แบบจำลองการควบคุมเครื่องใช้ไฟฟ้าด้วยเสียงพูด

Voice Controls Electric Appliances Model

โดย

นายชนม์พัฒน์ ตันติราพันธ์

นายชัยยุทธ์ ธนูแก้ว

รายงานนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

แบบจำลองการควบคุมเครื่องใช้ไฟฟ้าด้วยเสียงพูด

Voice Controls Electric Appliances Model

โดย

นายชนม์พัฒน์ ตันติราพันธ์ 41014087

นายชัยยุทธ์ ธนูแก้ว 41014096

อาจารย์ที่ปรึกษา

ผศ. วิภา แสงพิสิทธิ์

รายงานนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

ปริญญานิพนธ์ปีการศึกษา 2544

ภาควิชาโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง **แบบจำลองการควบคุมเครื่องใช้ไฟฟ้าด้วยเสียงพูด**

Voice Controls Electric Appliances Model

ผู้จัดทำ

1. นายชนม์พัฒน์ ตันติราพันธ์ 41014087

2. นายชัยยุทธ์ ธนูแก้ว 41014096

..... อาจารย์ที่ปรึกษา

(ผศ. วิภา แสงพิสิทธิ์)

แบบจำลองการควบคุมเครื่องใช้ไฟฟ้าด้วยเสียงพูด

VOICE CONTROLS ELECTRIC APPLIANCES MODEL

โดย นายชนม์พัฒน์ ตันติราพันธ์ 41014087
นายชัยยุทธ์ ธนูแก้ว 41014096

อาจารย์ที่ปรึกษา ผศ. วิภา แสงพิสิทธิ์

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้นำเสนอการควบคุมเครื่องใช้ไฟฟ้าด้วยเสียงพูด โดยการนำคอมพิวเตอร์ส่วนบุคคลมาใช้งานร่วมกับไมโครคอนโทรลเลอร์ตระกูล MCS-51 และมีการนำโปรแกรม MATLAB เข้ามาประยุกต์ใช้ในการประมวลผลเพื่อแยกแยะเสียงพูดด้วย

ในขั้นแรกคอมพิวเตอร์ส่วนบุคคลจะรับเสียงเข้าไปประมวลผลโดยใช้โปรแกรม MATLAB เพื่อทำการแยกแยะคำสั่ง จากนั้นนำผลที่ได้ส่งไปยังไมโครคอนโทรลเลอร์ MCS-51 ผ่านทางพอร์ต RS-232 โดยไมโครคอนโทรลเลอร์จะทำการจำลองการควบคุมเครื่องใช้ไฟฟ้าโดยแสดงผลด้วย LED ให้สามารถเปิดปิดได้ตามคำสั่ง

Abstract

This thesis presents about the model for controlling electric appliances by voice. The PC and MCS-51 are used for this purpose and MATLAB is employed to analyse the controlling voice.

First of all, a PC receives a voice to process by MATLAB, then the PC translates the processing result to the commanded signals. After, the results are transferred to MCS-51 via RS-232 port to generates the signals to control electric appliances which are modeled by LED.

สารบัญ

สารบัญภาพ

หน้า

หน้า

หน้า

สารบัญตาราง

บทที่ 1
บทนำ

การวิจัยทางด้านกรรมวิธีสัญญาณเสียงพูดในปัจจุบัน มีความก้าวหน้าไปมาก การรู้จำเสียงพูดจะปรากฏให้เห็นเป็นรูปเป็นร่างมากขึ้นในการใช้งาน ในปัจจุบันการรู้จำเสียงพูด (Speech Recognition) ถูกนำมาประยุกต์ใช้งานกันอย่างกว้างขวาง เช่น การให้บริการของธุรกิจธนาคาร การป้อนข้อมูลสันทนาการ โดยเฉพาะในระบบสื่อสารโทรคมนาคม จุดมุ่งหมายหลักของการรู้จำเสียงพูดก็คือ การเพิ่มพูนความสามารถของอุปกรณ์ต่างๆ ในการรับรู้และสื่อสารโต้ตอบกับมนุษย์ได้ เพื่อเพิ่มทางเลือกในการควบคุมสั่งการอุปกรณ์เครื่องมือต่างๆ โดยเฉพาะเครื่องคอมพิวเตอร์ ซึ่งการใช้เสียงพูดควบคุมสั่งการนี้ถือเป็นวิธีธรรมชาติมากที่สุดของมนุษย์ ศาสตร์ทางด้านการรู้จำเสียงพูดเกิดขึ้นมากกว่า 40 ปีแล้ว แต่เพิ่งจะเริ่มศึกษากันอย่างจริงจังในช่วงทศวรรษที่ผ่านมานี่เอง

การวิเคราะห์และรู้จำเสียงพูดของมนุษย์นั้นทำได้ยาก เนื่องจากการพูดของมนุษย์นั้นมีความซับซ้อนและแตกต่างกัน ความเร็วในการพูดและเสียงสูงต่ำของแต่ละบุคคล ยากที่จะใช้เครื่องมือในการวิเคราะห์ แต่ในปัจจุบันได้มีการพัฒนาวิธีการต่างๆ ทำให้การวิเคราะห์เสียงพูดมีความสะดวกมากขึ้น โดยความรู้ทางด้านสถิติเป็นวิธีหนึ่ง ที่นำมาประยุกต์ใช้ในการวิเคราะห์และจดจำเสียงพูด ร่วมกับเทคโนโลยีสมัยใหม่มีการนำเอาคอมพิวเตอร์มาช่วย เพื่อลดความยุ่งยากในการคำนวณ

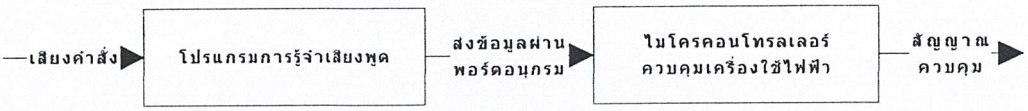
ปริญญานิพนธ์เสียงพูดควบคุมเครื่องใช้ไฟฟ้าฉบับนี้ ได้นำความรู้ทางด้านการรู้จำเสียงพูดมาประยุกต์ใช้งาน โดยได้ทำการได้ทำการแบ่งเนื้อหาที่ทำการศึกษาและทดลองออกเป็น 2 ส่วน ประกอบด้วย การรู้จำเสียงพูดและการจำลองการควบคุมเครื่องใช้ไฟฟ้า ในกระบวนการรู้จำเสียงพูดมีจุดประสงค์เพื่อให้คอมพิวเตอร์ประมวลผลและรับรู้ว่า ผู้ใช้ต้องการควบคุมเครื่องใช้ไฟฟ้าชนิดใด และให้เปิดหรือปิด โดยผลที่ประมวลได้ จะถูกส่งออกจากคอมพิวเตอร์ผ่านทางพอร์ตอนุกรม ซึ่งจะนำไปใช้ในการควบคุมเครื่องใช้ไฟฟ้า โดยการควบคุมเครื่องใช้ไฟฟ้าจะทำการจำลองการทำงานเปิดและปิดโดยการใช้ LED แสดงผลแทน

ในการทดลองได้ทำการบันทึกเสียงคำสั่งผ่านไมโครโฟน รูปแบบของเสียงคำสั่งที่ใช้เป็นเสียงคำสั่งความยาว 3 พยางค์ สัญญาณเสียงจะถูกบันทึกเก็บไว้เป็นข้อมูลที่โปรแกรม MATLAB สามารถเรียกใช้ได้ ไฟล์ที่ทำการบันทึกจะอยู่ในรูปของ ไฟล์.mat ซึ่งการบันทึกจะเป็นช่วงความถี่เสียงของมนุษย์ในช่วงความถี่ 300-3400 เฮิรตซ์ และสุ่มค่าของสัญญาณเสียงด้วยความถี่ 8000 เฮิรตซ์ ตามกฎของไนควิสต์ จากนั้นนำข้อมูลเสียงคำสั่งที่ได้ไปทำการตัดคำเพื่อหาขอบเขตของคำพูดในแต่ละพยางค์ นำเสียงในแต่ละพยางค์ไปหาขอบเขตความยาวของคำที่เหมาะสมกับการวิเคราะห์ และนำคำที่ได้ไปปรับแต่งค่าแซมปลิ้งของแต่ละคำให้เท่ากัน โดยวิธีการทำนอร์มอลไลซ์ เพื่อให้คำคำนั้นมีสัมประสิทธิ์ในการวิเคราะห์น้อยลง จะทำการแทนค่าของสัญญาณเสียงในแต่ละช่วงด้วยสัมประสิทธิ์ LPC สัมประสิทธิ์ที่ได้จะถูกปรับปรุงอีกครั้งหนึ่งเพื่อให้มีเสถียรภาพดียิ่งขึ้น และจะใช้หลักการของ Hidden Markov Model (HMM) มาประยุกต์ใช้ในการรู้จำเสียง เพื่อรับรู้ว่าเสียงคำสั่งที่ผู้ใช้ป้อนให้แก่เครื่องคอมพิวเตอร์นั้นคือคำสั่งให้อุปกรณ์ไฟฟ้าชนิดใด ทำงานเปิดหรือปิด

ในการส่งข้อมูลเสียงที่ประมวลผลได้ผ่านทางพอร์ตอนุกรมเพื่อใช้ในการจำลองการควบคุมเครื่องใช้ไฟฟ้านั้น เสียงคำสั่งที่ประมวลผลได้จะถูกกำหนดให้เท่ากับรหัสข้อมูลตัวหนึ่ง วงจรไมโครคอนโทรลเลอร์จะทำหน้าที่คอยตรวจสอบสัญญาณข้อมูลจากพอร์ตอนุกรมว่าตรงกับค่าที่กำหนดไว้หรือไม่ ถ้าข้อมูลที่ส่งออกมาจากพอร์ตอนุกรมตรงกับค่าที่ตั้งเอาไว้ ไมโครคอนโทรลเลอร์ก็จะทำการจำลองการทำงานของเครื่องใช้ไฟฟ้า โดยทำการเปิดหรือปิด LED ตัวที่ถูกกำหนดให้เป็นตัวแทนของอุปกรณ์ไฟฟ้าตัวซึ่งถูกสั่งงานให้เปิดหรือปิดดังกล่าว

บทที่ 2
ทฤษฎีหรือหลักการ

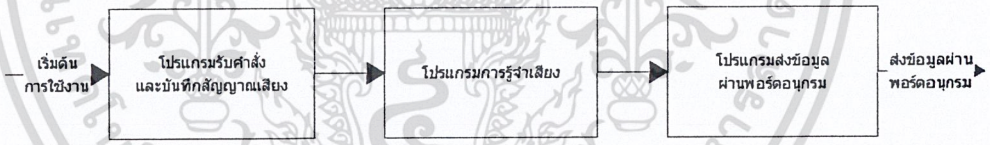
แบบจำลองการควบคุมเครื่องใช้ไฟฟ้าด้วยเสียงพูด มีการทำงานแบ่งเป็น 2 ส่วน ดังบล็อกไดอะแกรมในรูปที่ 2.1



