเนื่องจากอดีตจนถึงปัจจุบัน ทำให้เราได้เห็นถึงสิ่งต่างๆที่อยู่รอบตัวเราถูกนำมาพัฒนาปรับปรุงและเปลี่ยนแปลงให้ตอบสนองต่อความต้องการของมนุษย์และเพื่อให้สอดคล้องกับยุคสมัยไม่ว่าจะเป็น โทรศัพท์ ยานพาหนะ หรือที่เห็นได้เด่นชัดก็คือคอมพิวเตอร์ สิ่งต่างๆเหล่านี้ล้วนทำให้รูปแบบการดำเนินชีวิตของคนเราเปลี่ยนแปลงไป เช่นเดียวกับวิวัฒนาการของมนุษย์กับเสียงดนตรีที่อยู่ควบคู่กันจากอดีตจนถึงปัจจุบันและจะเป็นเช่นนี้เรื่อยไป เพราะว่ามนุษย์เราเป็นสิ่งมีชีวิต ที่มีความรู้สึกนึกคิด มีอารมณ์อ่อนไหว เช่นเดียวกับเสียงดนตรีที่มีความไพเราะและให้อารมณ์ในตัวของมันเอง วิวัฒนาการทางดนตรีนั้นได้รับการพัฒนามาเรื่อยๆอย่างต่อเนื่องเพื่อให้ตอบสนองต่อความต้องการมากที่สุด เมื่อพูดถึงดนตรีแล้วนั้น โน้ตดนตรีถือเป็นส่วนพื้นฐานของดนตรีเลยก็ว่าได้ อันเป็นส่วนสำคัญที่จะทำให้เราสามารถสร้างสรรค์เสียงเพลง หรือแทนเสียงเพลงเหล่านั้นออกมาให้อยู่ในรูปของสัญลักษณ์ ซึ่งไม่ว่าจะเป็นเครื่องดนตรีชนิดใดก็ล้วนแต่ใช้โน้ตดนตรีในการที่จะสร้างสรรค์เป็นบทเพลงขึ้นมา

ดังนั้นจากเหตุผลข้างต้น ผู้จัดทำโครงงานจึงมีแนวคิดที่จะทำการออกแบบและพัฒนาโปรแกรมและฮาร์ดแวร์ที่สามารถทำให้บุคคลทั่วไปที่ไม่รู้ทฤษฎีดนตรีและทฤษฎีของตัวโน้ต หรือเป็นบุคคลที่พอจะรู้เรื่องโน้ตดนตรีบ้างแล้ว มีความสะดวกยิ่งขึ้นในการที่จะเขียนเพลงหรือแต่งเพลงขึ้นมา โดยโครงงานชิ้นนี้จะสามารถที่จะนำเครื่องดนตรีมาเล่นใส่เครื่องคอมพิวเตอร์ แล้วส่วนประมวลผลต่างๆจะทำหน้าที่ทำให้เสียงดนตรีที่เข้าไปนั้นเขียนเป็นโน้ตดนตรีออกมา (โดยผู้เล่นจะต้องเล่นทีละตัวโน้ต-เท่านั้น) แล้วยังสามารถเป็นมาตรฐานในการตั้งเสียงของเครื่องดนตรีชนิดต่างๆได้ เพื่อเป็นทางเลือกหนึ่งในการให้ความสะดวกสบายเกี่ยวกับเสียงดนตรี และยังสามารถนำเสียงที่เล่นไปในตอนแรกนั้นกลับมาฟังพร้อมตัวโน้ตได้ หรือปริ้นโน้ตดนตรีออกมาดูได้

1.2 วัตถุประสงค์

1. เพื่อเป็นมาตรฐานในการตั้งเสียงของตัวโน้ตของเครื่องดนตรีชนิดต่างๆ
2. เพื่ออำนวยความสะดวกแก่ผู้หัดเขียนโน้ตดนตรี
3. เพื่อศึกษาทฤษฎีทางดนตรี ให้เข้าใจความหมายของโน้ตและสัญลักษณ์ต่างๆ อันจะทำให้โครงงานมีความถูกต้องและเป็นมาตรฐานสากล
4. เพื่อศึกษาวงจรกรองความถี่ในแบบต่างๆ

1.3 ผลที่คาดว่าจะได้รับ

1. ได้รับความรู้ ความเข้าใจเกี่ยวกับทฤษฎีดนตรีและโน้ตดนตรี
2. ได้รับความรู้ เกี่ยวกับการอินเตอร์เฟสระหว่างฮาร์ดแวร์กับเครื่องคอมพิวเตอร์
3. ได้นำความรู้ที่ได้เรียนมานำไปประยุกต์ใช้กับสาขาวิชาอื่นเพื่อให้เกิดประโยชน์
4. เครื่องดนตรีมีความถี่ของเสียงที่แน่นอนตามมาตรฐาน
5. ผู้ที่เขียนโน้ตดนตรีไม่เป็นมีความสะดวกในการแต่งเพลง
6. ได้ชิ้นงานตามที่ได้กล่าวมาในตอนต้น

1.4 ขอบเขตของการพัฒนา

1.4.1 สามารถเป็นตัวตั้งเสียงของเครื่องดนตรีให้มีความถี่ตรงตามมาตรฐาน

โดยจะรับเสียงของเครื่องดนตรีเข้ามาทางฮาร์ดแวร์ที่ได้จัดทำขึ้นมา โดย นำกีตาร์ไฟฟ้ามาเสียบLine in ที่ตัวฮาร์ดแวร์แล้วดีดเพื่อตั้งเสียง(ทีละหนึ่งตัวโน้ต) โปรแกรมจะทำการประมวลผลแล้วบอกความถี่และเสียงของตัวโน้ตว่าเป็นเสียงอะไรผ่านทางหน้าจอ และจะมีไฟแสดงผลให้ดูว่าตัวโน้ตนั้นมีความถี่ตรงตามมาตรฐานหรือยังโดยถ้ามีความถี่ต่ำกว่ามาตรฐานไฟสีแดงทางซ้ายจะติดแสดงว่าให้เพิ่มความถี่ขึ้นอีกถ้าได้มาตรฐานแล้วไฟสีเขียว ตรงกลางจะติด หรือถ้ามีความถี่เสียงเกินมาตรฐานไฟสีแดงทางขวาจะติดแสดงว่าให้ลดความถี่ลงจนไฟสีเขียวติดจึงจะตรงตามมาตรฐาน

1.4.2 สามารถบันทึกเสียงที่เข้าไปแล้วเขียนเป็นตัวโน้ต

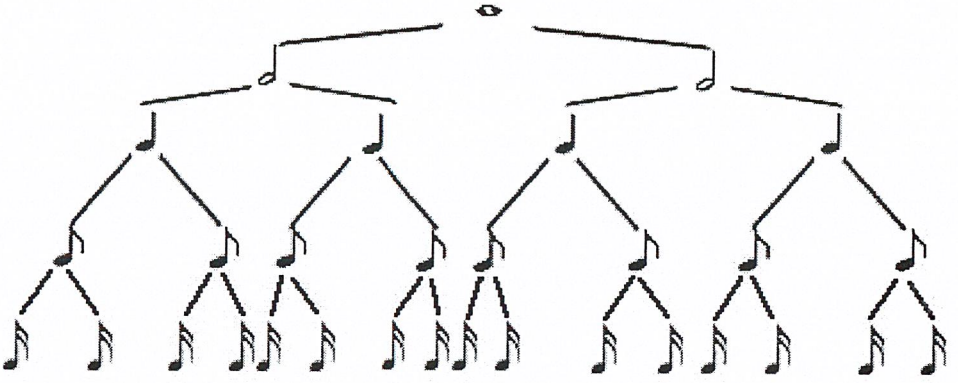
โดยสามารถบอกถึงเสียงที่เข้าไปได้ว่าเป็นโน้ตอะไรและนำเสียงที่เข้าไปเขียนเป็นโน้ตเพลงลงบรรทัดห้าเส้น และสามารถปริ้นออกมาได้ โดยในโปรแกรมส่วนนี้จะทำงานต่อจากโปรแกรมด้านบนคือเมื่อทำการตั้งสายเสร็จเรียบร้อยแล้วเรียกโปรแกรมในส่วนนี้ขึ้นมา เมื่อเล่นดนตรีลงไปโปรแกรมในส่วนนี้จะทำการนำเสียงที่เล่นไปนั้นแสดงออกเป็นตัวโน้ตบนบรรทัดห้าเส้น(ในที่นี้หมายถึงทีละหนึ่งตัวโน้ต)ซึ่งโปรแกรมจะรอรับเสียงและเขียนโน้ตไปเรื่อยๆจนกว่าจะกดปุ่มหยุดโปรแกรมก็จะหยุดการทำงาน และสามารถนำโน้ตที่เขียนเหล่านั้นปริ้นออกมาดูโน้ตได้ และสามารถบันทึกเสียงที่เล่นเข้าไปแล้วเรียกออกมาฟังได้

1.4.3 ข้อจำกัดและขอบเขตของโครงงานนี้

- สามารถตั้งเสียงเครื่องดนตรีได้บางชนิดเท่านั้น
- สามารถเขียนโน้ตลงบรรทัดห้าเส้นได้ทีละหนึ่งตัวโน้ตเท่านั้น
- เครื่องดนตรีที่นำมาใช้ได้ต้องมี line out ที่จะมาต่อเข้ากับฮาร์ดแวร์เท่านั้น

1.5 วิธีการดำเนินงาน

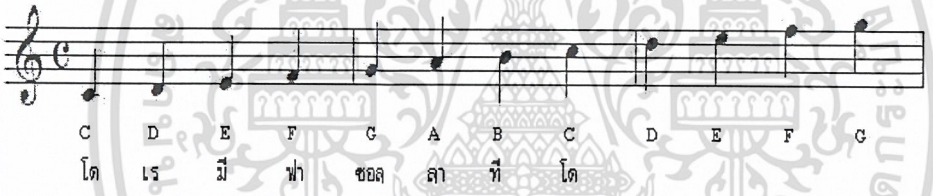
การดำเนินงานสำหรับโครงงานนี้นั้นจะแบ่งออกเป็น 3 ช่วงด้วยกันคือช่วงแรกจะเป็นช่วงของขั้นเตรียมการ ซึ่งในช่วงนี้จะทำการกำหนดเป้าหมายและขอบเขตของโครงงาน รวมทั้งทำการศึกษาเนื้อหาที่จำเป็นและเกี่ยวข้องกับโครงงานอันได้แก่ รายละเอียดเกี่ยวกับโน้ตดนตรี, ไฟล์มิดี้, วงจรกรองความถี่รูปแบบต่างๆ จากนั้นก็จะเข้าสู่ช่วงที่สองนั่นก็คือ ช่วงของขั้นดำเนินงานในช่วงนี้จะทำการออกแบบระบบและออกแบบฮาร์ดแวร์ทั้งหมดของโครงงาน โดยถูกแบ่งออกเป็น 2 ส่วนด้วยกันคือ ส่วนของการจัดการความถี่ที่รับเข้ามา และส่วนของการประมวลผลว่าความถี่ที่เข้ามานั้นเป็นโน้ตอะไร มีความเร็วเท่าใด เมื่อทำการสร้างทั้ง 2 ส่วนเสร็จสิ้นก็จะนำมารวมเข้าด้วยกันเพื่อให้สามารถทำงานร่วมกันได้ ซึ่งช่วงการดำเนินงานนี้ถือว่าเป็นช่วงที่มีความสำคัญเป็นอย่างมาก ดังนั้นจึงต้องใช้ความละเอียดและรอบคอบ อีกทั้งยังต้องหมั่นตรวจสอบอยู่เป็นระยะๆ ว่าการดำเนินงานที่ผ่านมานั้นตรงตามเป้าหมายและขอบเขตที่กำหนดหรือไม่ เพราะไม่เช่นนั้นอาจส่งผลให้ไม่เสร็จสิ้นทันเป้าหมายที่ได้วางไว้ ดังนั้นเมื่อทุกอย่างเป็นไปตามที่เราวางไว้ก็จะเข้าสู่ช่วงสุดท้ายนั้นก็คือ ช่วงของขั้นการประเมินผล ซึ่งแน่นอนว่าหากเรามีการตรวจสอบเป็นระยะๆในขั้นการดำเนินงานแล้ว โครงงานที่จัดทำย่อมเป็นไปตามสิ่งที่เราได้วางเป้าหมายไว้อย่างแน่นอน เราก็ตรวจสอบโปรแกรมขั้นสุดท้ายแล้วทำการสรุปผลการดำเนินงานทั้งหมดว่าที่ผ่านมานั่นเอง



รูปที่ 2-1 แสดงแผนภูมิกระจาย และการเทียบอัตราตัวโน้ต

2.3 ระดับเสียง และชื่อทางดนตรี (Pitches and Musical names)

เสียงดนตรีนั้นนอกจากจะมีความยาวนานแล้ว จะต้องมีระดับเสียง สูง – ต่ำ ด้วยโดยจัดเรียงจากเสียงงต่ำไปหาสูง 7 เสียง ดังนี้ C D E F G A B โดยจัดเป็นชุด เรียงจากต่ำสุดไปหาสูงสุด



รูปที่ 2-2 แสดงระดับเสียง และชื่อทางดนตรี

เมื่อเวลาที่เราเรียกชื่อที่มีระดับเสียงต่ำหรือสูงกว่าที่กำหนดไว้ ก็ให้เรียกชื่อซ้ำอีกทบหนึ่งไปเรื่อยๆ จะเรียกเสียงที่มีชื่อเรียกเหมือนกัน แต่ระดับเสียงต่างกัน ว่ามีระยะคู่แปด หรือ Octave

รูปที่ 2-3 แสดงระยะคู่แปด

จากรูปที่ 2-3 นั้นในการเทียบระดับเสียงสูงต่ำนั้น C D E F G A B ซึ่งมีจุดอยู่ด้านล่างจะมีเสียงที่ต่ำกว่า C D E F G A B และชุดสุดท้ายชุดที่มีจุดอยู่ด้านบนจะมีเสียงสูงกว่าในสองชุดแรกที่กล่าวมา C D E F G A B

2.4 บรรทัด 5 เส้น (Staff)

บรรทัด 5 เส้นจะเป็นเส้นบรรทัดที่ใช้ในการบันทึกโน้ตดังรูปที่ 2-3(a) โดยเส้นด้านล่างใช้ในการบันทึกโน้ตเสียงต่ำ และเส้นที่สูงขึ้นมาใช้ในการบันทึกระดับเสียงโน้ตที่สูงขึ้นไปเป็นลำดับ การบันทึกระดับเสียงต่างๆในบรรทัด 5 เส้น เราสามารถทำได้ทั้งเขียนคาบเส้นดังรูปที่ 2-4(a) และเขียนในช่อง ดังรูปที่ 2-4(b)

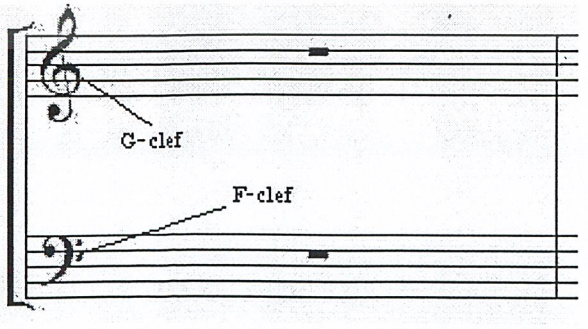
รูปที่ 2-4 แสดงบรรทัด 5 เส้น

(a) การบันทึกคาบเส้น (b) การบันในช่อง

รูปที่ 2-5 แสดงการบันทึกตัวโน้ตลงบนบรรทัด 5 เส้น

2.5 กุญแจ (Clef)

เป็นสัญลักษณ์ที่ใช้เป็นตัวกุญแจในการอ่านโน้ตต่างๆที่ถูกบันทึกลงบนบรรทัด 5 เส้น ถ้าไม่มีกุญแจบน บรรทัด 5 เส้น เราจะไม่สามารถอ่านระดับเสียงโน้ตออกมาได้ แต่จะทราบเพียงความแตกต่างของระดับเสียงของตัวโน้ตที่อยู่สูงกว่า หรือต่ำกว่ากุญแจที่เรานิยมใช้กันโดยทั่วไปมีอยู่ 2 ชนิดด้วยกันคือ กุญแจซอล (G Clef) และ กุญแจฟา (F clef) ดังแสดงในรูปที่ 2-6



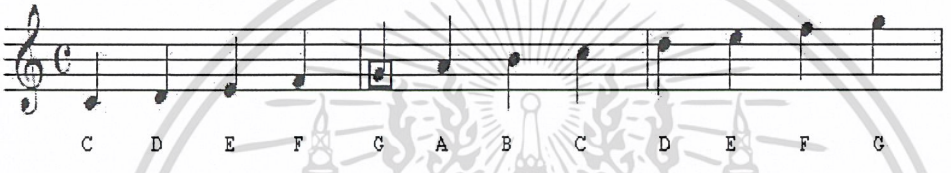
รูปที่ 2-6 แสดงลักษณะของกุญแจโน้ตดนตรี

2.5.1 กุญแจซอล

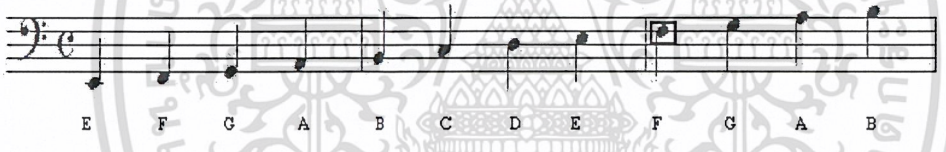
เป็นกุญแจที่ใช้ในระดับเสียงปานกลาง ถึง ระดับเสียงสูงมากที่สุด เช่น Guitar เป็นต้น หัวของกุญแจจะคาบอยู่บนเส้นบรรทัดที่ 2 ของบรรทัด 5 เส้น ดังนั้นตัวโน้ตที่บรรทึกคาบเส้นเดียวกับหัวของกุญแจประจำหลักนี้ก็คือ “ซอล” (G) ตามชนิดและชื่อของกุญแจนั้น ดังแสดงในรูปที่ 2-7(a)

2.5.2 กุญแจฟา

เป็นกุญแจที่ใช้สำหรับเสียงต่ำ ถึง ระดับเสียงต่ำมาก เช่น Bass เป็นต้น หัวของกุญแจจะคาบอยู่บนเส้นบรรทัดที่ 4 ของบรรทัด 5 เส้น ดังนั่นตัวโน้ตที่บรรทึกคาบเส้นเดียวกับหัวของกุญแจประจำหลักนี้ก็คือ “ ฟา ” (F) ตามชนิดและชื่อของกุญแจนั้น ดังแสดงในรูปที่ 2-7(b)



(a) กุญแจซอล

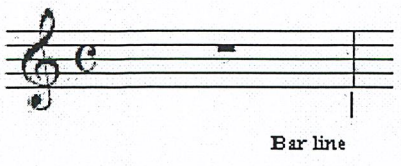


(b) กุญแจฟา

รูปที่ 2-7 แสดงโน้ตที่บันทึกลงบนกุญแจซอล และกุญแจฟา

2.6 เส้นกั้นห้องเพลง (Bar Line)

เส้นกั้นห้องเพลง จะเป็นเส้นสำหรับแบ่งกั้นห้องเพลง ซึ่งเราจะทำการกั้นห้องเพลงเมื่อไหร่นั้นขึ้นอยู่กับTime signatureที่กำหนดอีกทีหนึ่ง

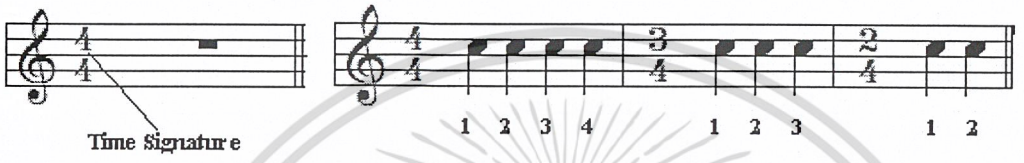


รูปที่ 2-8 แสดงเส้นกั้นห้องเพลง

2.7 เครื่องหมายกำหนดจังหวะ (Time Signature)

เครื่องหมายกำหนดจังหวะนั้นโดยปกติแล้ว เราจะพบเครื่องหมายนี้ได้หลังกุญแจ มีลักษณะการเขียนคล้ายตัวเลขในแบบเศษส่วน ดังแสดงในรูปที่ 2-9

ตอนเริ่มต้นของบทเพลงทุกเพลง จะมีเลข 2 ตัว ตัวหนึ่งอยู่ข้างบน อีกตัวหนึ่งอยู่ข้างล่าง ตัวเลขดังกล่าวคือ เครื่องหมายกำหนดจังหวะ ตัวเลขที่อยู่ด้านบนจะบอกจำนวนของจังหวะในแต่ละห้อง ส่วนตัวเลขที่อยู่ด้านล่าง จะบอกชนิดของตัวโน้ตที่มีค่าเท่ากับ 1 จังหวะ ยกตัวอย่างเช่น 4/4 หมายความว่า มี 4 จังหวะในแต่ละห้อง และเลข 4 ด้านล่างหมายความว่า โน้ตตัวดำ (QUARTER NOTE) มีค่าเท่ากับ 1 จังหวะ



รูปที่ 2-9 แสดงสัญลักษณ์เครื่องหมายกำหนดจังหวะ

2.8 ตัวหยุด (Rests)

ตัวหยุด เป็นเครื่องหมายที่ใช้เพื่อให้เงียบ หรือหยุดเสียงชั่วขณะ ซึ่งเสียงนั้นจะหยุดสั้นหรือนานเท่าใด ขึ้นอยู่กับ อัตราจังหวะและลักษณะของตัวหยุด ซึ่งลักษณะและอัตราตัวหยุดนั้น จะเทียบตรงกับตัวโน้ต แต่ตัวหยุดจะใช้ในทางตรงกันข้ามกับตัวโน้ตเท่านั้น ดังแสดงในตารางที่ 2-2

ตัวหยุด บันทึกไว้เพื่อให้เสียงเงียบเวลาบรรเลง

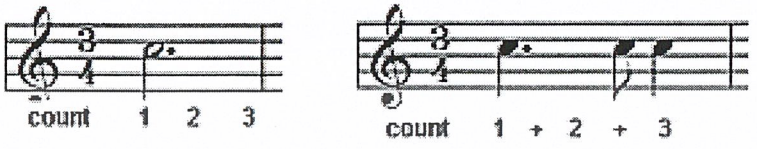
ตัวโน้ต บันทึกไว้เพื่อให้เกิดเสียงเวลาบรรเลง

	ตัวหยุดตัวกลม มีค่าหยุดเสียงเท่ากับโน้ตตัวกลม (หยุด 4 จังหวะ)
	ตัวหยุดตัวขาว มีค่าหยุดเสียงเท่ากับโน้ตตัวขาว (หยุด 2 จังหวะ)
	ตัวหยุดตัวดำ มีค่าหยุดเสียงเท่ากับโน้ตตัวดำ (หยุด 1 จังหวะ)
	ตัวหยุดตัวเขบ็ตหนึ่งชั้น มีค่าหยุดเสียงเท่ากับโน้ตตัวเขบ็ตหนึ่งชั้น (หยุด1/2 จังหวะ)

***ตารางที่ 2-2* แสดงประเภทของตัวหยุด**

2.9 โน้ตประจุด (Dotted Note)

สัญลักษณ์โน้ตประจุดนี้ หากเราอ่านไปเจอในบทเพลงก็จะมีความหมายว่า ให้เราเพิ่มค่าโน้ตเข้าไปครึ่งหนึ่ง (1/2 จังหวะ) ของโน้ตที่มัน dotted อยู่ เช่น ถ้าเราเห็น half note dotted (ตัวขาวมี dotted) ก็มีความหมายว่า ให้เราทำการเพิ่มค่าโน้ตนี้ไป 1 จังหวะ เพราะว่าลำพังตัวขาวปกติจะมีค่า 2 จังหวะ ครึ่งหนึ่งของ 2 ก็คือ 1 ฉะนั้น เราจึงเพิ่มค่าโน้ตตัวขาวไปอีก 1 จังหวะ ก็จะมีค่าเป็น 3 จังหวะนั่นเอง



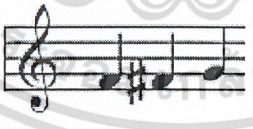
รูปที่ 2-10 แสดงตัวอย่างการใช้ Dotted Note

2.10 เครื่องหมายแปลงเสียง (Accidentals)

เครื่องหมายแปลงเสียง คือเครื่องหมายที่ใช้บันทึกขวางไว้หน้าตัวโน้ตที่ต้องการจะให้ระดับเสียงของตัวโน้ตนั้นๆ เปลี่ยนแปลงไปจากปกติ ไม่ว่าจะทำให้สูงขึ้นต่ำลง หรือกลับมาเท่าเดิมมีอยู่ 3 ชนิดด้วยกันคือ ช้าป, แฟล็ต และเนเจอรัล

2.10.1 การใช้เครื่องหมาย ชาร์ป (♯)

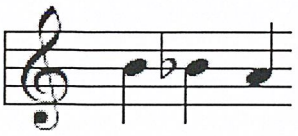
ถ้าต้องการจะให้เสียงเดิมเลื่อนสูงกว่าเดิมอีกครึ่งเสียง (1/2 จังหวะ) ให้ใช้เครื่องหมาย "#" บันทึกไว้หน้าตัวโน้ตที่ต้องการนั้น ในภาพ จะมีโน้ต F , F♯,G ซึ่ง F Sharp จะมีเสียงสูงกว่าF ครึ่งเสียง และมีเสียงต่ำกว่า G ครึ่งเสียง



รูปที่ 2-11 แสดงเครื่องหมาย Sharp

2.10.2 การใช้เครื่องหมาย แฟลท (♭)

ถ้าต้องการให้เสียงเดิมนั้นเลื่อนต่ำลงมาอีกครึ่งเสียง (1/2 จังหวะ) ให้ใช้เครื่องหมาย ♭ บันทึกไว้หน้าตัวโน้ตที่ต้องการนั้น ในภาพ จะมีโน้ต B, B♭, A ซึ่ง B Flat จะมีเสียงสูงกว่า B ครึ่งเสียง และมีเสียงต่ำกว่า A ครึ่งเสียง



รูปที่ 2-12 แสดงเครื่องหมาย Flat

2.10.3 การใช้เครื่องหมาย เนเจอรัล (♮)

ถ้าต้องการให้เสียงของตัวโน้ตที่ถูกเครื่องหมายแปลงเสียงบังคับไว้แต่แรกนั้นให้กลับสภาพตามเดิม ให้ใช้เครื่องหมาย ♮ บันทึกขวางหน้าตัวโน้ต จะมีผลทำให้เสียงของตัวโน้ตนั้นกลับสภาพปรกติ

