ระบบไฟสถานบันเทิงแบบอัตโนมัติตามจังหวะเสียงดนตรี

AUTOMATIC TEMPO-BASED ON-STAGE FLASHING LIGHT SYSTEM

โดย

นายกฤษดา บาตะศรี

นายกสานติ์ ดวงประทีป

นายณัฐธีร์ อรุณวิจิตรสกุล

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2553

ระบบไฟสถานบันเทิงแบบอัตโนมัติตามจังหวะเสียงดนตรี

AUTOMATIC TEMPO-BASED ON-STAGE FLASHING LIGHT SYSTEM

โดย

นายกฤษดา บาตะศรี

นายกสานติ์ ดวงประทีป

นายณัฐธีร์ อรุณวิจิตรสกุล

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2553

ผ่านการตรวจรูปเล่มแล้ว

(ลงชื่อ).....ผู้ตรวจ

ผ่านการตรวจรูปเล่มแล้ว

(ลงชื่อ).....ผู้ตรวจ

ปริญญานิพนธ์ปีการศึกษา 2553

สาขาวิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบไฟสถานบันเทิงแบบอัตโนมัติตามจังหวะเสียงดนตรี

AUTOMATIC TEMPO-BASED ON-STAGE FLASHING LIGHT SYSTEM

ผู้จัดทำ

1. นายกฤษดา มาตะศรี 50010051
2. นายกสานติ์ ดวงประทีป 50010060
3. นายณัฐธีร์ อรุณวิจิตรสกุล 50010458

..... อาจารย์ที่ปรึกษา

(ผศ. ดร. พิพัฒน์ พรหมมี)

..... อาจารย์ที่ปรึกษาร่วม

(ดร.มนตรี คำเงิน)

กิตติกรรมประกาศ

ขอขอบพระคุณ ผศ.ดร.พิพัฒน์ พรหมมี เป็นอย่างสูงที่ได้ช่วยแนะนำให้คำปรึกษาในการทำปริญญานิพนธ์นี้ ตลอดจนคณาจารย์ทุกท่านที่ได้ให้การอบรมสั่งสอนแก่ผู้จัดทำที่มีส่วนช่วยเหลือในการจัดทำปริญญานิพนธ์นี้ให้สำเร็จลุล่วงไปด้วยดี รวมถึงห้องสมุดคณะวิศวกรรมศาสตร์ที่ทำให้ได้มีแนวคิดในการทำโครงงานนี้

ขอขอบคุณเพื่อน และ รุ่นพี่ที่คอยเป็นกำลังใจและให้ความช่วยเหลืออยู่เสมอในการจัดทำ ตลอดจนเพื่อนร่วมปริญญานิพนธ์ที่ให้ความเป็นเพื่อนและกำลังใจในการฝ่าฟันอุปสรรค ร่วมคิดและแก้ปัญหา นับเป็นสิ่งล้ำค่าที่สุดที่หาไม่ได้ในตำราเรียน

สิ่งหนึ่งที่ขาดเสียไม่ได้ ขอขอบพระคุณบิดา มารดา และครอบครัวที่คอยช่วยเหลือเกื้อกูล ส่งเสริมให้ข้าพเจ้าได้ร่ำเรียนและเป็นกำลังใจในชีวิตของข้าพเจ้า

คณะผู้จัดทำ

ระบบไฟสถานบันเทิงแบบอัตโนมัติตามจังหวะเสียงดนตรี
AUTOMATIC TEMPO-BASED ON-STAGE FLASHING LIGTH SYSTEM

โดย นายกฤษดา บาตะศรี 50010051
นายกสานติ์ ดวงประทีป 50010060
นายณัฐธีร์ อรุณวิจิตรสกุล 50010458

อาจารย์ที่ปรึกษา ผศ.ดร. พิพัฒน์ พรหมมี
อาจารย์ที่ปรึกษาร่วม ดร. มนตรี คำเงิน

บทคัดย่อ

โครงงานนี้นำเสนอระบบไฟสถานบันเทิงแบบอัตโนมัติตามจังหวะเสียงดนตรี โดยระบบจะแยกแยะจังหวะดนตรี ช้า ปานกลาง และ เร็ว ด้วยโปรแกรมบนคอมพิวเตอร์ และ เชื่อมต่อกับไมโครคอนโทรลเลอร์ โดยจะทำงานร่วมกันแบบอัตโนมัติ และ ตามที่กำหนด ทั้งช่วงเวลา และ ตำแหน่งของหลอดไฟ โดยไมโครคอนโทรเลอร์จะเก็บรูปแบบการกระพริบของไฟสีต่างๆ ตามจังหวะดนตรี โดยสัญญาณควบคุมจะออกจากไมโครคอนโทรลเลอร์ไปยังชุดขับไตรแอกเพื่อขับหลอดไฟให้กระพริบตามจังหวะเสียงดนตรี และ โปรแกรมที่กำหนด

ABSTRACT

This project presents Automatic tempo-based on-stage flashing light system. Tempo of music (slow, medium and fast) is detected by using the program on the computer. The computer and microcontroller are connected and functioned for automatic or manual controlled of timing and position of the bulbs. Microcontroller provides the number of patterns for display based on the tempo of music. The system provides outputs for the bulb driver circuit. The several bulbs can be flashed based on the music tempo or manual programmed.

สารบัญ

สารบัญ (ต่อ)

สารบัญรูป

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญตาราง

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ในปัจจุบันธุรกิจบันเทิงได้เติบโตขึ้นมาพร้อมกับเทคโนโลยีที่พัฒนาขึ้น โดยเฉพาะธุรกิจสถานบันเทิงที่ขยายตัวขึ้นอย่างรวดเร็วตอบสนองพฤติกรรมทางสังคมของมนุษย์ที่เปลี่ยนไป ในสถานบันเทิงแต่ละแห่ง จะมีการจัดตกแต่งร้านให้เหมาะสมตามความต้องการของกลุ่มลูกค้า การแสดงดนตรี หรือมีการเปิดเพลงเพื่อสร้างบรรยากาศ โดยส่วนประกอบที่สำคัญที่จะช่วยสร้างความสวยงามและดึงดูดลูกค้าได้มากขึ้นคือการจัดรูปแบบของแสงไฟ ซึ่งจะเกี่ยวข้องกับอุปกรณ์ทางอิเล็กทรอนิกส์ที่มีให้เลือกใช้ได้หลายรูปแบบแตกต่างกันไป ความสามารถของอุปกรณ์เหล่านั้นจะขึ้นอยู่กับรูปแบบของการกระพริบของหลอดไฟที่หลากหลาย ทั้งนี้อุปกรณ์ที่มีความสามารถสูงก็จะมีราคาแพงตามไปด้วยเช่นกัน การจะสร้างอุปกรณ์เหล่านี้นั้นจะต้องศึกษาเกี่ยวกับการควบคุมหลอดไฟด้วยวงจรต่างๆ รวมไปถึงจังหวะของดนตรีในแต่ละเพลงเพื่อให้การทำงานสอดคล้องกัน

"จังหวะ" ในหลักการของดนตรีเป็นเรื่องที่สามารถเข้าใจโดยสัญชาตญาณ แต่ค่อนข้างยากที่จะนิยามหรือเจาะจงความหมาย ทำให้เป็นปรากฏการณ์ที่ขึ้นอยู่กับมุมมองของแต่ละบุคคล ซึ่งไม่มีพื้นฐานในทางทฤษฎีใดๆ ที่สามารถอธิบายทฤษฎีของจังหวะเสียงดนตรีได้อย่างชัดเจน สิ่งเดียวที่ตั้งอยู่บนพื้นฐานความเป็นจริงคือ สิ่งที่มนุษย์รับฟังและเข้าใจตรงกันว่าสัญญาณเสียงนั้นๆ ประกอบด้วยจังหวะและเป็นเสียงเพลง เสียงดนตรี

ในการจำลองปรากฏการณ์ทางฟิสิกส์จำเป็นต้องอยู่ภายใต้หลักการหรือทฤษฎีทางคณิตศาสตร์ และมีผลลัพธ์ที่มีค่าเป็นตัวเลขที่สามารถเป็นไปได้เสมอ แต่หลักการนั้นไม่สามารถวัดค่าหรือตีกรอบให้กับสิ่งที่เป็นนามธรรมได้ อาทิเช่น อารมณ์ ความรู้สึกของมนุษย์ ซึ่งไม่อยู่ภายใต้กฎเกณฑ์ทางคณิตศาสตร์ สิ่งที่ง่ายที่สุดที่มนุษย์สัมผัสรับรู้กลับเป็นเรื่องยากที่จะประมวลผลด้วยโปรแกรมใดๆ Beat Detection หรือการตรวจจับบีตทำงานภายใต้ความซับซ้อนดังกล่าว คือ สัมผัสจังหวะของเสียงเพลงที่เป็นธรรมชาติมากที่สุดสำหรับมนุษย์ ความรู้สึกแรกที่มนุษย์จะเคาะเท้าหรือเต้นเมื่อได้ยินเสียงเพลงตามจังหวะของเพลง ประเด็นสำคัญคือการสั่งงานให้คอมพิวเตอร์ที่ทำได้

เพียงประมวลผลให้รู้จักกับคำว่า "จังหวะ" หรือ "บีต" ที่เป็นเรื่องธรรมชาติสำหรับมนุษย์ ซึ่งมีหลายขั้นตอนวิธีการทั้งแบบแม่นยำมากและน้อยแตกต่างกันไป

สำหรับปริญญานิพนธ์นี้จะเป็นการออกแบบและสร้างระบบไฟสถานบันเทิงแบบอัตโนมัติตามจังหวะเสียงดนตรี ข้อดีของระบบคือสามารถเลือกรูปแบบของการกระพริบของหลอดไฟได้ทั้งความเร็วและตำแหน่งของหลอดไฟแต่ละสี เพื่อให้การกระพริบของหลอดไฟสอดคล้องกับจังหวะของเพลง ทำให้การทำงานยืดหยุ่นมากขึ้นและมีราคาประหยัดเมื่อเทียบกับอุปกรณ์ที่มีความสามารถในระดับเดียวกัน โดยระบบจะประกอบด้วยการแอพพลิเคชั่นบนคอมพิวเตอร์และโปรแกรมตรวจจับบีตเพื่อสั่งการให้หลอดไฟกระพริบตามเสียงดนตรี อีกทั้งยังมีรูปแบบการกระพริบอีกมากมายทั้งการสุ่มเลือกหรือกดคีย์แพด จากนั้นจะส่งข้อมูลไปยังไมโครคอนโทรลเลอร์ (เบอร์ AT89C52) ซึ่งเป็นส่วนที่เก็บรูปแบบการกระพริบของหลอดไฟโดยจะทำการสุ่มเลือกรูปแบบการกระพริบในหมวดต่างๆตามค่าอินพุตที่ได้รับจากคอมพิวเตอร์ หรือสามารถเลือกรูปแบบการกระพริบของหลอดแบบกำหนดเอง หรือเลือกให้หลอดไฟกระพริบตามจังหวะเสียงดนตรีโดยอัตโนมัติด้วยคีย์แพดขนาด 3x4 ซึ่งควบคุมโดยไมโครคอนโทรลเลอร์ตัวที่สอง (AT89C4051) ซึ่งการเลือกใช้งานจะสลับการทำงานด้วยสวิตซ์เพื่อเลือกว่าจะรับสัญญาณอินพุตจากคอมพิวเตอร์หรือจากคีย์แพดซึ่งต่อกับไมโครคอนโทรลเลอร์ตัวที่สอง จากนั้นไมโครคอนโทรลเลอร์ตัวแรกก็จะส่งสัญญาณควบคุมไปยังชุดขับไตรแอกเพื่อขับหลอดไฟให้กระพริบตามโปรแกรมต่อไป

1.2 วัตถุประสงค์

1) ออกแบบและสร้างระบบไฟสถานบันเทิงแบบอัตโนมัติตามจังหวะเสียงดนตรี
2) ศึกษาและทดลองเขียนโปรแกรมด้วยภาษา C# และ Java เพื่อออกแบบโปรแกรมบนคอมพิวเตอร์ และภาษา C ในการควบคุมไมโครคอนโทรลเลอร์ MCS-51
3) ศึกษาและทดลองเขียนแอพพลิเคชั่นด้วยโปรแกรม Visual Studio 2008 และโปรแกรม Processing
4) ศึกษาทฤษฎีของการวิเคราะห์สัญญาณเสียงและประยุกต์ใช้งาน
5) ศึกษาการทำงานและการสร้างวงจรไตรแอกเพื่อควบคุมหลอดไฟ
6) ศึกษาหลักการทำงานของวงจรโซลิดสเตทรีเลย์
7) ศึกษาการใช้งานไมโครคอนโทรลเลอร์ MCS-51

8) ประยุกต์หลักการต่างๆที่เกี่ยวข้องกับการตรวจจับบีตในเพลง เพื่อพัฒนาและนำไปออกแบบระบบไฟสถานบันเทิงแบบอัตโนมัติตามจังหวะเสียงดนตรีที่มีคุณภาพที่ดีในอนาคต

1.3 ขอบเขตของปริญญานิพนธ์

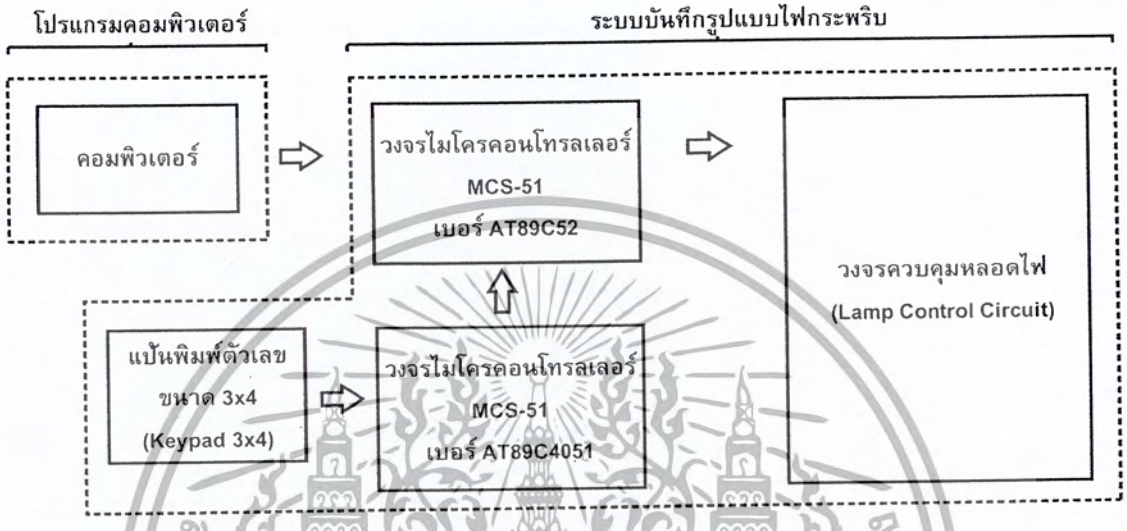
ขอบเขตการทำงานแบ่งเป็นสองส่วนหลักคือ ในส่วนของโปรแกรมบนคอมพิวเตอร์ และส่วนของระบบบันทึกรูปแบบการกระพริบของหลอดไฟ

ในส่วนของโปรแกรมบนคอมพิวเตอร์ จะเป็นการเขียนโปรแกรมภาษา C# ด้วยโปรแกรม Visual Studio 2008 เมื่อทำการเล่นเพลงจากในคอมพิวเตอร์ โปรแกรมนี้จะทำการเรียกใช้แอพพลิเคชั่นจาวาจากโปรแกรม Processing ซึ่งสามารถตรวจจับและทำการวิเคราะห์สัญญาณด้วยวิธีการต่างๆทำให้สามารถสั่งการให้ระบบบันทึกรูปแบบไฟกระพริบสามารถสั่งการหลอดไฟ ให้กระพริบตามจังหวะเสียงดนตรีได้ โดยโปรแกรมจะมีรูปแบบเป็นแอพพลิเคชั่นบนคอมพิวเตอร์สะดวกต่อการ เล่นเพลง, หยุดเพลง, เร่ง-ถอยหลัง และเรียงคิวเพลงได้ จากนั้นจะส่งสัญญาณไปเลือกรูปแบบการกระพริบของหลอดไฟบนไมโครคอนโทรเลอร์ ซึ่งเชื่อมต่อด้วยการส่งข้อมูลแบบอนุกรมตามมาตรฐาน RS-232

ส่วนที่สองทำการสร้างรูปแบบของไฟกระพริบและเก็บรูปแบบต่างๆไว้ในไมโครคอนโทรลเลอร์ (AT89C52) โดยเขียนโปรแกรมสั่งงานด้วยภาษาซี ระบบจะทำงานแบ่งเป็นสองลักษณะซึ่งสลับการทำงานด้วยสวิตซ์ คือ ในส่วนแรกจะรับค่าอินพุตที่ได้จากโปรแกรมบนคอมพิวเตอร์ ส่วนต่อมาจะเป็นการกระพริบของหลอดไฟในลักษณะแบบกำหนดเอง โดยการกดสวิตซ์ที่คีย์แพดขนาด 3x4 ควบคุมโดยไมโครคอนโทรลเลอร์ (AT89C4051) และจะเป็นตัวส่งสัญญาณควบคุมไปยังไมโครคอนโทรลเลอร์ (AT89C52) โดยสามารถใช้งานได้ทั้งหมด 12 ปุ่ม (12 รูปแบบ) แบ่งออกเป็น รูปแบบการกระพริบของเพลงช้า-กลาง-เร็วอย่างละ 3 ปุ่ม และแบบกำหนดได้ด้วยตนเอง (ควบคุมหลอดไฟด้วยตนเอง) 3 ปุ่ม รวมทั้งหมด 12 ปุ่ม

สัญญาณควบคุมการกระพริบของหลอดไฟจะส่งจากไมโครคอนโทรลเลอร์ (AT89C52) ไปยังวงจรควบคุมหลอดไฟ ซึ่งประกอบด้วย วงจรของโซลิดสเตทรีเลย์และวงจรไตรแอกแบบออปโตไอโซเลเตอร์ (TIC206C) ซึ่งทำงานที่โหลดตั้งแต่ 5 วัตต์ที่ 240 โวลต์ ไปจนถึง

โหลด 250 วัตต์ที่ 240 โวลต์ โดยมีแผ่นระบายความร้อนขนาดที่รองรับความร้อนได้ 17 องศาเซลเซียสต่อวัตต์ (17° c/w) บล็อกไดอะแกรมระบบไฟสถานบันเทิงแบบอัตโนมัติตามจังหวะเสียงดนตรีแสดงดังรูปที่ 1.1



รูปที่ 1.1 บล็อกไดอะแกรมระบบไฟสถานบันเทิงแบบอัตโนมัติตามจังหวะเสียงดนตรี

บทที่ 2

ทฤษฎีและหลักการที่เกี่ยวข้อง

2.1 ทฤษฎีการตรวจจับจังหวะหรือบีตของเฟดริก พาติน (Federic Patin)

ในช่วงหลายปีที่ผ่านมาการวิเคราะห์เพื่อแยกพัลส์ของจังหวะเสียงดนตรีโดยอัตโนมัติ เป็นหัวข้องานวิจัยที่น่าสนใจมาก เฟดริก พาติน (Federic Patin) เป็นผู้หนึ่งที่ได้ทดลองค้นคว้าการตรวจจับบีตหรือจับจังหวะของเพลงต่างๆ (Beat Detection) โดยมีความคิดริเริ่มมาจาก อีริค ดี เชียร์เรอร์ (Eric d. Sheirer) ผู้เขียนรายงานการวิจัย "การวิเคราะห์จังหวะและบีตของสัญญาณเสียงดนตรี (Tempo and beat analysis of acoustic musical signal)" ของมหาวิทยาลัยเคมบริดจ์ ประเทศอังกฤษ โดยการค้นคว้าของ เฟดริก พาติน ได้วางกรอบทฤษฎีและการคำนวณไว้น่าสนใจและเป็นประโยชน์ต่อการทำปริญญานิพนธ์นี้เป็นอย่างมาก

เฟดริกเลือกที่จะใช้หลักการของการวิเคราะห์สัญญาณ (Signal Processing) เพื่อการตรวจจับบีตที่มีประสิทธิภาพมากที่สุด อย่างไรก็ตามเขาเพียงแต่วางแนวทางและทฤษฎีไว้ แต่ยังไม่เคยทดลองใช้งานอย่างจริงจัง ทฤษฎีของเขากล่าวว่า ในการได้ยินของมนุษย์ การจะรับรู้จังหวะหรือบีตของเสียงเพลงนั้น เกิดขึ้นจากคลื่นเสียงจะถูกเปลี่ยนแปลงเป็นสัญญาณไฟฟ้า จากนั้นสมองจะตีความคลื่นสัญญาณไฟฟ้านั้นและจะรับรู้ว่าเป็นบีต สังเกตได้ว่ายิ่งคลื่นเสียงนั้นๆมีพลังงานเสียงมากเท่าใด สมองของมนุษย์จะรับรู้เสมือนเป็นลูกพลังงานเสียงขนาดใหญ่ซึ่งเข้าใจในทางธรรมชาติกันว่าเป็นบีต ซึ่งพลังงานเสียงที่มีขนาดใหญ่มากๆเกินกว่าพลังงานเสียงช่วงอื่นๆในเพลง พลังงานเสียงนั้นจะถูกจัดว่าเป็นบีตและถูกเรียกว่า "พลังงานเสียงสูงสุด (Sound energy peaks)"

จากทฤษฎีข้างต้น การตรวจจับพลังงานสูงสุดจะใช้เทคนิค การคำนวณค่าพลังงานเสียงเฉลี่ยเปรียบเทียบกับค่าพลังงานเสียงปัจจุบัน โดยจะทำการคำนวณทั้งเป็น 2 ช่องสัญญาณหรือโหมดสเตอริโอ กล่าวคือ จะทำการคำนวณทั้งฝั่งซ้าย $a[n]$ และฝั่งขวา $b[n]$ ของสัญญาณ $a[n]$ จะเป็นค่าแอมปลิจูดของสัญญาณรับค่าได้ในช่วง T_e วินาทีสำหรับสัญญาณฝั่งซ้าย $b[n]$ จะเป็นค่าแอมปลิจูดของสัญญาณรับค่าได้ในช่วง T_e วินาทีสำหรับสัญญาณฝั่งขวา

โดยจะทำการสร้างพัลส์จำนวน 1024 แซมเปิลเข้าไปในสัญญาณทั้ง $a[n]$ และ $b[n]$ เพื่อใช้ในการเก็บค่าพลังงานปัจจุบัน จะใช้บัฟเฟอร์ วิ่งไปตามสตรีมของสัญญาณไปเรื่อยๆ เพื่อเก็บค่าพลังงานปัจจุบัน จากนั้นแทนแซมเปิลจำนวน 44032 แซมเปิล ซึ่งประมาณค่าเท่ากับ 1 วินาทีของเพลงโดยทั่วไปเพื่อใช้เก็บค่าพลังงานเฉลี่ยขณะนั้นเนื่องจากการได้ยินของมนุษย์จะรับรู้บีตในขณะนั้นได้เพียง 1 วินาทีเท่านั้น จากนั้นบัฟเฟอร์ที่ใช้เก็บค่าพลังงานเฉลี่ยจะวิ่งตามสตรีมของสัญญาณทำให้มีความจำเพิ่มในการเก็บค่าพลังงานปัจจุบันใช้หลักการนี้ทำซ้ำไปเรื่อยๆจนสิ้นสุดสตรีมเพลง เมื่อทำการเก็บค่าพลังงานปัจจุบันและพลังงานเฉลี่ยของเพลงในขณะนั้นได้แล้วจะทำการเปรียบเทียบกัน หากค่าพลังงานปัจจุบันมีมากกว่าค่าพลังงานเฉลี่ยข้างเคียงแล้ว ค่าพลังงานปัจจุบันนั้นจะถูกจัดว่าเป็นพลังงานเสียงสูงสุดที่เกิดขึ้น ณ ขณะนั้นและถูกจัดว่าเป็นบีต

โดยที่ค่าพลังงานเฉลี่ยของเพลงจะไม่ทำการคำนวณทั้งเพลงเพราะว่าในเพลงๆหนึ่งจะมีทั้งส่วนที่มีดนตรีเข้มข้นและบางส่วนที่มีดนตรีค่อนข้างเบาบาง ดังนั้นค่าพลังงานเฉลี่ยจะเก็บค่าเป็นช่วงเวลาเท่านั้น และจะเปรียบเทียบกับค่าพลังงานปัจจุบันเพื่อให้เพลงที่มีดนตรีเบาบางทั้งเพลงแต่ช่วงท้ายเพลงมีดนตรีเข้มข้น ช่วงท้ายเพลงจะ ไม่ส่งผลกระทบต่อค่าพลังงานเฉลี่ยทั้งหมด โดยหลักการที่กล่าวมาข้างต้น เรียกว่า "การวิเคราะห์การตรวจจับบีตในเชิงสถิติ (Statistical streaming Beat detection) สามารถแสดงเป็นขั้นตอนได้ดังนี้

2.1.1 ขั้นตอนวิธีการตรวจจับบีตในเชิงสถิติ แบบที่ 1

2.1.1.1 แทนค่าอย่างละ 1024 แซมเปิลลงใน $a[n]$ และ $b[n]$ เพื่อคำนวณค่าพลังงานปัจจุบัน (Instant energy) ซึ่งในที่นี้จะแทนด้วยค่า e แทนด้วยสมการได้ดังสมการที่ 2.1

$$e = e_{stereo} = e_{right} + e_{left} = \sum_{k=i_0}^{i_0+1024} a[k]^2 + b[k]^2 \qquad (2.1)$$

2.1.1.2 คำนวณค่าพลังงานปัจจุบัน ณ ช่วงเวลา (Local average energy) หรือแทนด้วยค่า $< E >$ ด้วยค่าแซมเปิล 44032 แซมเปิล แสดงดังสมการที่ 2.2 (44100 คือค่า Sample rate ของไฟล์เพลง)

$$< E >= \frac{1024}{44100} \times \sum_{i=0}^{44032} (B[0][i])^2 + (B[1][i])^2 \qquad (2.2)$$

2.1.1.3 เลื่อนบัฟเฟอร์เก็บค่าไปทางขวาอีก 1024 แซมเปิล จะทำให้สามารถเก็บค่าแซมเปิลใหม่ได้ 1024 จากนั้นแซมเปิล ลบ 1024 แซมเปิลเก่าทิ้ง

2.1.1.4 เปรียบเทียบค่า e กับ $C \times < E >$ โดยที่ค่า C คือค่าคงที่ความไวต่อสิ่งกระตุ้น (ในที่นี้จะใช้ค่า 1.3) ซึ่งจะขออธิบายในหัวข้อที่ 2.1.3

2.1.1.5 หากค่า e มีขนาดพลังงานใกล้เคียงกับ $C \times < E >$ ค่าพลังงานปัจจุบันนั้นๆคือค่าบีต

จากขั้นตอนการตรวจจับบีต เราสามารถพัฒนาความเร็วและค่าความแม่นยำของการตรวจจับบีตได้โดยการเก็บค่าพลังงานเฉลี่ยที่ได้เป็นค่าๆหนึ่งลงในบัฟเฟอร์ โดยสร้างช่องเก็บข้อมูลให้บัฟเฟอร์สามารถเก็บได้ทั้งหมด 43 ช่องข้อมูล จะเรียกบัฟเฟอร์นี้ว่าบัฟเฟอร์ (E) ซึ่งจะมีค่าเท่ากับ $E[43]$ ทำให้ข้อมูลพลังงานปัจจุบันที่จะเก็บได้ถึง 42 ค่า หรือ $E[42]$ โดย 42 ค่าที่เหลือจะเป็นค่าพลังงานปัจจุบันค่าเก่าและจะทำการเปรียบเทียบภายในตัวบัฟเฟอร์ ทำให้เกิดความแม่นยำและรวดเร็วมากยิ่งขึ้น วิธีการคือการพัฒนาสมการที่ 2.2 ก่อนจะทำตามขั้นตอนที่เหลือข้างต้น ซึ่งสมการใหม่ที่พัฒนาขึ้นมานี้ แสดงได้ดังขั้นตอนวิธีการตรวจจับบีต แบบที่ 2

2.1.2 ขั้นตอนวิธีการตรวจจับบีตในเชิงสถิติแบบที่ 2

2.1.2.1 แทนค่าอย่างละ 1024 แซมเปิลลงใน $a[n]$ และ $b[n]$ เพื่อคำนวณค่าพลังงานปัจจุบัน (Instant energy) ซึ่งในที่นี้จะแทนด้วยค่า e แทนด้วยสมการที่ 2.1

2.1.2.2 คำนวณค่า $< E >$ และใช้ค่า $E[i]$ แสดงแทนช่องข้อมูลของบัฟเฟอร์

$$< E >= \frac{1}{43} \times \sum_{i=0}^{43} (E[i])^2 \qquad (2.3)$$

2.1.2.3 เลื่อนบัฟเฟอร์เก็บค่าไปทางขวาอีก 1 ช่องข้อมูล ทำให้มีช่องข้อมูลเพิ่มขึ้น 1 ช่องสำหรับการจัดเก็บค่าพลังงานเฉลี่ยใหม่และลบค่าพลังงานเฉลี่ยเก่าทิ้งออกไป

2.1.2.4 แทนค่าพลังงาน e ใหม่ที่ช่อง $E[0]$

2.1.2.5 เปรียบเทียบค่า e กับ $C \times < E >$ หากค่า e มีขนาดพลังงานใกล้เคียงกับ $C \times < E >$ ค่าพลังงานปัจจุบันนั้นๆคือค่าบีต

ในเพลงแนวเทคโน (Techno) หรือแนวแร็พ (Rap) จะมีความชัดเจนของพลังงานเสียงสูงสุดหรือบีต (ซึ่งจะใช้ค่า C ประมาณ 1.4)โดยค่า C คือค่าความไวต่อบีต (Beat Sensibility) แต่สำหรับแนวเพลงฮาร์ดร็อค (Hard Rock) หรือเพลงที่มีเสียงแตก (Distortion) อันเกิดจากเสียงรบกวนเป็นจำนวนมากนั้นอาจทำให้เกิดความแปรปรวนของพลังงานเสียงสูงสุด จึงต้องคำนวณค่า C เพื่อใช้กับเพลงแนวดังกล่าว เพื่อให้เกิดความแม่นยำสูงสุด สำหรับการคำนวณหาค่า C จะใช้ "หลักการคำนวณหาค่าความแปรปรวนของพลังงานเสียง (V)" ทำให้สามารถพัฒนาการตรวจจับบีตโดยเพิ่มขั้นตอนดังกล่าวเพื่อให้เห็นภาพที่ชัดเจนยิ่งขึ้น พิจารณาขั้นตอนที่วิธีการตรวจจับบีตแบบที่ 3

2.1.3 ขั้นตอนวิธีการตรวจจับบีตในเชิงสถิติ แบบที่ 3

2.1.3.1 แทนค่าอย่างละ 1024 แซมเปิลลงใน $a[n]$ และ $b[n]$ เพื่อคำนวณค่าพลังงานปัจจุบัน (Instant energy) ซึ่งในที่นี้จะแทนด้วยค่า e แทนด้วยสมการที่ 2.1

2.1.3.2 คำนวณค่าพลังงานเฉลี่ย $< E >$ จากค่าพลังงานปัจจุบัน (E) ที่อยู่ในบัฟเฟอร์ $E[i]$ โดยใช้สมการที่ 2.2

2.1.3.3 คำนวณค่าความแปรปรวน (V) ของค่าพลังงานปัจจุบัน (E) ได้ดังสมการที่ 2.4

$$V = \frac{1}{43} \times \sum_{i=0}^{43} (E[i] - < E >)^2 \qquad (2.4)$$

2.1.3.4 จากนั้นคำนวณหาค่า C ได้ดังสมการที่ 2.5 โดยสมการนี้จะได้มาจากการเปรียบเทียบค่าที่คาดว่าเหมาะสมของ V กับ C ในรูปแบบเชิงเส้น

$$C = (-0.0025714 \times V) + 1.514287 \qquad (2.5)$$

2.1.3.5 เลื่อนบัฟเฟอร์เก็บค่าไปทางขวาอีก 1 ช่องข้อมูล ทำให้มีช่องข้อมูลเพิ่มขึ้น 1 ช่องสำหรับการจัดเก็บค่าพลังงานเฉลี่ยใหม่และลบค่าพลังงานเฉลี่ยเก่าทิ้งออกไป

2.1.3.6 เปรียบเทียบค่า e กับ $C \times < E >$ หากค่า e มีขนาดพลังงานใกล้เคียงกับ $C \times < E >$ ค่าพลังงานปัจจุบันนั้นๆคือค่าบีต

ทั้งสามขั้นตอนการตรวจจับบีตที่ได้กล่าวมาทั้งหมดได้ทำการทดสอบกับแนวดนตรีส่วนหนึ่ง อาทิเช่น ป๊อบ (Pop), ร็อค (Rock), เมทัล (Metal), เทคโน (Techno), แร็พ (Rap), คลาสสิค (Classical), พังค์ (Punk) เป็นต้น ผลที่ได้รับค่อนข้างยากที่จะคาดเดา โดยจะขอพูดถึงเฉพาะขั้นตอนวิธีการตรวจจับบีตในเชิงสถิติแบบที่ 3 เพียงเท่านั้น เพราะเป็นขั้นตอนที่พัฒนามาจาก ขั้นตอนแบบที่ 1 และ 2 โดยขั้นตอนแบบที่ 3 จะใช้ได้ผลดีเฉพาะดนตรีที่มีเสียงรบกวนน้อยเท่านั้น และแม่นยำมากกับดนตรีแนวเทคโนและแร็พ แม้ว่าจะทำการพัฒนาค่า C แล้วก็ตาม ดนตรีจำพวก พังก์, ร็อค และฮาร์ดร็อค ก็ยังคงแม่นยำเป็นบางเพลงเพียงเท่านั้น ไม่สามารถใช้ได้ผลกับทุกเพลง ซึ่งสามารถสัมผัสได้โดยธรรมชาติในทันทีว่าจังหวะจริงๆกับผลที่ออกมาไม่สอดคล้องกัน

สำหรับการอธิบายความผิดพลาดนี้ อาจเกิดจากการที่กีตาร์และฟลุต มีช่วงความถี่ที่ทับซ้อนกัน มีพลังงานเสียงใกล้เคียงกันเพียงแต่มี Pitch ของเสียงที่แตกต่างกัน เมื่อผ่านขั้นตอนวิธีการทั้งหมดเสียงเหล่านี้จัดเป็นแค่เสียงรบกวนที่ไม่มีพลังงานเสียงที่มีประโยชน์ต่อการคำนวณและยังเพิ่มความผิดพลาดให้แก่การคำนวณเสียด้วย ดังนั้นจึงสามารถแก้ไขปัญหาเหล่านี้ได้โดยใช้

"การวิเคราะห์การตรวจจับบีตในเชิงสถิติแบบเลือกใช้ช่วงความถี่ (Statistical streaming Beat detection by Frequency selected)"

การวิเคราะห์แบบเลือกใช้ช่วงความถี่เป็นการพัฒนาหลักการของการวิเคราะห์การตรวจจับบีตในเชิงสถิติแบบเก่า เพื่อแก้ปัญหาสัญญาณที่มีเสียงรบกวนจำนวนมากดังที่ได้กล่าวไป โดยหลักการนี้จะทำการแก้ปัญหาด้วยการแบ่งความถี่ออกเป็นซับแบนด์ต่างๆทั้งช่วงความถี่สูงและต่ำ ซึ่งพลังงานเสียงของโดเมนเวลากับโดเมนความถี่จะมีค่าเท่ากัน ขั้นตอนแรกนำสัญญาณจากสัญญาณต้นแบบจาก $a[n]$ และ $b[n]$ ในลักษณะของสัญญาณเสียงของไฟล์ .wav หรือสัญญาณเสียงอินพุตจากไมโครโฟน ทำการสะสม 1024 แซมเปิลลงในบัฟเฟอร์ในแต่ละช่วงเวลา จากนั้นจะเข้าสู่กระบวนการแปลงฟูเรียร์อย่างง่าย (Fast fourier transform : FFT) ทำให้ได้ผลลัพธ์เป็นสเปกตรัมความถี่ 1024 สเปกตรัม แล้วทำการแบ่งสเปกตรัมทั้งหมดลงในแต่ละซับแบนด์ ยิ่งมีซับแบนด์มากก็ยิ่งเพิ่มความไวต่อการตรวจจับบีตมากขึ้น (ในที่นี้จะใช้ 32 ซับแบนด์) และใช้ได้ผลดีกับแนวเพลงต่างๆมากยิ่งขึ้น จากนั้นจะคำนวณพลังงานเสียงในแต่ละซับแบนด์เปรียบเทียบกับพลังงานเสียงเฉลี่ยก่อนหน้าของซับแบนด์นั้นๆ หากพลังงานเสียงปัจจุบันของซับแบนด์นั้นมีค่ามากกว่าพลังงานเสียงเฉลี่ยก่อนหน้า พลังงานเสียงนั้นคือบีต พิจารณาขั้นตอนวิธีการตรวจจับบีตในเชิงสถิติแบบแบ่งช่วงความถี่ ได้ดังนี้

2.1.4 ขั้นตอนวิธีการตรวจจับบีตในเชิงสถิติแบบแบ่งช่วงความถี่

2.1.4.1 คำนวณ FFT ของ 1024 แซมเปิลที่นำมาจากสัญญาณ $a[n]$ และ $b[n]$ เมื่อผ่านกระบวนการ FFT จะได้ค่าจำนวนเชิงซ้อน ซึ่งแบ่งออกเป็นส่วนจริง (Real part) จาก $a[n]$ และส่วนจินตภาพ (Imaginary Part) จาก $b[n]$ ดังสมการที่ 2.6 โดยการ FFT จะต้องคำนวณทั้ง 1024 แซมเปิล

$$(a_n) + i \times (b_n) \tag{2.6}$$

2.1.4.2 นำค่าที่ได้จาก FFT เก็บลงในบัฟเฟอร์ ดังนั้นบัฟเฟอร์ (B) จะเก็บค่าแอมปลิจูดของความถี่ไว้ทั้งหมด 1024 ค่า

2.1.4.3 แบ่งบัฟเฟอร์ออกเป็น 32 ซับแบนด์ ทำการคำนวณพลังงานในแต่ละซับแบนด์ และเก็บค่าพลังงานของแต่ละซับแบนด์ (E_s) ดังนั้น E_s จะมี 32 ขนาด และ $E_s[i]$ จะเก็บพลังงานของซับแบนด์ i เอาไว้ แสดงได้ดังสมการที่ 2.7

$$E_s[i] = \frac{32}{1024} \times \sum_{k=i\times 32}^{(i+1)\times 32} B[k] \tag{2.7}$$

2.1.4.4 จากนั้นคำนวณหาค่าพลังงานเฉลี่ยในแต่ละซับแบนด์ i (E_i) โดยใช้สมการที่ 2.8

$$< E_i >= \frac{1}{43} \times \sum_{k=0}^{42} Ei[k] \tag{2.8}$$

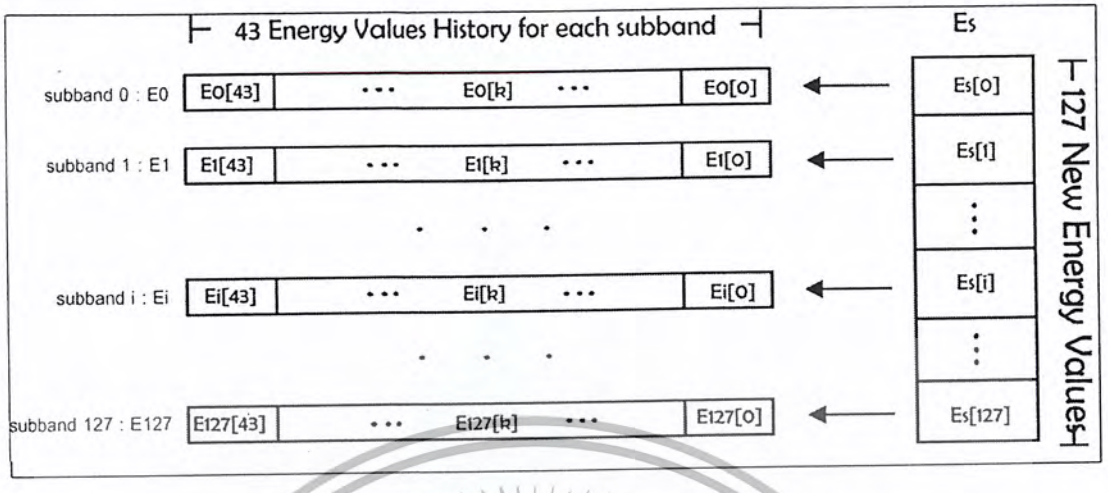
2.1.4.5 เลื่อนบัฟเฟอร์เก็บค่าไปทางขวาอีก 1 ช่องข้อมูล ทำให้มีช่องข้อมูลเพิ่มขึ้น 1 ช่องสำหรับการจัดเก็บค่าพลังงานเฉลี่ยใหม่ในแต่ละซับแบนด์และลบค่าพลังงานเฉลี่ยเก่าทิ้งออกไป

2.1.4.6 จัดเก็บค่าพลังงานเฉลี่ยของแต่ละซับแบนด์ใหม่ ลงใน $E_i[0]$ พิจารณาสมการที่ 2.9

$$E_i[0] = E_s[i] \tag{2.9}$$

2.1.4.7 หากในแต่ละซับแบนด์ i หาก $E_s[i] > (C\times < E_i >)$ ค่าพลังงานนั้นๆคือค่าบีต

สามารถแสดงการเก็บข้อมูลในแต่ละซับแบนด์ได้ดังรูปที่ 2.1



รูปที่ 2.1 การเก็บข้อมูลในแต่ละซับแบนด์ของเฟดริก พาติน

สำหรับค่า C ในขั้นตอนวิธีการนี้ไม่จำเป็นต้องทำการคำนวณเพราะวิธีการนี้ได้ทำการแยกค่าพลังงานลงในแต่ละซับแบนด์ทำให้เกิดค่าความแปรปรวนเป็นอย่างมาก สามารถใช้ค่า $C = 250$ ได้ทันที (ตั้งไว้ที่ค่าที่สูงมาก) ผลลัพธ์ค่อนข้างเป็นที่น่าพอใจสามารถแก้ปัญหาเสียงรบกวนของแนวเพลงต่างๆได้อาทิเช่น เสียงแตกของกีตาร์ (Distortion) เป็นต้น แต่เนื่องจากเสียงฉาบของกลองไม่สามารถนำมาคำนวณได้เพราะฉาบมีโดเมนความถี่ที่ทับซ้อนกับเครื่องดนตรีชนิดอื่นๆ

อีกแนวทางหนึ่งในการช่วยพัฒนาขั้นตอนวิธีการตรวจจับบีตในเชิงสถิติแบบแบ่งช่วงความถี่สามารถพัฒนาให้มีความละเอียดและแม่นยำมากขึ้นด้วยการแบ่งซับแบนด์เพิ่มขึ้นจากเดิม 32 ซับแบนด์เป็น 64 ซับแบนด์ แต่ก็จะทำให้การประมวลผลช้าลงเล็กน้อย อีกแนวทางหนึ่งคือ อาศัยหลักการระบบการได้ยินของมนุษย์ที่ไม่สมบูรณ์แบบ หูของมนุษย์นั้นมีลักษณะคล้ายกับวงจรกรองความถี่ต่ำ กล่าวคือ เสียงรบกวนที่มนุษย์ได้ยินจะได้ยินชัดเฉพาะ Pitch เสียงต่ำ หากเป็น Pitch เสียงสูงก็แทบจะไม่ได้ยินเลย ดังนั้นจึงทำการแบ่งความกว้างของแต่ละซับแบนด์ให้มีค่าไม่เท่ากัน ตัวอย่างเช่น ในซับแบนด์[0] จะมีค่าความถี่อยู่เพียง 2 ค่า แต่ในซับแบนด์[64] จะมีค่าความถี่อยู่ 20 ค่าเป็นต้นโดยความกว้างของแต่ละซับแบนด์ (wi) ของ n ซับแบนด์ สามารถแสดงได้ดังขั้นตอนการพัฒนาปรับปรุงวิธีการตรวจจับบีตในเชิงสถิติแบบแบ่งช่วงความถี่ด้วยการเพิ่มซับแบนด์และแบ่งความกว้างของซับแบนด์ได้ดังนี้

2.1.5 ขั้นตอนการพัฒนาปรับปรุงด้วยการเพิ่มซับแบนด์และแบ่งความกว้างของซับแบนด์

2.1.5.1 เพิ่มความกว้างของซับแบนด์แบบเชิงเส้น ด้วยสมการที่ 2.10

$$wi = a \times i + b \tag{2.10}$$

จากสมการที่ 2.10 ตัวอย่างการเพิ่มความกว้างของซับแบนด์ที่ 1 สามารถแสดงได้ดังสมการที่ 2.11

$$w1 = a + b \tag{2.11}$$

2.1.5.2 ค่าความกว้างของซับแบนด์รวมทั้งหมดจะต้องมีค่าไม่เกิน 1024 (ขนาดของบัฟเฟอร์) อธิบายได้ดังสมการที่ 2.12

$$\sum_{i=1}^{n} wi = 1024 = n \cdot b + a \cdot \sum_{i=1}^{n} i = n \cdot b + a \cdot \frac{n \cdot (n+1)}{2} = 1024 \tag{2.12}$$

เมื่อได้ค่า wi จากวิธีการข้างต้นแล้ว จะนำค่า wi ที่ได้และหลักการเพิ่มซับแบนด์เป็น 64 ซับแบนด์ไปประยุกต์กับสมการที่ 2.7 ทำให้ได้สมการที่ 2.13 ดังนี้

$$E_s[i] = \frac{wi}{1024} \times \sum_{k=\sum_{j=1}^{i-1} wj}^{\sum_{j=1}^{i} wj} B[k] \tag{2.13}$$

ภายหลังจากได้นำขั้นตอนการพัฒนาปรับปรุงด้วยการเพิ่มซับแบนด์และแบ่งความกว้างของซับแบนด์มาประยุกต์ใช้เพื่อให้ได้ผลลัพธ์ที่ละเอียดและแม่นยำมากขึ้น ให้แทนสมการที่ 2.7 ด้วยสมการ 2.13 ในขั้นตอนวิธีการตรวจจับบีตในเชิงสถิติแบบแบ่งช่วงความถี่ แต่ถ้าหาก

ต้องการจะเพิ่มคุณสมบัติในการตัดแบ่งช่วงความถี่ สามารถขั้นตอนการพัฒนาปรับปรุงด้วยวิธีการเลือกใช้ช่วงความถี่

2.1.5.5 ขั้นตอนการพัฒนาปรับปรุงด้วยวิธีการเลือกใช้ช่วงความถี่

1) ถ้า 'i' < 'N/2' ดังนั้น (if 'i' < 'N/2' then)

$$f = \frac{i \times f_e}{N} \tag{2.14}$$

2) แต่ถ้า 'i' >= 'N/2' ดังนั้น (else if 'i' >= 'N/2' then)

$$f = \frac{(N-i) \times f_e}{N} \tag{2.15}$$

โดยที่ i คือตำแหน่งซับแบนด์ภายในบัฟเฟอร์เก็บค่าเอาท์พุตของ FFT และ N คือขนาดของการแปลง FFT (1024) , f_e คือความถี่แซมเปิล (44100 แซมเปิล)

อีกวิธีการหนึ่งที่สามารถเพิ่มความเที่ยงตรงให้กับการตรวจจับบีตเป็นอย่างยิ่ง คือ การเพิ่มเงื่อนไขในขั้นตอนสุดท้าย เพื่อคัดเลือกค่าพลังงานสูงสุดที่ดีที่สุด โดยทำการเปรียบเทียบจากค่าความแปรปรวนของพลังงานในแต่ละซับแบนด์ หากค่าความแปรปรวนสูงหมายถึงค่าพลังงานสูงสุดนั้นมีความแตกต่างจากค่าพลังงานอื่นๆเป็นอย่างยิ่งทำให้มีความชัดเจนในการตรวจจับบีตมากยิ่งขึ้น วิธีการดังกล่าวสามารถทำได้ดังนี้

2.1.6 ขั้นตอนการพัฒนาปรับปรุงด้วยวิธีการเพิ่มเงื่อนไขการเปรียบเทียบค่าความแปรปรวนของพลังงานในแต่ละซับแบนด์

เพิ่มเงื่อนไข $V((Ei)) > V_o$ ลงในการเปรียบเทียบค่า e กับ $C \times < E >$ จะได้เงื่อนไขใหม่ดังนี้ เปรียบเทียบค่า e กับ $C \times < E >$ และ $V((E_i)) > V_0$ โดย V_0 คือค่าจำกัดความแปรปรวน ซึ่งสามารถคำนวณได้จากสมการที่ 2.16 ดังนี้

$$V((E_i)) = \frac{1}{43} \times \sum_{k=0}^{42} (E_i[k] - < E_i >)^2 \tag{2.16}$$

2.1.7 แนวคิดการตรวจจับบีตด้วยวิธีคัดกรองด้วยฟิลเตอร์

หากมีสัญญาณ $x(t)$ และ $y(t)$ สามารถที่จะสร้างฟังก์ชั่นที่จะเป็นไอเดียในการอธิบายความคล้ายคลึงกันของทั้งสองสัญญาณ โดยเรียกฟังก์ชั่นนี้ว่า "ฟังก์ชั่นความสัมพันธ์ของสัญญาณ" สามารถเขียนเป็นสมการได้ดังนี้

$$\gamma_{xy}(\alpha, \beta) = \int_{-\infty}^{+\infty} [x(t) \cdot y(\alpha \cdot (t - \beta)) \cdot dt] \tag{2.17}$$

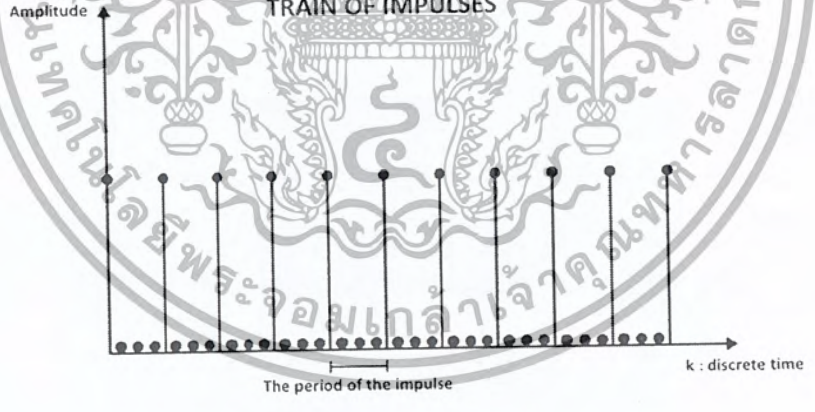
สมการนี้เป็นสมการการคำนวณหาพลังงานที่สามารถแลกเปลี่ยนระหว่างสัญญาณ x(t) กับสัญญาณ $y(t - \beta)$ ซึ่งดูคล้ายคลึงกัน และกำหนดให้มี β อยู่ในสมการเพื่อกำจัดปัญหาจุดกำเนิดสัญญาณบนแกนเวลา เพราะการนำสองสัญญาณมาเปรียบเทียบกันควรปราศจากความเป็นไปได้เกี่ยวกับแกนเวลา แต่จะต้องทราบว่าสมการที่ 2.17 จะต้องถูกเขียนให้อยู่ในรูปสมการที่คอมพิวเตอร์เข้าใจได้ จึงต้องเขียนให้อยู่ในรูปสมการที่ 2.18 ดังนี้

$$\gamma_{xy}(\beta) = \int_{k=-\infty}^{+\infty} [x[k] \cdot y[k - \beta]] \tag{2.18}$$

ประเด็นสำคัญคือสามารถประมาณค่าความคล้ายคลึงระหว่างเพลงกับขบวนอิมพัลส์ โดยใช้ฟังก์ชั่นตามสมการ ขบวนอิมพัลส์ซึ่งถูกแปลงมาในรูปของความถี่ (Beats per minute ที่ต้องการนำมาทดสอบ) ซึ่งเรียกว่า Comb Filter และพลังงานจากสมการ γ ทำให้สามารถคำนวณค่าเปรียบเทียบระหว่างเพลงกับขบวนอิมพัลส์ ด้วยเหตุนี้จึงคำนวณหาค่าจังหวะของขบวนอิมพัลส์ที่เกิดในเพลง ถ้าคำนวณหาพลังงานของสมการ γ ด้วยขบวนอิมพัลส์ที่ต่างกัน ก็จะเห็นพลังงานที่มีค่ามากที่สุดเกิดขึ้น ดังนั้นจึงทำให้ทราบหลักการในอัตราการเกิด Beats per minute พิจารณาจากสมการที่ 2.19 แทนพลังงานด้วย E_y และ $X[k]$ เป็นสัญญาณเสียงเพลง สำหรับ $Y[k]$ เป็นขบวนอิมพัลส์

$$E_y = \sum_{\beta=-\infty}^{+\infty} \gamma_{xy}[\beta]^2 \qquad (2.19)$$

เพื่อแสดงขบวนอิมพัลส์ให้เห็นภาพที่ชัดเจนขึ้น ขอให้พิจารณาขบวนอิมพัลส์โดยแสดงลักษณะเฉพาะของคาบเวลาระหว่างพัลส์ (Ti) ดังรูปที่ 2.2



รูปที่ 2.2 ขบวนอิมพัลส์โดยแสดงลักษณะเฉพาะของคาบเวลาระหว่างพัลส์ (Ti)

ดังนั้น คาบเวลา Ti ของอิมพัลส์จึงต้องสอดคล้องกับ Beats per minute ที่ต้องการทดสอบในเพลงรูปแบบนี้ทำให้ค่าของ Beats per minute กับ Ti แบบไม่ต่อเนื่องทางเวลาได้ดังสมการที่ 2.20

$$Ti = \frac{60}{BPM} \times fs \qquad (2.20)$$

จากสมการที่ 2.20 fs ก็คือ Sample Frequency โดยปกติอยู่ที่ 44100 เฮิร์ต(ตามมาตรฐานไฟล์ .wav) BPM หมายถึง Beats per minute rate หรืออัตราการเกิดบีตต่อนาที และ Ti คือคาบเวลาของขบวนอิมพัลส์ (Impulse Train) แต่เพราะว่าการคำนวณที่ยุ่งยากในสมการที่ 2.18 จึงใช้วิธีการมองที่แกนของความถี่โดยใช้ FFT ในการคำนวณผลลัพธ์ของ $X[k]$ และ $Y[k]$ (FFT ของ $X[k]$ และ $Y[k]$) จึงหาค่าพลังงานของสมการ γ ได้โดยตรงในแกนของความถี่ ดังนั้น E_y จะได้รูปแบบสมการดังนี้

$$E_y = \sum_{k=0}^{N} \left| X[k] \cdot Y[k] \right|^2 \qquad (2.21)$$

ในขั้นตอนนี้การคำนวณค่อนข้างสิ้นเปลืองทรัพยากรหากจะคำนวณข้อมูลทั้งหมดในเพลง จึงใช้วิธีการตัดช่วงกลางของเพลงออกมา 5 วินาที เป็นการสมมุติให้จังหวะของเพลงคงที่ตลอดและใช้ส่วนของกลางเพลงที่ดึงออกมาเป็นตัวแทนทั้งหมดของเพลง พิจารณาขั้นตอนการทำงานได้ดังนี้

2.1.7.1 ดึงข้อมูลจากบัฟเฟอร์ข้อมูลของเพลงทุกๆ 5 วินาที เก็บลงใน $a[k]$ และ $b[k]$ โดยค่าของ $a[k]$ และ $b[k]$ เป็น N จากนั้นกำหนดให้ $a[k]$ เป็นค่าทางฝั่งซ้าย และ $b[k]$ เป็นค่าทางฝั่งขวา (Stereo Stream)

2.1.7.2 แปลง FFT ของสัญญาณออกมา ได้ $a[k]$เป็นส่วนจริง (Real part) และ $b[k]$ เป็นส่วนจินตภาพ (Imaginary part) หลังจากนั้นเก็บค่าส่วนจริงลงใน $ta[k]$ และส่วนจินตภาพลงใน $tb[k]$

2.1.7.3 สำหรับ Beats Per Minute ที่นำมาทดสอบ ในตัวอย่างนี้เราใช้ตั้งแต่ 60 BPM (บีตระดับเพลงช้า) ถึง 180 BPM (บีตระดับเพลงเร็ว) ทำซ้ำกันทั้งหมด 10 ครั้ง จากนั้นแทน BPM ที่ใช้ทดสอบด้วย BPMc

2.1.7.4 คำนวณค่าของ Ti ได้โดยใช้สมการที่ 2.20

2.1.7.5 คำนวณค่าขบวนอิมพัลส์และเก็บค่าลงใน (i) และ (j) นำค่าที่เก็บได้ทั้งสองค่ามาเป็นสัญญาณสเตอริโอ โดยวิธีการสร้างขบวนอิมพัลส์พิจารณาได้จากคำสั่งด้านล่าง

```
for (k = 0; k < N; k++)
{
 if ((k % Ti) == 0)
 l[k] = j[k] = AmpMax;
 else
 l[k] = j[k] = 0;
 }
```

2.1.7.6 จากนั้นแปลง FFT ของสัญญาณได้ (i) และ (j) เก็บค่าที่ได้ลงใน $tl[k]$ และ $tj[k]$

2.1.7.7 สุดท้ายคำนวณค่าพลังงานความสัมพันธ์ระหว่างขบวนอิมพัลส์กับสัญญาณเพลง 5 วินาที โดยพิจารณาจากสมการที่ 2.21 จะได้สมการใหม่ดังนี้

$$E_{BPMc} = \sum_{k=0}^{N} \left| \left(ta[k] + i \cdot tb[k] \right) \cdot \left(tl[k] + i \cdot tj[k] \right) \right| \qquad (2.22)$$

2.1.7.8 บันทึกค่าผลลัพธ์ของ E_{BPMc}

2.1.7.9 จังหวะของเพลงนั้นเกิดจากค่าของ BPM max เมื่อค่าสูงสุดของ E_{BPMc} เกิดขึ้นจะได้ค่าอัตราการเกิด Beat per minute ของเพลง

ค่าของ AmpMax ที่ปรากฏขึ้นบนขบวนของอิมพัลส์ กำหนดโดยค่า Sample size ที่มาจากไฟล์ โดยปกติแล้วค่า Sample size จะอยู่ที่ 16 บิต เป็นค่าที่มีประสิทธิภาพดีเยี่ยม ถ้าใช้งานในรูปแบบของ Non Sign signal ค่าของ AmpMax จะอยู่ที่ 65535 แต่ถ้าใช้งานในรูปแบบ 16 บิต Sign sample ค่าของ AmpMax จะอยู่ที่ 32767

และ เพื่อให้บีตถูกตรวจจับได้ง่ายขึ้น สามารถเพิ่มประสิทธิภาพด้วยการทำให้สัญญาณ $a[k]$ และ $b[k]$ เกิดความแตกต่างด้วยสมการที่ 2.23 โดยนำขั้นตอนนี้ไปไว้ก่อนขั้นตอนที่ 2.1.1.2

$$\forall k \in [1, N-2]$$
$$a[k] = \frac{1}{2} fs \cdot (a[k+1] - a[k-1]) \qquad (2.23)$$
$$b[k] = \frac{1}{2} fs \cdot (b[k+1] - b[k-1])$$

2.1.8 การปรับปรุงประสิทธิภาพด้วยวิธีเลือกความถี่พลังงานเสียง (Frequency selected sound energy)

ในหัวข้อที่ 2.1.7 นั้น เกิดปัญหาหนึ่งคือบีตของฉาบกับบีตของกลองนั้นจับได้เหมือนกัน ซึ่งการตรวจจับบีตนั้นไม่เหมาะสมกับเพลงที่มีเสียงรบกวนมากๆเช่น ดนตรีแนวฮาร์ดร็อค เพื่อแก้ปัญหาดังกล่าวจึงทำให้เกิดกระบวนการวิธีเลือกความถี่พลังงานเสียง Frequency selected sound energy โดยทำการแยกสัญญาณแบ่งลงในซับแบนด์หลายๆซับแบนด์ เริ่มจากการแบ่ง $ta[k]$ และ $tb[k]$ ลงในหลายๆซับแบนด์ ค่าของอาร์เรย์จะถูกเรียกว่า $tas[k]$ และ $tbs[k]$ (‘s’ มีค่าตั้งแต่ 1-16) ขั้นตอนวิธีการจะมีบางส่วนที่คล้ายกันกับวิธีแรกแต่ขั้นตอนจะเพิ่มขึ้นมาเพื่อปรับปรุงประสิทธิภาพเมื่อทำการตรวจจับบีตกับเพลงแนวฮาร์ดร็อค หรือเพลงที่มีเสียงรบกวนเยอะแสดงได้ดังนี้