รูปที่ 2.1 บล็อกไดอะแกรมแสดงการทำงานของแบบจำลองการควบคุมเครื่องใช้ไฟฟ้าด้วยเสียงพูด

จากรูปที่ 2.1 เมื่อผู้ใช้งานต้องการสั่งงานเครื่องใช้ไฟฟ้าต้องทำการป้อนคำสั่งด้วยเสียงพูดผ่านทางไมโครโฟน คอมพิวเตอร์จะนำเสียงคำสั่งไปประมวลผลซึ่งจะทำให้รู้ว่า ผู้ใช้ต้องการควบคุมเครื่องใช้ไฟฟ้าชนิดใด ทำงานในโหมดเปิดหรือปิด ผลที่คอมพิวเตอร์ประมวลได้จะถูกส่งออกผ่านพอร์ตอนุกรมไปอุปกรณ์ไมโครคอนโทรลเลอร์ เพื่อจำลองการทำงานการควบคุมเครื่องใช้ไฟฟ้าต่อไป

ในส่วนของโปรแกรมที่ใช้ในการควบคุมเครื่องใช้ไฟฟ้าด้วยเสียงพูด มีบล็อกไดอะแกรมตามรูปที่ 2.2



รูปที่ 2.2 บล็อกไดอะแกรมของโปรแกรมที่ใช้ในการควบคุมเครื่องใช้ไฟฟ้าด้วยเสียงพูด

โปรแกรมที่ใช้ในการทดลองประกอบด้วย โปรแกรมรับเสียงคำสั่งและบันทึกสัญญาณเสียง โปรแกรมการรู้จำเสียง และโปรแกรมส่งข้อมูลที่ประมวลผลได้ส่งออกผ่านพอร์ตอนุกรม แต่ละส่วนมีหลักการทำงานดังนี้

- โปรแกรมรับเสียงคำสั่งและบันทึกสัญญาณเสียง สัญญาณเสียงที่ส่งผ่านมาจากไมโครโฟนจะถูกบันทึกไว้เพื่อใช้ในการประมวลผลในขั้นตอนการรู้จำเสียงต่อไป สำหรับมาตรฐานที่ใช้ในการบันทึกใช้แบบ 8 บิต โมโนสเตอริโอ 8000 แซมเปิ้ล

โปรแกรมรับเสียงคำสั่งและบันทึกสัญญาณเสียงเป็นคอมโพเนนท์ (component) ที่ได้ทำการสร้างขึ้นใน MATLAB ใช้ในการรับสัญญาณเสียง บันทึกสัญญาณเสียง ก่อนที่จะนำสัญญาณดังกล่าวส่งไปยังขั้นตอนการรู้จำเสียงต่อไป

- โปรแกรมการรู้จำเสียง เสียงที่ถูกบันทึกไว้จะถูกนำมาหาขอบเขตของเสียงที่แท้จริง หลังจากนั้นนำมาหาค่าพารามิเตอร์ต่างๆ ที่เหมาะสมที่ใช้เป็นตัวแทนของเสียงนั้น ก่อนที่จะนำมาผลที่ได้มาเปรียบเทียบกับโมเดลเสียงต้นแบบตามวิธีการของ Hidden Markov Model (HMM) เพื่อหาค่าความน่าจะเป็นสูงสุดว่าเสียงอินพุตที่รับเข้ามานั้นตรงกับโมเดลเสียงใด

- โปรแกรมส่งข้อมูลออกพอร์ตอนุกรม เมื่อทำการประมวลผลจนรู้ว่าเสียงคำสั่งเป็นเสียงใดแล้ว จะทำการแปลงผลที่ได้ให้อยู่ในรูปรหัสข้อมูล และทำการส่งไปยังอุปกรณ์ไมโครคอนโทรลเลอร์ เพื่อใช้ในการจำลองการควบคุมอุปกรณ์ไฟฟ้าต่อไป

2.1 ลักษณะของเสียงพูด

มนุษย์เปล่งเสียงพูดด้วยอวัยวะที่ใช้ในการออกเสียง ทำเสียงตามที่มีในระบบภาษาของตน แม้ว่าคนที่อยู่ในสังคมเดียวกันจะใช้ภาษาเดียวกันแต่ถ้าพิจารณาเสียงที่เปล่งออกมาจริงๆ แล้วแต่ละครั้งก็อาจจะสังเกตลักษณะที่แตกต่างกันได้ เราจึงสามารถจำเสียง จำวิธีพูดของคนที่เราคุ้นเคยได้ เสียงพูดที่จะอธิบายด้วยหลักเกณฑ์ทางวิทยาศาสตร์แม้ว่าในภาษาหนึ่งๆ จะมีเสียงต่างกันมากบ้างน้อยบ้าง แต่ละเสียงก็สามารถนำมาพิจารณาและอธิบายให้รู้ลักษณะการออกเสียงและตำแหน่งที่เกิดเสียงได้คำอธิบายนี้จะทำให้เข้าใจลักษณะเสียงทุกเสียง วิชาที่ว่าด้วยเสียงพูดเรียกว่า วิชาสัทศาสตร์ (Phonetics)

ในการศึกษาเสียงพูดแบ่งได้เป็น 2 ลักษณะ คือ

1. สรีรสัทศาสตร์ (Articulatory) เป็นการศึกษาเสียงพูดจากอวัยวะและการเคลื่อนไหวอวัยวะที่ทำให้เกิดเสียงพูด การอธิบายนี้ก็จะอธิบายโดยอาศัยลักษณะและอาการเคลื่อนไหวของอวัยวะที่เกี่ยวข้องในการเปล่งเสียงพูดนั้น

2. กลสัทศาสตร์ (Acoustic Phonetics) เป็นการศึกษาเสียงพูดจากลักษณะคลื่นเสียงที่ผู้พูดเปล่งออกมาแล้ว และผู้ฟังได้ยินว่ามีลักษณะทางกลศาสตร์อย่างไร การศึกษาตามแนวนี้ต้องอาศัยความรู้ทางฟิสิกส์ และคณิตศาสตร์ช่วยอธิบายลักษณะของคลื่นเสียง

2.1.1 การเกิดของเสียง (Speech Production)

การเกิดของเสียงแบ่งออกเป็น 3 ขั้นตอน คือ

ขั้นตอนที่ 1 จุดเริ่มต้น เป็นขั้นตอนที่ลมเริ่มถูกขับออกจากปอด ผ่านเข้าไปสู่ขั้นตอนที่ 2

ขั้นตอนที่ 2 การดัดแปลงลมที่เส้นเสียงอวัยวะ ที่ใช้ในขั้นตอนนี้คือส่วนที่ต่อจากปอดขึ้นมาถึงกล่องเสียง และที่กล่องเสียง เส้นเสียงจะทำหน้าที่ เป็นลิ้นปิดเปิดทำให้เกิดเสียง 2 ชนิด คือ

1. เสียงก้อง (Voiced) เกิดจากเส้นเสียงปิดกั้นลมไว้ ลมที่ผ่านออกมาจะเพิ่มแรงดันมากขึ้นจนเส้นเสียงเปิดสลับกันไป ทำให้เกิดเสียงก้องขึ้นมา ซึ่งสามารถเรียกความถี่ในการปิดเปิดของเส้นเสียงว่าความถี่มูลฐาน

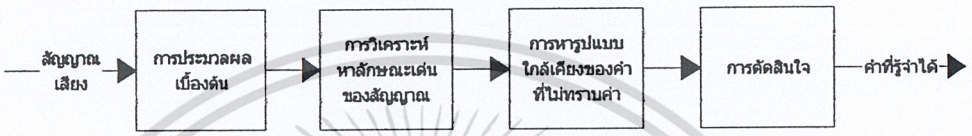
2. เสียงไม่ก้อง (Unvoiced) เสียงชนิดนี้เส้นเสียงจะเปิดตลอดเวลาที่ลมผ่าน ลมจึงผ่านออกมาได้สะดวกทำให้เกิดเสียงไม่ก้องขึ้น

ขั้นตอนที่ 3 การเปลี่ยนแปลงลักษณะเส้นเสียง อวัยวะที่ใช้คือส่วนที่ต่อจากกล่องเสียงจนถึงริมฝีปาก โดยลมที่ผ่านออกจากกล่องเสียงจะทำให้เกิดเสียงในลักษณะต่าง ซึ่งจะเกิดจากการเปลี่ยนแปลงของอวัยวะที่กล่าวไว้ในหัวข้อที่แล้ว

ทฤษฎีการรู้จำเสียง

2.2 โครงสร้างพื้นฐานของระบบการรู้จำเสียง

จากการศึกษารูปแบบการจำเสียงพูดแบบคำเดี่ยว (Isolated word recognition) โครงสร้างพื้นฐานของระบบการรู้จำเสียงพูด แสดงดังรูปที่ 2.3 โดยแบ่งออกได้เป็น 4 ส่วน ดังนี้

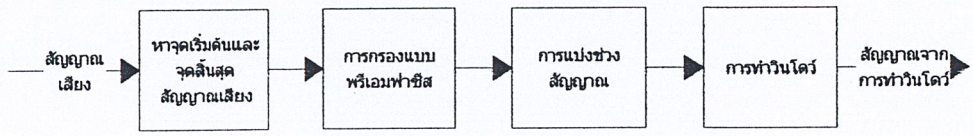


รูปที่ 2.3 บล็อกไดอะแกรมแสดงโครงสร้างพื้นฐานในระบบการรู้จำเสียง

1. การประมวลผลเบื้องต้น (Preprocessing) เพื่อจัดเตรียมข้อมูลให้เหมาะสมในการคำนวณขั้นต่อๆ ไป
2. การวิเคราะห์หาลักษณะเด่นของสัญญาณ (Feature measurement) เป็นขั้นตอนการสกัดเพื่อนำลักษณะเด่นของข้อมูลออกมา เพื่อลดจำนวนของข้อมูลเป็นการลดเวลาการประมวลผลในขั้นต่อๆ ไป
3. การหารูปแบบใกล้เคียงของคำที่ยังไม่ทราบค่า (Pattern similarity determination) โดยเทียบกับคำอ้างอิงที่มีอยู่ (Reference pattern)
4. การตัดสินใจ (Decision rule) เป็นกฎในการเลือกว่าคำที่ต้องการรู้จำ ว่าใกล้เคียงกับรูปแบบอ้างอิงของคำใดมากที่สุด

2.3 การประมวลผลสัญญาณเสียงเบื้องต้น (Speech Pre-Processing)

การประมวลผลสัญญาณเสียงเบื้องต้น เป็นกรรมวิธีในการจัดเตรียมข้อมูล จากข้อมูลดิบของเสียงพูดที่ได้เพื่อให้ได้ข้อมูลที่มีความเหมาะสมที่จะใช้ในขั้นตอนต่อไป ซึ่งขั้นตอนในการเตรียมสัญญาณแสดงดังรูปที่ 2.4