บทที่ 3
ไฟล์มิดี้

3.1 บทนำ

MIDI มาจากคำว่า Musical Instrument Digital Interface ซึ่งเป็นมาตรฐานในการสื่อสารข้อมูลแบบอนุกรม เพื่อใช้สำหรับสื่อสารระหว่างเครื่องดนตรีอิเล็กทรอนิกส์กัน หรือกับเครื่องคอมพิวเตอร์โดยข้อมูลที่ส่งไปนั้นเป็นข้อมูลแบบดิจิตอล ทำให้การแต่งเพลงหรือการบันทึกเสียงทำได้ง่ายขึ้นมาก สามารถนำเพลงมาแก้ไขได้สะดวกมากขึ้น

ไฟล์ MIDI ไม่ได้มีการเก็บเสียงดนตรีใดๆไว้เหมือนอย่างเทปเพลงหรือซีดีเพลง ข้อมูลทั้งหมดจะอยู่ในรูปของคำสั่งที่จะไปสั่งเครื่องดนตรีว่า ให้เปล่งเสียงโน้ตตัวใด(Note ON), ด้วยระดับความดังแค่ไหน (Velocity) และคำสั่งอื่นๆ ตามคุณสมบัติเฉพาะของเครื่องดนตรีแต่ละชนิด ด้วยเหตุที่เป็นไฟล์คำสั่งนี่เอง ทำให้มันมีขนาดที่เล็กมากๆ และจากความที่มันเป็นไฟล์คำสั่งแบบดิจิตอลนี่เอง นักคอมพิวเตอร์จึงสามารถนำข้อมูลดิจิตอลนี้มาพัฒนาด้วย จนในที่สุดทั้งคอมพิวเตอร์และเครื่องดนตรีก็สื่อสารกันได้อย่างสมบูรณ์โดยผ่านระบบ MIDIนี่เอง

3.2 รูปแบบของการส่งข้อมูลมิดี้

สำหรับการส่งข้อมูลในระบบ MIDI นั้นจะมีการส่งในลักษณะของ ชุดข้อมูล (Package) โดยที่ข้อมูล MIDI จะมีลักษณะดังนี้

MSB(Most Significant Bit) 1บิต	บิตของข้อมูล 7 บิต

ตารางที่ 3-1 แสดง ชุดข้อมูลของไฟล์ MIDI

- ถ้า MSB มีค่าเป็น 0 แสดงว่า ข้อมูลนี้เป็น ไบต์แสดงข้อมูล (Data Byte)
- ถ้า MSB มีค่าเป็น 1 แสดงว่า ข้อมูลนี้เป็น ไบต์แสดงสถานะ (Status Byte)

ทั้งนี้ในส่วนของไบต์แสดงสถานะนั้นจะสามารถแบ่งออกได้เป็น 2 ประเภทด้วยกันคือ ข้อมูลแสดงแชนแนล (Channel Message) และ ข้อมูลระบบ (System Message)

3.2.1 ข้อมูลแสดงแชนแนล

ไบต์แสดงสถานะประเภท ข้อมูลแสดงแชนแนลนั้นจะส่งข้อมูลตรงไปให้กับ แชนแนลใดแชนแนล หนึ่งในระบบเท่านั้น โดยที่ไบต์แสดงสถานะประเภทนี้ จะใช้บิตที่ 0 ถึงบิตที่ 3 เพื่อบอกถึง

แชนแนลที่ส่งไป ดังนั้นถ้าหากว่าอุปกรณ์ MIDI ที่เป็น Slave ตัวใดมีแชนแนลตรงกับ แชนแนลที่ส่งไป Slave ตัวนั้นก็จะตอบสนองต่อไบต์ข้อมูลที่ตามมา

ข้อมูลแสดงแชนแนล ยังสามารถแบ่งออกได้เป็น 2 ส่วนคือ ข้อมูลเสียง (Voice Message) และข้อมูลแสดงรูปแบบ (Mode Message)

3.2.1.1 ข้อมูลเสียง (Channel Voice Message)

โดยส่วนใหญ่แล้วข้อมูลแสดงแชนแนล จะเป็นประเภทข้อมูลเสียง ซึ่งหน้าที่ของข้อมูลเสียง คือ จะทำการสั่งให้ แชนแนลนั้นทำการเริ่มต้น หรือ หยุดเล่นโน้ตดนตรี เป็นต้น

ข้อมูลเสียงที่ใช้นั้นสามารถแสดงได้ดังตารางที่ 3-2

ข้อมูลเสียง	ความหมาย
Note on	ทำการเริ่มเล่นโน้ตดนตรี
Note off	ทำการหยุดเล่นโน้ตดนตรี
Control change	ควบคุมเกี่ยวกับตัวควบคุมเสียงบางประเภท เช่น ระดับเสียง (tone), ความดังของเสียง (volume)
Pitch Bend Change	เกี่ยวข้องกับการเอื้อนเสียงของแชนแนล
Program Change	ทำการเลือกเสียงของเครื่องดนตรีที่จะใช้ในแชนแนลนั้นๆ
Polyphonic Key	ข้อมูลชุดนี้จะถูกส่งหลังจากที่มีการเปลี่ยนแปลงความดัน เฉพาะ คีย์ ที่เราเล่นโน้ตดนตรีไปแล้ว
Channel Key Message	ข้อมูลชุดนี้จะเกี่ยวข้องกับการเปลี่ยนแปลง ความดัน ของ คีย์ เช่นเดียวกับ Polyphonic key แต่จะมีผลต่อเสียงทั้ง แชนแนล ไม่เฉพาะเจาะจง คีย์แบบ Polyphonic key

ตารางที่ 3-2 แสดงชนิดและความหมายของข้อมูลเสียง

3.2.1.2 ข้อมูลแสดงรูปแบบ (Channel Mode Message)

เป็นตัวควบคุมรูปแบบการทำงานของแต่ละ แชนแนล โดยจะทำการพิจารณาว่าจะให้แชนแนลนั้นจัดการกับข้อมูลเสียงที่ตามมาอย่างไร

ข้อมูลแสดงรูปแบบที่ใช้นั้น มีดังต่อไปนี้

1. Poly/Mono คือ การกำหนดให้อุปกรณ์ MIDI นั้นทำการเปล่งเสียงโน้ตดนตรีออกมาในแบบ Poly หรือ Mono โดยที่

- Mono การเปล่งเสียงนั้นจะสามารถ เล่นโน้ตดนตรีได้ทีละ 1 โน้ตเท่านั้น
- Poly การเปล่งเสียงนั้นจะสามารถเล่นโน้ตดนตรีได้ทีละหลายๆ โน้ตพร้อมกัน

2. Omni On/Off เป็นการบังคับให้ แชนแนล นั้นตอบสนองเฉพาะข้อมูลที่ตรงกับ แชนแนล ของตน หรือตอบสนองในทุกๆ แชนแนล

- ถ้า Omni On จะเป็นการตอบสนองในทุกๆ แชนแนล
- ถ้า Omni Off จะเป็นการตอบสนองเฉพาะข้อมูลที่ตรงกับ แชนแนล ของตน

โดยปกติแล้ว ถ้านำอุปกรณ์ MIDI มาต่อกันให้เป็นระบบแล้ว รูปแบบที่จะนำมาใช้งานก็คือ Poly: Omni Off

3.2.2 ข้อมูลระบบ

ข้อมูลประเภทนี้จะทำการส่งออกไปโดยไม่เจาะจงเฉพาะแชนแนลใด แชนแนลหนึ่งในระบบ แต่ในทุกๆแชนแนลนั้นจะต้องตอบสนองต่อข้อมูลระบบนี้ ซึ่งข้อมูลระบบนั้นสามารถแบ่งได้เป็น

3.2.2.1 Common Message

ข้อมูลชุดนี้จะเกี่ยวข้องกับการเลือกเพลง การเลือกจังหวะ การเลือกตำแหน่งของเพลงที่จะเล่น และการบังคับให้อุปกรณ์ อะนาล็อก (Analog) ปรับแต่งระบบเสียงของตัวเอง (tune request)

3.2.2.2 System Real Time

ข้อมูลชุดนี้จะเกี่ยวข้องโดยตรงกับการสั่งเริ่มเล่นเพลง (Start), การหยุดเพลง (Stop) และข้อมูลที่สำคัญอื่นๆ เช่น ซิสเต็มคล็อก (System clock) จะใช้เกี่ยวกับระดับความเร็วของจังหวะเพลงว่าจะให้เร็วหรือช้า ยิ่งความถี่ในการส่งซิสเต็มคล็อกมาก เพลงก็จะยิ่งเล่นเร็วขึ้น

3.3 โครงสร้างไฟล์ MIDI

ไฟล์ MIDI จะถูกสร้างให้อยู่ในรูปแบบโครงสร้างที่เรียกว่า *"Chunk"* โดยในแต่ละ Chunk จะมีโครงสร้างภายในดังแสดงในตารางที่ 3-3

ชนิด	ความยาว	ข้อมูล
4 ไบต์	4 ไบต์	X ไบต์

ตารางที่ 3-3แสดงโครงสร้างของข้อมูลประเภท Chunk

- *ชนิดของ Chunk* เป็นส่วนที่ใช้บอกถึงประเภทของ Chunk นั้นๆ มีขนาดในการเก็บข้อมูลเท่ากับ 4 ไบต์ แบ่งได้เป็น 2 ประเภทคือ Header Chunk และ Track Chunk
- *ความยาวของ Chunk* เป็นส่วนที่ใช้บอกความยาวของข้อมูลที่ตามหลังมาว่า มีขนาดเท่าใด โดยมีขนาดในการเก็บข้อมูลเท่ากับ 4 ไบต์

- *ข้อมูลของ Chunk* เป็นส่วนที่เก็บข้อมูลต่างๆ ของ Chunk โดยจะสามารถเก็บข้อมูลได้เท่าใดนั้น ก็ขึ้นอยู่กับค่าประกาศไว้ในส่วนของ ความยาวของChunk

รูปแบบของไฟล์ MIDI นั้นจะประกอบไปด้วยหนึ่ง Header Chunk ตามด้วย Track Chunk จำนวนตั้งแต่หนึ่ง Track Chunk ขึ้นไป

	Chunk				
	ประเภท	**ความยาว**	**ข้อมูล**		
ไฟล์ **MIDI**	MThd	6 ไบต์	*<format>*	*<tracks>*	*<division>*
	MTrk	*<length>*	*<Delta-Time><event>…*		
	.				
	.				
	MTrk	*<length>*	*<Delta-Time><event>…*		

ตารางที่ 3-4 แสดงโครงสร้างข้อมูลของไฟล์ MIDI

3.3.1 Header Chunk

Header Chunk จะเป็น ส่วนเริ่มต้นของไฟล์ MIDI เสมอและจะมีเพียง 1 Header Chunk เท่านั้น ซึ่งจะมีส่วนประกอบดังนี้

Header Chunk				
ประเภท	ความยาว	ข้อมูล		
4 ไบต์ แอสกี(ASCII)	4 ไบต์ (32 บิตไบนารี)	<length (6 ไบต์)>		
		16 บิต	16 บิต	16 บิต
MThd	<length>	<format>	<tracks>	<division>

ตารางที่ 3-5 แสดงโครงสร้างของข้อมูลส่วน Header Chunk

- *ส่วนบอกประเภทของ Chunk* เป็นตัวอักษร 4 ตัว ที่บอกว่า Chunk นี้เป็น Header Chunk โดยจะมีค่าเป็น “MThd” เสมอ ซึ่งจะมีขนาด 4 ไบต์ ซึ่งในรหัสแอสกี (ASCII) จะมีค่าเป็น ‘M = 4D’ ‘T = 54’ ‘h = 68’ ‘d = 64’

- *ความยาวของ Header Chunk* เป็นส่วนที่บอกความยาวของข้อมูลใน Header Chunk ว่ามีขนาดเท่าใด ซึ่งข้อมูลของ Header จะมีขนาดเป็น 6 ไบต์เสมอ

- *ข้อมูลของ Header Chunk* ประกอบไปด้วย 3 ส่วนย่อยๆ คือ

  - ฟอร์แมต (Format) เป็นส่วนที่บอกชนิด แทร็ก (Track) ของ MIDI มีขนาดในการเก็บข้อมูล 2 ไบต์โดย ฟอร์แมตสามารถแบ่งได้เป็น 3 ชนิด คือ

    1. Single Track จะมีค่าในส่วนของ ฟอร์แมต เป็น 00 00
    2. Multiple Track แบบ Synchronous จะมีค่าในส่วนของ ฟอร์แมต เป็น 00 01
    3. Multiple Track แบบ Asynchronous จะมีค่าในส่วนของ ฟอร์แมต เป็น 00 02

  - แทร็ก (Track) เป็นส่วนที่ใช้บอกจำนวนของ Track Chunk ในไฟล์ MIDI มีขนาดในการเก็บข้อมูลเท่ากับ 2 ไบต์

  - Division เป็นส่วนที่บอกค่าเวลาของ Delta-Time ต่อโน้ตตัวดำ (Quarter note) มีขนาดในการเก็บข้อมูลเท่ากับ 2 ไบต์

3.3.2 Track Chunk

Track Chunk คือ ส่วนที่ใช้เก็บข้อมูลคำสั่งต่างๆ ของ ไฟล์ MIDI ในไฟล์ MIDI นั้น ประกอบไปด้วย Track Chunk จำนวนตั้งแต่ 1 Track Chunk ขึ้นไป ซึ่งรายละเอียดของส่วนประกอบของ Track Chunk นั้นมีดังนี้

Track Chunk		
ประเภท	**ความยาว**	**ข้อมูล**
4 ไบต์ แอสกี(ASCII)	4 ไบต์ (32 บิตไบนารี)	<length ไบต์> (ข้อมูลชนิดไบนารี)
MTrk	<length>	<Delta-Time><event>…

ตารางที่ 3-6 แสดงโครงสร้างของข้อมูลส่วน Track Chunk

- *ส่วนบอกประเภทของ Chunk* เป็นตัวอักษร 4 ตัวที่บอกว่า Chunk นี้เป็น Track Chunk โดยจะมีค่าเป็น "MTrk" มีขนาด 4 ไบต์ ซึ่งในรหัสแอสกี จะมีค่าเป็น 'M = 4d' 'T = 54' 'r =72' 'k=6B'

- *ความยาวของ Track Chunk* เป็นส่วนที่บอกความยาวของ ข้อมูลใน Track Chunk ว่ามีความยาวเท่าใด โดยมีขนาดในการเก็บข้อมูลเท่ากับ 4 ไบต์

- *ข้อมูลของ Track Chunk* จะประกอบขึ้นด้วย Track Event ตั้งแต่ 1 Track Event ขึ้นไป โดยที่แต่ละ Track Event จะประกอบไปด้วยส่วนย่อยๆ 2 ส่วน คือ

  1. Delta-Time เป็นส่วนที่ใช้สำหรับบอกค่าเวลาของเหตุการณ์นั้นๆ ซึ่งมีขนาดในการเก็บข้อมูลเท่ากับ 1-2 Bytes

2. Event เป็นส่วนของคำสั่งในแต่ละเหตุการณ์ที่สั่งให้ MIDI ทำงาน ขนาดของ Event นั้นจะขึ้นอยู่กับคำสั่งนั้นๆ Event สามารถแบ่งได้ดังนี้ คือ
   - MIDI Event คือ ส่วนที่บอกสถานะที่กำลังทำงาน (Running Status) ของเสียง และรูปแบบใน Channel ที่ต้องการ
   - Meta Event คือ ส่วนที่บอกข้อมูลระบบ จะไม่มีการกำหนด Channel และ ทุก Channel จะต้องตอบสนองต่อข้อมูลของระบบนี้

3.3.2.1 MIDI Event

MIDI Event นั้นสามารถแบ่งได้เป็น 2 ส่วนย่อย ๆ คือ

ส่วนของคำสั่ง	ส่วนของข้อมูล

ตารางที่ 3-7 แสดงโครงสร้างของ MIDI Event

- *ส่วนของคำสั่ง* จะมีขนาดเท่ากับ 1 ไบต์โดยมี 4 บิตแรก เป็นส่วนบอกคำสั่งที่จะใช้ในการทำงาน และ 4 บิตหลัง คือ ค่าของหมายเลขแชนแนล (Channel Number) ที่จะกำหนดให้ทำงาน
- *ส่วนของข้อมูล* จะเป็นส่วนบอกรายละเอียดของข้อมูลที่จะรับคำสั่งไปทำงาน เช่น หมายเลขโน้ต น้ำหนักในการเล่น (Velocity) เป็นต้น

คำสั่ง	ข้อมูล	รายละเอียด
8X	**NN VV** NN คือ หมายเลขโน้ต VV คือ น้ำหนักในการเล่น	ช่วงเลิกเล่นโน้ต (Note Off)
9X	**NN VV** NN คือ หมายเลขโน้ต VV คือ น้ำหนักในการเล่น	ช่วงเล่นโน้ต (Note On)
AX	**NN VV** NN คือ หมายเลขโน้ต VV คือ น้ำหนักในการเล่น	เพิ่มความดันของคีย์หลังจากที่กดคีย์นั้นไปแล้ว (Polyphonic Key Pressure)
BX	**CC VV** CC คือ หมายเลขตัวควบคุม VV คือ หมายเลขตัวควบคุมใหม่	เปลี่ยนตัวควบคุมที่ใช้ในการปรับแต่งเสียง

CX	**PP** PP คือ หมายเลขเครื่องดนตรี	ทำการกำหนดชนิดของเครื่องดนตรี
DX	**CC** CC คือ หมายเลขแชนแนล	กำหนดความดันเฉลี่ยของทุกๆ คีย์ของแชนแนล
EX	**BB TT** BB คือ Least Significant Bit TT คือ Most Significant Bit	การเอื้อนเสียง (Pitch Wheel Change)

ตารางที่ 3-8 แสดงคำสั่งและความหมายต่างๆ ของ MIDI Event

3.3.2.2 Meta Event

Meta Event จะเป็นข้อมูลจำพวก ชื่อแทร็ก, ข้อความเกี่ยวกับลิขสิทธิ์ ซึ่งโดยปกติแล้วนั้นจะไม่ถูกแสดงผลออกมา ในขณะที่คำสั่งต่างๆจะถูกประมวลผล แต่ก็ยังคงเป็นส่วนประกอบที่มีประโยชน์ต่อไฟล์ MIDIอยู่ รูปแบบทั่วๆไปของ Meta Events มีดังนี้

FF <Type> <Length> <Data>

<Type>

เป็น Single byte ใช้เพื่อบอกประเภทของ Meta event ซึ่งจะมีค่าตั้งแต่ 00 – 7F

<Length>

ใช้เพื่อบอกขนาดของข้อมูลที่ตามมา

<Data>

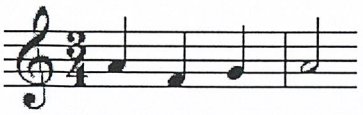
ข้อมูลต่างๆที่จะเก็บไว้ใน Meta Events ซึ่งจะมีขนาดตั้งแต่ 0 ไบต์ขึ้นไป

FF 00 02 SS SS	**Sequence Number**	
	ทำการตั้งลำดับหมายเลขของแทร็ก	
	SS SS	เป็นเลขฐานสองขนาด 16 บิต
FF 01 <length> <text>	**Text Event**	
	ข้อความเกี่ยวกับชื่อของแทร็ก	
	<length>	ค่าความยาว <text>
	<text>	ค่าแอสกีขนาดตามที่ประกาศไว้
FF 02 <length> <text>	**Copyright Notice**	
	ข้อความเกี่ยวกับลิขสิทธิ์	
FF 03 <length> <text>	**Track Name**	
	ข้อความบอกชนิดของ Track	

FF 04 <length> <text>	**Instrument Name**	
	ข้อความบอกชนิดของเครื่องดนตรีที่ใช้เล่น	
FF 05 <length> <text>	**Lyric**	
	ข้อความบอกทำนองของเพลง	
FF 06 <length> <text>	**Marker**	
	ข้อความระบุตำแหน่งที่จะเล่น	
FF 07 <length> <text>	**Cue Point**	
	ข้อความแนะนำเหตุการณ์ของเพลง	
FF 2F 00	**End of Track**	
	เป็นการบอกว่าจบ Track แล้ว	
FF 51 03 TT TT TT	**Set Tempo**	
	เป็นการกำหนดความเร็วของเพลง (Tempo) ถ้าไม่ได้ทำการตั้งไว้ค่ามาตรฐานจะเป็น 120 บีตต่อนาที ซึ่ง TT TT TT = 50 00 00	
	TT TT TT	ค่าจังหวะเป็นตัวเลขฐานสอง 24 บิต โดยมีหน่วยเป็น ไมโครวินาทีต่อโน้ตตัวดำ
FF 58 04 NN DD CC BB	**Time Signature**	
	ทำการกำหนดจังหวะของห้องเพลง	
	nn	ส่วนเศษ (Time signature numerator)
	dd	ตัวหาร (Time signature denominator)
	cc	MIDI Clocks ต่อ metronome tick
	bb	จำนวนของ 1/32 notes ต่อ 24 MIDI clocks (ค่ามาตรฐานคือ 8)
FF 59 02 SF MI	**Key Signature**	
	กำหนดคีย์ของเสียงเพลง 0 ใช้แทน C, จำนวนเต็มลบแทน เครื่องหมายแฟลตและ จำนวนเต็มบวกแทนเครื่องหมายชาร์ป	
	SF	จำนวนของ ชาร์ป หรือ แฟลต -7 = 7 แฟลต, 0 = C, +7 = 7 ชาร์ป
	MI	0 = Major key 1 = Minor key

ตารางที่ 3-9 แสดงคำสั่งและความหมายต่างๆของ Meta Event

3.4 รูปแบบตัวอย่างไฟล์ MIDI



รูปที่ 3-1 แสดงภาพโน้ตดนตรีที่จะทำการแปลงความให้อยู่ในรูปของไฟล์ MIDI

เมื่อทำการแปลงให้อยู่ในรูปของไฟล์ MIDI จะได้ดังนี้ (แสดงเป็นเลขฐาน 16)

```
4D 54 68 64 00 00 00 06  00 01 00 02 01 80 4D
54 72 6B 00 00 00 10 00  FF 58 04 03 02 24 08
00 FF 51 03 07 A1 20 00  4D 54 72 6B 00 00 00
2B 00 C0 18 00 90 45 60  83 00 80 45 00 00 90
41 60 83 00 80 41 00 00  90 43 60 83 00 80 43
00 00 90 45 60 86 00 80  45 00 00 FF 2F 00
```

ข้อมูลของส่วน Header Chunk

4D 54 68 64	→	Chunk Type ในรหัส ASCII มีค่าเป็น MThd (Header Chunk)
00 00 00 06	→	ความยาวของ Header เท่ากับ 6 Bytes
00 01	→	ฟอร์แมตของ MIDI นี้คือ Format 1 (Multiple Track)
00 02	→	ไฟล์ MIDI นี้มีจำนวน 2 Track
01 80	→	กำหนดความยาวเสียงของโน้ตตัวดำเท่ากับ 384 (ฐาน 10)

ข้อมูลของส่วน Track Chunk แรก

4D 54 72 6B	→	Chunk Type ในรหัส ASCII มีค่าเป็น MTrk (Track Chunk)
00 00 00 10	→	ความยาวของ Track นี้เท่ากับ 16 Bytes
FF 58 04 03 02 24 08	→	ทำการกำหนดจังหวะของห้องเพลง
FF 51 03 07 A1 20	→	ทำการกำหนดความเร็วของการเล่นต่อโน้ตตัวดำ 1 ตัว

ข้อมูลของส่วน Track Chunk ที่สอง

4D 54 72 6B	→	Chunk Type ในรหัส ASCII มีค่าเป็น MTrk (Track Chunk)
00 00 00 2B	→	ความยาวข้อมูล Track นี้เท่ากับ 43 Bytes
C0 18	→	C0 คือ การตั้งเครื่องดนตรีที่ Channel 1
		18 คือ หมายเลขเครื่องดนตรี มีค่าเป็น 24 ในเลขฐาน 10
90 45 60	→	90 คือ ให้ Channel 1 ทำการเริ่มเล่นโน้ต
		45 หมายถึง โน้ตที่จะเล่นคือ A
		60 คือ ความหนักของเสียงที่เล่น

83 00 → ค่าเวลาความยาวของเสียงของโน้ตตัวดำ

80 45 → 80 คือให้ Channel 1 ทำการหยุดเล่นโน้ต
45 คือ โน้ตที่จะหยุดเล่นคือ A

90 41 60 → 90 คือ ให้ Channel 1 ทำการเริ่มเล่นโน้ต
41 หมายถึง โน้ตที่จะเล่นคือ F
60 คือ ความหนักของเสียงที่เล่น

83 00 → ค่าเวลาความยาวของเสียงของโน้ตตัวดำ

80 41 → 80 คือให้ Channel 1 ทำการหยุดเล่นโน้ต
41 คือ โน้ตที่จะหยุดเล่นคือ F

90 43 60 → 90 คือ ให้ Channel 1 ทำการเริ่มเล่นโน้ต
43 หมายถึง โน้ตที่จะเล่นคือ G
60 คือ ความหนักของเสียงที่เล่น

83 00 → ค่าเวลาความยาวของเสียงของโน้ตตัวดำ

80 43 → 80 คือให้ Channel 1 ทำการหยุดเล่นโน้ต
43 คือ โน้ตที่จะหยุดเล่นคือ G

90 45 60 → 90 คือ ให้ Channel 1 ทำการเริ่มเล่นโน้ต
45 หมายถึง โน้ตที่จะเล่นคือ A
60 คือ ความหนักของเสียงที่เล่น

86 00 → ค่าเวลาความยาวของเสียงของโน้ตตัวขาว

80 45 → 80 คือให้ Channel 1 ทำการหยุดเล่นโน้ต
45 คือ โน้ตที่จะหยุดเล่นคือ A

FF 2F 0 → จบ Track

3.5 ความถี่ของตัวโน้ตดนตรี

ชื่อตัวโน้ต	เริ่มต้น(Hz)	มาตรฐาน(Hz)	สิ้นสุด(Hz)
A		27.5	28.3175
A#/Bb	28.32	29.135	30.0015
B	30.1	30.868	31.7855
-	-	-	-
C	31.8	32.703	33.6755
C#/Db	33.7	34.648	35.678
D	35.7	36.708	37.7995
D#/Eb	37.8	38.891	40.047
E	40.1	41.203	42.4285

F	42.5	43.654	44.9515
F#/Gb	45.0	46.249	47.624
G	47.7	48.999	50.456
G#/Ab	50.5	51.913	53.4565
A	53.5	55.0	56.635
A#Bb	56.7	58.270	60.0025
B	60.1	61.735	63.5705
-	-	-	-
C	63.6	65.406	67.351
C#/Db	67.4	69.296	71.356
D	71.4	73.416	75.599
D#/Eb	75.6	77.782	80.0945
E	80.1	82.407	84.857
F	84.9	87.307	89.903
F#/Gb	90.0	92.499	95.249
G	95.3	97.999	100.9145
G#Ab	101.0	103.83	106.915
A	107.0	110.0	113.27
A#/Bb	113.3	116.54	120.005
B	120.1	123.47	127.14
-	-	-	-
C	127.2	130.81	134.7
C#/Db	134.8	138.59	142.71
D	142.8	146.83	151.195
D#Eb	151.2	155.56	160.185
E	160.19	164.81	169.71
F	169.8	174.61	179.805
F#/Gb	179.9	185.00	190.5
G	190.6	196.00	201.825
G#/Ab	201.9	207.65	213.825
A	213.9	220.00	226.54
A#/Bb	226.6	233.08	240.01
B	240.2	246.94	254.285
-	-	-	-