2.1.8.1 ดึงข้อมูลจากบัฟเฟอร์ข้อมูลของเพลงทุกๆ 5 วินาที เก็บลงใน $a[k]$ และ $b[k]$ โดยค่าของ $a[k]$ และ $b[k]$ เป็น N จากนั้นกำหนดให้ $a[k]$ เป็นค่าทางฝั่งซ้าย และ $b[k]$ เป็นค่าทางฝั่งขวา (Stereo Stream)

2.1.8.2 สร้างความแตกต่างของสัญญาณ $a[k]$ และ $b[k]$ โดยสมการที่ 2.21

2.1.8.3 แปลง FFT ของสัญญาณออกมา ได้ $a[k]$เป็นส่วนจริง (Real part) และ $b[k]$ เป็นส่วนจินตภาพ (Imaginary part) หลังจากนั้นเก็บค่าส่วนจริงลงใน $ta[k]$ และส่วนจินตภาพลงใน $tb[k]$

2.1.8.4 สร้างอาเรย์ขนาดต่างๆ 16 ซับแบนด์เพื่อเก็บค่า $tas[k]$ และ $tbs[k]$ จากนั้นตัด $ta[k]$ และ $tb[k]$ ทิ้งด้วยกฎของลอการิธึม

2.1.8.5 สำหรับซับแบนด์ทั้งหมด $tas[k]$ และ $tbs[k]$ (s มีค่าตั้งแต่1-16) Ws = length ของ 's'

2.1.8.6 สำหรับ Beats Per Minute ที่นำมาทดสอบ ในตัวอย่างนี้เราใช้ตั้งแต่ 60 BPM (บีตระดับเพลงช้า) ถึง 180 BPM (บีตระดับเพลงเร็ว) ทำซ้ำกันทั้งหมด 10 ครั้ง จากนั้นแทน BPM ที่ใช้ทดสอบด้วย BPMc

2.1.8.7 คำนวณค่าของ Ti ได้โดยใช้สมการที่ 2.20

2.1.8.8 คำนวณค่าขบวนอิมพัลส์ และเก็บค่าลงใน (i) และ (j) นำค่าที่เก็บได้ทั้งสองค่ามาเป็นสัญญาณสเตอริโอ โดยวิธีการสร้างขบวนอิมพัลส์ พิจารณาได้จากคำสั่งด้านล่าง

```
for (k = 0; k < ws; k++)
{
if ((k % Ti) == 0)
l[k] = k[k] = AmpMax;
else
l[k] = j[k] = 0;
}
```

2.1.8.9 จากนั้นแปลง FFT ของสัญญาณได้ (i) และ (j) เก็บค่าที่ได้ลงใน $tl[k]$ และ $tj[k]$

2.1.8.10 สุดท้ายคำนวณค่าพลังงานความสัมพันธ์ระหว่างขบวนอิมพัลส์กับสัญญาณเพลง 5 วินาที โดยพิจารณาจากสมการที่ 2.22 จะได้สมการใหม่ดังนี้

$$E_{(BPMc,s)} = \sum_{k=0}^{ws} (tas[k] + i \cdot tbs[k]) \cdot (tl[k] + i \cdot tj[k]) | \qquad (2.24)$$

2.1.8.11 บันทึกค่าผลลัพธ์ของ E_{BPMc}

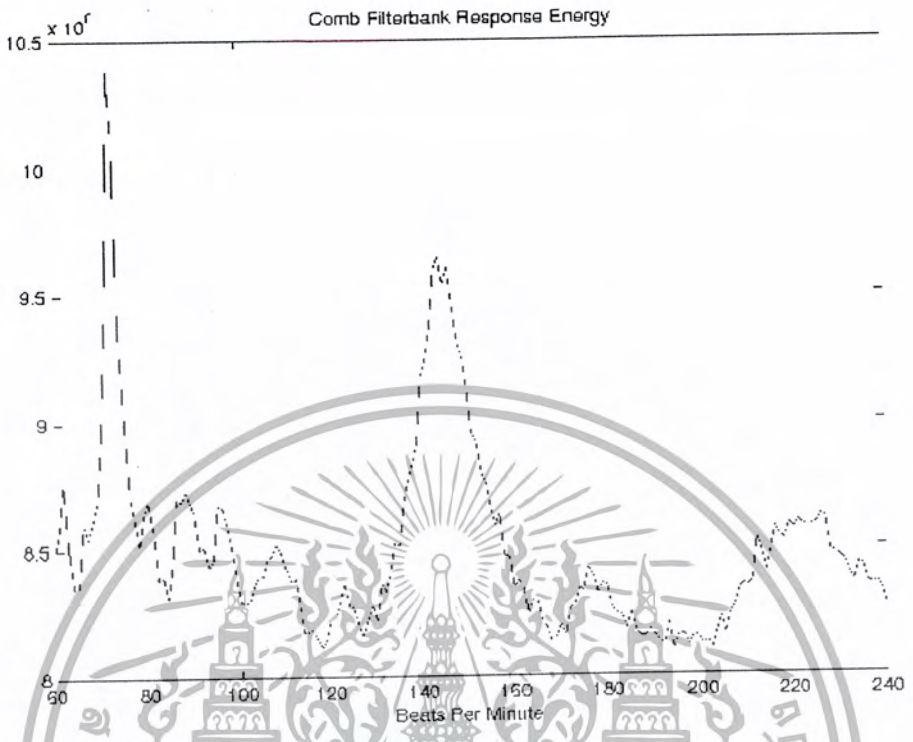
2.1.8.12 จังหวะของเพลงนั้นเกิดจากค่าของ $BPM \max$ เมื่อค่าสูงสุดของ E_{BPMc} เกิดขึ้นจะได้ค่าอัตราการเกิด Beat per minute ของเพลง

2.1.8.13 ทำตามขั้นตอนที่ 2.1.8.12 ในทุกๆซับแบนด์ เกิดค่า $BPM \max s$ ในทุกๆซับแบนด์ จึงสามารถเลือกมาพิจารณาได้

ในส่วนของการปรับปรุงประสิทธิภาพวิธีเลือกความถี่พลังงานเสียง (Frequency selected sound energy) ต้องเลือกค่าจังหวะตามหลักการของความถี่แบนด์วิธหรือถ้าต้องการค่าในทุกๆซับแบนด์ก็สามารถคำนวณ BPM ได้จาก $BPM \max s$ ทั้งหมดที่เกิดจาก $EBPM \max s$ ดังสมการที่ 2.25

$$BPM = \frac{1}{\sum_{s=1}^{16} (EBPM \max s)} \cdot \sum_{s=1}^{16} (EBPM \max s) \cdot (BPM \max s) \qquad (2.25)$$

พิจารณาจากรูปที่ 2.3 ผลลัพธ์นี้ได้จากโปรแกรม MATLAB จะเห็นได้ว่าค่า E_{BPMc} แสดงค่ามากที่สุดนั้นมีค่าระดับ $BPMc$ = 75 ฉะนั้น Beat per minute ของเพลงนี้จึงมีค่า 75 (75 ครั้งต่อนาที)



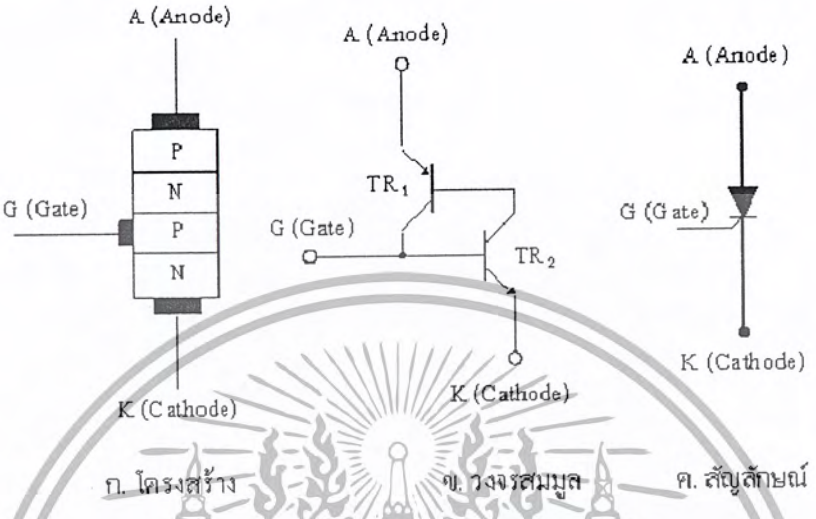
รูปที่ 2.3 กราฟแสดงค่า E_{BPMc} เพื่อใช้ในการหา Beat per minute ของเพลง [12]

2.2 ทฤษฎีพื้นฐานของเอสซีอาร์ (SCR)

2.2.1 โครงสร้างและสัญลักษณ์ของเอสซีอาร์

ซิลิคอน คอนโทรล เร็คติไฟร์เออร์ (Silicon Control Rectifier) หรือ เอสซีอาร์ (SCR) เป็นอุปกรณ์โซลิดสเตท (Solid-State) ที่ทำหน้าที่เป็นสวิตช์เปิด – ปิด (On – Off) วงจรทางอิเล็กทรอนิกส์ชนิดหนึ่ง รวมทั้งเอสซีอาร์ (SCR) ยังจัดเป็นอุปกรณ์สารกึ่งตัวนำประเภท "ไทริสเตอร์" (Thyristor) ข้อดีของสวิตช์อิเล็กทรอนิกส์คือจะไม่มีหน้าสัมผัสหรือเรียกว่าคอนแท็ค (Contact) ขณะปิด – เปิดจึงไม่ทำให้เกิดประกายไฟที่หน้าสัมผัสจึงมีความปลอดภัยสูงซึ่งสวิตช์ธรรมดาคือแบบกลไกที่มีหน้าสัมผัสจะไม่สามารถนำไปใช้ในบางสถานที่ได้สวิตช์อิเล็กทรอนิกส์

บางครั้งเรียกว่า "โซลิดสเตทสวิตช์" (Solid State Switch) แสดงโครงสร้าง วงจรสมมูล และสัญญลักษณ์ของเอสซีอาร์ได้ดังรูปที่ 2.4



รูปที่ 2.4 (ก) โครงสร้าง (ข) วงจรสมมูล และ (ค) สัญลักษณ์ของเอสซีอาร์ [2]

ถ้าดูจากรูปที่ 2.4 (ก) โครงสร้างของเอสซีอาร์ (SCR) ประกอบไปด้วยสารกึ่งตัวนำ 4 ชิ้นคือ พี – เอ็น – พี – เอ็น (P – N – P – N) มีจำนวน 3 รอยต่อ มีขาต่อออกมาใช้งาน 3 ขาคือ ขาแอโนด (A : Anode) ขาแคโทด (K : Cathode) และขาเกต (G : Gate) ซึ่งสภาวะการทำงานของเอสซีอาร์ (SCR) สามารถแบ่งการทำงานออกได้เป็น 2 สภาวะคือ

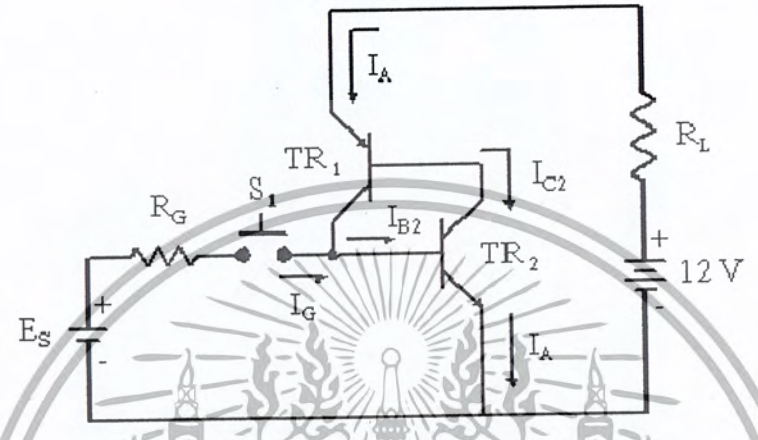
1) สภาวะนำกระแส เรียกว่า ON

2) สภาวะหยุดนำกระแส เรียกว่า OFF

2.2.2 สภาวะนำกระแสของเอสซีอาร์

พิจารณาจากรูปที่ 2.4 การที่จะทำให้เอสซีอาร์นำกระแสสามารถทำได้โดยจุดชนวนเรียกว่า "ทริกเกอร์" (Trigger) ด้วยกระแสเกต (I_G) ให้แก่เอสซีอาร์ (SCR) และที่ขั้วแอโนด (A) และแคโทด (K) ได้รับไบอัสตรงคือที่แอโนดได้แรงดันบวก (+) และที่แคโทดได้รับแรงดันลบ (-) ทำให้เกิดกระแส I_{B2} ไหลเข้าขาเบส (Base) ของทรานซิสเตอร์ T_{R2} ทำให้ T_{R2} อยู่ในสภาวะนำกระแส (ON) จะเกิดกระแสคอลเลคเตอร์ (I_{C2}) ไหลผ่าน T_{R2} ซึ่งก็คือกระแส I_{B1} ของ

ทรานซิสเตอร์ T_{R1} นั่นเอง ดังนั้น T_{R1} จึงนำกระแสด้วย ค่าความต้านทานระหว่างขั้วแอโนด (A) และแคโทด (K) จึงมีค่าต่ำมากเป็นผลให้เกิดกระแสแอโนด (I_A) ไหลผ่านอีมิตเตอร์ของ T_{R1} ไปออกที่อีมิตเตอร์ของ T_{R2} สภาวะการทำงานของเอสซีอาร์เปรียบเสมือนสวิตช์ปิดวงจร



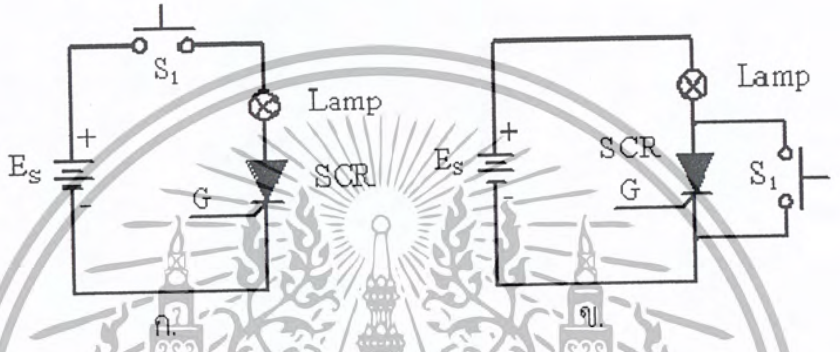
รูปที่ 2.5 การจุดชนวนให้เอสซีอาร์นำกระแส [2]

เมื่อเอสซีอาร์นำกระแสแล้วไม่จำเป็นต้องคงค่ากระแสเกต (I_G) ไว้ตลอดไป สามารถลดค่ากระแสเกตให้เป็นศูนย์ ($I_G = 0$) หรือปลดกระแสเกตออกได้โดยที่เอสซีอาร์ยังคงนำกระแสต่อไปเพราะ I_{B2} ที่ไหลเข้าเบสของ T_{R2} จะไหลมาจากคอลเลคเตอร์ของ T_{R1} ดังนั้นถึงแม้จะไม่มีกระแสเกตเอสซีอาร์ก็ยังคงนำกระแสต่อไปได้ ในสภาวะนำกระแสนี้ถ้าแหล่งจ่ายเป็นไฟกระแสสลับสามารถจะบังคับให้เอสซีอาร์นำกระแสได้มากหรือน้อยได้โดยเลือกมุมจุดชนวนที่เกตให้เหมาะสม

2.2.3 สภาวะหยุดนำกระแสของเอสซีอาร์

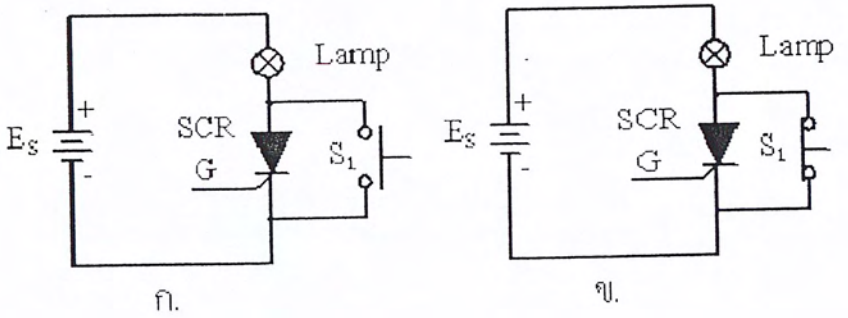
วิธีการทำให้เอสซีอาร์หยุดนำกระแสมีหลักการคือ ทำให้กระแสแอโนด (I_A) ลดลงจนต่ำกว่ากระแสโฮลดิ้ง (I_H : Holding Current คือค่ากระแสต่ำสุดที่ทำให้เอสซีอาร์นำกระแส) หรือ $I_A < I_H$ จึงจะทำให้เอสซีอาร์หยุดนำกระแสได้ซึ่งการที่จะทำให้เอสซีอาร์หยุดนำกระมี 2 วิธี คือ

2.2.3.1 แอโนดเคอเรนท์อินเทอรัพชั่น (Anode Current Interruption) โดยการตัดกระแส I_A ไม่ให้ไหลผ่านแอโนดของเอสซีอาร์ วิธีง่ายๆ ดังรูปที่ 2.6 (ก)โดยต่อสวิตช์อนุกรมแอโนด (A) ของเอสซีอาร์และเปิดสวิตช์เมื่อต้องการทำให้เอสซีอาร์หยุดทำงาน (Turn – Off) หรืออีกวิธีในรูปที่ 2.6 (ข) โดยต่อสวิตช์ระหว่างขั้วแอโนดและแคโทดของเอสซีอาร์เป็นการเปลี่ยนทางเดินของกระแสแอโนด (I_A) ไม่ให้ไหลผ่านเอสซีอาร์



รูปที่ 2.6 การทำให้เอสซีอาร์หยุดนำกระแสโดยวิธี Anode Current Interruption [2]

2.2.3.2 ฟอร์ชคอมมิวเทชั่น (Forced Commutation) วิธีนี้ทำได้โดยบังคับให้เอสซีอาร์ได้รับไบอัสกลับโดยใช้สวิตช์ขนานกับเอสซีอาร์เป็นตัวควบคุมการหยุดนำกระแสของเอสซีอาร์ดังรูปที่ 2.7 ถ้าสวิตช์เปิดวงจรเอสซีอาร์ยังคงนำกระแสอยู่ แต่ถ้าสวิตช์ปิดวงจรเอสซีอาร์จะหยุดนำกระแส เนื่องจากได้รับไบอัสกลับตลอดเวลาที่สวิตช์ยังคงปิดอยู่ โดยระยะเวลาในการบังคับให้เอสซีอาร์หยุดนำกระแสโดยให้ไบอัสกลับนี้จะต้องนานกว่าระยะเวลา Turn Off Time ซึ่งระบุไว้ในคู่มือ โดยทั่วไปค่าเวลานี้จะน้อยมาก (ประมาณไมโครวินาที)



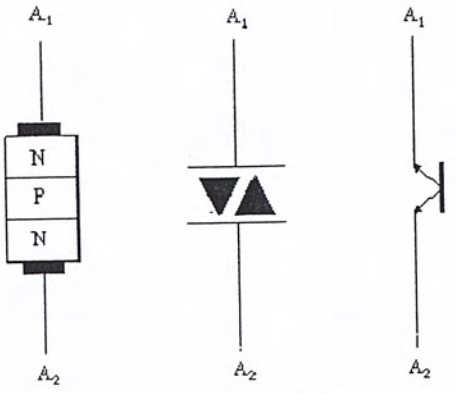
รูปที่ 2.7 การทำให้เอสซีอาร์หยุดนำกระแสโดยวิธี Forced Commutation [2]

2.3 ทฤษฎีพื้นฐานของไดแอก (Diac)

ไดแอก (Diac) หรือจะเรียกว่า ไดโอด-เอซี ก็ได้เป็นอุปกรณ์อิเล็กทรอนิกส์ จุดชนวนไตรแอกที่ถูกออกแบบให้มีการนำกระแสได้ 2 ทางที่แรงดันค่าหนึ่ง

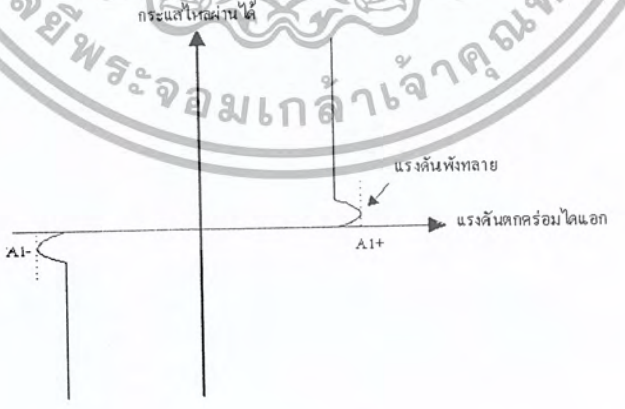
2.3.1 โครงสร้างและสัญลักษณ์ของไดแอก

รูปแบบโครงสร้างของไดแอกจะเป็นสาร P-N-P 3 ชั้น 2 รอยต่อเช่นเดียวกับทรานซิสเตอร์(Transistor) ดังแสดงในรูปที่ 2.8 แต่จะแตกต่างจากทรานซิสเตอร์ก็ตรงที่ ความเข้มในการโด๊ป (Dope) สารเป็นผลทำให้รอยต่อทั้งสองของไดแอกเหมือนกันจึงทำให้มีคุณสมบัติเป็นสวิตซ์ (Switch)ได้ 2 ทางและค่าแรงดัน (Voltage) ที่เริ่มต้นจะทำให้ไดแอกนำกระแสได้นั้นจะอยู่ในช่วง 29-30 โวลต์



รูปที่ 2.8 โครงสร้างและสัญลักษณ์ของไดแอก [3]

สำหรับการทำงานของไดแอกนั้นเมื่อพิจารณาจากรูปที่ 2.8 จะอาศัยช่วงของแรงดันพังทลาย (Break Over Voltage) เป็นส่วนของการทำงาน โดยป้อนแรงดันบวก (+) เข้าที่ขา A_1 และแรงดันลบ (-) เข้าที่ขา A_2 สังเกตุว่ารอยต่อN และ P ตรงบริเวณ A_1 จะอยู่ในลักษณะไบอัสกลับจึงไม่มีกระแสไหลจาก A_1 ไปยัง A_2 ได้ดังนั้นเมื่อเพิ่มแรงดันไบอัสดังกล่าวสูงขึ้นเรื่อยๆไปจนถึงค่าแรงดันค่าหนึ่งจะทำให้กระแสสามารถไหลทะลุผ่านรอยต่อ N-P มาได้ ส่วนรอยต่อตรง A_2 นั้นอยู่ในสภาวะไบอัสตรงอยู่แล้ว ดังนั้นกระแสที่ไหลผ่าน ไดแอกนี้จึงเหมือนกับเป็นกระแสที่เกิดจากการพังทลาย ของไดโอดและถ้าไม่มีการจำกัดกระแสแล้วไดแอกก็สามารถพังได้เหมือนกันถ้าหากสลับขั้วศักย์แรงดัน A_1 และ A_2 การทำงานของไดแอกก็จะเป็นเช่นเดียวกับกรณีดังกล่าวที่ผ่านมา สามารถเขียนเป็นกราฟแสดงความสัมพันธ์ของแรงดันตกคร่อมตัวไดแอก กับกระแสที่ไหลผ่านได้ดังแสดงดังรูปที่ 2.9



รูปที่ 2.9 กราฟแสดงลักษณะสมบัติของไดแอก [3]

เมื่อพิจารณาดูจากกราฟด้านบนเมื่อไดแอกนำกระแส ค่าแรงดันตกคร่อมตัวไดแอกจะลดค่าลงไปอีกเล็กน้อยซึ่งปกติจะลดลงจากค่าแรงดันพังทลายประมาณ 5 โวลต์ จากลักษณะสมบัติของไดแอกเราจะเห็นได้ว่าไดแอกนั้นเหมาะสมที่จะนำไปใช้เป็นตัวป้อนกระแสจุดชนวนให้กับอุปกรณ์ไตรแอก เพราะนำกระแสได้ 2 ด้าน

2.3.2 ตัวอย่างระดับค่าแรงดัน(Voltage) ของไดแอกในเบอร์ต่างๆ

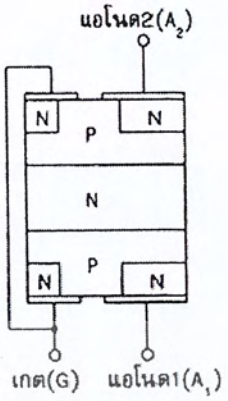
GT – 32 แถบสีแดง	V_{BO} = 27-37 V
GT – 35 แถบสีส้ม	V_{BO} = 30-40 V
GT – 40 แถบสีเหลือง	V_{BO} = 38-48 V
GT – 50 แถบสีเขียว	V_{BO} = 56-70 V

2.4 ทฤษฎีพื้นฐานของไตรแอก (Triac)

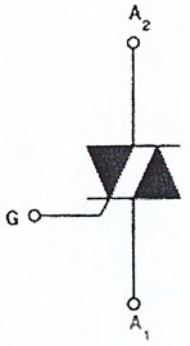
ไตรแอก (Triac) เป็นอุปกรณ์สารกึ่งตัวนำจำพวกไธริสเตอร์ เช่นเดียวกับ SCR แต่มีโครงสร้างแตกต่างไปจาก SCR เพราะไตรแอกนั้นเปรียบเสมือนการรวม SCR สองตัวไว้ด้วยกัน

2.4.1 โครงสร้าง สัญลักษณ์และวงจรสมมูลของไตรแอก

โครงสร้างของไตรแอกประกอบด้วยสารกึ่งตัวนำตอนใหญ่ 3 ตอน คือ PNP และในสารกึ่งตัวนำตอนใหญ่ มีสารกึ่งตัวนำตอนย่อยชนิด N อีก 4 ตอน ต่ออยู่กับสารกึ่งตัวนำชนิด P ทั้ง 2 ด้าน มีขาต่อออกมาใช้งาน 3 ขา ดังอธิบายได้ในรูปที่ 2.10 คือ



(ก) โครงสร้าง



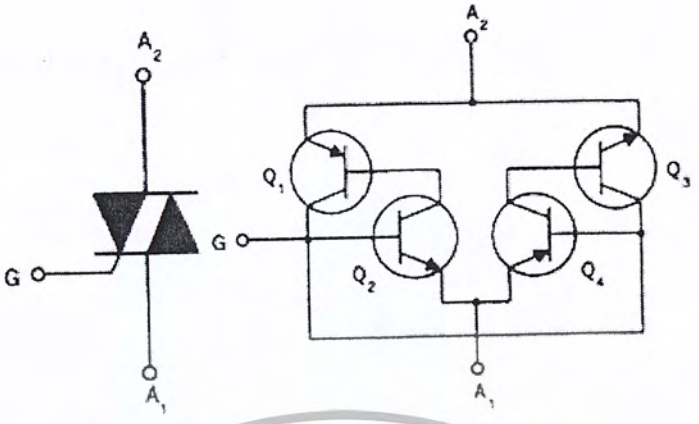
(ข) สัญลักษณ์

รูปที่ 2.10 (ก)โครงสร้างของไตรแอก (ข) สัญลักษณ์ของไตรแอก [4]

จากรูปที่ 2.10 (ก) ขาแอโนด 1 (Anode 1) ใช้ตัวย่อ A_1 หรืออาจเรียกว่าขาเมน เทอร์มินอล 1 (Main Terminal -1) ใช้ตัวย่อ MT_1 ขาแอโนด 2 (Anode 2) ใช้ตัวย่อ A_2 หรืออาจเรียกว่าขาเมน เทอร์มินอล 2 (Main Terminal -2) ใช้ตัวย่อ MT_2 และ ขาเกต (Gate) ใช้ตัวย่อ G สำหรับโครงสร้างและสัญลักษณ์ของไตรแอกพิจารณาได้ดังนี้

2.4.1.1 จากรูปที่ 2.10 (ก) อธิบายโครงสร้างเบื้องต้นของไตรแอก ประกอบ ด้วยสารกึ่งตัวนำตอนใหญ่ 3 ตอน PNP และสารกึ่งตัวนำตอนย่อยชนิด N อีก 4 ตอน ต่อกับสาร P ด้านละ 2 ตอน การต่อขาออกมาใช้งานทั้ง 3 ขา คือขา A_1 ,A_1 และ G ถูกต่อร่วมกับสารกึ่งตัวนำทั้งชนิด P และชนิด N ทำให้ไตรแอกสามารถนำกระแสได้กับแรงดันไฟสลับทั้งช่วงบวกและช่วงลบ

2.4.1.2 ส่วนรูปที่ 2.10 (ข) เป็นสัญลักษณ์ของไตรแอก ประกอบด้วยรูปสามเหลี่ยมและขีด 2 รูป ต่อกลับด้านกัน เหมือนกับ SCR สองตัวต่อขนานแบบหันหัวกลับทางกัน ขาเกตของ SCR ทั้งสองต่อร่วมกัน ดังนั้นวงจรสมมูลของไตรแอก จึงเหมือนกับวงจรสมมูลของ SCR สองตัวต่อร่วมกัน ลักษณะวงจรแสดงได้ดังรูปที่ 2.11



(ก) สัญลักษณ์ (ข) วงจรสมมูล

รูปที่ 2.11 (ก) สัญลักษณ์ของไตรแอก (ข) วงจรสมมูลของไตรแอก [4]

คุณสมบัติในการทำงานของไตรแอกเหมือนกับการทำงานของ SCR สามารถนำไปทำงานได้ทั้งแรงดันไฟตรงและแรงดันไฟสลับ การทำงานกับแรงดันไฟสลับ ไตรแอกสามารถนำกระแสได้ทั้งแรงดันไฟสลับช่วงบวกและแรงดันไฟสลับช่วงลบ โดยควบคุมการทำงานได้ด้วยขาเกต แรงดันที่มาควบคุมขาเกตให้ทำงาน หรือแรงดันกระตุ้นเกตใช้ได้ทั้งแรงดันบวกและแรงดันลบ

2.4.2 คุณสมบัติพื้นฐานของไตรแอกมีดังนี้

2.4.2.1 โดยปกติ ถ้าไม่มีสัญญาณทริกที่เกตไตรแอกจะไม่ทำงานโดยจะมีลักษณะเหมือนกับสวิตช์ที่ถูกเปิดวงจร

2.4.2.2 ถ้าในกรณีที่ MT2 และ MT1 ถูกป้อนด้วยแรงดันบวกและลบตามลำดับไตรแอกจะถูกกระตุ้นให้ทำงาน ได้โดยการป้อนสัญญาณพัลส์เพียงสั้น ๆ ที่เกตของมันโดยจะมีแรงดันตกคร่อมตัวมัน มีค่าประมาณ 1 หรือ 2 โวลต์ เท่านั้น และก็เช่นกันคือเมื่อไตรแอกเริ่มทำงานแล้ว ก็จะสามารถคงสภาพการทำงานอยู่ เช่นนั้นต่อไปเรื่อยๆ ตราบเท่าที่ยังมีกระแสไหลผ่านตัวมันอย่างต่อเนื่อง

2.4.2.3 หลังจากที่ไตรแอกคงสภาพการทำงานอยู่นั้น ทางเดียวที่จะหยุดการทำงานลงได้ก็โดยการลดปริมาณกระแสที่ไหลผ่านตัวมันลงให้มีค่าต่ำกว่ากระแสโฮลดิ้งของ

มัน ในกรณีที่ใช้ไตรแอกในการจ่ายกระแส AC การหยุดทำงานจะเกิดขึ้นอย่างอัตโนมัติ เมื่อแรงดันของไฟสลับเข้าใกล้จุดตัดศูนย์ที่เกิดขึ้น ทุกๆครึ่งคลื่น นั่นคือกระแสจะลดลงเป็นศูนย์

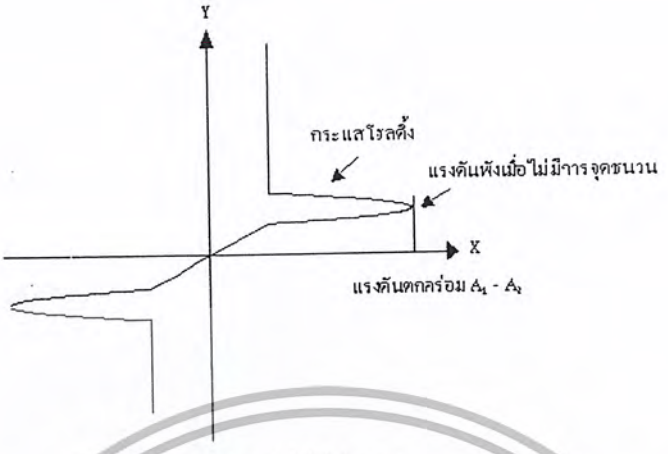
2.4.2.4 ไตรแอกถูกกระตุ้นให้ทำงานได้ ทั้งสัญญาณแบบบวกและลบที่ป้อนให้แก่ขาเกตโดยไม่คำนึงถึงขั้วที่ต่ออยู่ที่ MT1 และ MT2

2.4.2.5 ไตรแอกสามารถทนการกระชากของกระแสได้สูง เช่นโดยปกติสำหรับไตรแอกที่ทนกระแสปกติได้ 10 แอมแปร์ (rms) สามารถทนการกระชากของกระแสในช่วงหนึ่ง คาบเวลาของไฟ 60 เฮิรตซ์ได้สูงถึง 100 แอมแปร์ เป็นต้น

จากคุณสมบัติที่กล่าวมาในเรื่องของการนำกระแสนั้น จึงสามารถแบ่งการทำงานของไตรแอก ออกเป็น 4 แบบหรือ 4 ควอทเดรนท์ (Quadrant) แสดงการทำงานของไตรแอกทั้ง 4 ควอทเดรนท์ได้ดังรูปที่ 2.12

รูปที่ 2.12 การทำงานของไตรแอกทั้ง 4 ควอทเดรนท์ [4]

และสามารถอธิบายลักษณะกราฟแสดงสมบัติของไตรแอกได้ดังรูปที่ 2.13



รูปที่ 2.13 กราฟแสดงสมบัติของไตรแอก [4]

พิจารณาจากรูปที่ 2.13 กราฟแสดงสมบัติของไตรแอกเป็นการแสดงความสัมพันธ์ของกระกระแสที่ไหลระหว่าง A_2-A_1 และแรงดันตกคร่อมทั้งบวกและลบ ในขณะที่แรงดันคร่อม A_2-A_1 มีค่าเป็นบวกเมื่อเทียบกับ A_1 และถ้ายังไม่มีการจุดชนวน (Trigger) แล้ว จะมีค่าแรงดันระหว่าง A_2-A_1 ค่าๆหนึ่งที่ทำให้ไตรแอกนำกระแสเองได้ แรงดันนี้คือแรงดันพังทลายเหมือนกับ SCR แต่ถ้าให้แรงดัน A_2-A_1 นี้มีค่าน้อยกว่าแรงดันพังทลายแล้วทำการจุดชนวน ที่ขาเกต (G) ซึ่งกระแสเกตจะมีค่าเป็นบวกหรือลบก็ได้ ไตรแอกจะนำกระแสทันที กราฟความสัมพันธ์และข้อจำกัดต่างๆ จะเหมือนกับ SCR ในทำนองเดียวกันถ้าให้แรงดันที่ A_1 มีค่าเป็นบวกเมื่อเทียบกับ A_2 ส่วนของกราฟคือแกน X ทางด้านลบจะมีลักษณะคล้ายกันกับด้านบวก ถ้าเพิ่มแรงดันมากขึ้นจนถึงค่าแรงดันพังทลายก็จะทำให้ไตรแอกนำกระแสเองได้ และถ้าหากว่าไม่มีการจำกัดกระแสในตัวไตรแอกแล้ว ไตรแอกจะเกิดการเสียหายได้ โดยในขณะที่ไตรแอกนำกระแสถ้าลดค่ากระแสแอโนดลงจนถึงค่ากระแสต่ำสุดที่ยังคงทำให้ไตรแอกนำกระแสได้ ค่ากระแสต่ำสุดนี้เรียกว่า "โฮลดิ้ง" (I_H : Holding Current)ก็จะทำให้ไตรแอกหยุดนำกระแส

เนื่องจากไตรแอกสามารถนำกระแสไฟฟ้าได้ทั้งสองทาง จึงเหมาะกับการนำไปใช้กับไฟสลับมากกว่าเอสซีอาร์ (SCR) และสำหรับกระแสไฟสลับ (เป็นคลื่นรูปไซน์) จะมีอยู่ช่วงเวลาหนึ่งช่วงกระแสตัดกับเส้นศูนย์ของกราฟ ที่กระแสตกต่ำกว่ากระแสโฮลดิ้ง ดังนั้นจึงทำให้ไตรแอกหยุดนำกระแสเองและจะรอการจุดชนวนใหม่อีกครั้งและถ้าหากครึ่งลบของสัญญาณไฟสลับเข้า

มาก็จะนำกระแสทางด้านลบอีกเช่นเคย และจะหยุดนำกระแสเมื่อค่ากระแสลดลงต่ำกว่ากระแสโฮลดิ้ง

ตาราง 2.1 คุณสมบัติที่สำคัญของไตรแอกที่นิยมใช้

เบอร์	PIV	กระแสอาร์เอ็มเอส	V_{GT} (สูงสุด)	I_{GT} (สูงสุด)	I_H (สูงสุด)
C206D	400 V	3 A	2 V	5 mA	30 mA
2N6073	400 V	4 A	2.5 V	30 mA	70 mA
C226D	400 V	8 A	2.5 V	50 mA	60 mA
SC146D	400 V	10 A	2.5 V	50 mA	75 mA
TIC246D	400 V	15 A	2.5 V	50 mA	50 mA

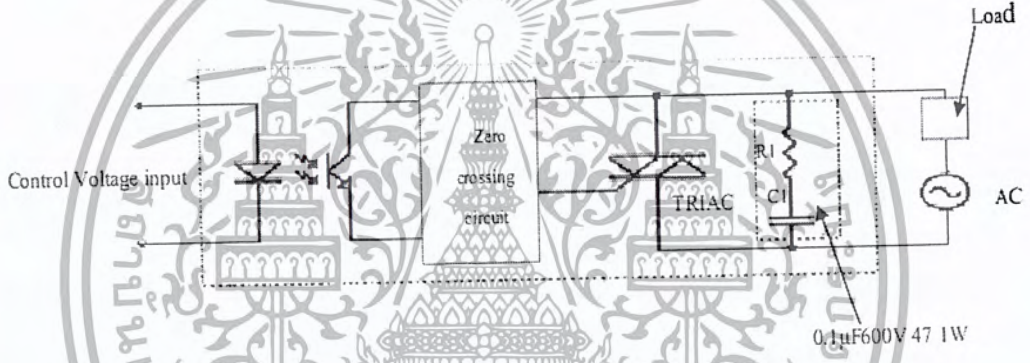
จากตารางที่ 2.1 คุณสมบัติที่สำคัญของไตรแอกที่นิยมใช้ แสดงคุณสมบัติที่สำคัญเบื้องต้นเช่น ค่า PIV , V_{GT} (สูงสุด), I_{GT} (สูงสุด), I_H (สูงสุด) ซึ่งเป็นค่าสำคัญในการพิจารณาเลือกใช้ไตรแอกให้เหมาะสมกับการใช้งาน

2.5 ทฤษฎีพื้นฐานของโซลิดสเตทรีเลย์ (Solid State Relay: SSR)

โซลิดสเตทรีเลย์ (Solid State Relay : SSR) ทำหน้าที่เป็นเหมือนสวิตช์อิเล็กทรอนิกส์ เป็นรีเลย์ประเภทหนึ่งซึ่งทำหน้าที่เหมือนรีเลย์กลไกแบบธรรมดาซึ่งภายในจะประกอบไปด้วย ขดลวด และหน้าสัมผัสเพียงแต่โซลิดสเตทรีเลย์จะมีโครงสร้างภายในเป็นอุปกรณ์อิเล็กทรอนิกส์สารกึ่งตัวนำที่ใช้สำหรับการตัดต่อวงจร แทนการตัดต่อวงจรด้วยหน้าสัมผัสเหมือนในรีเลย์แบบธรรมดาและยังใช้ในการปิด - เปิด ไฟสำหรับกระแสไฟ AC 110 โวลต์ หรือ DC 220 โวลต์ ได้ด้วย การใช้สารกึ่งตัวนำภายในทำให้โซลิดสเตทรีเลย์ไม่มีปัญหาด้านการสปาร์คของไฟ ทำงานได้รวดเร็ว และกำลังไฟที่ต้องการใช้ในการขับน้อย

2.5.1 หลักการทำงานของโซลิดสเตทรีเลย์

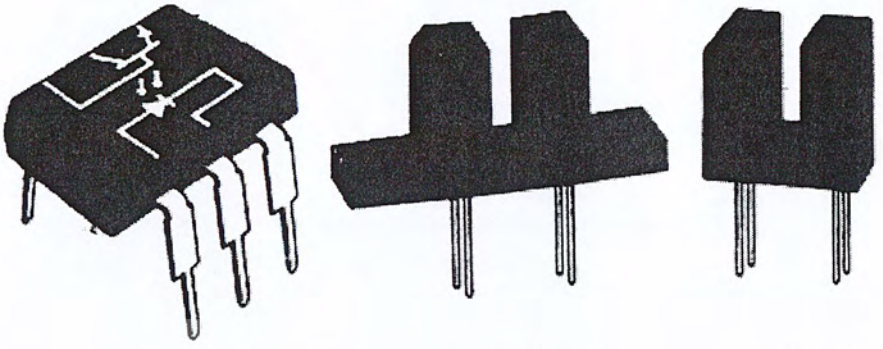
โครงสร้างการทำงานโดยทั่วไปของโซลิดสเตทรีเลย์ แสดงดังรูปที่ 2.14 ซึ่งจะเห็นว่าขั้วทางอินพุตจะทำหน้าที่รับสัญญาณควบคุมการเปิด และปิดของวงจรทางเอาต์พุต โดยอินพุตและเอาต์พุตจะแยกกันทางไฟฟ้าอย่างสิ้นเชิงทำให้ตัดปัญหาเรื่องกราวด์ลูปและการเชื่อมต่อทางไฟฟ้าที่อาจเกิดขึ้นได้ เพิ่มความปลอดภัยให้แก่วงจรและผู้ใช้งาน ซึ่งโดยทั่วไปการควบคุมเอาต์พุตจะควบคุมโดยการเชื่อมโยงทางแสงโดยจะใช้แสงเป็นตัวแปรในการส่งผ่านความสัมพันธ์ทั้งสองด้าน



รูปที่ 2.14 โครงสร้างการทำงานโดยทั่วไปของโซลิดสเตทรีเลย์ [5]

2.6 ทฤษฎีพื้นฐานของออพโตไอโซเลเตอร์ (Opto-Isolator)

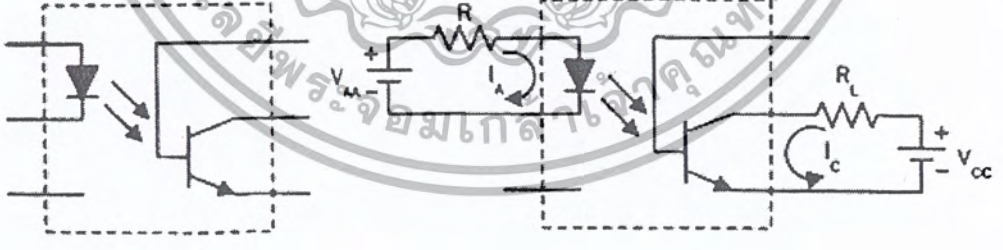
อุปกรณ์แยกกันทางแสงหรือออพโตไอโซเลเตอร์ (Opto-Isolator) เป็นอุปกรณ์อิเล็กทรอนิกส์ล้วน ที่ใช้การเชื่อมต่อกันทางแสงด้วยทิศทางการเคลื่อนที่ของแสงที่คงที่ภายในตัวอุปกรณ์ จะใช้หลักการเปลี่ยนสัญญาณไฟฟ้าเป็นสัญญาณแสง และเปลี่ยนกลับจากสัญญาณแสงเป็นสัญญาณไฟฟ้าตามเดิม สำหรับการเชื่อมต่อสัญญาณระหว่างวงจร 2 วงจรที่ต้องการแยกกันทางไฟฟ้าอย่างเด็ดขาด เพื่อป้องกันการรบกวนกันของสัญญาณไฟฟ้า รูปร่างและลักษณะของอุปกรณ์แยกกันทางแสง แสดงได้ดังรูปที่ 2.15



(ก) (ข) (ค)

รูปที่ 2.15 รูปร่างและลักษณะต่าง ๆ ของอุปกรณ์แยกกันทางแสงแบบ (ก) (ข) และ (ค) [6]

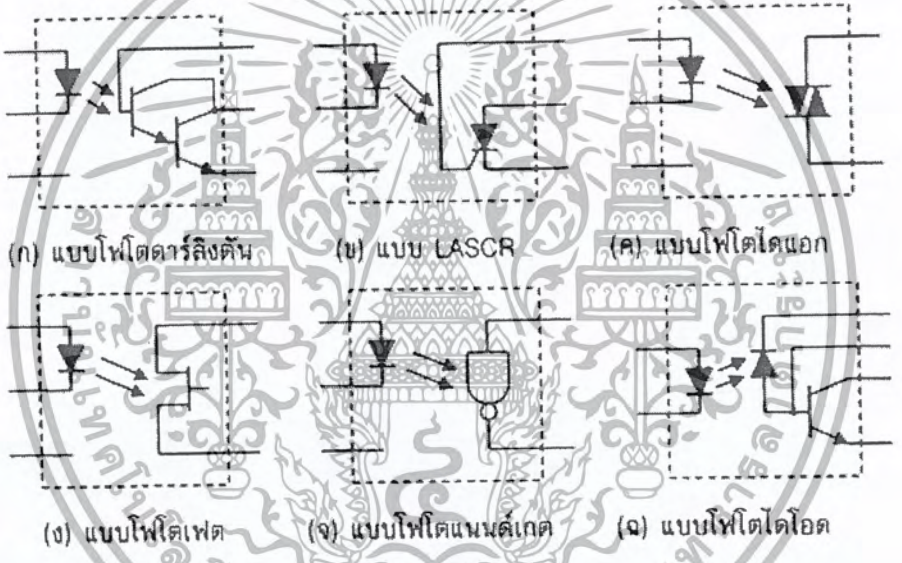
อุปกรณ์แยกกันทางแสงเบื้องต้น จะประกอบด้วยอุปกรณ์กำเนิดแสง ส่วนมากเป็นพวก LED กำเนิดแสงในย่านอินฟราเรด และอุปกรณ์รับแสงมักเป็นพวกโฟโตทรานซิสเตอร์ทั้ง LED โฟโตทรานซิสเตอร์มักถูกสร้างไว้ในตัวถังเดียวกันและต่อขาออกมาใช้งานแต่ละส่วน อาจสร้างขึ้นในรูปแบบของไอซี ดังรูปที่ 2.15 (ก) หรือสร้างแยกกันชัดเจนวางแนวขนานห่างกัน มีช่องว่างกั้นกลาง ดังรูปที่ 2.15 (ข) และ (ค) เพื่อให้อุปกรณ์บังแสง เช่น เฟืองหรือจานกลมแบนเจาะรูมาเคลื่อนที่ตัดผ่านลำแสงตรงบริเวณช่องว่าง สำหรับวงจรภายในอุปกรณ์แยกกันทางแสง จะมีวงจรภายในและวงจรใช้งานแสดงดังรูปที่ 2.16



(ก) วงจรภายใน (ข) วงจรใช้งาน

รูปที่ 2.16 วงจรภายในและวงจรใช้งานของอุปกรณ์แยกกันทางแสง [6]

จากรูปที่ 2.16 (ข) เมื่อจ่ายแรงดันไบอัสให้ LED และโฟโตทรานซิสเตอร์จะส่งผลให้ LED เปล่งแสงขึ้นส่องผ่านไปยังโฟโตทรานซิสเตอร์ ทำให้โฟโตทรานซิสเตอร์นำกระแส มีกระแส I_C ไหลในวงจรไปยังโหลด R_L แรงดันระหว่างอุปกรณ์ทั้งสองด้าน ทั้ง LED และด้านโฟโตทรานซิสเตอร์ ต้องใช้แรงดันมากหรือน้อยขึ้นอยู่กับระยะระหว่างตัวอุปกรณ์ทั้งสอง การจัดค่าแรงดันไบอัสที่เหมาะสมเป็นสิ่งสำคัญ และพอเหมาะกับค่าทนแรงดันของอุปกรณ์ด้วย นอกจากอุปกรณ์แยกกันทางแสงจะทำมาจาก LED และโฟโตทรานซิสเตอร์แล้ว ในส่วนของอุปกรณ์รับแสงยังสามารถผลิตขึ้นมาจากอุปกรณ์สารกึ่งตัวนำหลายชนิดเป็นอุปกรณ์แยกกันทางแสงหลายประเภทได้ แสดงดังรูปที่ 2.17



รูปที่ 2.17 อุปกรณ์แยกกันทางแสงแบบต่าง ๆ [6]

จากรูปที่ 2.17 เป็นอุปกรณ์แยกกันทางแสงหลายแบบ อุปกรณ์กำเนิดแสงใช้ LED อินฟราเรดเป็นหลัก ส่วนอุปกรณ์รับแสงสามารถผลิตขึ้นได้จากอุปกรณ์สารกึ่งตัวนำหลายชนิด เพื่อให้เกิดความเหมาะสมกับวงจรและอุปกรณ์ทางเอาต์พุตที่จะเชื่อมต่อไป

รูปที่ 2.17 (ก) อุปกรณ์รับแสงแบบโฟโตดาร์ลิงตัน ช่วยให้สามารถใช้งานได้กับวงจรที่ต้องการกระแสทำงานสูง แต่ความไวในการทำงานของวงจรลดลง

รูปที่ 2.17 (ข) อุปกรณ์รับแสงแบบ LASCR ช่วยการต่อเชื่อมวงจรทางอินพุตที่มีระดับแรงดันต่ำ ไปควบคุมวงจรทางเอาต์พุตที่มีระดับแรงดันสูง หรืออุปกรณ์เครื่องมือกลทางไฟฟ้าบางชนิด

รูปที่ 2.17 (ค) อุปกรณ์รับแสงแบบโฟโตไดแอก สร้างขึ้นมาเพื่อประยุกต์ใช้งานเป็นอุปกรณ์กระตุ้นขาเกตไตรแอก โดยที่วงจรอินพุตมีระดับแรงดันต่ำ

รูปที่ 2.17 (ง) อุปกรณ์รับแสงแบบโฟโตเฟต ใช้งานได้กับวงจรเอาต์พุตที่ต้องการอิมพีแดนซ์สูงในการต่อเชื่อม ที่ต้องการแรงดันในการทำงานมากกว่ากระแสในการทำงาน

รูปที่ 2.17 (จ) อุปกรณ์รับแสงแบบโฟโตแนนด์เกต เป็นพวกอุปกรณ์ดิจิตอลอิเล็กทรอนิกส์ใช้ในการต่อเชื่อมวงจรแอนาล็อกและวงจรดิจิทัลเข้าด้วยกัน

รูปที่ 2.17 (ฉ) อุปกรณ์รับแสงแบบโฟโตไดโอดต่อร่วมกับ NPN ทรานซิสเตอร์ สามารถใช้งานได้กับโหลดที่ต้องการกระแสสูงขึ้น พร้อมกับมีการขยายสัญญาณในตัวก่อนส่งต่อไปให้วงจรเอาต์พุตอุปกรณ์แยกกันทางแสงหรือออพโตไอโซเลเตอร์ ส่วนมากมักผลิตขึ้นมาในรูปของไอซี อาจมีอุปกรณ์แยกกันทางแสงชุดเดียวหรือหลายชุดก็ได้

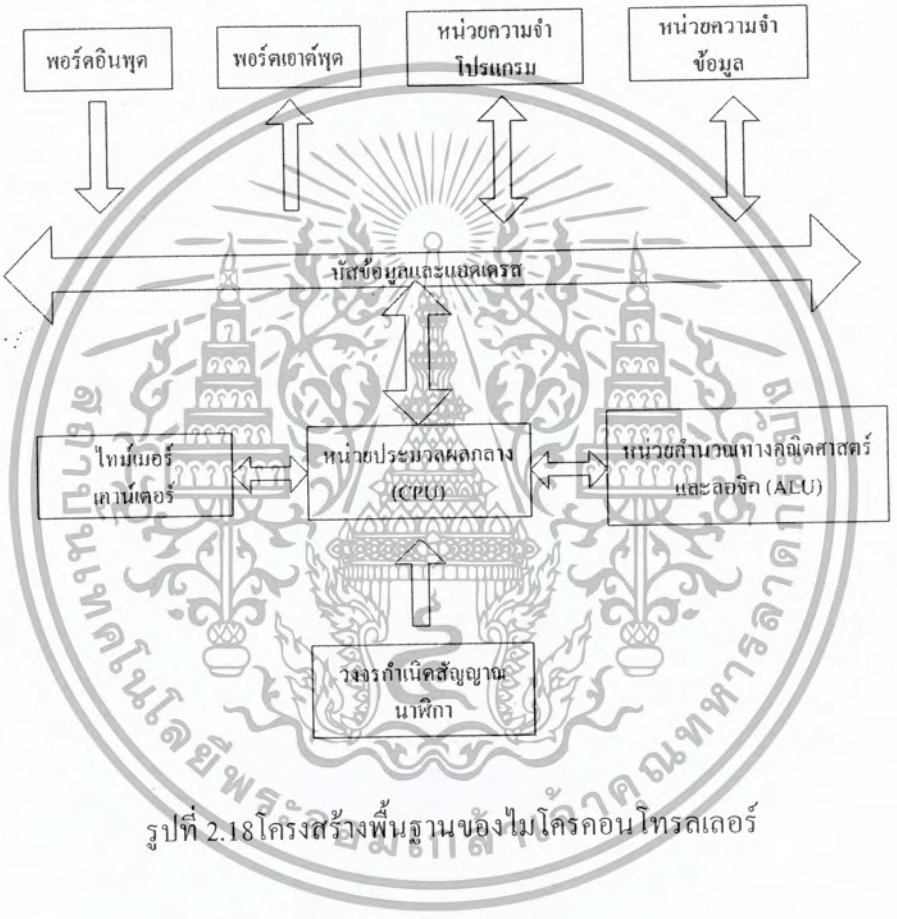
2.7 ทฤษฎีพื้นฐานของไมโครคอนโทรลเลอร์ (Microcontrollier)

2.7.1 ความหมายของไมโครคอนโทรลเลอร์

ไมโครคอนโทรลเลอร์ (Microcontroller) มาจากคำ 2 คำ คำหนึ่งคือ ไมโคร (Micro) หมายถึงขนาดเล็ก และคำว่าคอนโทรลเลอร์ (controller) หมายถึงตัวควบคุมหรืออุปกรณ์ควบคุมขนาดเล็ก แต่ในตัวอุปกรณ์ควบคุมขนาดเล็กนี้ ได้บรรจุความสามารถที่ทำงานเสมือนกับระบบคอมพิวเตอร์ที่คนโดยส่วนใหญ่คุ้นเคย นั่นคือ ภายในไมโครคอนโทรลเลอร์ได้รวมเอาซีพียู, หน่วยความจำ และพอร์ต ซึ่งเป็นส่วนประกอบหลักสำคัญของระบบคอมพิวเตอร์เข้าไว้ด้วยกัน โดยที่สามารถเขียนโปรแกรมเพื่อกำหนดรูปแบบการทำงานและควบคุมได้อย่างอิสระตามความต้องการ

การควบคุมไมโครคอนโทรลเลอร์นั้นกะทำโดยผ่านกระบวนการควบคุมโดยโปรแกรมที่เขียนขึ้นเพื่อบอกถึงการทำงานของไมโครคอนโทรลเลอร์ โดยลักษณะที่ถูกระบุขึ้น

โดยผู้เขียนโปรแกรมควบคุม ซึ่งควบคุมการทำงานทั้งหมดของไมโครคอนโทรลเลอร์ ในการกำหนดพอร์ตให้เป็นอินพุตหรือเอาต์พุต และยังสามารถกำหนดหน่วยความจำภายในซึ่งเป็นที่เก็บข้อมูลและเป็นที่พักข้อมูลตามความต้องการ โดยในการทำงานของไมโครคอนโทรลเลอร์แต่ละคำสั่ง จะอ้างอิงเวลาจากสัญญาณนาฬิกาที่ส่งให้กับไมโครคอนโทรลเลอร์เป็นหลัก สามารถแสดงโครงสร้างพื้นฐานของไมโครคอนโทรลเลอร์ได้ดังรูปที่ 2.18



รูปที่ 2.18โครงสร้างพื้นฐานของไมโครคอนโทรลเลอร์

2.7.2 ไมโครคอนโทรลเลอร์ในตระกูลต่างๆ

2.7.2.1 MCS-51 ไมโครคอนโทรลเลอร์ MCS-51 หรือ Intel 8051 เป็นไมโครคอนโทรลเลอร์ที่ถูกพัฒนาขึ้น โดยบริษัท Intel เมื่อปี ค.ศ. 1980 และได้รับความนิยมสูงสุดในขณะนั้นจนถึงปัจจุบัน ก็ยังคงมีใช้กันอยู่อย่างแพร่หลาย MCS-51 เป็นไมโครคอนโทรลเลอร์ที่มีสถาปัตยกรรมและฟังก์ชั่นการใช้งานไม่ค่อยซับซ้อนเท่าใดนักและยัง

เป็นไมโครคอนโทรลเลอร์ที่มีความเร็วในการทำงานต่ำ ทำให้ในปัจจุบันความนิยมใช้ MCS-51 จึงลดลง แต่อย่างไรก็ตาม MCS-51 ก็เหมาะที่จะเป็นไมโครคอนโทรลเลอร์ตัวแรกๆ ที่จะช่วยให้ศึกษาเรียนรู้พื้นฐานการพัฒนาโปรแกรมบนไมโครคอนโทรลเลอร์เป็นอย่างดี

2.7.2.2 PIC ไมโครคอนโทรลเลอร์ PIC เป็นไมโครคอนโทรลเลอร์ที่ถูกพัฒนาขึ้น โดยบริษัท Microchip คำว่า PIC ย่อมาจาก Peripheral Interface Controller ในปัจจุบัน PIC เป็นไมโครคอนโทรลเลอร์ ที่ได้รับความนิยมอย่างมากเพราะ PIC มีฟังก์ชั่นการใช้งานมากมาย มีความเร็วในการทำงานสูงและราคาก็ไม่สูงมากด้วย ดังนั้นจึงทำให้ไมโครคอนโทรลเลอร์ตระกูล PIC ที่ขายในท้องตลาด มีให้เลือกมากมายหลายรูปแบบแล้วแต่การใช้งาน จุดเด่นประการสำคัญประการหนึ่งของ PIC ก็คือสามารถโปรแกรม PIC ผ่านพอร์ตอนุกรมได้อีกด้วย

2.7.2.3 AVR ไมโครคอนโทรลเลอร์ AVR เป็นไมโครคอนโทรลเลอร์ที่ถูกพัฒนาขึ้น โดยบริษัท Atmel เมื่อปี ค.ศ. 1996 AVR เป็นไมโครคอนโทรลเลอร์ตัวแรกๆที่ใช้หน่วยความจำแฟลช เป็นหน่วยความจำโปรแกรมแทนหน่วยความจำ ROM หรือ EPROM แบบเดิม ไมโครคอนโทรลเลอร์ AVR แบ่งออกตามระดับความจุของข้อมูลและฟังก์ชั่นการใช้งาน ปัจจุบัน AVR เป็นไมโครคอนโทรลเลอร์ตระกูลหนึ่งที่ได้รับความนิยมไม่แพ้ PIC เพราะมีความเร็วในการใช้งานสูง มีฟังก์ชั่นใช้งานมากมาย รวมถึงยังโปรแกรมผ่านพอร์ตสื่อสารอนุกรมได้เช่นเดียวกับ PIC อีกทั้งในขณะโปรแกรมยังไม่ต้องถอดชิปตัวถังออกจากวงจรด้วย หรือที่เรียกกันว่า In-System Programmable