รูปที่ 2.4 แสดงขั้นตอนการประมวลผลสัญญาณเสียงเบื้องต้น

การประมวลผลสัญญาณเสียงเบื้องต้น มีขั้นตอนดังต่อไปนี้

2.3.1 การหาขอบเขตและการนอร์มอลไลซ์สัญญาณเสียง

เนื่องจากเสียงพูดที่นำมาประมวลผล แต่ละเสียงที่จุดเริ่มต้นและจุดสิ้นสุดของสัญญาณแตกต่างกัน ดังนั้นจึงต้องมีการหาขอบเขตที่แท้จริงของสัญญาณเสียง หลังจากการหาขอบเขตของสัญญาณเรียบร้อยแล้ว จะนำสัญญาณเสียงที่ได้มาทำการนอร์มอลไลซ์ เพื่อให้แต่ละสัญญาณเสียงมีความยาวเท่ากัน เป็นการสะดวกในการวิเคราะห์ในขั้นตอนต่อไป

2.3.2 การพรีเอมฟาซีส (Preemphasis)

เนื่องจากสัญญาณเสียงพูดของมนุษย์ จะมีองค์ประกอบส่วนใหญ่อยู่บริเวณความถี่ต่ำ เมื่อเทียบกับความถี่ปฏิบัติงานไม่เกิน 4 kHz ดังนี้ เพื่อให้อัตราส่วนของสัญญาณเสียงต่อสัญญาณรบกวน (SNR) มีค่าคงที่ตลอดช่วงความถี่ปฏิบัติงานจึงทำการพรีเอมฟาซีส โดยเน้นความถี่สูงมีขนาดสูงขึ้น

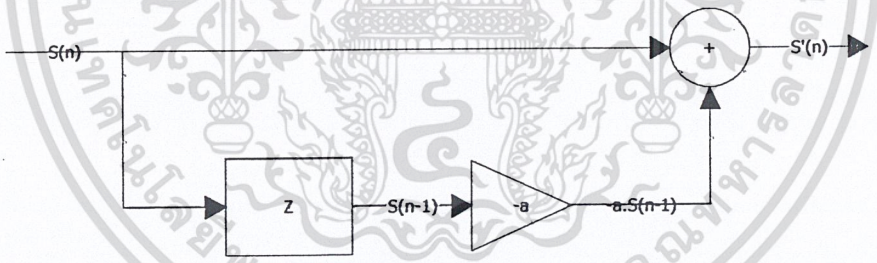
การพรีเอมฟาซีส คือ การกรอง สัญญาณด้วยวงจรกรองความถี่สูงผ่าน (High Pass Filter) ซึ่งนิยมใช้วงจรอันดับหนึ่ง ซึ่งตัวกรองเชิงเลขแบบง่ายที่สุด มีรูปแบบสมการดังนี้

$$y(n) = x(n) - ax(n-1) \tag{2.1}$$

มีฟังก์ชันถ่ายโอนเป็น

$$H(z) = 1 - az^{-1} \qquad \text{เมื่อ } 0.9<a<1.0 \tag{2.2}$$

สมมติว่าสัญญาณเดิมเป็น S(n) เมื่อผ่านวงจรกรองแล้วจะเป็น S' (n)



รูปที่ 2.5 วงจรกรองความถี่สูงผ่าน

จะได้ว่า

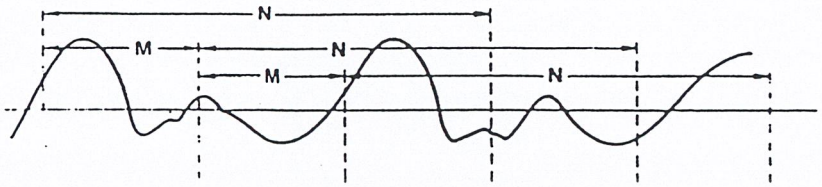
$$S'(n) = S(n) - a.S(n-1) \tag{2.3}$$

ยิ่ง a เข้าใกล้ I เท่าใด ความถี่สูงจะถูกขยายมากขึ้นเท่านั้น ค่า a ที่นิยมสำหรับใช้ในการหาพารามิเตอร์ของ LPC คือค่า 15/16 = 0.9375

2.3.3 การแบ่งช่วงสัญญาณ (Frame Blocking)

สัญญาณที่ผ่านการพรีเอมฟาซีสแล้ว ถูกตัดมาวิเคราะห์ทีละเฟรม เฟรมละ N ตัวอย่าง สัญญาณในการวิเคราะห์ที่แต่ละช่วงของ N ตัวอย่าง สัญญาณนั้น จะวิเคราะห์ โดยเลื่อนเป็นระยะ M ช่วงสัญญาณ

จะหมดสัญญาณเสียงที่นำมาวิเคราะห์ ดังรูปที่ 2.4 ซึ่งระยะ N ที่เหมาะสมจะต้องมีค่ามากกว่า M จะทำให้วิเคราะห์สัญญาณได้แม่นยำ แต่ถ้าค่า N น้อยเกินไปการวิเคราะห์จะล่าช้า



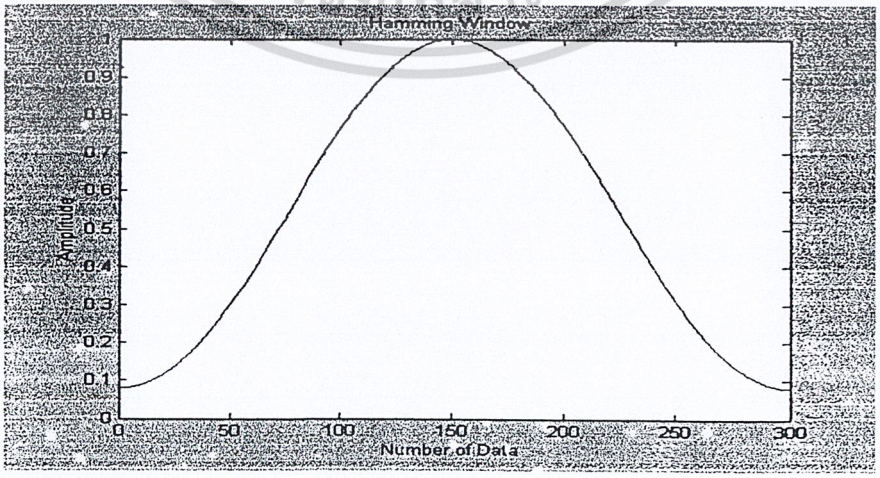
รูปที่ 2.6 การแบ่งช่วงของสัญญาณ

เนื่องจากสัญญาณเสียงมีการเปลี่ยนแปลงคุณสมบัติตามเวลา แต่ถ้าแบ่งเสียงออกเป็นเฟรมสั้นๆ เสียงจะมีการเปลี่ยนแปลงคุณสมบัติน้อยมาก ทำให้การวิเคราะห์ทำได้ง่ายขึ้น ในการทดลองนี้จึงทำการแบ่งสัญญาณเสียงออกเป็นเฟรมๆ ละ M แซมเปิ้ล (Sample) และทำการเลื่อนเฟรมไปเรื่อยๆ จนหมด

2.3.4 การวินโดว์ (Windowing)

เนื่องจากการที่ตัดสัญญาณเสียงมาวิเคราะห์ทีละเฟรมจะทำให้ที่ขอบของเฟรมเกิดความไม่ต่อเนื่องของสัญญาณ ถ้ามองในโดเมนความถี่สูง ดังนั้นเพื่อที่จะลดองค์ประกอบทางความสูงเหล่านี้ จะต้องคูณด้วยฟังก์ชันวินโดว์เพื่อลดความไม่ต่อเนื่องของสัญญาณที่ขอบ และไม่ทำให้ สเปคตรัมของสัญญาณที่ช่วงความความถี่ต่ำเปลี่ยนไปมากนัก ในที่นี้จะใช้ฟังก์ชันวินโดว์แฮมมิ่ง (Hamming Window Function) ซึ่งนิยามโดยสมการ

$$w(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right) \quad \text{เมื่อ} \quad n = 0, 1, 2, \ldots, N-1 \qquad (2.4)$$



รูปที่ 2.7 แสดงฟังก์ชั่นวินโดว์แบบแฮมมิ่ง

ในการวิเคราะห์เสียงโดยใช้ฟังก์ชันวินโดว์ จะพบว่าสัญญาณเสียงที่ผ่านการกรองโดยวินโดว์นั้น จะมีการแกว่างขึ้นลงมากน้อยขึ้นกับช่วงเวลาของวินโดว์ (ความกว้างของวินโดว์) คือถ้าช่วงของวินโดว์สั้นจะมีการแกว่งขึ้นลงอย่างรวดเร็ว และถ้าช่วงเวลาของการวินโดว์มากจะมีการแกว่างขึ้นลงช้าๆ ดังนั้น การเลือกช่วงเวลาของการวินโดว์ ต้องให้อยู่ในช่วงที่เหมาะสม คือไม่ให้เอาต์พุตของสัญญาณแกว่งช้าหรือเร็วเดินไป อยู่ในช่วงระหว่าง 10-30 ms

เมื่อคูณกับฟังก์ชันแล้ว ก็จะได้

$$x'(n) = w(n).x(n) \tag{2.5}$$

2.4 การวิเคราะห์หาคุณลักษณะเด่นของสัญญาณเสียง (Feature extraction)

การวิเคราะห์หาคุณลักษณะของเด่นของสัญญาณเสียง เป็นเทคนิคในการแปลงข้อมูลที่มีอยู่มากมาย ให้เป็นส่วนเล็กๆ ซึ่งส่วนเล็กๆ นี้จะแสดงคุณสมบัติของคลื่นเสียงนั้นๆ โดยจะใช้วิการป้อนสัญญาณผ่านวงจรกรอง (Filter) และการประเมินเชิงเส้น (Linear Predictive Coding)

หลักการพื้นฐานของการประเมินเชิงเส้น คือ การประเมินเสียงจากผลรวมเชิงเส้นของสัญญาณเสียงในอดีต โดยอาศัยเกณฑ์กำลังสองของสัญญาณความคลาดเคลื่อน มีค่าต่ำสุดในการสัมประสิทธิ์ การประเมินเชิงเส้น กล่าวคือหลังจากการทำพรีเอมฟาซีสและวินโดว์ครบทุกตัวในหนึ่งเฟรมแล้ว จะทำการคำนวณออโตคอร์รีเลชั่น คือจะทำการหาค่า สัมประสิทธิ์ LPC และอัตราขยาย G มีขั้นตอนแสดงดังรูปที่ 2.8

รูปที่ 2.8 แสดงขั้นตอนในการวิเคราะห์หาคุณลักษณะเด่นของสัญญาณเสียง

2.4.1 การคำนวณออโตคอร์รีเลชั่น (Autocorrelation)