C	254.30	261.63	269.40
C#/Db	269.41	277.18	285.42
D	285.43	293.66	302.39
D#Eb	302.40	311.13	320.38
E	320.39	329.63	339.43
F	339.44	349.23	359.61
F#/Gb	359.62	369.99	380.99
G	381.00	391.99	403.65
G#/Ab	403.66	415.31	427.65
A	427.66	440.00	453.08
A#/Bb	453.09	466.16	480.02
B	480.03	493.88	508.56
-	-	-	-
C	508.6	523.25	538.81
C#/Db	538.9	554.37	570.85
D	571.0	587.33	604.79
D#Eb	604.9	622.25	640.755
E	640.9	659.26	678.86
F	679.0	698.46	719.225
F#/Gb	719.4	739.99	761.99
G	762.0	783.99	807.49
G#/Ab	807.6	830.61	855.305
A	855.4	880.00	906.165
A#/Bb	906.3	932.33	960.05
B	960.2	987.77	1017.135
-	-	-	-
C	1017.3	1046.5	1077.6
C#/Db	1077.7	1108.7	1141.7
D	1147.8	1174.7	1209.6
D#Eb	1209.7	1244.5	1281.5
E	1281.6	1318.5	1357.7
F	1357.8	1396.9	1438.45
F#/Gb	1438.5	1480.0	1524.0
G	1524.1	1568.0	1614.6

G#/Ab	1614.7	1661.2	1710.6
A	1710.7	1760.0	1812.35
A#/Bb	1812.4	1864.7	1920.1
B	1920.2	1975.2	2034.25
-	-	-	-
C	2034.3	2093.0	2155.25
C#/Db	2155.3	2217.5	2283.4
D	2283.5	2349.3	2419.15
D#Eb	2419.2	2489.0	2563.0
E	2563.1	2637.0	2715.4
F	2715.4	2793.8	2876.9
F#/Gb	2877.0	2960.0	3048.0
G	3048.1	3136.0	3229.2
G#/Ab	3229.2	3322.4	3421.2
A	3421.3	3520.0	3624.65
A#/Bb	3624.7	3729.3	3840.2
B	3840.3	3951.1	4068.55
C	4068.6	4186.0	

ตารางที่ 3-10 แสดงชื่อตัวโน้ตและความถี่ของตัวโน้ต

บทที่ 4
ไมโครคอนโทรลเลอร์ตระกูล MCS-51

4.1 บทนำ

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 นั้นถือว่ามีความสำคัญเป็นอย่างมาก ทั้งนี้เพราะว่าโปรแกรมที่จัดทำนั้นจะรับค่ามาจากการแปลงสัญญาณของไมโครคอนโทรลเลอร์ ซึ่งแน่นอนว่าถ้าไมโครคอนโทรลเลอร์ประมวลผลผิดพลาดก็จะทำให้โปรแกรมอ่านและแสดงค่าผิดพลาดตามไปด้วย

ดังนั้นในบทนี้จะขอกล่าวถึงกระบวนการและขั้นตอนการทำงานของไมโครคอนโทรลเลอร์ตระกูล MCS-51 ที่ได้นำมาใช้ ว่ามีการทำงานและประสิทธิภาพเป็นเช่นใดบ้าง รวมทั้งเหตุใดจึงเลือกเอาเทคนิคนั้นๆมาใช้

4.2 ไมโครคอนโทรลเลอร์ตระกูล MCS-51

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 มีสมาชิกในตระกูลหลายเบอร์ด้วยกัน แต่ละเบอร์จะมีคุณสมบัติบางอย่างแตกต่าง เช่น มีหน่วยความจำภายในสำหรับเก็บโปรแกรมและข้อมูลภายในชิปเพิ่มขึ้น มีวงจรเปลี่ยนค่าสัญญาณอนาล็อกเป็นดิจิตอลในตัว สามารถรับสัญญาณอินเตอร์รัพต์ได้หลายชนิด ทำกระบวนการ DMA (Direct Memory Access) ได้ในตัว มีรีจิสเตอร์สำหรับใช้เป็นไทม์เมอร์หรือเคาน์เตอร์เพิ่มขึ้นคุณสมบัติพิเศษที่แตกต่างกันของไมโครคอนโทรลเลอร์แต่ละเบอร์ในตระกูลนี้ดังแสดงในตาราง

ไมโครคอนโทรลเลอร์เบอร์ที่นับได้ว่าเป็นเบอร์พื้นฐานสำหรับตระกูล MCS-51 นี้ ได้แก่ เบอร์ 8051,8031,8751 โดยเบอร์ 8051 จัดเป็นสมาชิกตัวแรกในตระกูล ซึ่งมีหน่วยความจำสำหรับเก็บโปรแกรมภายในชิป เป็น ROM ขนาด 4 กิโลไบต์ และหน่วยความจำสำหรับเก็บข้อมูลทั่วไปภายใน MCS-51 (RAM)เองจำนวน 128 ไบต์ มีพอร์ตขนาด 8 บิต 4พอร์ต มีรีจิสเตอร์สำหรับใช้เป็นไทม์เมอร์หรือเคาน์เตอร์ขนาด 16 บิต รวม 2 ตัว รับสัญญาณอินเตอร์รัพต์จากภายนอกได้ 2 ชนิด สามารถรับและส่งข้อมูลแบบอนุกรมผ่านทางพอร์ตสื่อสารข้อมูลแบบอนุกรม มีวงจรออสซิลเลเตอร์เพื่อสร้างสัญญาณนาฬิกาควบคุมการทำงานในตัวเอง ส่วนเบอร์ 8751 จะมีคุณสมบัติเหมือน เบอร์ 8051 ทุกอย่าง ต่างกันเพียงชนิดของหน่วยความจำสำหรับเก็บโปรแกรมภายในชิป คือ เบอร์ 8751 จะเป็น EPROM แทนที่จะเป็น ROM ส่วนเบอร์ 8031 จะเหมือนกับเบอร์ 8051 ต่างกันเพียงในเบอร์ 8031 ไม่มีหน่วยความจำสำหรับเก็บโปรแกรมภายในชิปเท่านั้น

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 ใช้แรงดันไฟเพียง 5 โวลต์ในการทำงานส่วนกระแสไฟฟ้าที่ใช้จะแตกต่างกันไปตามชนิดของเทคโนโลยีที่ใช้ในการผลิตเบอร์ของ
ไมโครคอนโทรลเลอร์ตระกูลนี้ที่มีตัวอักษร C อยู่ตรงกลางเบอร์ เช่น 80C31, 80C51 จะเป็นเบอร์ของชิปที่ผลิตโดยอาศัยเทคโนโลยี CHMOS ซึ่งใช้พลังงานในการทำงานน้อยกว่าและสามารถควบคุมการใช้พลังงานของตัวชิปได้จากโปรแกรมเพื่อการประหยัดในระบบ

MCS-51 เป็นตระกูลของไมโครคอนโทรลเลอร์ที่ถูกพัฒนาขึ้นมาจากตระกูล MCS-48 ดังนั้นจึงมีความสามารถเหนือกว่าหลายอย่าง ข้อดีของ MCS-51 คือสามารถใช้ความถี่ได้ถึง 12 เมกะเฮิรตซ์ หรือสำหรับบางเบอร์ในตระกูลสามารถใช้ได้ 16 เมกะเฮิรตซ์ ทำให้ช่วงเวลาในการทำงานแต่ละคำสั่งน้อยมาก เมื่อใช้ความถี่ 12 เมกะเฮิรตซ์ คำสั่งที่ใช้เวลาน้อยสุดจะใช้เวลาเพียง 1 ไมโครวินาที ส่วนคำสั่งที่ใช้เวลามากที่สุดจะใช้เวลาเพียง 4 ไมโครวินาทีเท่านั้น

4.3 การทำงานของ MCS-51

คอมพิวเตอร์จะทำงานด้วยวงจรที่เรียกว่าฮาร์ดแวร์ (Hardware) ประกอบขึ้นมาเพียงอย่างเดียวไม่ได้ จะต้องมีโปรแกรมหรือคำสั่งที่จัดเรียงไว้ให้คอมพิวเตอร์ทำงานตามลำดับใน 8051 ก็เช่นกัน ผู้ใช้จะต้องเขียนโปรแกรมเป็นภาษาเครื่อง ซึ่งอยู่ในรูปของเลขฐาน 2 เก็บไว้ในหน่วยความจำประเภท Program Memory แต่ละคำสั่งของ 8051 อาจประกอบด้วย 1 , 2 หรือ 3 ไบต์ แล้วแต่ว่าจะเป็นคำสั่งให้ทำงานอะไร คอมพิวเตอร์ก็จะเหมือนกับคนที่จะต้องทำงานตามคำสั่ง เมื่อรับคำสั่งแล้วก็จะไปทำตามคำสั่งนั้นเสร็จสิ้นแล้วก็กลับมารับคำสั่งต่อไป

เมื่อเริ่มป้อนไฟเลี้ยงให้กับ 8051 ซึ่งมีวงจร Program Counter ซึ่งเป็นวงจรนับ (Counter Circuit) ชนิดหนึ่ง ส่งค่าตำแหน่งหน่วยความจำสำหรับโปรแกรมลงไปบนบัส (Bus) หมายเลข 1 บัสนี้มีขนาด 16 บิต ค่าตำแหน่งหน่วยความจำนี้จะถูกส่งไปเก็บไว้ที่ Program ADDR Register ที่เป็นวงจร Latch ข้อมูลซึ่งเป็นค่าตำแหน่งหน่วยความจำ จะปรากฏที่บนบัส 16 บิต หมายเลข 2 ถ้าเป็นค่าตำแหน่งหน่วยความจำหลังจากรีเซตค่าตำแหน่งหน่วยความจำจะเป็น 0000H หน่วยความจำสำหรับโปรแกรมจะเลือกได้ว่าเป็น ROM ภายในหรือภายนอก 8051 โดยการป้อนสภาวะลอจิกเข้าไปที่ 8051 ทางขา EA ซึ่งต่ออยู่กับส่วน Timing and Control ทำหน้าที่เป็นวงจรถอดรหัส (Decoder) แล้วสร้างสัญญาณควบคุมต่อไปถ้าสัญญาณลอจิก 0 เข้าไปที่ขา EA จะเป็นการเลือกใช้ ROM ภายใน 8051 โดยใช้วงจร Timing and Control จะสร้างสัญญาณไปยัง ROM ภายในให้ส่งข้อมูลที่เป็นคำสั่งจากตำแหน่งที่ส่งมาทางบัสหมายเลข 2 ข้อมูลจาก ROM จะถูกส่งไปยังบัสหมายเลข 3 ที่เรียกว่า Internal Data Bus แล้วนำไปเก็บไว้ที่ Instruction Register (เป็นวงจร Latch) เพื่อส่งต่อไปให้กับวงจร Timing and Control ทำการถอดรหัสแล้วควบคุมการทำงานส่วนอื่นๆต่อไปแล้วแต่ว่าเป็นคำสั่งให้ทำงานอะไรในกรณีที่เลือก ROM ภายนอก 8051 โดยป้อนสัญญาณลอจิก 1 เข้าไปที่ขา EA จะทำให้วงจร Timing and Control ส่งสัญญาณไปยังพอร์ต 0 และพอร์ต 2 เพื่อส่งค่าตำแหน่งหน่วยความจำบนบัสหมายเลข 2 ออกไปชี้หน่วยความจำภายนอก จากนั้นจะอ่านข้อมูลที่เป็นคำสั่งกลับเข้าทางพอร์ต 0 ไปยัง Internal Data Bus แล้วไปเก็บที่ Instruction Register เรียกว่าเป็นช่วงของการ Fetch (Fetch cycle) ช่วงต่อไปจะเป็นช่วงของการทำงานตามคำสั่งเรียกว่า Execute Cycle เช่น ถ้าเป็นคำสั่งให้บวกข้อมูลในรีจิสเตอร์ Accumulator กับข้อมูลจากหน่วยความจำ Data Memory ภายใน RAM ตำแหน่ง 23H วงจร Timing and Control ก็จะส่งสัญญาณให้ Instruction Register ส่งค่าตำแหน่งหน่วยความจำ 23H ลงไปยัง Internal Data Bus แล้วนำข้อมูลไปเก็บไว้ที่ RAM ADDR Register เพื่อใช้ชี้ตำแหน่งหน่วยความจำ RAM จากนั้น Timing and Control จะสั่ง RAM ส่งข้อมูลที่เก็บอยู่ในหน่วยความจำตำแหน่ง 23H ลงมายัง Internal Data Bus แล้วนำข้อมูลไป

เก็บไว้ที่ TMP1(เป็นวงจร Latch) ซึ่งจะทำหน้าที่เก็บสถานะ ผลลัพธ์ของการทำงานใน ALU เช่นผลลัพธ์การบวกมีค่าเกิน 8 บิต ก็จะทำให้บิตหนึ่งใน PSW ถูก SET เป็น 1

4.4 โครงสร้างหน่วยความจำ

หน่วยความจำสำหรับ MCS-51 จะมี 2 ชนิดคือ หน่วยความจำที่ใช้เก็บโปรแกรม (ROM) กับหน่วยความจำที่ใช้เก็บข้อมูลในการประมวลผล(RAM) MCS-51 บางเบอร์เช่น 8051 , 8052 จะมีหน่วยความจำโปรแกรมภายในชิป และ MCS-51 ทุกเบอร์สามารถอ้างหน่วยความจำโปรแกรมภายนอกได้มากที่สุด 64K และอ้างหน่วยความจำข้อมูลภายนอกได้มากที่สุด 64K สำหรับหน่วยความจำ RAM ภายใน จะประกอบไปด้วยพื้นที่ใช้งานทั่วไป, รีจิสเตอร์แบงค์ , พื้นที่ใช้งานระดับบิต และรีจิสเตอร์ฟังก์ชันพิเศษ เราอาจเขียนไดอะแกรมของหน่วยความจำของ 8031 ได้ดังรูปที่ 2.13 โดยในรูปจะบอกด้วยว่าขาใดจะแอคทีฟ

ใน 8031 จะมีหน่วยความจำภายในตั้งแต่ตำแหน่ง 00H ถึง FFH และสามารถอ้างหน่วยความจำโปรแกรมภายนอกได้ 64K ตำแหน่ง ถ้าอ่านข้อมูลจากหน่วยความจำโปรแกรมขา PSEN จะแอคทีฟ นอกจากนี้ 8031 สามารถอ้างหน่วยความจำข้อมูลภายนอกได้ 64K ตำแหน่ง โดยการติดต่อกับหน่วยความจำนี้ ขา RD และ WR จะแอคทีฟ สำหรับหน่วยความจำข้อมูลภายในนั้นจะแบ่งออกได้ดังนี้

1 ชุดรีจิสเตอร์ 4 ชุด แต่ละชุดเรียกว่ารีจิสเตอร์แบงค์ที่ตำแหน่ง 00K ถึง 1FH โดยแต่ละชุดประกอบด้วยรีจิสเตอร์ R0 ถึง R7
2 หน่วยความจำที่สามารถเข้าถึงขอมูลระดับบิตได้ ตำแหน่ง 20H ถึง 2FH
3 หน่วยความจำใช้งานทั่วไปตำแหน่ง 30H ถึง 7FH
4 รีจิสเตอร์ฟังก์ชันพิเศษ ตำแหน่ง 80H ถึง FFH

4.5 หน่วยความจำใช้งานทั่วไป

จะเห็นได้ว่าใน 8031 จะมีหน่วยความจำ RAM สำหรับใช้งานทั่วไปจำนวน 80 ไบต์ ตั้งแต่ตำแหน่ง 30H ถึง 7FH ตำแหน่งเหล่านี้สามารถอ้างตำแหน่งแบบ Direct Addressing Modeหรือ Indirect Addressing Mode การย้ายข้อมูลจากตำแหน่งที่เก็บโดยตรง (ตำแหน่ง 5FH) เรียกว่าการอ้างตำแหน่งแบบ Direct Addressing Mode นอกจากนี้ยังสามารถอ่านข้อมูลโดยรีจิสเตอร์ R0 หรือ R1 การการตัวชี้ตำแหน่งได้ เรียกว่าการอ้างตำแหน่งแบบ Indirect Addressing Mode Bit-addressable

RAM ใน MCS-51 จะมีหน่วยความจำที่สามารถอ้างข้อมูลในระดับบิตไดตั้งแต่ตำแหน่ง 20H ถึง 2FH รวม16 ไบต์ โดยสามารถ SET, CLEAR, AND, OR ทางลอจิกได้ จำนวนบิตที่ใช้งานได้ทั้งหมดมีจำนวน 128 บิต (8 บิต x 16 ไบต์)

4.6 รีจิสเตอร์ฟังก์ชันพิเศษ (Special Function Register)

ใน MCS-51 รีจิสเตอร์จะใช้หน่วยความจำ RAM ภายในชิป โดยส่วนหนึ่งเป็นรีจิสเตอร์พิเศษ (Special Function Register: SFR) ซึ่งมีทั้งหมด 21 ตัว โดยรีจิสเตอร์พิเศษต่างๆจะเริ่มที่หน่วยความจำตั้งแต่ 80H ถึง FFH ซึ่งมีทั้งหมด 128 ตำแหน่ง แต่จะเป็นรีจิสเตอร์ฟังก์ชันพิเศษเพียง 21 ตำแหน่ง แต่ถ้าเป็น 8032/8051 จะใช้ 26 ตำแหน่งหรือมี SFR 26 ตัว

4.7 Program Status World

รีจิสเตอร์ตัวนี้เรียกย่อๆว่า PSW จะอยู่ที่ตำแหน่ง D0H ซึ่งสามารถเข้าถึงข้อมูลระดับบิตได้โดย รีจิสเตอร์นี้จะเป็นตัวบอกสภาวะต่างๆของไมโครคอนโทรลเลอร์ ความหมายของแต่ละบิตแสดงได้ดังตารางที่ 2.13.1

บิต	ชื่อบิต	ตำแหน่ง	ความหมาย
PSW.7	CY	D7H	Carry Flag
PSW.	AC	D7H	Auxiliary Carry Flag
PSW.	F0	D7H	Flag 0
PSW.	RS1	D7H	บิตสำหรับการเลือก Register Bank 1
PSW.	RS0	D7H	บิตสำหรับการเลือก Register Bank 0
			00 = Bank 0; Address 00H – 07H 01 = Bank 1; Address 08H – 0FH 10 = Bank 2; Address 10H – 17H 11 = Bank 3; Address 18H – 1FH
PSW.	OV	D7H	Overflow Flag
PSW.	-	D7H	Reserved
PSW.	P	D7H	Even Parity Flag

ตารางที่ 4-1แสดงบิตและหน้าที่ต่างๆใน PSW

แฟลกตัวทด Carry Flag (CF) บิตนี้เป็นบิตที่ 7 ของ PSW บิตนี้จะมีความสำคัญหากมีการกระทำทางคณิตศาสตร์ โดยบิตนี้จะ Set เมื่อเกิดการทดของตัวทดบิตที่ 7 ขณะทำการบวกเลขหรือ Set เมื่อเกิดการยืมของบิตที่ 7 เมื่อเกิดการลบเลข

แฟลกตัวช่วยทด Auxiliary Carry Flag เมื่อมีการบวกแบบ Binary-Code-Decimal (BCD) บิต Auxiliary Carry Flag (AC) หรือบิตตัวช่วยทดจะถูก Set เมื่อมีการทดจากบิตที่ 3 ไปบิตที่ 4 หรือถ้าใน Lower Nibble มีค่าระหว่าง 0AH-0FH เนื่องจากรหัส BCD นี้มีค่าได้มากที่สุดแค่ 9 ถ้าหากการบวกเลข

แบบ BCD จะต้องตามด้วยคำสั่ง DAA (Decimal Adjust Accumulator) เพื่อปรับค่าที่มีค่าเกิน 9 โดยบวกเลข 6 เข้าไป จะทำให้เป็นรหัส BCD ที่แทนเลขฐานสิบได้

แฟลกศูนย์ (Flag 0) เป็น Flag ที่ผู้ใช้สามารถใช้งานทั่วไปได้

บิตเลือกรีจิสเตอร์แบงก์ (Register Bank Select Bits) ตามที่เราทราบมาแล้วว่าใน MCS-51 จะมีชุดรีจิสเตอร์อยู่ 4 ชุด ถ้าจะเลือกให้ใช้ชุดใดแอดทีฟจะกำหนดได้ในบิต RS1และ RS2 ของ PSW และจะ Clear ตัวเองเมื่อระบบถูกรีเซต

แฟลกโอเวอร์โฟลว์ Overflow Flag แฟลก OV จะถูก Set หลังจากการกระทำทางคณิตศาสตร์แล้วเกิด Overflow คือ จำนวนที่เกิดจากการบวกหรือการลบ มีค่าเกินกว่าที่จำนวนไบต์จะเป็นไปได้คือมากกว่า +128 น้อยกว่า -128

บิตพาริตี้ (Parity Bit) พาริตี้บิต (P) ในบิตที่บอกค่าพาริตี้ของรีจิสเตอร์ Accumulator ซึ่งอาจตรวจสอบความถูกต้องของข้อมูลได้ โดยจะเซตหรือเคลียร์ ขึ้นกับผลที่เกิดขึ้นกับ Accumulator

4.8 รีจิสเตอร์ B (B Register)

รีจิสเตอร์ B จะอยู่ตำแหน่ง F0H ของหน่วยความจำข้อมูลภายใน เป็นรีจิสเตอร์ที่สามารถใช้งานทั่วไปได้ โดยทั่วไปรีจิสเตอร์นี้จะใช้คูณหรือหารกับรีจิสเตอร์ Accumulator เช่น การทำคำสั่ง MUL AB ซึ่งเป็นการคูณแบบ 8 บิต โดยผลลัพธ์ที่ได้จะมีขนาด 16 บิต ซึ่งรีจิสเตอร์ A จะเก็บค่า 8 บิตต่ำ รีจิสเตอร์ B จะเก็บค่า 8 บิตสูง สำหรับการหารโดยการทำคำสั่ง DIV AB โดยค่าใน A จะถูกหารด้วย B ผลลัพธ์ที่ได้จะเก็บใน รีจิสเตอร์ AB โดย B จะเก็บค่า 8 บิตต่ำและ A จะเก็บค่า 8 บิตสูง รีจิสเตอร์ B นี้สามารถเข้าถึงข้อมูลระดับบิตได้ โดยตำแหน่งของบิตคือตำแหน่ง F0H ถึง F7H

4.9 ตัวชี้สแตค (Stack Pointer)

Stack Pointer (SP) เป็นรีจิสเตอร์ขนาด 8 บิต อยู่ที่ตำแหน่ง 81H การเขียนค่าเข้าไปในตำแหน่งที่ SP ชี้อยู่นี้ เรียกว่า “Pushing” สำหรับการอ่าน ค่าที่ SP ชี้อยู่ เรียกว่า “Popping” ค่าของ SP จะเพิ่มขึ้นหนึ่งก่อนที่จะเขียนข้อมูลลงไป และจะลดลงหนึ่งเมื่ออ่านข้อมูลออกมาแล้ว หากโปรแกรมทำคำสั่ง CALL จะใช้รีจิสเตอร์สแตคนี้เก็บค่าตำแหน่งเดิมของโปรแกรม (PC) ก่อนที่จะทำโปรแกรมย่อยเมื่อทำโปรแกรมย่อยเสร็จแล้วจะคืนค่าในสแตคให้กับ PC ตามเดิม โดยปกติค่า PC จะกำหนดให้อยู่ใน RAM ภายใน

4.10 รีจิสเตอร์ Data Pointer (DPTR)

รีจิสเตอร์นี้ใช้สำหรับชี้ตำแหน่งรหัสโปรแกรมหรือข้อมูลในหน่วยความจำ โดยรีจิสเตอร์ขนาด 16 บิต ซึ่งประกอบด้วยรีจิสเตอร์ 2 ตัว คือ DPL ตำแหน่งที่ 82H โดยจะเก็บเป็น 8 บิตต่ำ และ DPH ตำแหน่ง 83H โดยเก็บเป็นค่า 8 บิตสูง รีจิสเตอร์ทั้งสองตัวนี้จะรวมกันกลายเป็นรีจิสเตอร์ 16 บิต

4.11 รีจิสเตอร์พอร์ต (Port Registers)

ใน MCS-51 ค่าของพอร์ตจะหมายถึงค่าของหน่วยความจำด้วย หากต้องการส่งขอมูลออกไปที่พอร์ต ก็เพียงแต่เขียนข้อมูลไปที่หน่วยความจำตำแหน่งที่พอร์ตนั้นอยู่ และถ้าหากต้องการจะอ่านข้อมูลจากพอร์ต ก็เพียงอ่านค่าจากตำแหน่งที่หน่วยความจำนั้นอยู่ ใน MCS-51 พอร์ต 0 จะอยู่ที่ตำแหน่ง 80H , พอร์ต 1 จะอยู่ที่ตำแหน่ง 90H , พอร์ต 2 จะอยู่ที่ตำแหน่ง A0H และ พอร์ต 3 จะอยู่ที่ตำแหน่ง B0H พอร์ต 0,2 และ 3 โดยทั่วไปแล้วจะไม่ใช้หากติดต่อกับหน่วยความจำภายนอกหรือใช้เป็นพอร์ตพิเศษ โดยปกติแล้วจะใช้พอร์ต 1 ในการติดต่อกับอุปกรณ์ภายนอก พอร์ตทุกพอร์ตสามารถอ้างข้อมูลในระดับบิตได้

4.12 รีจิสเตอร์เวลา (Timer Registers)

ใน MCS-51 เบอร์ 8051 จะมีรีจิสเตอร์ที่ใช้นับและจับเวลาขนาด 16 บิต 2 ตัว คือ Timer 0 อยู่ที่ตำแหน่ง 8AH และ 8CH โดยตำแหน่ง 8AH หมายถึง TL0 ซึ่งจะเป็น 8 ไบต์ต่ำ และหมายถึง 8 ไบต์สูง รีจิสเตอร์อีกตัวคือ Timer 1 โดยแบ่งเป็น TL1 อยู่ที่ตำแหน่ง 8BH เป็นไบต์ต่ำ และ TH1 อยู่ที่ตำแหน่ง 8DH เป็นไบต์สูง การใช้ Timer จะต้องกำหนดการทำงานในรีจิสเตอร์ TMOD (Timer/Count Mode Control Register) ซึ่งอยู่ในตำแหน่ง 88H เสียก่อน