2.7.2.4 ARM เป็นไมโครคอนโทรลเลอร์ที่ถูกพัฒนาขึ้นโดยบริษัท ARM Limited คำว่า ARM ย่อมาจาก Advance RISC Machine ไมโครคอนโทรลเลอร์ ARM นิยมใช้เป็นอย่างมากในอุปกรณ์ประเภทระบบสมองกลฝังตัว (Embedded System) เช่น โทรศัพท์มือถือ, เครื่อง PDA, เครื่องคิดเลข เป็นต้น ARM เป็นไมโครคอนโทรลเลอร์ที่มีสถาปัตยกรรมและฟังก์ชั่นการใช้งานที่ซับซ้อน เหมาะสำหรับการใช้งานพัฒนาโปรแกรมไมโครคอนโทรลเลอร์ระดับสูง

สำหรับในปริญญานิพนธ์นี้จะเลือกใช้ไมโครคอนโทรลเลอร์ตระกูล MCS-51 ในการศึกษาออกแบบและสร้างระบบไฟสถานบันเทิงแบบอัตโนมัติตามจังหวะเสียงดนตรี เพราะ MCS-51 มีโครงสร้างและวิธีการที่ไม่ซับซ้อนมากนัก เหมาะแก่การเริ่มต้นและศึกษาพื้นฐานของ

ไมโครคอนโทรลเลอร์ทำให้สามารถทำความเข้าใจได้ง่ายและรวดเร็วกว่าไมโครคอนโทรลเลอร์ตัวอื่นๆ

2.7.3 ภาษาที่ใช้ในการเขียนไมโครคอนโทรลเลอร์

2.7.3.1 ภาษาเครื่อง เป็นภาษาระดับต่ำที่สุด ประกอบไปด้วยรหัสเลขฐาน 2 คือ 0 กับ 1 เท่านั้น ซึ่งเป็นภาษาที่ไมโครคอนโทรลเลอร์เข้าใจ แต่มนุษย์จะทำความเข้าใจได้ยาก เพราะต้องอาศัยการจดจำรหัสคำสั่งต่างๆ รวมถึงต้องเข้าใจโครงสร้างภายในของไมโครคอนโทรลเลอร์ด้วย ดังนั้นจึงได้มีการคิดค้นสิ่งที่เรียกว่า คอมไพล์เออร์ (Complier) ขึ้นมาเพื่อทำให้มนุษย์สามารถเขียนภาษาระดับสูงที่มนุษย์สามารถเข้าใจได้ โดยคอมไพล์เออร์จะทำหน้าที่เปลี่ยนภาษาระดับสูงเหล่านั้นให้กลายเป็นภาษาเครื่องเอง

2.7.3.2 ภาษาแอสเซมบลี เป็นภาษาที่ใช้รหัสคำสั่งเป็นตัวอักษรภาษาอังกฤษมาแทนคำสั่งรหัสเลขฐาน 2 ในแบบของภาษาเครื่องทำให้ภาษาแอสเซมบลีกลายเป็นภาษาที่มนุษย์ทำความเข้าใจได้ง่ายขึ้น นอกจากนั้นแล้ว ยังเป็นภาษาที่ทำให้โปรแกรมทำงานได้อย่างรวดเร็ว เพราะมีการสั่งงานไปที่ฮาร์ดแวร์โดยตรง ด้วยเหตุนี้ทำให้ผู้พัฒนาโปรแกรมจำเป็นจะต้องเข้าใจสถาปัตยกรรมภายในของไมโครคอนโทรลเลอร์อย่างละเอียดด้วย ซึ่งกลายเป็นข้อด้อยของภาษาแอสเซมบลีไป คอมไพล์เออร์ที่ทำหน้าที่แปลงภาษาแอสเซมบลีให้เป็นภาษาเครื่องเรียกว่า แอสเซมเบลอร์

2.7.3.3 ภาษา C เป็นภาษาระดับสูงที่มีความใกล้เคียงกับภาษามนุษย์ ทำให้ทำความเข้าใจได้ง่าย นอกจากนั้นแล้วการเขียนโปรแกรมภาษา C ก็ไม่ต้องจำเป็นเข้าใจในโครงสร้างภายในของไมโครคอนโทรลเลอร์อย่างละเอียด เพียงแต่เข้าใจการเขียนโปรแกรมแบบโครงสร้างก็เพียงพอแล้วภาษาCสามารถใช้ในการเข้าถึงโครงสร้างภายในของไมโครคอนโทรลเลอร์ได้โดยตรง ทำให้โปรแกรมที่เขียนขึ้นทำงานได้รวดเร็ว ดังนั้นภาษา C จึงเป็นที่นิยมแพร่หลายในการเขียนโปรแกรมไมโครคอนโทรลเลอร์ คอมไพล์เออร์ที่ใช้ในการแปลงภาษา C เป็นภาษาเครื่องมีอยู่มากมาย แต่ในปริญญานิพนธ์นี้เลือกที่จะใช้ Keil C51 (uVision 2) ซึ่งจะอธิบายและกล่าวถึงในหัวข้อถัดไป

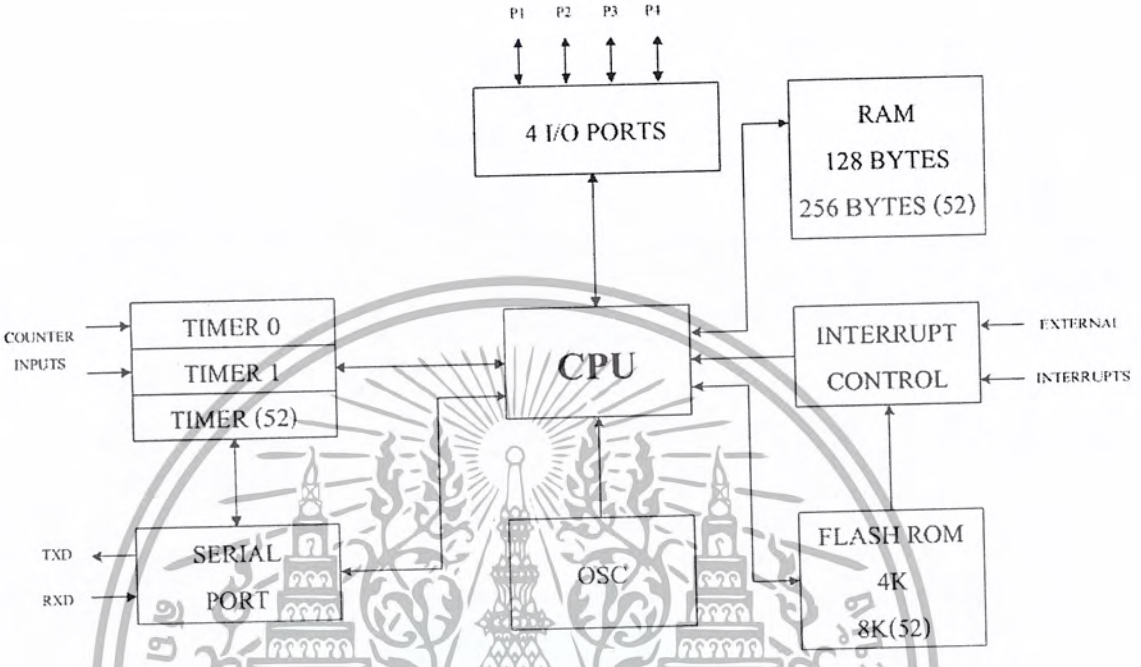
2.7.4 กลุ่มของไมโครคอนโทรลเลอร์ตระกูล MCS-51

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 ส่วนใหญ่นั้นจะมี CPU ขนาด 8 บิต โดยแต่ละเบอร์ก็จะมีความจุหน่วยความจำ จำนวนพอร์ตอินพุต-เอาต์พุต จำนวนขาที่แตกต่างกันไปตามลักษณะการใช้งานแต่ละเบอร์ ยกตัวอย่างเช่น เบอร์ AT89C2051 มีขนาดของหน่วยความจำแบบแฟลช 2 Kbyte หน่วยความจุข้อมูล 128 bytes แต่มีจำนวน 40 ขา ส่วนเบอร์ AT89C51 มีขนาดหน่วยความจำแบบแฟลช 4 Kbyte หน่วยความจำข้อมูล 128 bytes แต่มีจำนวน 40 ขา ซึ่งทำให้มีจำนวนพอร์ตอินพุต-เอาต์พุต มากกว่าเบอร์ AT89C2051 ดังนั้นจึงสามารถติดต่อกับอุปกรณ์ภายนอกได้หลายอุปกรณ์มากกว่า แต่ราคาก็สูงตามไปด้วยเช่นกัน แต่อย่างไรก็ตามไม่ว่าไอซีจะมีจำนวน 20 ขาหรือ 40 ขา ก็จะมี พอร์ต 1 และ พอร์ต 3 มาให้เป็นหลักเสมอ ทำให้สามารถใช้งานฟังก์ชั่นพิเศษได้ครอบคลุมได้เช่นเดียวกันสามารถพิจารณาได้จากคุณสมบัติของไมโครคอนโทรลเลอร์ตระกูล MCS-51 ได้ดังตารางที่ 2.2

ตารางที่ 2.2 คุณสมบัติของไมโครคอนโทรลเลอร์ตระกูล MCS-51

เบอร์	หน่วยความจำข้อมูล	หน่วยความจำโปรแกรม	จำนวน I/O	ความเร็ว	จำนวนขา
AT89C2051	128 Ram	2 K flash	15	12MHz	20
AT89C4051	128 Ram	4 K flash	15	24MHz	20
AT89C51	128 Ram	4 K flash	32	24MHz	40
AT89C52	256 Ram	8 K flash	32	24MHz	40
AT89C55WD	256 Ram	20 K flash	32	24MHz	40
AT89C51RC	512 Ram	32 K flash	32	24MHz	40
AT89C51ED2	2 K Ram	64 K flash	34	60MHz	40

2.7.5 โครงสร้างและสถาปัตยกรรมของไมโครคอนโทรลเลอร์เบอร์ AT89C51



รูปที่ 2.19 โครงสร้างภายในของไมโครคอนโทรลเลอร์เบอร์ AT89C51

2.7.5.1 คุณสมบัติของไมโครคอนโทรลเลอร์เบอร์ AT89C51

1) เป็นไมโครคอนโทรลเลอร์ที่มี CPU ขนาด 8 บิต

2) มีหน่วยความจำข้อมูล (RAM) ขนาด 128 byte

3) มีหน่วยความจำโปรแกรม (ROM) แบบแฟลชขนาด 4 Kbyte สามารถทนต่อการเขียนลบได้ 1000 ครั้งและคงค่าข้อมูลที่เขียนไว้ได้ 10 ปี

4) มีพอร์ตอินพุต-เอาต์พุตทั้งหมด 4 พอร์ต พอร์ตละ 8 บิต รวม 32 บิต โดยแต่ละบิตสามารถใช้ได้ทั้งเป็น อินพุตและเอาต์พุต

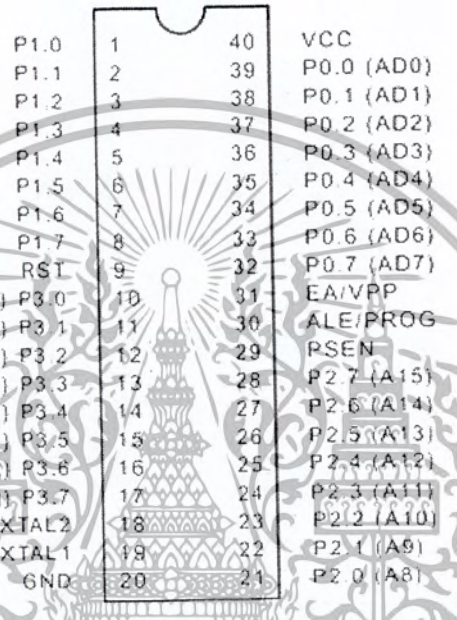
5) มีไทม์เมอร์/เคาน์เตอร์ขนาด 16 บิต 2 ตัวคือไทม์เมอร์ 0และไทม์เมอร์ 1

6) สามารถรองรับแหล่งกำเนิดอินเตอร์รัพต์ได้ 5 แหล่ง คือ อินเตอร์รัพภายนอกที่ขา INT0 และ INT1 อินเตอร์รัพต์จากไทม์เมอร์ 0 และไทม์เมอร์ 1 และอินเตอร์รัพต์จากพอร์ตสื่อสารอนุกรม

7) มีวงจรสื่อสารอนุกรมแบบสองทิศทาง (full duplex)

8) ทำงานที่สัญญาณนาฬิกา 0-24 MHz

2.7.5.2 การจัดขาของไมโครคอนโทรลเลอร์เบอร์ AT89C51 จะมีการจัดขาที่เหมือนกันแสดงลักษณะการจัดขาของไมโครคอนโทรลเลอร์ AT89C51 ในแบบตัวถังแบบ DIP 40 ขาแสดงดังรูปที่ 2.20



รูปที่ 2.20 การจัดขาของไมโครคอนโทรลเลอร์เบอร์ AT89C51 [7]

จากรูปที่ 2.20 หน้าที่และการใช้งานของแต่ละขามีดังนี้

1) **VCC** เป็นขาสำหรับต่อไฟเลี้ยง 5 โวลต์

2) **GND** เป็นขาสำหรับต่อลงกราวนด์

3) **XTAL 1/XTAL 2** เป็นขาทำหน้าที่ต่อกับตัวผลิตสัญญาณนาฬิกาให้กับไอซี ซึ่งโดยปกติจะอยู่ที่ 11.0592 MHz

4) **RST (Reset)** เป็นขาอินพุตเพื่อเริ่มต้นการทำงานใหม่ของไมโครคอนโทรลเลอร์ โดยการป้อนสัญญาณ ลอจิก 1 ให้โดยเป็นเวลาไม่น้อยกว่า 2 แมชชีนไซเคิล

5) **ALE/PROG (Address Latch Enable)** เป็นขาสัญญาณเอาต์พุตเพื่อแลตซ์ค่าแอดเดรสตำแหน่งข้อมูล (Address Bus, A0 - A7) ในการติดต่อกับหน่วยความจำภายนอก และเป็นขาสัญญาณเอาต์พุตเพื่อควบคุมการโปรแกรมให้กับไมโครคอน โทรลเลอร์

6) **PSEN (Program Store Enable)** เป็นขาสัญญาณสโตรป (พัลส์ต่ำ) เมื่อต้องการอ่านข้อมูลจากหน่วยความจำโปรแกรมภายนอก โดยการส่งสัญญาณสโตรปนี้ 2 ครั้งใน 1 พัลส์สัญญาณนาฬิกา

7) **EA/VPP (External Access)** เป็นขาสัญญาณอินพุตเพื่อทำหน้าที่เลือกใช้งานหน่วยความจำภายนอกและภายใน โดยการกำหนดสภาวะลอจิก ถ้าให้เป็นลอจิก 1 จะเลือกใช้งานหน่วยความจำภายใน ถ้าให้เป็น ลอจิก 0 จะเลือกใช้งานหน่วยความจำภายนอกในกรณีทำการโปรแกรมมโครคอนโทรลเลอร์ขานี้จะต่อเข้ากับแรงดันไฟฟ้า 12.75

8) **Port 0 (P0.0 – P0.7)** เป็นขาทำหน้าที่อินพุตและเอาต์พุตกับอุปกรณ์ภายนอก แบบ Open drain (ไม่มีตัวต้านทาน pull up ภายใน) ดังนั้นการใช้งานพอร์ต 0 นี้จึงจำเป็นต่อตัวต้านทาน pull up ด้วย นอกจากนั้น ยังทำหน้าที่เป็นขา Address Bus (A0 – A7) ในการติดต่อกับหน่วยความจำภายนอกและ Data Bus (D0-D7)ในการรับข้อมูลการโปรแกรมให้กับไมโครคอนโทรล เลอร์อีกด้วย

9) **Port 1 (P1.0 - P1.7)** เป็นขาทำหน้าที่อินพุตและเอาต์พุตกับอุปกรณ์ภายนอก แบบมีตัวต้านทาน pull up ภายใน

10) **Port 2 (P2.0 - P2.7)** เป็นขาทำหน้าที่อินพุตและเอาต์พุตกับปกรณ์ภายนอก แบบมีตัวต้านทาน pull up ภายใน และยังเป็นขา Address Bus (A8 - A15) ในการติดต่อกับหน่วยความจำภายนอกอีกด้วย

11) **Port 3 (P3.0 - P3.7)** เป็นขาทำหน้าที่อินพุตและเอาต์พุตกับอุปกรณ์ภายนอกแบบมีตัวต้านทาน pull up ภายใน และยังทำหน้าที่เป็นขาของฟังก์ชั่นพิเศษต่างๆ ดังต่อไปนี้

- P3.0 / RXD รับข้อมูลแบบอนุกรม
- P3.1 / TXD ส่งข้อมูลแบบอนุกรม
- P3.2 / INT0 อินเตอร์รัพภายนอกหมายเลข 0
- P3.3 / INT1 อินเตอร์รัพภายนอกหมายเลข 1
- P3.4 / T0 Timer / Counter ตัวที่ 1

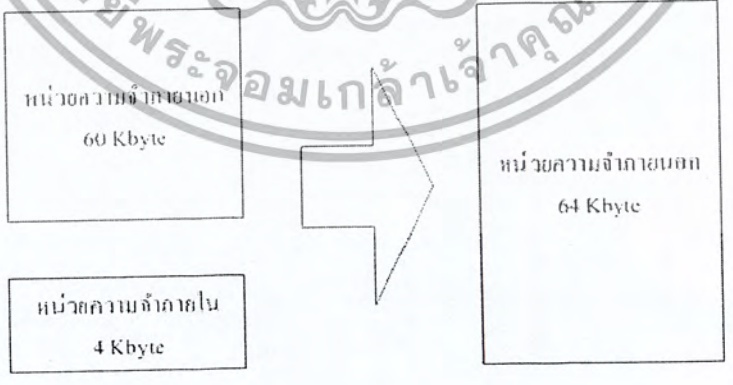
- P3.5 / T1 Timer / Counter ตัวที่ 2
- P3.6 / WR สัญญาณในการเขียนข้อมูลหน่วยความจำภายนอก
- P3.7 / RD สัญญาณในการอ่านข้อมูลหน่วยความจำภายนอก

2.7.6 โครงสร้างหน่วยความจำของไมโครคอนโทรลเลอร์ตระกูล MCS-51

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 นั้นมีหน่วยความจำอยู่ 2 แบบที่ใช้เป็นที่เก็บข้อมูล คือ หน่วยความจำโปรแกรม (ROM) และหน่วยความจำข้อมูล (RAM) ซึ่งมีรายละเอียดแต่ละหน่วยความจำดังต่อไปนี้

2.7.6.1 หน่วยความจำโปรแกรม (Program Memory)

มีพื้นที่ไว้เพื่อเก็บโปรแกรมที่ถูกเขียนขึ้นและไม่สามารถเขียนลงหน่วยความจำนี้ได้ขณะทำงาน แต่จะทำหน้าที่โดยการอ่านคำสั่งของโปรแกรม เพื่อไปควบคุมไมโครคอนโทรลเลอร์ให้ทำงาน ตามลักษณะที่ถูกออกแบบโดยผู้เขียนโปรแกรม ซึ่งโดยทั่วไปแล้วในตัวไมโครคอนโทรลเลอร์นั้น มีหน่วยความจำที่มีขนาดแตกต่างกันออกไป ยกตัวอย่างเช่น ไมโครคอนโทรลเลอร์เบอร์ AT89C51 มีขนาดหน่วยความจำโปรแกรม 4 Kbyte และสามารถขยายเพิ่มเติมด้วยหน่วยความจำภายนอกได้ถึง 64 Kbyte แสดงการจัดสรรหน่วยความจำโปรแกรม AT89C51 ได้ดังรูปที่ 2.21



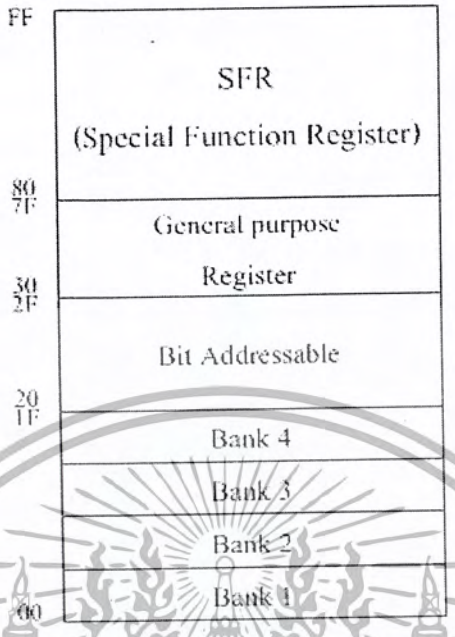
รูปที่ 2.21 การจัดสรรหน่วยความจำโปรแกรมไมโครคอนโทรลเลอร์ AT89C51

2.7.6.2 หน่วยความจำข้อมูล (Data Memory)

มีหน้าที่เก็บข้อมูลต่างๆ ในขณะที่โปรแกรมทำงานโดยทำการเรียกข้อมูลที่ต้องการของหน่วยความจำข้อมูลที่ต้องการมาใช้ หรือการนำข้อมูลมาเก็บไว้ในหน่วยความจำข้อมูล ยกตัวอย่างเช่น ไมโครคอนโทรลเลอร์เบอร์ AT89C51 จัดแบ่งหน่วยความจำข้อมูลออกเป็น 2 ส่วนคือ หน่วยความจำข้อมูลภายใน และหน่วยความจำข้อมูลภายนอก

1) หน่วยความจำข้อมูลภายใน รับความจำข้อมูลภายในของไมโครคอนโทรลเลอร์เบอร์ AT89C51 แบ่งออกเป็น 4 ส่วน พิจารณาถึงโครงสร้างหน่วยความจำข้อมูลภายในของ AT89C51 ได้ดังรูปที่ 2.22 คือ

- ที่ตำแหน่ง 00H - 1FH จำนวน 32 Kbyte เป็นพื้นที่ของรีจิสเตอร์ R0 - R7 ซึ่งมีอยู่ด้วยกัน 4ชุดนั้นหมายความว่ามีจำนวนถึง 32 รีจิสเตอร์
- ที่ตำแหน่ง 20H - 2FH จำนวน 16 byte สามารถเข้าใช้งานลักษณะไบต์หรือบิตได้ คือ สามารถอ้างถึงตำแหน่งแบบบิตได้ โดยตรง
- ที่ตำแหน่ง 30H - 7FH หน่วยความจำข้อมูลทั่วไป โดยสามารถเข้าถึงข้อมูลได้ในลักษณะไบต์เท่านั้น
- ที่ตำแหน่ง 80H - FFH สามารถเข้าถึงโดยตรง เป็นหน่วยความจำที่ทำหน้าที่พิเศษ



รูปที่ 2.22 โครงสร้างหน่วยความจำข้อมูลภายในของ AT89C51

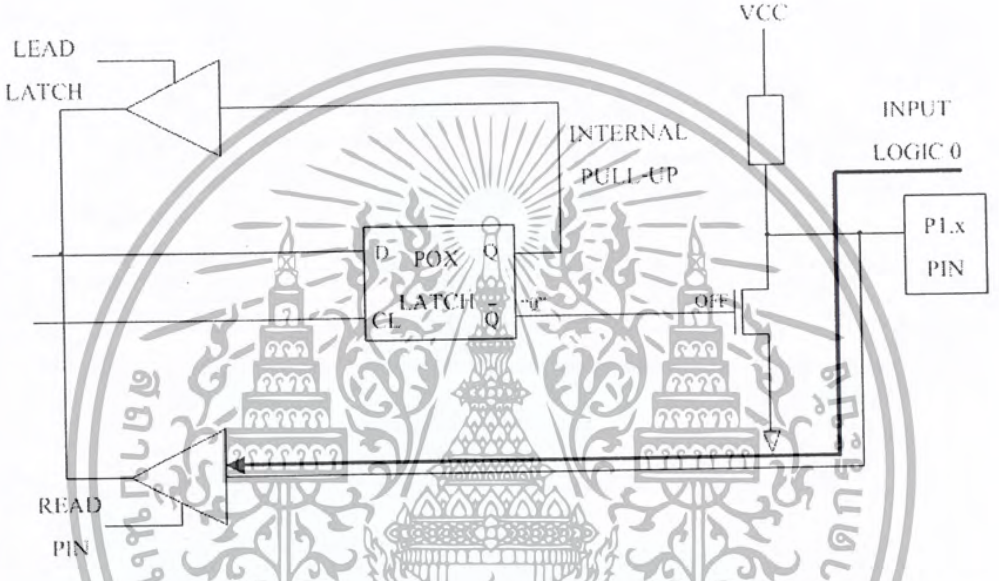
2) หน่วยความจำข้อมูลภายนอก หน่วยความจำข้อมูลภายนอกจะอยู่ภายนอกชิปไมโครคอนโทรลเลอร์เบอร์ AT89C51 ใช้สำหรับเก็บตัวแปรและข้อมูลทุกชนิดที่ไม่สามารถเก็บอยู่ภาย ในชิปได้ ซึ่ง AT89C51 สามารถอ้างหน่วยความจำข้อมูลภายนอกได้มากถึง 64 Kbyte หรือ 65,536 ตำแหน่ง

2.7.7 การใช้งาน MCS-51 เป็นพอร์ตอินพุตและเอาต์พุต

การใช้งานพอร์ตต่างๆ ของไมโครคอนโทรลเลอร์ MCS-51 เป็นพอร์ตที่สามารถติดต่อแบบสองทิศทางดังนั้นจึงต้องเข้าใจการกำหนดลักษณะการทำงานให้กับพอร์ตของไมโครคอนโทรลเลอร์ MCS-51 ดังต่อไปนี้

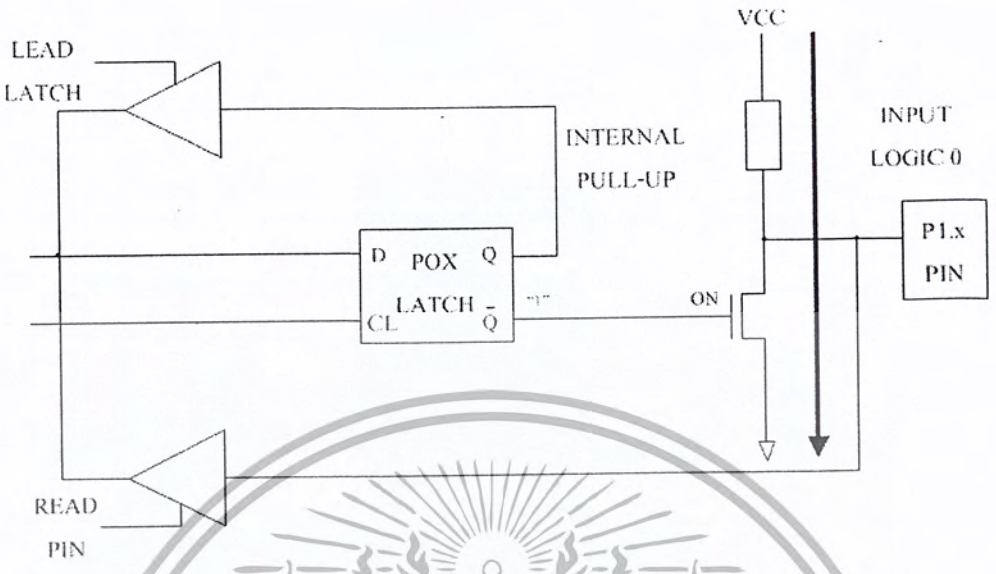
2.7.7.1 การกำหนดให้เป็นพอร์ตอินพุตการกำหนดลักษณะการทำงานเป็นพอร์ตพอร์ตอินพุตนั้น ต้องเริ่มจากการเขียนโปรแกรมเริ่มต้น โดยให้มีสภาวะลอจิก 1 เพื่อการทำงานของตัวเฟต ที่อยู่ภายใต้โครงสร้างขอพอร์ตจะหยุดทำงานลง ดังนั้น สัญญาณของพอร์ตถูก

ต่อเชื่อมกับวงจร R-Pull up ภายในโดยตรง ซึ่งมีค่าประมาณ 50 กิโลโอห์ม ส่งผลให้ขาพอร์ตมีสภาวะลอจิก 1 สามารถรับลอจิก 0 จากอุปกรณ์ภายนอกได้ง่าย แต่สำหรับ พอร์ต 0 ซึ่งไม่มี R-Pull up ทำให้เมื่อใช้งานควรต่อ R-Pull up เพื่อทำหน้าที่เป็น External Pull up ให้กับพอร์ต และการให้พอร์ตทำงานลักษณะอินพุตนั่นคือการรับสัญญาณไฟฟ้าเข้ามาที่พอร์ต ซึ่งก้อคือสัญญาณไฟฟ้า ลอจิก 0 นั่นเอง ดังรูปที่ 2.23

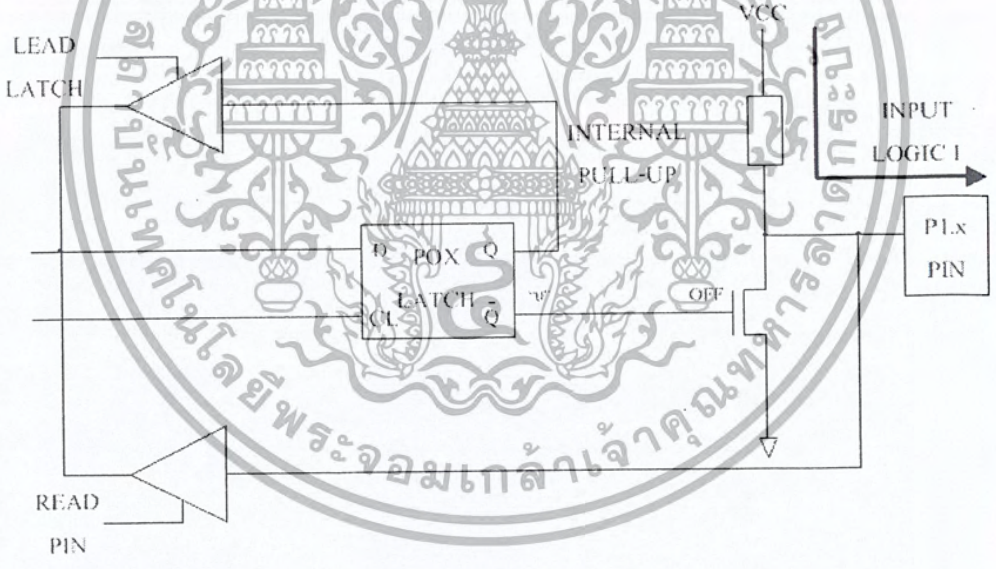


รูปที่ 2.23 แสดงการทำงานของพอร์ตลักษณะอินพุต

2.7.7.2 การกำหนดเป็นพอร์ตเอาต์พุต โดยปกติแล้วไมโครคอน โทรลเลอร์ MCS-51 จะกำหนดให้พอร์ตทำงานในลักษณะเอาต์พุตเป็นปกติอยู่แล้ว เช่น เมื่อกำหนดให้มีสภาวะลอจิก 0 ให้ปรากฏที่พอร์ตนั้นๆ ฟลิปฟลอปก็จะคงค่าลอจิก 0 ไว้และส่งลอจิก 1 ไปที่เฟต ทำให้เฟตทำงาน ส่งผลให้เอาต์พุตเป็นลอจิก 0 แต่ถ้าต้องการให้เอาต์พุตเป็นลอจิก 1 ก็กำหนดให้มีสภาวะเป็นลอจิก 1 ให้ปรากฏที่พอร์ตนั้นๆ ฟลิปฟลอปก็จะส่งลอจิก 0 ไปขับเฟต เฟตจะหยุดการทำงาน มีผลทำให้มีลอจิก 1 ตามตัวต้านทาน R-Pull up แต่ในลักษณะของพอร์ต 0 ซึ่งไม่มี R-Pull up ดังนั้น เมื่อต้องการให้เอาต์พุตมีสภาวะเป็นลอจิก 1 จึงจำเป็นต้องต่อ R-Pull up ให้พอร์ต 0 เสมอ สามารถพิจารณาลักษณะการทำงานเป็นพอร์ตเอาต์พุตเมื่อกำหนดให้สภาวะลอจิกเป็น 0 ได้ดังรูปที่ 2.24 และเมื่อกำหนดให้สภาวะลอจิกเป็น 1 ได้ดังรูปที่ 2.25



รูปที่ 2.24 ลักษณะการทำงานเป็นพอร์ตเอาต์พุต เมื่อกำหนดให้สภาวะลอจิกเป็น 0



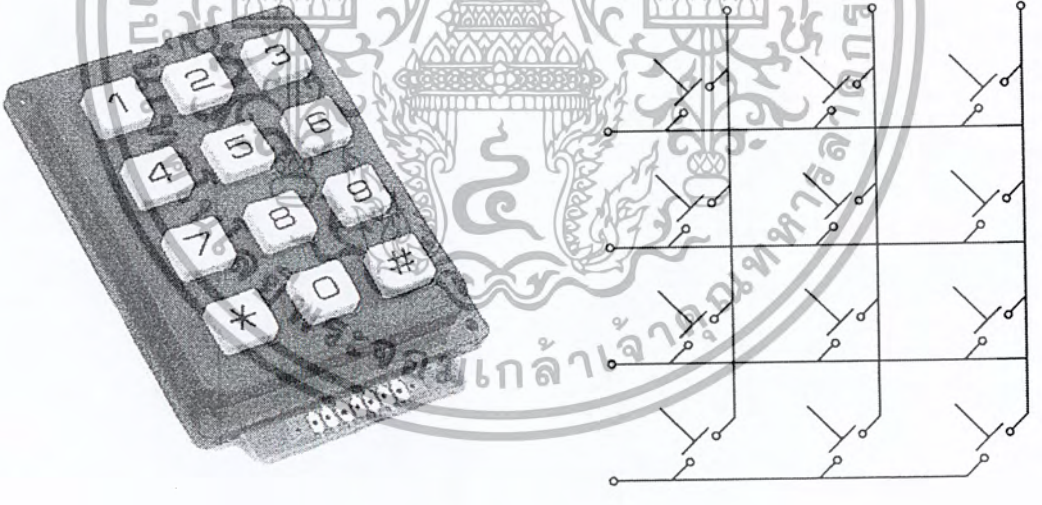
รูปที่ 2.25 ลักษณะการทำงานเป็นพอร์ตเอาต์พุต เมื่อกำหนดให้สภาวะลอจิกเป็น 1

ในการทำงานของลักษณะเป็นพอร์ตเอาต์พุตนั้น มีกระแสสูงสุดได้ 10 mA สำหรับบิตเดียว แต่หากใช้รวมกันทั้งหมดของพอร์ต (P1.0 - P1.7) จะมีกระแสจ่ายได้สูงถึง 26 mA แต่การทำงานของพอร์ต 0 เนื่องจากไม่มี R-Pull up แต่อยู่ภายในจะทำให้ได้กระแส

15 mA สูงสุดของบิต แต่หากใช้พอร์ต 0 ทั้งหมด8 Bit จะได้กระแสรวมกันสูงสุด 71 mA ซึ่งกระแสทั้งหมดนี้เรียกว่า กระแสซอร์ช และที่สำคัญในการต่อใช้งานบนคอมพิวเตอร์นั้นไม่จำเป็นต้องต่ออุปกรณ์ที่ช่วยในการส่งสัญญาณหรือที่เรียกว่า บัฟเฟอร์ แต่หากการต่อใช้งานจริงแล้ว ควรจะมีบัฟเฟอร์ในการต่อใช้งานอย่างยิ่ง เพื่อช่วยให้อุปกรณ์ที่ต่อร่วมกับไมโครคอนโทรลเลอร์นั้นมีกำลังในการขับที่ดีและคงที่พร้อมทั้งยังช่วยให้ตัวอุปกรณ์ไม่ต้องต่อโดยตรงกับไมโครคอนโทรลเลอร์ ในกรณีที่ตัวอุปกรณ์นั้นเกิดข้อบกพร่องอีกด้วย

2.8 หลักการทำงานและโครงสร้างของคีย์แพด

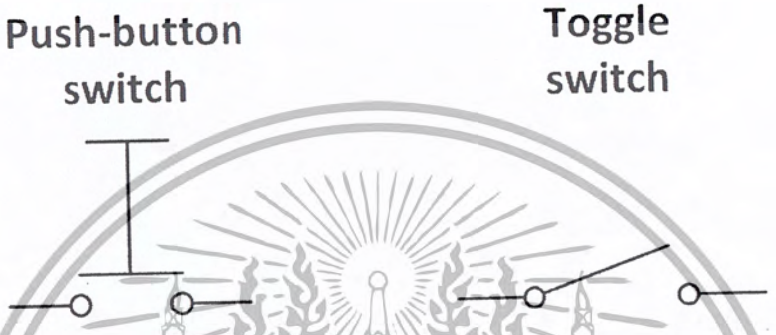
คีย์แพด (Keypad) คืออุปกรณ์ที่นำเอาสวิตซ์หลายๆตัวมาต่อกันแบบแมทริกซ์ ซึ่งจะช่วยลดจำนวนสายสัญญาณของข้อมูลให้น้อยลง ในขณะที่ใช้สวิตซ์จำนวนมากๆ แถมยังลดขนาดของวงจรให้มีขนาดเล็กลงด้วย คีย์แพดจึงเปรียบเสมือนคีย์บอร์ดเล็กๆนั่นเอง ลักษณะโครงสร้างของคีย์แพดจะมีลักษณะดังรูปที่ 2.26



รูปที่ 2.26 ลักษณะโครงสร้างของคีย์แพด [8]

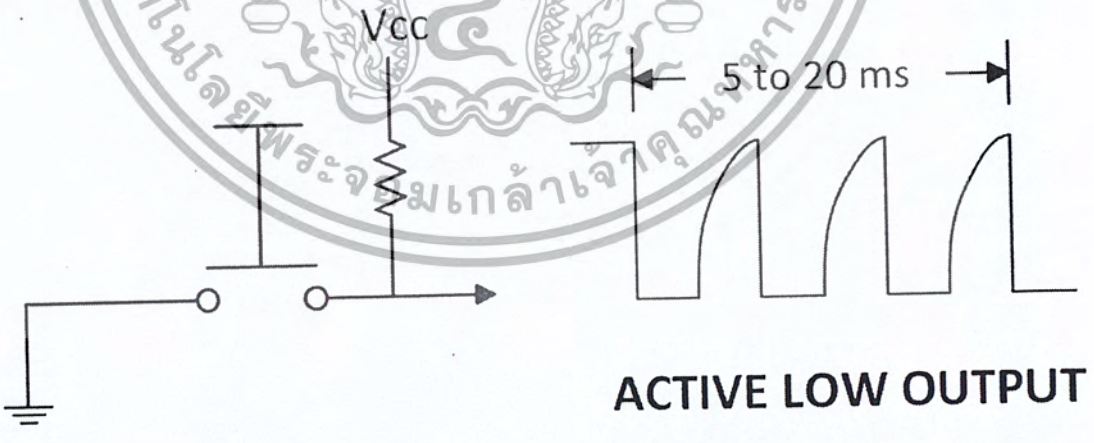
2.8.1 กลไกการทำงานของคีย์แพด

กลไกการทำงานของสวิตซ์เมื่อกดปุ่มสวิตซ์ จะทำให้เส้นลวดทองแดงสัมผัสกันเกิดการครบวงจรแสดงดังรูปที่ 2.27



รูปที่ 2.27 การทำงานของสวิตซ์

จากโครงสร้างการทำงานของสวิตซ์จะมีปัญหาที่เรียกว่า Contact Bounce หรือการกดสัมผัส นั่นคือเมื่อกดสวิตซ์ เอาต์พุตที่ได้จะเกิดเป็นพัลส์ขึ้นมาในช่วงแรกๆ ที่เป็นเช่นนี้เพราะว่าเวลากดสวิตซ์ หน้าเชื่อมต่อของสวิตซ์ไม่ได้เชื่อมต่อทันทีทันใด แสดงได้ดังรูปที่ 2.28



รูปที่ 2.28 การเกิด Contact Bounce หรือการกดสัมผัส

เพื่อให้ได้เอาต์พุตที่ถูกต้องจากการกดสวิตซ์ก่อนส่งไปยังไมโครคอนโทรลเลอร์ จะต้องหน่วงเวลาไว้ระยะหนึ่งก่อน เพื่อแน่ใจว่าสวิตซ์สัมผัสกันคงที่แล้ว กระบวนการนี้เรียกว่า การ “Debouncing” ดังนั้น ในการใช้งานคีย์แพดจึงประกอบไปด้วย 3 ขั้นตอนดังนี้

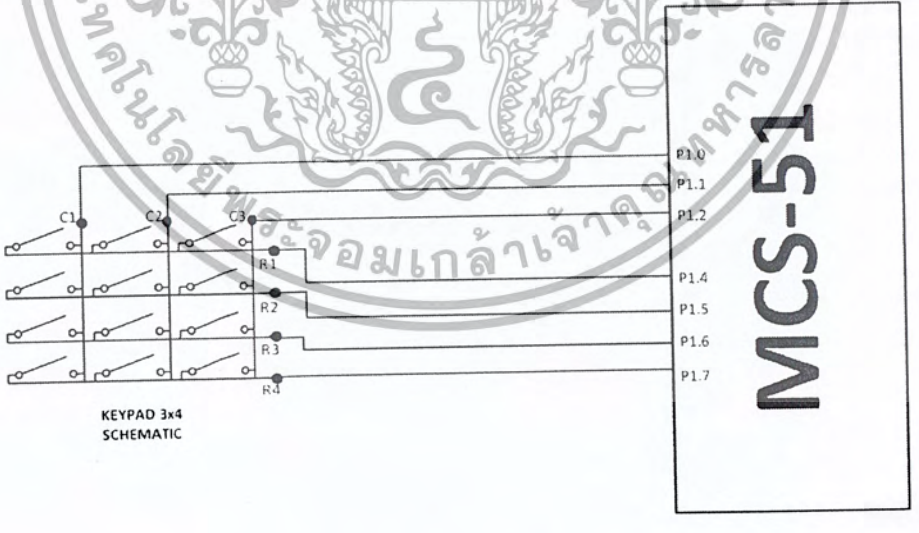
1) Keypad Scaning สำหรับตรวจสอบว่าสวิตซ์ไหนถูกกด

2) Keypad Debouncing เพื่อความแน่ใจว่าสวิตซ์นั้นถูกกดจริง

3) Table Lookup สำหรับหารหัสแอสกี้ (ASCII Code) ของคีย์ที่กดนั้น เพื่อส่งให้ไมโครคอนโทรลเลอร์หรือคอมพิวเตอร์ต่อไป

จากขั้นตอนการทำงานทั้ง 3 นี้สามารถไปประยุกต์ใช้กับคีย์บอร์ดที่เชื่อมต่อกับคอมพิวเตอร์ได้

2.8.2 การเชื่อมต่อคีย์แพดเข้ากับไมโครคอนโทรลเลอร์



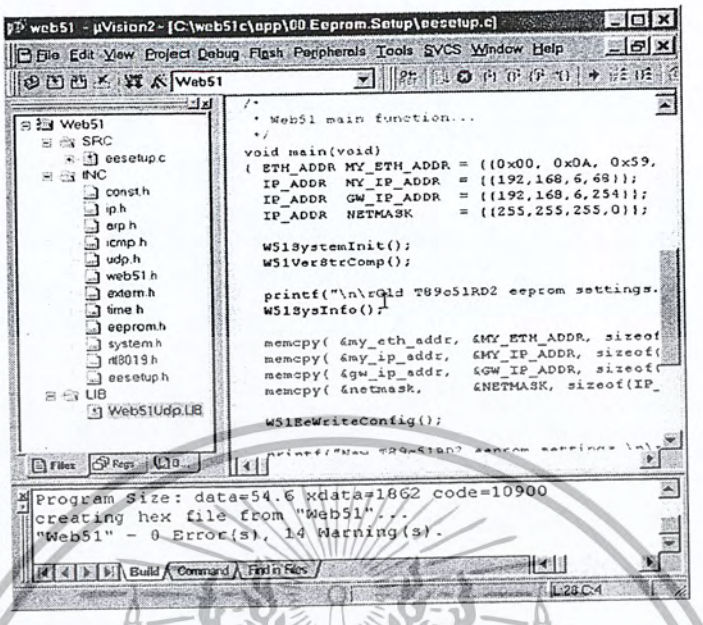
รูปที่ 2.29 การเชื่อมต่อคีย์แพดเข้ากับไมโครคอนโทรลเลอร์

จากรูปที่ 2.29 การเชื่อมต่อคีย์แพด ขนาด 3x4 เข้ากับไมโครคอนโทรลเลอร์ที่พอร์ต 1 โดยใช้สายทั้งหมด 7 เส้น เป็นสายของคอลัมน์ 3 เส้นและสายของโรล 4 เส้น โดยไมโครคอนโทรลเลอร์จะทำการส่งข้อมูลบิต “0” ไปยัง P1.0 - P1.2 ตามลำดับ ในทุกครั้งที่มีการส่งข้อมูลไปยังสายคอลัมน์คีย์แพด ไมโครคอนโทรลเลอร์จะทำการอ่านค่าที่ P1.4 - P1.7 เข้ามาด้วย หากไม่มีการกดคีย์ค่าของ P1.4 - P1.7 ก็จะเป็น “1” ทั้งหมด ถ้าหากกมีการกดคีย์ค่าของ P1.4 - P1.7 ก็จะไม่เป็น “1111” อีกต่อไป เป็นการแจ้งให้ทราบว่ามีการกดคีย์แพดแล้วจากนั้นไมโครคอนโทรลเลอร์ก็จะทำการค้นหาตำแหน่งต่อไป ในการค้นหาตำแหน่งนั้น สิ่งที่จะได้มาอย่างแรกคือ ค่าตำแหน่งของคีย์นั้น จากนั้นก็จะนำค่าตำแหน่งนั้นไปเปิดตารางข้อมูล (ตารางที่ 3.1) เพื่อที่จะได้ค่าที่ต้องการนำไปแสดงผลที่แท้จริง

2.9 โปรแกรม Keil C51 (uVision 2)

โปรแกรม Keil C51 (uVision 2) นั้นได้เข้ามามีบทบาทเป็นอย่างมากสำหรับการเขียนโปรแกรมต่างๆเพื่อลงไมโครคอนโทรลเลอร์ ซึ่งมีความสะดวกสบายอย่างมากเพราะการเขียนด้วยภาษา C นั้น เป็นภาษาที่เข้าใจได้ง่าย แก้ไขได้อย่างสะดวก จึงทำให้โปรแกรมนี้เป็นที่นิยมมากในผู้ใช้ไมโครคอนโทรลเลอร์ ตระกูล MCS-51

โดยโปรแกรม Keil C51 นั้นเป็นโปรแกรมที่สร้างขึ้นโดยบริษัท Keil Software โปรแกรมนี้จะเรียกกันในอีกชื่อหนึ่งว่า uVision 2 ซึ่งง่ายในการเขียนโปรแกรมภาษา C ซึ่งสามารถแปลงเป็น HEX ไฟล์ และมีไอซีไมโครคอนโทรลเลอร์ให้เลือกมากมายหลายแบบหน้าต่างการใช้งานโปรแกรมแสดงดังรูปที่ 2.30



รูปที่ 2.30 หน้าต่างการใช้งานโปรแกรม Keil C51 (uVision 2) [9]

2.10 โปรแกรม Visual Studio 2008

Visual C# 2008 เป็นเครื่องมือที่ช่วยในการเขียนภาษา C# ซึ่งภาษา C# เป็นภาษาโปรแกรมยุคใหม่ที่ถูกสร้างขึ้นมาสำหรับการพัฒนาซอฟต์แวร์ภายใต้เทคโนโลยี Microsoft .NET ซึ่งเป็นเทคโนโลยีที่ได้รับความนิยมสูงสุดในปัจจุบัน

Visual C# นั้นถูกแนะนำครั้งแรกพร้อมกับ Visual Studio .NET ในชื่อของ Visual C# .NET แต่มาโด่งดังและได้รับความนิยมอย่างสูงในเวอร์ชั่นถัดมานั่นคือ Visual C# 2005 ซึ่งมาพร้อมกับ Visual Studio 2005 และเวอร์ชั่นล่าสุดคือ Visual C# 2008 ซึ่งได้รับการพัฒนาให้รองรับการพัฒนาแอพพลิเคชันรูปแบบต่างๆได้มากมายทั้งที่เป็นแอพพลิเคชันบน Windows, อินเทอร์เน็ต, Mobile เป็นต้น

แต่เดิมนั้นเรามักจะเรียกการพัฒนาซอฟต์แวร์ (Software Development) ว่าการเขียนโปรแกรม (Programming) เพราะมีแค่งานเขียนโปรแกรม แต่สำหรับยุคนี้แล้วการพัฒนา

ซอฟต์แวร์มีมากกว่าการเขียนโปรแกรม โดยมีตั้งแต่การออกแบบหน้าตา, ออกแบบฐานข้อมูล, การเขียนโปรแกรม, การทดสอบโปรแกรม และการเผยแพร่ผลงานที่พัฒนาขึ้นมา นอกจากนี้ขอบเขตของการพัฒนาแอพพลิเคชันยังครอบคลุมไปถึงการสร้างแอพพลิเคชันให้ทำงานบนอินเทอร์เน็ต, สร้างซอฟต์แวร์สำหรับโทรศัพท์มือถือ หรือแม้กระทั่งเขียนเป็นชุดคำสั่งสำหรับหุ่นยนต์

รูปที่ 2.31 โปรแกรม Visual Studio 2008 [10]

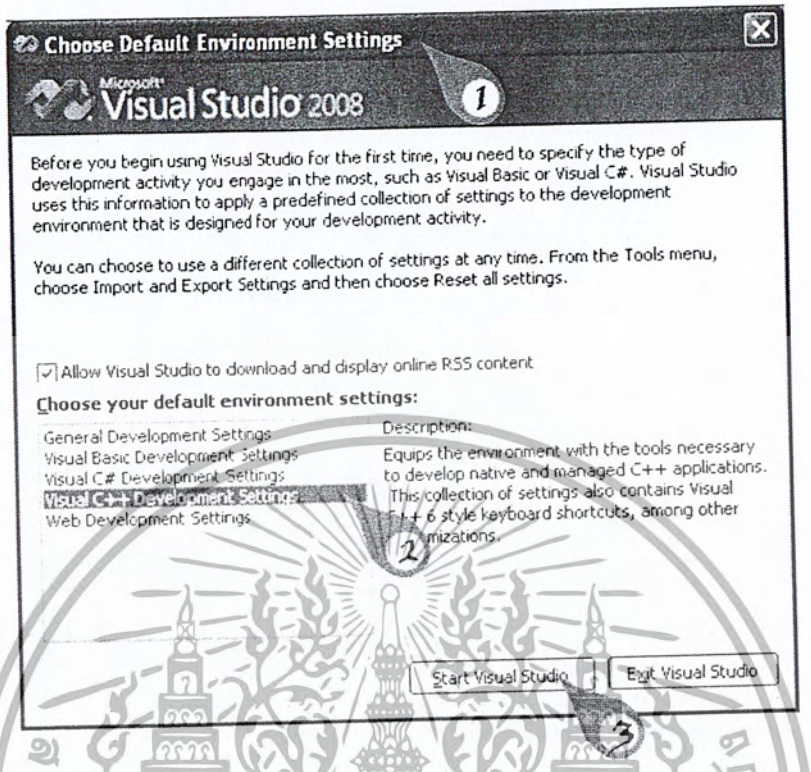
2.10.1 เริ่มต้นใช้งาน Visual C# 2008

เมื่อเราเริ่มต้นใช้งาน Visual Studio 2008 เป็นครั้งแรกจะมีขั้นตอนเพื่อเลือกใช้ Visual C# 2008 เป็นภาษาหลักสำหรับเขียนโปรแกรม ดังนี้

2.10.1.1 คลิกปุ่ม Start > All Programs > Microsoft Visual Studio 2008 > Microsoft Visual Studio 2008 จะปรากฏหน้าต่าง Choose Default Environment Setting ขึ้นมา

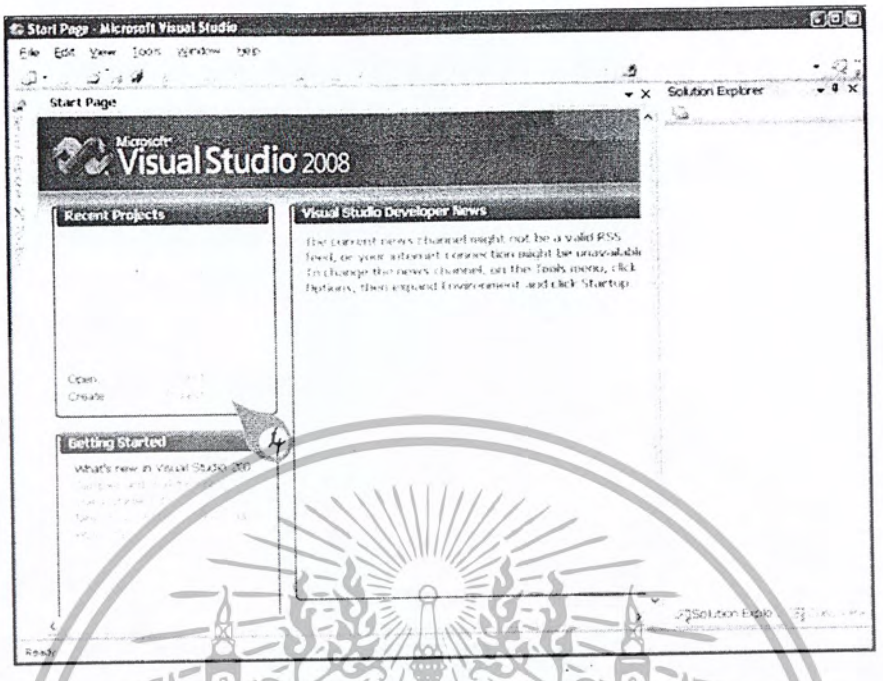
2.10.1.2 คลิกเลือก Visual C# Development Settings เพื่อใช้ภาษา C# เป็นภาษาหลักสำหรับเขียนโปรแกรม

2.10.1.3 คลิกปุ่ม Start Visual Studio



รูปที่ 2.32 ตัวอย่างการเริ่มต้นใช้งานโปรแกรม Visual Studio 2008

2.10.1.4 จะปรากฏหน้าต่าง Start Page ของ Visual Studio 2008 คลิกลิงค์ Creat Project เพื่อสร้างโปรเจ็กต์ใหม่

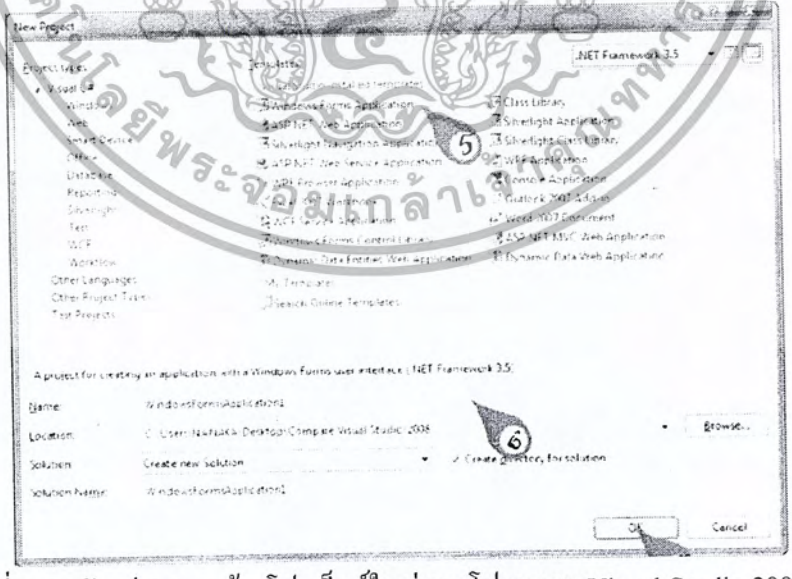


รูปที่ 2.33 ตัวอย่างการสร้างโปรเจ็กต์ใหม่ของโปรแกรม Visual Studio 2008(2)

2.10.1.5 คลิกไอคอน Windows Forms Application

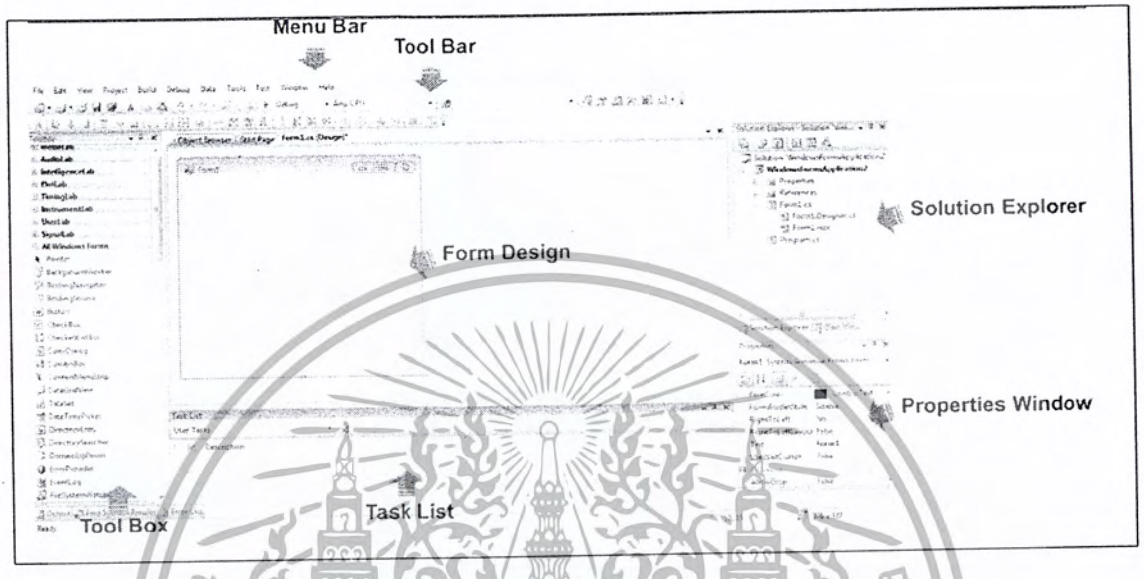
2.10.1.6 ตั้งชื่อ Project ซึ่งก็คือแอพพลิเคชันที่จะสร้างขึ้น

2.10.1.7 คลิกปุ่ม OK



รูปที่ 2.34 ตัวอย่างการสร้างโปรเจ็กต์ใหม่ของโปรแกรม Visual Studio 2008(3)

2.10.1.8 จากนั้นจะพบกับหน้าจอหลักของ Visual Studio 2008 หรือที่เรียกว่า IDE (ย่อมาจาก Integrated Development Environment) ซึ่งประกอบด้วยส่วนต่างๆดังนี้

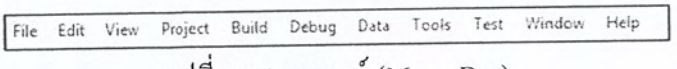


รูปที่ 2.35 ตัวอย่างหน้าจอหลักของโปรแกรม Visual Studio 2008(4)

2.10.2 เครื่องมือต่างๆภายในหน้าจอหลักของโปรแกรม

2.10.2.1 เมนูบาร์ (Menu Bar)

เมนูบาร์เป็นเมนูหลักที่รวบรวมคำสั่งควบคุมการทำงานของ Visual Studio 2008 โดยจัดเป็นกลุ่มคำสั่งแยกตามประเภทการใช้งาน สามารถเรียกใช้ได้โดยการใช้เมาส์คลิกจากเมนูหรือใช้คีย์ลัดจากคีย์บอร์ดก็ได้ มีกลุ่มคำสั่งเรียกใช้งานดังนี้



รูปที่ 2.36 เมนูบาร์ (Menu Bar)

1) File กลุ่มคำสั่งเกี่ยวกับการสร้างโปรเจกต์ใหม่ เปิดโปรเจกต์ บันทึกโปรเจกต์

2) Edit กลุ่มคำสั่งที่ช่วยสร้างและแก้ไขการทำงานของโปรเจกต์ให้ง่ายขึ้น เช่น Copy, Paste และ Undo เป็นต้น