สมมติว่าสัญญาณเดิมเป็น S(n) การประมาณ ค่าสัญญาณเป็น S'(n) ดังนั้นสามารถอธิบายการประมาณเชิงเส้นด้วยสมการดังนี้

$$S'(n) = \sum_{k=1}^{p} \alpha_k S(n-k) \tag{2.6}$$

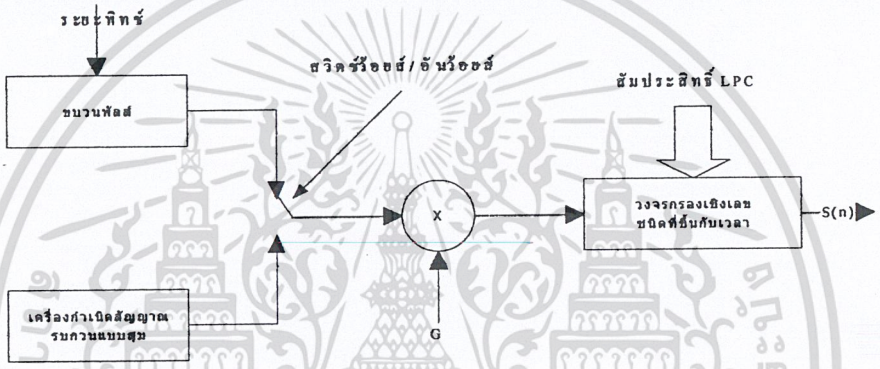
เมื่อ α_k เป็นค่าคงที่ เรียกวิธีการประมาณเชิงเส้นอันดับ P โดยมีเงื่อนไขว่า ค่า α_k ที่ใช้ในการประมาณที่จะต้องทำให้ ผลรวมของกำลังสองของความคลาดเคลื่อน $\{S(n)-S'(n)\}^2$ มีค่าน้อยที่สุด ซึ่งใช้ในการประมาณเชิงเส้น (Autocorrelation Method) หรือวิธีอัตตสัมพันธ์

การคำนวณออโตคอร์รีเลชั่น เป็นวิธีการหาสัมประสิทธิ์ LPC โดยฟังก์ชันออโตคอร์รีเลชั่น ซึ่งเป็นการเปรียบเทียบสัญญาณกับสัญญาณของตัวมันเองที่ถูกเปลี่ยนไปตามแกนเวลา ที่ใช้วิธีนี้ เนื่องจากเป็นการคำนวณที่มีการแก้สมการทีละน้อยกว่าวิธีอื่นๆ และมีความแน่นอนในด้านเสถียรภาพ อีกทั้งที่การเก็บข้อมูลที่น้อยกว่า

คุณสมบัติของฟังก์ชันออโตคอร์รีเลชั่น

1. เป็นฟังชันก์คู่ R(k) = R(-k)
2. จะเป็นค่าสูงสุดเมื่อเปรียบเทียบสัญญาณของตัวมันเองที่ตำแหน่งแกนเวลาเดียวกันคือ R(0) = max

จากหลักการพื้นฐานของการประมาณเชิงเส้นและแบบจำลองระบบสร้างสัญญาณเสียงสามารถเขียนบล็อกไดอะแกรมทำประมาณเชิงเส้นมาสร้างสัญญาณเสียงพูดได้ดังรูปที่ 2.9



รูปที่ 2.9 บล็อกไดอะแกรมแสดงโมเดลการสร้างสัญญาณเสียงพูดอย่างง่าย

จากรูปที่ 2.7 สามารถเขียนสมการได้เป็น

$$S(n) = G * u(n) + \sum_{k=1}^{p} \alpha_k S(n-k) \tag{2.7}$$

การประมาณเชิงเส้นโดยใช้สัมประสิทธิ์ $\{\alpha_k\}$ คือ

$$S'(n) = \sum_{k-1}^{p} \alpha_k S(n-k) \tag{2.8}$$

ดังนั้น ความคลาดเคลื่อน คือ

$$e(n) = S(n) - S'(n)$$

$$e(n) = S(n) - \sum_{k-1}^{p} \alpha_k S(n-k) \tag{2.9}$$

ฟังก์ชันถ่ายโอนระหว่าง e(n) และ S(n) คือ

$$A(z) = E(z) / S(z)$$

$$= 1 - \sum_{k=1}^{p} \alpha_{k^z}{}^{-k} \tag{2.10}$$

จากสมการที่ (2.7) และ (2.9) จะเห็นได้ว่าถ้า $\{\alpha_k\} = \{a_k\}$ แล้ว

$$e(n) = G * u(n) \tag{2.11}$$

ดังนั้น ค่าผลรวมของกำลังสองของความคลาดเคลื่อน

$$E_n = \sum_m e_n^2(m)$$

$$E_n = \sum_m [S(m) - S'(m)]^2 \tag{2.12}$$

โดยที่ n คือ ช่วงที่ n ของสัญญาณที่ใช้คำนวณ เพราะฉะนั้นเพื่อให้ได้ค่า E_n ต่ำที่สุดจะต้องมีเงื่อนไขว่า

$$\frac{\partial E_n}{\partial \alpha_i} = 0 \qquad \text{เมื่อ} \qquad i = 1, 2, 3, \ldots, p$$

จากสมการ (2.12)

$$\frac{\partial E_n}{\partial \alpha_i} = 2s(m-i)\sum_m \left[s_n(m) - \sum_{k=1}^{p} \alpha_k s_n(m-k) \right] \quad \text{เมื่อ } i = 1, 2, 3, \ldots..p$$

$$= -2\left[\sum_m (m)s(m-i) - \sum_{k=1}^{p} \sum_m \alpha_k s(m-k)s(m-i) \right]$$

$$\frac{\partial E_n}{\partial \alpha_i} = 0 \qquad \text{ก็ต่อเมื่อ}$$

$$\sum_{k=1}^{p} \alpha_k \sum_m s_n(m-k)s_n(m-i) = \sum_m s_n(m)s_n(m-i) \tag{2.13}$$

เมื่อ i = 1, 2, 3,....,p

ถ้าเรากำหนดให้ $\phi_n(i,j) = \sum s_n(m-k)s_n(m-i)$ เพราะฉะนั้น

$$\sum_{k=1}^{p} \alpha_k \phi_n(i,j) = \phi_n(i,o) \tag{2.14}$$

โดยสมการ (2.12)-(2.13) จะได้ว่า

$$E_n = \sum_m s_n^2(m) - \sum_{k=1}^{p} \alpha_k \sum_m s_n(m)s_n(m-k)$$

และจาก $\phi_n(i,j) = \sum s_n(m-k)s_n(m-i)$

$$E_n = \phi_n(0,0) - \sum_{k=1}^{p} \alpha_k \phi_n(o,k) \tag{2.15}$$

สมมติว่าใน 1 เฟรม ของสัญญาณที่ตัดมามีจำนวน N ตัวอย่าง คือ $s_n(0)$, $s_n(1)$, $s_n(2)$,..., $s_n(N-1)$ ในที่นี้เราให้ $s_n(m) = 0$ เมื่อ $m < 0$ หรือ $m > N-1$ เพราะฉะนั้น

$$\phi_n(i,j) = \sum_m s_n(m-k)s_n(m-i)$$

$$= \sum_{m=0}^{N-I-(t-k)} s_n(m)s_n(m+i-k) \qquad 0 \le k \le p, 1 \le i \le p$$

ให้ $$R_n(k) = \sum_{m}^{N-I-k} s_n(m)s_n(m+k) \qquad \text{เมื่อ} \quad k= 0, 1, 2,\ldots p \qquad (2.16)$$

ดังนั้น จากสมการ (2.5) และ (2.16) จะได้ว่า

$$R_n(k) = \sum_{m=0}^{N-1-k} x'(m)x'(m+k) \qquad (2.17)$$

จากสมการที่ (2.14) จะได้ว่า

$$\sum_{k=1}^{p} \alpha_k R_n(|i-k|) = R_n(i) \qquad \text{เมื่อ} \quad i= 0, 1, 2,\ldots p$$

สามารถเขียนให้อยู่ในรูปเมตริกได้เป็น

$$\begin{bmatrix} R_n(0) & R_n(1) & \cdots & R_n(p-t) \\ R_n(1) & R_n(0) & \cdots & R_n(p-2) \\ \vdots & \vdots & \ddots & \\ R_n(p-t) & R_n(p-2) & \cdots & R_n(0) \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_p \end{bmatrix} = \begin{bmatrix} R_n(1) \\ R_n(2) \\ \vdots \\ R_n(p) \end{bmatrix}$$

หรือ

$$R_n, \alpha = r_n \qquad (2.18)$$

เมื่อ

$$R_n = \begin{bmatrix} R_n(0) & R_n(1) & \cdots & R_n(p-t) \\ R_n(1) & R_n(0) & \cdots & R_n(p-2) \\ \vdots & \vdots & \ddots & \vdots \\ R_n(p-t) & R_n(p-2) & \cdots & R_n(0) \end{bmatrix}, \alpha = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_p \end{bmatrix}, r_n = \begin{bmatrix} R_n(1) \\ R_n(2) \\ \vdots \\ R_n(p) \end{bmatrix}$$

2.4.2 การหาค่าของพารามิเตอร์ LPC

พารามิเตอร์ LPC ได้แก่ สัมประสิทธิ์ α และอัตราขยาย G

เมื่อเราได้ค่า $R_n(0), R_n(1), R_n(2),\ldots,R_n(p)$ จากสมการ (2.18) แล้วก็สามารถหาค่า α นั่นคือ

$$\alpha = R_n^{-1} \cdot r_n$$

และจากสมการ (2.7) และ (2.9) จะได้ว่า

$$e(n) = G \cdot u(n)$$

$$\therefore \qquad E_n = \sum_{m=0}^{N-1} e^2(m) = G \sum_{m=0}^{N-1} u^2(n) \qquad (2.19)$$

จากสมการ (2.15) จะได้ว่า

$$E_n = \phi_n(0,0) - \sum_{k=1}^{p} \alpha_k \phi_n(0,k)$$

$$= R_n(0) - \sum_{k=1}^{p} \alpha_k R_{n(k)} \qquad (2.20)$$

และจากสมการ (2.19) เราสามารถหาค่า G ได้ตรงจาก

$$G^2 = \frac{R_n(0) - \sum_{k=1}^{p} \alpha_k R_n(k)}{\sum_{m=0}^{N-1} u^2(m)} \qquad (2.21)$$

ขั้นตอนนี้จะได้วิเคราะห์สัญญาณไปหนึ่งเฟรมแล้ว โดยในหนึ่งเฟรมนี้จะได้ค่าสัมประสิทธิ์ทั้งหมด 12 ค่า (P=12) ; $\alpha_1, \alpha_2, \alpha_3, \alpha_p$