4.13 รีจิสเตอร์พอร์ตอนุกรม (Serial Port Registers)

MCS-51 จะมีพอร์ตสื่อสารอนุกรมอยู่ (Serial Port) ภายในชิป ซึ่งสามารถจะรับหรือส่งข้อมูลได้ โดยติดต่อผ่านรีจิสเตอร์ SBUF (Serial Data Buffer) ซึ่งอยู่ในตำแหน่ง 99H โดยถ้าต้องการส่งข้อมูลแบบอนุกรมให้เขียนข้อมูลไปที่จิสเตอร์นี้ ตัว Serial Port สามารถโปรแกรมให้ทำงานได้ 4 โหมด โดยโปรแกรมผ่านรีจิสเตอร์ SCON (Serial Port Control Registers) ตำแหน่ง 98H

4.14 รีจิสเตอร์อินเทอร์รัพท์ (Interrupt Port Registers)

MCS-51 สามารถ Interrupt ได้ 5 ตำแหน่งโดยมี 2-Priority ตัว Interrupt นี้จะถูก Disable หลังจากระบบถูกรีเซต และจะ Enabled หลังจากที่เขียนข้อมูลไปที่รีจิสเตอร์ IE หรือตำแหน่ง A8H ลำดับความสำคัญสามารถเซตได้ที่รีจิสเตอร์ IP หรือตำแหน่ง 88H

4.15 Power Control Register (PCON)

รีจิสเตอร์ PCON อยู่ที่ตำแหน่ง 87H ใช้หยุดการทำงานของ MCS-51 โดยจะหยุดจ่ายสัญญาณนาฬิกาให้ระบบ ทำให้ข้อมูลต่างๆภายใน MCS-51 ไม่มีการเปลี่ยนแปลงนอกจากนี้ยังลดพลังงานไฟฟ้าที่จ่ายให้ MCS-51 ลงด้วย

4.16 หน่วยความจำภายนอก (External Memory)

MCS-51 สามารถอ้างหน่วยความจำข้อมูลภายนอกได้ 64K และอ้างหน่วยความจำโปรแกรมภายนอกได้ 64K จะใช้พอร์ต 0 ในการอ้างตำแหน่งหน่วยความจำ 8 บิตล่างและใช้พอร์ต 0 เป็นพอร์ตข้อมูล (DATA) ด้วย โดยขา A LE มาเป็น Latch ข้อมูลพอร์ต 0 และใช้พอร์ต 2 เป็นขาอ้างตำแหน่ง 8 บิตบน(รวมขาอ้างตำแหน่ง 16 เส้น ซึ่งอ้างได้ 64K)เนื่องจากพอร์ต 0 จะใช้งาน 2 หน้าที่ในการติดต่อกับหน่วยความจำ จะใช้วิธี Multiplex ระหว่าง Address กับ Data พิจารณาจากรูป ถ้าต้องการติดต่อกับหน่วยความจำที่เก็บข้อมูล 8 บิต และเก็บได้ 64K จะต้องใช้สายสัญญาณ 24 เส้น คือ เป็นขา Address 16 เส้น และขาข้อมูล 8 เส้น

4.17 การติดต่อกับหน่วยความจำโปรแกรมภายนอก

ในการอ่านข้อมูลจากหน่วยความจำโปรแกรมภายนอก MCS-51 จะส่งค่าตำแหน่งของหน่วยความจำออกไปก่อน ซึ่งค่าตำแหน่งจะเก็บอยู่ใน PC โดยส่งออกไปทางพอร์ต 0 และพอร์ต 2 จากนั้นเวลาต่อมาจะส่งขา ALE ให้เป็นลอจิก “0” เพื่อ Latch ขา Address ของ 8 บิตต่ำ คือ พอร์ต 0 จากนั้นจะส่งสัญญาณทางขา PSEN ให้เป็นลอจิก “0” เพื่ออ่านข้อมูลซึ่งจะได้ Opcode เข้าไปทางขา Data Bus คือ พอร์ต 0

สำหรับการต่อหน่วยความจำกับ MCS-51 โดยขา EA จะต่อเป็น “0” เพื่อบอก MCS-51 ว่าอ่านหน่วยความจำโปรแกรมภายนอก สำหรับการ Multiplex จะใช้ฟลิปฟลอป 8 ตัวเบอร์ 74373 เก็บค่าตำแหน่ง 8 บิตต่ำเอาไว้ เมื่อ MCS-51 ส่งค่าตำแหน่งพอร์ตออกไปเวลาต่อมาจะส่งขา ALE ให้เป็น “0” ซึ่งจะใช้ขานี้ต่อกับ 74373 เพื่อให้ Latch ข้อมูลสำหรับขา PSEN จะต่อกับขา Output Enable (OE) ของหน่วยความจำ

4.18 การติดต่อกับหน่วยความจำข้อมูลภายนอก

หน่วยความจำข้อมูลภายนอก MCS-51 สามารถอ่านและเขียนได้ ในการติดต่อกับหน่วยความจำข้อมูลภายนอก MCS-51 จะส่งขา Address ออกไปทางพอร์ต 0 และพอร์ต 2 จากนั้นจะส่งขา ALE เพื่อไป Latch Address 8 บิตต่ำ โดยการอ่านเขียนข้อมูลนั้นจะใช้ขา RD หรือ P3.7 และขา WR หรือ P3.6 ตามลำดับ

4.19 Reset Operation

การรีเซตหรือเริ่มตนทำงานใหม่ของ MCS-51 จะต้องให้ลอจิก “1” ที่ขา RST เป็นเวลา 2 Machine Cycle (1 Machine Cycle เท่ากับ 12 Clock) จากนั้นให้กลับเป็นลอจิก “0” การรีเซตอาจทำได้โดยใช้สวิตช์กดหรือใช้วิธี Power-up โดยใช้ตัว R-C ต่อเป็นวงจร

เมื่อ MCS-51 ถูกรีเซต ค่ารีจิสเตอร์ต่างๆจะถูกกำหนดค่าตาราง โดย PC จะชี้ไปอยู่ที่ตำแหน่งเริ่มต้น คือ 0000H เมื่อขา RST กลับเป็น “0” MCS-51 จะเริ่มทำโปรแกรมที่ตำแหน่งแรก

4.20 TIMER

ตัว Timer อาจพิจารณาได้ง่ายๆว่าเป็นตัวฟลิปฟลอปมาต่อเรียงกันโดยมี Clock เป็นอินพุต สำหรับเอาต์พุตที่ออกมาจากฟลิปฟลอปแต่ละตัวจะถูกหารด้วย 2 พิจารณาการต่อฟลิปฟลอป ถ้าใส่ Clock เข้าไปในฟลิบฟลอปตัวแรก ความถี่ของ Clock ที่ออกจากเอาต์พุตตัวแรกจะถูกหารด้วย 2 และเอาต์พุตที่จะต่อกับฟลิปฟลอปตัวที่สอง และสัญญาณที่ออกมาจะถูกหารด้วย 2 อีก ดังนั้นถ้ามีฟลิปฟลอปต่ออยู่ n Stage จะหารสัญญาณนาฬิกาได้ 2^n ถ้าให้เอาต์พุต Stage สุดท้ายของ Timer เป็น Overflow Flip-flop หรือ Flag และจะให้เอาต์พุตออกมาเมื่อมีการนับเป็น Overflow เช่น ถ้าเป็นตัวนับแบบ 16 บิต(มีฟลิปฟลอปต่ออยู่ 16 ตัว) วงจรจะนับตั้งแต่ 0000H ถึง FFFFH เมื่อฟลิปฟลอปเปลี่ยนจาก 0000H จะให้บิต Overflow ออกมา ใน MCS-51 จะมีตัวจับเวลาอยู่ภายในชิป ถ้าเป็นเบอร์ 8051 หรือ 8031 จะมี 2 ตัว คือ Timer 0 และ Timer 1 แต่ถ้าเป็นเบอร์ 8052 จะมีเพิ่มอีกหนึ่งตัวคือ Timer 2 รีจิสเตอร์ต่างๆที่เกี่ยวข้องกับการใช้ Timer

บทที่ 5
การออกแบบและการดำเนินงาน

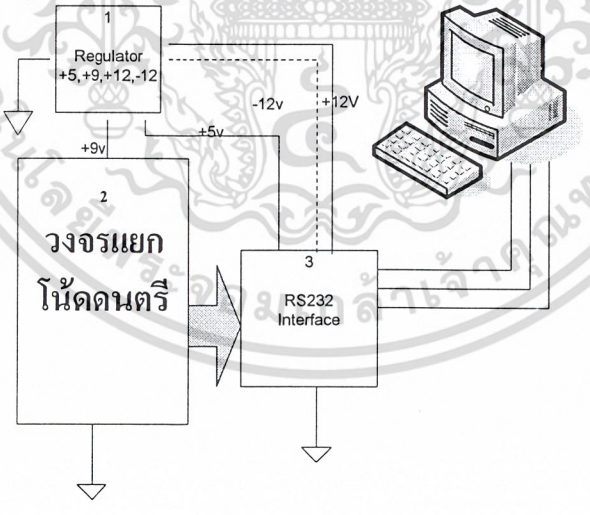
5.1 บทนำ

เนื้อหาในบทนี้จะได้กล่าวถึงขั้นตอนการออกแบบโครงงาน ซึ่งภายหลังจากการออกแบบนั้นได้ทำการแบ่งการทำงานของโครงงานออกเป็นส่วนๆ โดยโครงงานจะเริ่มจากทำการรับอินพุตเข้ามา แล้วนำเสียงที่ได้มาผ่านกระบวนการแปลงเป็นสัญญาณดิจิตอลเพื่อทำการส่งให้เครื่องคอมพิวเตอร์ประมวลผล จากนั้นเครื่องคอมพิวเตอร์จะทำการประมวลผลค่าที่ได้แล้ววิเคราะห์ว่าเป็นโน้ตอะไรความเร็วเท่าไร เสร็จแล้วทำการตรวจสอบหาตำแหน่งของบรรทัด 5 เส้นเพื่อเขียนโน้ตตัวนั้นลงบรรทัด 5 เส้น

5.2 ขั้นตอนการทำงานของโครงงาน

ส่วนของฮาร์ดแวร์แบ่งออกได้เป็น 3 ส่วนสำคัญคือ

1. ส่วนของแหล่งจ่ายไฟ (Regulator)
2. ส่วนของวงจรแยกเสียงโน้ตดนตรี
3. ส่วนของการอินเทอร์เฟสกับเครื่องคอมพิวเตอร์

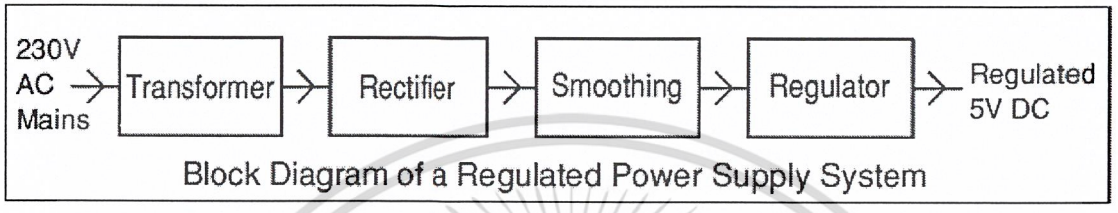


รูปที่ 5-1 แสดงขั้นตอนการออกแบบวงจร

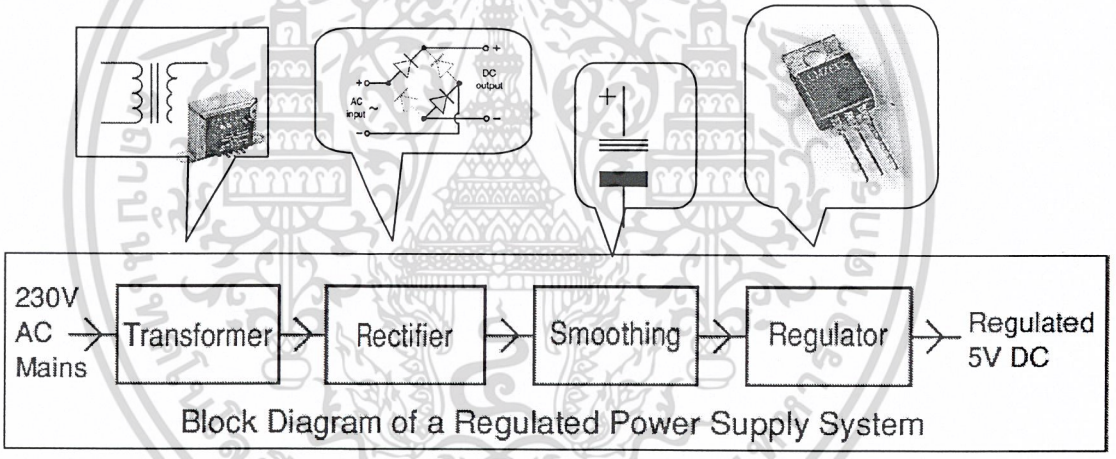
5.3 รายละเอียดของ Hardware แต่ละส่วน

1. ส่วนของแหล่งจ่ายไฟ

ใช้แหล่งกำเนิดพลังงาน +9Vสำหรับจ่ายพลังงานเลี้ยงภายในเครื่อง และ +5V สำหรับจ่ายพลังงานให้กับวงจรดิจิตอล TTL และ +12,-12 สำหรับวงจรอินเทอร์เฟสกับคอมพิวเตอร์ (ซึ่งตามมาตรฐานของการเชื่อมต่อ RS-232 โดยใช้ IC สำเร็จรูป MAX232) ซึ่งแหล่งกำเนิดพลังงานทั้งหมดมีวงจรดังรูปที่ 5-2



รูปที่ 5-2 แสดงไดอะแกรมของวงจรจ่ายไฟ



รูปที่ 5-3 แสดงรายละเอียดของอุปกรณ์แต่ละ Block

ในวงจรจำเป็นต้องใช้ระดับของแรงดัน 3 ระดับด้วยกันคือ

0v,+5 v สำหรับวงจร TTL

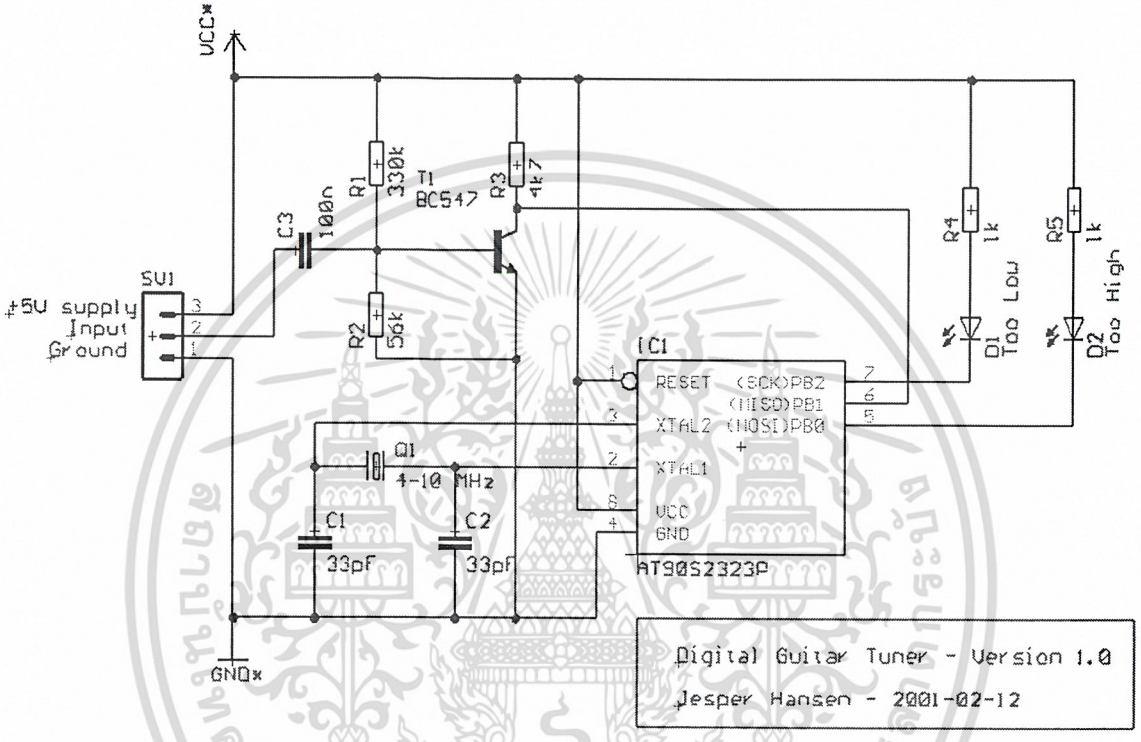
0v, +9 v สำหรับวงจร แยกเสียงดนตรี

-12v,0,+12v สำหรับวงจรอินเทอร์เฟส

โดยเราจะใช้ IC ที่ทำหน้าที่เป็นตัว Regulator ตระกูล 78xx (สำหรับแรงดันบวก)และ 79xx (สำหรับแรงดันลบ)

2. วงจรที่ใช้ในการแยกโน้ตแต่ละตัว

ในส่วนนี้เราจะนำเอามาตรฐานความถี่ของโน้ตแต่ละตัวมาเป็นค่าพารามิเตอร์ที่ใช้ในการออกแบบ โดยจะใช้วงจรกรองความถี่ Band pass ฟิลเตอร์ ใช้ในการออกแบบค่าวงจรที่ใช้ในการแยกโน้ตแต่ละตัว ซึ่งในโครงงานนี้จะใช้ไอซีเบอร์ AT90S2323P ซึ่งเป็นไอซีสำเร็จรูปในการแยกเสียงของตัวโน้ต และได้จัดทำวงจรกรองความถี่ของตัวโน้ตขึ้นมาดังในรูปที่ 5-4



รูปที่ 5-4 แสดงภาพวงจรที่ใช้แยกความถี่เสียง

จากรูปที่ 5-4 การทำงานของวงจรจะทำงานโดยรับอินพุตเข้ามาคือสัญญาณเสียงจากเครื่องดนตรี จากนั้นจะมี C3 เป็นตัวคับปิ้งสัญญาณ แล้วทำการขยายสัญญาณให้เพิ่มขึ้นโดยผ่านทางทรานซิสเตอร์ แล้วจะนำสัญญาณที่ได้นั้นเข้าไปทางขาที่หก ของไอซีเบอร์ AT90S2323P ภายในไอซีเบอร์ AT90S2323P นั้นจะเป็นชุดประมวลคำสั่งแบบสำเร็จรูป ที่จะใช้สำหรับแยกเสียงตัวโน้ตต่างๆที่ป้อนเข้ามาในวงจร และจะแสดงผลออกมาในรูปของสัญญาณไบนารีขนาดสิบหกบิต ซึ่งต่อมาจะทำการส่งให้ MCS-51 ทำการแปลงให้เป็นเลขฐานสิบหก ซึ่งความถี่ของโน้ตต่างๆที่ป้อนให้กับวงจรจะแสดงตามตารางที่ 5-1

ชื่อตัวโน้ต	เริ่มต้น(Hz)	มาตรฐาน(Hz)	สิ้นสุด(Hz)
A		27.5	28.3175
A#/Bb	28.32	29.135	30.0015
B	30.1	30.868	31.7855
-	-	-	-
C	31.8	32.703	33.6755
C#/Db	33.7	34.648	35.678
D	35.7	36.708	37.7995
D#/Eb	37.8	38.891	40.047
E	40.1	41.203	42.4285
F	42.5	43.654	44.9515
F#/Gb	45.0	46.249	47.624
G	47.7	48.999	50.456
G#/Ab	50.5	51.913	53.4565
A	53.5	55.0	56.635
A#Bb	56.7	58.270	60.0025
B	60.1	61.735	63.5705
-	-	-	-
C	63.6	65.406	67.351
C#/Db	67.4	69.296	71.356
D	71.4	73.416	75.599
D#/Eb	75.6	77.782	80.0945
E	80.1	82.407	84.857
F	84.9	87.307	89.903
F#/Gb	90.0	92.499	95.249
G	95.3	97.999	100.9145
G#Ab	101.0	103.83	106.915
A	107.0	110.0	113.27
A#/Bb	113.3	116.54	120.005
B	120.1	123.47	127.14
-	-	-	-
C	127.2	130.81	134.7
C#/Db	134.8	138.59	142.71

D	142.8	146.83	151.195
D#Eb	151.2	155.56	160.185
E	160.19	164.81	169.71
F	169.8	174.61	179.805
F#/Gb	179.9	185.00	190.5
G	190.6	196.00	201.825
G#/Ab	201.9	207.65	213.825
A	213.9	220.00	226.54
A#/Bb	226.6	233.08	240.01
B	240.2	246.94	254.285
-	-	-	-
C	254.30	261.63	269.40
C#/Db	269.41	277.18	285.42
D	285.43	293.66	302.39
D#Eb	302.40	311.13	320.38
E	320.39	329.63	339.43
F	339.44	349.23	359.61
F#/Gb	359.62	369.99	380.99
G	381.00	391.99	403.65
G#/Ab	403.66	415.31	427.65
A	427.66	440.00	453.08
A#/Bb	453.09	466.16	480.02
B	480.03	493.88	508.56
-	-	-	-
C	508.6	523.25	538.81
C#/Db	538.9	554.37	570.85
D	571.0	587.33	604.79
D#Eb	604.9	622.25	640.755
E	640.9	659.26	678.86
F	679.0	698.46	719.225
F#/Gb	719.4	739.99	761.99
G	762.0	783.99	807.49
G#/Ab	807.6	830.61	855.305
A	855.4	880.00	906.165

A#/Bb	906.3	932.33	960.05
B	960.2	987.77	1017.135
-	-	-	-
C	1017.3	1046.5	1077.6
C#/Db	1077.7	1108.7	1141.7
D	1147.8	1174.7	1209.6
D#Eb	1209.7	1244.5	1281.5
E	1281.6	1318.5	1357.7
F	1357.8	1396.9	1438.45
F#/Gb	1438.5	1480.0	1524.0
G	1524.1	1568.0	1614.6
G#/Ab	1614.7	1661.2	1710.6
A	1710.7	1760.0	1812.35
A#/Bb	1812.4	1864.7	1920.1
B	1920.2	1975.2	2034.25
-	-	-	-
C	2034.3	2093.0	2155.25
C#/Db	2155.3	2217.5	2283.4
D	2283.5	2349.3	2419.15
D#Eb	2419.2	2489.0	2563.0
E	2563.1	2637.0	2715.4
F	2715.4	2793.8	2876.9
F#/Gb	2877.0	2960.0	3048.0
G	3048.1	3136.0	3229.2
G#/Ab	3229.2	3322.4	3421.2
A	3421.3	3520.0	3624.65
A#/Bb	3624.7	3729.3	3840.2
B	3840.3	3951.1	4068.55
C	4068.6	4186.0	

ตารางที่ 5-1 แสดงความถี่ของตัวโน้ตที่ป้อนเข้าไปในวงจร

จากการนำความถี่ป้อนให้กับวงจรแล้วเมื่อส่งค่าให้กับ MCS-51 ก็จะทำการแปลงสัญญาณที่ได้นั้นให้อยู่ในรูปของเลขฐานสิบหกดังแสดงในตารางที่ 5-2 ถึงตารางที่ 5-6

ชื่อโน้ตดนตรี	b	Tuner	#	C	D	E	F	G	A	B	#	G1	G2	G3	used	used
C	1	0	1	0	1	1	1	1	1	1	1	0	0	0	1	1
C#/Db	1	0	1	0	1	1	1	1	1	1	0	0	0	0	1	1
D	1	0	1	1	0	1	1	1	1	1	1	0	0	0	1	1
D#/Eb	1	0	1	1	0	1	1	1	1	1	0	0	0	0	1	1
E	1	0	1	1	1	0	1	1	1	1	1	0	0	0	1	1
F	1	0	1	1	1	1	0	1	1	1	1	0	0	0	1	1
F#/Gb	1	0	1	1	1	1	0	1	1	1	0	0	0	0	1	1
G	1	0	1	1	1	1	1	0	1	1	1	0	0	0	1	1
G#/Ab	1	0	1	1	1	1	1	0	1	1	0	0	0	0	1	1
A	1	0	1	1	1	1	1	1	0	1	1	0	0	0	1	1
A#/Bb	1	0	1	1	1	1	1	1	0	1	0	0	0	0	1	1
B	1	0	1	1	1	1	1	1	1	0	1	0	0	0	1	1

ตารางที่ 5-2 แสดงค่าที่วัดได้จากวงจรจากตัวโน้ตความถี่ที่ 127.2 ถึง 254.285(Hz)

จากตารางที่ 5-2 ค่าของโน้ตต่างๆจะแยกค่าได้ดังนี้

โน้ตตัว C (โด) จากตารางมีค่า 1010111111100011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ AFE3 เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว C ที่ความถี่ 130.81 (Hz)

โน้ตตัว C#/Db จากตารางมีค่า 1010111111000011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ AFC3 เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว C#/Db ที่ความถี่ 138.59 (Hz)

โน้ตตัว D (เร) จากตารางมีค่า 1011011111100011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ B7E3

เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว D ที่ความถี่ 146.83 (Hz)

โน้ตตัว D#/Eb จากตารางมีค่า 1011011111000011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ B7C3

เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว D#/Eb ที่ความถี่ 155.56 (Hz)

โน้ตตัว E (มี) จากตารางมีค่า 1011101111100011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BBE3

เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว E ที่ความถี่ 164.81 (Hz)

โน้ตตัว F (ฟา) จากตารางมีค่า 1011110111100011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BDE3

เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว F ที่ความถี่ 174.61 (Hz)

โน้ตตัว F#/Gb จากตารางมีค่า 1011110111000011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BDC3

เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว F#/Gb ที่ความถี่ 185.00 (Hz)

โน้ตตัว G(ซอล)จากตารางมีค่า 1011111011100011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BEE3

เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว G ที่ความถี่ 196.00 (Hz)

โน้ตตัว G#/Ab จากตารางมีค่า 1011111011000011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BEC3

เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว G# /Ab ที่ความถี่ 207.65 (Hz)

โน้ตตัว A(ลา) จากตารางมีค่า 1011111101100011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BF63

เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว A ที่ความถี่ 220.00 (Hz)

โน้ตตัว A#/Bb จากตารางมีค่า 1011111101000011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BF43

เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว A#/Bb ที่ความถี่ 233.08 (Hz)

โน้ตตัว B(ที) จากตารางมีค่า 1011111110100011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BFA3

เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว B ที่ความถี่ 246.94 (Hz)