3) View กลุ่มคำสั่งเกี่ยวกับการแสดงผลหน้าต่าง IDE ของ Visual Studio 2008

4) Project กลุ่มคำสั่งเกี่ยวกับจัดการ โปรเจกต์ เช่น การเพิ่ม Form และ Reference เป็นต้น

5) Build กลุ่มคำสั่งเกี่ยวกับการคอมไพล์โปรเจกต์

6) Debug กลุ่มคำสั่งเกี่ยวกับการหาข้อผิดพลาดของโปรเจกต์

7) Data กลุ่มคำสั่งเกี่ยวกับการติดต่อกับฐานข้อมูล

8) Format กลุ่มคำสั่งเกี่ยวกับการออกแบบ Form

9) Tools กลุ่มคำสั่งเรียกเครื่องมือต่างๆ

10) Test กลุ่มคำสั่งเกี่ยวกับการทดสอบการทำงานของโปรแกรม

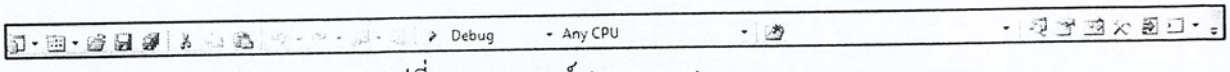
11) Analyze กลุ่มคำสั่งเกี่ยวกับการวัดประสิทธิภาพของโปรแกรม

12) Window กลุ่มคำสั่งเกี่ยวกับการจัดการรูปแบบการแสดงผล IDE ของ Visual Studio 2008

13) Help คำสั่งให้ความช่วยเหลือของ Visual Studio 2008

2.10.2.2 ทูลบาร์ (Toolbar)

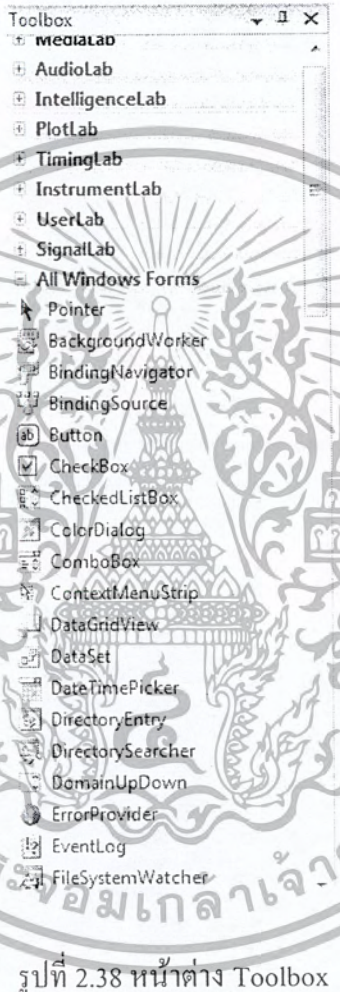
ทูลบาร์เป็นเครื่องมือที่ช่วยให้เรียกใช้งานคำสั่งในเมนูบาร์บางคำสั่งที่ใช้งานบ่อยครั้งได้สะดวกขึ้น



รูปที่ 2.37 ทูลบาร์ (Tool Bar)

2.10.2.3 หน้าต่าง Toolbox

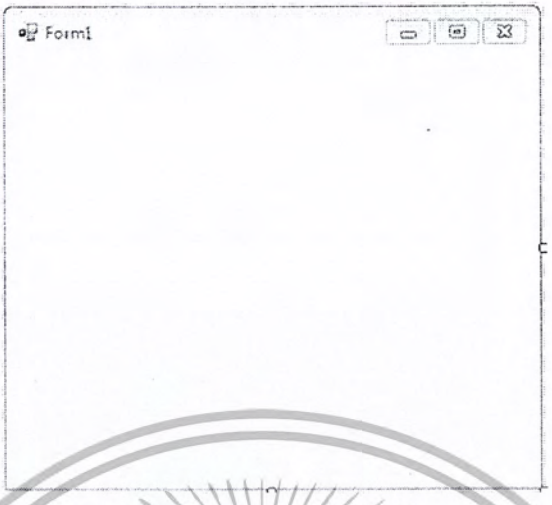
Toolbox เป็นหน้าต่างที่แสดงคอนโทรลและคอมโพเนนต์ต่างๆ เพื่อให้สะดวกในการสร้างแอพพลิเคชัน ซึ่งมีการจัดแบ่งคอนโทรลและคอมโพเนนต์ต่างๆ ออกเป็นกลุ่มตามลักษณะการใช้งาน



รูปที่ 2.38 หน้าต่าง Toolbox

2.10.2.4 หน้าต่าง Form Designer

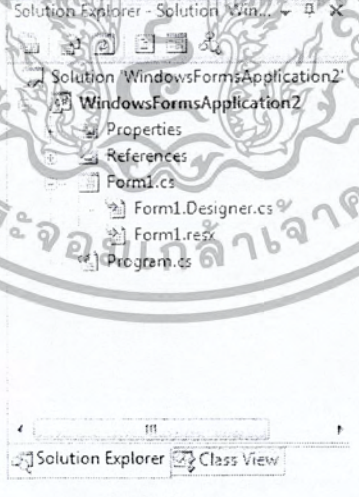
Form Designer เป็นหน้าต่างที่ใช้สำหรับการออกแบบหน้าตาของแอพพลิเคชัน โดยลากตอนโทรลต่างๆจาก Toolbox Window มาวางบน Form ตามต้องการ Visual Studio 2008 จะสร้างโค้ดการออกแบบให้โดยอัตโนมัติเป็นความสามารถของ Visual Studio 2008 ที่ช่วยให้เราพัฒนาแอพพลิเคชันได้ง่ายและมีประสิทธิภาพมากขึ้น



รูปที่ 2.39 หน้าต่าง Form Designer

2.10.2.5 Solution Explorer

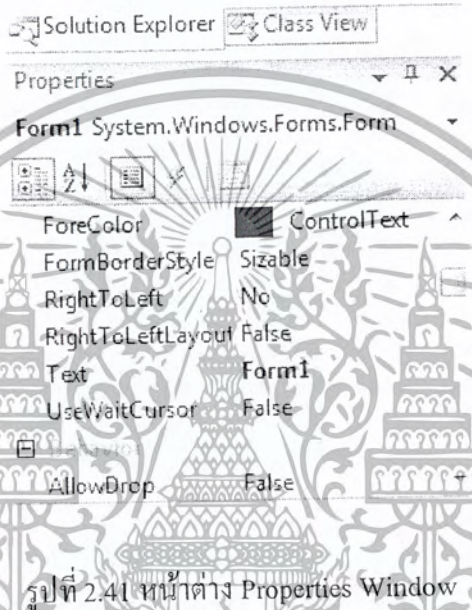
Solution Explorer เป็นหน้าต่างแสดงรายการของไอเท็ม (item) ที่มีอยู่โปรเจ็กต์ทั้งหมด เช่น Form Module, Component และ Class เป็นต้น ซึ่งในมุมของ Visual Studio นั้นจะมองว่า Solution ก็คือแอพพลิเคชันที่ผู้พัฒนากำลังสร้างขึ้นมา



รูปที่ 2.40 หน้าต่าง Solution Explorer

2.10.2.6 Properties Window

Properties Window เป็นหน้าต่างแสดงและกำหนดคุณสมบัติขององค์ประกอบต่างๆที่จะเห็นเป็นหน้าตาของแอพพลิเคชันนั่นก็หมายถึง การกำหนดค่าให้กับคอนโทรลและออบเจ็กต์ เช่น กำหนดชื่อ, สี และ ขนาด เป็นต้น (นอกจากจะกำหนดผ่านหน้าต่างนี้เรายังกำหนดผ่านการเขียนคำสั่งใน C# ได้ด้วย)



รูปที่ 2.41 หน้าต่าง Properties Window

2.10.2.7 หน้าต่าง Task List

Task List เป็นหน้าต่างแสดงคำอธิบายต่างๆหรือเป็นคำแนะนำที่เป็นประโยชน์เกี่ยวกับการสร้างแอพพลิเคชันในแต่ละขั้นตอน อาจจะแสดงตำแหน่งของข้อผิดพลาดที่เกิดขึ้น รวมทั้งอนุญาตให้เราเขียนโน้ต/บันทึกสั้นๆ ไว้เตือนความจำได้ด้วย

รูปที่ 2.42 หน้าต่าง Task List

2.11 โปรแกรม Processing

Processing เป็นโปรแกรมที่เปิดให้ดาวน์โหลดได้ฟรีสำหรับผู้สนใจทุกคนที่ต้องการจะสร้างโปรแกรม โดยใช้ภาษาจาวาและยังถูกพัฒนาขึ้นมาสำหรับผู้ที่ต้องการสร้างหรือตกแต่งรูปภาพ ภาพเคลื่อนไหว อีกด้วยในตอนแรกนั้นโปรแกรม Processing ได้ถูกพัฒนาขึ้นมาสำหรับการให้บริการเกี่ยวกับการสร้างซอฟต์แวร์ และไว้สำหรับผู้เริ่มต้นเพื่อฝึกหัดการเขียนโปรแกรมเบื้องต้นในรูปแบบของ Visual Context Processing ต่อมาตัวโปรแกรมยังพัฒนาฟังค์ชั่นต่างๆ เพื่อนสร้างและประมวลผล สำหรับงานของมืออาชีพ ในตอนนี้กว่าหมื่นคนของนักเรียน นักศึกษา ศิลปิน นักออกแบบ นักค้นคว้า ได้หันมาใช้ Processing ในการศึกษาค้นคว้าและสร้างผลิตภัณฑ์ที่เป็นซอฟต์แวร์ รูปภาพ ภาพเคลื่อนไหว เสียงโดยโปรแกรม Processing สามารถดาวน์โหลดได้ที่ http://www.processing.org/download โดยโปรแกรมและหนังสือ "A Programming Handbook for Visual Designers and Artists" ของ Casey Reas และ Ben Fry แสดงได้ดังรูปที่ 2.43

รูปที่ 2.43 โปรแกรม Processing และหนังสือ "A Programming Handbook for Visual Designers and Artists" [11]

2.11.1 คุณสมบัติของโปรแกรม Processing

2.11.1.1 Processing เปิดให้ดาวน์โหลดได้โดยไม่เสียค่าใช้จ่ายและเปิดให้ผู้ใช้งานพัฒนาโปรแกรมได้ (Open Source)

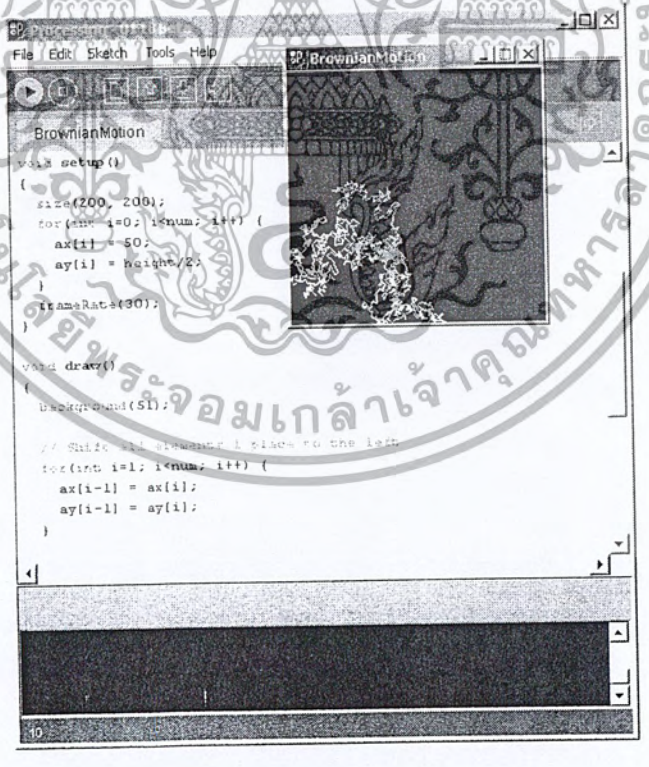
2.11.1.2 Processing สามารถใช้งานร่วมกับภาพ 2 มิติ 3 มิติ และไฟล์ PDF

2.11.1.3 ใช้งานได้บน OS GNU/Linux, Mac OS X, and Windows

2.11.1.4 มีมากกว่า 100 Libraries Software ที่ใช้สำหรับโปรแกรมเสียง วิดีโอ และอื่นๆ

2.11.2 หน้าต่างโปรแกรม Processing และคำสั่งบนโปรแกรม

หน้าต่างโปรแกรม Processing และคำสั่งบนโปรแกรมแสดงได้ดังรูปที่ 2.44



รูปที่ 2.44 หน้าต่างโปรแกรม Processing [11]

คำสั่งของโปรแกรม Processing

1) Run คอมไพล์โค้ดที่เราเขียนขึ้น และ ประมวลผล

2) Stop หยุดการทำงานของคำสั่ง Run

3) New สร้างหน้าใส่คำสั่งเปล่าขึ้นมาใหม่

4) Open เป็นคำสั่งที่ใช้ในการเรียกดูไฟล์จากเครื่องคอมพิวเตอร์

5) Save เป็นคำสั่งที่ใช้ในการบันทึกไฟล์ลงในเครื่องคอมพิวเตอร์

6) Export เป็นการบันทึกไฟล์งานในรูปแบบนามสกุลอื่นๆ เพื่อประโยชน์ในการใช้งานร่วมกับโปรแกรมอื่นๆ

2.11.3 คำสั่ง Toolbar ของโปรแกรม Processing

2.11.3.1 File หมวดจัดการไฟล์งาน ประกอบไปด้วยคำสั่ง

1) New (Ctrl+N) สร้างหน้าต่างใส่คำสั่งเปล่าขึ้นมาใหม่

2) Open (Ctrl+O) เปิดไฟล์งานที่save ไว้ขึ้นมา

3) Sketchbook เรียกดูไฟล์Sketch จาก Sketchbook folder

4) Examples เรียกดูไฟล์โปรแกรมตัวอย่างของ Processing

5) Close (Ctrl+W) ปิดการทำงานของโปรแกรมที่เขียนอยู่

6) Save (Ctrl+S) บันทึกไฟล์ที่ใช้งานอยู่

7) Save as... (Ctrl+Shift+S) บันทึกไฟล์ใหม่

8) Export (Ctrl+E) บันทึกไฟล์เป็นนามสกุลอื่นเพื่อใช้งานร่วมกันโปรแกรมที่เกี่ยวข้อง

9) Export Application (Ctrl+Shift+E) บันทึกไฟล์เป็นแอพพลิเคชั่นจาวาพร้อมใช้งาน

10) Page Setup (Ctrl+Shift+P) ยังใช้งานไม่ได้ในตอนนี้

11) Print (Ctrl+P) ยังใช้งานไม่ได้ในตอนนี้

12) Preferences (Ctrl+,) ปรับค่าการทำงานของโปรแกรม Processing

13) Quit (Ctrl+Q) ออกจากโปรแกรมProcessing

2.11.3.2 Edit หมวดแก้ไขข้อความที่กำลังเขียนลงในโปรแกรม

1) Undo (Ctrl+Z) กลับไปยังคำสั่งหรือการทำงานก่อนหน้านี้

2) Redo (Ctrl+Y) ยกเลิกคำสั่ง Undo

3) Cut (Ctrl+X) ลบข้อความคำสั่งที่คัดลอกไว้

4) Copy (Ctrl+C) คัดลอกคำสั่งหรือข้อความต่างๆ

5) Copy for Discourse (Shift+Ctrl+C) คัดลอกด้วยฟังก์ชั่น Discourse กรุณาอ่านรายละเอียดเพิ่มเติมที่เว็บไซท์

6) Paste (Ctrl+V) วางข้อความที่คัดลอกไว้ลงบนโปรแกรม Processing

7) Select All (Ctrl+A) เลือกข้อความหรือคำสั่งเนื้อหาทั้งหมดที่พิมพ์หรือเขียนไว้บนโปรแกรม Processing

8) Comment/Uncomment (Ctrl+/) โน้ตข้อความเพิ่มเติมหรือยกเลิก

9) Increase Indent (Ctrl+]) เพิ่มพื้นที่เขียนโปรแกรม

Decrease Indent (Ctrl+[) ลดพื้นที่เขียนโปรแกรม

10) Find (Ctrl+F) หาตัวอักษรหรือประโยคบนหน้าต่างคำสั่ง 11) Find Next (Ctrl+G) ค้นหาถัดไป

2.11.3.3 Sketch เป็นคำสั่งเริ่มการทำงานและหยุดการทำงานรวมไปถึงการเพิ่มมีเดียไฟล์และไลบารีของไฟล์

1) Run (Ctrl+R) Compile หรือ RUN เริ่มต้นการทำงานของคำสั่งโปรแกรมProcessing

2) Present (Ctrl+Shift+R) แสดงผลที่เกิดขึ้นจากคำสั่งของโปรแกรม

3) Stop หยุดการทำงานของคำสั่ง RUN

4) Import Library เพิ่ม libaries ลงในหน้าคำสั่ง

5) Show Sketch Folder เปิดโฟลเดอร์ล่าสุดที่ทำการ Sketch

6) Add File เพิ่มหรือดึงไฟล์ลงในหน้าต่างคำสั่ง

2.11.3.4 Tools เครื่องมือจัดการฟอนต์และรูปแบบต่างๆ

1) Auto Format (Ctrl-T) เป็นตัวจัดการปรับฟอนต์บนหน้าต่างคำสั่งของเราให้เป็นแบบปกติ

2) Create Font... เปลี่ยนฟอนต์เป็นแบบของ Processing เพื่อประยุกต์การใช้งาน กรุณาอ่านรายละเอียดเพิ่มเติมที่เว็บไซท์

3) Color Selector ปรับสี

4) Archive Sketch แปลงไฟล์ Sketch ให้เป็น.zip

5) Fix Encoding and Reload เปลี่ยนรูปแบบการเข้ารหัสกรุณาอ่านรายละเอียดเพิ่มเติมที่เว็บไซท์

2.10.3.5 Help การสอนการใช้งานของโปรแกรม

1) Getting Started เปิดแหล่งข้อมูลที่มาของโปรแกรมProcessing

2) Troubleshooting เปิดตัวช่วยแก้ปัญหาต่างๆของโปรแกรมProcessing

3) Reference เปิดข้อมูลหรือแหล่งที่มาต่างๆของโปรแกรม

4) Find in Reference (Ctrl+Shift+F) ค้นหาข้อมูลต่างๆเกี่ยวกับโปรแกรม

5) Frequently Asked Questions ตอบคำถามพื้นฐานที่พบบ่อยๆ

6) Visit Processing.org (Ctrl+5) เข้าไปสู่เว็บไซท์ของโปรแกรม Processing

7) About Processing เรียกดูข้อมูลเนื้อหาคร่าวๆเกี่ยวกับโปรแกรม Processing

บทที่ 3

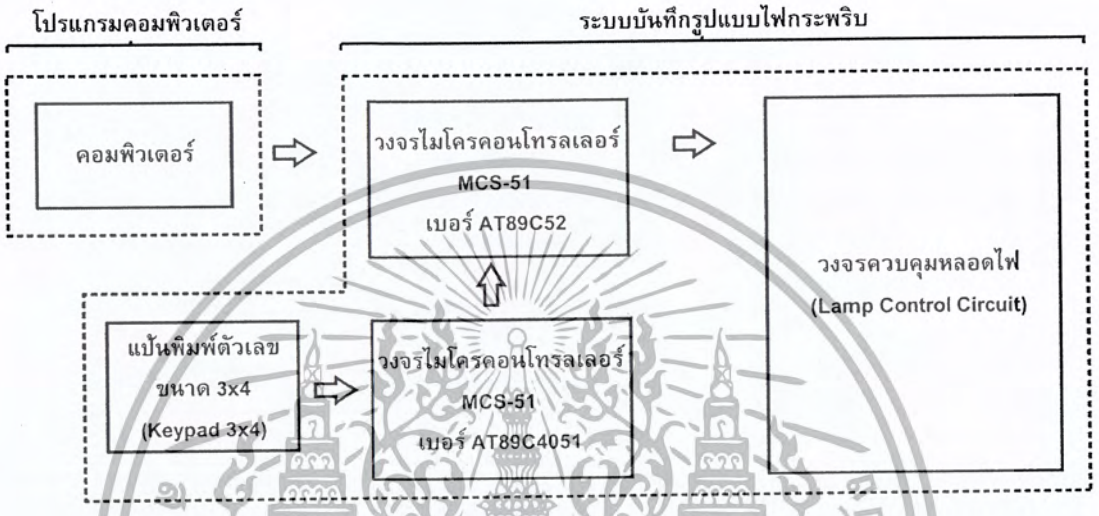
การออกแบบและการจัดทำปริญญานิพนธ์

การออกแบบระบบการทำงานของระบบไฟสถานบันเทิงแบบอัตโนมัติตามจังหวะเสียงดนตรี มีหลักการทำงานโดยรวมของระบบสองส่วนที่สำคัญคือ ในส่วนแรกจะเป็นการเขียนโปรแกรมบนคอมพิวเตอร์เพื่อตรวจจับบีตที่เกิดขึ้นจากเพลงต่างๆที่เล่นผ่านโปรแกรมที่สร้างขึ้นบนโปรแกรมจะมีโหมดการทำงานสามโหมดคือ สั่งการให้หลอดไฟกระพริบตามจังหวะเสียงเพลง (Beat Detection Mode), สุ่มเลือกรูปแบบการกระพริบตามระดับความเร็วของเพลง (Random Mode) และการกระพริบตามรูปแบบสำเร็จรูป (Manual Mode) จากนั้นคอมพิวเตอร์จะส่งข้อมูลไปสั่งการให้ระบบบันทึกรูปแบบการกระพริบของหลอดไฟซึ่งประกอบด้วยไมโครคอนโทรลเลอร์ทำการสั่งการให้หลอดไฟกระพริบตามโหมดที่เลือกโดยการเชื่อมต่อด้วยการส่งข้อมูลแบบอนุกรมตามมาตรฐาน RS-232

ส่วนที่สองจะเป็นระบบบันทึกรูปแบบการกระพริบของหลอดไฟจะเกี่ยวข้องกับการเขียนโปรแกรมสั่งงานไมโครคอนโทรลเลอร์ให้บันทึกรูปแบบการกระพริบของหลอดไฟและส่งสัญญาณไปควบคุมวงจรควบคุมหลอดไฟให้กระพริบตามรูปแบบที่กำหนดโดยจะมีโหมดการทำงานที่สามารถเลือกรูปแบบการกระพริบของหลอดไฟโดยควบคุมการทำงานด้วยคีย์แพด

3.1 การออกแบบ

การออกแบบระบบทั้งหมดสามารถแสดงเป็นบล็อกไดอะแกรม ได้ดังนี้



รูปที่ 3.1 บล็อกไดอะแกรมระบบไฟสถานบันเทิงแบบอัตโนมัติตามจังหวะเสียงดนตรี

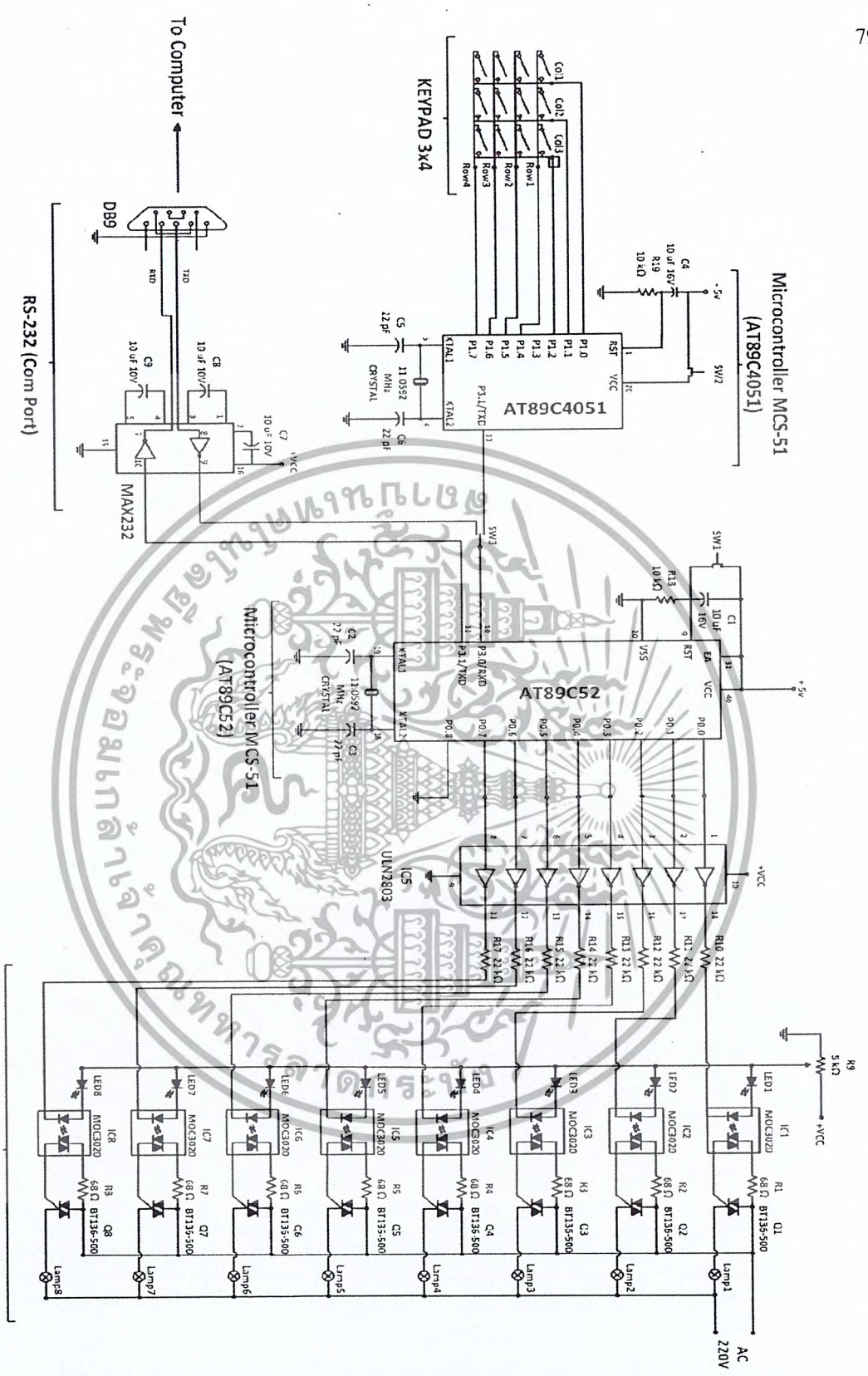
จากรูปที่ 3.1 ประกอบด้วยการทำงานหลักๆ 2 ส่วนได้แก่ การเขียนโปรแกรมบนคอมพิวเตอร์ในรูปแบบของโปรแกรมตรวจจับบีตด้วยภาษาจาวาโดยใช้โปรแกรม Processing และการเขียนแอพพลิเคชั่นบนคอมพิวเตอร์ในรูปแบบของโปรแกรมเล่นเพลงด้วยภาษา C# ด้วยโปรแกรม Visual Studio 2008 ซึ่งมีโหมดการควบคุมการกระพริบของหลอดไฟให้เรียกใช้งานประกอบด้วยสามโหมดคือ กระพริบตามจังหวะเสียงเพลง (Beat Detection Mode), สุ่มเลือกการกระพริบตามระดับความเร็วของเพลง (Select Mode), การกระพริบตามรูปแบบสำเร็จรูป (Manual Mode)

อีกส่วนหนึ่งคือส่วนบันทึกรูปแบบการกระพริบของหลอดไฟซึ่งเกี่ยวข้องกับการเขียนโปรแกรมสั่งงานไมโครคอนโทรลเลอร์เพื่อบันทึกรูปแบบการกระพริบของหลอดไฟให้กระพริบตามรูปแบบที่กำหนด บล็อกไดอะแกรมในส่วนนี้ประกอบด้วย 4 ส่วนหลัก ได้แก่ วงจรไมโครคอนโทรลเลอร์ MCS-51 เบอร์ AT89C52, วงจรไมโครคอนโทรลเลอร์ MCS-51 เบอร์

AT89C4051, คีย์แพดขนาด 3x4 (Keypad 3x4), วงจรควบคุมหลอดไฟ (Lamp Control Circuit) และวงจรมาตรฐาน RS-232 ซึ่งจะกล่าวเพิ่มเติมต่อไป

3.1.1 การออกแบบระบบบันทึกรูปแบบไฟกระพริบ

การออกแบบทางด้านระบบบันทึกรูปแบบไฟกระพริบเกี่ยวข้องทั้งในส่วนของการออกแบบวงจรและการเขียนโปรแกรมสั่งงานไมโครคอนโทรลเลอร์เพื่อบันทึกรูปแบบการกระพริบของหลอดไฟให้กระพริบตามรูปแบบที่กำหนด วงจรในส่วนนี้ แสดงได้ดังรูปที่ 3.2

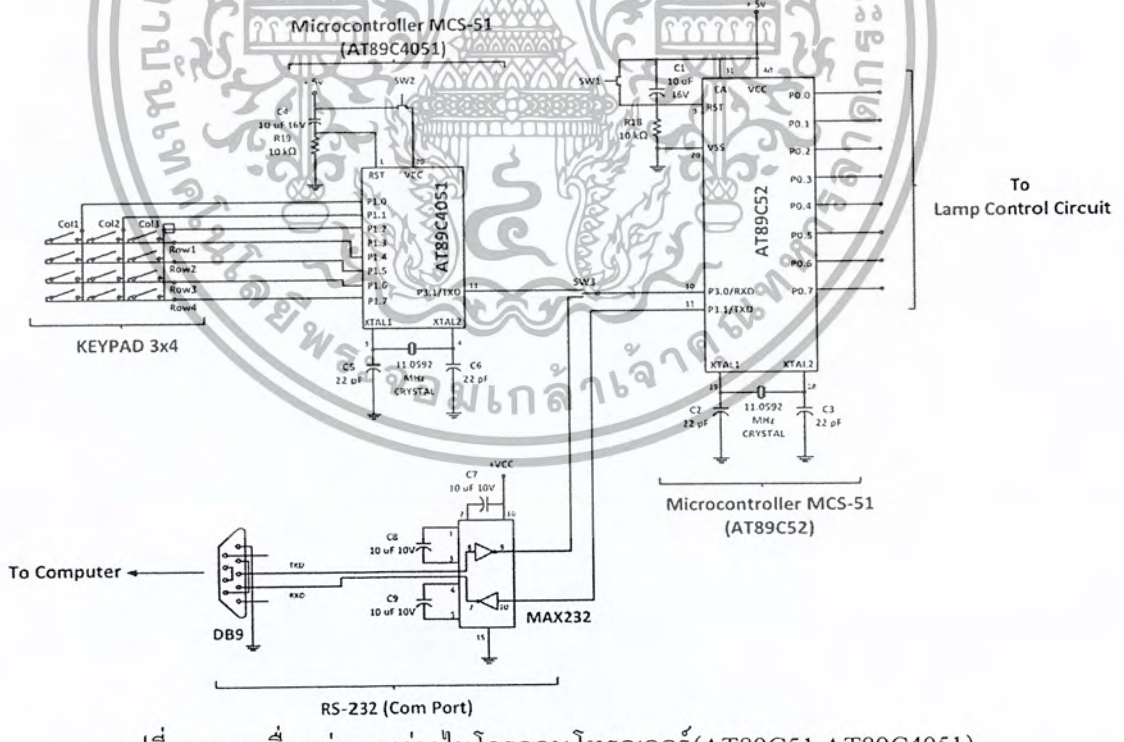


รูปที่ 3.2 วงจรรวมของระบบไฟสถานบันเทิงแบบอัตโนมัติตามจังหวะเสียงดนตรี

จากรูปที่ 3.2 วงจรประกอบด้วย 5 ส่วน ได้แก่ คีย์แพดขนาด 3x4 (Keypad 3x4) , วงจรไมโครคอนโทรลเลอร์ MCS-51 เบอร์ AT89C52, วงจรควบคุมหลอดไฟ (Lamp Control Circuit) และวงจรมาตรฐานการเชื่อมต่อ RS-232

หากพิจารณาจากรูปที่ 3.2 จะเห็นว่าไมโครคอนโทรลเลอร์ (AT89C52) ไมโครคอนโทรลเลอร์ (AT89C4051) และ วงจรมาตรฐานการเชื่อมต่อ RS-232 จะเชื่อมถึงกันอยู่ด้วยสวิตซ์ (SW3) จึงจะขออธิบายในสี่ส่วนที่เชื่อมต่อกันก่อนได้แก่ คีย์แพดขนาด 3x4 (Keypad 3x4) วงจรไมโครคอนโทรลเลอร์ MCS-51 (AT89C52) , วงจรไมโครคอนโทรลเลอร์ MCS-51 (AT89C4051) และ วงจรมาตรฐานการเชื่อมต่อ RS-232 หลังจากนั้นจะอธิบายในส่วนของวงจรควบคุมหลอดไฟต่อไป

3.1.1.1 การออกแบบการเชื่อมต่อระหว่างไมโครคอนโทรลเลอร์ (AT89C51,AT89C4051), คีย์แพด และ วงจรมาตรฐาน RS-232

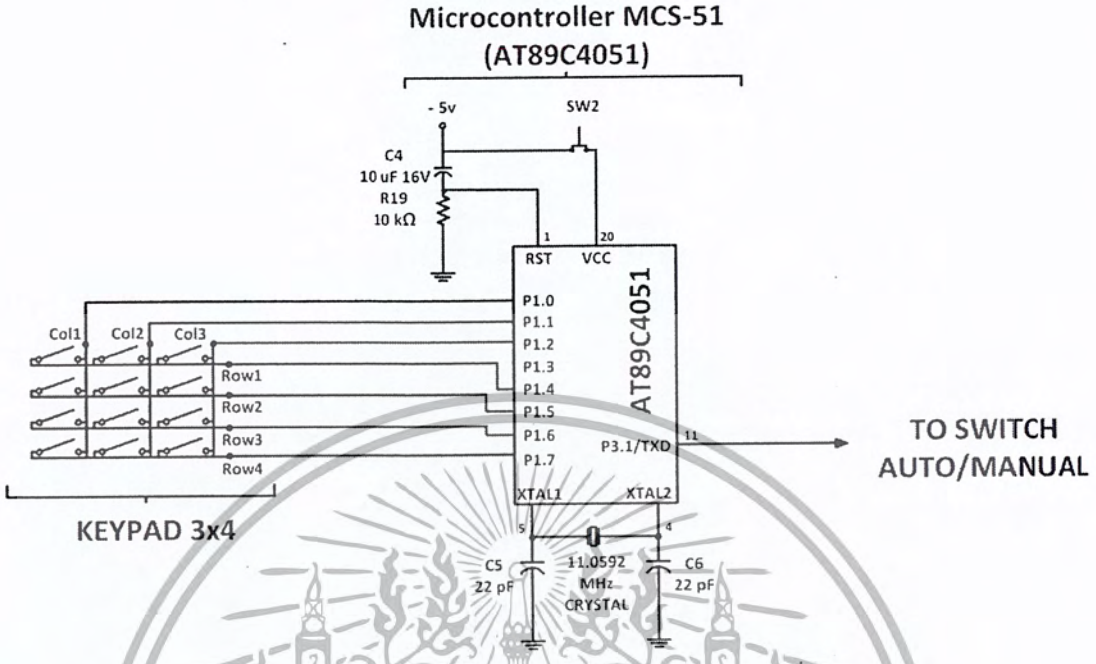


รูปที่ 3.3 การเชื่อมต่อระหว่างไมโครคอนโทรลเลอร์(AT89C51,AT89C4051), คีย์แพด และ วงจรมาตรฐาน RS-232

เมื่อพิจารณารูปที่ 3.3 จะพบว่าการเชื่อมต่อระหว่างไมโครคอนโทรลเลอร์ (AT89C52,AT89C4051), คีย์แพด และ วงจรมาตรฐาน RS-232 เชื่อมต่อเข้าด้วยกันด้วยสวิตซ์สองทาง (SW3) โดยคีย์แพดจะถูกเชื่อมต่อเข้ากับไมโครคอนโทรลเลอร์ เบอร์ AT89C4051 เนื่องจากต้องการส่งข้อมูลผ่านพอร์ตสื่อสารอนุกรม(ขา Tx ของ AT89C4051)เพื่อทำการใช้งานการอินเตอร์รัพท์จากพอร์ตอนุกรม ซึ่งจะทำให้การควบคุมรูปแบบไฟกระพริบได้อย่างอิสระเมื่อมีการส่งข้อมูลจากทั้งสองทางเช่นเมื่อมีการเปลี่ยนโหมดรับค่าอินพุตจากคีย์แพดหรือจากคอมพิวเตอร์

สำหรับการเชื่อมต่ออีกรูปแบบหนึ่งคือการเชื่อมต่อระหว่างคอมพิวเตอร์ผ่านวงจรมาตรฐาน RS-232 เชื่อมต่อเข้าที่พอร์ต P3.0/RXD ของไมโครคอนโทรลเลอร์ เบอร์ AT89C52 ซึ่งเป็นพอร์ตเดียวกันกับการเชื่อมต่อของไมโครคอนโทรลเลอร์ (AT89C4051) โดยจะรับค่าอินพุตจากคอมพิวเตอร์ (ค่าอินพุตที่ใช้ระบุรูปแบบการกระพริบของหลอดไฟที่ได้รับจากการประมวลผลแล้วด้วยคอมพิวเตอร์) เพื่อเลือกรูปแบบการกระพริบของหลอดไฟเช่นเดียวกัน จึงใช้สวิตซ์สองทาง (SW3) เพื่อสลับการทำงานของทั้งสองวงจรโดยแยกการทำงานเป็นสองลักษณะและเลือกทำงานอย่างใดอย่างหนึ่งเท่านั้น กล่าวคือ ลักษณะแรกจะเป็นการเลือกรูปแบบการกระพริบของหลอดไฟโดยรับค่าจากคอมพิวเตอร์ หรือ ลักษณะที่สองจะเป็นการเลือกรูปแบบการกระพริบด้วยคีย์แพด โดยทั้งสองแบบจะใช้การเชื่อมต่อผ่านพอร์ตอนุกรม (Serial Port) ซึ่งกันและกัน

ในส่วนแรกนี้จะกล่าวถึงการออกแบบการเชื่อมต่อและใช้งานคีย์แพด (คีย์แพด) ขนาด 3x4 เพื่อใช้งานเป็นสวิตซ์ให้กับไมโครคอนโทรลเลอร์ (AT89C4051) เพื่อแสดงรูปแบบของรูปแบบไฟกระพริบที่ต่างกัน เนื่องจากคีย์แพดประกอบด้วยสวิตซ์จำนวนหลายตัวมาต่อกันเป็นแมทริกซ์ และใช้เทคนิคการถอดรหัสและการเลือกว่าคีย์ใดถูกกดให้ดูคีย์ในแนวที่แถวและคอลัมน์ตรงกัน การออกแบบใช้หลักการง่ายๆว่าเมื่อคีย์ใดถูกกดค่าแถวและคอลัมน์ตรงที่ตรงกับคีย์นั้นจะมีค่าลอจิกเป็น 0 พร้อมกัน



รูปที่ 3.4 การเชื่อมต่อระหว่างไมโครคอนโทรลเลอร์ (AT89C4051) กับคีย์แพด

พิจารณาจากรูปที่ 3.4 ได้ทำการออกแบบโดยใช้เทคนิค Keypad Scanning โดยทำการเชื่อมต่อคีย์แพดเข้าที่พอร์ต 1 ของไมโครคอนโทรลเลอร์ดังรูป ทำการเลือกใช้พอร์ต P1.0 - P1.2 ซึ่งก็คือคอลัมน์ 1 ถึง 3 เป็นคีย์แสกนโดยไมโครคอนโทรลเลอร์จะทำการส่งข้อมูลบิต “0” ไปยัง P1.0 - P1.2 ตามลำดับ ในทุกครั้งที่มีการส่งข้อมูลไปยังสายคอลัมน์คีย์แพด ไมโครคอนโทรลเลอร์จะทำการอ่านค่าที่ P1.4 - P1.7 เข้ามาด้วย หากไม่มีการกดคีย์ค่าของ P1.4 - P1.7 ก็จะเป็น “1” ทั้งหมด ถ้าหากมีการกดคีย์ค่าของ P1.4 - P1.7 ก็จะไม่เป็น “1111” อีกต่อไป เป็นการแจ้งให้ทราบว่ามีการกดคีย์แพดแล้วจากนั้นไมโครคอนโทรลเลอร์ก็จะทำการค้นหาตำแหน่งของคีย์ที่กดความสัมพันธ์ระหว่างคอลัมน์และคีย์ที่ถูกเลือกแสดงได้ดังตารางที่ 3.1 ต่อไปนี้

ตาราง 3.1 ความสัมพันธ์ระหว่างคอลัมน์และคีย์ที่ถูกเลือก

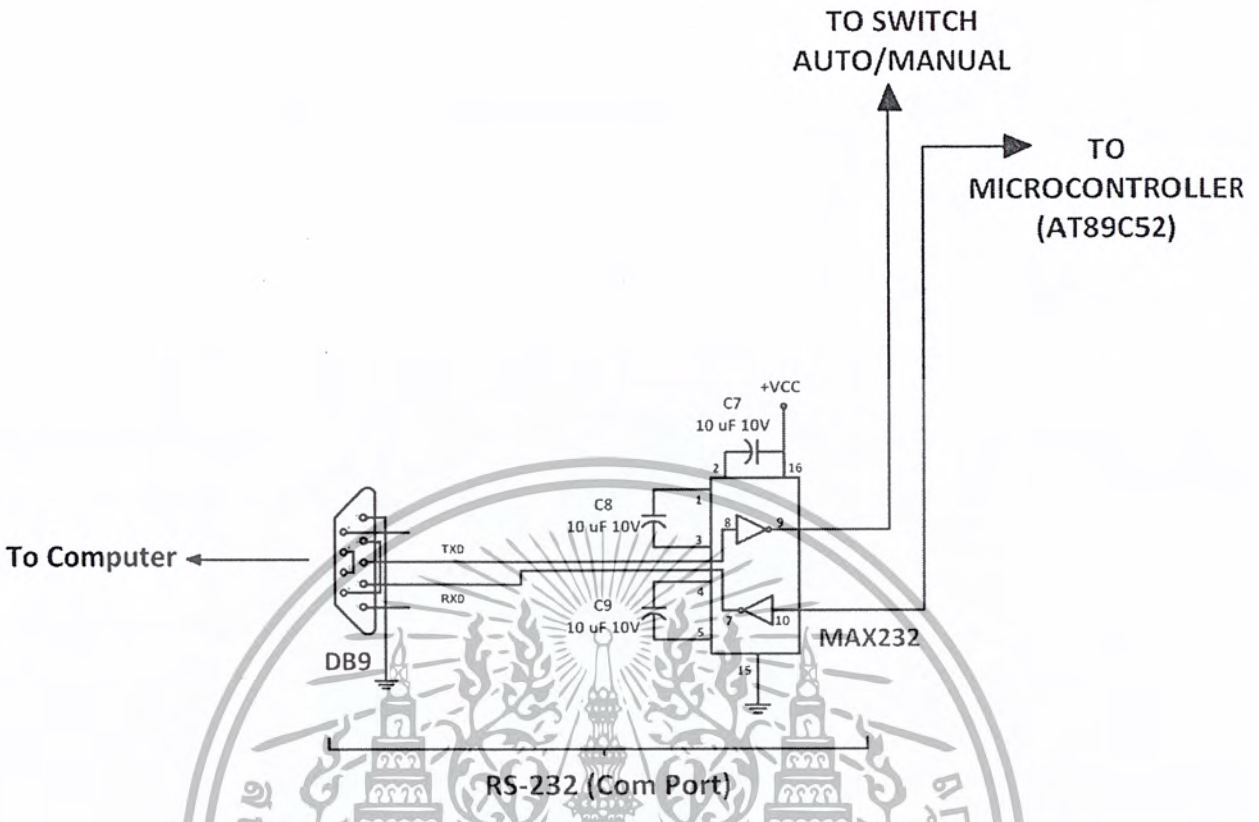
P1.0	P1.1	P1.2	คีย์ที่ถูกเลือก
0	1	1	1 , 4 , 7 และ *
1	0	1	2 , 5 ,8 และ 0
1	1	0	3 , 6 , 9 และ #

จากตารางอธิบายได้ว่าเมื่อคีย์แสกนใดมีค่าเป็นลอจิก 0 คีย์ที่คาดว่าจะถูกกดจะมีแค่คีย์ที่อยู่ในคอลัมน์เดียวกับคีย์แสกนเท่านั้น ยกตัวอย่างเช่น ถ้าคีย์แสกน P1.0 มีค่าเป็นลอจิก 0 คีย์ที่คาดว่าจะถูกกดจะมีเพียงคีย์ 1, 4, 7 และ * เท่านั้น ในปริญญานิพนธ์นี้เลือกใช้คีย์แพดสวิตซ์เพียง 6 ปุ่มเท่านั้นคือคีย์ 1-6 ใช้เป็นสวิตซ์แสดงรูปแบบของรูปแบบดไฟกระพริบที่ต่างกัน สถานะการทำงานของแต่ละคีย์แสดงดังตารางที่ 3.2 ต่อไปนี้

ตารางที่ 3.2 สถานะการทำงานของแต่ละคีย์

คีย์ที่ถูกเลือก	พอร์ตที่มีสถานะลอจิกเป็น 0
1	P1.0 , P1.4
2	P1.1 , P1.4
3	P1.2 , P1.4
4	P1.0 , P1.5
5	P1.1 , P1.5
6	P1.2 , P1.5

ต่อมาไมโครคอนโทรลเลอร์ (AT89C4051) จะส่งสัญญาณเป็นค่าอินพุตไปยังสวิตซ์สองทาง (SW3) ไปยังไมโครคอนโทรลเลอร์ (AT89C52) เพื่อเลือกรูปแบบการกระพริบของหลอดไฟต่อไป ในส่วนของการรับค่าจากคอมพิวเตอร์นั้นจะอธิบายในส่วนของวงจรมาตรฐาน RS-232 ที่เชื่อมต่อเข้ากับสวิตซ์สองทางก่อนจะเชื่อมต่อไปยังไมโครคอนโทรลเลอร์ (AT89C52) เพื่อเลือกรูปแบบการกระพริบของหลอดไฟ โดยอัตโนมัติ สามารถแสดงได้ดังรูปที่ 3.5

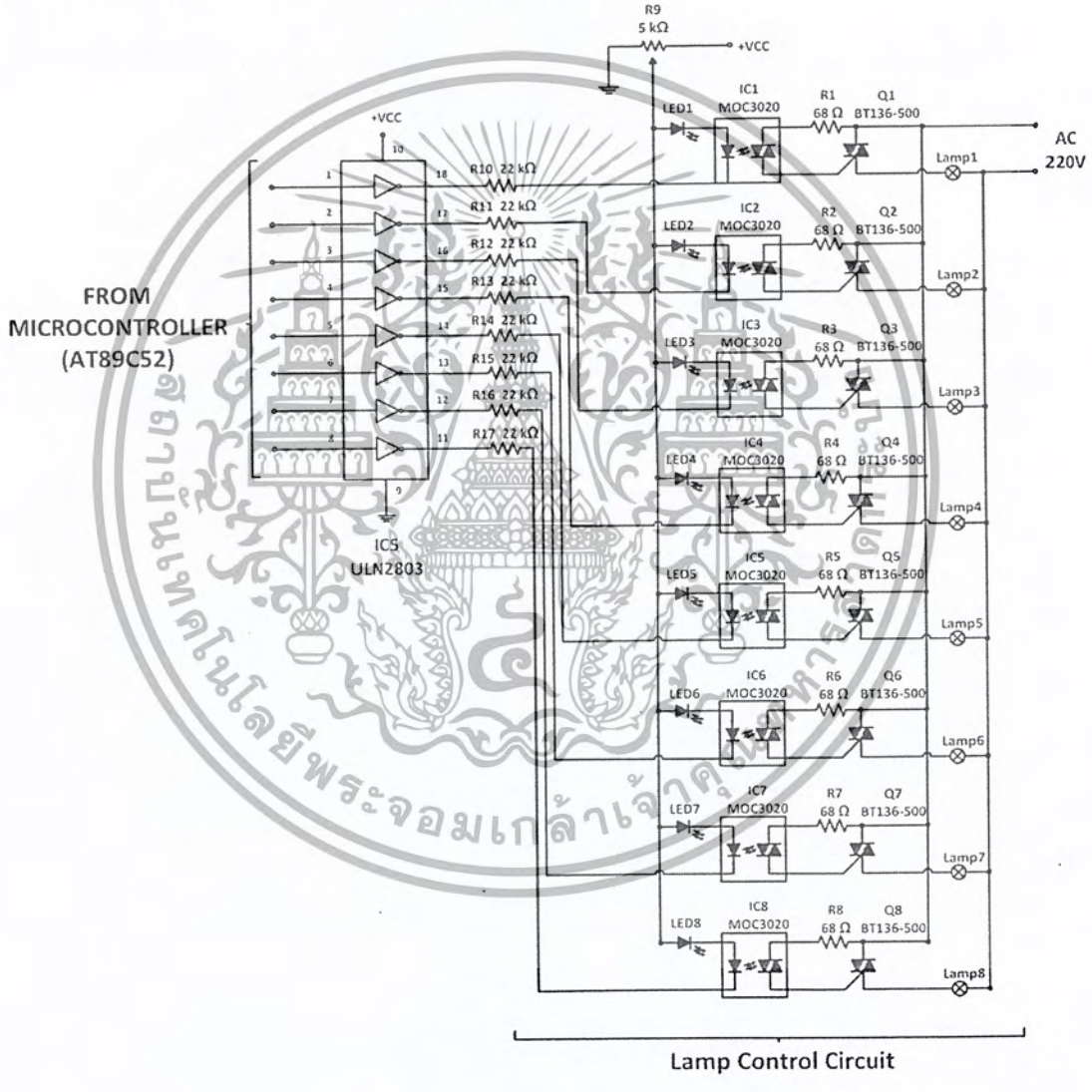


รูปที่ 3.5 การเชื่อมต่อระหว่างวงจรมาตรฐาน RS-232 กับ
ไมโครคอนโทรลเลอร์ (AT89C52) และสวิตซ์สองทาง

RS-232 ย่อมาจาก Recommended Standard-232 เป็นมาตรฐานการเชื่อมต่อข้อมูลแบบอนุกรม กำหนดโดย EIA (Electronics Industry Association) หรือ สมาคมผู้ประกอบการอุตสาหกรรมอิเล็กทรอนิกส์ของอเมริกา ใช้กับการสื่อสารแบบจุดต่อจุด โดยใช้สายเชื่อมต่อ DB แบบ 25 และ 9 เข็มที่ไม่ประสานจังหวะระหว่างคอมพิวเตอร์กับอุปกรณ์ต่อพ่วง หากพิจารณาจากรูปที่ 3.5 จะพบว่าการเชื่อมต่อดังกล่าวมีไอซี MAX-232 เป็นสำคัญ เมื่อคอมพิวเตอร์ทำการส่งข้อมูลรูปแบบการกระพริบที่ต้องการ ก็จะทำการส่งข้อมูลผ่านพอร์ตอนุกรมผ่านทางสาย DB9 เข้าสู่วงจร MAX-232 และทำการส่งข้อมูลออกทางขา 9 ไปยังสวิตซ์สองทางก่อนจะเข้าสู่พอร์ต P3.0/RXD ของไมโครคอนโทรลเลอร์ (AT89C52) เพื่อเลือกรูปแบบการกระพริบของหลอดไฟโดยอัตโนมัติ

3.1.1.2 การออกแบบวงจรควบคุมหลอดไฟ

เป็นส่วนที่ใช้ควบคุมการกระพริบของหลอดไฟ จากวงจรได้กำหนดช่องการควบคุมไว้ 4 ช่องควบคุม (ขา 1-4 ของ IC5) ซึ่งแต่ละช่องควบคุมนั้นมีโครงสร้างที่เหมือนกันหมด ดังนั้นการอธิบายการทำงานของวงจรจะขอเน้นอธิบายเพียงช่องควบคุมช่องเดียว พิจารณาจากรูปที่ 3.6 วงจรควบคุมหลอดไฟ ประกอบการอธิบายดังนี้



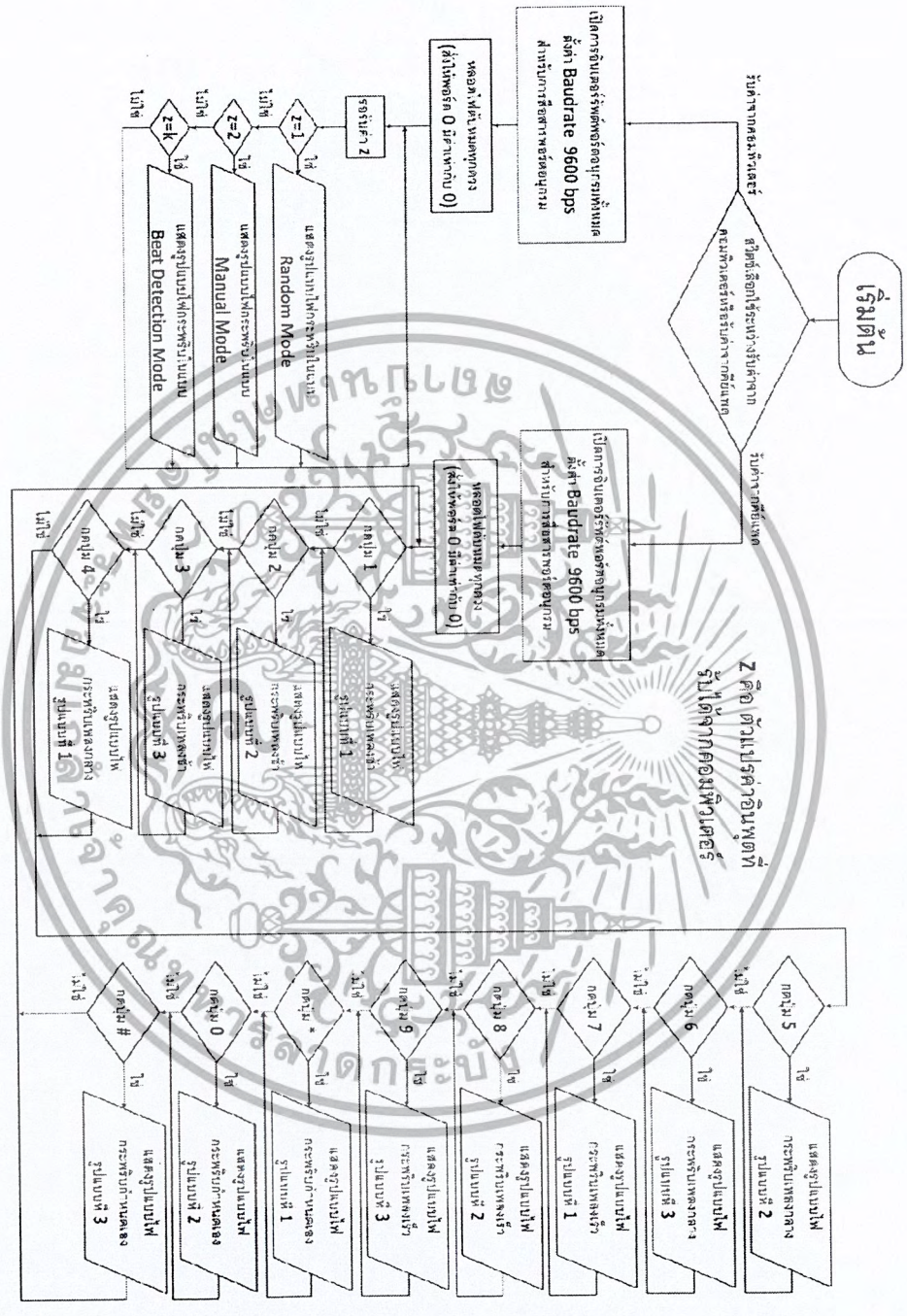
รูปที่ 3.6 วงจรควบคุมหลอดไฟ (Lamp Control Circuit)

วงจรจะประกอบด้วย IC5 (ULN2803) ซึ่งเป็นไอซีไดรเวอร์ โดยภายในจะบรรจุอินเวอร์เตอร์เกต แบบคอลเล็กเตอร์เปิดจำนวน 8 ตัว โดยสามารถใช้กับแรงดันสูงสุดได้ถึง 50 โวลต์ และให้กระแสเอาต์พุตได้สูงสุดในแต่ละขา 500 mA นอกจากนั้นยังมีไดโอดป้องกันไว้ที่ทุกขาของเอาต์พุต ทำให้สามารถต่อกับโหลดที่เป็นขดลวดได้ โดยจะทำหน้าที่เป็นอินเวอร์เตอร์ควบคุมสัญญาณที่รับมาจากพอร์ต 0 ของไมโครคอนโทรลเลอร์ซึ่งจะอยู่ในลักษณะของสัญญาณระดับลอจิก (“0”หรือ “1”) สัญญาณที่ออกจาก IC5 จะถูกป้อนเข้าที่ขา 2 ของ IC1 (MOC3020) เป็นออพโตไอโซเลเตอร์ภายในประกอบด้วย LED และอุปกรณ์รับแสงแบบโฟโตไดแอก แหล่งจ่ายไฟ 12 V จะต่ออยู่กับ LED1 ซึ่งเป็น LED ขนาด5mm. เป็นตัวบอกสถานะการทำงานของ LED ภายใน IC1

หากสัญญาณอินพุตที่เข้ามาที่ขา 2 ของ IC1เป็นสัญญาณลอจิก “1” LED ทางด้านแคโทด ภายใน IC1 จะไม่สว่างซึ่งจะทำให้โฟโตไดแอกไม่สามารถนำกระแสไปกระตุ้นขาเกตของไตรแอก Q1ได้ จึงไม่มีการขับแรงดันควบคุมหลอดไฟทางเอาท์พุต ในทางกลับกันหากสัญญาณอินพุตที่เข้ามาที่ขา 2ของ IC1เป็นสัญญาณลอจิก “0” LED ภายใน IC1 จะสว่างซึ่งจะทำให้โฟโตไดแอกสามารถนำกระแสไปกระตุ้นขาเกตของไตรแอก Q1ได้ จึงมีการขับแรงดันควบคุมหลอดไฟทางเอาต์พุตได้ นั่นคือเอาต์พุตของไตรแอกทำงานตามสภาวะทางอินพุตของ IC1

ข้อควรระวังสำหรับระดับลอจิกที่จะป้อนเข้ามายัง LED ภายใน IC1นั้นจะต้องมีระดับแรงดันไม่เกิน 3 VDC และกระแสสูงสุดไม่เกิน 50 mA หากระดับแรงดันลอจิกอินพุตเกินกว่านี้ควรเพิ่มตัวต้านทานจำกัดกระแสและแรงดันอนุกรมให้กับ LED ภายใน IC1ด้วย

3.1.2 การเขียนโปรแกรมสั่งการไมโครคอนโทรลเลอร์



รูปที่ 3.7 โฟลว์ชาร์ตของโปรแกรม

ในส่วนนี้ใช้ภาษาซีในการเขียนโปรแกรมสั่งงานไมโครคอนโทรลเลอร์ AT89C52 และ AT89C4051 สำหรับรับค่าจากคีย์แพดและสำหรับรับค่าจากคอมพิวเตอร์ ให้ทำการเก็บรูปแบบการกระพริบของหลอดไฟและส่งข้อมูลไปยังวงจรควบคุมหลอดไฟเมื่อมีการกดคีย์แพดหรือได้รับค่าอินพุตจากคอมพิวเตอร์ จากโฟลว์ชาร์ตของโปรแกรม รูปที่ 3.7 รายละเอียดของโปรแกรมหลักสามารถอธิบายได้ดังนี้

เมื่อระบบเริ่มทำงานจะทำการสวิตซ์เลือกรับค่าระหว่างรับค่าอินพุตจากคอมพิวเตอร์หรือเลือกรับค่าจากคีย์แพด หากรับค่าอินพุตจากคอมพิวเตอร์โปรแกรมจะสั่งให้ไมโครคอนโทรลเลอร์ (AT89C52) ทำการเปิดการอินเตอร์รัพต์พอร์ตอนุกรมทั้งหมดและตั้งค่า บอร์ดเรต เท่ากับ 9600 บิตต่อวินาที สำหรับการสื่อสารพอร์ตอนุกรม จากนั้นไมโครคอนโทรลเลอร์จะรอรับค่าอินพุตจากคอมพิวเตอร์ (ค่า z) เมื่อได้รับค่า z ไมโครคอนโทรเลอร์ก็จะประมวลผลและทำการแสดงรูปแบบการกระพริบของหลอดไฟในโหมด Random Mode , Manual Mode หรือ Beat Detection Mode