2.4.3 การเปลี่ยนพารามิเตอร์ LPC เป็นสัมประสิทธิ์เซปสตรัม

หลักจากการหาสัมประสิทธิ์ LPC และ Gain ใน 1 เฟรมแล้ว จะเปลี่ยนให้เป็น สัมประสิทธิ์เซปสตรัม เนื่องจากการรู้จำเสียงพูดนั้น สัมประสิทธิ์เซปสตรัมเป็นพารามิเตอร์ที่มีลักษณะน่าเชื่อถือได้ดีกว่าสัมประสิทธิ์ LPC ทั้งยังมีความสัมพันธ์ใกล้ชิดกับการรับรู้เสียง ตามความรู้สึกของมนุษย์โดยแท้จริง สัมประสิทธิ์เซปสตรัมสามารถหาได้โดยตรงจากสัมประสิทธิ์ LPC ดังนี้

$C_0 = InG$ เป็นสัมประสิทธิ์ตัวแรกซึ่งเป็นแกน

$Q \approx \frac{3}{2}p$ โดย p = 10 ดังนั้น Q = 12 ไม่รวมกับแกน (C_0) จะได้ว่า สัมประสิทธิ์เซปสตรัมใน 1 เฟรม = 12 ตัว

$$C_m = a_m + \sum_{k=1}^{m-1} \left(\frac{k}{m}\right) C_k a_{m-k} \quad ; \quad 1 \le m \le p$$

$$C_m = \sum_{k=1}^{m-1} \left(\frac{k}{m}\right) C_k a_{m-k} \quad ; \quad m > p$$

2.4.4 การเวทค่าพารามิเตอร์ (Parameter Weighting)

เนื่องจากสัมประสิทธิ์เซปสตรัมที่ได้นั้น ช่วงลำดับต้น ๆ และลำดับท้าย ๆ ของเฟรมที่นำมาวิเคราะห์ จะเกิดความคลาดเคลื่อนมากกว่าปริมาณส่วนอื่น เพราะฉะนั้นจึงทำการถ่วงน้ำหนักเพื่อลดค่าความคลาดเคลื่อนดังกล่าวนี้ ด้วยฟังก์ชันเวทติ้ง ดังนี้คือ

$$w_m = \left[1 + \frac{Q}{2}\sin\left(\frac{\pi m}{Q}\right)\right] \quad ; \quad 1 \le m \le Q \qquad (2.22)$$

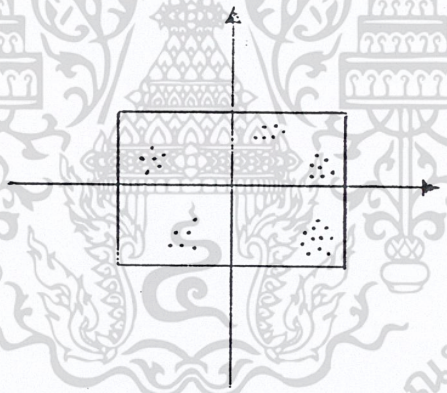
จะได้พารามิเตอร์สุดท้ายคือ

$$C'_m = C_m w_m \qquad (2.23)$$

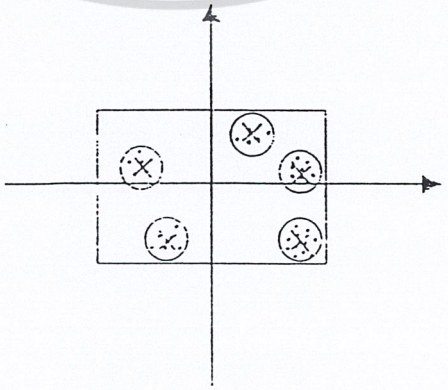
จากนั้นจะพิจารณาให้ครบทุกเฟรมของข้อมูล เมื่อพิจารณาเรียบร้อยแล้วก็จะนำไปจัดกลุ่มเสียงและสร้างจำลองเสียงเพื่อใช้ในการเปรียบเทียบต่อไป

2.5 การจัดระดับเวกเตอร์ (Vector Quantization)

เวกเตอร์ควอนไตซ์เซชั่น เป็นวิธีการลดไดเมนชั่น (dimension) หรือจำนวนข้อมูลให้น้อยลง เวกเตอร์อินพุต หรือ เซตเทรนนิ่ง หรือ พารามิเตอร์ที่ได้จากขั้น LPC จะถูกเลือกมากลุ่มหนึ่งซึ่งใช้เป็นตัวแทนของข้อมูลจำนวนหนึ่งหรือเรียกว่าการหา codebook อินพุตที่เข้ามาจะถูกทำการเปรียบเทียบกับ codebook ที่มีอยู่โดยจะพิจารณาว่าอินพุต ที่เข้ามานั้นห่างจาก codebook ใดน้อยที่สุด อินพุตดังกล่าวจะถูกแทนด้วยเวกเตอร์โค้ด (index) นั้น อินพุตทุกตัวเป็นสมาชิกของเวกเตอร์โค้ดใดๆ จะถูกนำมาหาจุดศูนย์กลางร่วมใหม่ และนำจุดศูนย์กลางที่ได้นี้ไปทำการหาความคลาดเคลื่อนกับสมาชิกทุกตัว ถ้าค่าความคลาดเคลื่อนที่ได้มีค่ามากกว่าค่าที่กำหนดไว้ค่าหนึ่งหรือค่าที่ยอมรับได้ ก็จะนำศูนย์กลางใหม่นั้นไปเป็น codebook แทน และจะทำการจัดกลุ่มอินพุตเข้ากับ codebook ใหม่ที่ได้และหาความคลาดเคลื่อนอีกครั้งทำอย่างนี้ซ้ำๆ จนกระทั่งค่าความคลาดเคลื่อนมีค่าน้อยถึงค่าที่ยอมรับได้ ก็จะถือได้ว่า codebook ที่ดีที่สุดจะเป็นตัวแทนของอินพุตทั้งหมด จะสังเกตได้ว่าทุกครั้งที่มีการหา codebook ใหม่นั้น ค่าความคาดเคลื่อนที่ได้จะมีค่าลดลงทุกครั้งด้วย

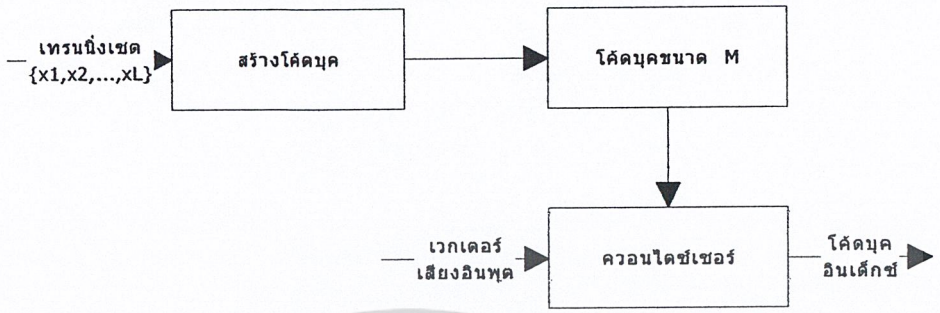


รูปที่ 2.10 แสดงการกระจายเฟรมของเสียงพูด โดยแต่ละจุดแทนเฟรมของเสียง



รูปที่ 2.11 การรวมกลุ่มของเฟรมเสียงเพื่อสร้างโค้ด X

ตัวอย่างเวกเตอร์ควอนไตซ์เซชั่น สมมติให้เวกเตอร์แต่ละตัวมี 2 มิติ และทำการหา Codebook ขนาด 8 เวกเตอร์ ทั้งหมดจะถูกจัดเข้ากลุ่มของ Codebook ต่างๆ แล้วทำการหาจุดศูนย์ใหม่โดยการเฉลี่ยค่าเวกเตอร์สมาชิกทุกตัวที่อยู่ในกลุ่มเดียวกัน ผลที่ได้คือ Codebook 8 ตัว เป็นตัวแทนของเวกเตอร์ทั้งหมด



รูปที่ 2.12 บล็อกไดอะแกรมของเวกเตอร์ควอนไตซ์

การทำงานของควอนไตซ์แบบเวกเตอร์ แบ่งเป็น 2 ขั้นตอนดังนี้

1. การสร้างโค้ดบุค (codebook) โดยวิธี K-means

จากขั้นตอนการประมาณเชิงเส้นเสียงตัวอย่างจำนวนมาก จะได้เทรนนิ่งเซตซึ่งประกอบด้วยเวกเตอร์สเปคตรัมจำนวน L เฟรม ; $X = \{x_i; 1 \le i \le L\}$ เฟรมละ P มิติ ; $X = \lfloor x_1, x_2, ..., x_p \rfloor$ แล้วนำข้อมูลที่ได้มาทำการสร้างกลุ่มของแบบอ้างอิง

ในระบบการรับรู้เสียงพูดแบบต่าง จะใช้อ้างอิงของคำหนึ่งๆ จากผู้พูดจำนวนมาก เพื่อที่จะได้ครอบคลุมถึงความแปรปรวนต่างๆ ที่เกิดระหว่างผู้พูดแต่ละคน เนื่องจาถ้าใช้แบบอ้างอิงจำนวนมาก เวลาที่ใช้ในการตอบสนองจะมาก เนื้อที่หน่วยความจำสำรองที่ใช้เก็บแบบอ้างอิงจะเพิ่ม และเมื่อเพิ่มแบบอ้างอิงไประดับหนึ่ง ความถูกต้องในการรับรู้จะเริ่มคงที่ ดังนั้นการจัดกลุ่มแบบอ้างอิงใหม่ เพื่อให้ได้แบบอ้างอิงที่เหมาะสม และสามารถใช้เป็นตัวแทนของแบบอ้างอิงที่มีอยู่ทั้งหมดได้ อัลกอริทึมที่ใช้ได้แก่ K-means Algorithm ขั้นตอนการสร้างโค้ดบุคมีดังนี้

1) นำเทรนนิ่งเซตมาใช้ในการสร้างโค้ดบุค

ขนาดโค้ดบุคของการควอนไตซ์ แบบเวกเตอร์ คือ $M=2^B$ เวกเตอร์ และเพื่อที่จะหาเซตของ M โค้ดบุคที่ดีที่สุด จำนวนเวกเตอร์อินพุตจะต้องมากกว่าขนาดโค้ดบุกมากๆ