ชื่อโน้ตดนตรี	b	Tuner	#	C	D	E	F	G	A	B	#	G1	G2	G3	used	used
C	1	0	1	0	1	1	1	1	1	1	1	0	0	1	1	1
C#/Db	1	0	1	0	1	1	1	1	1	1	0	0	0	1	1	1
D	1	0	1	1	0	1	1	1	1	1	1	0	0	1	1	1
D#/Eb	1	0	1	1	0	1	1	1	1	1	0	0	0	1	1	1
E	1	0	1	1	1	0	1	1	1	1	1	0	0	1	1	1
F	1	0	1	1	1	1	0	1	1	1	1	0	0	1	1	1

F#/Gb	1	0	1	1	1	1	0	1	1	1	0	0	0	1	1	1
G	1	0	1	1	1	1	1	0	1	1	1	0	0	1	1	1
G#/Ab	1	0	1	1	1	1	1	0	1	1	0	0	0	1	1	1
A	1	0	1	1	1	1	1	1	0	1	1	0	0	1	1	1
A#/Bb	1	0	1	1	1	1	1	1	0	1	0	0	0	1	1	1
B	1	0	1	1	1	1	1	1	1	0	1	0	0	1	1	1

ตารางที่ 5- 3 แสดงค่าที่วัดได้จากวงจรจากตัวโน้ตความถี่ที่ 261.63 ถึง 508.56(Hz)

จากตารางที่ 5-3 ค่าของโน้ตต่างๆจะแยกค่าได้ดังนี้

โน้ตตัว C (โด) จากตารางมีค่า 1010111111100111 ซึ่งแปลงเป็นเลขฐานสิบหกคือ AFE7 เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว C ที่ความถี่ 261.63 (Hz)

โน้ตตัว C#/Db จากตารางมีค่า 1010111111000111 ซึ่งแปลงเป็นเลขฐานสิบหกคือ AFC7 เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว C#/Db ที่ความถี่ 277.18 (Hz)

โน้ตตัว D (เร) จากตารางมีค่า 1011011111100111 ซึ่งแปลงเป็นเลขฐานสิบหกคือ B7E7 เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว D ที่ความถี่ 293.66 (Hz)

โน้ตตัว D#/Eb จากตารางมีค่า 1011011111000111 ซึ่งแปลงเป็นเลขฐานสิบหกคือ B7C7 เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว D#/Eb ที่ความถี่ 311.13 (Hz)

โน้ตตัว E (มี) จากตารางมีค่า 1011101111100111 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BBE7 เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว E ที่ความถี่ 329.63 (Hz)

โน้ตตัว F (ฟา) จากตารางมีค่า 1011110111100111 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BDE7 เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว F ที่ความถี่ 349.23 (Hz)

โน้ตตัว F#/Gb จากตารางมีค่า 1011110111000111 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BDC7 เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว F#/Gb ที่ความถี่ 369.99 (Hz)

โน้ตตัว G(ซอล)จากตารางมีค่า 1011111011100111 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BEE7 เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว G ที่ความถี่ 391.99 (Hz)

โน้ตตัว G#/Ab จากตารางมีค่า 1011111011000111 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BEC7 เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว G# /Ab ที่ความถี่ 415.31 (Hz)

โน้ตตัว A(ลา) จากตารางมีค่า 1011111101100111 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BF67 เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว A ที่ความถี่ 440.00 (Hz)

โน้ตตัว A#/Bb จากตารางมีค่า 1011111101000111 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BF47 เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว A#/Bb ที่ความถี่ 466.16 (Hz)

โน้ตตัว B(ที) จากตารางมีค่า 1011111110100111 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BFA7 เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว B ที่ความถี่ 493.88 (Hz)

ชื่อโน้ตดนตรี	b	Tuner	#	C	D	E	F	G	A	B	#	G1	G2	G3	used	used
C	1	0	1	0	1	1	1	1	1	1	1	0	1	0	1	1
C#/Db	1	0	1	0	1	1	1	1	1	1	0	0	1	0	1	1
D	1	0	1	1	0	1	1	1	1	1	1	0	1	0	1	1
D#/Eb	1	0	1	1	0	1	1	1	1	1	0	0	1	0	1	1
E	1	0	1	1	1	0	1	1	1	1	1	0	1	0	1	1
F	1	0	1	1	1	1	0	1	1	1	1	0	1	0	1	1
F#/Gb	1	0	1	1	1	1	0	1	1	1	0	0	1	0	1	1
G	1	0	1	1	1	1	1	0	1	1	1	0	1	0	1	1
G#/Ab	1	0	1	1	1	1	1	0	1	1	0	0	1	0	1	1
A	1	0	1	1	1	1	1	1	0	1	1	0	1	0	1	1
A#/Bb	1	0	1	1	1	1	1	1	0	1	0	0	1	0	1	1
B	1	0	1	1	1	1	1	1	1	0	1	0	1	0	1	1

ตารางที่ 5-4 แสดงค่าที่วัดได้จากวงจรจากตัวโน้ตความถี่ที่ 508.6 ถึง 1017.135(Hz)

จากตารางที่ 5-4 ค่าของโน้ตต่างๆจะแยกค่าได้ดังนี้

โน้ตตัว C (โด) จากตารางมีค่า 1010111111101011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ AFEB เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว C ที่ความถี่ 523.25 (Hz)

โน้ตตัว C#/Db จากตารางมีค่า 1010111111001011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ AFCB เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว C#/Db ที่ความถี่ 554.37 (Hz)

โน้ตตัว D (เร) จากตารางมีค่า 1011011111101011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ B7EB

เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว D ที่ความถี่ 587.33 (Hz)

โน้ตตัว D#/Eb จากตารางมีค่า 1011011111001011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ B7CB เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว D#/Eb ที่ความถี่ 622.25 (Hz)

โน้ตตัว E (มี) จากตารางมีค่า 1011101111101011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BBEB เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว E ที่ความถี่ 659.26 (Hz)

โน้ตตัว F (ฟา) จากตารางมีค่า 1011110111101011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BDEB เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว F ที่ความถี่ 698.46 (Hz)

โน้ตตัว F#/Gb จากตารางมีค่า 1011110111001011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BDCB เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว F#/Gb ที่ความถี่ 739.99 (Hz)

โน้ตตัว G(ซอล)จากตารางมีค่า 1011111011101011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BEEB เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว G ที่ความถี่ 783.99 (Hz)

โน้ตตัว G#/Ab จากตารางมีค่า 1011111011001011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BECB เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว G# /Ab ที่ความถี่ 830.61 (Hz)

โน้ตตัว A(ลา) จากตารางมีค่า 1011111101101011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BF6B เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว A ที่ความถี่ 880.00 (Hz)

โน้ตตัว A#/Bb จากตารางมีค่า 1011111101001011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BF4B เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว A#/Bb ที่ความถี่ 932.33 (Hz)

โน้ตตัว B(ที) จากตารางมีค่า 1011111110101011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BFAB เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว B ที่ความถี่ 987.77 (Hz)

ชื่อโน้ตดนตรี	b	Tuner	#	C	D	E	F	G	A	B	#	G1	G2	G3	used	used
C	1	0	1	0	1	1	1	1	1	1	1	0	1	1	1	1
C#/Db	1	0	1	0	1	1	1	1	1	1	0	0	1	1	1	1
D	1	0	1	1	0	1	1	1	1	1	1	0	1	1	1	1
D#/Eb	1	0	1	1	0	1	1	1	1	1	0	0	1	1	1	1
E	1	0	1	1	1	0	1	1	1	1	1	0	1	1	1	1
F	1	0	1	1	1	1	0	1	1	1	1	0	1	1	1	1
F#/Gb	1	0	1	1	1	1	0	1	1	1	0	0	1	1	1	1

G	1	0	1	1	1	1	1	0	1	1	1	0	1	1	1	1
G#/Ab	1	0	1	1	1	1	1	0	1	1	0	0	1	1	1	1
A	1	0	1	1	1	1	1	1	0	1	1	0	1	1	1	1
A#/Bb	1	0	1	1	1	1	1	1	0	1	0	0	1	1	1	1
B	1	0	1	1	1	1	1	1	1	0	1	0	1	1	1	1

ตารางที่ 5-5 แสดงค่าที่วัดได้จากวงจรจากตัวโน้ตความถี่ที่ 1017.3 ถึง 2034.25(Hz)

จากตารางที่ 5-5 ค่าของโน้ตต่างๆจะแยกค่าได้ดังนี้

โน้ตตัว C (โด) จากตารางมีค่า 1010111111101111 ซึ่งแปลงเป็นเลขฐานสิบหกคือ AFEF เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว C ที่ความถี่ 1046.5 (Hz)

โน้ตตัว C#/Db จากตารางมีค่า 1010111111001111 ซึ่งแปลงเป็นเลขฐานสิบหกคือ AFCF เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว C#/Db ที่ความถี่ 1108.7 (Hz)

โน้ตตัว D (เร) จากตารางมีค่า 1011011111101111 ซึ่งแปลงเป็นเลขฐานสิบหกคือ B7EF เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว D ที่ความถี่ 1174.7 (Hz)

โน้ตตัว D#/Eb จากตารางมีค่า 1011011111001111 ซึ่งแปลงเป็นเลขฐานสิบหกคือ B7CF เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว D#/Eb ที่ความถี่ 1244.5 (Hz)

โน้ตตัว E (มี) จากตารางมีค่า 1011101111101111 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BBEF เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว E ที่ความถี่ 1318.5 (Hz)

โน้ตตัว F (ฟา) จากตารางมีค่า 1011110111101111 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BDEF เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว F ที่ความถี่ 1396.9 (Hz)

โน้ตตัว F#/Gb จากตารางมีค่า 1011110111001111 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BDCF เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว F#/Gb ที่ความถี่ 1480.0 (Hz)

โน้ตตัว G(ซอล)จากตารางมีค่า 1011111011101111 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BEEF เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว G ที่ความถี่ 1568.0 (Hz)

โน้ตตัว G#/Ab จากตารางมีค่า 1011111011001111 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BECF เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว G# /Ab ที่ความถี่ 1661.2 (Hz)

โน้ตตัว A(ลา) จากตารางมีค่า 1011111101101111 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BF6F เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว A ที่ความถี่ 1760.0 (Hz)

โน้ตตัว A#/Bb จากตารางมีค่า 1011111101001111 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BF4F เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว A#/Bb ที่ความถี่ 1864.7 (Hz)

โน้ตตัว B(ที) จากตารางมีค่า 1011111110101111 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BFAF เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว B ที่ความถี่ 1975.5 (Hz)

ชื่อโน้ตดนตรี	b	Tuner	#	C	D	E	F	G	A	B	#	G1	G2	G3	used	used
C	1	0	1	0	1	1	1	1	1	1	1	1	0	0	1	1
C#/Db	1	0	1	0	1	1	1	1	1	1	0	1	0	0	1	1
D	1	0	1	1	0	1	1	1	1	1	1	1	0	0	1	1
D#/Eb	1	0	1	1	0	1	1	1	1	1	0	1	0	0	1	1
E	1	0	1	1	1	0	1	1	1	1	1	1	0	0	1	1
F	1	0	1	1	1	1	0	1	1	1	1	1	0	0	1	1
F#/Gb	1	0	1	1	1	1	0	1	1	1	0	1	0	0	1	1
G	1	0	1	1	1	1	1	0	1	1	1	1	0	0	1	1
G#/Ab	1	0	1	1	1	1	1	0	1	1	0	1	0	0	1	1
A	1	0	1	1	1	1	1	1	0	1	1	1	0	0	1	1
A#/Bb	1	0	1	1	1	1	1	1	0	1	0	1	0	0	1	1
B	1	0	1	1	1	1	1	1	1	0	1	1	0	0	1	1

ตารางที่ 5-6 แสดงค่าที่วัดได้จากวงจรจากตัวโน้ตความถี่ที่ 2034.3 ถึง 4186.0(Hz)

จากตารางที่ 5-6 ค่าของโน้ตต่างๆจะแยกค่าได้ดังนี้

โน้ตตัว C (โด) จากตารางมีค่า 1010111111110011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ AFF3 เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว C ที่ความถี่ 2093.0 (Hz)

โน้ตตัว C#/Db จากตารางมีค่า 1010111111010011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ AFD3

เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว C#/Db ที่ความถี่ 2217.5 (Hz)
โน้ตตัว D (เร) จากตารางมีค่า 1011011111110011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ B7F3
เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว D ที่ความถี่ 2349.3 (Hz)
โน้ตตัว D#/Eb จากตารางมีค่า 1011011111010011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ B7D3
เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว D#/Eb ที่ความถี่ 2489.0 (Hz)
โน้ตตัว E (มี) จากตารางมีค่า 1011101111110011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BBF3
เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว E ที่ความถี่ 2637.0 (Hz)
โน้ตตัว F (ฟา) จากตารางมีค่า 1011110111110011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BDF3
เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว F ที่ความถี่ 2793.8 (Hz)
โน้ตตัว F#/Gb จากตารางมีค่า 1011110111010011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BDD3
เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว F#/Gb ที่ความถี่ 2960.0 (Hz)
โน้ตตัว G(ซอล)จากตารางมีค่า 1011111011110011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BEF3
เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว G ที่ความถี่ 3136.0 (Hz)
โน้ตตัว G#/Ab จากตารางมีค่า 1011111011010011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BED3
เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว G# /Ab ที่ความถี่ 3322.4 (Hz)
โน้ตตัว A(ลา) จากตารางมีค่า 1011111101110011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BF73
เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว A ที่ความถี่ 3520.0 (Hz)
โน้ตตัว A#/Bb จากตารางมีค่า 1011111101010011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BF53
เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว A#/Bb ที่ความถี่ 3729.3 (Hz)
โน้ตตัว B(ที) จากตารางมีค่า 1011111110100011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BFA3
เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว B ที่ความถี่ 3951.1 (Hz)

จากตารางด้านบนนี้เป็นoutputจากฮาร์ดแวร์ที่ได้ทำการทดลองประมวลความถี่ของตัวโน้ตต่างๆซึ่งในฮาร์ดแวร์ตัวนี้จะรวมย่านต่างๆของโน้ตแต่ละ Octave เข้าไว้ด้วยกันโดยให้แสดงผลรวมเป็นโน้ตตัวเดียวก่อนเพื่อจะได้สะดวกต่อการนำผลไปทดลองกับส่วนประมวลผลอื่นๆ

จากตารางจะอธิบายได้ดังนี้

จากโน้ตตัว โด(C) จะมี Logic 0 ที่ Tuner กับ C เท่านั้นแสดงว่าเป็นโน้ตตัวโด(C)ที่ได้มาตรฐานถ้ามี Logic 0 ในช่อง b (แฟลต) หรือ # (ชาร์ป) แสดงว่ายังไม่ได้มาตรฐาน ถ้ามี Logic 0 ที่ b(แฟลต) แสดงว่าให้เพิ่มความถี่ขึ้นไป ถ้ามี Logic 0 ที่ # (ชาร์ป) แสดงว่าให้ลดความถี่ลง

ในโน้ต C# หรือ Db ซึ่งเป็นโน้ตที่มีความถี่ที่อยู่ในย่านเดียวกันจากตารางด้านบนนั้นจะมี Logic 0 ติดที่ Tuner, C และ # (ทีอยู่ด้านหลัง)เท่านั้น แสดงว่าเป็นความถี่ที่ได้

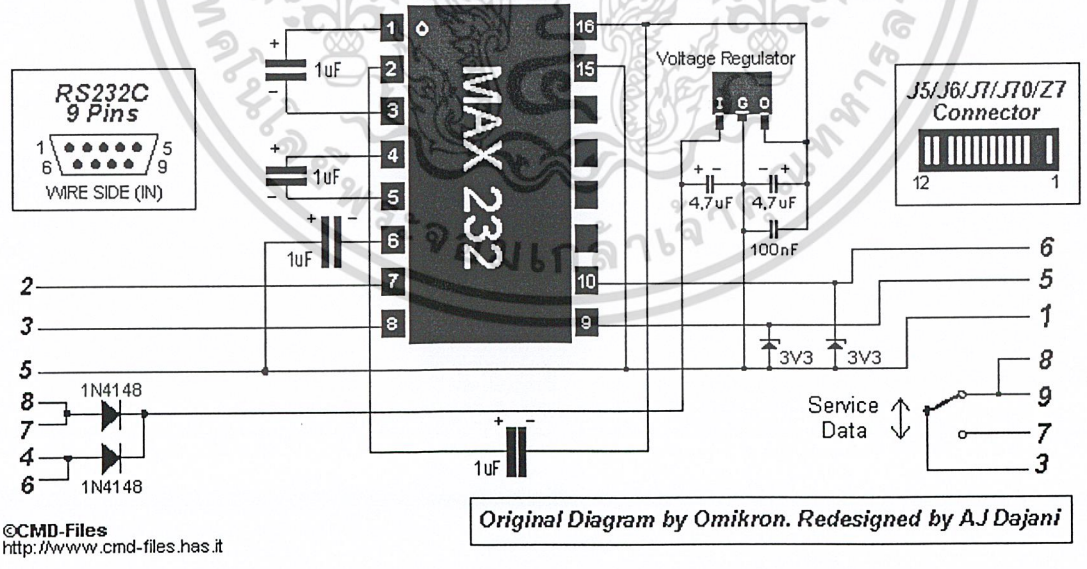
มาตรฐานถ้ามี Logic 0 ในช่อง b (แฟลต) หรือ # (ชาร์ป) แสดงว่ายังไม่ได้มาตรฐาน ถ้ามี Logic 0 ที่ b(แฟลต) แสดงว่าให้เพิ่มความถี่ขึ้นไป ถ้ามี Logic 0 ที่ # (ชาร์ป) แสดงว่าให้ลดความถี่ลง

จากตารางที่ 5-2 ถึงตารางที่ 5-6 นั้นจะเป็นการยกตัวอย่างการนำเอาความถี่ของตัวโน้ตในความถี่ต่างๆป้อนเข้าไปในวงจรแยกตัวโน้ตแล้วนำค่าที่ได้นั้นแปลงเป็นเลขฐานสิบหก โดยใช้ MCS-51 เป็นตัวจัดการแปลงค่า

เมื่อทำการแปลงสัญญาณจากไบนารีให้เป็นเลขฐานสิบหกแล้วนั้น ก็จะส่งข้อมูลใหม่ที่ได้จากการแปลงของ MCS-51 ส่งให้กับชุดวงจรอินเทอร์เฟสเพื่อส่งข้อมูลให้กับเครื่องคอมพิวเตอร์

3. ชุดวงจรอินเทอร์เฟส

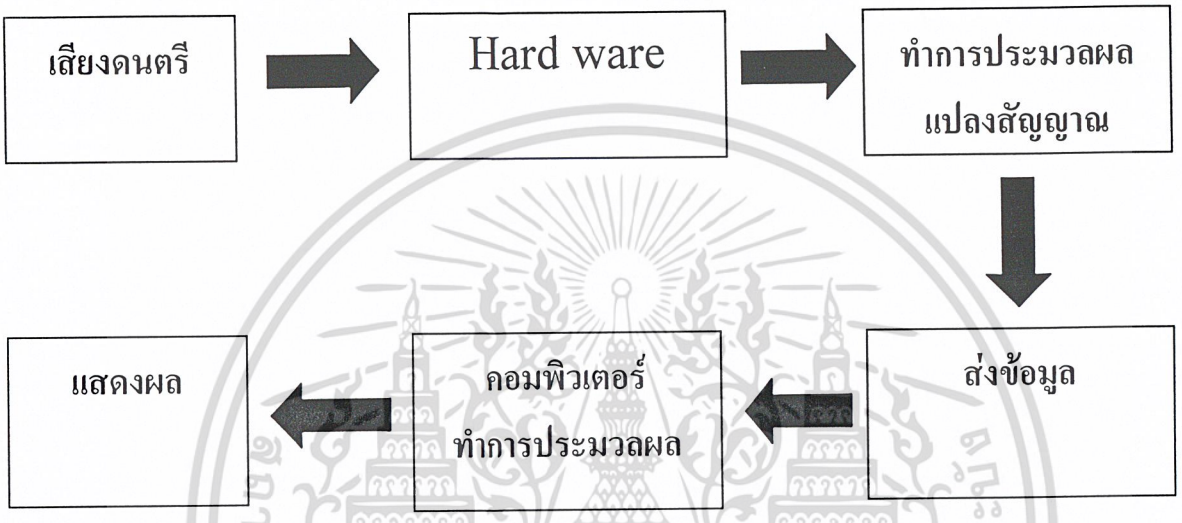
ในโครงงานชุดนี้เราจะใช้ IC สำเร็จรูปยี่ห้อ MAX232 มาใช้ในการอินเทอร์เฟสระหว่าง เครื่องคอมพิวเตอร์กับชุดฮาร์ดแวร์ที่ทำการออกแบบมา โดยวงจรที่ออกแบบมาเพื่อใช้ในการเชื่อมต่อ โดยใช้ IC MAX232 ซึ่งได้ทดลองแล้วและได้ทำการทดลองโดยใช้โปรแกรมสำเร็จรูปปรากฏว่าสามารถติดต่อกันได้โดยในขั้นตอนแรกๆจะทำการทดลองโดยติดต่อจาก COM1 และ COM2 เพื่อให้เข้าใจในการติดต่อกันก่อนแล้วจึงทดลองกับวงจร



รูปที่ 5-5 แสดงภาพวงจรMax 232

จากรูปด้านบนเป็นรูปวงจรที่ได้ทำการต่อทดลองแต่ยังไม่ได้นำมาต่อรวมกับส่วนที่ประมวลผลความถี่และส่วนวงจรจ่ายไฟ ซึ่งทางผู้จัดทำได้ทำการทดลองทีละวงจร

จากที่ได้กล่าวมาข้างต้น สามารถนำขั้นตอนการทำงานโดยรวมของโครงงาน มาเขียนเป็นโฟลว์ชาร์ตได้ดังนี้

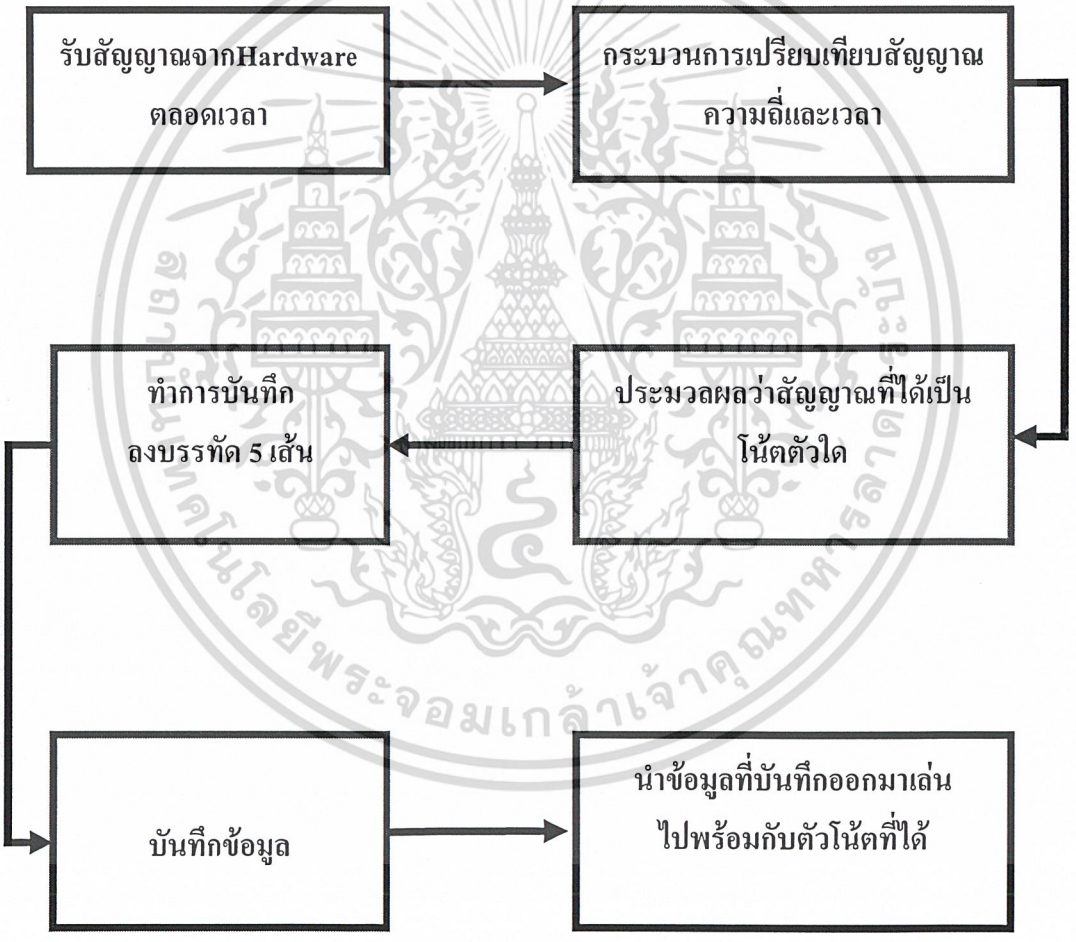


รูปที่ 5-6 แสดงขั้นตอนการทำงานของโครงงาน

จากนั้นทำการเขียนโปรแกรมเพื่อรับค่าที่ได้จากฮาร์ดแวร์เพื่อทดสอบการรับส่งค่าที่ได้ โดยในขั้นตอนแรกใช้โปรแกรม Hyper Terminal ในการติดต่อสื่อสารระหว่างตัวฮาร์ดแวร์กับเครื่องคอมพิวเตอร์ เพื่อทดสอบว่ารับส่งข้อมูลได้ถูกต้องหรือไม่ หลังจากนั้นก็พิจารณาค่าที่ได้ แล้วนำมาเขียนโปรแกรมเพื่อให้รองรับกับค่าที่ตัวฮาร์ดแวร์ส่งมาให้กับเครื่องคอมพิวเตอร์

5.4 Software Design

ในโครงงานชิ้นนี้มีอุปกรณ์ฮาร์ดแวร์เป็นตัวส่งข้อมูล เพราะฉะนั้นในส่วนของ Software นั้นจะต้องคอยรับค่าของฮาร์ดแวร์ได้ตลอดเวลา เมื่อได้ค่าที่มาจากฮาร์ดแวร์แล้วนั้นก็จะเข้าสู่กระบวนการเปรียบเทียบค่าที่ได้และเวลาที่รับสัญญาณนั้น เพื่อเปรียบเทียบว่าสัญญาณที่ได้นั้นเป็นโน้ตอะไรและมีความเร็วเท่าไร แล้วนำมาเปรียบเทียบกับเวลามาตรฐานจริงที่ได้กำหนดขึ้นมาเพื่อนับจังหวะของโน้ตดนตรี ต่อมาจะเข้าสู่การประมวลผลสัญญาณ เพื่อเปรียบเทียบว่าเป็นโน้ตตัวกลม,ตัวขาว,ตัวดำ หรือตัวเขบ็ต จากนั้นนำค่าที่ได้เป็นตัวโน้ตแล้วมาเขียนลงบนบรรทัดห้าเส้น พร้อมกับสามารถบันทึกข้อมูลที่ได้แล้วเล่นเพลงนั้นออกมาได้ด้วย สามารถนำขั้นตอนการทำงานโดยรวมของโปรแกรม มาเขียนเป็นโฟลว์ชาร์ตได้ดังนี้