แต่ถ้าหากเลือกรับค่าจากคีย์แพดโปรแกรมจะสั่งให้ไมโครคอนโทรลเลอร์ (AT89C4051) ทำการเปิดการอินเตอร์รัพต์พอร์ตอนุกรมทั้งหมดและตั้งค่า บอร์ดเรต เท่ากับ 9600 บิตต่อวินาที สำหรับการสื่อสารพอร์ตอนุกรม จากนั้นจะตั้งค่าพอร์ต P0=0 เพื่อทำให้ไม่มีการติดของหลอดไฟเมื่อระบบเริ่มทำงานจากนั้นจะทำการเริ่มแสกนบิตโดยจะเริ่มแสกนทีละคอลัมน์ เริ่มจากระบบจะสั่งให้บิตแสกนทุกบิตมีลอจิก 1 และคอลัมน์ 1 มีลอจิก 0 เพื่อทำการแสกนเฉพาะในคอลัมน์ 1 ก่อน จากนั้นไมโครคอนโทรลเลอร์จะรอรับค่าจากการกดคีย์แพด หากมีการกดคีย์ใดคีย์หนึ่งในคอลัมน์ 1 ระบบก็จะประมวลผลรูปการกระพริบของหลอดไฟตามค่าคีย์ที่กด เช่น หากมีการกดคีย์ 4 หมายถึงคอลัมน์ 1 และแถวที่ 1 มีค่าลอจิกเป็น 0 ระบบก็จะทำการประมวลผลรูปแบบการกระพริบของหลอดไฟ 4 เมื่อเสร็จสิ้นการประมวลผลระบบจะทำการแสกนบิตในคอลัมน์1ต่อไปเรื่อยๆ จนครบทุกบิต จากนั้นระบบจะสั่งให้บิตแสกนทุกบิตมีลอจิก1 และคอลัมน์ 2 มีลอจิก 0 เพื่อทำการแสกนบิตในคอลัมน์ที่ 2 และในทำนองเดียวกันเมื่อทำเสร็จสิ้นการแสกนในคอลัมน์ที่ 2ระบบจะสั่งให้บิตแสกนทุกบิตมีลอจิก 1 และคอลัมน์ 3 มีลอจิก 0 เพื่อทำการแสกนบิตในคอลัมน์ที่ 3 เมื่อทำการแสกนบิตครบทุกคอลัมน์แล้วระบบก็จะกลับมาแสกนบิตเริ่มที่คอลัมน์1ใหม่ไปเรื่อยๆจากการทำงานของระบบที่ทำการแสกนบิตทีละคอลัมน์จะไม่มีปัญหาการ

ทำงานเมื่อทำการแสกนคอลัมน์หนึ่งอยู่แล้วทำการกดคีย์ที่อยู่คนละคอลัมน์กันเพราะระบบประมวลผลด้วยความเร็วสูงทำให้การตอบสนองต่อการกดคีย์แต่ละครั้งเป็นไปดังที่ต้องการ จากนั้นไมโครคอนโทรเลอร์ (AT89C4051) จะส่งสัญญาณควบคุมเป็นค่าอินพุตไปยังไมโครคอนโทรลเลอร์ (AT89C52) เพื่อเลือกรูปแบบการกระพริบของหลอดไฟต่อไป

3.1.3 การเขียนโปรแกรมเพื่อหน่วงเวลาของหลอดไฟ

สำหรับโปรแกรมภาษาซีที่ใช้ในการควบคุมรูปแบบไฟกระพริบบนไมโครคอนโทรลเลอร์ (AT89C52) นั้นจะมีการใช้คำสั่งหน่วงเวลาซึ่งเกี่ยวข้องกับการการคำนวณหาเวลาของสัญญาณนาฬิกา 1 ลูก (1 Cycle) สำหรับในการทดสอบนี้ใช้สัญญาณนาฬิกา XTAL ที่มีค่า 11.0592 MHz ดังนั้นเวลาของสัญญาณนาฬิกา 1 ลูก สามารถคำนวณได้ดังนี้

เวลาของสัญญาณนาฬิกา 1 ลูก = 1 / ความถี่สัญญาณนาฬิกา (XTAL)

$= 1 / 11.0592 \times 10^{6}$

$= 9.04225 \times 10^{6}$ วินาที

จากนั้นจะคำนวณหาค่าเวลาที่ใช้ไปใน 1 แมชชีนไซเคิล ซึ่งเวลาที่ใช้ใน 1 แมชชีนไซเคิลจะเท่ากับเวลาของสัญญาณนาฬิกา 12 ลูก จะได้

เวลาใน 1 แมชชีนไซเคิล $= 12 \times 9.04225 \times 10^{-8}$

$= 1.08507 \times 10^{-6}$ วินาที

สมมุติว่าต้องการเวลา 200 uS สามารถคำนวณได้ดังนี้

คำนวณหาค่ารีโหลดที่ 200 uS $= 200 \times 10^{-6} / 1.0857 \times 10^{-6}$

$= 184.3199 = 184$

เนื่องจากไทเมอร์นับค่าขึ้น ดังนั้นจะได้ค่ารีโหลด (8bit) คือ

$= 184 \times 1.08507 \times 10^{-6}$

$= 199.65228 \times 10^{-6}$ วินาที

$= 199.65228$ uS

โดยโปรแกรมบนไมโครคอนโทรลเลอร์จะมีฟังก์ชั่นการหน่วงเวลาของหลอดไฟแสดงดังนี้

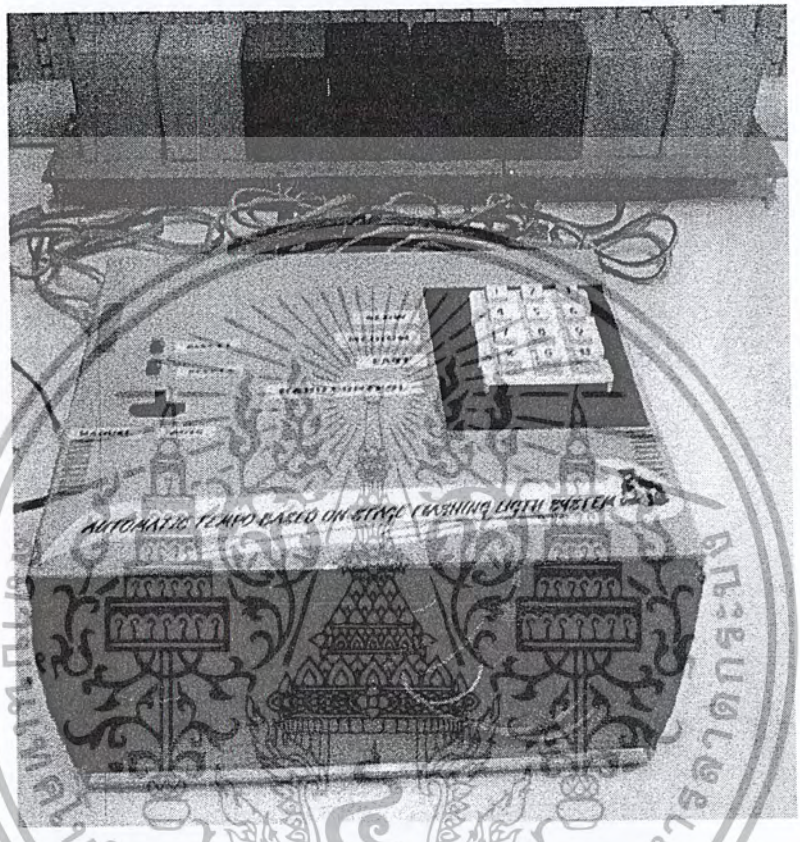
```
void Del_200uS(unsigned int DelLop)
{
   TR0=1;
   while(DelLop--)
   {
      while(!TF0);
      TF0=0;
   }
   TR0=0;
}
```

ฟังก์ชั่นหน่วงเวลาดังกล่าวจะใช้การหน่วงเวลาด้วยไทเมอร์ เพื่อให้เกิดการโอเวอร์โฟลวทุก 200 uS โดยเริ่มจากการกำหนดให้ Timer0 เริ่มทำงาน (TR0 = 1) จากนั้นวนลูป while.. จนกว่าตัวแปร DelLop มีค่าเท่ากับ 0 และซ้อนลูป while.. เพื่อให้เกิดการโอเวอร์โฟลว (TF0 = 1) เมื่อเกิดการโอเวอร์โฟลวก็จะทำการเคลียร์ TF0 และสั่งการให้ Timer0 หยุดทำงาน (TR0 = 0)

เมื่อต้องการจะเรียกใช้ฟังก์ชั่นหน่วงเวลาสามารถเรียกใช้ได้ด้วยคำสั่ง Del_200uS() ยกตัวอย่างเช่น หากต้องการให้เกิดการหน่วงเวลา 1 วินาทีจะต้องใช้คำสั่ง Del_200uS(5000) ค่าพารามิเตอร์เท่ากับ 5000 ซึ่งเป็นตัวคูณค่า 200 uS ให้ได้ค่าเวลาที่ 1 วินาที

3.1.4 ฮาร์ดแวร์และการเชื่อมต่อที่เสร็จสมบูรณ์

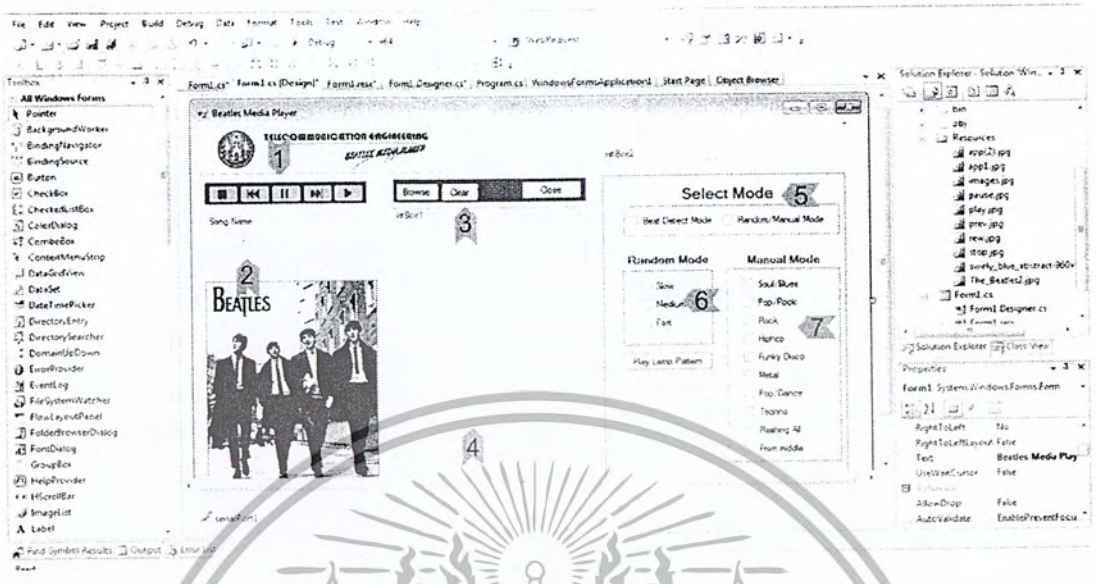
ชิ้นงานที่เสร็จสมบูรณ์แสดงดังรูปที่ 3.8



รูปที่ 3.8 ชิ้นงานที่เสร็จสมบูรณ์

3.1.5 การออกแบบแอพพลิเคชั่นบนคอมพิวเตอร์

การออกแบบแอพพลิเคชั่นบนคอมพิวเตอร์จะใช้โปรแกรม Visual Studio 2008 และใช้ภาษา C# ในการออกแบบแอพพลิเคชั่น แสดงการออกแบบแอพพลิเคชั่นด้วยโปรแกรม Visual Studio 2008 ได้ดังรูปที่ 3.9



รูปที่ 3.9 การออกแบบแอพพลิเคชั่นด้วยโปรแกรม Visual Studio 2008

แอพพลิเคชั่นที่ถูกเขียนโปรแกรมขึ้นเป็นลักษณะของ โปรแกรมเล่นเพลงซึ่งจะมีทูลบาร์ (Tool Bar) และฟังก์ชั่นในการใช้งานดังต่อไปนี้

1) Play Bar เป็นทูลบาร์สำหรับเล่นเพลง, หยุดเพลง, หยุดเพลงชั่วคราว และ ปุ่มเร่ง-ถอยหลัง

2) Song Name แสดงชื่อเพลงที่กำลังเล่นอยู่ในขณะนั้น

3) Queue Control เป็นทูลบาร์สำหรับเรียกเล่นเพลงจากไฟล์ในคอมพิวเตอร์ (Browse), ล้างคิวเพลงทั้งหมดใน Queue List (Clear) และ ปิดโปรแกรม (Close)

4) Queue List หน้าต่างแสดงคิวเพลงที่เลือกเล่นเรียงตามลำดับ

5) Select Mode เลือกโหมดการทำงาน

- Beat Detection Mode ฟังก์ชั่นโหมดการกระพริบตามจังหวะเสียงดนตรี

- Random/Manual Mode ฟังก์ชั่นโหมดการกระพริบแบบ Random และแบบ Manual จะสามารถใช้งานได้ (ลักษณะการใช้งานอธิบายดังข้อ 6 และ 7)

6) Random Mode แสดงการกระพริบตามระดับความเร็ว ได้แก่ สุ่มเลือกรูปแบบการกระพริบของระดับความเร็วเพลงช้า (Slow), ระดับเพลงกลาง, ระดับเพลงเร็ว

7) Manual Mode แสดงการกระพริบตามรูปแบบการกระพริบสำเร็จรูปของโปรแกรม

3.1.6 การเขียนโปรแกรมบนคอมพิวเตอร์เพื่อตรวจจับบีต

การออกแบบโปรแกรมในส่วนนี้เป็นการประยุกต์ใช้ทฤษฎีการตรวจจับจังหวะหรือบีตของเฟดริก พาติน (Federic Patin) ซึ่งได้อธิบายไว้ในหัวข้อ 2.1 เพื่อวิเคราะห์การตรวจจับบีตก่อนจะสั่งการไปยังระบบบันทึกรูปแบบไฟกระพริบให้กระพริบตามบีตของเพลง โดยใช้หลักการของเฟดริก พาติน หลังจากได้ทำการศึกษาวิธีการต่างๆจึงได้ทำการเลือกใช้กระบวนการจับบีตด้วยขั้นตอนวิธีการตรวจจับบีตในเชิงสถิติแบบแบ่งช่วงความถี่ ในหัวข้อ 2.1.4 ซึ่งมีขั้นตอนและวิธีการดังต่อไปนี้

3.1.6.1 คำนวณ FFT ของ 1024 แซมเปิลที่นำมาจากสัญญาณ $a[n]$ และ $b[n]$ เมื่อผ่านกระบวนการ FFT จะได้ค่าจำนวนเชิงซ้อน ซึ่งแบ่งออกเป็นส่วนจริง (Real part) จาก $a[n]$ และส่วนจินตภาพ (Imaginary Part) จาก $b[n]$ ดังสมการที่ 3.1 โดยการ FFT จะต้องคำนวณทั้ง 1024 แซมเปิล

$$(a_n) + i \times (b_n) \tag{3.1}$$

3.1.6.2 นำค่าที่ได้จาก FFT เก็บลงในบัฟเฟอร์ ดังนั้นบัฟเฟอร์ (B) จะเก็บค่าแอมพลิจูดของความถี่ไว้ทั้งหมด 1024 ค่า

3.1.6.3 แบ่งบัฟเฟอร์ออกเป็น 32 ซับแบนด์ ทำการคำนวณพลังงานในแต่ละซับแบนด์ และเก็บค่าพลังงานของแต่ละซับแบนด์ (E_s) ดังนั้น E_s จะมี 32 ขนาด และ $E_s[i]$ จะเก็บพลังงานของซับแบนด์ i เอาไว้ แสดงได้ดังสมการที่ 3.2

$$E_s[i] = \frac{32}{1024} \times \sum_{k=i\times 32}^{(i+1)\times 32} B[k] \tag{3.2}$$

3.1.6.4 จากนั้นคำนวณหาค่าพลังงานเฉลี่ยในแต่ละซับแบนด์ i (E_i) โดยใช้สมการที่ 3.3

$$< E_i >= \frac{1}{43} \times \sum_{k=0}^{42} Ei[k] \tag{3.3}$$

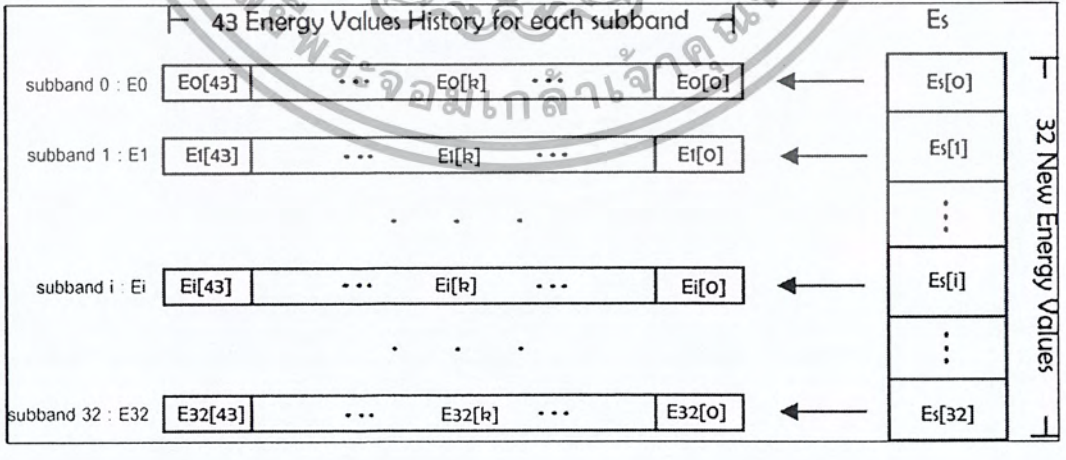
3.1.6.5 เลื่อนบัฟเฟอร์เก็บค่าไปทางขวาอีก 1 ช่องข้อมูล ทำให้มีช่องข้อมูลเพิ่มขึ้น 1 ช่องสำหรับการจัดเก็บค่าพลังงานเฉลี่ยใหม่ในแต่ละซับแบนด์และลบค่าพลังงานเฉลี่ยเก่าทิ้งออกไป

3.1.5.6 จัดเก็บค่าพลังงานเฉลี่ยของแต่ละซับแบนด์ใหม่ ลงใน $E_i[0]$ พิจารณาสมการที่ 3.4

$$E_i[0] = E_s[i] \tag{3.4}$$

3.1.6.7 หากในแต่ละซับแบนด์ i หาก $E_s[i] > (C \times < E_i >)$ ค่าพลังงานนั้นๆคือค่าบีต

สามารถแสดงการเก็บข้อมูลในแต่ละซับแบนด์ได้ดังรูปที่ 3.9



รูปที่ 3.10 แสดงการเก็บข้อมูลในแต่ละซับแบนด์

โดยสามารถคำนวณค่าความแปรปรวน (V) และค่า C ได้จากสมการดังต่อไปนี้

$$V = \frac{1}{43} \times \sum_{i=0}^{43} (E[i] - < E >)^2 \quad (3.5)$$

$$C = (-0.0025714 \times V) + 1.514287 \quad (3.6)$$

จากทฤษฎีที่กล่าวมาข้างต้นนำมาประยุกต์ใช้งานโดยเขียนโปรแกรมคำนวณค่าบีตที่เกิดขึ้นด้วยโปรแกรม Processing โดยผลการจับตรวจบีตด้วยโปรแกรมในสุดท้ายแล้วจะใช้กลุ่มเมทธอด(Methods) จากคลาส BeatDetect ทำการเลือกย่านความถี่เพื่อแสดงค่าการเกิดบีต กลุ่มเมทธอดเหล่านั้นได้แก่

1) isOnset( ) ใช้ในการจับบีตภายในเฟรมข้อมูลคล้ายกับวิธีการตรวจจับบีตในเชิงสถิติแบบที่ I

2) isKick( ) ใช้ในการจับบีตเมื่อเกิดบีตจากการเหยียบกระเดื่องกลอง

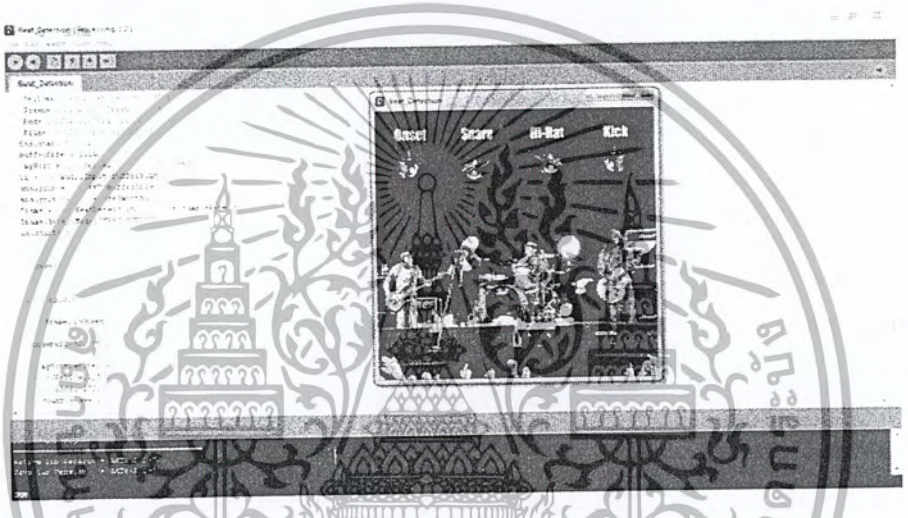
3) isSnare( ) ใช้ในการจับบีตเมื่อเกิดบีตจากการตีสแนร์กลอง

4) isHat( ) ใช้ในการจับบีตเมื่อเกิดบีตจากการตีไฮแฮทกลอง

ในการทำงานของโปรแกรมนั้นส่วนแรกต้องทำการสร้างสตรีมไฟล์ก่อนซึ่งเป็นการรับอินพุตข้อมูลสัญญาณเสียงที่ผ่านซาวการ์ด จากนั้นสร้างบัฟเฟอร์ เพื่อใช้เก็บข้อมูลในการคำนวณค่าพลังงาน หลังจากคำนวณค่าพลังงานต่างๆที่เกิดขึ้น สุดท้ายก้อจะใช้กลุ่มเมทธอดที่กล่าวมาแยกจับบีตตามความสามารถของแต่ละตัว

ดังนั้นหากต้องการใช้งานในส่วนของ Beat Detection Mode ของแอพพลิเคชั่นโปรแกรม Visual Studio 2008 ถูกเขียนโปรแกรมให้เรียกแอพพลิเคชั่นภาษาจาวาของโปรแกรม Processing ซึ่งเป็นไฟล์ .exe ออกมาใช้งาน โดยการเขียนโปรแกรมใน Processing นั้นเป็นการเขียนโปรแกรมภาษาจาวาที่เกี่ยวข้องกับการตรวจจับบีตโดยใช้วิธีการของขั้นตอนวิธีการตรวจจับบีตดังที่กล่าวมาข้างต้น เมื่อทำการเขียนโปรแกรมให้คอมพิวเตอร์ประมวลผลตามขั้นตอนวิธีการ

ต่างๆแล้ว โปรแกรม Processing จะสามารถตรวจจับบีตได้ตามหลักการ และ โปรแกรมที่เขียนด้วย Visual Studio 2008 จะทำการเรียกใช้การประมวลผลของ Processing (ในกรณีที่ผู้ใช้งานเลือกใช้งาน Beat Detection Mode หรือฟังก์ชั่นโหมดการกระพริบตามจังหวะเสียงดนตรี) เพื่อส่งสัญญาณควบคุมผ่านพอร์ตอนุกรม (มาตรฐาน RS-232) ไปยังไมโครคอนโทรลเลอร์ (AT89C52) ก่อนจะส่งสัญญาณควบคุมวงจรควบคุมหลอดไฟให้กระพริบตามจังหวะบีตที่ประมวลผลได้ โดยหน้าต่างโปรแกรม Processing ขณะประมวลผลแสดงได้ดังรูปที่ 3.11



รูปที่ 3.11 หน้าต่างโปรแกรม Processing ขณะทำการประมวลผล

3.2 เครื่องมือที่ใช้ในการทดลอง

3.2.1) ออสซิลโลสโคป (Oscilloscope)

3.2.2) แหล่งจ่ายไฟ (Power Supply)

3.2.3) โวลต์มิเตอร์ (Volt Meter)

3.2.4) อแดปเตอร์ 5 โวลต์ , 12 โวลต์ (Adapter 5V, 12V)

3.2.5) คอมพิวเตอร์พกพา

3.3 การจัดเก็บผลการทดลอง

การจัดเก็บผลการทดลองแบ่งเป็น 3 ส่วนคือ ส่วนที่เป็นผลการทดลองของระบบจัดเก็บรูปแบบการกระพริบของหลอดไฟและชุดควบคุมการกระพริบของหลอดไฟ ซึ่งในส่วนนี้จะแบ่งผลการทดลองเป็นสองส่วนย่อย ได้แก่ การวัดผลสัญญาณในระดับลอจิกโดยใช้ออสซิลโลสโคปวัดสัญญาณลอจิกที่ออกจากพอร์ต 0 ของไมโครคอนโทรลเลอร์จะเป็นการแสดงตัวอย่างผลรูปแบบของไฟกระพริบ 3 รูปแบบ และการวัดผลรูปแบบการกระพริบของหลอดไฟที่ถูกต่ออยู่กับวงจรควบคุมหลอดไฟ โดยจะแสดงผลเป็นภาพถ่ายรูปแบบไฟกระพริบทั้งหมด 3 รูปแบบเพื่อพิจารณาประกอบกับผลที่ได้จากออสซิลโลสโคป โดยใช้กล้องดิจิตอลในการเก็บผลการทดลอง

ส่วนที่สองจะเป็นผลการทดลองการเขียนแอพพลิเคชั่น Beatles Music Player ซึ่งก็คือแอพพลิเคชั่นที่เขียนด้วยโปรแกรม Visual Studio 2008 เพื่อควบคุมการเล่นเพลงและมีโหมดการควบคุมหลอดไฟให้เลือกใช้งาน

ในส่วนสุดท้ายจะเป็นผลการทดลองการบันทึกจากการตรวจจับบีตของโปรแกรมในส่วนที่เป็นการตรวจจับบีตโดยใช้วิธีการวัดผลโดยการเปรียบเทียบค่าบีตต่อนาทีมาตรฐานที่ได้จากสัญญาณเสียงเมโทรนอม (Metronome) ผสมสัญญาณเสียงรบกวน กับค่าบีตต่อนาทีเฉลี่ยที่ได้จากการเก็บค่าด้วยเมทอด isOnset, isSnare() ซึ่งมีความสามารถในการจับบีตในย่านความถี่ของอุปกรณ์กลองเหตุผลที่ต้องใช้สองเมทธอดนี้เพราะทั้งสองเมทธอดนี้สามารถตรวจจับบีตในย่านของเสียงเมโทรนอมได้ดี และเพียงพอที่จะเปรียบเทียบผลความแม่นยำของการจับบีตของโปรแกรมได้ทั้งหมด โดยค่าบีตต่อนาทีมาตรฐานที่ได้จากเมโทรนอมนั้นสามารถใช้โปรแกรม Guitar Pro 5 ซึ่งมีฟังก์ชั่นเล่นเสียงเมโทรนอมหรือเสียงบีตต่อนาทีได้

บทที่ 4

ผลการทดลอง

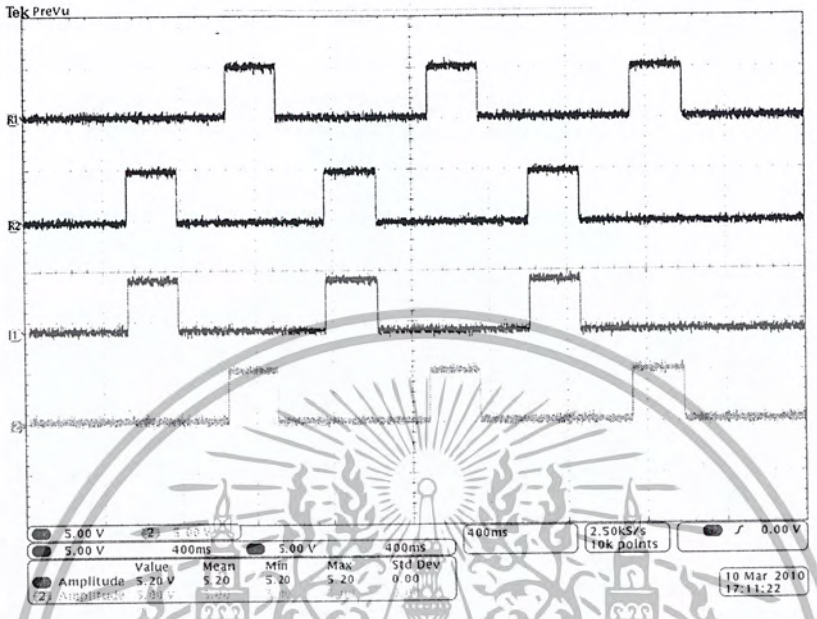
ผลการทดลองแบ่งเป็น 3 ส่วนคือส่วนที่เป็นผลการทดลองของระบบจัดเก็บรูปแบบการกระพริบของหลอดไฟและชุดควบคุมการกระพริบของหลอดไฟ ซึ่งในส่วนนี้จะแบ่งผลการทดลองเป็นสองส่วนย่อย ได้แก่ การวัดผลสัญญาณในระดับลอจิกโดยใช้ออสซิลโลสโคปวัดสัญญาณลอจิกที่ออกจากพอร์ต 0 ของไมโครคอนโทรลเลอร์จะเป็นการแสดงผลรูปแบบของไฟกระพริบทั้งหมด 3 รูปแบบ และส่วนที่เป็นการวัดผลรูปแบบการกระพริบของหลอดไฟที่ถูกต่ออยู่กับวงจรควบคุมหลอดไฟ โดยจะแสดงผลเป็นภาพถ่ายรูปแบบไฟกระพริบทั้ง 3 รูปแบบโดยใช้กล้องดิจิตอลในการเก็บผลการทดลอง ผลการทดลองในส่วนที่สองจะเป็นผลการทดลองการเขียนแอพพลิเคชั่น Beatles Music Player ซึ่งก็คือแอพพลิเคชั่นที่เขียนด้วยโปรแกรม Visual Studio 2008 เพื่อควบคุมการเล่นเพลงและมีโหมดการควบคุมหลอดไฟให้เลือกใช้งาน และผลการทดลองในส่วนสุดท้ายจะเป็นผลของโปรแกรมตรวจจับบีตของเพลงที่เล่นด้วยโปรแกรมที่สร้างขึ้นบนคอมพิวเตอร์ ผลการทดลองจะแสดงผลการตรวจจับบีตในหนึ่งนาที(BPM) ของโปรแกรมที่สร้างขึ้นเปรียบเทียบกับค่าบีตต่อนาทีมาตรฐานที่ได้จากเมโทรนอมโดยใช้โปรแกรม Guitar Pro 5 เป็นโปรแกรมสร้างสัญญาณเสียงเมโทรนอมและกำหนดค่าบีตต่อนาทีมาตรฐาน

4.1 ผลการทดลองของระบบจัดเก็บรูปแบบการกระพริบของหลอดไฟและชุดควบคุมการกระพริบของหลอดไฟ

4.1.1 ผลการทดลองโดยการวัดผลสัญญาณระดับลอจิก

ในส่วนนี้จะทำการวัดสัญญาณลอจิกที่ออกจากพอร์ต 0 ของไมโครคอนโทรเลอร์ด้วยออสซิลโลสโคป จะมีสัญญาณทั้งหมด 4 สัญญาณที่มีลอจิกเหมือนหรือต่างกันไปตามรูปแบบการกระพริบของหลอดไฟ ประกอบด้วยสัญญาณอ้างอิง R1, R2, 1 และสัญญาณอ้างอิง 2 ซึ่งก็คือระดับสัญญาณที่ออกจากพอร์ต P0.0 ถึงพอร์ต P0.3 ตามลำดับ ซึ่งจะอธิบายเป็นตัวอย่างผลการทดลองเนื่องจากมีรูปแบบไฟกระพริบเป็นจำนวนมาก ซึ่งผลการทดลองแสดงได้ดังต่อไปนี้

4.1.1.1 ผลการทดลองรูปแบบการกระพริบ ตัวอย่างที่ 1



รูปที่ 4.1 สัญญาณที่วัดจากพอร์ต P0.0 ถึงพอร์ต P0.3 ของรูปแบบการกระพริบ ตัวอย่างที่ 1

จากรูปที่ 4.1 ทำการเปรียบเทียบผลที่ได้กับค่าที่กำหนดในการออกแบบซอฟต์แวร์ โดยลอจิก "0" คือระดับสัญญาณ 0 โวลต์ และลอจิก "1" คือระดับสัญญาณประมาณ 5 โวลต์

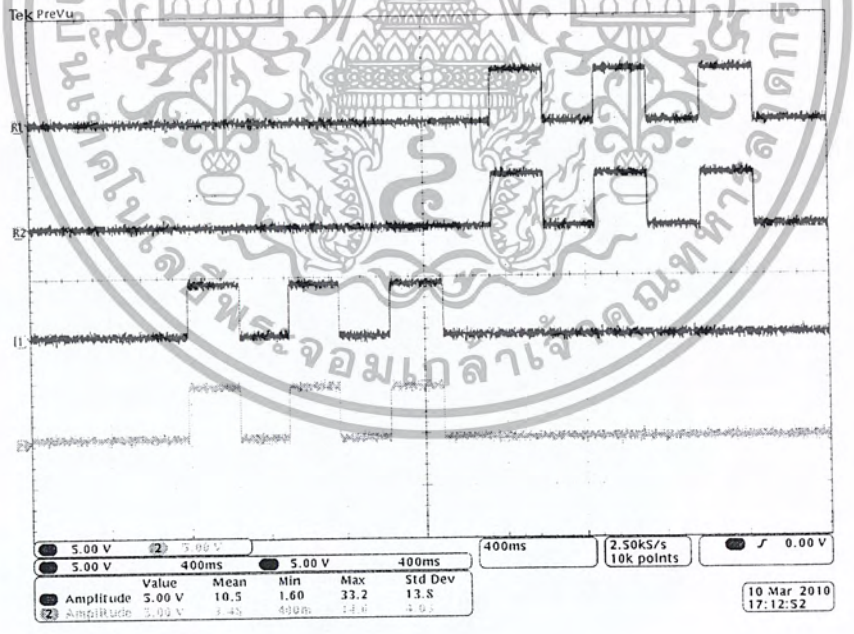
ตารางที่ 4.1 ตารางการเปรียบเทียบผลที่ได้กับค่าที่กำหนดในการออกแบบซอฟต์แวร์ (ตัวอย่างที่ 1)

ค่าที่กำหนดบนซอร์ฟแวร์แสดงเป็นเลขฐาน 16 (พิจารณาเฉพาะ 4 บิตล่าง)	สัญญาณลอจิกที่อ่านจากออสซิลโลสโคป เริ่มนับจากสัญญาณพัลส์ลูกแรก			
	P0.3	P0.2	P0.1	P0.0
06	0	1	1	0
00	0	0	0	0
09	1	0	0	1
00	0	0	0	0
06	0	1	1	0
00	0	0	0	0

ตารางที่ 4.1 ตารางการเปรียบเทียบผลที่ได้กับค่าที่กำหนดในการออกแบบซอฟต์แวร์ (ตัวอย่างที่ 1) (ต่อ)

ค่าที่กำหนดบนซอร์ฟแวร์แสดงเป็นเลขฐาน 16 (พิจารณาเฉพาะ 4 บิตล่าง)	สัญญาณลอจิกที่อ่านจากออสซิลโลสโคป เริ่มนับจากสัญญาณพัลส์ลูกแรก			
	P0.3	P0.2	P0.1	P0.0
09	1	0	0	1
00	0	0	0	0
06	0	1	1	0
00	0	0	0	0
09	1	0	0	1
00	0	1	1	0

4.1.1.2 ผลการทดลองรูปแบบการกระพริบ ตัวอย่างที่ 2



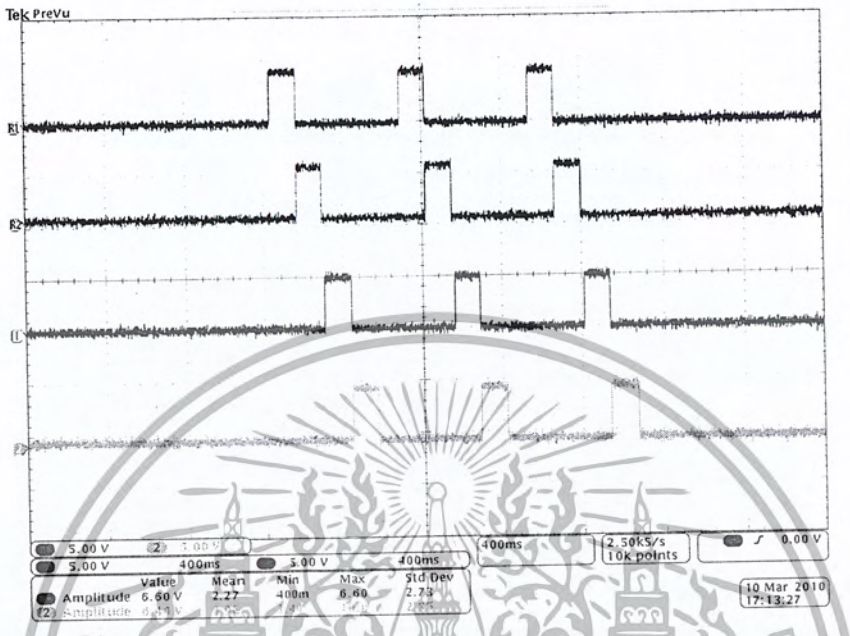
รูป 4.2 สัญญาณที่วัดจากพอร์ต P0.0 ถึงพอร์ต P0.3 รูปแบบการกระพริบ ตัวอย่างที่ 2

จากรูปที่ 4.2 ทำการเปรียบเทียบผลที่ได้กับค่าที่กำหนดในการออกแบบซอฟต์แวร์ โดยลอจิก “0” คือระดับสัญญาณ 0 โวลต์ และลอจิก “1” คือระดับสัญญาณประมาณ 5 โวลต์

ตารางที่ 4.2 ตารางการเปรียบเทียบผลที่ได้กับค่าที่กำหนดในการออกแบบซอฟต์แวร์ (ตัวอย่างที่ 2)

ค่าที่กำหนดบนซอร์ฟแวร์แสดงเป็นเลขฐาน 16 (พิจารณาเฉพาะ 4 บิตล่าง)	**สัญญาณลอจิกที่อ่านจากออสซิลโลสโคป เริ่มนับจากสัญญาณพัลส์ลูกแรก**			
	P0.3	**P0.2**	**P0.1**	**P0.0**
0C	1	1	0	0
00	0	0	0	0
0C	1	1	0	0
00	0	0	0	0
0C	1	1	0	0
00	0	0	0	0
03	0	0	1	1
00	0	0	0	0
03	0	0	1	1
00	0	0	0	0
03	0	0	1	1
00	0	0	0	0

4.1.1.5 ผลการทดลองรูปแบบการกระพริบตัวอย่างที่ 3



รูป 4.3 สัญญาณที่วัดจากพอร์ต P0.0 ถึงพอร์ต P0.3 รูปแบบการกระพริบ ตัวอย่างที่ 3

จากรูปที่ 4.3 ทำการเปรียบเทียบผลที่ได้กับค่าที่กำหนดในการออกแบบซอฟต์แวร์ โดยลอจิก “0” คือระดับสัญญาณ 0 โวลต์ และลอจิก “1” คือระดับสัญญาณประมาณ 5 โวลต์

ตารางที่ 4.3 ตารางการเปรียบเทียบผลที่ได้กับค่าที่กำหนดในการออกแบบซอฟต์แวร์ ตัวอย่างที่ 3

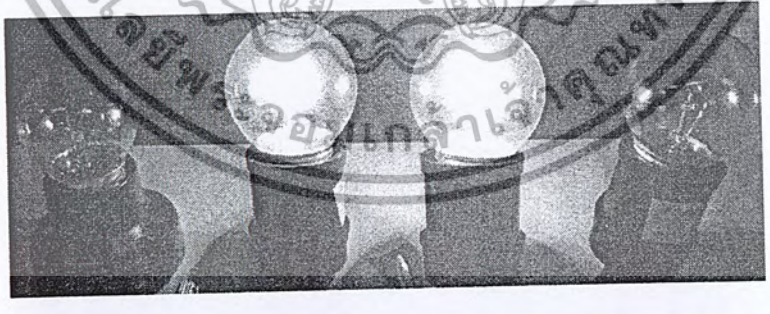
ค่าที่กำหนดบนซอร์ฟแวร์แสดงเป็นเลขฐาน 16 (พิจารณาเฉพาะ 4 บิตล่าง)	**สัญญาณลอจิกที่อ่านจากออสซิลโลสโคป เริ่มนับจากสัญญาณพัลส์ลูกแรก**			
	P0.3	**P0.2**	**P0.1**	**P0.0**
01	0	0	0	1
02	0	0	1	0
04	0	1	0	0
08	1	0	0	0
00	0	0	0	0
01	0	0	0	1
02	0	0	1	0

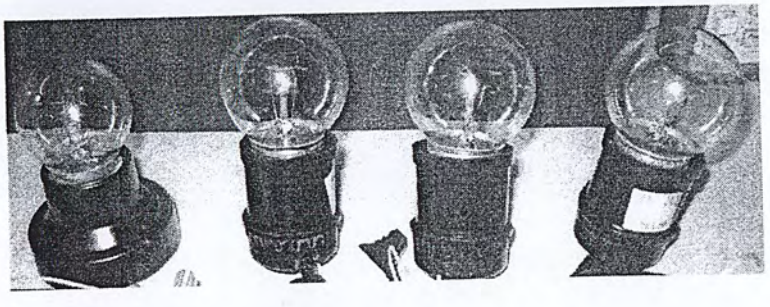
ตารางที่ 4.3 ตารางการเปรียบเทียบผลที่ได้กับค่าที่กำหนดในการออกแบบซอฟต์แวร์ ตัวอย่างที่ 3 (ต่อ)

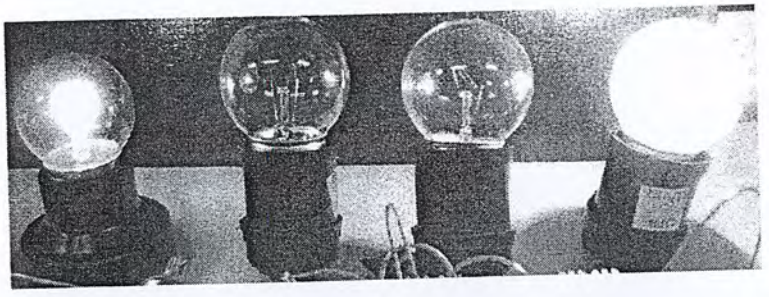
ค่าที่กำหนดบนซอร์ฟแวร์แสดงเป็นเลขฐาน 16 (พิจารณาเฉพาะ 4 บิตล่าง)	สัญญาณลอจิกที่อ่านจากออสซิลโลสโคป เริ่มนับจากสัญญาณพัลส์ลูกแรก			
	P0.3	P0.2	P0.1	P0.0
04	0	1	0	0
08	1	0	0	0
00	0	0	0	0
01	0	0	0	1
02	0	0	1	0
04	0	1	0	0
08	1	0	0	0
00	0	0	0	0

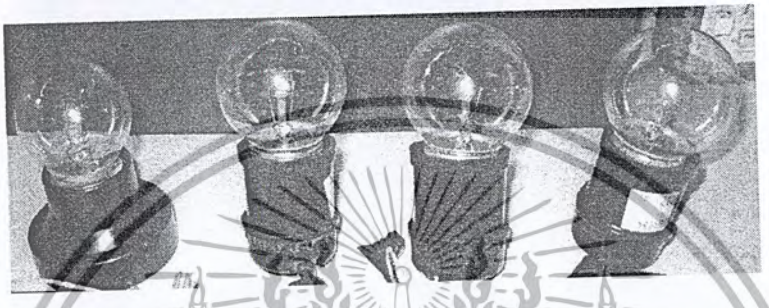
4.1.2 ผลการทดลองโดยวัดผลรูปแบบการกระพริบของหลอดไฟแสดงเป็นภาพถ่าย

ในส่วนนี้เป็นการแสดงผลรูปแบบการกระพริบของหลอดไฟเมื่อกดคีย์ต่างๆบนคีย์แพด ผลการทดลองจะแสดงเป็นภาพถ่ายดังต่อไปนี้

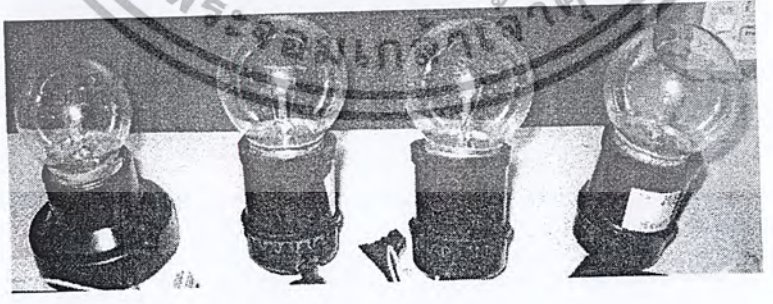


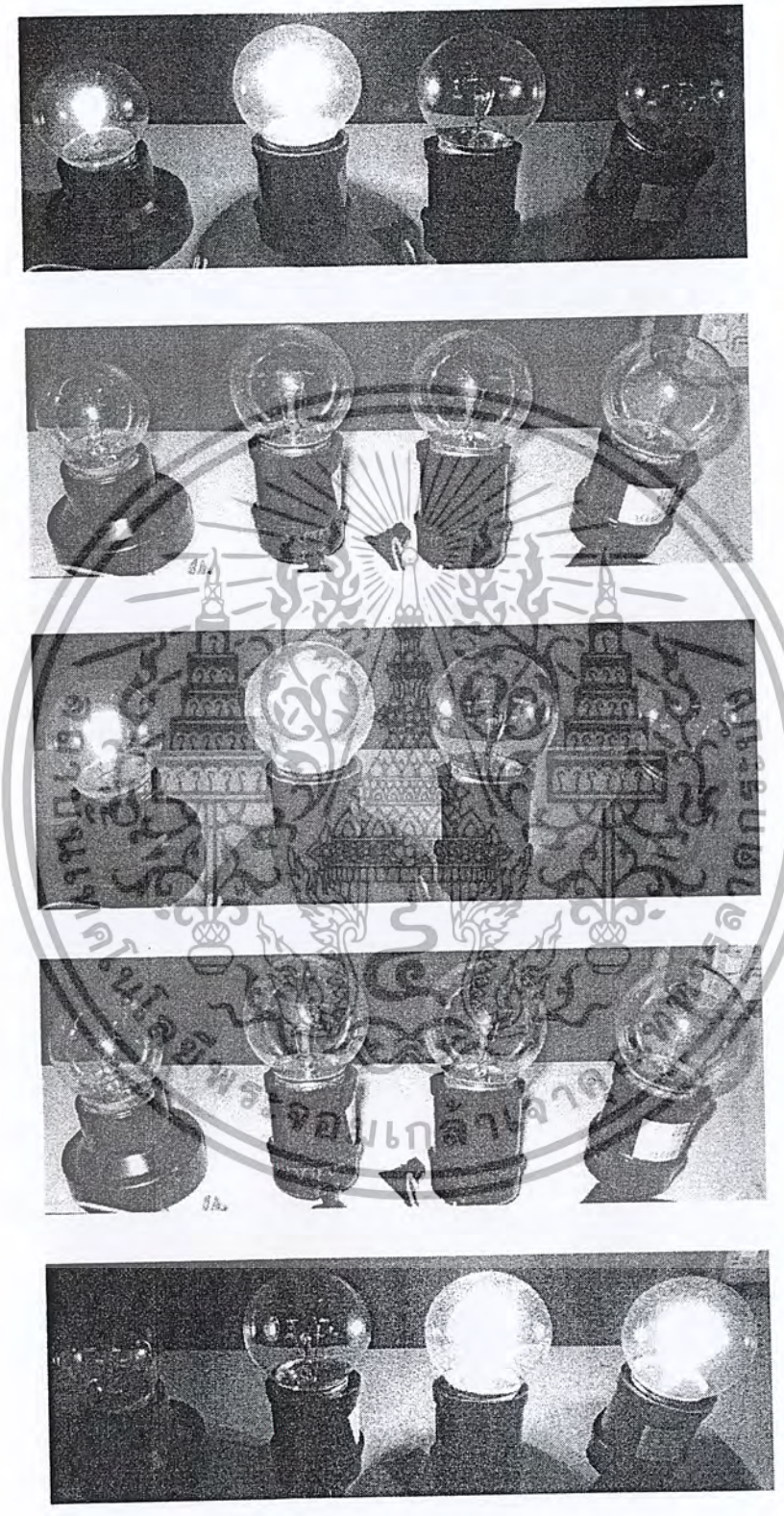




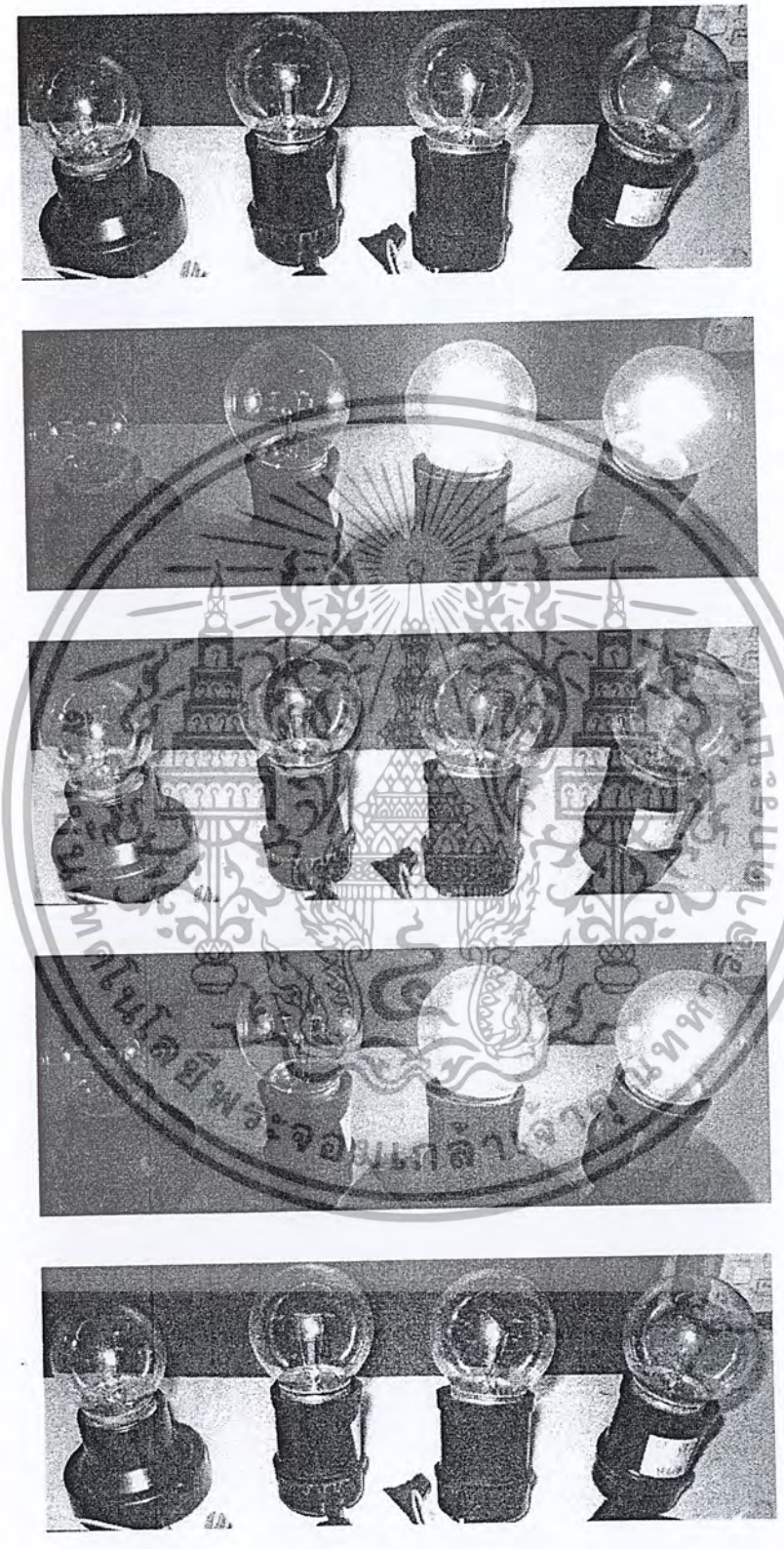


รูปที่ 4.4 รูปแบบการกระพริบของหลอดไฟ ตัวอย่างที่ 1

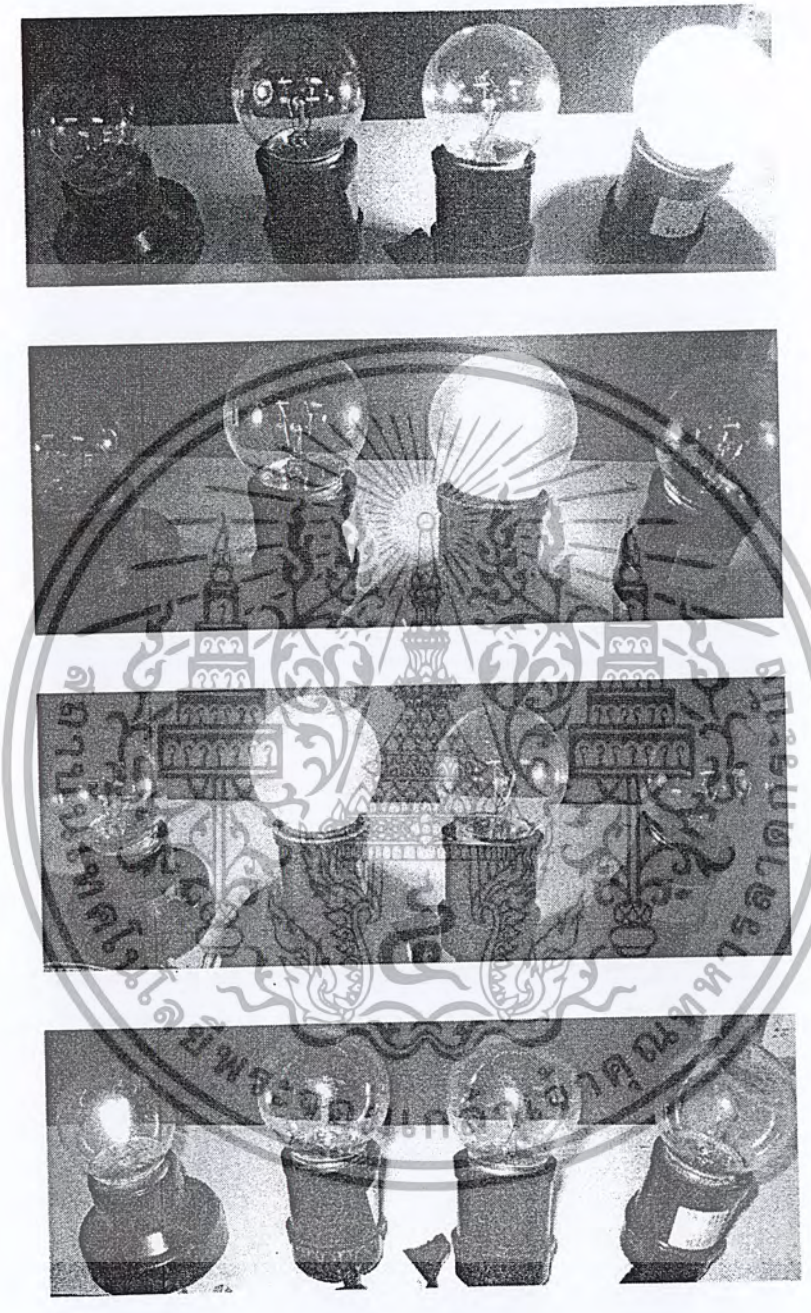




รูปที่ 4.5 รูปแบบการกระพริบของหลอดไฟ ตัวอย่างที่ 2



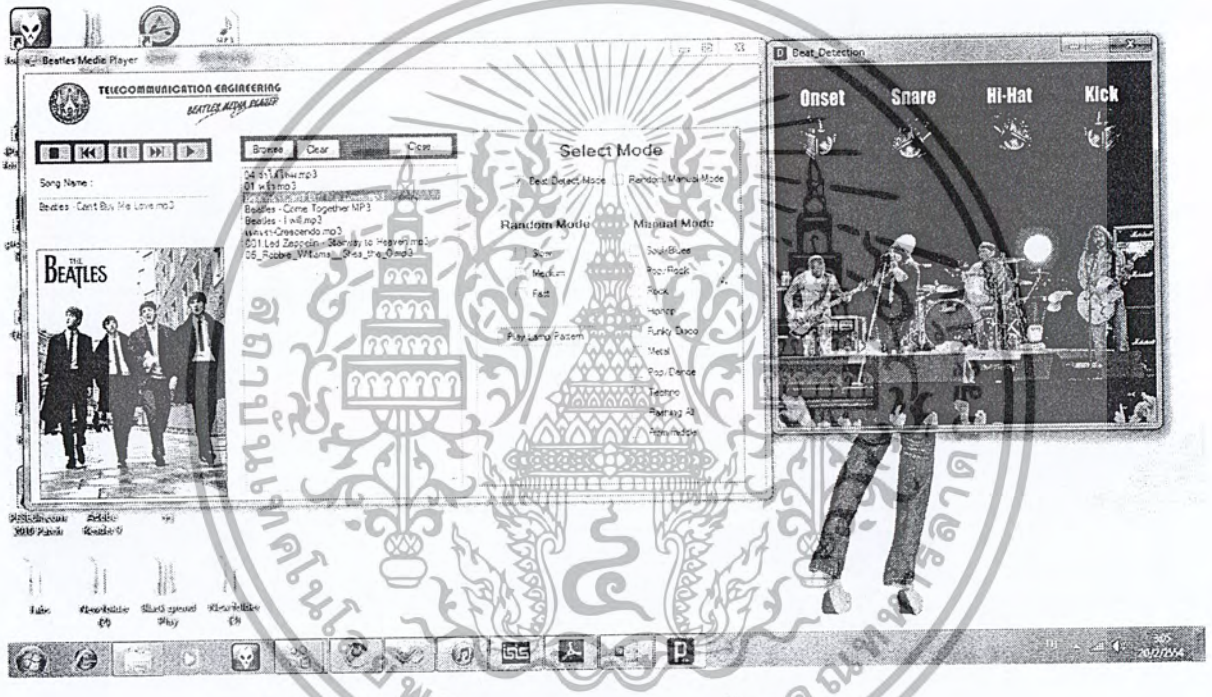
รูปที่ 4.6 รูปแบบการกระพริบของหลอดไฟ ตัวอย่างที่ 2 (2)