2) การสุ่มค่าเริ่มต้น

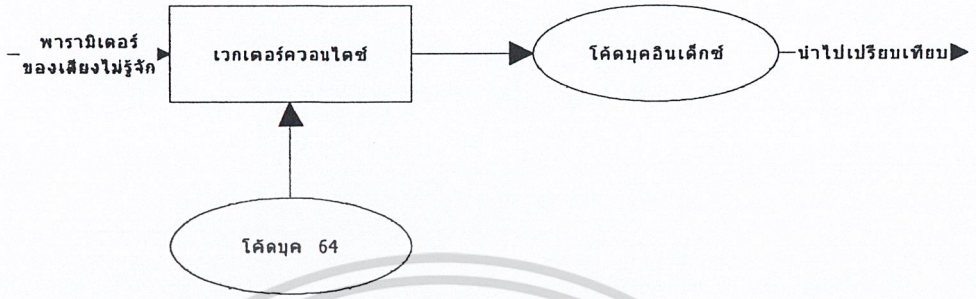
การสุ่มค่าเริ่มต้น เป็นวิธีหนึ่งในการออกแบบโค้ดบุค ซึ่งคือ การเลือกค่าเริ่มต้นของโค้ดบุคเรียกโค้ดบุค ที่ได้จากการสุ่มค่าเริ่มต้นนี้ว่า แรนดอมโค้ดบุค (random codebook) ถึงแม้วิธีนี้จะไม่ใช่วิธีที่ดีนัก แต่โค้ดบุกที่ได้จากการสุ่มก็ได้ผลเป็นที่ยอมรับ

3) การหาความคลาดเคลื่อน

การหาความคลาดเคลื่อน เป็นส่วนที่จำเป็นและเป็นประโยชน์ต่อการออกแบบโค้ดบุค สมการทางพีชคณิตที่ใช้ในการหาระยะทาง มีหลายวิธี แต่ที่นำมาใช้ก็คือ การหาความคลาดเคลื่อนกำลังสองรวม (Total square error) ซึ่งเป็นวิธีที่ง่ายและรวดเร็ว

ถ้าสัญญาณมี P มิติ สามารถหาระยะห่างระหว่างสัญญาณอินพุต (x) กับเวกเตอร์โค้ด (y) โดยสมการ

$$d(v_1, v_2) = \|v_1 - v_2\|^2 = \sum_{i=0}^{k-1} (x_1 - y_1)^2 \qquad (2.24)$$



รูปที่ 2.13 ขั้นตอนของเวกเตอร์ควอนไตเซชั่น

4) การจัดกลุ่ม (classification) และการหาจุดศูนย์กลางของกลุ่ม (center cluster)

การจัดกลุ่มเป็นการแบ่งเวกเตอร์อินพุตเข้าไปตามกลุ่มต่างๆ ของแรนดอมโค้ดบุค โดยพิจารณาระยะทางหรือความคลาดเคลื่อนน้อยที่สุด ของแต่ละเวกเตอร์อินพุต x กับเวกเตอร์โค้ดบุค y ซึ่งเป็นโค้ดบุค จากนั้นจะทำการหาค่าเฉลี่ยของแต่ละกลุ่ม เพื่อเป็นค่ากลางของกลุ่มนั้นๆ จะได้

$$\bar{y} = \frac{1}{L} \sum_{i=1}^{L} x_i$$

$\bar{y}$ เป็นจุดศูนย์กลางซึ่งเป็นเวกเตอร์ที่อยู่ตรงกลางของ $\{x_i\}_{i=1}^{L}$ ซึ่งแต่ละมิติจะไม่ขึ้นแก่กันหมายความว่าแต่ละ y_k เป็นค่ากลางของ $\{x_i\}_{i=1}^{L}$ ทำ 2 ขั้นตอน จะเกิดการลู่เข้า (convergent) โดยความคลาดเคลื่อนรวมจะต่ำกว่าค่าหนึ่งๆ ซึ่งค่าความคลาดเคลื่อนรวมจะลดลงทุกครั้งที่มีการคำนวณซ้ำใหม่ จึงขึ้นกับค่าที่กำหนดว่าจะต้องการให้ความคลาดเคลื่อนรวมร้อยกว่า ค่ากลางดังกล่าวของแต่ละกลุ่มจะถูกเก็บเวกเตอร์โค้ดจะได้ว่า y เป็นควอนไตซ์ของ ค่า x

โดย q(.) เป็นโอเปอร์เรเตอร์ของควอนไตซ์ y ถูกเรียกว่าเอาท์พุตเวกเตอร์ของค่า x โดย y เป็นค่าใดค่าหนึ่งใน $Y = \{y_i; 1 \le i \le M\}$ โดย $y_i = [y_{i1} \quad y_{i2} \quad \dots \quad y_{ip}]$ Y เป็นเซตของโค้ดบุค M เป็นขนาดของโค้ดบุค และ $\{y_i\}$ เป็นเซตของเวกเตอร์โค้ด y_i อาจเรียกว่าโค้ดอ้างอิง และ M อาจเรียกว่าจำนวนระดับชั้น จำทำการแบ่งเวกเตอร์ x ไปใน M เซลล์ $\{C_i; 1 \le i \le M\}$ เมื่อ X อยู่ในเซล C_i

$$q(x) = y_i \quad \text{ถ้า} \quad x \in C_i$$

2. ขั้นตอนการเปรียบเทียบ

เวกเตอร์ควอนไตเซชั่น ที่ใช้เพื่อการออกแบบรับรู้เสียงพูดนั้น มีจำนวนควอนไตเซอร์ M ตัวซึ่งหมายถึงมี M ระดับเสียงเพื่อการรับรู้ แต่ละระดับเสียงพิจารณามาจากเซลของข้อมูลเทรนนิ่ง ซึ่ง M เป็นศัชนีระดับ แต่ละเซตของเทรนนิ่งในแต่ละระดับจะเก็บเสียงที่อยู่ในระดับเดียวกัน

เมื่อมีเสียงที่เราไม่ทราบ (unknown) ; x_i เข้ามา จะเป็นอินพุตเข้าไปยังทุกๆ ควอนไตเซอร์ค่าดัชนีระดับ (index) ที่ถูกเลือก จะเป็นระดับที่มีความคลาดเคลื่อนน้อยที่สุด D(c) เมื่อ i = 1, 2,...,M ซึ่งความคลาดเคลื่อนเฉลี่ยนั้นหาได้จากการวัดระยะทางโดยใช้วิธีการหาความคลาดเคลื่อนกำลังสองรวม

2.6 การสร้างแบบจำลอง (Pattern Matching)

หมายถึง การกำหนดรูปแบบขึ้นนำมาเปรียบเทียบกับสิ่งที่เราไม่ทราบ โดยการจัดกลุ่มของสัญญาณที่ไม่รู้จัก (unknown signal) ซึ่งเป็นสัญญาณที่ไม่คงที่ให้อยู่ในกลุ่มใดกลุ่มหนึ่ง วิธีที่เลือกใช้ในการสร้างและหารูปแบบของคำพูดที่เหมือนอยู่สองแนวทาง

- Dynamic Time Warping (DTW)
- Hidden Markov Models (HMM)

วิธีที่เลือกใช้ในการสร้างแบบจำลองคือ Hidden Markov Models (HMM) ซึ่งเป็นวิธีการสร้างแบบจำลองที่นำมาประยุกต์ใช้ในการจำเสียงพูด แบบจำลองมาร์คอฟ แบ่งออกเป็น 2 ประเภท คือ แบบต่อเนื่อง (Continuous) และแบบไม่ต่อเนื่อง (Discrete-time) ในที่นี้จะเลือกแบบไม่ต่อเนื่องเพราะเป็นวิธีการที่ซับซ้อนน้อยกว่าและใช้กับคำพูดสั้น ๆ

Hidden Markov Models (HMM)

HMM เป็นวิธีการที่ใช้ทฤษฎีความน่าจะเป็นมาอธิบายการเกิดเหตุการณ์ของลำดับปรากฏการณ์ 2 ลำดับคือ สเตท (State) และ ปรากฏ (observation) โดยเหตุการณ์ที่สังเกตเห็นจะเห็นได้เพียงเอาต์พุตของแต่ละสเตทในลักษณะที่เป็นค่าปรากฏเท่านั้น จะไม่ทราบแน่ชัดว่าอยู่ในสเตทใด

ส่วนประกอบของแบบจำลอง

1. N คือจำนวนสเตทในแบบจำลอง โดยสามารถย้ายจากสเตทหนึ่งไปยังอีกสเตทหนึ่งได้ เราใช้เซตของสเตทเป็น {1,2,.....,N} และสเตทที่เวลา t ใดๆ เป็น q_t

2. M คือจำนวนของค่าต่อหนึ่งสเตท แทนด้วยสัญลักษณ์ $V=\{v_1,v_2,\ldots,v_M\}$

3. $A = \{a_{ij}\}$ คือความน่าจะเป็นในการเป็นสเตทที่ $a_{ij} = P\{q_t = j | q_{t-1} i\}$ เมื่อ $1 \le i, j \le N$

4. $B = \{b_i(k)\}$ คือความน่าจะเป็นของการเกิดค่าปรากฏที่สเตท $b_i(k) = P\{O_t = v_k | q_t = i\}$ เมื่อ i = 1, 2, 3,...,N

5. $\pi = \{\pi_I\}$ คือค่าความน่าจะเป็นที่สเตทจะเห็นสเตทเริ่มต้นเมื่อ $\pi_I = P(q_1 = i)$

โครงสร้างแบบจำลอง HMM

แบ่งตามลักษณะการเปลี่ยนสเตท (transition) ของเมตริกซ์ A

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$$

1. แบบ Egordic Model หรือ Fully Connected Model แบบจำลองนี้ทุกสเตทสามารถเปลี่ยนสเตทไปยังสเตทอื่นๆ ได้ทุกๆ สเตท ดังรูปที่ 2.14 เป็นตัวอย่างของแบบจำลองที่มี N-4

2. แบบ Left-Right Model หรือ Bakis Model แบบจำลองนี้การเปลี่ยนสเตทจะเปลี่ยนจากซ้ายไปขวามีคุณสมบัติการเปลี่ยนดังนี้

2.1 $a_{ij} = 0; J < i$ หมายความว่าเมื่อผ่านสเตทใดไปแล้วจะไม่มีการย้อนกลับมายังสเตทนั้น

2.2 $\pi_i = 1; i = 1$ หมายความว่า ลำดับของสเตทต้องเริ่มต้นที่สเตทที่ 1 สเตทที่เหลือจึงมีความน่าจะเป็นที่จะเป็นสเตทเริ่มต้นเท่ากับศูนย์

Lift-Right Model นี้มีกฎข้อบังคับการเปลี่ยนแปลงสเตทดังนี้

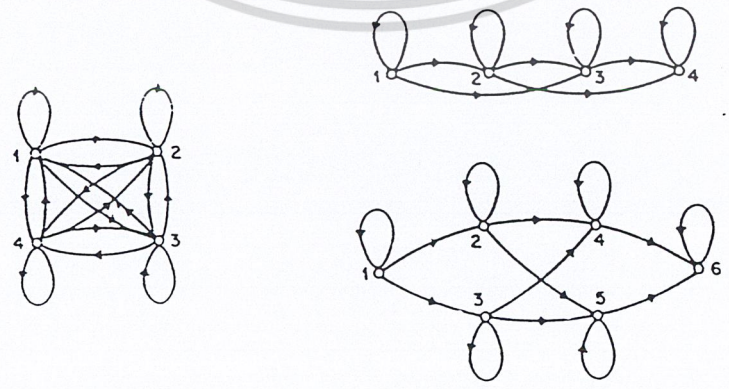
$A_{ij} = 0$ เมื่อ $J > i + \Delta i$ โดยค่าของ $\Delta i = 2$ หมายความว่าการเปลี่ยนสเตทจะสามารถกระโดดข้ามเปลี่ยนไปยังสเตทข้างหน้าได้ไม่เกิน 2 สเตท จะได้เมตริกการเปลี่ยนสเตทเป็น