รูปที่ 5-7 แสดงขั้นตอนการทำงานของโปรแกรม

บทที่ 6

ผลการทดลอง

6.1บทนำ

ในส่วนของการทดลองนั้นจะเป็นการนำเอาเสียงของโน้ตดนตรีมาทำการทดสอบกับ ฮาร์ดแวร์และโปรแกรมที่ได้จัดทำขึ้นมา เพื่อพิจารณาถึงความถูกต้องของฮาร์ดแวร์และโปรแกรมที่ได้ทำการจัดทำขึ้นมาว่ามีประสิทธิภาพในการทำงานแยกเสียงของตัวโน้ตต่างๆได้หรือไม่

6.2 ประสิทธิภาพของฮาร์ดแวร์

จากที่ได้กล่าวไปแล้วว่าวิธีการที่ได้นำมาใช้เพื่อทดสอบประสิทธิภาพในการแยกเสียงของตัวโน้ตนั้น จะใช้วงจรในการแยกเสียงของโน้ตดนตรีมาก่อนแล้วส่งค่าให้ไมโครคอนโทรลเลอร์ประมวลผล โดยแปลงให้เป็นเลขฐานสิบหกก่อนส่งให้คอมพิวเตอร์ประมวลผลผ่านทางพอร์ทอนุกรมซึ่งจะได้ค่าตามตารางที่ 6-1 ถึงตารางที่ 6-4

ชื่อโน้ตดนตรี	b	Tuner	#	C	D	E	F	G	A	B	#	G1	G2	G3	used	used
C	1	0	1	0	1	1	1	1	1	1	1	0	0	0	1	1
C#/Db	1	0	1	0	1	1	1	1	1	1	0	0	0	0	1	1
D	1	0	1	1	0	1	1	1	1	1	1	0	0	0	1	1
D#/Eb	1	0	1	1	0	1	1	1	1	1	0	0	0	0	1	1
E	1	0	1	1	1	0	1	1	1	1	1	0	0	0	1	1
F	1	0	1	1	1	1	0	1	1	1	1	0	0	0	1	1
F#/Gb	1	0	1	1	1	1	0	1	1	1	0	0	0	0	1	1
G	1	0	1	1	1	1	1	0	1	1	1	0	0	0	1	1
G#/Ab	1	0	1	1	1	1	1	0	1	1	0	0	0	0	1	1
A	1	0	1	1	1	1	1	1	0	1	1	0	0	0	1	1
A#/Bb	1	0	1	1	1	1	1	1	0	1	0	0	0	0	1	1
B	1	0	1	1	1	1	1	1	1	0	1	0	0	0	1	1

ตารางที่ 6-1 แสดงการแปลงค่าให้เป็นเลขฐานสิบหกจากตัวโน้ตความถี่ที่ 127.2 ถึง 254.285(Hz)

จากตารางที่ 6-1 ค่าของโน้ตต่างๆจะแยกค่าได้ดังนี้

โน้ตตัว C (โด) จากตารางมีค่า 1010111111100011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ AFE3
เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว C ที่ความถี่ 130.81 (Hz)

โน้ตตัว C#/Db จากตารางมีค่า 1010111111000011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ AFC3
เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว C#/Db ที่ความถี่ 138.59 (Hz)

โน้ตตัว D (เร) จากตารางมีค่า 1011011111100011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ B7E3
เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว D ที่ความถี่ 146.83 (Hz)

โน้ตตัว D#/Eb จากตารางมีค่า 1011011111000011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ B7C3
เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว D#/Eb ที่ความถี่ 155.56 (Hz)

โน้ตตัว E (มี) จากตารางมีค่า 1011101111100011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BBE3
เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว E ที่ความถี่ 164.81 (Hz)

โน้ตตัว F (ฟา) จากตารางมีค่า 1011110111100011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BDE3
เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว F ที่ความถี่ 174.61 (Hz)

โน้ตตัว F#/Gb จากตารางมีค่า 1011110111000011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BDC3
เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว F#/Gb ที่ความถี่ 185.00 (Hz)

โน้ตตัว G(ซอล)จากตารางมีค่า 1011111011100011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BEE3
เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว G ที่ความถี่ 196.00 (Hz)

โน้ตตัว G#/Ab จากตารางมีค่า 1011111011000011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BEC3
เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว G# /Ab ที่ความถี่ 207.65 (Hz)

โน้ตตัว A(ลา) จากตารางมีค่า 1011111101100011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BF63
เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว A ที่ความถี่ 220.00 (Hz)

โน้ตตัว A#/Bb จากตารางมีค่า 1011111101000011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BF43
เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว A#/Bb ที่ความถี่ 233.08 (Hz)

โน้ตตัว B(ที) จากตารางมีค่า 1011111110100011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BFA3
เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว B ที่ความถี่ 246.94 (Hz)

ชื่อโน้ตดนตรี	b	Tuner	#	C	D	E	F	G	A	B	#	G1	G2	G3	used	used
C	1	0	1	0	1	1	1	1	1	1	1	0	0	1	1	1
C#/Db	1	0	1	0	1	1	1	1	1	1	0	0	0	1	1	1
D	1	0	1	1	0	1	1	1	1	1	1	0	0	1	1	1
D#/Eb	1	0	1	1	0	1	1	1	1	1	0	0	0	1	1	1
E	1	0	1	1	1	0	1	1	1	1	1	0	0	1	1	1
F	1	0	1	1	1	1	0	1	1	1	1	0	0	1	1	1
F#/Gb	1	0	1	1	1	1	0	1	1	1	0	0	0	1	1	1
G	1	0	1	1	1	1	1	0	1	1	1	0	0	1	1	1
G#/Ab	1	0	1	1	1	1	1	0	1	1	0	0	0	1	1	1
A	1	0	1	1	1	1	1	1	0	1	1	0	0	1	1	1
A#/Bb	1	0	1	1	1	1	1	1	0	1	0	0	0	1	1	1
B	1	0	1	1	1	1	1	1	1	0	1	0	0	1	1	1

ตารางที่ 6-2 แสดงค่าที่วัดได้จากวงจรจากตัวโน้ตความถี่ที่ 261.63 ถึง 508.56(Hz)

จากตารางที่ 6-2 ค่าของโน้ตต่างๆจะแยกค่าได้ดังนี้

โน้ตตัว C (โด) จากตารางมีค่า 1010111111100111 ซึ่งแปลงเป็นเลขฐานสิบหกคือ AFE7 เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว C ที่ความถี่ 261.63 (Hz)

โน้ตตัว C#/Db จากตารางมีค่า 1010111111000111 ซึ่งแปลงเป็นเลขฐานสิบหกคือ AFC7 เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว C#/Db ที่ความถี่ 277.18 (Hz)

โน้ตตัว D (เร) จากตารางมีค่า 1011011111100111 ซึ่งแปลงเป็นเลขฐานสิบหกคือ B7E7 เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว D ที่ความถี่ 293.66 (Hz)

โน้ตตัว D#/Eb จากตารางมีค่า 1011011111000111 ซึ่งแปลงเป็นเลขฐานสิบหกคือ B7C7 เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว D#/Eb ที่ความถี่ 311.13 (Hz)

โน้ตตัว E (มี) จากตารางมีค่า 1011101111100111 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BBE7

เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว E ที่ความถี่ 329.63 (Hz)

โน้ตตัว F (ฟา) จากตารางมีค่า 1011110111100111 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BDE7

เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว F ที่ความถี่ 349.23 (Hz)

โน้ตตัว F#/Gb จากตารางมีค่า 1011110111000111 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BDC7

เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว F#/Gb ที่ความถี่ 369.99 (Hz)

โน้ตตัว G(ซอล)จากตารางมีค่า 1011111011100111 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BEE7

เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว G ที่ความถี่ 391.99 (Hz)

โน้ตตัว G#/Ab จากตารางมีค่า 1011111011000111 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BEC7

เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว G# /Ab ที่ความถี่ 415.31 (Hz)

โน้ตตัว A(ลา) จากตารางมีค่า 1011111101100111 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BF67

เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว A ที่ความถี่ 440.00 (Hz)

โน้ตตัว A#/Bb จากตารางมีค่า 1011111101000111 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BF47

เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว A#/Bb ที่ความถี่ 466.16 (Hz)

โน้ตตัว B(ที) จากตารางมีค่า 1011111110100111 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BFA7

เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว B ที่ความถี่ 493.88 (Hz)

ชื่อโน้ตดนตรี	b	Tuner	#	C	D	E	F	G	A	B	#	G1	G2	G3	used	used
C	1	0	1	0	1	1	1	1	1	1	1	0	1	0	1	1
C#/Db	1	0	1	0	1	1	1	1	1	1	0	0	1	0	1	1
D	1	0	1	1	0	1	1	1	1	1	1	0	1	0	1	1
D#/Eb	1	0	1	1	0	1	1	1	1	1	0	0	1	0	1	1
E	1	0	1	1	1	0	1	1	1	1	1	0	1	0	1	1
F	1	0	1	1	1	1	0	1	1	1	1	0	1	0	1	1
F#/Gb	1	0	1	1	1	1	0	1	1	1	0	0	1	0	1	1
G	1	0	1	1	1	1	1	0	1	1	1	0	1	0	1	1

G#/Ab	1	0	1	1	1	1	1	0	1	1	0	0	1	0	1	1
A	1	0	1	1	1	1	1	1	0	1	1	0	1	0	1	1
A#/Bb	1	0	1	1	1	1	1	1	0	1	0	0	1	0	1	1
B	1	0	1	1	1	1	1	1	1	0	1	0	1	0	1	1

ตารางที่ 6-3 แสดงค่าที่วัดได้จากวงจรจากตัวโน้ตความถี่ที่ 508.6 ถึง 1017.135(Hz)

จากตารางที่ 6-3 ค่าของโน้ตต่างๆจะแยกค่าได้ดังนี้

โน้ตตัว C (โด) จากตารางมีค่า 1010111111101011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ AFEB เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว C ที่ความถี่ 523.25 (Hz)

โน้ตตัว C#/Db จากตารางมีค่า 1010111111001011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ AFCB เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว C#/Db ที่ความถี่ 554.37 (Hz)

โน้ตตัว D (เร) จากตารางมีค่า 1011011111101011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ B7EB เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว D ที่ความถี่ 587.33 (Hz)

โน้ตตัว D#/Eb จากตารางมีค่า 1011011111001011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ B7CB เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว D#/Eb ที่ความถี่ 622.25 (Hz)

โน้ตตัว E (มี) จากตารางมีค่า 1011101111101011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BBEB เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว E ที่ความถี่ 659.26 (Hz)

โน้ตตัว F (ฟา) จากตารางมีค่า 1011110111101011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BDEB เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว F ที่ความถี่ 698.46 (Hz)

โน้ตตัว F#/Gb จากตารางมีค่า 1011110111001011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BDCB เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว F#/Gb ที่ความถี่ 739.99 (Hz)

โน้ตตัว G(ซอล)จากตารางมีค่า 1011111011101011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BEEB เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว G ที่ความถี่ 783.99 (Hz)

โน้ตตัว G#/Ab จากตารางมีค่า 1011111011001011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BECB เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว G# /Ab ที่ความถี่ 830.61 (Hz)

โน้ตตัว A(ลา) จากตารางมีค่า 1011111101101011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BF6B เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว A ที่ความถี่ 880.00 (Hz)

โน้ตตัว A#/Bb จากตารางมีค่า 1011111101001011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BF4B เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว A#/Bb ที่ความถี่ 932.33 (Hz)

โน้ตตัว B(ที) จากตารางมีค่า 1011111110101011 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BFAB เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว B ที่ความถี่ 987.77 (Hz)

ชื่อโน้ตดนตรี	b	Tuner	#	C	D	E	F	G	A	B	#	G1	G2	G3	used	used
C	1	0	1	0	1	1	1	1	1	1	1	0	1	1	1	1
C#/Db	1	0	1	0	1	1	1	1	1	1	0	0	1	1	1	1
D	1	0	1	1	0	1	1	1	1	1	1	0	1	1	1	1
D#/Eb	1	0	1	1	0	1	1	1	1	1	0	0	1	1	1	1
E	1	0	1	1	1	0	1	1	1	1	1	0	1	1	1	1
F	1	0	1	1	1	1	0	1	1	1	1	0	1	1	1	1
F#/Gb	1	0	1	1	1	1	0	1	1	1	0	0	1	1	1	1
G	1	0	1	1	1	1	1	0	1	1	1	0	1	1	1	1
G#/Ab	1	0	1	1	1	1	1	0	1	1	0	0	1	1	1	1
A	1	0	1	1	1	1	1	1	0	1	1	0	1	1	1	1
A#/Bb	1	0	1	1	1	1	1	1	0	1	0	0	1	1	1	1
B	1	0	1	1	1	1	1	1	1	0	1	0	1	1	1	1

ตารางที่ 6-4 แสดงค่าที่วัดได้จากวงจรจากตัวโน้ตความถี่ที่ 1017.3 ถึง 2034.25(Hz)

จากตารางที่ 6-4 ค่าของโน้ตต่างๆจะแยกค่าได้ดังนี้

โน้ตตัว C (โด) จากตารางมีค่า 1010111111101111 ซึ่งแปลงเป็นเลขฐานสิบหกคือ AFEF เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว C ที่ความถี่ 1046.5 (Hz)

โน้ตตัว C#/Db จากตารางมีค่า 1010111111001111 ซึ่งแปลงเป็นเลขฐานสิบหกคือ AFCF เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว C#/Db ที่ความถี่ 1108.7 (Hz)

โน้ตตัว D (เร) จากตารางมีค่า 1011011111101111 ซึ่งแปลงเป็นเลขฐานสิบหกคือ B7EF เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว D ที่ความถี่ 1174.7 (Hz)

โน้ตตัว D#/Eb จากตารางมีค่า 1011011111001111 ซึ่งแปลงเป็นเลขฐานสิบหกคือ B7CF เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว D#/Eb ที่ความถี่ 1244.5 (Hz)

โน้ตตัว E (มี) จากตารางมีค่า 1011101111101111 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BBEF เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว E ที่ความถี่ 1318.5 (Hz)

โน้ตตัว F (ฟา) จากตารางมีค่า 1011110111101111 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BDEF เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว F ที่ความถี่ 1396.9 (Hz)

โน้ตตัว F#/Gb จากตารางมีค่า 1011110111001111 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BDCF เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว F#/Gb ที่ความถี่ 1480.0 (Hz)

โน้ตตัว G(ซอล)จากตารางมีค่า 1011111011101111 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BEEF เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว G ที่ความถี่ 1568.0 (Hz)

โน้ตตัว G#/Ab จากตารางมีค่า 1011111011001111 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BECF เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว G# /Ab ที่ความถี่ 1661.2 (Hz)

โน้ตตัว A(ลา) จากตารางมีค่า 1011111101101111 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BF6F เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว A ที่ความถี่ 1760.0 (Hz)

โน้ตตัว A#/Bb จากตารางมีค่า 1011111101001111 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BF4F เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว A#/Bb ที่ความถี่ 1864.7 (Hz)

โน้ตตัว B(ที) จากตารางมีค่า 1011111110101111 ซึ่งแปลงเป็นเลขฐานสิบหกคือ BFAF เป็นค่าความถี่ที่ได้มาตรฐานของโน้ตตัว B ที่ความถี่ 1975.5 (Hz)

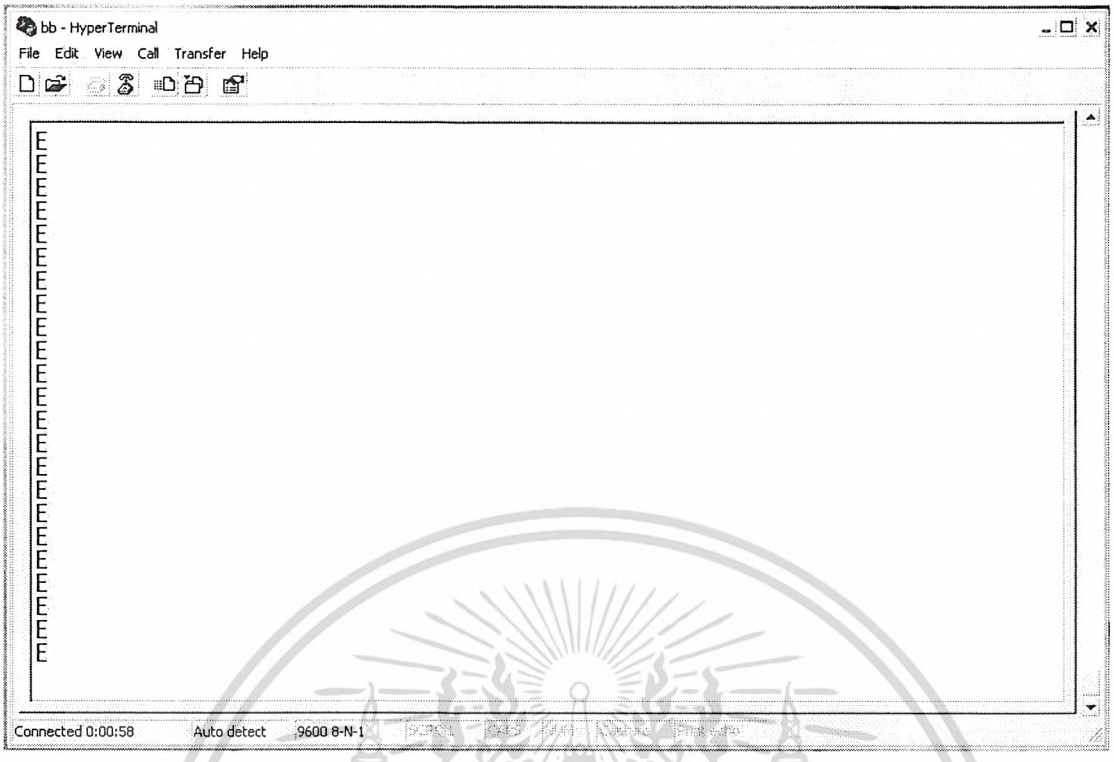
จากตารางด้านบนนี้เป็นoutputจากฮาร์ดแวร์ที่ได้ทำการทดลองประมวลความถี่ของตัวโน้ตต่างๆ ซึ่งในฮาร์ดแวร์ตัวนี้จะรวมย่านต่างๆของ โน้ตแต่ละ Octave เข้าไว้ด้วยกันโดยให้แสดงผลรวมเป็นโน้ตตัวเดียวก่อนเพื่อจะได้สะดวกต่อการนำผลไปทดลองกับส่วนประมวลผลอื่นๆ

จากตารางจะอธิบายได้ดังนี้

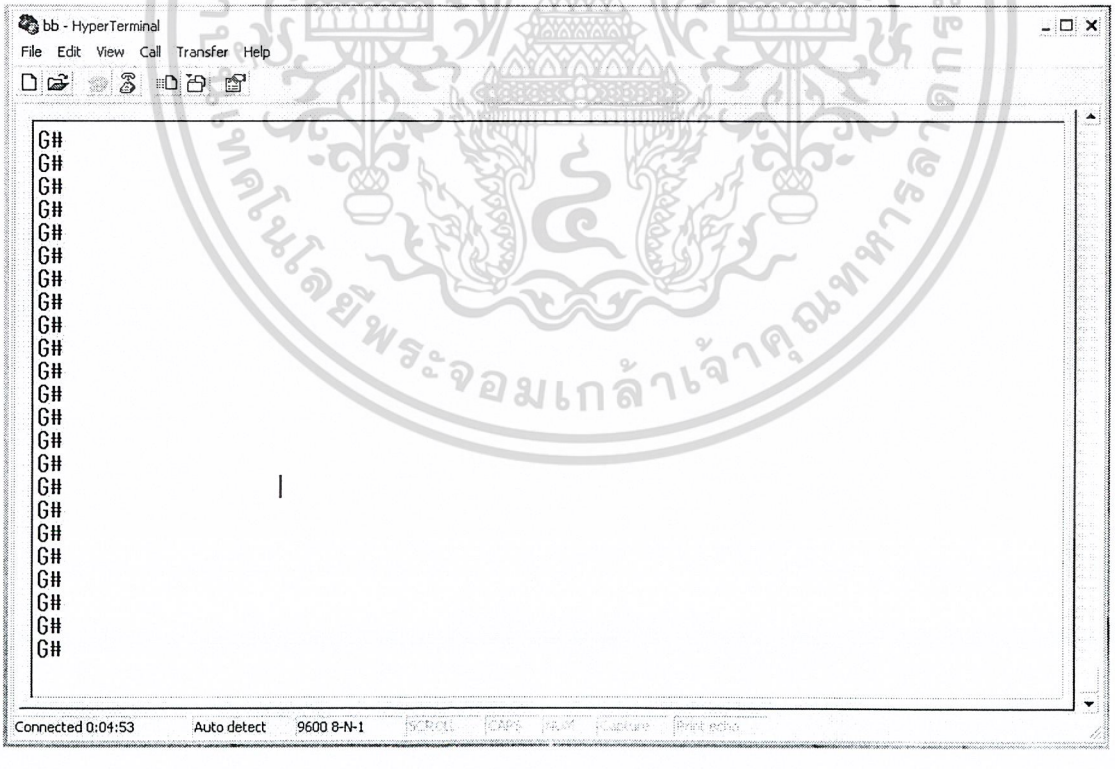
จากโน้ตตัว โด(C) จะมี Logic 0 ที่ Tuner กับ C เท่านั้นแสดงว่าเป็นโน้ตตัวโด (C)ที่ได้มาตรฐาน ถ้ามี Logic 0 ในช่อง b (แฟลต) หรือ # (ชาร์ป) แสดงว่ายังไม่ได้มาตรฐาน ถ้ามี Logic 0 ที่ b(แฟลต) แสดงว่าให้เพิ่มความถี่ขึ้นไป ถ้ามี Logic 0 ที่ # (ชาร์ป) แสดงว่าให้ลดความถี่ลง

ในโน้ต C# หรือ Db ซึ่งเป็นโน้ตที่มีความถี่ที่อยู่ในย่านเดียวกันจากตารางด้าน บนนั้นจะมี Logic 0 ติดที่ Tuner, C และ # (ทีอยู่ด้านหลัง)เท่านั้น แสดงว่าเป็นความถี่ที่ได้มาตร

จากนั้นทำการทดลองส่งค่าให้กับคอมพิวเตอร์โดยใช้โปรแกรม Hyper Terminal ในการสื่อสารระหว่างคอมพิวเตอร์กับฮาร์ดแวร์เพื่อพิจารณาค่าที่ได้ ดังรูปที่ 6-1และ 6-2

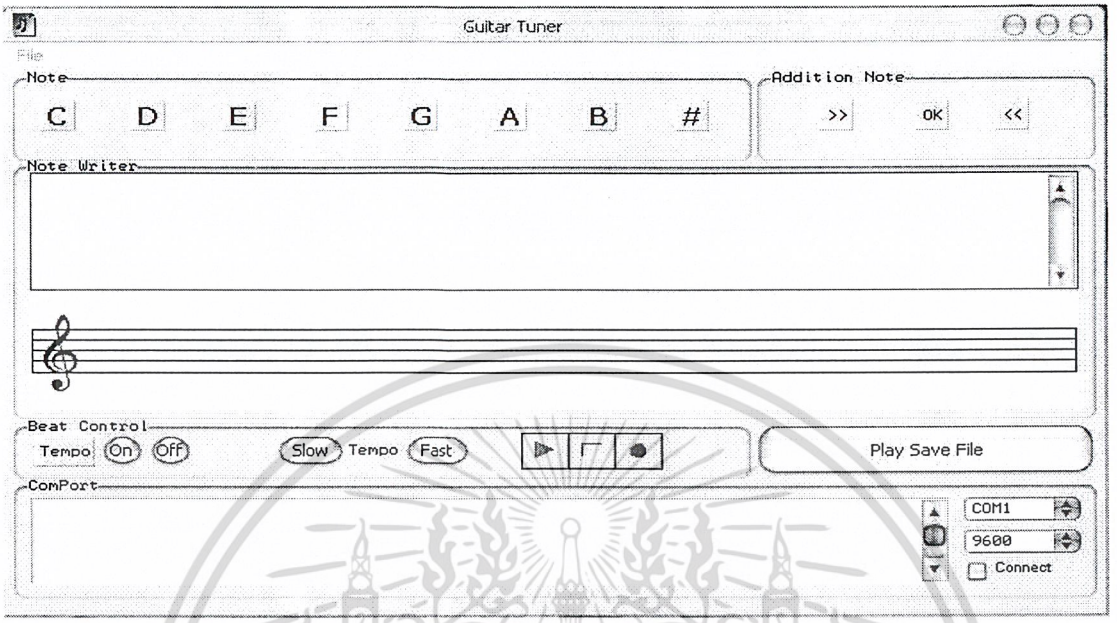


รูปที่6-1 แสดงการส่งค่าโน้ตตัวมี(E)ให้คอมพิวเตอร์



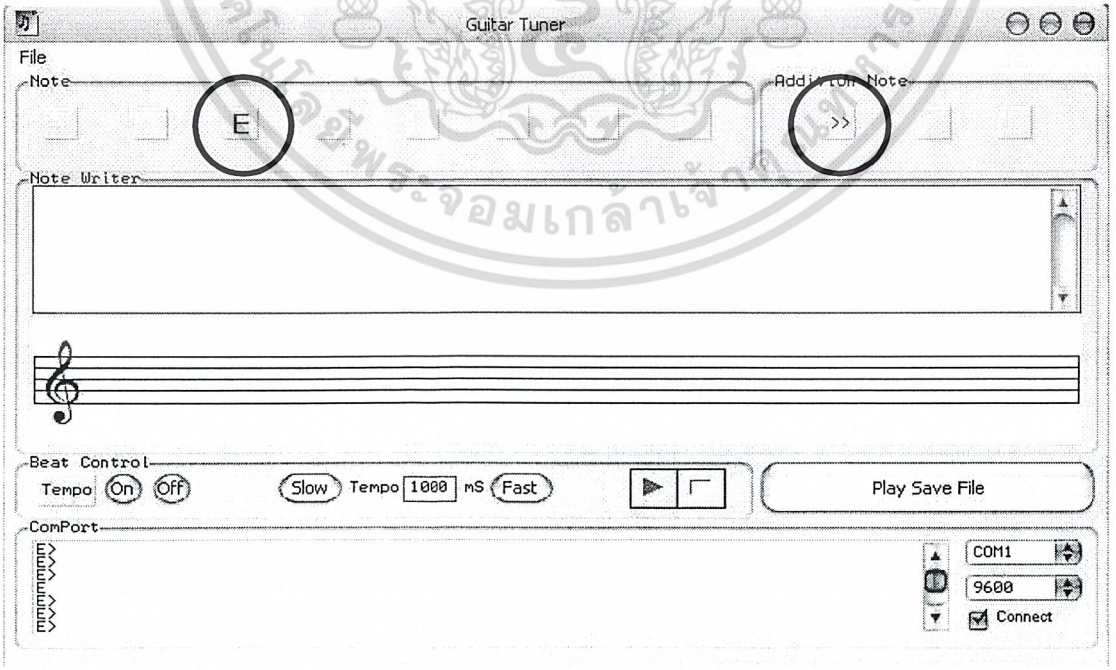
รูปที่6-2 แสดงการส่งค่าโน้ตตัวซอลชาร์ป(G#)ให้คอมพิวเตอร์