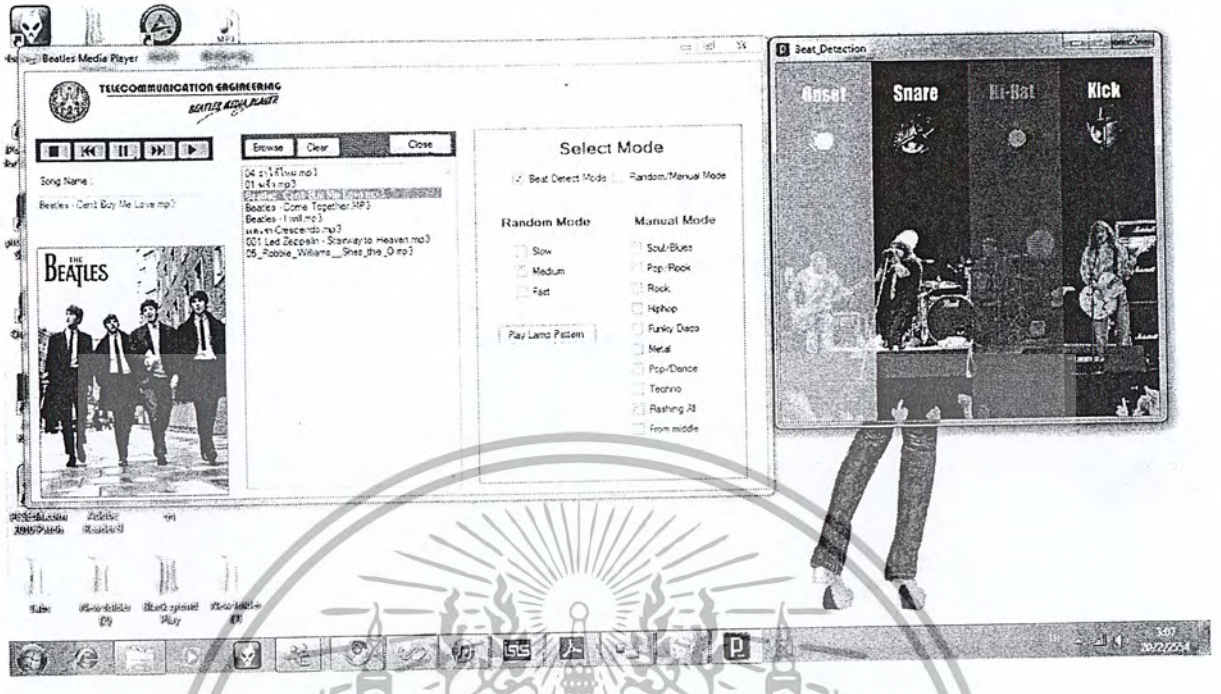
รูปที่ 4.7 รูปแบบการกระพริบของหลอดไฟ ตัวอย่างที่ 3

4.2 ผลการเขียนแอพพลิเคชั่น Beatles Music Player

ผลการทดลองการเขียนแอพพลิเคชั่นบนโปรแกรม Visual Studio 2008 จะแสดงรูปลักษณะของโปรแกรมที่ใช้ในการเล่นเพลงและเลือกโหมดการกระพริบของหลอดไฟ สามารถศึกษาการใช้งานได้ในหัวข้อ 3.1.5 และศึกษาข้อมูลพื้นฐานของโปรแกรม Visual Studio 2008 ได้ในหัวข้อ 2.10 ผลการเขียนแอพพลิเคชั่น Beatles Music Player แสดงผลเป็นรูปถ่ายได้ดังนี้



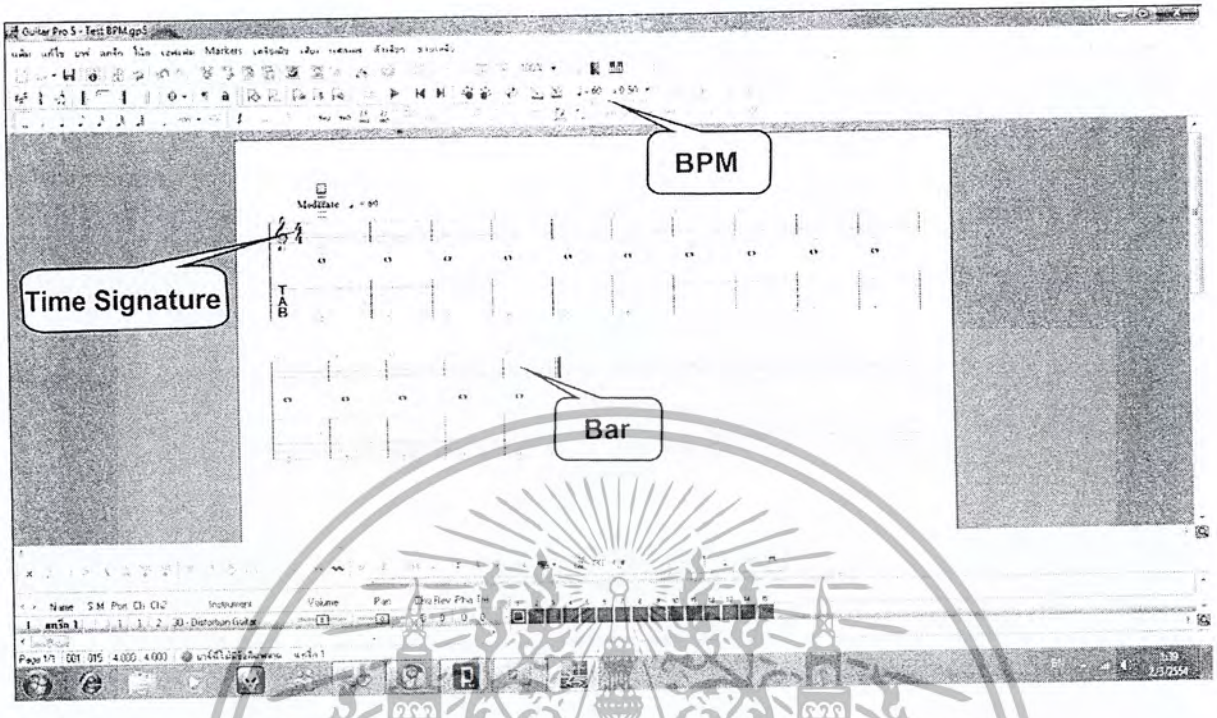
รูปที่ 4.8 หน้าต่างแอพพลิเคชั่น Beatles Music Player



รูปที่ 4.9 หน้าต่างแอพพลิเคชั่น Beatles Music Player (2)

4.3 ผลการทดลองการบันทึกจากการตรวจจับบีต

ผลการทดลองการบันทึกจากการตรวจจับบีตใช้วิธีการวัดผลโดยการเปรียบเทียบค่าบีตต่อนาทีมาตรฐานที่ได้จากสัญญาณเสียงเมโทรนอมปนสัญญาณเสียงรบกวน กับค่าบีตต่อนาทีเฉลี่ยที่ได้จากการเก็บค่าด้วยเมททอด isOnset, isSnare() ซึ่งมีความสามารถในการจับบีตในช่องความถี่เสียงเมโทรนอมโดยค่าบีตต่อนาทีมาตรฐานที่ได้จากเมโทรนอมนั้นสามารถใช้โปรแกรม Guitar Pro 5 ซึ่งมีฟังก์ชั่นเล่นเสียงเมโทรนอมหรือเสียงบีตต่อนาที เป็นค่าบีตต่อนาทีมาตรฐานเพื่อใช้เปรียบเทียบ แสดงหน้าต่างโปรแกรม Guitar Pro 5 ได้ดังรูปที่ 4.8



รูปที่ 4.10 หน้าต่างโปรแกรม Guitar Pro 5

จากรูปที่ 4.10 โปรแกรม Guitar Pro 5 สามารถเลือกความเร็วของเมโทรนอมได้ ซึ่งหน่วยความเร็วของเมโทรนอมคือ บีตต่อนาที (BPM) สามารถคำนวณได้โดยพิจารณาจากอัตรากำหนดจังหวะ (Time Signature) ในที่นี้ใช้อัตรากำหนดจังหวะประเภท 4/4 จะมีห้องทางดนตรี (Measure) ที่ถูกแบ่งออกแต่ละห้องหรือแต่ละบาร์ ในแต่ละห้องจะมีจังหวะเมโทรนอมตามอัตรากำหนดจังหวะ ดังนั้นด้วยอัตรากำหนดจังหวะประเภท 4/4 หมายถึงมีเสียงเมโทรนอมทั้งหมด 4 ครั้ง ต่อ 1 ห้อง หากต้องการความเร็วเมโทรนอมที่ 60 ครั้งต่อนาที หรือมีความเร็วที่ 60 บีตต่อนาที (1 ครั้งต่อ 1 วินาที) จำนวนห้องทางดนตรีจึงมีทั้งหมด 15 ห้อง

จากนั้นจะทำการวัดผลการทดลองโดยใช้ค่าบีตต่อนาทีมาตรฐานที่ได้จากสัญญาณเสียงเมโทรนอมรวมกับสัญญาณรบกวน โดยการเปิดเสียงแตก (Distortion) ของกีตาร์เล่นควบคู่ไปกับสัญญาณเสียงของเมโทรนอม จากนั้นนำไปเปรียบเทียบกับกับค่าบีตต่อนาทีเฉลี่ยที่ได้จากการเก็บค่าด้วยเมททอด isOnset, isSnare() โดยค่าบีตต่อนาทีมาตรฐานที่ต้องการนั้นจะทำการทดสอบที่ความเร็ว 60, 92, 120, 152, 180 บีตต่อนาที จากนั้นทำการเปรียบเทียบกับเมททอด isOnset, isSnare() ทั้งหมด 3 ครั้งเพื่อหาค่าบีตต่อนาทีเฉลี่ย (BPM_{av}) แสดงได้ดังตารางที่ 4.4 และ 4.5

ตารางที่ 4.4 ตารางบันทึกค่าการเกิดบีตต่อนาทีเปรียบเทียบจากค่ามาตรฐานกับค่าเฉลี่ยจากการตรวจจับด้วยเมททอด isOnset()

บีตต่อนาที (BPM)	isOnset()			
มาตรฐาน	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	บีตต่อนาทีเฉลี่ย(BPM_{av})
60	61	62	63	62.00
92	94	93	93	93.33
120	121	125	121	12233
152	129	142	131	134.33
180	133	132	132	132.33

ตารางที่ 4.5 ตารางบันทึกค่าการเกิดบีตต่อนาทีเปรียบเทียบจากค่ามาตรฐานกับค่าเฉลี่ยจากการตรวจจับด้วยเมททอด isSnare()

บีตต่อนาที (BPM)	isSnare()			
มาตรฐาน	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	บีตต่อนาทีเฉลี่ย(BPM)
60	61	60	60	60.33
92	92	93	91	92.00
120	125	123	120	122.67
152	139	148	139	142.00
180	142	160	144	148.67

จากตารางที่ 4.4, 4.5 สามารถสรุปผลการทดลองได้ว่า ในช่วงความเร็วที่ 60, 92, 120 บีตต่อนาทีมาตรฐานนั้น ค่าบีตต่อนาทีเฉลี่ยยังคงมีค่าที่ใกล้เคียง แต่ในช่วงความเร็วที่ 152, 180 บีตต่อนาทีมาตรฐาน ค่าบีตต่อนาทีเฉลี่ยจะเริ่มมีค่าผิดพลาดมากขึ้น จึงสรุปได้ว่าเมื่อบีตต่อนาทีมีค่ามากขึ้น หรือเมื่อเพลงเร็วขึ้น ค่าบีตต่อนาทีเฉลี่ยที่ตรวจจับได้จะมีค่าผิดพลาดมากขึ้นแต่ตามหลักการทำงานนั้นจะสามารถใช้

เมททอดหลายตัวในขั้นตอนสุดท้ายของการส่งค่าการเกิดบีตจึงสามรถทดแทนการทำงานกันได้เมื่อเพลงมีความเร็วหรือช้ามากๆ

บทที่ 5

สรุปผลและข้อเสนอแนะ

5.1 สรุปผล

จากการออกแบบและสร้างระบบไฟสถานบันเทิงแบบอัตโนมัติตามจังหวะเสียงดนตรีที่ได้กล่าวมาในบทก่อนหน้านี้ทั้งหมด สามารถสรุปผลการทดลองได้ดังต่อไปนี้

5.1.1 สรุปผลการทดลองการออกแบบระบบบันทึกรูปแบบการกระพริบและควบคุมการกระพริบของหลอดไฟ

ผลการทดลองได้ผลการทำงานของระบบบันทึกรูปแบบการกระพริบของหลอดไฟตรงตามเป้าหมายที่วางไว้กล่าวคือ ระบบสามารถสั่งการให้หลอดไฟกระพริบตามโปรแกรมที่กำหนดได้และบันทึกรูปแบบการกระพริบของหลอดไฟไว้ในไมโครคอนโทรลเลอร์เพื่อรอการสั่งงานจากในส่วนโปรแกรมบนคอมพิวเตอร์หรือจากคีย์แพดได้ จากการทดสอบการทำงานจริงเมื่อทำการเปลี่ยนโหมดการรับค่าอินพุตจากโปรแกรมบนคอมพิวเตอร์หรือจากคีย์แพด การทำงานเป็นไปอย่างราบรื่น กล่าวคือการกระพริบของหลอดไฟสามารถตอบสนองได้อย่างทันทีทันใดเมื่อทำการเปลี่ยนโหมดเหตุผลเนื่องมาจากการออกแบบการเขียนโปรแกรมบนไมโครคอนโทรลเลอร์(AT89C52)ในส่วนของการรับข้อมูลเป็นแบบอินเตอร์รัพต์พอร์ตอนุกรมทำให้สามารถรับค่าอินพุตได้อย่างต่อเนื่องและตอบสนองต่อค่าอินพุตที่เป็นปัจจุบัน

5.1.2 สรุปผลการทดลองการออกแบบโปรแกรม Beatles Music Player และโปรแกรมตรวจจับจังหวะหรือบีต

ผลการทดลองการทำงานของโปรแกรม Beatles Music Player หรือโปรแกรมที่ใช้ในการเล่นเพลงและมีโหมดการกระพริบของหลอดไฟให้เลือกใช้งานตามที่ได้กล่าวไว้ในบทที่ 3 โปรแกรมสามารถทำการเรียกเพลงจากไฟล์ในคอมพิวเตอร์เพื่อเล่นเพลงได้ตามที่กำหนด ในส่วนโหมดการกระพริบของหลอดไฟก็สามารถส่งสัญญาณไปควบคุมระบบ

บันทึกรูปแบบการกระพริบและควบคุมการกระพริบของหลอดไฟได้เช่นกัน และในส่วนสุดท้ายเป็นผลการทำงานของโปรแกรมตรวจจับจังหวะหรือบีต ผลการทำงานของโปรแกรมสามารถทำการตรวจจับบีตของเพลงได้ตามที่กำหนดและสามารถตอบสนองการทำงานเมื่อมีการเรียกใช้งานจากโปรแกรม Beatles Music Player ซึ่งเป็นโปรแกรมที่ใช้ในการเล่นเพลงได้

5.2 ข้อเสนอแนะ

การทำงานของโปรแกรมตรวจจับจังหวะหรือบีตของเพลงจะมีข้อผิดพลาดเกิดขึ้นบ้างเมื่อทำการใช้งานกับเพลงช้าหรือเพลงอคูสติก(Acoustic) เนื่องจากเพลงประเภทดังกล่าวมีค่าพลังงานที่ใกล้เคียงกันในแต่ละช่วงเวลากล่าวคือเป็นเพลงประเภทไม่มีการเน้นจังหวะหรือบีตของการเล่นมากนัก แนวทางแก้ไขคือการนำข้อมูลเสียงเพลงที่จะนำมาทดสอบไปผ่านการแยกเป็นซับแบนด์หลายๆซับแบนด์ตามความถี่ที่กำหนดให้มีจำนวนซับแบนด์มากขึ้น แต่ทั้งนี้ต้องคำนึงถึงการใช้ทรัพยากรที่มากขึ้นของหน่วยความจำบนคอมพิวเตอร์เพื่อให้การทำงานราบรื่นด้วย หรือประยุกต์ใช้ทฤษฎีอื่นๆดังที่กล่าวไว้ในหัวข้อที่ 2.1 ในบทที่ 2 ซึ่งอาจให้ผลการทำงานที่ดีกว่า คณะผู้จัดทำหวังว่าจะมีผู้ที่สนใจในปริญญานิพนธ์นี้จะนำการทดลองไปพัฒนาต่อไปและขอให้ประสบความสำเร็จอย่างที่ตั้งใจไว้

บรรณานุกรม

Eric D. Scheirer. *"Tempo and beat analysis of acoustic musical signals."* Beat-tracking acoustic signals , Cambridge Massachusetts , 1997

Casey Reas and Ben Fry, *Processing : A Programing Handbook for Visual Designers and Artists* , London England : Massachusetts Institute of Technology, 2007

รศ. สมยศ จุณณปิยะ, *Microcontroller Applications*, กรุงเทพฯ : ภาควิชาโทรคมนาคม สถาบันเทคโนโลยีเจ้าคุณทหารลาดกระบัง, 2550.

อุดม รานอก, *ภาษา C สำหรับงานควบคุม Microcontroller MCS-51*, นนทบุรี : ไอดีซีฯ, 2548

Federic Patin. "Beat Detection Algorithm." http://www.flipcode.com/misc /BeatDetectionAlgorithms.pdf/

Krister Olsson. "Ess r2 sound library." http://www.tree-axis.com/Ess/index.html

[1] "Beat Detection Algorithm." http:// www.flipcode.com/misc/BeatDetectionAlgorithms.pdf

[2] "PC interface Hardware and computer devices resource." http://www.english.thaiio.com/ เอสซีอาร์-scr/

[3] "Kingsolder." http://www.kingsolder.com/electronics/equipment/diac.asp

[4] "Kingsolder." http://www.kingsolder.com/electronics/equipment/triac.asp

[5] "en-Wikipedia." http://en.wikipedia.org/wiki/Solid_state_relay

[6] "Kingsolder."" http://www.kingsolder.com/electronics/light/opto-isolator.asp

[7] "Circuit and Datasheets World." http://circuits.datasheetdir.com/17/AT89C51-pinout.jpg

[8] "Acroname." http://www.acroname.com/robotics/parts/R257-3X4-KEYPAD.jpg

[9] "Keil Official website." http://www.keil.com/

[10] "Visual Studio 2008." http://kris.vizz.pl/blog/images/VS2008.jpg

[11] "Processing." http://www.processing.org/

[12] "Beat Detection Algorithms." http://www.flipcode.com/misc/

ภาคผนวก ก

เอกสารอ้างอิง

Tempo and beat analysis of acoustic musical signals

Eric D. Scheirer[a)]
Machine Listening Group, E15-401D MIT Media Laboratory, Cambridge, Massachusetts 02139

(Received 27 December 1996; revised 26 August 1997; accepted 15 September 1997)

A method is presented for using a small number of bandpass filters and banks of parallel comb filters to analyze the tempo of, and extract the beat from, musical signals of arbitrary polyphonic complexity and containing arbitrary timbres. This analysis is performed causally, and can be used predictively to guess when beats will occur in the future. Results in a short validation experiment demonstrate that the performance of the algorithm is similar to the performance of human listeners in a variety of musical situations. Aspects of the algorithm are discussed in relation to previous high-level cognitive models of beat tracking. © *1998 Acoustical Society of America.*
[S0001-4966(98)02801-X]

PACS numbers: 43.75.Yy, 43.75.St [WJS]

INTRODUCTION

Automatic extraction of rhythmic pulse from musical excerpts has been a topic of active research in recent years. Also called *beat-tracking* and *foot-tapping*, the goal is to construct a computational algorithm capable of extracting a symbolic representation which corresponds to the phenomenal experience of "beat" or "pulse" in a human listener.

"Rhythm" as a musical concept is intuitive to understand, but somewhat difficult to define. Handel writes "The experience of rhythm involves movement, regularity, grouping, and yet accentuation and differentiation" (Handel, 1989, p. 384) and also stresses the importance of the phenomenalist point of view—there is no "ground truth" for rhythm to be found in simple measurements of an acoustic signal. The only ground truth is what human listeners agree to be the rhythmic aspects of the musical content of that signal.

As contrasted with "rhythm" in general, "beat" and "pulse" correspond only to "the sense of equally spaced temporal units" (Handel, 1989). Where "meter" and "rhythm" associate with qualities of grouping, hierarchy, and a strong/weak dichotomy, "pulses" in a piece of music are only periodic at a simple level. For our purposes, the *beat* of a piece of music is the sequence of equally spaced phenomenal impulses which define a tempo for the music. This paper is only concerned with beat and tempo. The grouping and strong/weak relationships which define rhythm and meter are not considered.

It is important to note that there is no simple relationship between polyphonic complexity—the number and timbres of notes played at a single time—in a piece of music, and its rhythmic complexity or pulse complexity. There are pieces and styles of music which are texturally and timbrally complex, but have straightforward, perceptually simple rhythms; and there also exist musics which deal in less complex textures but are more difficult to rhythmically understand and describe.

The former sorts of musical pieces, as contrasted with the latter sorts, have a "strong beat," and it is with them that this paper is predominantly concerned. For these kinds of music, the rhythmic response of listeners is simple, immediate, and unambiguous, and every listener will agree on the rhythmic content. Rhythmically complex music is discussed toward the end of the paper.

Previous approaches

There is a large body of work originating in the music-psychology community which attempts to group musical *onsets* together into a rhythmic context; that is to say, to construct a model which subsumes multiple onsets separated in time into a rhythmic clock, "hierarchy," grouping, or oscillatory model.

Povel and Essens presented research (Povel and Essens, 1985) on the association of "internal clocks" with temporal onset signals. They described an algorithm which could, given a set of inter-onset intervals as input, identify the clock which a listener would associate with such a sequence of intervals. Their research was particularly interested in the way that perceived accents lead to the internal clock. Although obviously related to music, their research purports to examine time intervals in general rather than being restricted to musical stimuli. Parncutt's recent work (Parncutt, 1994) extends this type of model to include a great deal of structural information about duration and phenomenal accent.

Desain and Honing have contributed many results to the computational modeling of beat-tracking. Their models (Desain and Honing, 1992; Desain, 1995) typically also begin with inter-onset intervals and associate a rhythmic pulse with the interval stream. However, unlike the Povel/Essens and Parncutt models, these models are *process models*—they process the input sequentially rather than all-at-once—a necessary aspect of a model of human rhythmic perception. Desain's "(de)composable" model calculates rhythmic expectations due to each of the possible inter-onset times in a rhythmic stream, and sums them to create an overall rhythmic expectation.

Large and Kolen have described a beat-tracking model (Large and Kolen, 1994) based on nonlinear oscillators. The model takes a stream of onsets as input, and uses a gradient-descent method to continually update the period and phase of an oscillator. In this manner, the oscillator is matched with

[a)]Electronic mail: eds@media.mit.edu

the input stream, and the resulting oscillation process seems to be a good match for the human perception of beat.

Longuet-Higgens and Lee have written many papers (for example, Longuet-Higgens and Lee, 1984) on the induction of rhythmic hierarchies from monophonic time sequences. They are more interested in the development of theories which describe the relationship of rhythm, meter, and phrasing than on the boot-strapping process which creates a tempo and beat percept. Tempo perception may be viewed as "underlying" their models.

These approaches, and others such as Rosenthal (1993) and Brown (1993), require that robust onset detection precede beat analysis, which entails an important restriction to their applicability. The models do not operate on acoustic signals, but on symbolic data such as event lists or MIDI. As the extraction of onsets from multitimbral, polyphonic music is itself a difficult problem, this is a serious restriction of any model which claims to treat human rhythm perception. There has been little attempt to merge these sorts of models with real-time acoustic pattern recognition to allow them to work with acoustic data.

More recently, there has been some research attempting to extract rhythm and/or pulse information directly from acoustic signals. Goto has demonstrated a system which combines both low-level "bottom-up" signal processing and high-level pattern matching and "agent-based" representations to beat-track and do simple rhythmic grouping for popular music (Goto, in press). His method extracts drum patterns from a signal and uses a template-matching model to determine the beat from the drum track. This system runs in real time on a parallel-processing computer and has been used to control interactive-graphics displays from ecological music signals. His description does not directly address the equivalent processing of signals without drums, but it seems that the required musical knowledge base would be much more difficult to acquire.

N. P. Todd's work (Todd, 1994) has described algorithms which detect onsets in monophonic music under certain timbral constraints, and then group these onsets in a rhythmic framework using a multi-scale smoothing model. The onset model used is a simple one based on leaky integration. The resulting "rhythmogram" representation conceives of pulse, and in some cases, meter and phrase, perception as a very low-level process arising directly from the time- and loudness-integration properties of the auditory periphery. The model as presented can be implemented in an incremental manner, but was only tested using toy examples (although, interestingly, a speech example was included).

All of the abovementioned research uses what has been described as a *transcriptive* metaphor for analysis (Scheirer, 1996). That is, the music is first segmented, or assumed to already be segmented, into notes, onsets, timbres, and so forth. Post-processing algorithms are then used to group rhythms and track beats. As high-quality polyphonic music transcription algorithms are still years in the future—the state-of-the-art systems cannot transcribe pieces more complex than four-voice piano music (Martin, 1996)—it seems logical for practical reasons to attempt to construct systems which can arrive at a musical understanding of a piece of music without going through a transcription step. Further, as the validity of the transcriptive metaphor as a framework for music perception has been challenged (Scheirer, 1996), it is scientifically appropriate as well.

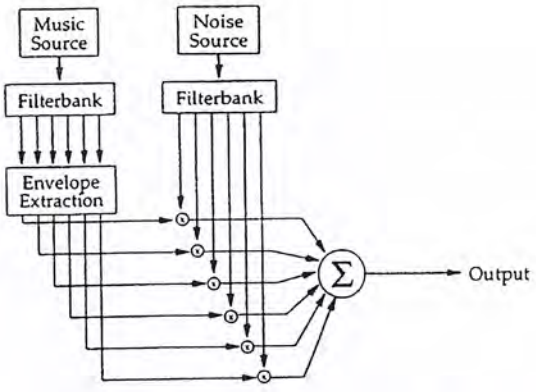


FIG. 1. Creating a "modulated noise" signal from a music signal. The output signal, for many sorts of frequency filterbanks, will have the same rhythmic percept as the input music signal, indicating that the amplitude envelopes of the bands are a sufficient representation for rhythmic analysis.

In the body of this paper, the following topics are discussed: psychoacoustic demonstrations which lead to processing simplifications for beat-tracking, the construction of the algorithms themselves, example results from test signals and ecological signals, a validation experiment which compares the behavior of the algorithm to that of human subjects, the relationship of this model to previous models of rhythm perception, and finally, conclusions about beat-tracking and rhythmic grouping and a description of future work to be pursued in these directions.

I. PSYCHOACOUSTIC SIMPLIFICATION

One of the key difficulties with the transcriptive models of rhythmic perception described above is the complexity of grouping harmonic partials together to form notes, and determining the onset times of those notes. Even if simplifying assumptions about the pitch and timbral content are made, identifying attack and release times is no easy task (Scheirer, in press).

However, it seems from a psychoacoustic demonstration on beat perception that certain kinds of signal manipulations and simplifications can be performed without affecting the perceived pulse content of a musical signal. Consider the signal flow network shown in Fig. 1.

An "amplitude-modulated noise" is constructed by signal by vocoding a white noise signal with the subband envelopes of a musical signal. This is accomplished by performing a frequency analysis of the music (processing through a filterbank of bandpass filters, perhaps, or grouping output from FFT bins together), and also of a white-noise signal from a pseudo-random generator. The amplitude of each band of the noise signal is modulated with the amplitude envelope of the corresponding band of the musical filterbank output, and the resulting noise signals are summed together to form an output signal.

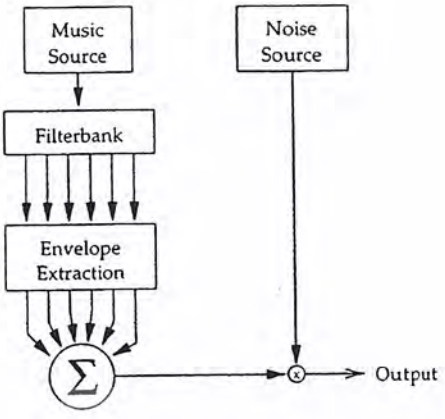


FIG. 2. A noise signal which does not have the same rhythmic characteristics as the musical input, indicating that the sum of the amplitude envelopes is not a sufficient representation for rhythm analysis. Certain types of nonlinear combination by frequency channel are evidently present in the beat perception facility.

For many kinds of frequency filterbanks, the resulting noise signal has a rhythmic percept which is significantly the same as that of the original music signal. Even if there are very few, very broad bands (for example, four three-octave bands covering the audible spectrum), the pulse and meter characteristics of the original signal are instantly recognizable (sound example #1a) [Audio examples for this paper can be found on the author's WWW site at http://sound.media.mit.edu/eds/beat/].

Since the only thing preserved in this transformation is the amplitude envelopes of the filterbank outputs, it stands to reason that only this much information is necessary to extract pulse and meter from a musical signal; that is, algorithms for pulse extraction can be created which operate only on this much input data, and "notes" are not a necessary component for hearing rhythm. This is a vast reduction of input data size from the original signal. Shannon has reported a similar effect for the perception of speech (Shannon, 1995).

Certain other kinds of simplifications are not possible. For example, if only one band is used, or equivalently, the subband envelopes are linearly combined before modulating the noise (Fig. 2) (Vercoe, 1994) a listener can no longer perceive the rhythmic content of many signals (sound example #1b). Thus it seems that separating the signal into subbands and maintaining the subband envelopes separately is necessary to do accurate rhythmic processing.

Stated another way, the algorithm in Fig. 2 is a method for generating new signals whose representation under a filterbank-envelope-and-sum process is the same as a given piece of music. However, since these new signals often do not bear a perceptual equivalency with the originals, the filter-envelope-sum framework must be *inadequate* to represent data in the musical signal which is important for rhythmic understanding. This fact immediately leads to a psychoacoustic hypothesis regarding rhythmic perception: some sort of cross-band rhythmic integration, not simply summation across frequency bands, is performed by the auditory system.

A psychoacoustic experiment to examine the exact properties of filterbank-and-envelope manipulations which do not disturb rhythm perception is underway; in the meantime, it seems important that a rhythmic processing algorithm should treat frequency bands separately, combining results at the end, rather than attempting to perform beat-tracking on the sum of filterbank outputs.

II. DESCRIPTION OF ALGORITHM

The beat-tracking algorithm to be presented here bears most resemblance to the method of Large and Kolen (Large and Kolen, 1994) in that it uses a network of resonators to phase-lock with the beat of the signal and determine the frequency of the pulse. However, the particular method used here is somewhat different; the resonators are analytically much simpler than theirs, a bank of resonators is used rather than gradient descent, and more pre- and post-processing of the signal is necessary in order to accurately extract the desired information, as the present model operates on acoustic data rather than an event stream.

A rhythmic pulse is described in terms of a frequency and phase component, just as for a periodic sound waveform; the frequency of the pulse in a rhythmic musical signal is the tempo or rate of the rhythm, and the phase of the pulse indicates where the "downbeat" of the rhythm occurs. That is, the times at which a pulse occurs can be defined to have zero phase, and thus the points in time exactly in-between pulses have phase of π radians, etc. It is important to note that while human pitch recognition is only sensitive to signal phase under certain unusual conditions, rhythmic response is crucially a phased phenomenon—tapping on the beat is not at all the same as tapping against the beat, or slightly ahead of or behind the beat, even if the frequency of tapping is accurate.

Figure 3 shows an overall view of the tempo-analysis algorithm as a signal flow network. The functionality will be briefly described, and then more details given piece-by-piece in the following sections. The algorithms here were developed empirically; however, in Sec. V their relationship to existing models of rhythm perception is discussed.

As the signal comes in, a filterbank is used to divide it into six bands. For each of these subbands, the amplitude envelope is calculated and the derivative taken. Each of the envelope derivatives is passed on to another filterbank of *tuned resonators*; in each resonator filterbank, one of the resonators will phase-lock, the one for which the resonant frequency matches the rate of periodic modulation of the envelope derivative.

The outputs of the resonators are examined to see which ones are exhibiting phase-locked behavior, and this information is tabulated for each of the bandpass channels. These tabulations are summed across the frequency filterbank to arrive at the frequency (tempo) estimate for the signal, and reference back to the peak phase points in the phase-locked resonators determines the phase of the signal.

A. Frequency analysis and envelope extraction

As discussed in Sec. I, envelopes extracted from a small number of broad frequency channels are sufficient information to rhythmically analyze a musical signal, at least for

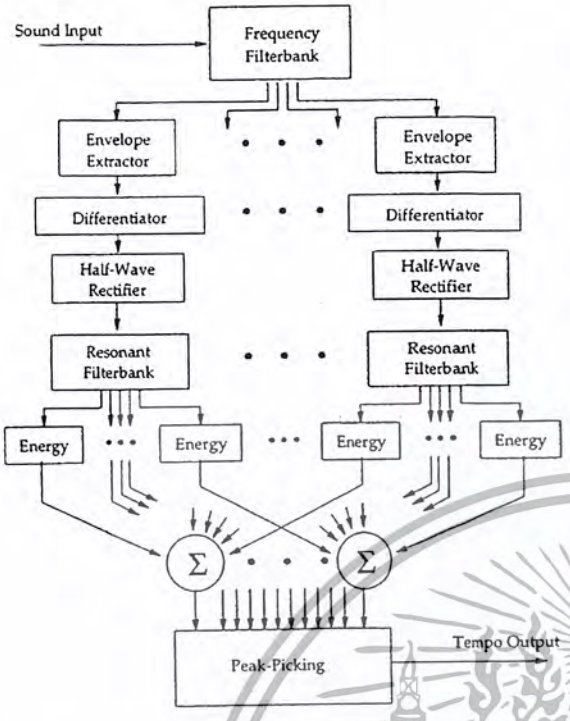


FIG. 3. Schematic view of the processing algorithm. See text for details.

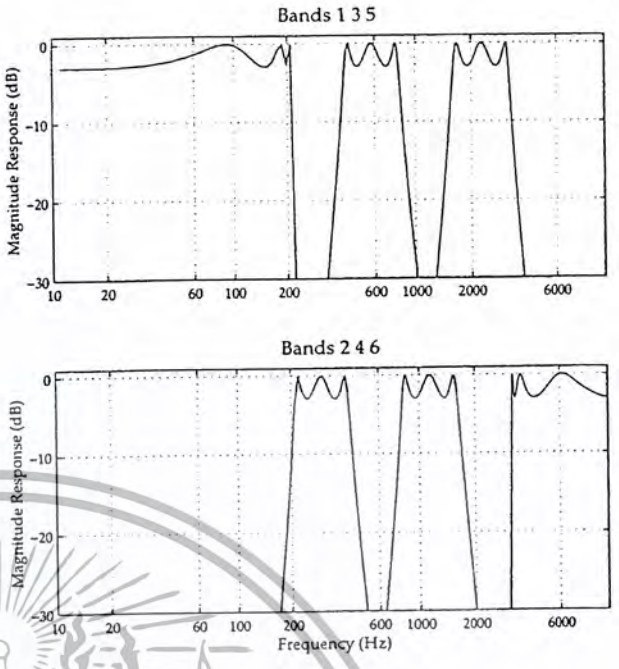


FIG. 4. Magnitude response of the frequency filterbank used in the system, plotted in two pieces for clarity. The upper plot shows the first, third, and fifth bands; the lower, the second, fourth, and sixth. Each filter is a sixth-order elliptic filter, with 3 dB of passband ripple and 40 dB of stopband rejection.

human listeners. Further, empirical studies of the use of various filterbanks with this algorithm have demonstrated that the algorithm is not particularly sensitive to the particular bands or implementations used; it is expected that psychoacoustic investigation into rhythmic perception of amplitude-modulated noise signals created with the various vocoder filterbanks would confirm that the same is true of human rhythmic perception.

The filterbank implementation in the algorithm has six bands; each band has sharp cutoffs and covers roughly a one-octave range. The lowest band is a low-pass filter with cutoff at 200 Hz; the next four bands are bandpass, with cutoffs at 200 and 400 Hz, 400 and 800 Hz, 800 and 1600 Hz, and 1600 and 3200 Hz. The highest band is high pass, with cutoff frequency at 3200 Hz. Each filter is implemented using a sixth-order elliptic filter, with 3 dB of ripple in the passband and 40 dB of rejection in the stopband. Figure 4 shows the magnitude responses of these filters.

The envelope is extracted from each band of the filtered signal through a rectify-and-smooth method. The rectified filterbank outputs are convolved with a 200-ms half-Hanning (raised cosine) window. This window has a discontinuity at time $t=0$, then slopes smoothly away to 0 at 200 ms. It has a low-pass characteristic, with a cutoff frequency at about 10 Hz (''frequency'' in this case referring to envelope spectra, not waveform spectra), where it has a -15 dB response, and 6-dB/octave smooth rolloff thereafter.

The window's discontinuity in time means that it has nonlinear phase response; it passes slow envelope frequencies with much more delay than rapid ones. High frequencies, above 20 Hz, are passed with approximately zero delay; 0 Hz is delayed about 59 ms and 7 Hz advanced about 14 ms. Thus there is a maximum blur of about 73 ms between these envelope frequencies.

This window performs energy integration in a way similar to that in the auditory system, emphasizing the most recent inputs but masking rapid modulation; Todd (1992) examines the use of temporal integration filters which are directly constructed from known psychoacoustic properties. After this smoothing, the envelope can be decimated for further analysis; the next stages of processing operate on the decimated band envelopes sampled at 200 Hz. There is little energy left in the envelope spectra at this frequency, but it aids the phase-estimation process (see below) to maintain a certain precision of oversampled envelope resolution.

After calculating the envelope, the first-order difference function is calculated and half-wave rectified; this rectified difference signal will be examined for periodic modulation. The derivative-of-envelope function performs a type of onset filtering process (see, for example, Smith's work on difference-of-Gaussian functions for onset segmentations Smith, 1994) but the explicit segmentation, thresholding, or peak-peaking of the differenced envelope is not attempted. The subsequent modulation detectors in the algorithm are sensitive, similar to the sensitivity of autocorrelation, to ''imperfections'' in an onset track. The half-wave rectified envelope difference avoids this pitfall by having broader (in time) response to perceptual attacks in the input signal. This process might be considered similar to detecting onset points in the signal bands, and then broadening them via low-pass filtering.

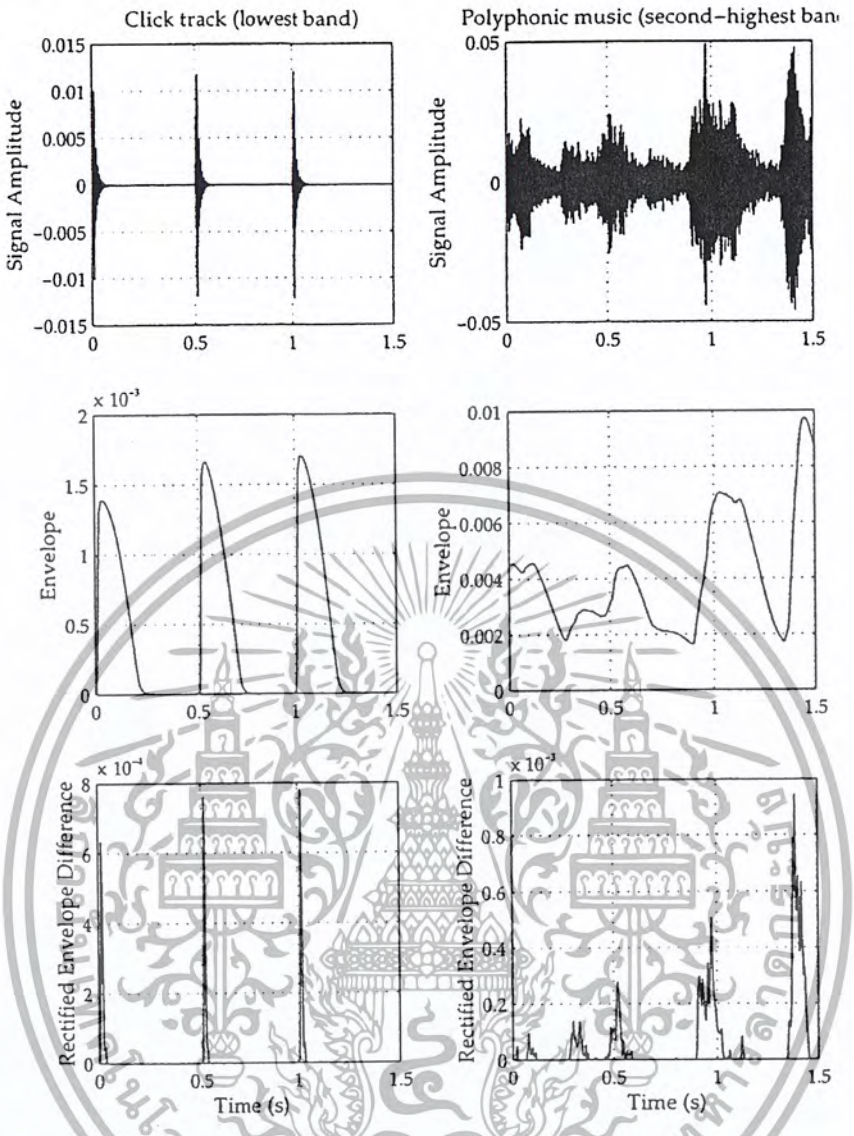


FIG. 5. Envelope extraction process, for a 2-Hz click track (left) and a polyphonic music example (right). The top panels show the audio waveforms; the middle panels, the envelopes; and the bottom, the half-wave rectified difference of envelopes. The lowest filterbank band is shown for the click track, the second-highest for the music. See text for details on algorithms.

Figure 5 shows the envelope extraction process for one frequency band in each of two signals, a 2-Hz click track and a polyphonic music example. The lowest band is shown for the click track, and the second highest for the music track.

B. Resonators and tempo analysis

After the envelope has been extracted and processed for each channel, a filterbank of comb filter resonators is used to determine the tempo of the signal. While comb filters are often used in reverberators and other sorts of audio signal processing, they also have properties which make them suitable for acting as resonators in the phase-locking pulse extraction process.

In particular, if we stimulate a comb filter with delay T and gain α with a right-sided pulse train of height A and period κ, we get reinforcement (resonance) if $T=\kappa$. Let x_t and y_t be the input and output signals at time t; the equation of the filter is then $y_t=\alpha y_{t-T}+(1-\alpha)x_t$, and

$$y_0=(1-\alpha)A$$

$$y_\kappa=\alpha(1-\alpha)A+(1-\alpha)A=(1-\alpha)A(1+\alpha)$$

$$y_{2\kappa}=(1-\alpha)A(\alpha^2+\alpha+1)$$

$$\vdots$$

$$y_{n\kappa}=(1-\alpha)A\left(\sum_{i=0}^{n}\alpha^i\right).$$

And so $\lim_{n\to\infty} y_{n\kappa}=[(1-\alpha)A]/(1-\alpha)=A$.

On the other hand, if $T\neq\kappa$, the convergence is to a smaller value. Let λ be the least common multiple (common

period) of T and κ; there is only reinforcement every T/λ periods, and by a similar logic as the above,

$$\lim_{n\to\infty} y_{n\lambda} = \frac{(1-\alpha)A}{1-\alpha^{T/\lambda}},$$

and since $|\alpha|<1$ if the filter is to be stable, and $T/\lambda \geq 1$,

$$1-\alpha^{T/\lambda} \geq 1-\alpha.$$

So a filter with delay matching (or evenly dividing) the period of a pulse train will have larger (more energetic) output than a filter with mismatched delay.

We can see that this is true for any periodic signal by doing the analysis in the frequency domain. The comb filter with delay T and gain α has magnitude response

$$|H(e^{j\omega})| = \left|\frac{1-\alpha}{1-\alpha e^{-j\omega T}}\right|,$$

which has local maxima wherever $\alpha e^{-j\omega T}$ gets close to 1, i.e., at the Tth roots of unity, which can be expressed as

$$e^{-j2\pi n/T}, \quad 0 \leq n < T.$$

Using Fourier's theorem we know that these frequency-domain points are exactly those at which a periodic signal of period T has energy. Thus the comb filter with delay T will respond more strongly to a signal with period T than any other, since the response peaks in the filter line up with the frequency distribution of energy in the signal.

For each envelope channel of the frequency filterbank, a filterbank of comb filters is implemented, in which the delays vary by channel and cover the range of possible pulse frequencies to track. The output of these resonator filterbanks is summed across frequency subbands. By examining the energy output from each resonance channel of the summed resonator filterbanks, the strongest periodic component of the signal may be determined. The frequency of the resonator with the maximum energy output is selected as the tempo of the signal.

The α parameter for each comb filter is set differently, so that each filter has equivalent half-energy time. That is, a comb filter of period T has an exponential curve shaping its impulse response. This curve reaches half-energy output at the time t when $\alpha^{T/t}=0.5$. Thus α is set separately for each resonator, at $\alpha = 0.5^{t/T}$. A half-energy time of 1500–2000 ms seems to give results most like human perception.

Figure 6 shows the summed filterbank output for a 2-Hz pulse train and for a polyphonic music example. The horizontal axis is labeled with "metronome marking" in beats per minute; this is a direct mapping of the delay of the corresponding comb filter. That is, for the 2-Hz power envelope signal, a feedback delay of 100 samples corresponds to a 500-ms resonance period, or a tempo of 120 bpm.

In the pulse train plot in Fig. 6, a clear, large peak occurs at 120 bpm, and additional smaller peaks at tempi which bear a simple harmonic relationship (3::2 or 4::5, for example) to the main peak. In the music plot, there are two peaks, which correspond to the tempi of the quarter note and half note in this piece. If the width of the upper plot were extended, a similar peak at 60 bpm would be visible.

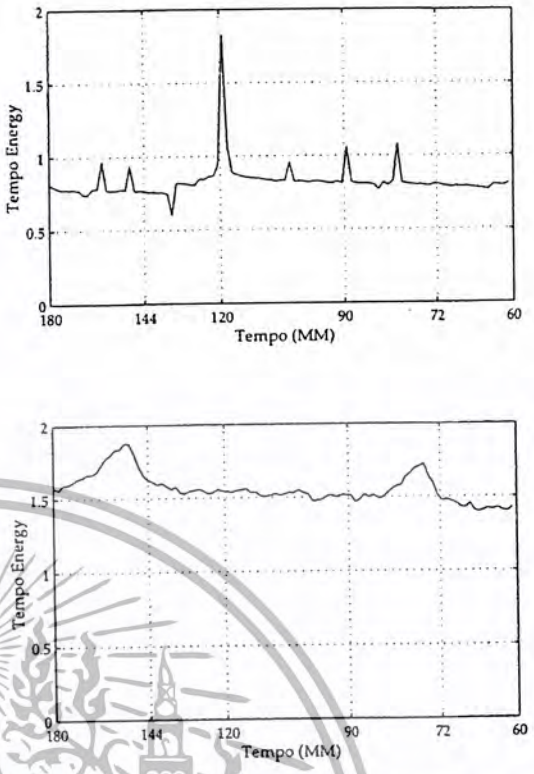


FIG. 6. Tempo estimates, after tracking 5 s of a 2-Hz click track (top) and of a polyphonic music example (bottom). The x-axes are labeled in beats per minute, that is, 120 MM = 2 Hz. The polyphonic music shows more overall energy, but the tempo is still seen clearly as peaks in the curve.

C. Phase determination

It is relatively simple to extract the phase of the signal once its tempo is known, by examining the output of the resonators directly, or even better, by examining the internal state of the delays of these filters. The implementations of the comb filters for the resonator filterbank have lattices of delay-and-hold stages. The vector w of delays can be interpreted at a particular point in time as the "predicted output" of that resonator; that is, the next n samples of envelope output which the filter would generate in response to zero input.

The sum of the delay vectors over the frequency channels for the resonators corresponding to the tempo determined in the frequency extraction process are examined. The peak of this prediction vector is the estimate of when the next beat will arrive in the input, and the ratio $\omega = 2\pi(t_n - t)/T$, where t_n is the time of the next predicted beat, t the current time, and T the period of the resonator, is the phase ω of the tempo being tracked. The phase and period may thus be used to estimate beat times as far into the future as desired.

The implementation of the model performs the phase analysis every 25 ms and integrates evidence between frames in order to predict beats. Since re-estimation occurs multiple times between beats, the results from each phase analysis can be used to confirm the current prediction and adjust it as

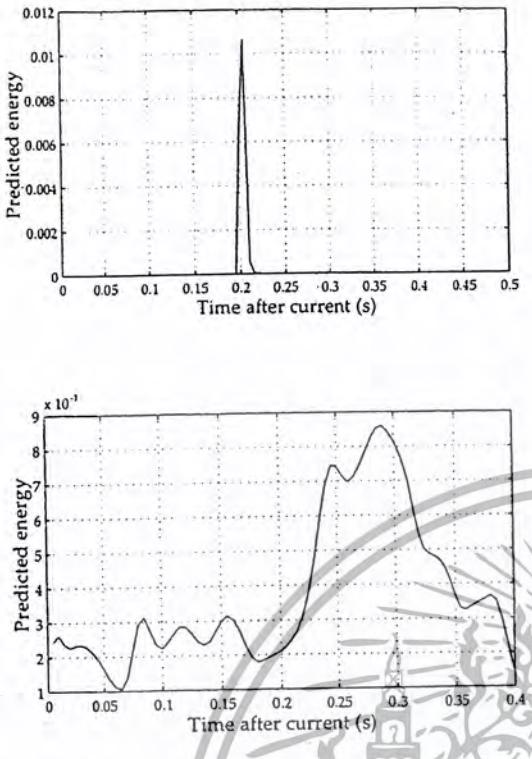


FIG. 7. Phase estimates, after tracking 5 s of a 2-Hz click track (top) and a polyphonic music example (bottom). The x-axis in each case covers the next full period of the resonator tracking the tempo, and the peak of the curve shows where the next beat is predicted to occur: about 210 ms in the future for the upper case, and 290 ms for the lower.

needed. Currently, this prediction/adjustment is done in an *ad hoc* manner, requiring only that several successive frames make the same beat prediction within a certain tolerance, and average all of these estimates to arrive at the final prediction. This stage is the appropriate one for the inclusion of high-level information, nondeterministic elements, or more sophisticated rhythmic modeling; see Sec. VI.

Figure 7 shows the phase peaks for a 2-Hz pulse train, and for a polyphonic music example. In the upper plot, as the tempo is 120 bpm, the x-axis covers the next half-second of time; and for the lower plot, the estimated tempo is 149 bpm (see Fig. 6), so one period is approximately 400 ms.

D. Comparison with autocorrelation methods

There is a certain analytical similarity between this bank-of-comb-filters approach and previous autocorrelation methods for finding tempo. Insofar as both are ways of detecting periodic energy modulations in a signal, they are performing similar calculations. However, there are several advantages to expressing these operations as multiple comb filters over expressing them as autocorrelation.

Predominantly, comb filtering implicitly encodes aspects of rhythmic hierarchy, where autocorrelation does not. That is, a comb filter tuned to a certain tempo τ has peak response to stimuli at tempo τ, but also lesser response to stimuli with tempi at multiples $(2\tau,3\tau)$, fractions $(\tau/2,\tau/3)$, and simple rational relationships $(3/2\tau,3/4\tau$, etc). The autocorrelation only has this shared response for fractional tempi, not multiples or rationally related tempi. An autocorrelation model asserts that a click track at 60 bpm gives no sense of tempo at 120 bpm, which seems intuitively wrong. The comb filter model asserts instead, that there is such a sense, but a reduced one when compared to a click track to 120 bpm.

These responses can be understood if we imagine building an autocorrelation filter at some lag, versus a comb filter at that same delay, in an FIR manner (that is, to unroll the usual IIR expression of the comb filter into an infinitely long "FIR" filter). The autocorrelation requires only a single tap on a delay line, since it only compares "one cycle back" in time. The comb filter requires an infinite number of taps, since it compares (with less and less weight) infinitely far back in time.

Autocorrelation methods are zero phase, which means that some other method of determining signal phase must be used. The comb filtering method shown here is phase preserving, and so provides a way of simultaneously extracting tempo and phase, as discussed in the previous section. The fact that the tempo and phase representations arise together gives us additional advantages in constructing higher-level processing algorithms treating the output of the beat-tracker.

One advantage of autocorrelation schemes is that they are more efficient in memory usage than banks of comb filters, as the various lags can all access the same delay line—which is why the autocorrelation is zero phase—whereas each comb filter must maintain a delay line of its own. In return for the extra memory usage, the comb filters provide estimates of output energy at each phase angle of each lag, where the autocorrelation accumulates it and only presents the summary.

Ultimately, it is representationally satisfying to have the frequency and phase of the signal explicitly encoded in the processing units of the algorithm. In an autocorrelation methodology, the rhythmic oscillations of the signal are only represented as post-processed summary results; whereas in the comb filtering method, the filter states themselves explicitly represent the rhythmic content—that is, there is an element of the processing network which phase-locks to and oscillates in synchrony with the signal.

III. IMPLEMENTATION AND COMPLEXITY

The algorithms described above have been implemented in $C++$ code; the resulting program causally processes audio files captured from compact disks or other audio recordings, or coming in via a live microphone input. In this section, the parameters available for controlling the speed and accuracy of the program are described.

A. Program parameters

The current implementation of the system has a number of parameters which can be used to control the accuracy/speed relationship of the algorithms. The program will run in real time on a very fast desktop workstation such as a DEC Alpha, depending on the settings of these parameters and the sampling rate of the incoming audio stream. It is also clear, due to the highly parallel structure of Fig. 3, that the algo-

rithm could efficiently make use of a multiple-processor architecture. This has not yet been accomplished, however.

There are four major areas where the performance and accuracy of the system can be tuned, and control over three of them has been implemented. The algorithm has been tested for audio at sampling rates from 8 KHz to 44.1 KHz and gives roughly equivalent qualitative performance in all of these.

1. Frequency filterbank

As discussed in Sec. II, there is a fair amount of latitude in choosing a frequency filterbank for decomposing the incoming audio stream without affecting human rhythmic perception, and the speed of the system will vary a great deal with the complexity of these filters (since there is a fair CPU load for implementing high-order filters in real time on high-bandwidth audio), and their number (since for each of the frequency channels, a full resonator filterbank structure is implemented).

The performance of the beat-tracking program using filterbanks other than the six-channel sixth-order IIR filterbank described above has not been tested.

2. Envelope sampling rate

The decimation rate of the channel envelopes affects the speed and performance of the system. There are two major implications for using a slow envelope sampling rate: (1) there are many resonator frequencies which cannot be represented accurately with integer delays in the comb filters; and (2) the phase extraction can only be performed with accuracy equal to the envelope sampling rate, since the vector of delays has the same sampling rate.

In tradeoff to this, using a fast sampling rate for the envelopes entails a lot of work in the comb filtering, since the number of multiplies in each comb filter varies proportionately to this rate. Empirical testing over a variety of musical examples suggests that the envelopes should be sampled at least 100 Hz or so for best performance.

3. Number of resonators per frequency channel

The amount of computing incorporated in tracking and analysis of the comb filter resonators varies directly with their number. If too few resonators are used, however, a problem develops with sampling the tempo spectrum too sparsely. That is, since each resonator is attempting to phase-lock to one particular frequency (not to a range of frequencies), if there is no resonator tuned close to the tempo of a particular signal, that signal cannot be accurately tracked.

Also affecting this sparsity consideration is the range of resonator frequencies to be tracked. The wider the range of tempi to track, the sparser a fixed number of resonators will spread over that range.

Good results have been generated using a bank of 150 resonators for each channel, covering a logarithmically spaced range of frequencies from 60 bpm (1 Hz) to 240 bpm (3 Hz).

4. Analysis frame rate

In this particular implementation, a higher-level averaging scheme is used to decide where (at what times) to deduce beats in the input signal. That is, for each analysis frame, the phases of the resonators are examined; the evidence here suggests future beat locations. These suggestions are combined over multiple analysis frames; when several frames in a row point to the same future beat location, evidence accumulates for that time, and a beat is actually assigned there.

Thus the frequency with which the procedure of examining and summing the outputs and internal states of the resonators is executed has a strong effect upon the performance and speed of the program. Good results can be obtained if the analysis frame rate is at least 15 Hz.

Real-time performance cannot be obtained with the parameter values shown above; on an Alpha 3000 using highly optimized filtering and analysis code, with the envelope rate set to 75 Hz, 50 resonators per subband, and frames of beat predictions analyzed every 10 Hz, the required performance for real-time operation on 22-KHz input is reached. This real-time performance includes reading the sound file from disk and playing it back with short noise bursts added to highlight the beats. At this level of accuracy, the algorithm still performs acceptably well on some, but not all, musical examples.

B. Behavior tuning

In addition to controlling the tradeoff between program speed and accuracy, the behavior of the algorithm can be tuned with the α parameters in the comb filters. These parameters can be viewed as controlling whether to value old information (the beat signal extracted so far) or new information (the incoming envelopes) more highly. Thus if α is large (close to unity), the algorithm tends to "lock on" to a beat, and follow that tempo regardless of the new envelope information. On the other hand, if α is small, the beat-track can be easily perturbed by changes in the periodicity of the incoming signal. Manipulating these parameters for the comb filter structure is computationally similar to manipulating the windowing function of a narrowed autocorrelation.

Higher-level or domain-specific knowledge could be used to set this parameter based on previous information. For example, in rock or pop music, the beat is usually quite steady, so a high value for α would be appropriate; while for classical music, particularly styles including many tempo changes, a smaller value would be more optimal.

IV. VALIDATION

It is somewhat of a difficult proposition to evaluate the construction of an ecological beat-tracking model, for there are few results in the literature dealing with listeners' tempo responses to actual musical excerpts. Most psychophysical research has dealt primarily with special cases consisting of simple tones in unusual temporal relationships, which will typically be more difficult to track than "real music" for a listener. Conversely, most beat-tracking systems have been

TABLE I. Performance of the beat-tracking algorithm, summarized by musical genre. Results were auditioned and classified into groups by qualitative success level. "Urban" styles include rap, funk, and R & B music; "Quiet" includes muzak and an "easy-listening" example. All sounds are available via the WWW.

Genre	No. of cases	Correct	Partial	Wrong
Rock	17	13	3	1
Country	3	3	0	0
Urban	9	7	1	1
Latin	5	3	2	0
Classical	9	4	4	1
Jazz	8	3	1	4
Quiet	3	2	0	1
Reggae	2	2	0	0
Non-Western	4	4	0	0
Total	60	41	11	8

evaluated intuitively, by using a small number of test cases (whether acoustic or MIDI-based) and checking that the algorithm "works right."

In this section, the performance of the algorithm is evaluated in both qualitative and quantitative manners. Results are provided on the qualitative performance for 60 ecological music excerpts, with sound examples publicly available for listening. Results are also provided from a short validation pilot experiment which was conducted to confirm that the performance of the algorithm is like the performance of human listeners.

A. Qualitative performance

Examples of many different types of music have been tested with the implemented algorithm, using a short application which reads a sound sample off of disk, causally beat-tracks it, and writes a new sound file with clicks (short noise bursts) added to the signal where beats are predicted to occur. A selection of these sound files is available for listening via the World Wide Web ("results" page), and the results are summarized below. The wide set of input data contains 60 examples, each 15 s long, of a number of different musical genres. Rock, jazz, funk, reggae, classical, "easy-listening," dance, and various non-Western music are represented in the data set and can be tracked properly. Some of the examples have drums, some do not; some have vocals, some do not. Five of the examples would be judged by human listeners to have no "beat." Table I summarizes the results by musical genre, and some qualitative descriptions of typical results are provided below.

Forty-one of 60 samples (68%) have been qualitatively classified as being tracked accurately, and another 11 (18%) as being tracked somewhat accurately. This accuracy percentage is not directly comparable to that reported for other systems, because the data set used here is more difficult. All of the "easy" cases of rock-and-roll with drums keeping a straightforward beat were tracked correctly; and five of the eight examples not tracked accurately are said by human listeners to have no "beat" to begin with. It is premature to interpret these results as indicative of consistent genre-to-genre differences in accuracy; there are too few examples and the within-genre differences in accuracy too great.

For the cases which track correctly, there is a startup period between 2 and 8 s long during which the resonant filters have not yet built up an accurate picture of the signal. After this period, for most signals, the algorithm has settled down and begun to track the signal accurately, placing the clicks in the same locations a human listener would. Examining some of the other, incorrectly tracked examples, is instructive and highlights some of the deficiencies of this method.

Examples #1, #2, and #57 are all up-tempo jazz cases in which human listeners do perceive a strong beat, but no beat is ever extracted by the system. In these three cases, the beat is described by syncopated instrumental lines and complex drum patterns. That is, there is not actually very much energy modulating at the frequency which is the perceptual beat tempo for humans. Human listeners have a great ability to induce "apparent" frequencies from complicated modulation sequences. For these examples, the algorithm is not able to find a pulse frequency, and so the beat output is more-or-less random.