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 \\ 0 & a_{22} & a_{23} & a_{24} \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & a_{44} \end{bmatrix}$$

จะเห็นว่าสเตทสุดท้ายมีสัมประสิทธิการเปลี่ยนสเตทเป็น $a_{NN} = 1$, $a_{Ni} = 0$ เมื่อ I<N

แบบจำลองนี้เหมาะกับสัญญาณที่มีการเปลี่ยนแปลงต่อเนื่องเช่น คำพูด

3. แบบ Parallel Lift-Right Model มีคุณสมบัติการเปลี่ยนสเตทคล้ายแบบที่ 2 แต่มีความยืดหยุ่น



รูปที่ 2.14 แสดงแบบจำลองต่างๆ ของ HMM

ปัญหาของ HMM

ปัญหาของ HMM มี 3 ข้อ ซึ่งต้องใช้อัลกอริทึมวิธีต่างๆ ในการคำนวณเพื่อแก้ปัญหานี้

ปัญหาที่ 1 เมื่อลำดับของค่าปรากฏ $O = \{o_1, o_2, ..., o_T\}$และมีโมเดล $\lambda = (A, B, \pi)$ เราจะคำนวณหาค่า $P(O|\lambda)$ของลำดับของค่าปรากฏได้อย่างไร

ปัญหาที่ 2 เมื่อลำดับของค่าปรากฏ $O = \{o_1, o_2, ..., o_T\}$และแบบจำลอง $\lambda = (A, B, \pi)$เราจะหาลำดับสเตท $q = \{q_1, q_2, ..., q_T\}$ ที่เหมาะสมในการให้ค่าปรากฏนั้นได้อย่างไร

ปัญหาที่ 3 จะหาแบบจำลอง $\lambda = (A, B, \pi)$ที่ให้ค่า $P(O|\lambda)$มากที่สุดได้อย่างไร

ลำดับของค่าปรากฏที่ใช้ปรับค่าพารามิเตอร์ A, B และ π เพื่อให้ได้แบบจำลองที่ดีที่สุดนั้นเรียกว่าลำดับเทรนนิ่ง (training sequence)

การคำนวณเพื่อแก้ปัญหาของ HMM

1) การแก้ปัญหาข้อที่ 1 เป็นการคำนวณว่าแบบจำลอง λ ให้ได้ความน่าจะเป็นที่จะได้ลำดับว่าปรากฏมากน้อยเพียงใด มีวิธีการเพื่อช่วยแก้ปัญหาโดยกระบวนการต่อไปนี้

1.1 กระบวนการไปข้างหน้า (Forward Procedure)

เมื่อกำหนดให้ตัวแปรไปข้างหน้า (Forward variable)

$$\alpha_i(i) = P(o_1, o_2 ... o_T, q_t = i|\lambda)$$

หมายถึงความน่าจะเป็นของการเกิดลำดับปรากฏ $O_1, O_2 ... O_T$ ที่อยู่ในสเตท i ณ เวลา t โดยมีแบบจำลองเป็น λ โดยสามารถหา $\alpha_i(i)$ ได้ดังนี้

1.1.1 การเริ่มต้น (initialization) เมื่อกำหนด $\alpha_t(i) = \pi_i b_i(o_i)$ ที่เวลาเริ่มต้น t=1 และเหตุการณ์เริ่มต้น O_1 เมื่อ $1 \leq i \leq N$

1.1.2 การเหนี่ยวนำ (induction)

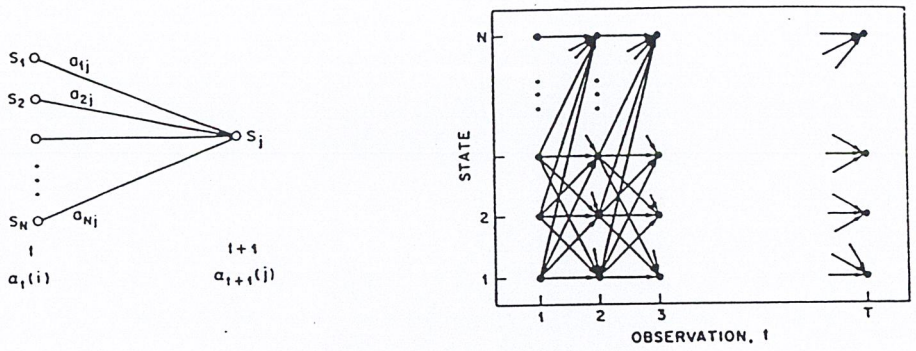
$$\alpha_{t=1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(o_{t+1}) \qquad (2.25)$$

เมื่อ $1 \leq t \leq T-1$ หมายถึง ความน่าจะเป็นสเตท j ที่เวลา t+1 ได้มาจากสเตท I ที่เป็นไปได้ถึง N สเตทที่เวลา t ดังรูปที่ 2.16

1.1.3 การสิ้นสุด (termination)

$$P(o|\lambda) = \sum_{i=1}^{N} \alpha_t(i) \qquad (2.26)$$

ความน่าจะเป็นของลำดับค่าปรากฏ O ได้จากผลรวม $\alpha_t(i)$ จากทุกๆ สเตทเมื่อ $1 \leq j \leq N$



รูปที่ 2.15 กระบวนการไปข้างหน้า

1.2 กระบวนการย้อนกลับ (Backward Procedure)

กำหนดให้ตัวแปรย้อนกลับ(Backward variable) $\beta_T(i) = P(o_{t+1}, o_{t+2}, \ldots, o_T, q_t = i|\lambda)$ หมายถึง ของลำดับค่าปรากฏส่วนหลัง จากเวลา t+1 ไปจนจบ โดยกำหนดว่าต้องอยู่ที่สเตท I ความน่าจะเป็นของลำดับค่าปรากฏส่วนหลัง จากเวลา ที่เวลา t และมีการจำลองเป็น λ เราจะคำนวณหา $\beta_{T(i)}$ ได้ดังนี้

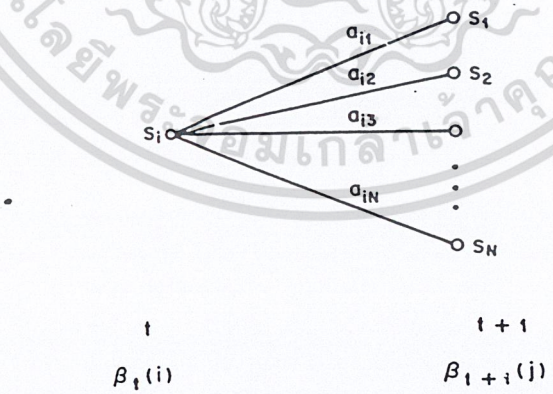
1. การเริ่มต้น (initialzation)

$$\beta_T(i) = 1 \text{ เมื่อ } 1 \le j \le N$$

2. การเหนี่ยวนำ (induction)

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(o_{t+1}) \beta_{t+1}(i) \qquad (2.27)$$

เมื่อ $t = T-1, T-2, \ldots, 1$ และ $1 \le i \le N$



รูปที่ 2.16 กระบวนการย้อนกลับ

รูปที่ 2.16 แสดงถึงค่าปรากฏที่จะอยู่ที่สเตท i ที่เวลา t โดยคำนึงถึงลำดับค่าปรากฏจากเวลา t+1 ซึ่งต้องพิจารณาสเตท j ที่จะเป็นไปได้ทั้งหมด ณ เวลา t+1 โดยจะขึ้นอยู่กับค่า a_{ij} และ $b_j(O_{t+1})$

2) การแก้ปัญหาที่ 2 เพื่อหาลำดับสเตทที่เหมาะสม

เราจะใช้**วิธี** วิทเทอร์บี อัลกอริทึม (Viterbi Algorithm) เพื่อหาลำดับสเตทที่ดีที่สุด ณ เวลา t หนึ่งๆ เมื่อกำหนดลำดับเหตุการณ์ $O = (o_1, o_2, \ldots, o_T)$ โดยนิยามให้

$$\delta_t(i) = \max_{q_1, q_2, \ldots, q_t} P[q_1, q_2, \ldots, q_{t-1}, q_t = i, o_1, o_2, \ldots, o_t | \lambda]$$

หมายถึง ความน่าจะเป็นสูงสุดของเส้นทาง (path) ณ เวลา t ซึ่งเริ่มนับจากเหตุการณ์ที่เวลาเริ่มต้น จนถึงเวลา t สเตท i และโดยการอาศัยคุณสมบัติการเหนี่ยวนำ (induction) เราจะได้

$$\delta_{t+1}(j) = [\max \delta_t(i) a_{ij}] b_j(o_{t+1})$$

เราสามารถหาลำดับสเตทที่ดีที่สุดโดยใช้กระบวนการต่อไปนี้ เมื่อกำหนดให้ $\psi_{t(i)}$ เป็นอาเรย์ (array)

1. การเริ่มต้น (initialization)

$$\delta_1(i) = \pi_i b_i(o_1) \quad \text{เมื่อ} \quad 1 \le i \le N$$

$$\psi_1(i) = 0$$

2. การย้อนกลับ (recursion)]

$$\delta_t(j) = \max_{1 \le i \le N} [\delta_{t-1}(i) a_{ij}] b_j(o_t) \quad \text{เมื่อ} \quad 2 \le t \le T, 1 \le j \le N$$

$$\psi_t(j) = \arg \max_{1 \le i \le N} [\delta_{t-1}(i) a_{ij}] \quad \text{เมื่อ} \quad 2 \le t \le T, 1 \le j \le N$$

3. การสิ้นสุด (termination)

$$P^* = \max_{1 \le i \le N} [\delta_T(i)]$$

$$P_t^* = \arg \max_{1 \le i \le N} [\delta_T(i)]$$

4. เส้นทางเดินย้อนกลับ (Path backtracking)

$$q_t^* = \psi_{t+1}(q_{t+1}^*) \quad \text{เมื่อ} \quad t = T\text{-}1, T-2, \ldots, 2, 1$$

3) การแก้ปัญหาที่ 3 เพื่อหาโมเดลที่จะให้ผลตามลำดับค่าปรากฏหนึ่งๆ โดยเลือกค่าพารามิเตอร์ A, B, π ที่ดีที่สุด โดยใช้ กระบวนการทำซ้ำ (Iterative) วิธีที่เราเลือกใช้คือวิธี บาม – เวลช์ (Baum – Welch) **หรือ** EM (Expectation – Maximization)