จากนั้นทำการทดลองเขียนโปรแกรมรับค่าจากฮาร์ดแวร์กับคอมพิวเตอร์เพื่อวิเคราะห์ค่าแล้วเขียนเป็นตัวโน้ต ซึ่งจะทำเป็นขั้นตอน ดังรูปที่ 6-3



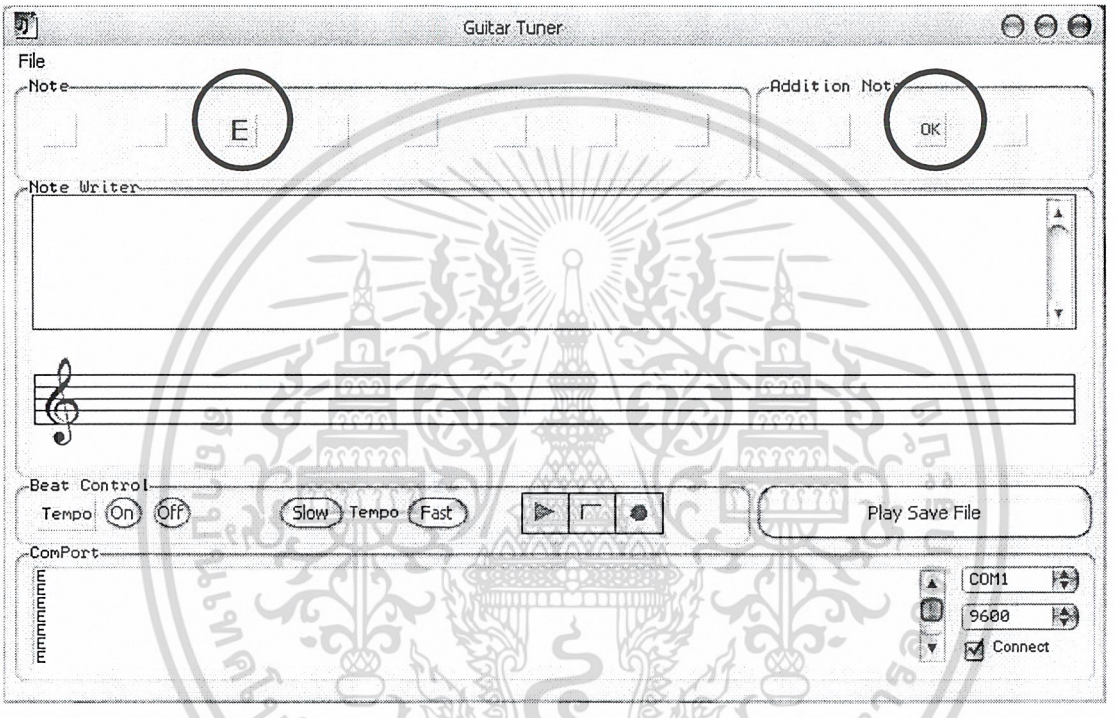
รูปที่6-3 แสดงหน้าตารูปโปรแกรมที่ได้จัดทำขึ้นมา

ขั้นตอนต่อมาทำการส่งค่าจากฮาร์ดแวร์ให้กับโปรแกรมเพื่อวิเคราะห์ว่าข้อมูลที่ส่งมานั้นเป็นโน้ตอะไรถูกต้องหรือไม่ ดังรูปที่ 6-4



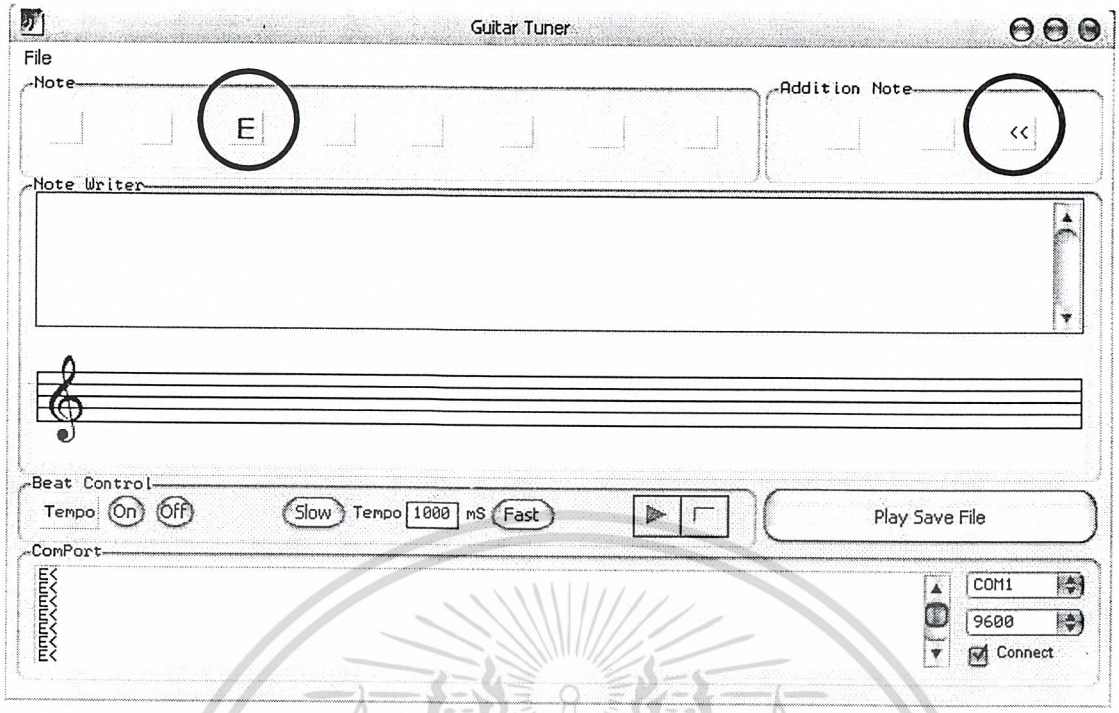
รูปที่6-4 แสดงค่าโน้ตตัวมี(E)ต่ำกว่ามาตรฐานที่ฮาร์ดแวร์ส่งให้กับโปรแกรม

จากรูปที่ 6-4 นั้น จะเป็นการส่งค่าโน้ตตัวมี(E) จากฮาร์ดแวร์ให้กับโปรแกรม ซึ่งจากในรูปที่ 6-4 นั้นจะต้องทำการปรับแต่งเสียงของเครื่องดนตรีให้ได้มาตรฐานก่อน จากรูปที่ 6-4 นั้นมีเครื่องหมายให้เพิ่มความถี่ขึ้นแสดงอยู่ด้วย แสดงว่าต้องเพิ่มความถี่ขึ้นไปอีก ถ้าเป็นกีตาร์ ก็ให้หมุนลูกบิดเพิ่มเสียงให้สูงขึ้นอีก จนกว่าจะมีเครื่องหมาย ok ติดขึ้นเพียงอย่างเดียว ดังรูปที่ 6-5



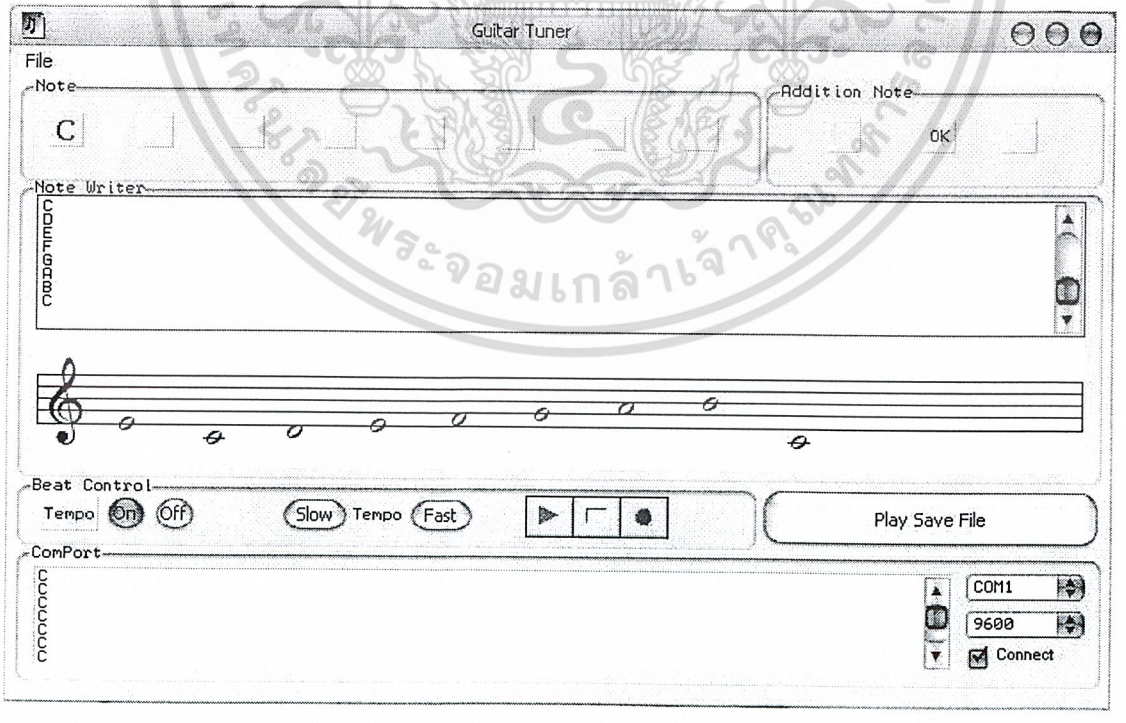
รูปที่6-5 แสดงการตั้งเสียงที่ตรงตามมาตรฐาน

จากรูปที่ 6-5 จะเป็นการตั้งสายหรือเครื่องดนตรีที่ได้ตรงตามมาตรฐาน แต่ถ้าเสียงของเครื่องดนตรีนั้นมีความถี่สูงกว่าเสียงโน้ตมาตรฐานโปรแกรมก็จะปรากฏเครื่องหมายให้ลดความถี่ลง แสดงว่าผู้เล่นจะต้องปรับความถี่ลดลงมา ถ้าเป็นกีตาร์ ก็ให้หมุนลูกบิดลดเสียงลง จนกว่าจะมีเครื่องหมาย ok ติดขึ้นเพียงอย่างเดียว ดังรูปที่ 6-6



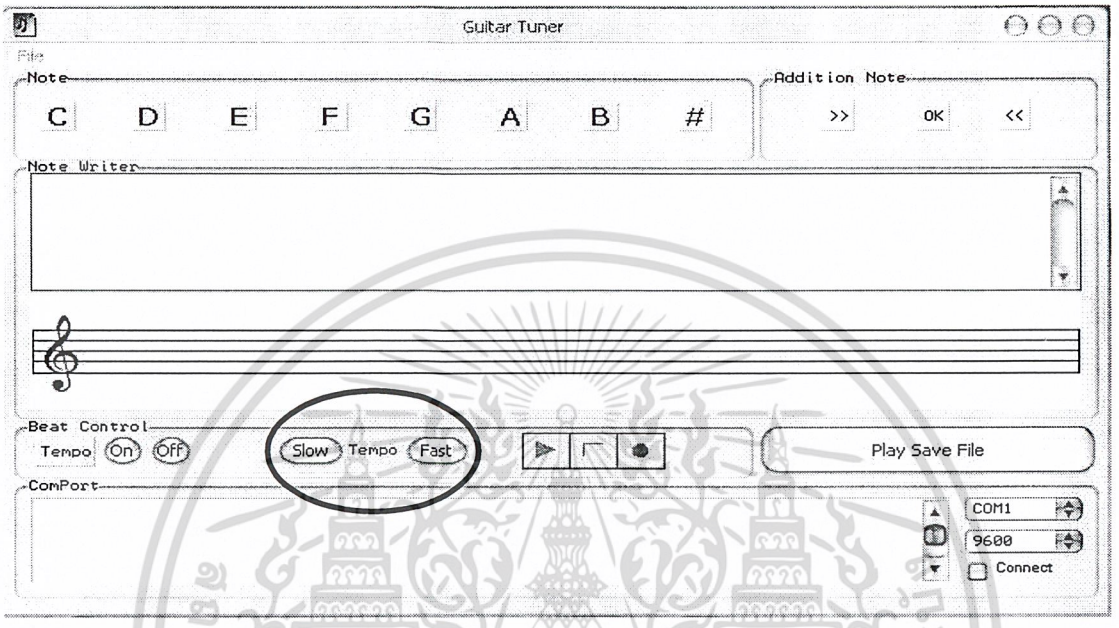
รูปที่6-6 แสดงค่าโน้ตตัวมี(E)สูงกว่ามาตรฐานที่ฮาร์ดแวร์ส่งให้กับโปรแกรม

ต่อมาเป็นการทดลองนำค่าที่ได้นั้นไปเปรียบเทียบตัวโน้ตและเวลา เพื่อแยกแยะว่าเป็นโน้ตตัวกลม,ตัวขาว,ตัวดำ หรือตัวเขบ็ต แล้วทำการเขียนลงบรรทัดห้าเส้น ดังรูปที่ 6-7

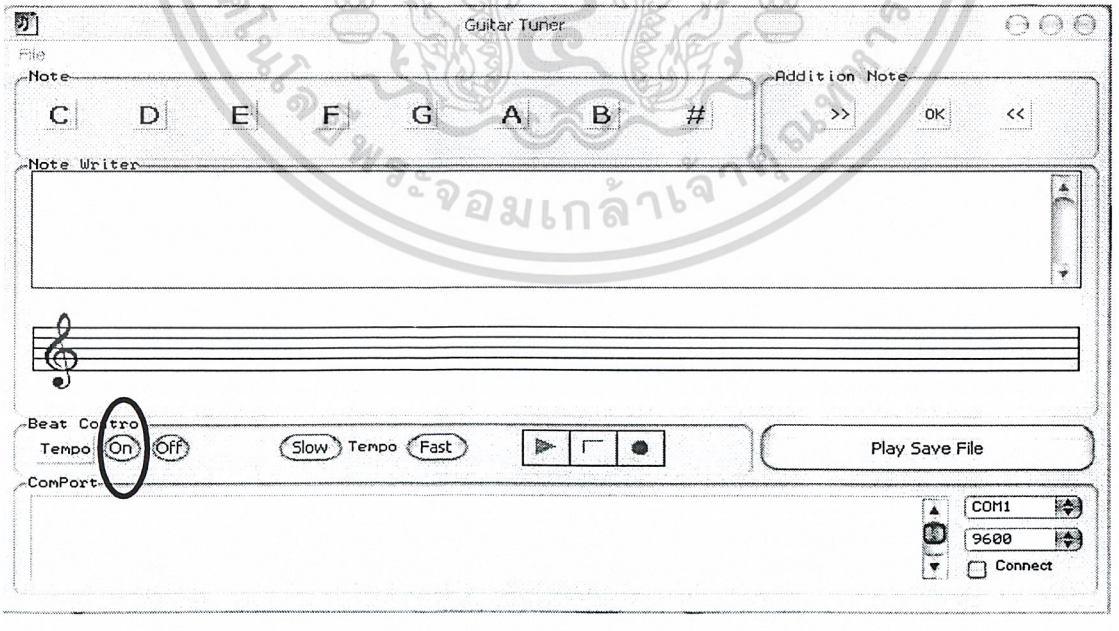


รูปที่ 6-7 แสดงการเขียนโน้ตที่ได้ลงบรรทัดห้าเส้น

ต่อมาเป็นขั้นตอนในการเลือกจังหวะ ในโปรแกรมนั้นผู้ใช้สามารถเลือกได้ว่าจะใช้จังหวะช้าหรือเร็วเท่าไรโดยสามารถกดเลือกจังหวะได้ตรงปุ่ม Slow และ Fast ดังรูปที่ 6-8 จากนั้นให้กดปุ่ม On เสียงของจังหวะก็จะดังขึ้นมา ดังรูปที่ 6-9

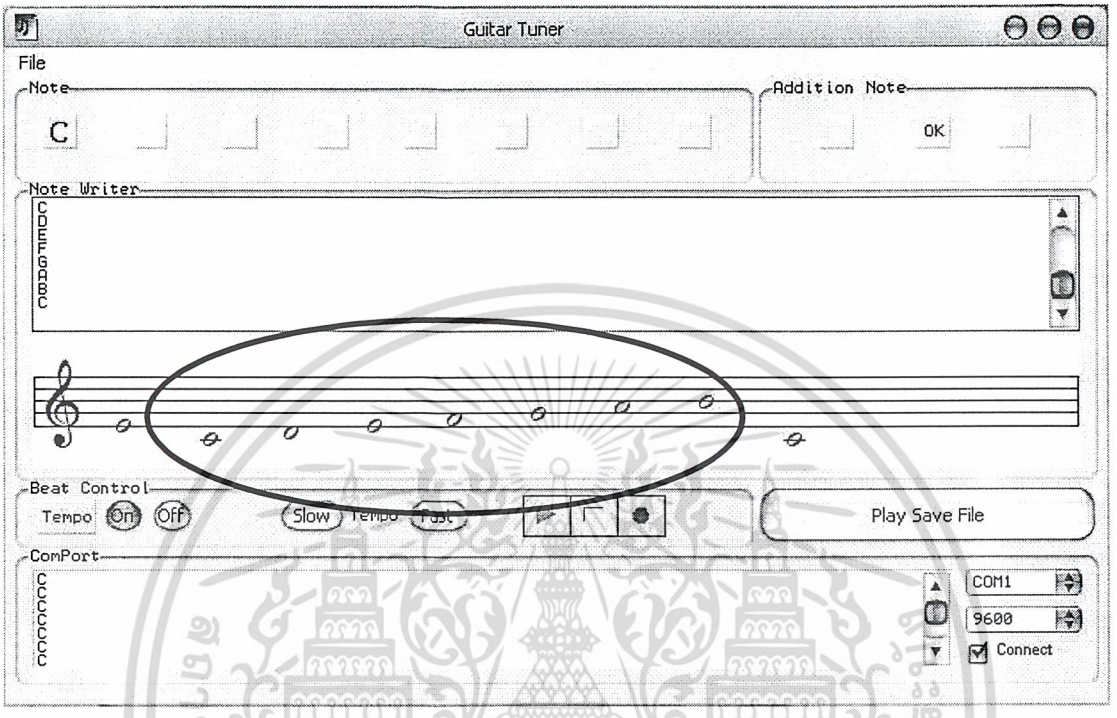


รูปที่ 6-8 แสดงปุ่มเลือกจังหวะในโปรแกรม



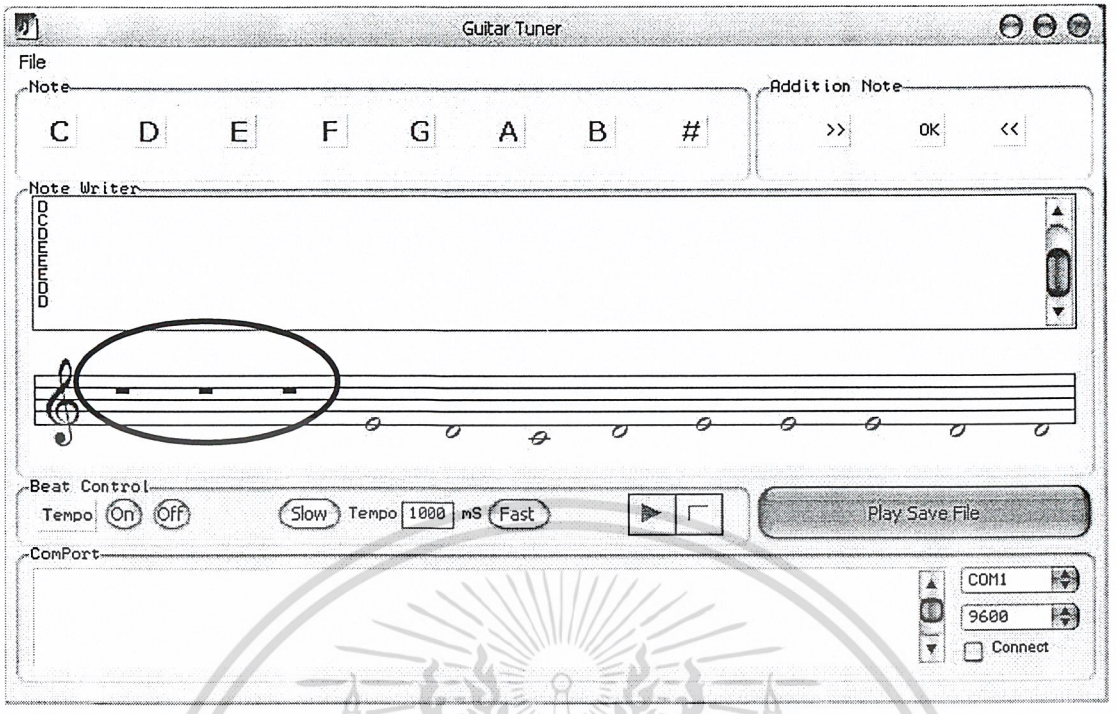
รูปที่ 6-9 แสดงปุ่มเปิดเริ่มการนับจังหวะ

เมื่อกำหนดจังหวะที่แน่นอนได้แล้ว ขั้นตอนต่อมาก็ลองเล่นดนตรีเข้าไปโดยลองไล่เสียงตัวโน้ตจากโน้ตตัวโด(C) ไปจนถึงโน้ตตัวที(B) ดังรูปที่ 6-10



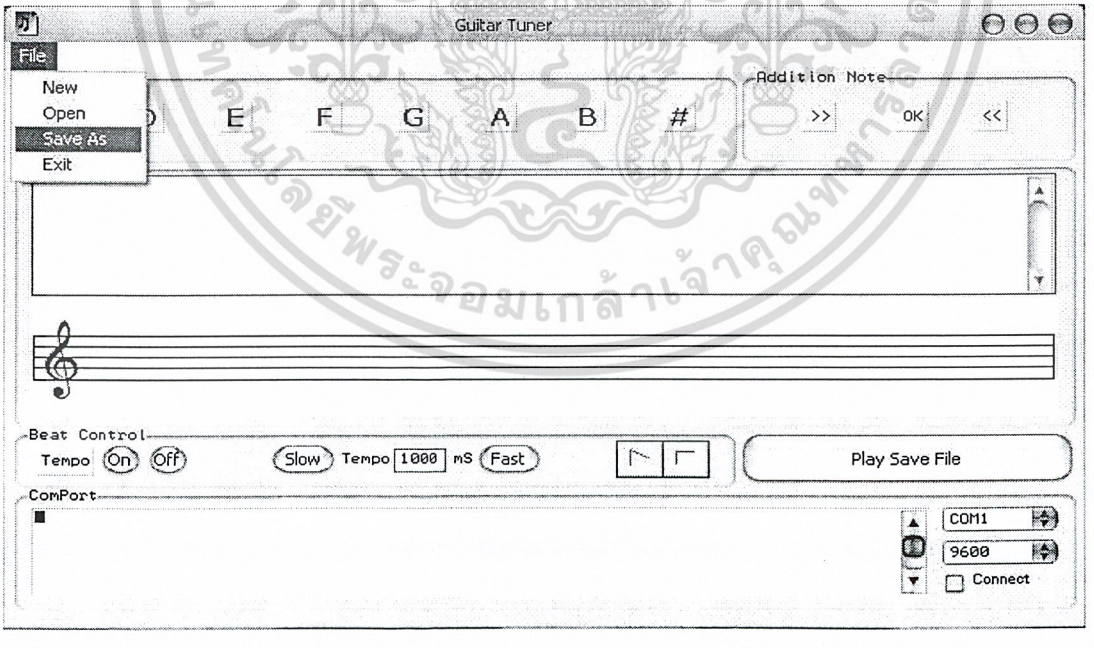
รูปที่ 6-10 แสดงการไล่เสียงตัวโน้ตจากโด(C)ถึงโน้ตตัวที(B)

ต่อมาเป็นการทดลองหยุดเล่นก่อนสามจังหวะแล้วจึงเล่นดนตรีเข้าไป ดังรูปที่ 6-11 จะเห็นได้ว่าโปรแกรมจะแสดงตัวหยุดขึ้นมาก่อนในขณะที่ผู้ใช้โปรแกรมไม่ได้เล่นอะไรเลย จากนั้นจึงเล่นดนตรีเข้าไปโปรแกรมก็จะทำการเขียนโน้ตที่ได้เล่นนั้น แสดงที่บรรทัดห้าเส้นได้ทันที



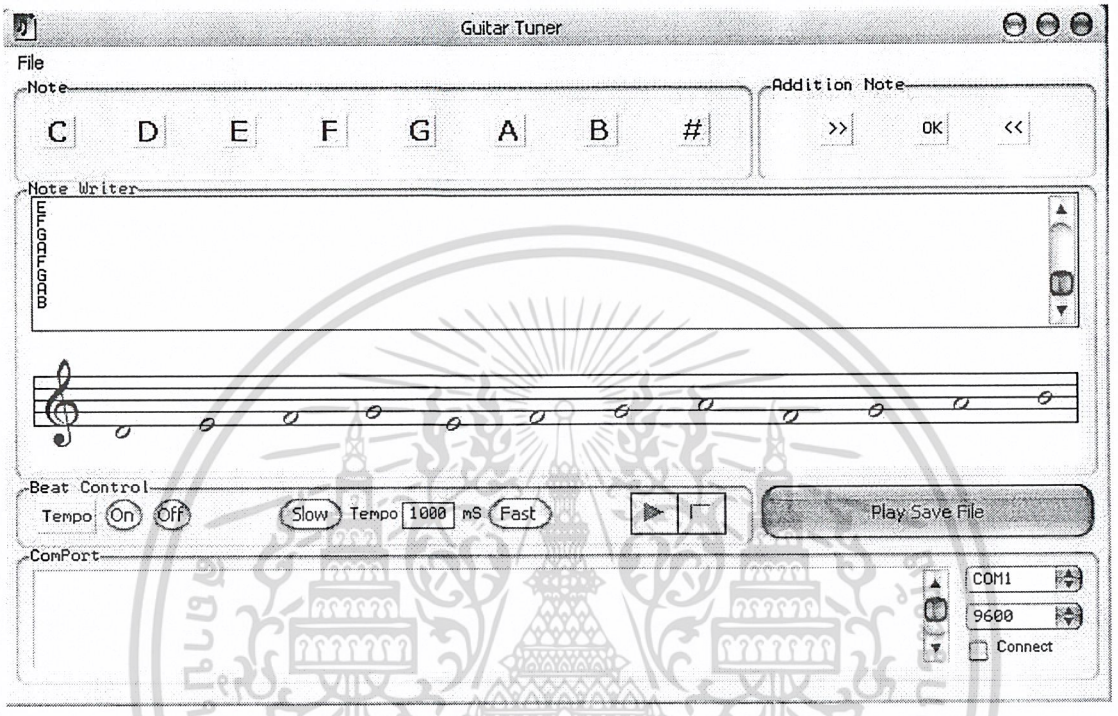
รูปที่ 6-11 แสดงตัวหยุดเมื่อผู้ใช้โปรแกรมไม่ได้เล่น