The same is apparent in example #37, which is a pop tune that has a "mixed" or "clave" beat—the beat is not even, but subdivided into oddly spaced groups. Each two measures, containing 16 eighth notes between them, are divided into a 3-3-3-3-2-2 pattern. A human listener has no trouble understanding the relationship between this pattern and a more common 4-4-4-4 pattern, but the algorithm seems to assume that the groups of three are the basic beat, and then get confused when the pattern doesn't come out right.

Among the examples judged as being tracked with some accuracy, but not entirely correctly, the most common problem is phase shifting. For example, in example #16, a jazz piano trio, the beat estimate is correct on the frequency, but switches back and forth between assigning beats to the "upbeat" or the "downbeat." Although this behavior is not unlike some human jazz listeners, a human would likely be more consistent in deciding where to place the beat. This behavior could be easily corrected by adding a small amount of high-level knowledge to the beat-tracking system.

Similar to this, in example #7, a rhythm and blues tune, the algorithm is uncertain about assigning the beat to the quarter-note pulse or to the eighth-note pulse, and so switches back and forth between them. A human listener might also suffer from similar confusion, but would likely make an arbitrary decision and then stay with it unless the music changed radically.

Other than these two sorts of confusions for certain rhythmically complex musics, the algorithm seems to perform quite successfully at tracking the musical beats.

1. Tempo modulation

As Todd correctly points out (Todd, 1994), to be an accurate model of human rhythm perception (and, of course, to be maximally useful as a music analysis tool), a beat-

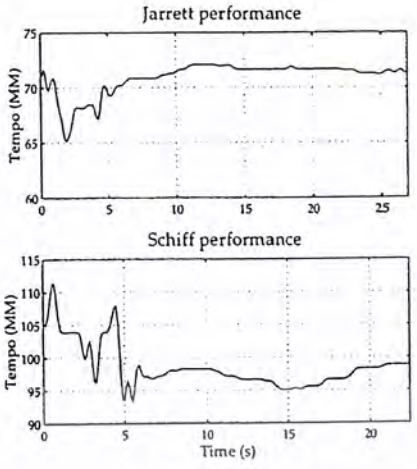


FIG. 8. "Tempo curve" for two performances of the same piece of music. Each tempo track has a short startup period during which the tempo estimation is unstable; after that there are clear differences in the two performances. The timescales are slightly different to make the performance scales align (the same musical excerpt is used in both cases).

tracking system must be robust under expressive tempo modulation. The algorithm described here is able to follow many types of tempo modulations; this is effected in the signal processing network by simply examining, over time, the resonator producing the most energetic output. That is, when the tempo of a signal modulates, the response of the resonator corresponding to the old tempo will die away, and that of the resonator corresponding to the new tempo will gain.

Figure 8 shows "tempo curves" (Desain and Honing, 1992) for two expressively modulated performances of a piece of music (Keith Jarrett and Andras Schiff performances, of the beginning of the G-minor fugue from book I of Bach's Bach's *Well-Tempered Clavier* [sound example 3]). The algorithm is quite sensitive to the variations in tempo over time.

B. Validation experiment

A short validation experiment has been conducted to confirm the qualitative results given in the previous section. This experiment was not intended to highlight important psychoacoustic effects in beat perception, but only to test whether the beat-tracking algorithm performs generally like a human listener.

1. Subjects

Five adult listeners, all graduate students and staff members at the MIT Media Laboratory, participated in the experiment. All were experienced musicians with normal hearing.

2. Overview of procedure

Subjects listened to seven musical examples, drawn from different musical genres, through headphones. They indicated their understanding of the beat in the music by tapping along with the music on a computer keyboard.

3. Materials

Seven musical excerpts from the above set were used. Each was digitally sampled from an FM radio tuner to produce a monophonic 22-KHz sound file, 15 s long. A computer interface was created on a DEC Alpha workstation with which the musical excerpts were presented to subjects at a comfortable listening level over AKG-K240M headphones.

The musical excerpts were as follows: a Latin-pop song at moderately fast tempo (#10), a jazz piano trio at fast tempo (#17), a "classic rock" song at moderately slow tempo (#20), an excerpt from a Mozart symphony at moderate tempo (#40), an "alternative rock" song at moderately slow tempo (#45), and a piano etude with varying tempo (#56).

A click track "step function" was also created for the experiment, in which 10-ms white noise bursts were presented at a tempo of 120 bpm (interonset time of 500 ms) for 6 s, then at a tempo of 144 bpm (interonset time of 417 ms) for 4.6 s, then again at 120 bpm for 6 more s. This stimulus is used to evaluate the response of human listeners and the beat-tracking algorithm to sudden changes in tempo.

A musical expert (the author) assigned exact beat times to each excerpt by listening repeatedly and placing "click" sounds in the perceptually appropriate positions. This task was different than the tapping task in which the subjects participated; the expert listened repeatedly to each stimulus, placing beats, listening to results, and adjusting the beat position if necessary. It is considered to be more accurate and robust than the real-time tapping task, although there is little literature on humans performing either of these sorts of judgments [see Drake *et al.* (1997) and Parncutt (1994) for two other "tapping tasks"]. The expert labeling was conducted separately from the tapping experiment, the expert did not know the results of the experiment or the algorithm execution, and the subjects were not presented with the expert data. The resulting "ground truth" beat times are used for the evaluation of results, below.

4. Detailed procedure

Subjects were seated in front of the computer terminal and instructed in the task: they were to listen to short musical examples and tap along with them using the space bar on the keyboard. They were instructed to tap at whatever tempo felt appropriate to the musical excerpt, but to attempt to tap in equal intervals (a pilot experiment revealed that some subjects like to "drum along" in rhythmic or even syncopated patterns with the music if they are not instructed otherwise). They listened to a 120-bpm click-track as a training sample to indicate they understood the procedure, and then proceeded with each of the seven experimental trials.

All seven trials were run in the same sequence for each listener, in a single block. The experiment was not counterbalanced based on an assumption that there is little training effect in this task. After each trial, the subject was instructed by the interface to press a key different than the space bar to continue to the next trial. The entire experiment took approximately 5 min per subject. The computer interface re-

corded the time of each tap, accurate to approximately 10 ms, and saved the times to a disk file for analysis.

Finally, the beat-tracking algorithm was executed on each of these seven stimuli to produce beat times as estimated by the model described in the previous sections. These beat times were saved to a disk file and analyzed for comparison with the human beat times. The algorithm parameters were adjusted to give optimum performance for this set of trials, but not changed from trial-to-trial.

5. Dependent measures

The human and algorithmic beat-tracks were analyzed in two ways. First, the beat placements were compared to the ideal placements as judged by the expert listener; then, the regularity of tapping was assessed by examining the variance of interonset times.

To compare the beat placements, a matching comparison was conducted. Each beat placed by a human subject or by the beat-tracking model was matched with the closest (in time) comparison beat in the expert beat-track. Initially, only the beats actually placed by the expert were used, but since some subjects and the algorithm tapped twice as fast as the expert on some examples, beats were allowed to be matched to the midpoint between expert beats. The root-mean-square deviations of the subject's taps from the expert's taps were collected for each subject and trial, averaging across taps within a trial.

This rms deviation is a measure of how close the tapper came to the "ideal" beat locations. If it is very low, all of the tapper's placements were very close to expert judgments; if high, the tapper's placements were randomly distributed compared to the expert judgments.

This measure leaves open an important aspect of beat-tracking, which is regularity. As described in the qualitative results, the algorithm sometimes demonstrates unusual behavior by switching from one tempo to another, or from off-the-beat to on-the-beat, in the middle of a trial. To evaluate the regularity of tapping, the variance of interonset interval was calculated for each trial-by-subject, each trial by the model, and each trial by the expert. Note that, as described above, the human subjects were explicitly encouraged to tap regularly.

Again, the expert's behavior is taken as ideal; if the variance is larger for some tapper than for the expert, it indicates that the tapping was irregular relative to the expert. If the variance is smaller, it indicates that the tapping was more regular than the expert (not necessarily a positive aspect in the case of changing tempi). Irregularity generally arises in this data from leaving out beats, each occurrence of which adds an inter-onset interval twice as large as the rest, increasing the variance.

6. Results and discussion

The beat-placement comparison is shown in Fig. 9. Results indicate that the performance of the algorithm in placing beats in logical locations was at least comparable to the human subjects tested for all the musical cases; in four of the seven cases, the model was the most or second-most accurate tapper. This indicates that whenever a beat position was chosen by the algorithm, the position was very close to the ideal beat position as determined by the expert judgment.

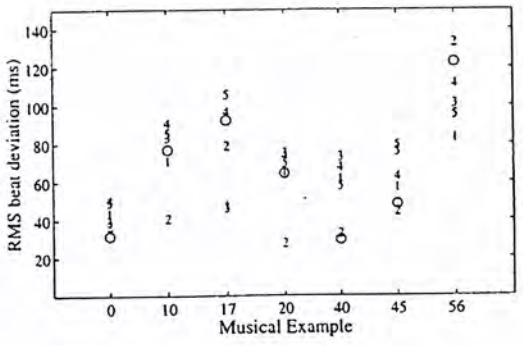


FIG. 9. Scatter plot of human (subj. number) and model (O) beat position accuracy for each of the seven experimental trials. Trial '0' corresponds to the click-track step function. Each point measures how accurate that subject was, relative to the expert, in placing beats in time. The expert judgments are at zero variance for each column. For each trial, the algorithm beat position was at least comparable to the performance of the human subjects. Overall, the algorithm performance showed a highly significant positive correlation with the human subject performance [$r=0.814$; $p(\mathrm{df}=5)<0.015$].

The regularity comparison is shown in Fig. 10. Results here indicate that the algorithm was as regular as a human listener for five of the seven trials, and less consistent for two of the trials. In one case, it and several of the human subjects were more consistent than the expert. More *post hoc* analysis is necessary to understand why the algorithm performance is irregular in these trials; preliminary results suggest that these two stimuli have relatively slow onsets carrying the beat (violins in one case, electronically gated drum sounds in the other).

These two results are consistent with the qualitative re-

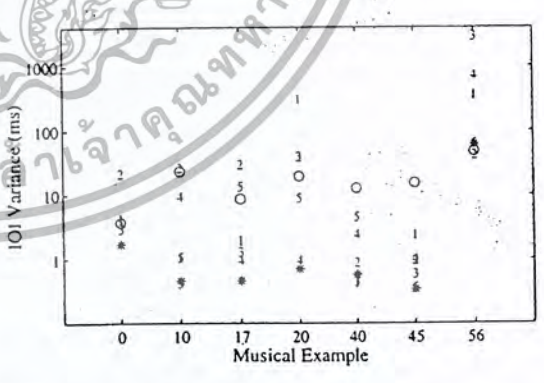


FIG. 10. Scatter plot of human (subj. number), model (O), and expert (*) IOI variances for each of the seven experimental trials. Trial "0" corresponds to the click-track step function. Each point shows the regularity of tapping of a subject for one trial; large values represent less regular tapping. For trials #40 and #45, the algorithm was not as consistent in tapping as a human listener. Overall, the algorithm performance showed a highly significant positive correlation with the human subject performance, and both the algorithm and the human subjects showed highly significant positive correlations with the expert judgement [$r=0.889$, $r=0.863$, $r=0.995$, respectively; $p(\mathrm{df}=5)<0.01$ in each case].

sults described above. When the algorithm chooses to place a beat, it does so with great accuracy and musical relevance; however, for certain musical excerpts, it is somewhat inconsistent in its tapping regularity. That is, for these examples, it drops beats or shifts phase more often than a human listener. This is not a bad result, because it is exactly this inconsistency which could best be addressed by including high-level information in the model (such as simply including instructions to "try to tap regularly").

V. DISCUSSION

In previous sections, the construction of a beat-tracking system has been approached from a largely empirical perspective. However, it is also valuable to compare the resulting algorithm to previous work on pulse perception in humans.

A. Processing level

Perhaps the most obvious difference between the method presented here and much of the previous work on beat-tracking is that this algorithm knows almost nothing about musical timbre, genres, or even notes or onsets. This approach to tempo analysis might be called a "perceptual model" of tempo, to contrast it with cognitive structuralist models.

That is to say, in models such as Povel and Essens (1985), Desain (1995), or Goto (in press), there are two stages of processing represented (the first is implicit in the Povel/Essen and Desain models). The first stage processes the acoustic stream, classifying the various pieces of sound into onsets and time intervals, separating the streams of sound, and understanding the accent structure and timbre of various components. Then, the second stage places these events in relationship to each other in order to determine the tempo and phase of the signal.

In contrast to this, the model presented here agrees with the viewpoint of Todd (1994), in which tempo and rhythm are low-level "perceptual judgments" about sound, with little cognition or memory required for processing. This viewpoint is intuitively appealing for at least one major reason, which is that certain features of tempo and beat are processed in non-attended auditory streams. Music listeners, even nonmusicians, often have the experience of conducting a conversation and suddenly realizing that they have been tapping their foot to background music. If the foot-tapping process requires cognitive structuring of the input data, it seems likely that other cognitive hearing tasks such as speech-understanding would interfere.

The finding of Levitin and Cook (1996) that there is a great ability for listeners to learn and remember absolute musical tempo implies that tempo is a simple, low-level perceptual quality. The body of initial work on rhythm perception in non-human animals (for example, Hulse *et al.*, 1984) would seem to imply similar conclusions.

The resemblance between the algorithm as drawn in Fig. 3 and modern models of pitch hearing is striking. Both models contain frequency-decomposition front ends followed by temporal integration. This comparison is explored in depth in Scheirer (1997) and leads to the question of whether pitch and tempo perception might be related auditory phenomena.

Studies such as that of Povel and Essens (1985) have demonstrated convincingly that beat perception may be explained with a model in which a perceptual clock is aligned with the accent structure of the input. A clock model is fully compatible with the method proposed here; it seems natural and intuitive to posit such an internal clock. However, the Povel and Essens model of clock induction, and similarly the Parncutt model, relies heavily on structural qualities of the input, such as a sophisticated model of temporal accent, to function.

Todd has argued that such phenomena do not need to be modeled cognitively, but rather can be explained as natural emergent qualities of known psychoacoustic properties of masking and temporal integration. This model agrees here as well, for it has demonstrated empirically that musical signals can be accurately beat-tracked without any such factors explicitly taken into account. However, a more thorough evaluation of this model would include testing it on the unusual and difficult sequences tested in the course of developing accent models, to determine if changes to weighting factors or integration constants need to be made in order to replicate these psychophysical effects.

B. Prediction and retrospection

Desain's recent work on beat-tracking has included valuable discussion of the role of prediction and retrospection in rhythmic understanding. Clearly, prediction is a crucial factor in an accurate model of human rhythm perception, as simply to synchronize motor motion (like foot-tapping) with an auditory stream requires prediction. There is a pleasing symmetry between Desain's "complex expectancy" curves and the phase-prediction vectors extracted here from the comb filter delay lines (as in Fig. 7).

Desain, citing Jones and Boltz (1989), draws attention to the utility of considering prediction and retrospection to be similar aspects of a single process. "Retrospection" refers to the manner in which new stimulus material affects the memory of previous events. Although there is no retrospection included in the model—remembrance would seem to be an inherently cognitive process—the phase-prediction curves could be used as input for this process as well.

When evaluating this model, it is important to keep in mind the complexity of introspection on musical phenomena. Although after-the-fact, listeners have made a rhythmic model of the very beginning of a musical phrase, it is clear that this model must have arisen via retrospection, for there is not enough information in the signal alone to form it progressively. Simply because a listener feels that he "understands" the rhythm of the beginning of a musical segment does not mean that the beginning itself contains sufficient information to allow such understanding.

C. Tempo versus rhythm

The effects which are not explained with this model are those related to grouping of stimuli into a rhythmic hierarchy. There are many known effects in this area, ranging from

the low-level, such as Povel and Okkerman's work on perceived accents in nonaccented sequences (Povel and Okkerman, 1981) to very broad theories of generative rhythmic modeling such as the well-known Lerdahl and Jackendoff work (Lerdahl and Jackendoff, 1983).

This model is compatible with and complementary to the bulk of this research, since most of the theories assume that a temporal framework has already been created. Synthesis of a model which operates from an acoustic source and one which includes musical assumptions and explanation should be possible, and would then represent a very robust theory of rhythmic understanding.

However, the model presented here should not be taken as attempting to explain rhythm perception as well as tempo; the viewpoint is rather that these processes are to some extent separable and may be addressed and modeled independently.

VI. CONCLUSION AND FUTURE WORK

An algorithm has been described which can successfully beat-track digital audio representing music of many different types. The music does not have to contain drums or any other specific timbres, and it does not have to conform to any predetermined set of musical templates. The beat-tracking procedure can be run in real-time on an advanced desktop workstation.

There are still aspects of the algorithm which are inadequately tested and understood. For example, would it be equally accurate but more efficient with a different filterbank, or could it be made more accurate in this way? What would be the implications of using a different temporal integration function, with different or more psychoacoustically accurate properties? What about using an entirely perceptually motived front end? These questions are unfortunately still unaddressed.

Errors still made by the algorithm are typically due to the inability to understand beat relationships at various tempi; that is, a human listener intuitively understands the way eighth-note patterns group to form quarter-note and half-note patterns, and while some processing of this sort is done implicitly in the resonators due to phase-locking at harmonic ratios, it would clearly make the algorithm more robust to have an explicit model of this sort of rhythmic grouping.

Perhaps the way to build a system that can track complicated beat patterns is to construct it in two layers. The lower layer would be a simple perceptual beat extraction system as described here, which finds the level at which the pulse is evenly divided in time. Then, a higher-level grouping model selects and processes the beats to form an model of the rhythmic hierarchy present in the signal, based on pattern-recognition detection of accent structures and instrumental beat patterns. Building a system in this manner would allow us to leverage much of the existing work in cognitive rhythm models to apply to the analysis of digital audio as well as symbolically represented music.

ACKNOWLEDGMENTS

Thanks are due the Interval Research Corporation for their support of this work. Also, Masataka Goto of Waseda University provided valuable discussions on rhythm-tracking and research on which to build. Dan Levitin provided the suggestion to base experimental validation on an expert listener. The suggestions of two anonymous reviewers improved the scope and quality of the paper immensely. As always, Professor Barry Vercoe and the Machine Listening Group at the Media Lab, especially Bill Gardner and Dan Ellis (now at ICSI-Berkeley), have been essential through their comments, critiques, and discussions.

Brown, J. C. (1993). "Determination of the meter of musical scores by autocorrelation," J. Acoust. Soc. Am. **94**, 1953–1957.

Desain, P. (1995). "A (de)composable theory of rhythm perception," Music Perception **9**, 439–454.

Desain, P., and Honing, H. (1992). *Music, Mind, and Machine: Studies in Computer-Music, Music Cognition, and Artificial Intelligence* (Thesis Publishers, Amsterdam).

Drake, C., Penel, A., Bigand, E., and Stefan, L. (1997). "Tapping in time with musical and mechanical sequences," in *Proceedings of the 1997 European Society for Cognition of Music*.

Goto, M., and Muraoka, Y. (in press). "Music understanding at the beat level: Real-time beat tracking for audio signals," in *Readings in Computational Auditory Scene Analysis*, edited by D. Rosenthal and H. Okuno (Erlbaum, Mahwah, NJ).

Handel, S. (1989). *Listening* (MIT, Cambridge, MA).

Hulse, S. H., Humpal, J., and Cynx, J. (1984). "Discrimination and generalization of rhythmic and arrhythmic sound patterns by European starlings (*Sturnus vulgaris*)," Music Perception **1**, 442–464.

Jones, M. R., and Boltz, M. (1989). "Dynamic attending and responses to time," Psychol. Rev. **96**, 459–491.

Large, E., and Kolen, J. F. (1994). "Resonance and the perception of musical meter," Connection Science **6**, 177–208.

Lerdahl, F., and Jackendoff, R. (1983). *A Generative Theory of Tonal Music* (MIT, Cambridge, MA).

Levitin, D. J., and Cook, P. R. (1996). "Memory for musical tempo: Additional evidence that auditory memory is absolute," Percept. Psychophys. **58**, 927–935.

Longuet-Higgens, H. C., and Lee, C. S. (1984). "The rhythmic interpretation of monophonic music," Music Perception **1**, 424–441.

Martin, K. (1996). "Automatic transcription of simple polyphonic music," J. Acoust. Soc. Am. **100**, 2813A.

Parncutt, R. (1994). "A perceptual model of pulse salience and metrical accent in musical rhythms," Music Perception **11**, 409–464.

Povel, D.-J., and Essens, P. (1985). "Perception of temporal patterns," Music Perception **2**, 411–440.

Povel, D.-J., and Okkenman, H. (1981). "Accents in equitone sequences," Percept. Psychophys. **30**, 565–572.

Rosenthal, D. (1992). "Machine rhythm: Computer emulation of human rhythm perception," Ph.D. thesis, MIT Media Laboratory.

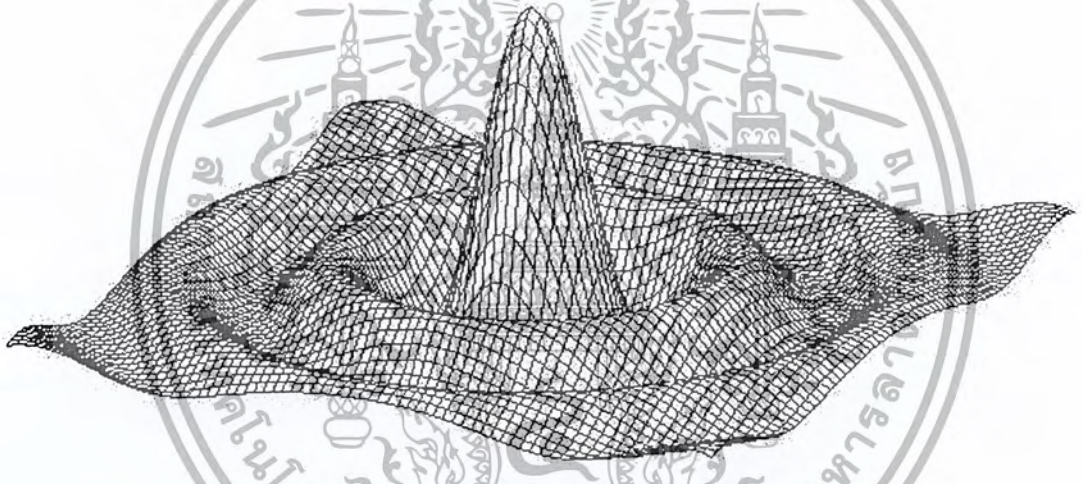
Scheirer, E. (1996). "Bregman's chimerae: Music perception as auditory scene analysis," in *Proceedings of the 1996 International Conference on Music Perception and Cognition*, Montreal.

Scheirer, E. (1997). "Pulse tracking with a pitch tracker," in *Proceedings of the 1997 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, Mohonk, NY.

Scheirer, E. (in press). "Using musical knowledge to extract expressive performance from audio recordings," in *Readings in Computational Auditory Scene Analysis*, edited by H. Okuno and D. Rosenthal (Erlbaum, Mahwah, NJ).

Shannon, R. V., Zeng, F.-G., Wyngoski, J., Kamath, V., and Ekelid, M. (**1995**). "Speech recognition with primarily temporal cues," Science **270**, 303–304.
Smith, L. S. (**1994**). "Sound segmentation using onsets and offsets," J. New Music Res. **23**, 11–23.
Todd, N. P. McA. (**1994**), "The auditory 'primal sketch': A multiscale model of rhythmic grouping," J. New Music Res. **23**, 25–70.
Vercoe, B. L. (**1994**). "Perceptually-based music pattern recognition and response," in *Proceedings of the 1994 International Conference on Music Perception and Cognition*.

BEAT DETECTION ALGORITHMS



By Frédéric Patin aka YOV408.
Contact : Frederic.Patin@supelec.fr
Homepage : http://yov408.free.fr
Last modified : Friday 21th of February 2003

DISCLAIMER

This document is to be distributed for free and without any modification from its original state. The author declines all responsibility in the damage this document or any of the things you will do with it might do to anyone or to anything. This document and any of its contents is not copyrighted and is free of all rights, you may thus use it, modify it or destroy it without breaking any international law. However according to the author's will, you may not use this document for commercial profit directly, but you may use indirectly its intellectual contents; in which case I would be pleased to receive a mail of notice or even thanks. This is my first tutorial and I am still a student, you must assume that this document is probably not free of small errors and bugs. In the same state of mind, those algorithms are not fully optimised, they are explained for pedagogical purposes and you may find some redundant computations or other voluntary clumsiness. Please be indulgent and self criticise everything you might read. Hopefully, lots of this stuff was taken in sources and books of reference; as for the stuff I did: it has proven some true efficiency in test programs I made and which work as wanted. As said in the introduction: If you have any question or any comment about this text, please send it to the above email address, I'll be happy to answer as soon as possible.

INTRODUCTION

Simulating a physical phenomena which obeys to known mathematical equations is, with a number of approximations, always feasable. But what about more abstract concepts, such as feelings, which do not follow any laws ? The simplest things we can feel are often the hardest things to capture in a program. Beat detection follows this rule : feeling the beat of a song comes naturally to humans or animals. Indeed it is only a feeling one gets when listening to a melody, a feeling which will make you dance in rhythm or hit a table with your hands on the melody beats. Therefore, how can we learn this beat detection to a machine that can only compute logical operations ? In fact there are a number of algorithms which manage to approximate, more or less accurately, this beat detection. We will first study the statistical approach of beat detection on a streaming source and secondly a filtering approach of rhythm extraction on a static song.

This guide assumes the reader has basic signal processing understanding (FFT, convolutions and correlations should sound common) maybe some stuff in statistics will also help (Variance, Average, Principal Components Analysis, will be quoted among others). The point here is not to actually write the code of these algorithms, but more to understand how they work and to be able to adapt or create the appropriate algorithm to a situation. If you have a question or a comment about this text, please send it to the above email address, I'll be happy to answer as soon as possible. Anyway, the aim here is to give more precise ideas on the subject of beat detection to the reader. Enjoy.

Statistical streaming beat detection

1 – Simple sound energy

a - A first analysis.

The human listening system determines the rhythm of music by detecting a pseudo – periodical succession of beats. The signal which is intercepted by the ear contains a certain energy, this energy is converted into an electrical signal which the brain interprets. Obviously, The more energy the sound transports, the louder the sound will seem. But a sound will be heard as a **beat** only if his energy is greatly superior to the sound's energy history, that is to say if the brain detects a **brutal variation in sound energy**. Therefore if the ear intercepts a monotonous sound with sometimes big energy peaks it will detect beats, however, if you play a continuous loud sound you will not perceive any beats. Thus, the beats are big variations of sound energy. This first analysis will bring us to our simplest model : **Sound energy peaks**.

In this model we will detect sound energy variations by computing **the average sound energy** of the signal and comparing it to the **instant sound energy**. Lets say we are working in stereo mode with two lists of values : (an) and (bn). (an) contains the list of sound amplitude values captured every Te seconds for the left channel, (bn) the list of sound amplitude values captured every Te seconds for the right channel. So we want to compute the instant energy and the average energy of the signal. The instant energy will in fact be the energy contained in 1024 samples (1024 values of a[n] and b[n]), 1024 samples represent about 5 hundreds of second which is pretty much 'instant'. The average energy should not be computed on the entire song, some songs have both intense passages and more calm parts. The instant energy must be compared to the nearby average energy, for example if a song has an intense ending, the energy contained in this ending shouldn't influence the beat detection at the beginning. **We detect a beat only when the energy is superior to a local energy average.** Thus we will compute the average energy on say : 44032 samples which is about 1 second, that is to say we will assume that the hearing system only remembers of 1 second of song to detect beat. This 1 second time (44032 samples) is what we could call the human ear energy persistence model; it is a compromise between being to big and taking into account too far away energies, and being too small and becoming to close to the instant energy to make a valuable comparison.

The history buffer where we will keep the last 44032 samples wil contain in fact too lists of samples (B[0]) and (B[1]) corresponding to the left (an) and to the right (bn) channels history.

Simple sound energy algorithm #1:

Every 1024 samples :

- Use the 1024 new samples taken in a[n] and b[n] to compute the instant energy 'e', using the following formula (i0 is the position of the 1024 samples to process) :

$$e = e_{stereo} = e_{right} + e_{left} = \sum_{k=i_0}^{i_0+1024} a[k]^2 + b[k]^2$$

(R1)

- Compute the local average energy '<E>' on the 44100 samples of a history buffer (B) :

$$<E>=\frac{1024}{44100}\times\sum_{i=0}^{44032}(B[0][i])^2+(B[1][i])^2$$

(R2)

- Shift the 44032 history buffer (B) of 1024 indexes to the right so that we make room for the 1024 new samples and evacuate the oldest 1024 samples.
- Move the 1024 new samples on top of the history buffer.
- Compare 'e' to 'C * <E>' where C is a constant which will determine the sensibility of the algorithm to beats. A good value for this constant is 1.3. If 'e' is superior to 'C * <E>' then we have a beat !

b - Some direct optimisations.

This was the basic version of the algorithm, its speed and accurecy can be improved quite easily. The algorithm can be optimised by **keeping the energy values computed on 1024 samples in history instead of the samples themselves**, so that we don't have to compute the average energy on the 44100 samples buffer (B) but on the instant energies history we will call (E). This sound energy history buffer (E) **must** correspond to approximately **1 second of music**, that is to say it must contain the energy history of 44032 samples (calculated on groups of 1024) if the sample rate is 44100 samples per second. Thus E[0] will contain the newest energy computed on the newest 1024 samples, and E[42] will contain the oldest energy computed on the oldest 1024 samples. We have 43 energy values in history, each computed on 1024 samples which makes 44032 samples energy history, which is equivalent to 1 second in real time. The count is good. The value of 1 second represents the persistance of the music energy in the human ear, it was obtain with experimentations but it may varry a little from a person to another, just adjust it as you feal. So here is what the algorithm becomes :

Simple sound energy algorithm #2 :

Every 1024 samples :

- Compute the instant sound energy 'e' on the 1024 new sample values taken in (an) and (bn) using the following formula (R1)
- Compute the average local energy <E> with (E) sound energy history buffer :

$$<E>=\frac{1}{43}\times\sum_{i=0}^{43}(E[i])^2$$

(R3)

- Shift the sound energy history buffer (E) of 1 index to the right. We make room for the new energy value and flush the oldest.

- Pile in the new energy value 'e' at E[0].
- Compare 'e' to 'C*<E>'.

c - Sensitivity detection.

The imediate draw back of this algorithm is **the choice of the 'C' constant**. For example in hno and rap music beats are quite intense and precise so 'C' should be quite high (about 1.4); ereas for rock and roll, or hard rock which contains a lot of noise, the beats are more confused and 'C' ould be low (about 1.1 or 1.0). There is a way, to make the algorithm determine automatically the good oice for the 'C' constant. We must compute **the variance of the energies** contained in the energy tory buffer (E). This variance, which is nothing but the average of (Energy Values – Energy average = -<E>), will quantify how marked the beats of the song are and thus will give us a way to compute the ue of the 'C' constant. The formula to calculate the variance of the 43 E[i] values is described below 4). Finally, the greater the variance is the more sensitive the algorithm should be and thus the smaller will become. We can choose a linear decrease of 'C' with 'V' (the variance) and for example when V 200, C ? 1.0 and when V ? 25, C ? 1.45 (R5). This is our new version of the sound energy beat ection algorithm :

Simple sound energy algorithm #3 :

Every 1024 samples :

- Compute the instant sound energy 'e' on the 1024 new samples taken in (an) and (bn) using the following formula (R1).
- Compute the average local energy <E> with (E) sound energy history buffer using formula (R3).
- Compute the variance 'V' of the energies in (E) using the following formula :

$$V = \frac{1}{43} \times \sum_{i=0}^{43} (E[i] - <E>)^2$$

(R4)

- Compute the 'C' constant using a linear degression of 'C' with 'V', using a linear regression with values (R5) :

$$C = (-0.0025714 \times V) + 1.5142857$$

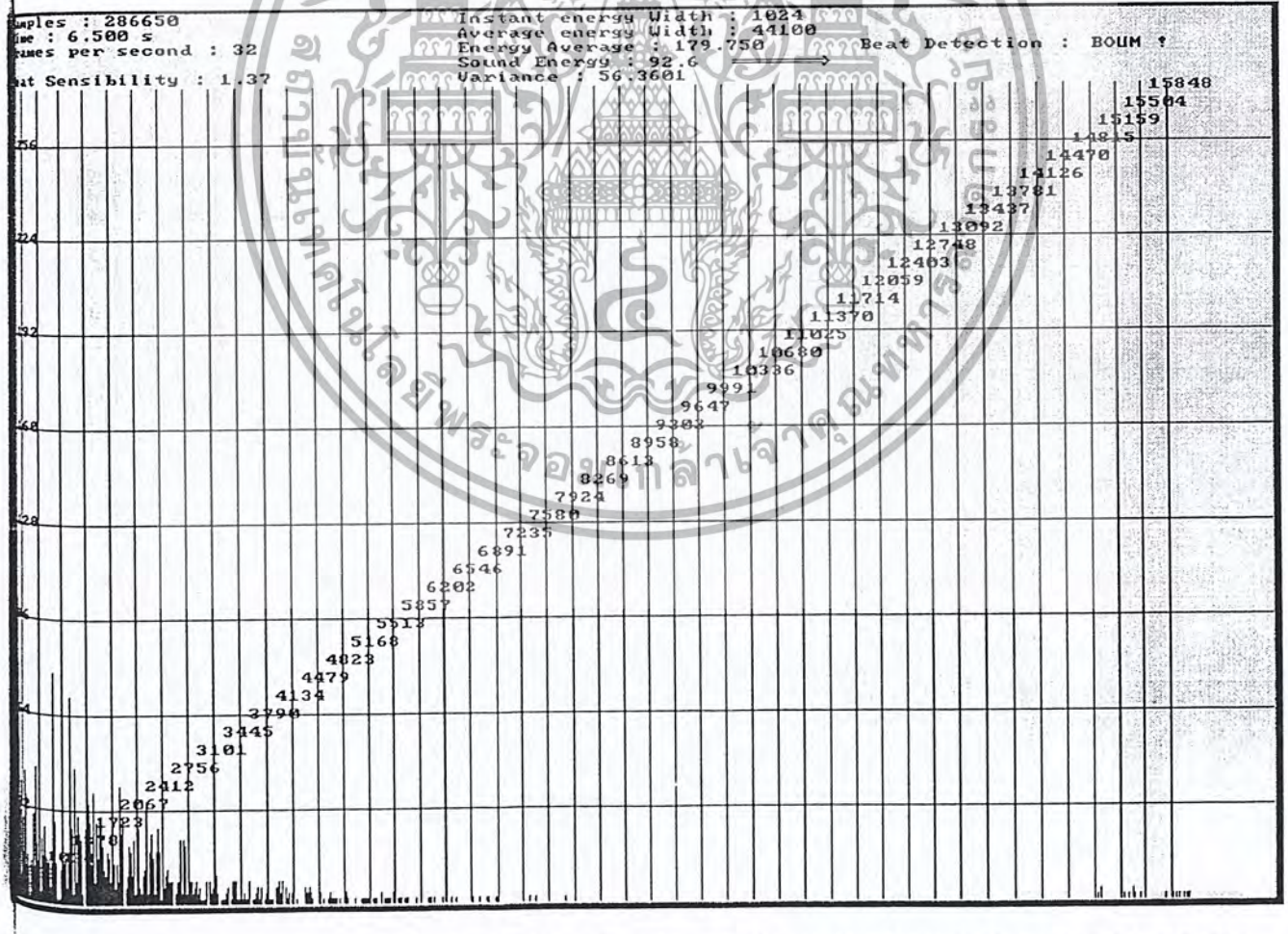
(R6)

- Shift the sound energy history buffer (E) of 1 index to the right. We make room for the new energy value and flush the oldest.
- Pile in the new energy value 'e' at E[0].
- Compare 'e' to 'C*<E>', if superior we have a beat !

Those three algorithms were tested with several types of music, among others : pop, rock, etal, techno, rap, classical, punk. The fact is the results are quite unpredictable. I will only talk about *mple beat detection algorithm #3* as the #2 and #1 are only pedagogical intermediates to get to the *#3*. early, **the beat detection is very accurate and sounds right with techno and rap**, the beats are ry precise and the music contains very few noise. The algorithm is quite satisfying for that kind of music d if you aim to use beat detection on techno you can stop reading here, the rest won't change anything your beat detection. However, even if the improvement of the dynamic 'C' calculations ameliorates ngs alot, **the beat detection on punk, rock and hard rock, is sometimes quite approximate**. We n feel it doesn't really get the rythm of the song. Indeed the algorithm detects energy peaks. metimes you can hear a drum beat which is sank among other noises and which goes trough the orithm without being detected as a beat.

To explain this phenomena lets say a guitare and flute make alternatively an amplitude nstant note. Each time the first finishes the other starts. The note made by the guitare and the note ade by the flute have the same energy but the ear detects a certain rhythm because the notes of the struments are at different pitch. For our algorithm (who is one might say colorblind) it is just an plitude constant noise with no energy peaks. This partly explains why the algorithm doesn't detect ecisely beats in songs with a lot of instruments playing at different rythms and simultaneously. Our next alysis will make us walk through this difficulty.

Comparing the results we have obtained with the *Simple beat detection algorithm #3* to its mputing cost, this algorithm is very efficient. If you are not looking for a perfect beat detection than I ecommend you use it. Here is a screenshot of a program I made using this algorithm. You fill find the naries and the sources on my homepage.



The spectrum analyser is not useful for the beat detection it is only for visual matters, but you can see at the top some of the parameters the program computes to execute the algorithm. (fig1)

2 – *Frequency selected sound energy.*

a - The idea and the algorithm.

The issue with our last analysis of beat detection is that it is colorblind. We have seen that this could raise quite a few problems for noisy like songs in rock or pop music. What we must do is give our algorithm the abbility to determine on which frequency subband we have a beat and if it is powerful enough to take it into account. Basically we will try to detect **big sound energy variations in particular frequency subbands**, just like in our last analysis; unless this time we will be able to seperate beats regarding their color (frequency subband). Thus If we want to give more importants to low frequency beats or to high frequency beats it should be more easy. Notice that the energy computed in the time domain is the same as the energy computed in the frequency domain, so we don't have any difference between computing the energy in time domain or in frequency domain. For maths freaks this is called the Parseval Theorem.

Okay that was just a bit of sport, lets go back to the mainstream; Here is how the **frequency selected sound energy** algorithm works: The source signals are still coming from (an) and (bn). (an) and (bn) can be taken from a wave file, or directly from a streaming microphon or line input. Each time we have accumulated 1024 new samples, we will pass to the frequency domain with a **Fast Fourier Transform (FFT)**. We will thus obtain a 1024 frequency spectrum. We then divide this spectrum to however many subbands we like, here I will take 32. The more subbands you have, the more sensitive the algorithm will be but the harder it will become to adapt it to lots of different kinds of songs. Then we compute the **sound energy** contained in **each of the subbands** and we compare it to the **recent energy average corresponding to this subband**. If one or more subbands have an energy superior to their average we have detected a beat.

The great progress with the last algorithm is that we now know more about our beats, and thus we can use this information to change an animation, for example. So here is more precisely the *Frequency selected sound energy algorithm #1* :

Frequency selected sound energy algorithm #1 :

Every 1024 samples :

- Compute the FFT on the 1024 new samples taken in (an) and (bn). The FFT inputs a complex numeric signal. We will say (an) is the real part of the signal and (bn) the imagenary part. Thus the FFT will be made on the 1024 complex values of :

$$(a_n) + i \times (b_n)$$

You can find FFT tutorials and codes in C, Visual Basic or C++ on my homepage in the 'tutorials' section or by typing 'FFT' on www.google.com

- From the FFT we obtain 1024 complex numbers. We compute the square of their module and store it into a new 1024 buffer. This buffer (B) contains the 1024 frequency amplitudes of our signal.

- Divide the buffer into 32 subbands, compute the energy on each of these subbands and store it at (Es). Thus (Es) will be 32 sized and Es[i] will contain the energy of subband 'i' :

$$Es[i] = \frac{32}{1024} \times \sum_{k=i\times32}^{(i+1)\times32} B[k]$$

(R7)

- Now, to each subband 'i' corresponds an energy history buffer called (Ei). This buffer contains the last 43 energy computations for the 'i' subband. We compute the average energy <Ei> for the 'i' subband simply by using :

$$< Ei >= \frac{1}{43} \times \sum_{k=0}^{42} Ei[k]$$

(R8)

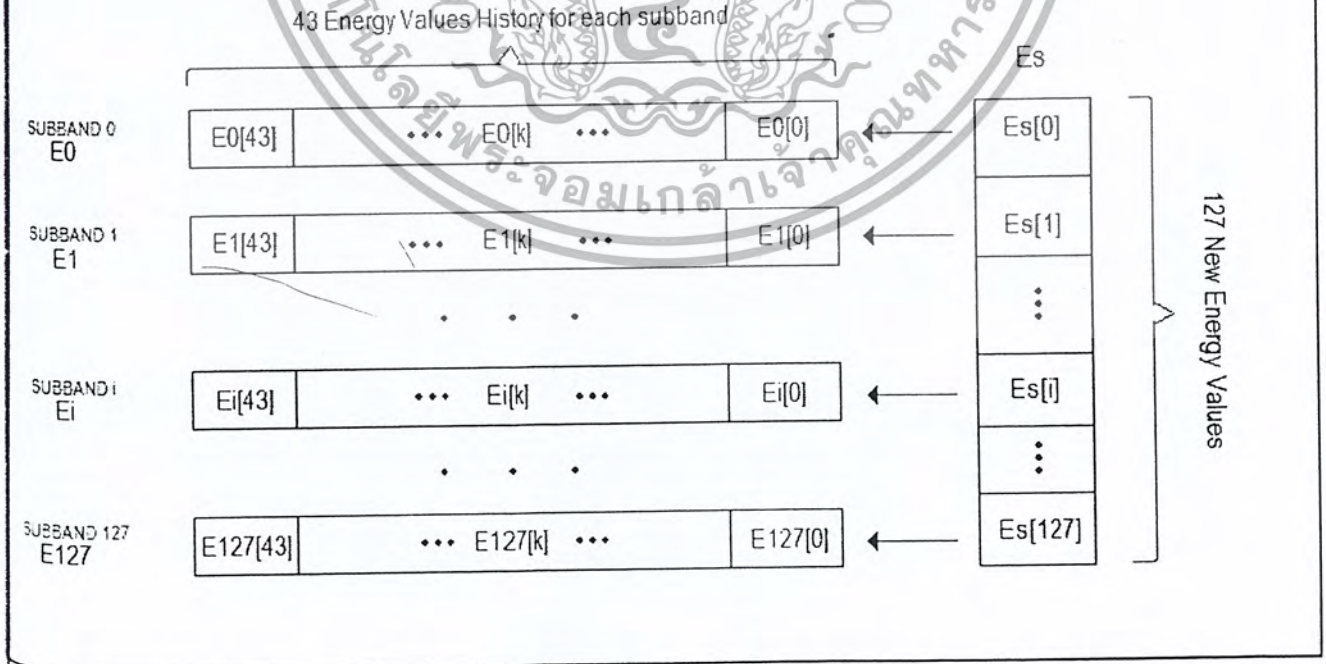
- Shift the sound energy history buffers (Ei) of 1 index to the right. We make room for the new energy value of the subband 'i' and flush the oldest.
- Pile in the new energy value of subband 'i' : Es[i] at Ei[0].

$$Ei[0] = Es[i]$$

(R9)

- For each subband 'i' if Es[i] > (C*<Ei>) we have a beat !

To help out visualising how the data piles work have a look at this scheme:



This is how the energy data is organized. (fig2)

Now the 'C' constant of this algorithm has nothing to do with the 'C' of the first algorithm, ecause we deal here with separated subbands the energy varies globally much more than with orblind algorithms. Thus 'C' must be about 250. The results of this algorithm are convincing, it detects r example a symbal rhythm among other heavy noises in metal rock, and indeed the algorithm eparates the signal into subbands, therefore the **symbal rhythm cannot pass trough the algorithm thout being recognized because it is isolated in the frequency domain from other sounds.** wever the complexity of the algorithm makes it useful only if you are dealing with very noisy sounds, other cases, *Simple beat detection algorithm #3* will do the job.

b - Enhancements and beat decision factors.

There are ways to enhance a bit more our *Frequency selected sound energy algorithm #1*. st we will increase the **number of subbands from 32 to 64**. This will take obviously more computing re but it will also give us more precision in our beat detection. The second way to develop the curacy of the algorithm uses the defaults of human ears. Human hearing system is not perfect; in fact transfer function is more like a low pass filter. We hear more easily and more clearly low pitched ises than high pitch noises. This is why it is preferable to make a logarithmic repartition of the ubbands. That is to say that subband 0 will contain only say 2 frequencies whereas the last subband, ll contain say 20. **More precisely the width 'wi' of the 'n' subbands indexed 'i' can be obtained sing this argument :**

- *Linear increase of the width of the subband with its index :*

$$wi = a \times i + b$$

(R10)

- *We can choose for example the width of the first subband :*

$$w1 = a + b$$

(R11)

- *The sum of all the widths must not exceed 1024 ((B)'s size) :*

$$\sum_{i=1}^{n} wi = 1024 = n \cdot b + a \cdot \sum_{i=1}^{n} i = n \cdot b + a \cdot \frac{n \cdot (n+1)}{2} = 1024$$

(R12)

Once you have equations (R11) and (R12) it is fairly easy to extract 'a' and 'b', and thus to d the law of the 'wi'. This calculus of 'a' and 'b' must be made manually and 'a' and 'b' defined as nstants in the source; indeed they do not vary during the song.

So in fact in *Frequency selected sound energy algorithm #1*, all we have to modify is the mber of subbands we will take equal to 64 and the (R7) relation. This relation becomes:

$$Es[i] = \frac{wi}{1024} \times \sum_{k=\sum_{j=1}^{i-1} wj}^{\sum_{j=1}^{i} wj} B[k]$$

(R7) '

It may seem rather complicated but in fact it is not (☺). Replacing this relation (R7) with R7)' we have created *Frequency selected sound energy algorithm #2*. If you have musics with very tight rd rapid beats, you may want to compute the stuff more frequently than every 1024 samples, but this is rly for special cases, normally the beat should not be shorter than 1/40 of second.

Using the advantages of Frequency selected beat detection you can also enhance the beat ecision factor. Up to now it was based on a simple comparison between the instant energy of the bband and the average energy of the subband. This algorithm enables you to decide beats differently. ou may want for examples to cut beats which correspond to high pitch sounds if you run techno music you may want to keep only [50-4000Hz] beats if you are working with speech signal. This algorithm as the advantage of being perfectly adaptable to any kind or category of signal which was not the case *Simple beat detection algorithm #3*. Notice that the correspondants between index 'i' of the FFT ansform and real frequency is given by formula :

- If 'i' < 'N/2' then :

$$f = \frac{i \times fe}{N}$$

(R13)

- Else 'i' >= 'N/2' then :

$$f = \frac{(N-i) \times fe}{N}$$

(R14)

So 'i' is the index of the value in the FFT output buffer, N is the size of the FFT transform ere 1024), fe is the sample frequency (here 44100). Thus index 256 corresponds to 10025 Hz. This rmula may be useful if you want to create your own subbands and you want to know what the rrespondants **between indexes and real frequency** are.

Another way of filtering the beats, or selecting them, is choosing only those which are arked and precise enough. As we have seen before, **to detect the accuracy of beats we must ompute the variance of the energy histories for each subband**. If this variance is high it means that e have great differences of energy and thus that the beat are very intense. Thus all we have to do is mpute this variance for each subband and add a test in the beat detection decision. To the "Es[i]>

"<Ei>)" condition we will add "and V((Ei))>V0". V0 will be the variance limit, with experience 150 is a asonable value. Now the V((Ei)) value is easy to compute, just follow the following equality if you don't e how :

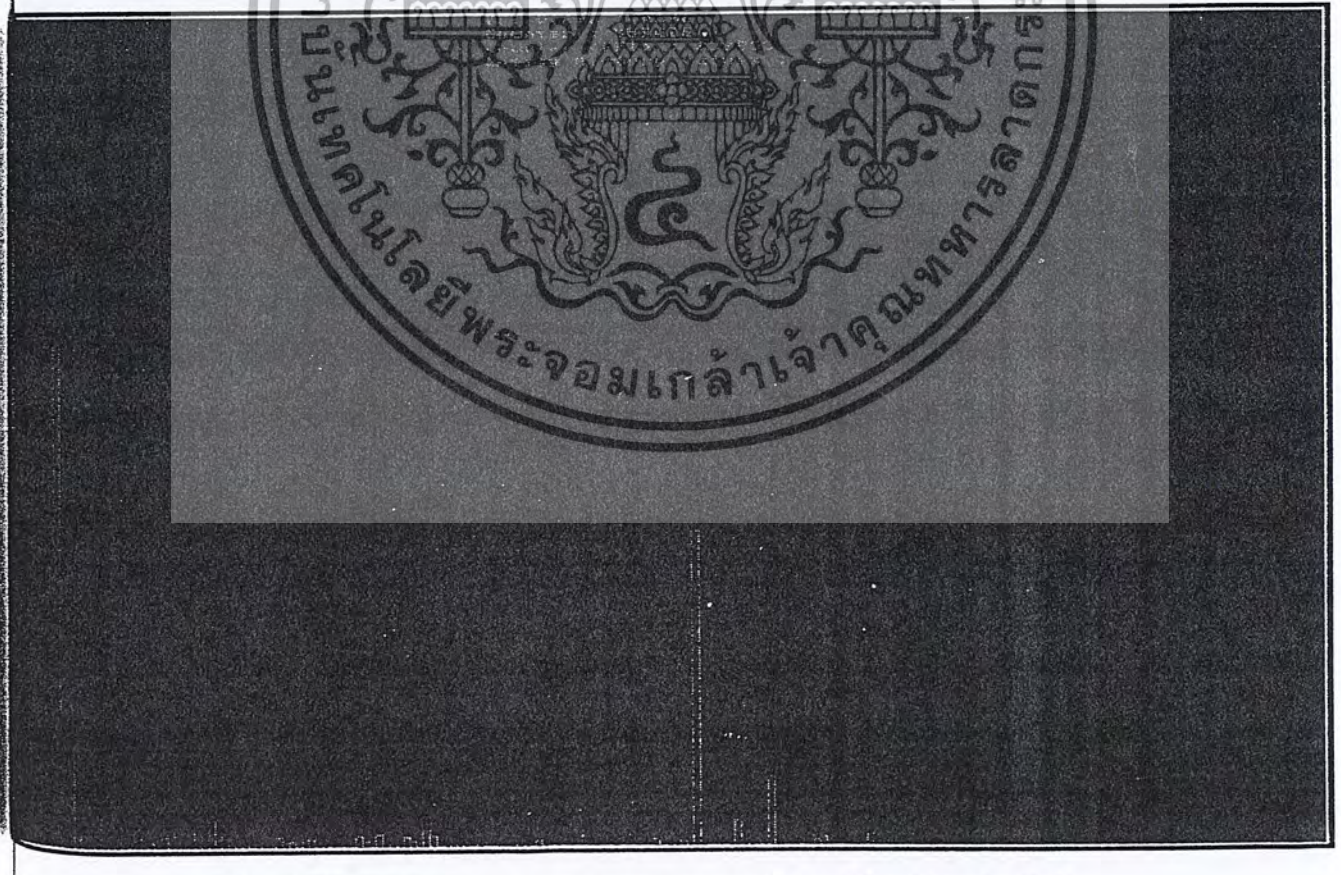
$$V((Ei)) = \frac{1}{43} \times \sum_{k=0}^{42} \left(Ei[k] - <Ei> \right)^2$$

(R15)

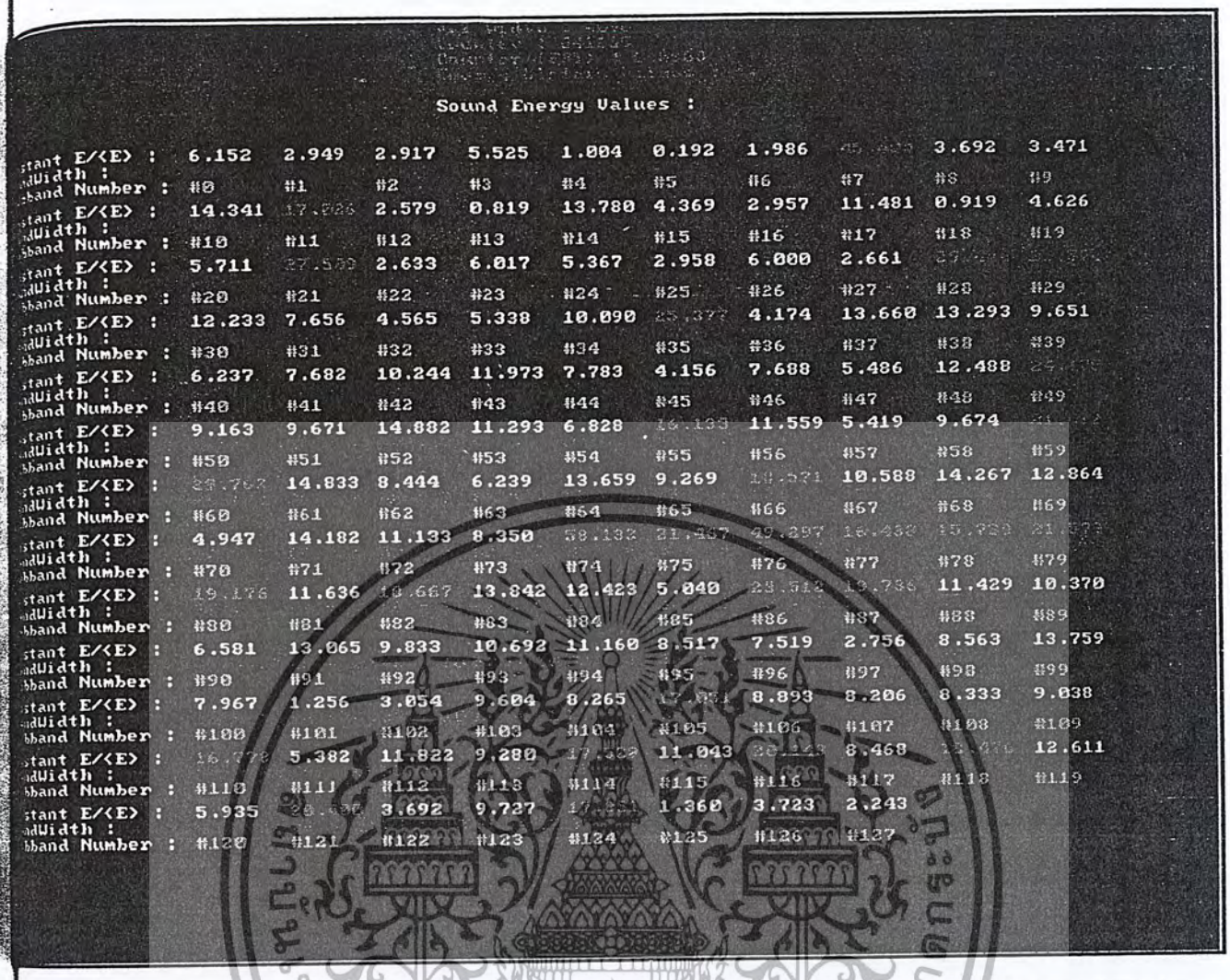
The last (finally) way to enhance your beat detection, is to make the source signal pass rough a **derivation filter.** Indeed differentiating the signal makes big variations of amplitude more arked and thus makes energy variations more marked and recognisable later in the algorithm. I haven't ed this optimisation but according to some sources this is quite useful. If you try it please give me your inion on it!

Concerning the results of the *Frequency selected sound energy algorithm #2* I must admit ey are way more satisfying than the *Simple sound energy algorithm #3*. In a song the algorithm atches the bass beats as well as the tight cymbals hits. I insist on the fact that you may also select the eats in very different ways, which becomes quite useful if you know you are going to run techno music th low pitch beats for example. You may also select the beats differently according to there accuracy th the variance criteria. There are many other ways to decide beats; it is up to you to explore them and d the one which fits the most your needs.

I used *Frequency selected sound energy algorithm #2* algorithm in a demo program of hich you can see some screenchots just below. One can see quite clearly that there is a beat in low equencies (probably a bass or drum hit) and also a high pitch beat (probably a cymbal or such):



The histogram represents the instant energy of the 128 subbands. (fig3)



On this screenshot you can see the 128 subbands E/<E> ratios, and the bandwidth of the subbands. When the algorithm detects a beat in a subband, it overwrites the ratio in red. (fig4)

The reason why I called this part of the document, 'Statistical streaming beat detection', is at the energy can be considered as a random variable, of which we have been calculating values over me. Thus we could interpret those values as a sampling for the statistical analysis of a random variable. ut one can push this approach further. When we have separated the energies histories into 128 bbands we created in fact 128 random energy variables. We can then apply some of the general tatistical methods of analysis. For example, the principal components analysis method will enable you determine if some of the subbands are directly linked or independent. This would help us to regroup me subbands which are directly linked and thus make the beat decision more efficient. However, this ethod is basically just far too computing expensive and maybe just too hard to implement comparing to e results we want. If you are looking for a really good challenge in beat detection you could push in this rection.

Filtering rhythm detection

1 – Derivation and Comb filters.

a - Intercorrelation and train of impulses.

While in the first part of this document we had seen beat detection as a statistical sound energy problem, we will approach it here as a **signal processing issue**. I haven't tried myself this algorithm but I greatly inspired myself of some work which was done and tested with this algorithm (see the sources section), so it really works but I won't detail so much its implementation as I did in the first part. The basic idea is the following. **If we have two signals x(t) and y(t)**, we can evaluate a function which will give us an idea of how **much those two signals are similar**. This function called 'inter correlation function' is the following:

$$\gamma_{xy}(\alpha,\beta)=\int_{-\infty}^{+\infty}\left[x(t)\cdot y(\alpha\cdot(t-\beta))\cdot dt\right]$$

(R16)

For our purposes we will always take alpha=1. This function quantifies the energy that can be exchanged between the signal x(t) and the signal y(t – ß), it gives an evaluation of how much those two functions look like each other. The role of the 'ß' is to eliminate the time origin problem; two signals could be compared without regarding their possible time axis translation. As you may have noticed, we are creating this algorithm for a computer to execute; basically we have to re-write this formula in discrete mode, it becomes:

$$\gamma_{xy}[\beta]=\sum_{k=-\infty}^{+\infty}\left[x[k]\cdot y[k-\beta]\right]$$

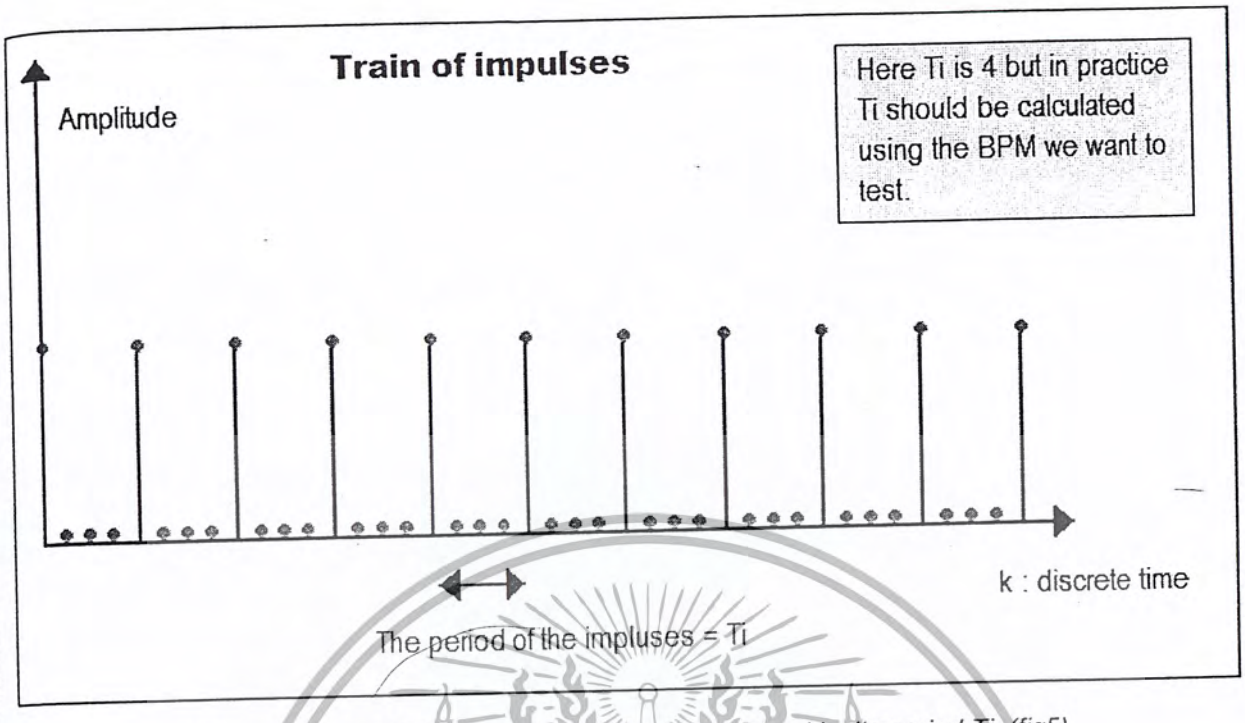
(R17)

Now the point is we can valuate the similarity of our song and of a train of impulses using this function. The train of impulses being at a known frequency (the beats per minute we want to test) is what we call a comb filter. **The energy of the ? function gives us an evaluation of the similarity between our song and this train of impulses, thus it quantifies how much the rhythm of the train of impulses is present in the song**. If we compute the energy of the ? function for different train of impulses, we will be able to see for which of them the energy is the biggest, and thus we will know what is the principal Betas Per Minute rate. This is called **combfilter processing**. We will note 'Ey' this energy, where x[k] is our song signal and y[k] our train of impulses:

$$Ey=\sum_{\beta=-\infty}^{+\infty}\gamma_{xy}[\beta]^2$$

(R18)

By the way, here is what a train of impulses looks like :



This is the train of impulses function, it is caracterised by its period Ti. (fig5)

So the period Ti of the impulses must correspond to the beats per minutes we want to test our song. The formula that links a beat per minute value with the Ti in discrete time is the following:

$$Ti = \frac{60}{BPM} \times fs$$

(R19)

- fs is the sample frequency, if you are using good quality wave files, it is usually of 44100. BPM is the beats per minute rate. Finally, Ti is the number of indexes between each impulse.