ต่อมาเป็นการบันทึกข้อมูลที่เล่นไปเก็บไว้เพื่อเรียกฟังข้อมูลที่ได้เล่นไปนั้นได้อีก ดังรูปที่ 6-12



รูปที่ 6-12 แสดงการบันทึกข้อมูลที่ได้เล่นไว้

File ที่บันทึกเข้าไปนั้นจะเป็นไฟล์ .txt ซึ่งภายในโปรแกรมนี้นั้นจะมีไฟล์เสียงที่เป็น Midi ของเสียงตัวโน้ตต่างๆไว้แล้ว สามารถเล่นและฟังข้อมูลที่ได้บันทึกไว้ได้โดย เข้าไปที่ File แล้วกด Open จากนั้นทำการเลือกข้อมูลที่ต้องการจะดูและฟังนั้นขึ้นมา แล้วกดปุ่ม Play Save File โปรแกรมก็จะทำการแสดงตัวโน้ตดนตรีขึ้นมาพร้อมกับเสียงเพลงให้ผู้ใช้ได้ฟัง ดังรูปที่ 6-13



รูปที่ 6-13 แสดงการนำข้อมูลที่ได้บันทึกไว้มาแสดง

บทที่ 7
บทสรุปและแนวทางในการพัฒนา

7.1 บทสรุป

โปรแกรมและฮาร์ดแวร์ที่ได้หลังจากทำการจัดทำขึ้นมานั้นการทำงานโดยรวมแล้ว ถือว่ามีประสิทธิภาพในการทำงานที่จะวิเคราะห์เสียงของตัวโน้ตได้ดี ซึ่งสามารถสรุปออกมาได้ดังนี้

- ในส่วนของฮาร์ดแวร์นั้นสามารถที่จะวิเคราะห์เสียงของตัวโน้ตต่างๆได้ดี และรวดเร็ว และส่งข้อมูลให้กับเครื่องคอมพิวเตอร์ได้ถูกต้อง
- โปรแกรมสามารถทำงานได้ตามที่ผู้จัดทำได้คาดหมายไว้ เกือบครบทุกรูปแบบ
- โปรแกรมมีรูปร่างหน้าตาที่ใช้งานได้ง่าย

แม้จะมีประสิทธิภาพในการทำงานที่ดี แต่ทั้งนี้บางส่วนของฮาร์ดแวร์และโปรแกรมก็ยังนั้นคงมีข้อจำกัด ในการทำงานหลงเหลืออยู่ ซึ่งสามารถจำแนกออกมาได้ดังนี้

1. ในส่วนของฮาร์ดแวร์นั้นสามารถที่จะวิเคราะห์เสียงของตัวโน้ตต่างๆได้ดี และรวดเร็ว แต่ตัวฮาร์ดแวร์นั้นยังไม่สามารถที่จะแบ่ง Octave ของโน้ตดนตรีได้
2. โน้ตที่เป็นตัวเขบ็ด 1 ชั้น หรือเขบ็ด 2ชั้น ซึ่งเป็นโน้ตที่มีความเร็วค่อนข้างสูง ในตัวโปรแกรมนั้นบางครั้งหรือในบางจังหวะที่เร็วๆก็ไม่สามารถที่จะวิเคราะห์ได้ทัน
3. ความผิดพลาดของเครื่องดนตรี คือ ในกีต้าร์ไฟฟ้านั้นจะมีวงจรไฟฟ้าที่รับเสียงจากสายกีต้าร์ ซึ่งเรียกว่า Pickups ซึ่งมีอยู่ 2 ชนิดคือ Hambucking และ Single Coil ซึ่งในการทดลองนั้นแบบ Single Coil จะมีความถี่เข้ามากวนมาก ซึ่งความถี่ที่เข้ามารบกวนนั้นบางครั้งก็ตรงกับความถี่ของโน้ตบางตัวทำให้ในบางครั้งวงจรและโปรแกรมจะเขียนโน้ตอยู่ตลอดเวลา

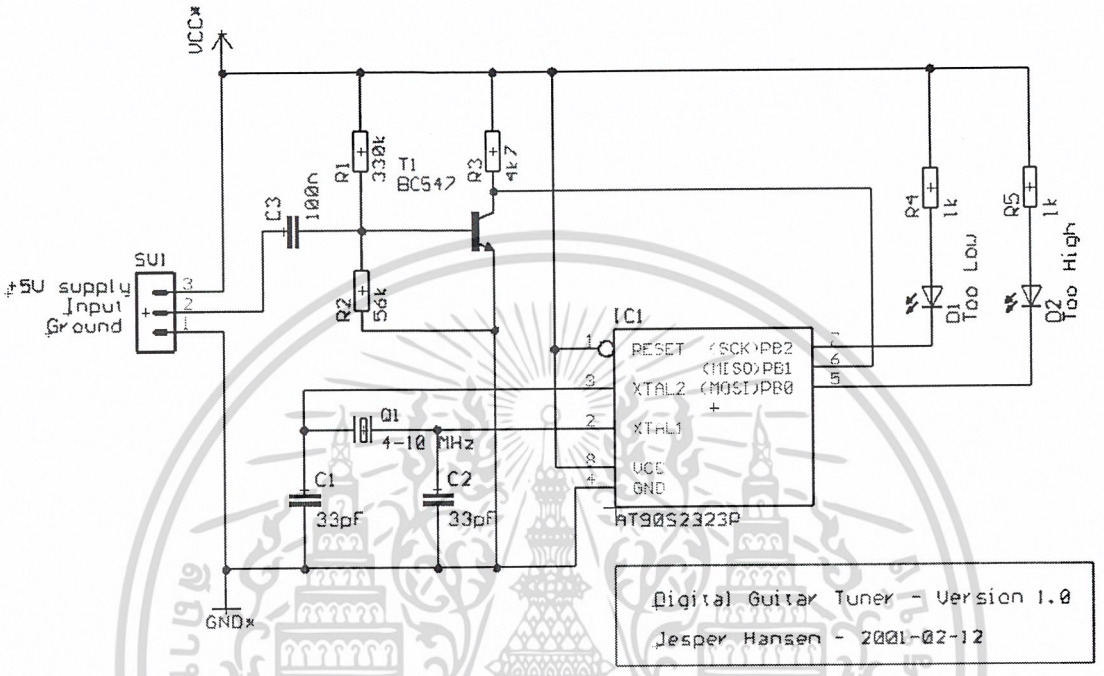
7.2 แนวทางในการพัฒนาต่อ

ภายหลังจากที่ได้ทำการศึกษาเกี่ยวกับปัญหาต่างๆที่ยังคงหลงเหลืออยู่ในฮาร์ดแวร์และโปรแกรมแล้วนั้น ก็สามารถสรุปแนวทางที่สามารถนำพัฒนาต่อเพื่อเพิ่มศักยะภาพของฮาร์ดแวร์และโปรแกรมให้ดียิ่งขึ้นดังนี้

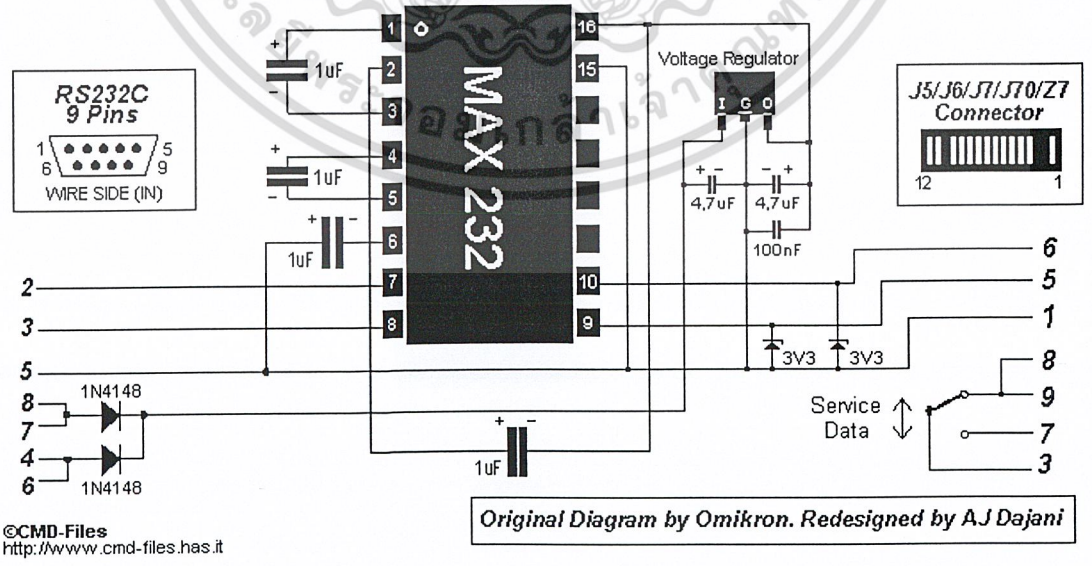
1. ในส่วนของฮาร์ดแวร์นั้น ควรจะหาวิธีการหรือจัดทำวงจรที่สามารถแบ่งย่านความถี่หรือที่เรียกว่า Octave ของโน้ตดนตรีขึ้นมา เพื่อให้กระบวนการวิเคราะห์ความถี่เสียงของตัวโน้ตนั้นถูกต้องและสมบูรณ์แบบมากขึ้นนั่นเอง

2. ควรศึกษาเรื่องทฤษฎีดนตรีเพิ่มขึ้นเป็นอย่างมาก เพราะในโครงงานนี้ต้องทำให้ถูกต้องตามทฤษฎีดนตรี ซึ่งในโครงานนี้ในบางส่วนยังไม่ถูกต้องตามทฤษฎีเท่าที่ควร
3. พัฒนาโปรแกรมให้ใช้งานได้ง่ายขึ้น หรือพัฒนารูปแบบของไฟล์เอาต์พุตให้หลากหลายขึ้น เช่น แปลงออกมาให้อยู่ในรูปของ Wave, MP3

ภาคผนวก ก.
รูปวงจรทั้งหมดในโครงงาน



รูปที่ ก-1 แสดงภาพวงจรที่ใช้แยกความถี่เสียง



รูปที่ ก-2 แสดงภาพวงจรMax 232

ภาคผนวก ข.

รหัสค่าเสียงของตัวโน้ต

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

ตารางที่ ข-1 รหัสค่าเสียงตัวโน้ตเมื่อเทียบเป็นเลขฐานสิบที่ Octave ต่างๆ

มาตรฐานของ MIDI จะกำหนดให้ค่าเสียงตัวโน้ต 60 เป็น C มาตรฐาน (Middle C) ซึ่งรหัสค่าเสียงตัวโน้ตอื่นๆ จะได้มาจากการเทียบกับค่า C มาตรฐานนั่นเอง

Code โปรแกรมของโครงงาน

```
unit Guitar_Tuner_Code;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, Menus, StdCtrls, ExtCtrls, CPortCtl, CPort, XPMan, MPlayer,
  ExtDlgs;

type
  TForm1 = class(TForm)
    MainMenu1: TMainMenu;
    Filse1: TMenuItem;
    New1: TMenuItem;
    Open1: TMenuItem;
    SaveAs1: TMenuItem;
    Exit1: TMenuItem;
    TunerGroupBox: TGroupBox;
    C_Panel: TPanel;
    D_Panel: TPanel;
    E_Panel: TPanel;
    F_Panel: TPanel;
    G_Panel: TPanel;
    A_Panel: TPanel;
    B_Panel: TPanel;
    GroupBox1: TGroupBox;
    Lok_Panel: TPanel;
    Cok_Panel: TPanel;
    Rok_Panel: TPanel;
    GroupBox2: TGroupBox;
    ComPort: TComPort;
    ComTerminal: TComTerminal;
    ComComboBox1: TComComboBox;
    ComComboBox2: TComComboBox;
```

```
ConnectCheckBox: TCheckBox;
GroupBox3: TGroupBox;
LEDTimer: TTimer;
TempoTimer: TTimer;
Memo1: TMemo;
GroupBox4: TGroupBox;
Button1: TButton;
Button2: TButton;
Label1: TLabel;
Button3: TButton;
Tempo_Status: TPanel;
MediaPlayer1: TMediaPlayer;
OpenDialog1: TOpenDialog;
S_Panel: TPanel;
Button5: TButton;
Image1: TImage;
OpenPictureDialog1: TOpenPictureDialog;
SaveDialog1: TSaveDialog;
Button4: TButton;
PlayTimer: TTimer;
TempoEdit: TEdit;
Label2: TLabel;
procedure Exit1Click(Sender: TObject);
procedure ConnectCheckBoxClick(Sender: TObject);
procedure ClearNote;
procedure ComTerminalChar(Sender: TObject; Ch: Char);
procedure NoteProgress(Rx: String);
procedure LEDTimerTimer(Sender: TObject);
procedure Button2Click(Sender: TObject);
procedure Button3Click(Sender: TObject);
procedure TempoTimerTimer(Sender: TObject);
procedure Button1Click(Sender: TObject);
procedure Button4Click(Sender: TObject);
procedure Button5Click(Sender: TObject);
```

```
    procedure Button6Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure Plot_Staff(cmp:String);
    procedure New1Click(Sender: TObject);
    procedure SaveAs1Click(Sender: TObject);
    procedure Open1Click(Sender: TObject);
    procedure PlayTimerTimer(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
  Rx : String;
  CMD  : Boolean;
  Room : Integer;
  Play : Boolean;
  F1    : TextFile;
implementation

{$R *.dfm}
//---------------------------------------------------------------------------
procedure TForm1.Exit1Click(Sender: TObject);
begin
  Close;
end;
//---------------------------------------------------------------------------
procedure TForm1.ConnectCheckBoxClick(Sender: TObject);
begin
  if ConnectCheckBox.Checked = True then
    ComPort.Connected := True
  else
```

```
    ComPort.Connected := False;
end;
//---------------------------------------------------------------------------
procedure TForm1.ClearNote;
begin
  C_Panel.Font.Color   := clBtnFace;
  D_Panel.Font.Color   := clBtnFace;
  E_Panel.Font.Color   := clBtnFace;
  F_Panel.Font.Color   := clBtnFace;
  G_Panel.Font.Color   := clBtnFace;
  A_Panel.Font.Color   := clBtnFace;
  B_Panel.Font.Color   := clBtnFace;
  Lok_Panel.Font.Color := clBtnFace;
  Cok_Panel.Font.Color := clBtnFace;
  Rok_Panel.Font.Color := clBtnFace;
  S_Panel.Font.Color := clBtnFace;
end;
//---------------------------------------------------------------------------
procedure TForm1.NoteProgress(Rx: String);
begin
  if        Pos('C', Rx) > 0 then begin
    ClearNote;
    C_Panel.Font.Color := clRed;
  end else if Pos('D', Rx) > 0 then begin
    ClearNote;
    D_Panel.Font.Color := clRed;
  end else if Pos('E', Rx) > 0 then begin
    ClearNote;
    E_Panel.Font.Color := clRed;
  end else if Pos('F', Rx) > 0 then begin
    ClearNote;
    F_Panel.Font.Color := clRed;
  end else if Pos('G', Rx) > 0 then begin
    ClearNote;
```

```
  G_Panel.Font.Color := clRed;
 end else if Pos('A', Rx) > 0 then begin
  ClearNote;
  A_Panel.Font.Color := clRed;
 end else if Pos('B', Rx) > 0 then begin
  ClearNote;
  B_Panel.Font.Color := clRed;
 end;

 if Pos('#', Rx) > 0 then
 begin
  S_Panel.Font.Color := clGreen;
 end;

 if      Pos('>', Rx) > 0 then begin
  Lok_Panel.Font.Color := clRed;
 end else if Pos('<', Rx) > 0 then begin
  Rok_Panel.Font.Color := clRed;
 end else begin
  Cok_Panel.Font.Color := clRed;
 end;
end;
//---------------------------------------------------------------------------
procedure TForm1.ComTerminalChar(Sender: TObject; Ch: Char);
begin
 if (Ch = #13) and (Rx <> '') then
 begin
  Rx  := Trim(Rx);
  if CMD = True then
  begin
   CMD := False;
   NoteProgress(Rx);
   Plot_Staff(Rx);
   Memo1.Lines.Append(Rx);
```

```
    end
    else
      NoteProgress(Rx);
    Rx := '';
  end else Rx := Rx + Ch;
end;
//---------------------------------------------------------------------------
procedure TForm1.LEDTimerTimer(Sender: TObject);
begin
  ClearNote;
  LEDTimer.Enabled := False;
end;
//---------------------------------------------------------------------------
procedure TForm1.Button2Click(Sender: TObject);
begin
  TempoTimer.Interval := TempoTimer.Interval - 100;
  TempoEdit.Text := IntToStr(TempoTimer.Interval);
  PlayTimer.Interval := TempoTimer.Interval;
end;
//---------------------------------------------------------------------------
procedure TForm1.Button3Click(Sender: TObject);
begin
  if MediaPlayer1.FileName <> '' then TempoTimer.Enabled := True;
end;
//---------------------------------------------------------------------------
procedure TForm1.TempoTimerTimer(Sender: TObject);
begin
  if Tempo_Status.Font.Color <> clGray then
    Tempo_Status.Font.Color := clGray
  else
    Tempo_Status.Font.Color := clRed;
  if CMD = True then
  begin
    Plot_Staff('S');
```

```
    Memo1.Lines.Append('S');
  end;
  CMD := True;
  MediaPlayer1.Play;
end;
//---------------------------------------------------------------------------
procedure TForm1.Button1Click(Sender: TObject);
begin
  TempoTimer.Interval := TempoTimer.Interval + 100;
  TempoEdit.Text := IntToStr(TempoTimer.Interval);
  PlayTimer.Interval := TempoTimer.Interval;
end;
//---------------------------------------------------------------------------
procedure TForm1.Button4Click(Sender: TObject);
begin
  PlayTimer.Interval := TempoTimer.Interval;
  AssignFile(F1, OpenDialog1.Filename);
  Reset(F1);
  PlayTimer.Enabled := True;
end;
//---------------------------------------------------------------------------
procedure TForm1.Button5Click(Sender: TObject);
begin
  TempoTimer.Enabled := False;
end;
//---------------------------------------------------------------------------
procedure TForm1.Button6Click(Sender: TObject);
var
  x, y : integer;
begin
 x := 2;
  for y := 0 to Image1.Height-1 do
    Memo1.Lines.Append(IntToStr(y)+'='+IntToStr(Image1.Canvas.Pixels[x,y] and $FF));
end;
```

```
//---------------------------------------------------------------------------
procedure TForm1.Plot_Staff(cmp:String);
var
  BitMap : TBitMap;
  x,y,shx,shy : Integer;
  Z : Byte;
  sharp : Boolean;
begin
 cmp := StringReplace(cmp, '<', '', [rfReplaceAll]);
 cmp := StringReplace(cmp, '>', '', [rfReplaceAll]);
 if Room >= 12 then
  begin
    Room := 0;
    OpenPictureDialog1.FileName := 'staff.bmp';
    Image1.Picture.LoadFromFile(OpenPictureDialog1.FileName);
  end;
  sharp := False;
  OpenPictureDialog1.FileName := 'note.bmp';
  shx := (50*Room)+50;
  if        cmp = 'C' then begin
    shy := 53+4;
    OpenPictureDialog1.FileName := 'note_.bmp';
  end else if cmp = 'D' then begin
    shy := 53;
  end else if cmp = 'E' then begin
    shy := 45+4;
  end else if cmp = 'F' then begin
    shy := 45;
  end else if cmp = 'G' then begin
    shy := 38+4;
  end else if cmp = 'A' then begin
    shy := 38;
  end else if cmp = 'B' then begin
    shy := 31+4;
```

```
end else if cmp = 'S' then begin
  shy := 31;
  OpenPictureDialog1.FileName := 'stop.bmp';
end else if cmp = 'C#' then begin
  shy := 53+4;
  OpenPictureDialog1.FileName := 'note_.bmp';
  sharp := True;
end else if cmp = 'D#' then begin
  shy := 53;
  sharp := True;
end else if cmp = 'E#' then begin
  shy := 45+4;
  sharp := True;
end else if cmp = 'F#' then begin
  shy := 45;
  sharp := True;
end else if cmp = 'G#' then begin
  shy := 38+4;
  sharp := True;
end else if cmp = 'A#' then begin
  shy := 38;
  sharp := True;
end else if cmp = 'B#' then begin
  shy := 31+4;
  sharp := True;
end else
  shy := 0;
BitMap := TBitMap.create;
BitMap.LoadFromFile(OpenPictureDialog1.FileName);
for x:=0 to BitMap.Width-1 do
  for y:=0 to BitMap.Height-1 do
  begin
    Z := (Image1.Canvas.Pixels[x+shx,y+shy] and  BitMap.Canvas.Pixels[x,y]) and $FF;
    Image1.Canvas.Pixels[x+shx,y+shy] := RGB(Z,Z,Z);
```

```
  end;
 BitMap.Free;
 if sharp = True then
 begin
  shx := shx + 10;
  OpenPictureDialog1.FileName := 'sharp.bmp';
  BitMap := TBitMap.create;
  BitMap.LoadFromFile(OpenPictureDialog1.FileName);
  for x:=0 to BitMap.Width-1 do
   for y:=0 to BitMap.Height-1 do
   begin
    Z := (Image1.Canvas.Pixels[x+shx,y+shy] and  BitMap.Canvas.Pixels[x,y]) and $FF;
    Image1.Canvas.Pixels[x+shx,y+shy] := RGB(Z,Z,Z);
   end;
  BitMap.Free;
 end;
 Room := Room + 1;
end;
//---------------------------------------------------------------------------
procedure TForm1.FormCreate(Sender: TObject);
begin
 TempoEdit.Text := IntToStr(TempoTimer.Interval);
 Room := 0;
 MediaPlayer1.FileName := 'tempo.wav';
 MediaPlayer1.Open;
end;
//---------------------------------------------------------------------------
procedure TForm1.New1Click(Sender: TObject);
begin
 Room := 0;
 Play := False;
 OpenPictureDialog1.FileName := 'staff.bmp';
 Image1.Picture.LoadFromFile(OpenPictureDialog1.FileName);
 Memo1.Clear;
```

```
end;
//---------------------------------------------------------------------------
procedure TForm1.SaveAs1Click(Sender: TObject);
begin
 TempoTimer.Enabled := False;
 if SaveDialog1.Execute then Memo1.Lines.SaveToFile(SaveDialog1.FileName);
end;
//---------------------------------------------------------------------------
procedure TForm1.Open1Click(Sender: TObject);
begin
 TempoTimer.Enabled := False;
 if OpenDialog1.Execute then Play := True;
end;
//---------------------------------------------------------------------------
procedure TForm1.PlayTimerTimer(Sender: TObject);
var
 Str : String;
begin
 if Play = True then
 begin
//   MediaPlayer1.Play;
   if not Eof(F1) then
   begin
     Readln(F1, Str);
     Plot_Staff(Str);
     Memo1.Lines.Append(Str);
//     MediaPlayer1.Close;
     if Pos('#', Str) > 0 then
     begin
       if Pos('A', Str) > 0 then begin
         MediaPlayer1.FileName := 'A#.wav';
       end else if Pos('C', Str) > 0 then begin
         MediaPlayer1.FileName := 'C#.wav';
       end else if Pos('D', Str) > 0 then begin
```

```
    MediaPlayer1.FileName := 'D#.wav';
   end else if Pos('F', Str) > 0 then begin
    MediaPlayer1.FileName := 'F#.wav';
   end else if Pos('G', Str) > 0 then begin
    MediaPlayer1.FileName := 'G#.wav';
   end else begin
    MediaPlayer1.FileName := 'Tempo.wav';
   end;
  end
  else
  begin
   if Pos('A', Str) > 0 then begin
    MediaPlayer1.FileName := 'A.wav';
   end else if Pos('B', Str) > 0 then begin
    MediaPlayer1.FileName := 'B.wav';
   end else if Pos('C', Str) > 0 then begin
    MediaPlayer1.FileName := 'C.wav';
   end else if Pos('D', Str) > 0 then begin
    MediaPlayer1.FileName := 'D.wav';
   end else if Pos('E', Str) > 0 then begin
    MediaPlayer1.FileName := 'E.wav';
   end else if Pos('F', Str) > 0 then begin
    MediaPlayer1.FileName := 'F.wav';
   end else if Pos('G', Str) > 0 then begin
    MediaPlayer1.FileName := 'G.wav';
   end else begin
    MediaPlayer1.FileName := 'Tempo.wav';
   end;
  end;
  MediaPlayer1.Open;
  MediaPlayer1.Play;
 end
 else
 begin
```

```
    CloseFile(F1);
    PlayTimer.Enabled := False;
   end;
  end;
end;
//---------------------------------------------------------------------------
end.
```

บรรณานุกรม

[1] Mark Clay. "MIDI File Format." [Online].
Available : http://ourworld.compuserve.com/homepages/mark_clay/midi.htm. 1996.

[2] สมนึก อุ่นแก้ว: "*ทฤษฎีดนตรี แนวปฏิบัติ*", พิมพ์ครั้งที่ 2, Musico 2537.

[3] เล็ก วงศ์สว่าง: "*Picking*", พิมพ์ครั้งที่ 2, วงศ์สว่างการพิมพ์ 2523.

[4] George Hansper. "An Introduction to MIDI."[Online].
Available : http://www.crystal.apana.org.au/ghansper/midi_introduction/contents.html
1998.

[5] Harmony Central ,Inc. All rights reserved. "MIDI Documentation." [Online].
Available : http://www.hamony-central.com/MIDI/Doc/doc.html. 1995-97

[6] Steve Winder. *Analog and Digital Filter Design 2nd*. USA. Newnes an imprint of Elsevier Science.

[7] Kendall L. Su. *Analog Filters.* Atlanta. Chapman & Hall.

[8] http://www.kpsec.freeuk.com

[9] http://www.anotherurl.com/library/rs232.htm

[10] http://www.seattlerobotics.org/encoder/aug97/cable.html