Now because it is quite computing expensive to compute the (R17) formula, **we pass to the frequency domain with a FFT and compute the product of the X[v] and Y[v]** (FFT's of x[k] and y[k]). Then we compute the energy of the ? function directly in the frequency domain, thus Ey is given by the following formula:

$$E_y = \sum_{k=0}^{N} \left| X\left[k\right] \cdot Y\left[k\right] \right|^2$$

(R17)'

b - The algorithm.

Note that this algorithm is much too computing expensive to be ran on a streaming source, on a song in a whole. It is executed on a part of the song, like 5 seconds taken somewhere in the middle. Thus we assume that the tempo of the song is overall constant and that the middle of the song is tempo-representative of the rest of the song. So finally here are the steps of our algorithm:

Derivation and Combfilter algorithm #1 :

- Choose roughly 5 seconds of data in the middle of the song and copy it to the 'a[k]' and 'b[k]' lists. The dimension of those lists is noted N. As usual 'a[k]' is the array for the left values, and 'b[k]' is the array for the right values.
- Compute the FFT of the complex signal made with a[k] for the real part of the complex signal and b[k] the imaginary part of the complex signal as seen in *Frequency selected sound energy algorithm #1.* Store the real part of the result in ta[k] and the imaginary in tb[k].
- For all the beats per minute you want to test, for example from 60 BPM to 180 BPM per step of 10, we will note BPMc the current BPM tested :
  - Compute the Ti value corresponding to BPMc using formula (R19).
  - Compute the train of impulses signal and store it in (l) and (j). (l) and (j) are purely identical, we take two lists of values to have a stereo signal so that we don't have dimension issues further on. Here is how the train of impulses is generated :

$$
\begin{aligned}
&for(k=0;k<N;k++)\{\\
&\quad if((k\%Ti)==0)\\
&\quad\quad l[k]=j[k]=AmpMax;\\
&\quad else\\
&\quad\quad l[k]=j[k]=0;\\
&\}
\end{aligned}
$$

  - Compute the FFT of the complex signal made of (l) and (j). Store the result in tl[k] and tj[k].
  - Finally compute the energy of the correlation between the train of impulses and the 5 seconds signal using (R17)' which becomes :

$$E_{BPMc}=\sum_{k=0}^{N}\left|(ta[k]+i\cdot tb[k])\cdot(tl[k]+i\cdot tj[k])\right|$$

(R20)

  - Save E(BPMc) in a list.

- The rhythm of the song is given by the BPMmax, where the max of all the E(BPMc) is taken. We have the beat rate of the song !

The AmpMax constant which appears in the train of impulse generation, is given by the mple size of the source file. Usually the sample size is 16 bits, for a good quality .wav. If you are rking in non-signed mode, the AmpMax value will be of 65535, and more often if you work with 16 bits ned samples the AmpMax will be 32767.

One of the first ameliorations that could be given to *Derivation and Combfilter algorithm #1* ndeed adding a **derivation filter** before the combfilter processing. **To make beats more detectable, differentiate the signal**. This accentuates when the sound amplitude changes. So instead of dealing h a[k] and b[k] directly we first transform them using the formula below. This modification added before e second step of *Derivation and Combfilter algorithm #1* constitutes *Derivation and Combfilter algorithm*

$$\forall k \in [0, N-1]$$
$$a[k] = fs \cdot (a[k+1] - a[k])$$
$$b[k] = fs \cdot (b[k+1] - b[k])$$

(R21)

As I said before, I haven't tested this algorithm myself; I can't compare its results with the gorithms of the first part. However, throughout the researches I made this seems to be the algorithm iversally accepted as being the most accurate. It does have disadvantages: Great computing power nsumption, can only be done on a small part of a song, assumes the rhythm is constant throughout the ng, no possible streaming input (for example from a microphone), the signal needs to be known in dvance. However it returns a deeper analysis than the first part algorithms, indeed for each BPM rate u have a value of its importance in the song.

2 – *Frequency selected processing*

a - Curing colorblindness

As in the first part of the tutorial, we have the same problem our last algorithm doesn't make e difference between cymbals beat or a drum beat, so the beat detection becomes a bit dodgy on very isy music like hard rock. To heal our algorithm we will proceed as we had done in *Frequency selected und energy algorithm #1* **we will separate our source signal into several subbands**. Only this time e have much more computing to do on each subband so we will be much more limited in their number. I ink that 16 is a good value. **We will use a logarithmic repartition of these subbands** as we had done efore.

Let's modify the *Derivation and Combfilter algorithm #2*. The values particular to a subband ill be characterized with a little 's' at the end of the name of the variable. So we will separate ta[k] and b[k] into 16 subbands. Each subbands values array will be called tas[k] and tbs[k] ('s' varies from 1 to 16).

- Choose roughly 5 seconds of data in the middle of the song and copy it to the 'a[k]' and 'b[k]' lists. The dimension of those lists is noted N. As usual 'a[k]' is the array for the left values, and 'b[k]' is the array for the right values.
- Differentiate the a[k] and b[k] signals using (R21).
- Compute the FFT of the complex signal made with a[k] for the real part of the complex signal and b[k] the imaginary part of the complex signal as seen in *Frequency selected sound energy algorithm #1.* Store the real part of the result in ta[k] and the imaginary in tb[k].
- Generate the 16 subband array values tas[k] and tbs[k] by cutting ta[k] and tb[k] with a logarithmic rule.
- For all the subbands tas[k] and tbs[k] (s goes from 1 to 16). Ws is the length of the 's' subband.
- For all the beats per minute you want to test, for example from 60 BPM to 180 BPM per step of 10, we will note BPMc the current BPM tested :

  - Compute the Ti value corresponding to BPMc using formula (R19).
  - Compute the train of impulses signal and store it in (l) and (j). (l) and (j) are purely identical, we take two lists of values to have a stereo signal so that we don't have dimension issues further on. Here is how the train of impulses is generated :

$$
\begin{aligned}
&for(k=0;k<ws;k++)\{\\
&\quad if((k\%Ti)==0)\\
&\qquad l[k]=j[k]=AmpMax;\\
&\quad else\\
&\qquad l[k]=j[k]=0;\\
&\}
\end{aligned}
$$

  - Compute the FFT of the complex signal made of (l) and (j). Store the result in tl[k] and tj[k].
  - Finally compute the energy of the correlation between the train of impulses and the 5 seconds signal using (R17)' which becomes :

$$E_{BPMc,s}=\sum_{k=0}^{ws}\left|(tas[k]+i\cdot tbs[k])\cdot(tl[k]+i\cdot tj[k])\right|$$

(R22)

- Save E(BPMc,s) in a list.

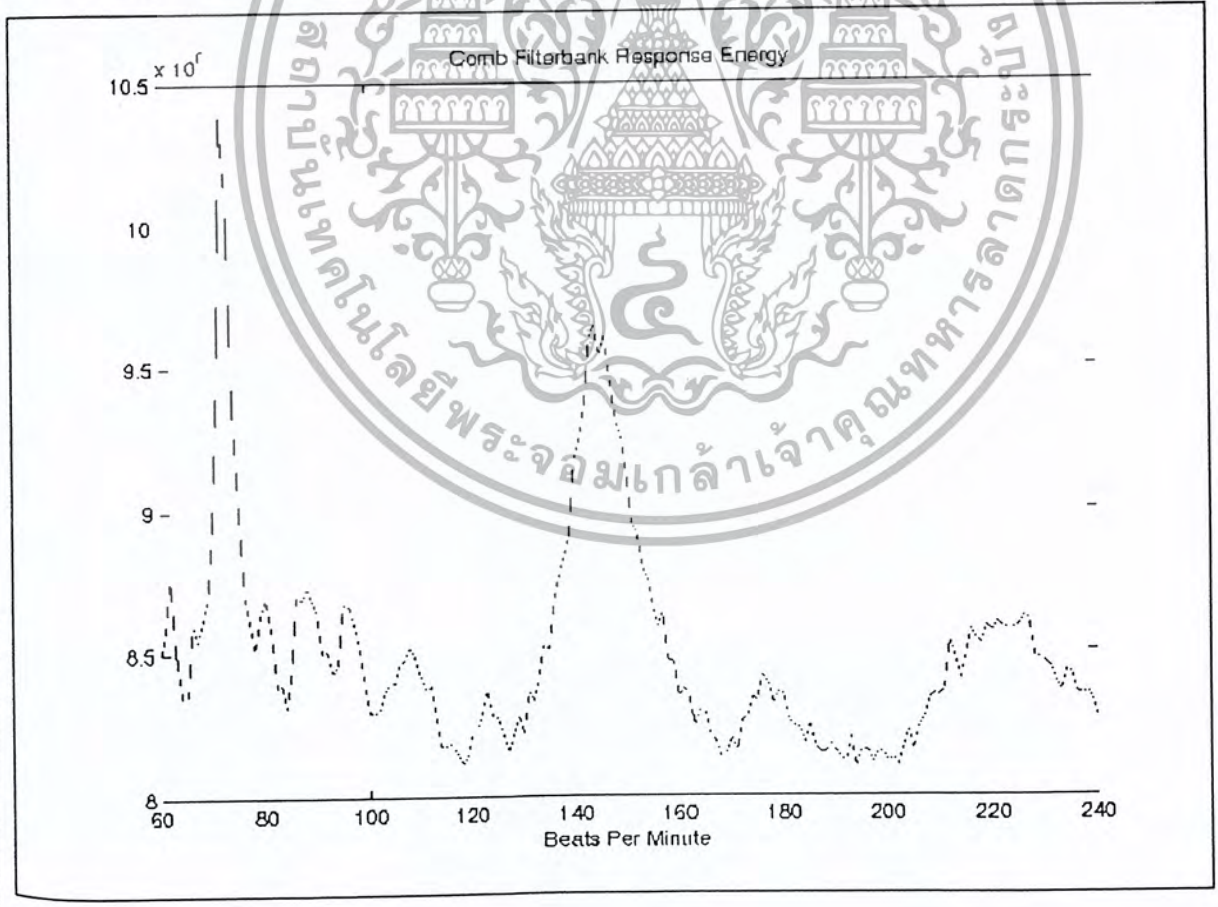
- The rhythm of the subband is given by the BPMmaxs, where the max of all the E(BPMc,s) is taken (s is fixed). We will call this max EBPMmaxs. We have the beat rate of the subband we store it in a list.
- We do this for all the subbands, we have a list of BPMmaxs for all subbands, we can than decide which one to consider.

As in *Frequency selected sound energy algorithm #2* we will then be able to decide the thm according to frequency bandwidth criteria. Or, if you want to take all the subbands into count you can compute the final BPM of the song, by calculating the barycentre of all the Mmaxs affected with their max of E(BPMc,s). Like this:

$$BPM = \frac{1}{\sum_{s=1}^{16}(EBPM\max s)} \sum_{s=1}^{16}(EBPM\max s)\bullet(BPM\max s)$$

(R23)

I must admit I haven't concretely tested this algorithm. Others have done this already and re an overview of the results for *Derivation and Combfilter algorithm #2* (source: p://www.owlnet.rice.edu/~elec301/Projects01/beat_sync/beatalgo.html).



This is plot of the EBPMc values function of BPMc. The algorithm will take the max of the EBPMc lue to find the final BPM of the song. Here we see this max is reached for BPMc=75. The final BPM is 75 ! (fig6)

CONCLUSION

Finding algorithms for beat detection is very frustrating. It seems so obvious to us, humans, to hear those beats and somehow so hard to formalise it. We managed to approximate more or less accurately and more or less efficiently this beat detection. But the best algorithms are always the ones you make yourself, the ones which are adapted to your problem. The more the situation is precise and defined the easier it is! This guide should be used as a source of ideas. Even if beat detection is far from being a crucial topic in the programming scene, it has the advantage of using lots of signal processing and mathematical concepts. More than an end in itself, for me, beat detection was a way to train to signal processing. I hope you will find some of this stuff useful.

SOURCES AND LINKS

- http://www.owlnet.rice.edu/~elec301/Projects01/beat_sync/beatalgo.html : This site greatly inspired me for the second part of the tutorial, very well explained.
- http://www.cs.princeton.edu/~gessl/papers/amta2001.pdf : Audio analysis using the discrete wavelet transform.
- Tempo and beat analysis of acoutsic musical signals, by Eric D., schreiner.
- Pacemaker, Parviainen, Olli.

ภาคผนวก ข

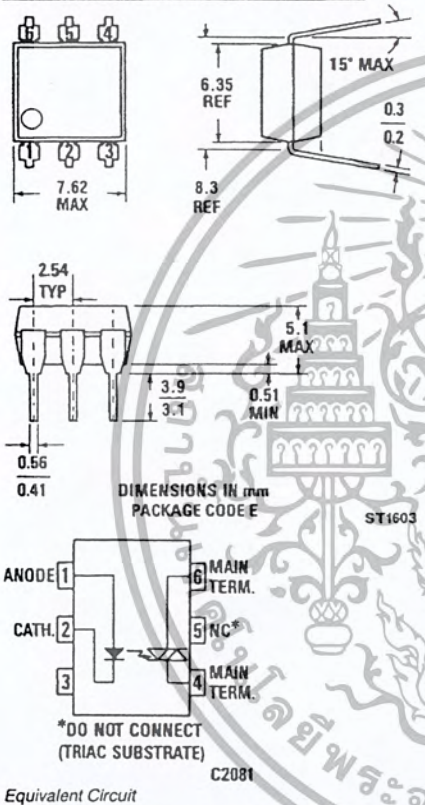
ข้อมูลอุปกรณ์อ้างอิง

QT OPTOELECTRONICS

NON-ZERO-CROSSING TRIACS

MOC3020 MOC3021 MOC3022 MOC3023

PACKAGE DIMENSIONS



Equivalent Circuit

DESCRIPTION

The MOC3020, MOC3021, MOC3022 and MOC3023 are optically isolated triac driver devices. These devices contain a GaAs infrared emitting diode and a light activated silicon bilateral switch, which functions like a triac. This is designed for interfacing between electronic controls and power triacs to control resistive and inductive loads for 240 VAC operations.

FEATURES

- Excellent I_T stability—IR emitting diode has low degradation
- High isolation voltage—minimum 7500 VAC peak
- Underwriters Laboratory (UL) recognized—File #E90700

APPLICATIONS

- European applications for 240 VAC
- Triac driver
- Industrial controls
- Traffic lights
- Vending machines
- Motor control
- Solid state relay

ABSOLUTE MAXIMUM RATINGS

TOTAL PACKAGE

Storage temperature	−55°C to 150°C
Operating temperature	−40°C to 100°C
Lead temperature (soldering, 10 sec)	260°C

INPUT DIODE

Forward DC current	50 mA
Reverse voltage	3 V
Peak forward current (1 μs pulse, 300 pps)	3.0 A
Power dissipation (25°C ambient)	100 mW
Derate linearly (above 25°C ambient)	1.33 mW/°C

OUTPUT DRIVER

Off-state output terminal voltage		400 Volts
On-state RMS current	T_A=25°C	100 mA
(Full cycle, 50 to 60 Hz)	T_A=70°C	50 mA
Peak nonrepetitive surge current (PW=10 ms, DC=10%)		1.2 A
Total power dissipation (25°C ambient)		300 mW
Derate above 25°C		4.0 mW/°C

ELECTRO-OPTICAL CHARACTERISTICS 125°C Temperature Unless Otherwise Specified

INDIVIDUAL COMPONENT CHARACTERISTICS

CHARACTERISTIC	SYMBOL	MIN.	TYP.	MAX.	UNITS	TEST CONDITIONS
INPUT DIODE Forward voltage	V_f		1.2	1.50	V	I_f = 10 mA
Junction capacitance	C_j		50		pF	V_R = 0 V, f = 1 MHz
Reverse leakage current	I_R			100	μA	V_R = 3.0 V
OUTPUT DETECTOR Peak blocking current, either direction	I_{DRM}	—	10	100	nA	V_{DRM} = 400 V, Note 1
Peak on-state voltage, either direction	V_{TM}	—	2.5	3.0	Volts	I_{TM} = 100 mA Peak

Note 1. Test voltage must be applied within dv/dt rating.

TRANSFER CHARACTERISTICS

DC CHARACTERISTICS		SYMBOL	MIN.	TYP.	MAX.	UNITS	TEST CONDITIONS
LED trigger current (current required to latch output)	MOC3020	I_{FT}	—	—	30	mA	Main terminal voltage = 3.0 V, R_L = 150Ω
	MOC3021	I_{FT}	—	—	15	mA	
	MOC3022	I_{FT}	—	—	10	mA	
	MOC3023	I_{FT}	—	—	5	mA	
Holding current		I_H	—	100	—	μA	Either direction

TRANSFER CHARACTERISTICS

CHARACTERISTICS	SYMBOL	MIN.	TYP.	MAX.	UNITS	TEST CONDITIONS
dv/dt RATING Critical rate of rise of off-state voltage	dv/dt	—	12	—	V/μs	Static dv/dt, T_A = 85°C (see Fig. 3)
Critical rate of rise of commutating voltage	dv/dt	—	0.2	—	V/μs	Commutating dv/dt I_{LOAD} = 15 mA (see Fig. 4)

ISOLATION CHARACTERISTICS

CHARACTERISTICS	SYMBOL	MIN.	TYP.	MAX.	UNITS	TEST CONDITIONS
Isolation voltage	V_{ISO}	5300			V_{AC}RMS	I_{I-O} < 1 μA, 1 Minute
	V_{ISO}	7500			V_{AC}PEAK	I_{I-O} < 1 μA, 1 Minute
Isolation resistance	R_{ISO}	10^{11}			ohms	V_{I-O} = 500 VDC
Isolation capacitance	C_{ISO}		0.5		pF	f = 1 MHz

Note 1: Ratings apply to either polarity of pin 6 — referenced to pin 4. Voltages must be applied within dv/dt rating.

TYPICAL ELECTRICAL CHARACTERISTIC CURVES

(25°C Free Air Temperature Unless Otherwise Specified)

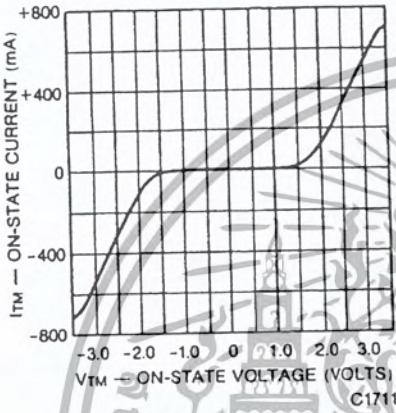


Fig. 1. On-State Characteristics

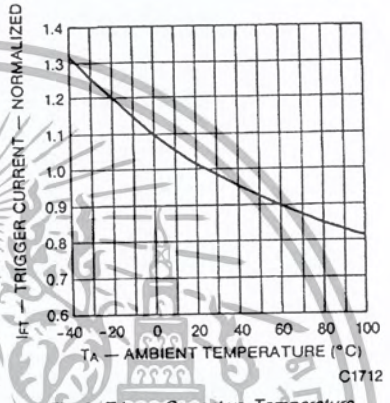
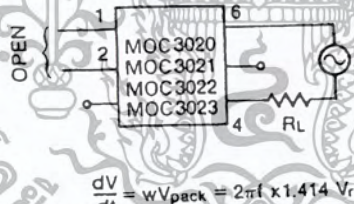


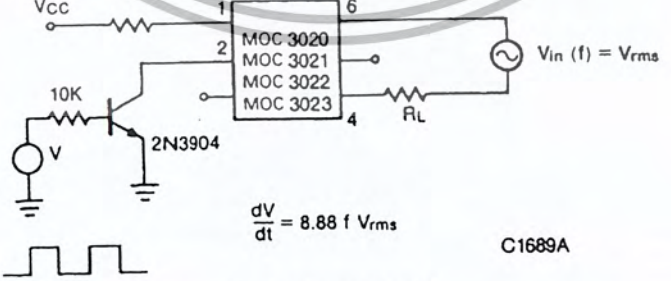
Fig. 2. Trigger Current vs. Temperature

TEST CIRCUITS FOR dV/dt MEASUREMENTS



$$\frac{dV}{dt} = \omega V_{pack} = 2\pi f \times 1.414\ V_{rms}$$
$$= 8.88\ f\ V_{rms}$$

Fig. 3. Static dV/dt



$$\frac{dV}{dt} = 8.88\ f\ V_{rms}$$

C1689A

Fig. 4. Commutating dV/dt

Order this document by ULN2803/D

MOTOROLA

ULN2803
ULN2804

Octal High Voltage, High Current Darlington Transistor Arrays

OCTAL PERIPHERAL DRIVER ARRAYS

SEMICONDUCTOR TECHNICAL DATA

The eight NPN Darlington connected transistors in this family of arrays are ideally suited for interfacing between low logic level digital circuitry (such as TTL, CMOS or PMOS/NMOS) and the higher current/voltage requirements of lamps, relays, printer hammers or other similar loads for a broad range of computer, industrial, and consumer applications. All devices feature open–collector outputs and free wheeling clamp diodes for transient suppression.

The ULN2803 is designed to be compatible with standard TTL families while the ULN2804 is optimized for 6 to 15 volt high level CMOS or PMOS.

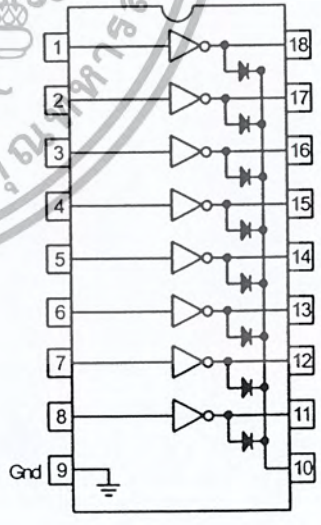
A SUFFIX
PLASTIC PACKAGE
CASE 707

MAXIMUM RATINGS (T_A = 25°C and rating apply to any one device in the package, unless otherwise noted.)

Rating	Symbol	Value	Unit
Output Voltage	V_O	50	V
Input Voltage (Except ULN2801)	V_i	30	V
Collector Current – Continuous	I_C	500	mA
Base Current – Continuous	I_B	25	mA
Operating Ambient Temperature Range	T_A	0 to +70	°C
Storage Temperature Range	T_{stg}	−55 to +150	°C
Junction Temperature	T_J	125	°C

$R_{\theta JA}$ = 55°C/W
Do not exceed maximum current limit per driver.

PIN CONNECTIONS



ORDERING INFORMATION

Device	Characteristics		
	Input Compatibility	$V_{CE(Max)}/I_{C(Max)}$	Operating Temperature Range
ULN2803A ULN2804A	TTL, 5.0 V CMOS 6 to 15 V CMOS, PMOS	50 V/500 mA	T_A = 0 to + 70°C

Rev 1

ELECTRICAL CHARACTERISTICS (T_A = 25°C, unless otherwise noted)

Characteristic		Symbol	Min	Typ	Max	Unit
Output Leakage Current (Figure 1)		I_{CEX}				µA
(V_O = 50 V, T_A = +70°C)	All Types		–	–	100	
(V_O = 50 V, T_A = +25°C)	All Types		–	–	50	
(V_O = 50 V, T_A = +70°C, V_I = 6.0 V)	ULN2802		–	–	500	
(V_O = 50 V, T_A = +70°C, V_I = 1.0 V)	ULN2804		–	–	500	
Collector–Emitter Saturation Voltage (Figure 2)		$V_{CE(sat)}$				V
(I_C = 350 mA, I_B = 500 µA)	All Types		–	1.1	1.6	
(I_C = 200 mA, I_B = 350 µA)	All Types		–	0.95	1.3	
(I_C = 100 mA, I_B = 250 µA)	All Types		–	0.85	1.1	
Input Current – On Condition (Figure 4)		$I_{I(on)}$				mA
(V_I = 17 V)	ULN2802		–	0.82	1.25	
(V_I = 3.85 V)	ULN2803		–	0.93	1.35	
(V_I = 5.0 V)	ULN2804		–	0.35	0.5	
(V_I = 12 V)	ULN2804		–	1.0	1.45	
Input Voltage – On Condition (Figure 5)		$V_{I(on)}$				V
(V_{CE} = 2.0 V, I_C = 300 mA)	ULN2802		–	–	13	
(V_{CE} = 2.0 V, I_C = 200 mA)	ULN2803		–	–	2.4	
(V_{CE} = 2.0 V, I_C = 250 mA)	ULN2803		–	–	2.7	
(V_{CE} = 2.0 V, I_C = 300 mA)	ULN2803		–	–	3.0	
(V_{CE} = 2.0 V, I_C = 125 mA)	ULN2804		–	–	5.0	
(V_{CE} = 2.0 V, I_C = 200 mA)	ULN2804		–	–	6.0	
(V_{CE} = 2.0 V, I_C = 275 mA)	ULN2804		–	–	7.0	
(V_{CE} = 2.0 V, I_C = 350 mA)	ULN2804		–	–	8.0	
Input Current – Off Condition (Figure 3) (I_C = 500 µA, T_A = +70°C)	All Types	$I_{I(off)}$	50	100	–	µA
DC Current Gain (Figure 2) (V_{CE} = 2.0 V, I_C = 350 mA)	ULN2801	h_{FE}	1000	–	–	–
Input Capacitance		C_I	–	15	25	pF
Turn–On Delay Time (50% E_I to 50% E_O)		t_{on}	–	0.25	1.0	µs
Turn–Off Delay Time (50% E_I to 50% E_O)		t_{off}	–	0.25	1.0	µs
Clamp Diode Leakage Current (Figure 6) (V_R = 50 V)	T_A = +25°C T_A = +70°C	I_R	–	–	50 100	µA
Clamp Diode Forward Voltage (Figure 7) (I_F = 350 mA)		V_F	–	1.5	2.0	V

TEST FIGURES

(See Figure Numbers in Electrical Characteristics Table)

Figure 1.

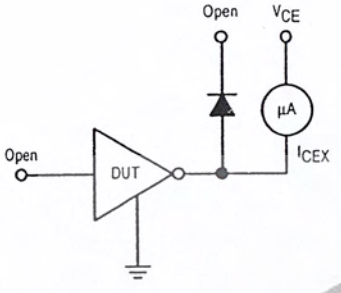


Figure 2.

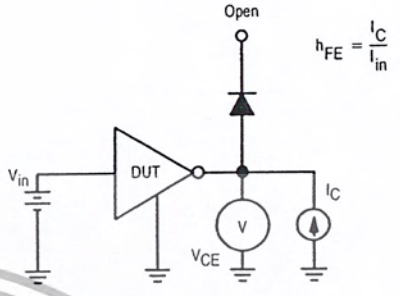


Figure 3.

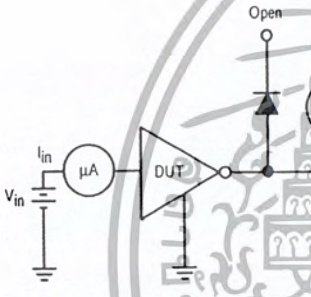


Figure 4.

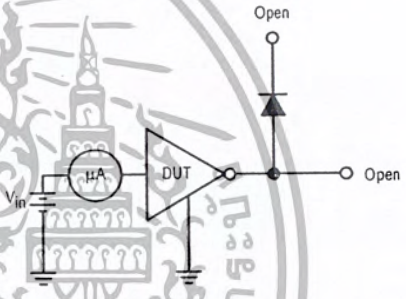


Figure 5.

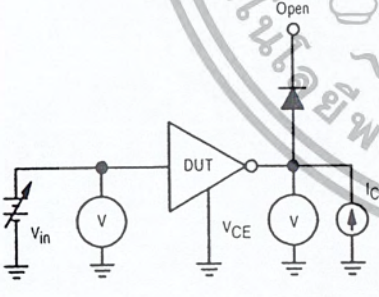


Figure 6.

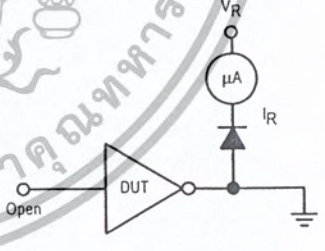
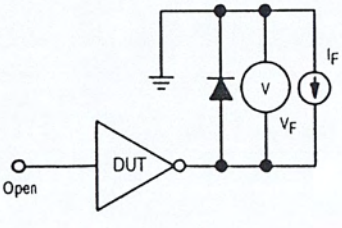


Figure 7.



TYPICAL CHARACTERISTIC CURVES – T_A = 25°C, unless otherwise noted

Output Characteristics

Figure 8. Output Current versus Saturation Voltage

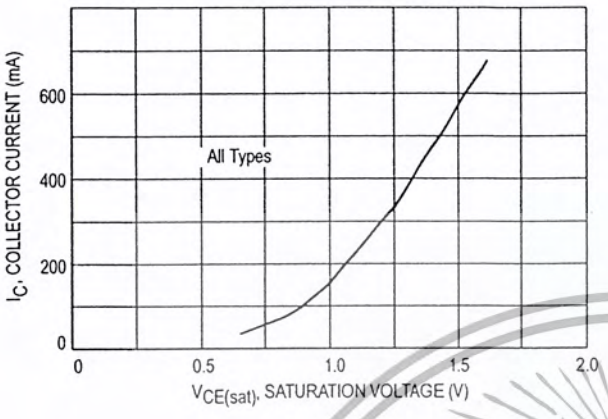
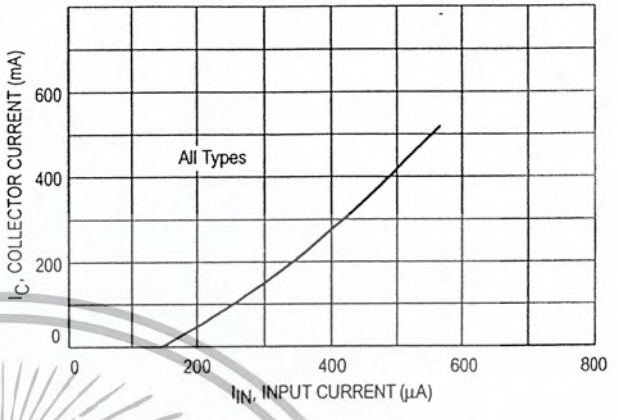


Figure 9. Output Current versus Input Current



Input Characteristics

Figure 10. ULN2803 Input Current versus Input Voltage

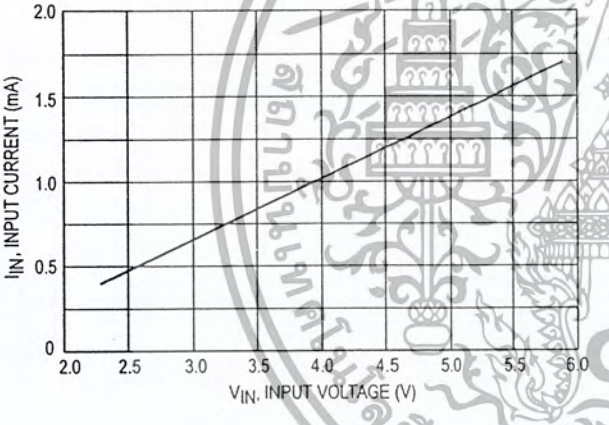


Figure 11. ULN2804 Input Current versus Input Voltage

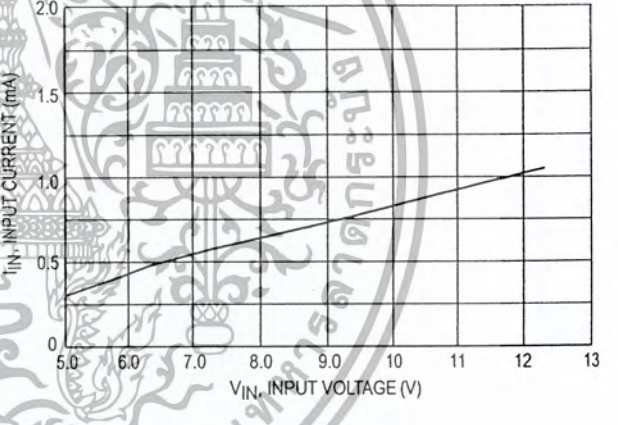
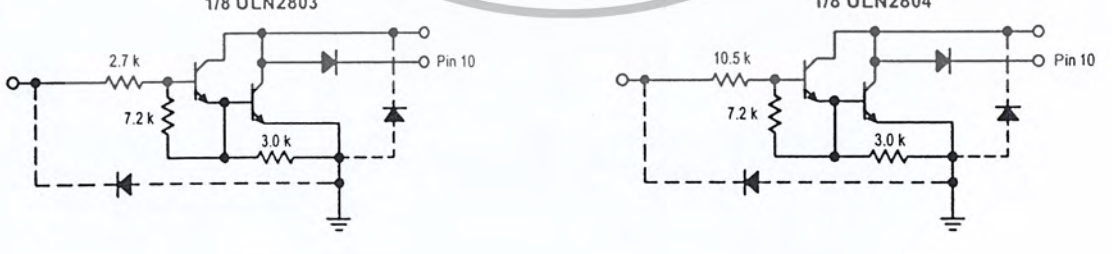
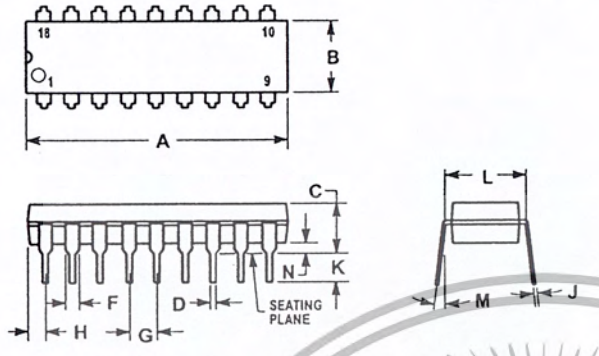


Figure 12. Representative Schematic Diagrams



OUTLINE DIMENSIONS

A SUFFIX
PLASTIC PACKAGE
CASE 707–02
ISSUE C



NOTES:
1. POSITIONAL TOLERANCE OF LEADS (D), SHALL BE WITHIN 0.25 (0.010) AT MAXIMUM MATERIAL CONDITION, IN RELATION TO SEATING PLANE AND EACH OTHER.
2. DIMENSION L TO CENTER OF LEADS WHEN FORMED PARALLEL.
3. DIMENSION B DOES NOT INCLUDE MOLD FLASH.

	MILLIMETERS		INCHES	
DIM	MIN	MAX	MIN	MAX
A	22.22	23.24	0.875	0.915
B	6.10	6.60	0.240	0.260
C	3.56	4.57	0.140	0.180
D	0.36	0.56	0.014	0.022
F	1.27	1.78	0.050	0.070
G	2.54 BSC		0.100 BSC	
H	1.02	1.52	0.040	0.060
J	0.20	0.30	0.008	0.012
K	2.92	3.43	0.115	0.135
L	7.62 BSC		0.300 BSC	
M	0°	15°	0°	15°
N	0.51	1.02	0.020	0.040

How to reach us:
USA/EUROPE/Locations Not Listed: Motorola Literature Distribution;
P.O. Box 20912; Phoenix, Arizona 85036. 1–800–441–2447 or 602–303–5454

MFAX: RMFAX0@email.sps.mot.com – TOUCHTONE 602–244–6609
INTERNET: http://Design–NET.com

JAPAN: Nippon Motorola Ltd.; Tatsumi–SPD–JLDC, 6F Seibu–Butsuryu–Center, 3–14–2 Tatsumi Koto–Ku, Tokyo 135, Japan. 03–81–3521–8315

ASIA/PACIFIC: Motorola Semiconductors H.K. Ltd.; 8B Tai Ping Industrial Park, 51 Ting Kok Road, Tai Po, N.T., Hong Kong. 852–26629298

GENERAL DESCRIPTION

Glass passivated triacs in a plastic envelope, intended for use in applications requiring high bidirectional transient and blocking voltage capability and high thermal cycling performance. Typical applications include motor control, industrial and domestic lighting, heating and static switching.

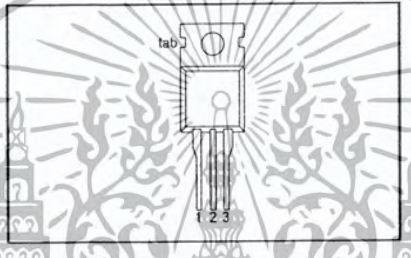
QUICK REFERENCE DATA

SYMBOL	PARAMETER	MAX.	MAX.	MAX.	UNIT
	BT136-	**500**	**600**	**800**	
	BT136-	**500F**	**600F**	**800F**	
	BT136-	**500G**	**600G**	**800G**	
V_{DRM}	Repetitive peak off-state voltages	500	600	800	V
$I_{T(RMS)}$	RMS on-state current	4	4	4	A
I_{TSM}	Non-repetitive peak on-state current	25	25	25	A

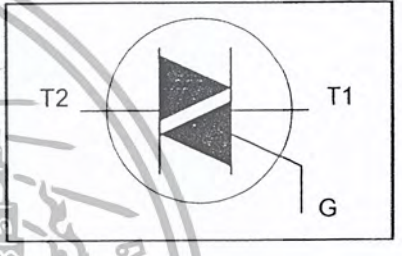
PINNING - TO220AB

PIN	DESCRIPTION
1	main terminal 1
2	main terminal 2
3	gate
tab	main terminal 2

PIN CONFIGURATION



SYMBOL



LIMITING VALUES

Limiting values in accordance with the Absolute Maximum System (IEC 134).

SYMBOL	PARAMETER	CONDITIONS		MIN.	MAX.			UNIT
					-500	-600	-800	
V_{DRM}	Repetitive peak off-state voltages			-	500[1]	600[1]	800	V
$I_{T(RMS)}$	RMS on-state current	full sine wave; $T_{mb} \leq 107$ °C		-		4		A
I_{TSM}	Non-repetitive peak on-state current	full sine wave; $T_j = 25$ °C prior to surge						
		t = 20 ms		-		25		A
		t = 16.7 ms		-		27		A
I^2t	I^2t for fusing	t = 10 ms		-		3.1		A^2s
dI_T/dt	Repetitive rate of rise of on-state current after triggering	I_{TM} = 6 A; I_G = 0.2 A; dI_G/dt = 0.2 A/µs						
			T2+ G+	-		50		A/µs
			T2+ G-	-		50		A/µs
			T2- G-	-		50		A/µs
			T2- G+	-		10		A/µs
I_{GM}	Peak gate current			-		2		A
V_{GM}	Peak gate voltage			-		5		V
P_{GM}	Peak gate power			-		5		W
$P_{G(AV)}$	Average gate power	over any 20 ms period		-		0.5		W
T_{stg}	Storage temperature			-40		150		°C
T_j	Operating junction temperature			-		125		°C

1 Although not recommended, off-state voltages up to 800V may be applied without damage, but the triac may switch to the on-state. The rate of rise of current should not exceed 3 A/µs.

THERMAL RESISTANCES

SYMBOL	PARAMETER	CONDITIONS	MIN.	TYP.	MAX.	UNIT
$R_{th\ j\text{-}mb}$	Thermal resistance junction to mounting base	full cycle	-	-	3.0	K/W
		half cycle	-	-	3.7	K/W
$R_{th\ j\text{-}a}$	Thermal resistance junction to ambient	in free air	-	60	-	K/W

STATIC CHARACTERISTICS

T_j = 25 °C unless otherwise stated

SYMBOL	PARAMETER	CONDITIONS	MIN.	TYP.	MAX.			UNIT
		BT136-			...	...F	...G	
I_{GT}	Gate trigger current	V_D = 12 V; I_T = 0.1 A						
		T2+ G+	-	5	35	25	50	mA
		T2+ G-	-	8	35	25	50	mA
		T2- G-	-	11	35	25	50	mA
		T2- G+	-	30	70	70	100	mA
I_L	Latching current	V_D = 12 V; I_{GT} = 0.1 A						
		T2+ G+	-	7	20	20	30	mA
		T2+ G-	-	16	30	30	45	mA
		T2- G-	-	5	20	20	30	mA
		T2- G+	-	7	30	30	45	mA
I_H	Holding current	V_D = 12 V; I_{GT} = 0.1 A	-	5	15	15	30	mA
V_T	On-state voltage	I_T = 5 A	-	1.4		1.70		V
V_{GT}	Gate trigger voltage	V_D = 12 V; I_T = 0.1 A	-	0.7		1.5		V
		V_D = 400 V; I_T = 0.1 A; T_j = 125 °C	0.25	0.4		-		V
I_D	Off-state leakage current	V_D = $V_{DRM(max)}$; T_j = 125 °C	-	0.1		0.5		mA

DYNAMIC CHARACTERISTICS

T_j = 25 °C unless otherwise stated

SYMBOL	PARAMETER	CONDITIONS	MIN.			TYP.	MAX.	UNIT
		BT136-	...	...F	...G			
dV_D/dt	Critical rate of rise of off-state voltage	V_{DM} = 67% $V_{DRM(max)}$; T_j = 125 °C; exponential waveform; gate open circuit	100	50	200	250	-	V/µs
dV_{com}/dt	Critical rate of change of commutating voltage	V_{DM} = 400 V; T_j = 95 °C; $I_{T(RMS)}$ = 4 A; dI_{com}/dt = 1.8 A/ms; gate open circuit	-	-	10	50	-	V/µs
t_{gt}	Gate controlled turn-on time	I_{TM} = 6 A; V_D = $V_{DRM(max)}$; I_G = 0.1 A; dI_G/dt = 5 A/µs	-	-	-	2	-	µs

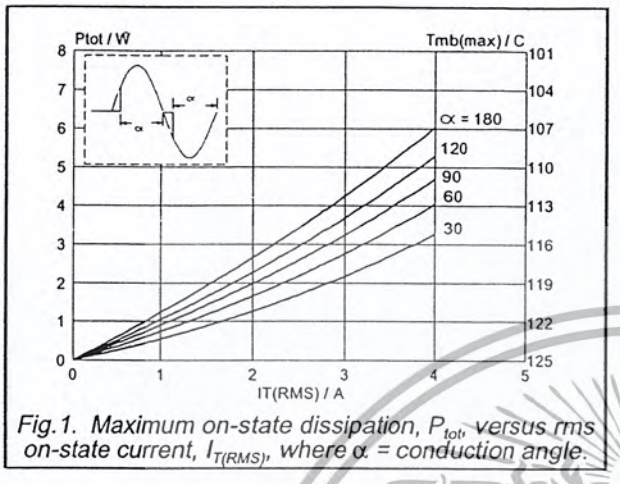


Fig.1. Maximum on-state dissipation, P_{tot}, versus rms on-state current, $I_{T(RMS)}$, where α = conduction angle.

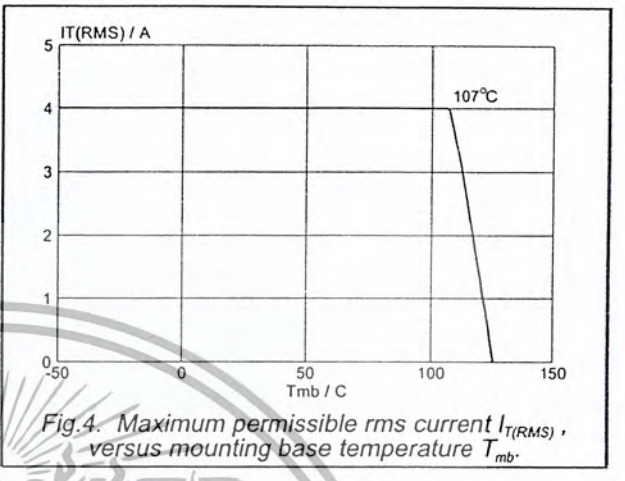


Fig.4. Maximum permissible rms current $I_{T(RMS)}$, versus mounting base temperature T_{mb}.

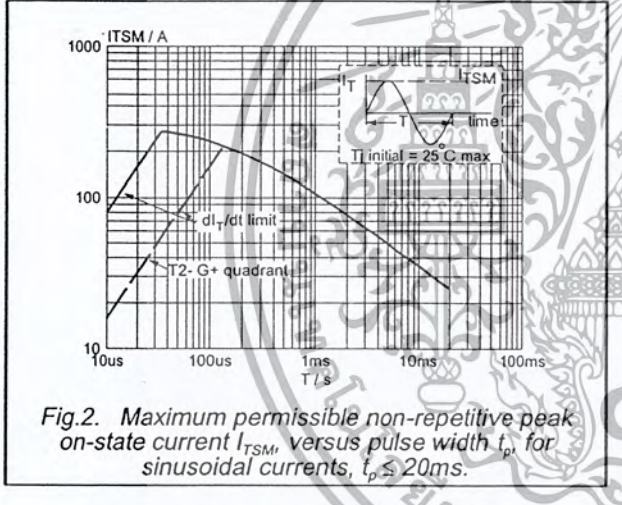


Fig.2. Maximum permissible non-repetitive peak on-state current I_{TSM}, versus pulse width t_p, for sinusoidal currents, $t_p \leq 20ms$.

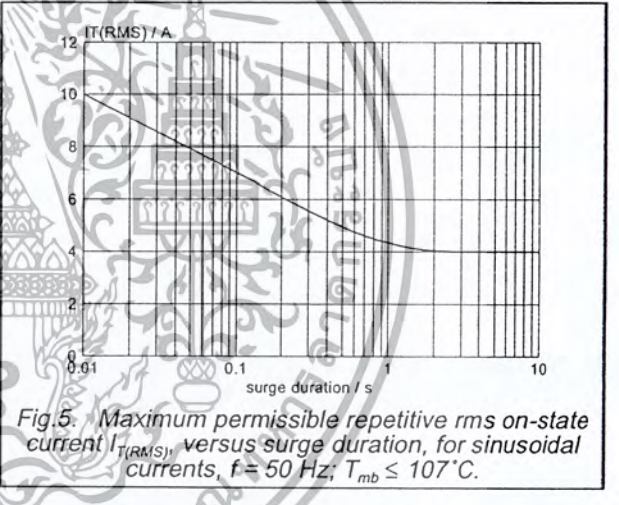


Fig.5. Maximum permissible repetitive rms on-state current $I_{T(RMS)}$, versus surge duration, for sinusoidal currents, f = 50 Hz; $T_{mb} \leq 107°C$.

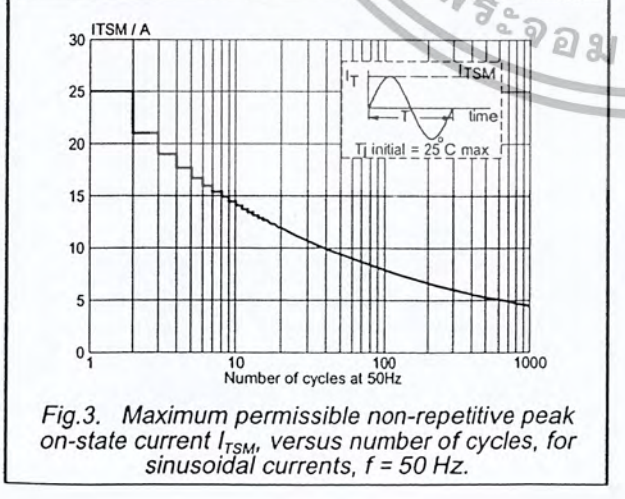


Fig.3. Maximum permissible non-repetitive peak on-state current I_{TSM}, versus number of cycles, for sinusoidal currents, f = 50 Hz.

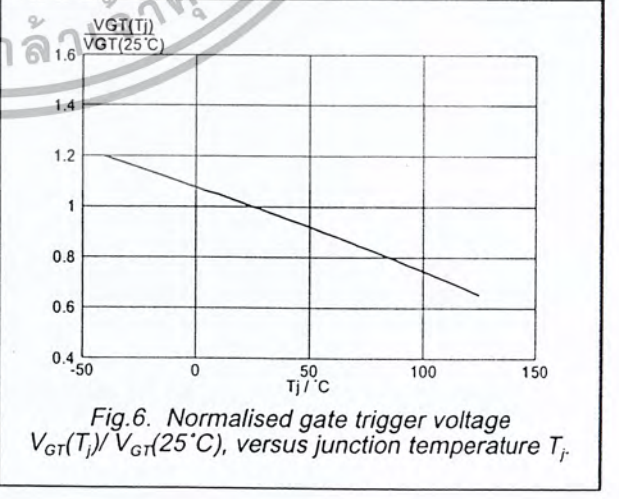


Fig.6. Normalised gate trigger voltage $V_{GT}(T_j)/ V_{GT}(25°C)$, versus junction temperature T_j.

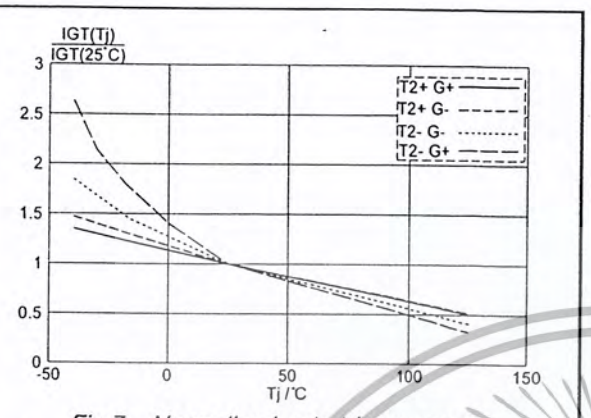


Fig.7. Normalised gate trigger current $I_{GT}(T_j)/ I_{GT}(25°C)$, versus junction temperature T_j.

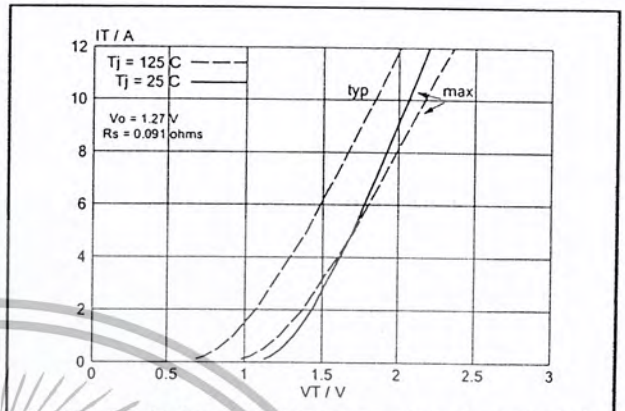


Fig.10. Typical and maximum on-state characteristic.

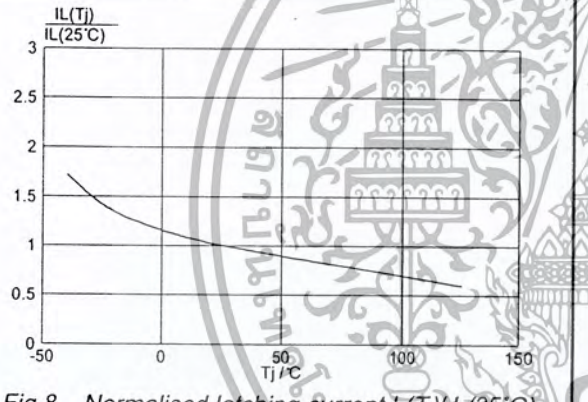


Fig.8. Normalised latching current $I_L(T_j)/ I_L(25°C)$, versus junction temperature T_j.

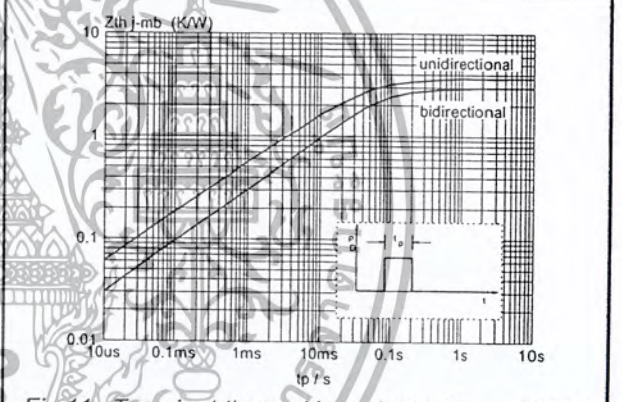


Fig.11. Transient thermal impedance $Z_{th\ j\text{-}mb}$, versus pulse width t_p.

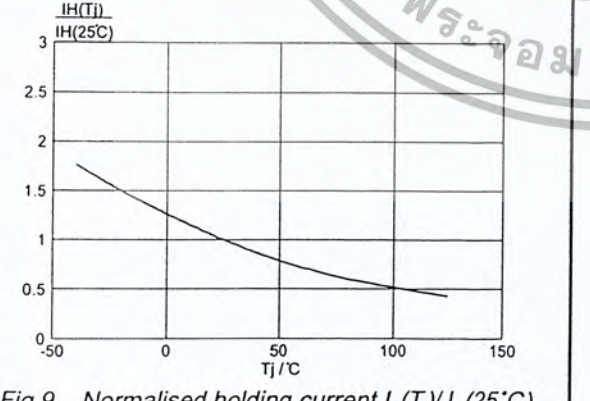


Fig.9. Normalised holding current $I_H(T_j)/ I_H(25°C)$, versus junction temperature T_j.

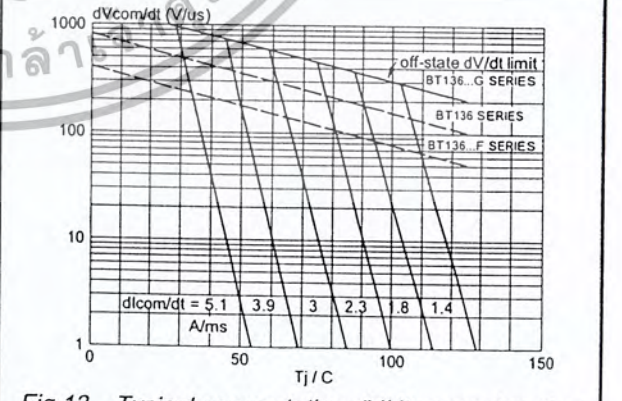


Fig.12. Typical commutation dV/dt versus junction temperature, parameter commutation dI_T/dt. The triac should commutate when the dV/dt is below the value on the appropriate curve for pre-commutation dI_T/dt.

MECHANICAL DATA

Dimensions in mm

Net Mass: 2 g

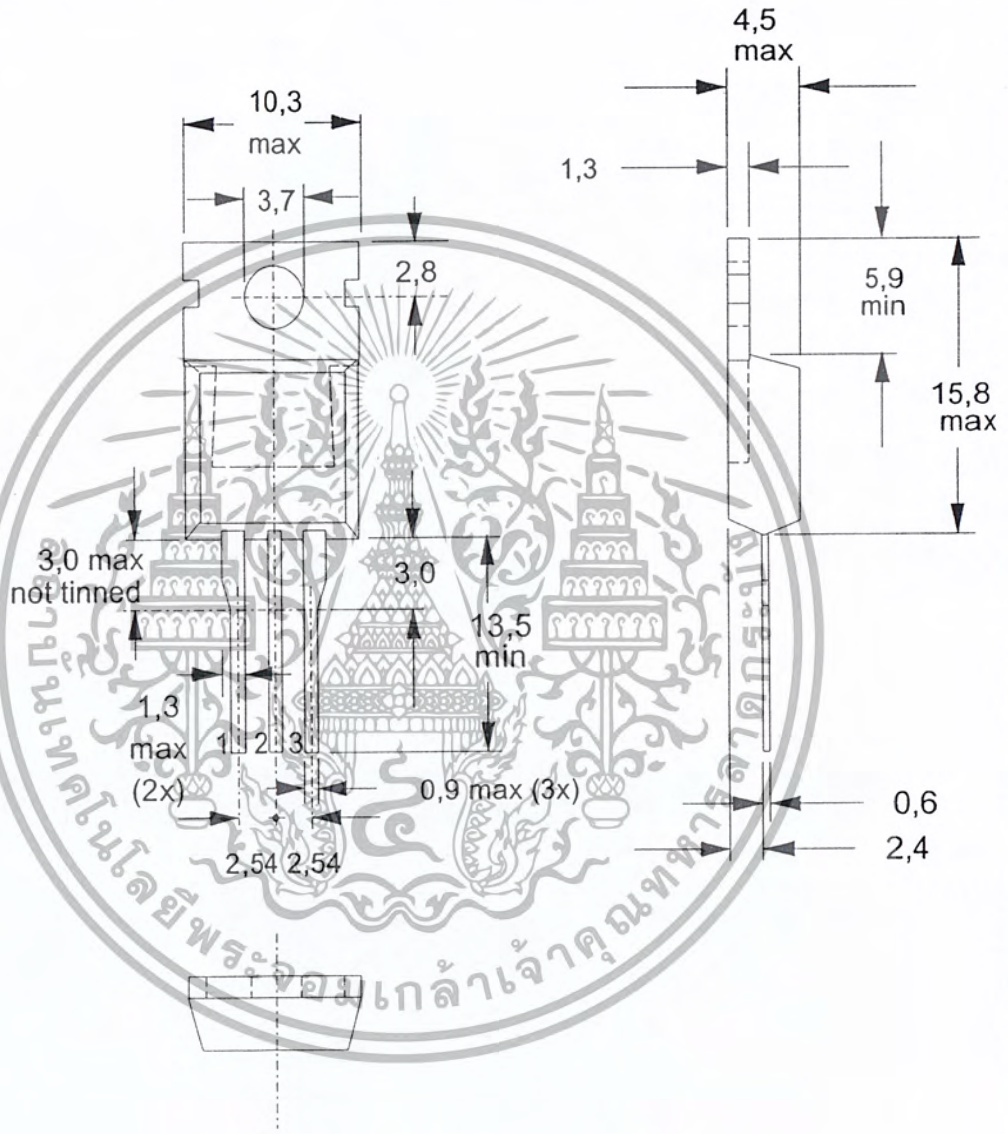


Fig.13. TO220AB; pin 2 connected to mounting base.

Notes
1. Refer to mounting instructions for TO220 envelopes.
2. Epoxy meets UL94 V0 at 1/8".

DEFINITIONS

Data sheet status	
Objective specification	This data sheet contains target or goal specifications for product development.
Preliminary specification	This data sheet contains preliminary data; supplementary data may be published later.
Product specification	This data sheet contains final product specifications.
Limiting values	
Limiting values are given in accordance with the Absolute Maximum Rating System (IEC 134). Stress above one or more of the limiting values may cause permanent damage to the device. These are stress ratings only and operation of the device at these or at any other conditions above those given in the Characteristics sections of this specification is not implied. Exposure to limiting values for extended periods may affect device reliability.	
Application information	
Where application information is given, it is advisory and does not form part of the specification.	

LIFE SUPPORT APPLICATIONS

These products are not designed for use in life support appliances, devices or systems where malfunction of these products can be reasonably expected to result in personal injury. Philips customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify Philips for any damages resulting from such improper use or sale.