เมื่อนิยามให้

1. $\gamma_t(i) = P(q_t = i | O, \lambda)$

หมายถึงความน่าจะเป็นที่อยู่สเตท i ณ เวลา t โดยกำหนดลำดับเหตุการณ์ O และแบบจำลอง λ ให้สามารถแสดงค่า $\gamma_t(i)$ ได้ดังนี้

$$\gamma_t(i) = \frac{P(O, q_t = i | \lambda)}{P(O | \lambda)}$$

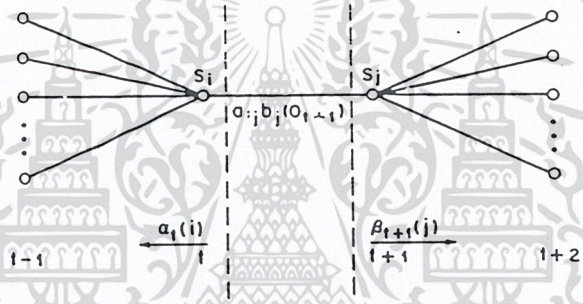
$$= \frac{P(O, q_t = i|\lambda)}{\sum_{i=1}^{N} P(O, q_t = i|\lambda)}$$

เนื่องจาก $P(O, q_t = i|\lambda)$ มีค่าเท่ากับ $\alpha_t(i)\beta_t(i)$ จึงทำให้

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(t)}{\sum_{I=1}^{N} \alpha_t(i)\beta_t(i)} \qquad (2.28)$$

2. $\varepsilon_t(I, j) = P(q_t = I, q_{t+1} j|O, \lambda)$

หมายถึง ความน่าจะเป็นที่จะอยู่ที่สเตท i ที่เวลา t และ เสตท j ที่เวลา t+1 เมื่อกำหนดแบบจำลอง และลำดับค่าปรากฏให้



รูปที่ 2.17 แสดงลำดับการคำนวณการเกิดค่าปรากฏร่วมซึ่งจอยู่สเตท i ที่เวลา t และอยู่ที่สเตท j ที่เวลา t+1

ซึ่งจากนิยามของตัวแปรไปข้างหน้า และตัวแปรย้อนกลับ สามารถนำมาสัมพันธ์กับ $\varepsilon_t(i, j)$

$$\varepsilon_t(i, j) = \frac{P(q_t = i, q_{t+1} = j, O|\lambda)}{P(O|\lambda)}$$

$$= \frac{\alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)} \qquad (2.29)$$

และจะได้ความสัมพันธ์ของ $\gamma_t(i)$ กับ $\varepsilon_t(i, j)$ ดังนี้

$$\gamma_t(i) = \sum_{j=i}^{N} \varepsilon_t(i, j) \qquad (2.30)$$

และ

$\sum_{t=1}^{T-1} \gamma_t(i)$ = จำนวนของการเปลี่ยนสเตทออกจากสเตท i ในลำดับค่าปรากฏ O

$\sum_{t=1}^{T-1}\varepsilon_t(i)$ = จำนวนของการเปลี่ยนสเตทจากสเตท i ไป j ในลำดับค่าปรากฏ O

ดังนั้นสามารถหาค่าพารามิเตอร์ได้ดังนี้

$$\pi_i' = \gamma_1(i) \text{ เมื่อ } 1 \le i \le N \tag{2.31}$$

$$a_{ij}' = \frac{\sum_{t=1}^{T}\varepsilon_{t-1}(i,j)}{\sum_{t=1}^{T}\gamma_{t-1}(i)} \tag{2.32}$$

$$b_j'(k) = \frac{\sum_{t=1,O_1=v_k}^{T}\gamma_t(i)}{\sum_{t=1}^{T}\gamma_t(i)} \tag{2.33}$$

จากกระบวนการข้างต้น ถ้าเราคำนวณซ้ำๆ โดยให้ $\lambda' = (A', B', \pi')$ แทน $\lambda = (A, B, \pi)$ ซึ่งเป็นแบบจำลองเริ่มต้นแล้ว จะทำให้ความน่าจะเป็นของการเกิดลำดับค่าปรากฏ O ดีขึ้น จนกระทั่งถึงจุดวิกฤต ซึ่งเราจะได้จุดวิกฤตของฟังก์ชันความน่าจะเป็นในกรณีที่ $\lambda' = \lambda$ หรือถ้า λ' มีความน่าจะเป็นมากกว่าแบบจำลอง λ ในลักษณะที่ $P(O|\lambda') > P(O|\lambda)$ นั่นก็คือเราก็จะได้แบบจำลอง λ' ใหม่ ที่น่าจะทำให้เกิดลำดับปรากฏ O ได้ดีกว่า

การปรับค่าพารามิเตอร์ของ Hidden Markov Model

1. การสเกลลิ่ง (Scaling)

เนื่องจาก $\alpha_t(i)$ จะประกอบไปด้วยผลรวมของเทอมจำนวนมาก ซึ่งก็คือ

$$\left(\prod_{s=1}^{t-1} a_{q_s q_{s+1}} \prod_{s=1}^{t} b_{q_s}(O_s)\right)$$

และเนื่องจากแต่ละเทอมของ a และ b มีค่าน้อยกว่า 1 อยู่แล้วเมื่อพิจารณาผลรวมของการคูณค่าจะทำให้ค่าที่ได้น้อยลงเรื่อยๆ แสดงว่าเมื่อ t มากขึ้น แต่ละเทอมของ $\alpha_t(i)$ จะเข้าสู่ศูนย์ทำให้ Dynamic Range ของการคำนวณ $\alpha_t(i)$ เกิน Range ของคอมพิวเตอร์ ทำให้ค่าที่ได้ไม่ถูกต้อง จึงได้มีการสเกลลิ่งเพื่อทำให้ $\alpha_t(i)$ อยู่ภายใน Dynamic Range ของคอมพิวเตอร์ การสเกลลิ่งทำได้โดยการคูณ $\alpha_t(i)$ โดยสัมประสิทธิ์การสเกลลิ่ง ซึ่งสัมประสิทธิ์นี้ไม่ขึ้นอยู่กับ i การสเกลลิ่ง $B_t(i)$ ก็เช่นเดียวกัน หลังจากคำนวณค่า การสเกลลิ่งก็จะตัดกันหมดไปเอง พิจารณาจากสมการ

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1}\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{\sum_{t=1}^{T}\sum_{J=1}^{N}\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)} \tag{2.34}$$

เมื่อให้ $\alpha_t(i)$ แทน α ที่ยังไม่ได้สเกลลิ่ง

$\alpha_t(i)$ เป็น α ที่สเกลลิ่งแล้ว

$\alpha_t(i)$ แทนเวอร์ชั่นของ α ก่อนการสเกลลิ่ง

เมื่อ t=1 จะได้ $\alpha_t(i) = c_t\alpha_t(i)$

เมื่อ

$$c_t = \frac{1}{\sum_{i=1}^{N}\alpha_t(i)}$$

เมื่อ $2 \le t \le T$ คำนวณ $\alpha_t(i)$ จากสมการ (2.25) ในเทอมของ $\alpha_t(i)$ ค่าก่อน

$$\hat{\alpha}_r(i) = \sum_{j=1}^{N}\hat{\alpha}_{t-1}(i)a_{ij}b_i(O_t) \tag{2.35}$$

เมื่อสัมประสิทธิ์การสเกลลิ่งเป็น

$$c_1 = \frac{1}{\sum_{i=1}^{N}\hat{\alpha}_t(i)}$$

เมื่อให้ $\alpha_t(i) = c_t\alpha_t(i)$

จากสมการ (2.35) จะเขียนได้ว่า

$$\alpha_t(i) = \frac{\sum_{j=1}^{N}\hat{\alpha}_{t-1}(j)a_{ji}b_i(O_t)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\hat{\alpha}_{t-1}(j)a_{ji}b_j(O_t)} \tag{2.36}$$

และโดยการเหนี่ยวนำจะได้

$$\hat{\alpha}_{t-1}(j) = \left(\prod_{\tau=1}^{t-1}c_\tau\right)\alpha_{t-1}(j)$$

จะได้ว่า

$$\bar{a}_{ij}(i) = \frac{\sum_{j=1}^{N}\alpha_{t-1}(j)\left(\prod_{\tau=1}^{t-1}c_\tau\right)a_{ji}b_i(o_t)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_{t-1}(j)\left(\prod_{\tau=1}^{t-1}c_\tau\right)a_{ji}b_i(o_t)} = \frac{\alpha_t(i)}{\sum_{i=1}^{N}\alpha_t(i)} \tag{2.37}$$

นั่นคือจะสเกล $\alpha_t(i)$ ได้โดยหารด้วยผลรวมของ $\alpha_t(i)$ ทั้งหมดและสเกล $B_t(i)$ด้วยค่าเดียวกันนี้ในเทอมของการสเกลนี้สมการ (2.34) จะเป็น

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1}\alpha_t(i)a_{ij}b_j(O_{t+1})\hat{\beta}_{t+1}(j)}{\sum_{t=1}^{T-1}\sum_{j=1}^{N}\hat{\alpha}_t(i)a_{ij}b_i(O_{t+1})\hat{\beta}_{t+1}(j)} \tag{2.38}$$

โดยแต่ละ $\alpha_t(i)$, $B_{t+1}(i)$ จะได้เป็น

$$\alpha_t(i) = \left[\prod_{s=1}^{t} c_s\right]\hat{\alpha}_t(i) = c_t\alpha_t(i) \tag{2.39}$$

$$\beta_{t+1}(i) = \left[\prod_{s=t+1}^{T} c_s\right]\hat{\beta}_{t+1}(j) = D_{t+1}\beta_{t+1}(j) \tag{2.40}$$

ดังนั้นสมการ (2.38) จะเขียนได้เป็น

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1} C_t\alpha_t(i)a_{ij}b_i(o_{t+1})D_{t+1}\beta_{t+1}(j)}{\sum_{t=1}^{T-1}\sum_{j=1}^{N} C_t\alpha_t(i)a_{ij}b_j(o_{t+1})D_{t+1}\beta_{t+1}(j)} \tag{2.41}$$

ซึ่งเทอม C_tD_{t+1} จะเขียนในเทอม

$$C_tD_{t+1} = \prod_{s=1}^{t} c_s \prod_{s=t+1}^{T} c_s = \prod_{s=1}^{T} c_s = C_t \tag{2.42}$$

ซึ่งไม่ขึ้นกับเวลา t ดังนั้น C_tD_{t+1} จะตัดทิ้ง ทั้งเศษและส่วนของสมการ (2.41) ซึ่งทำให้สูตรการคำนวณซ้ำๆ (reestimate) เดิมกลับคืนมา กระบวนการสเกลลิ่ง ดังกล่าวนี้สามารถใช้ได้กับสัมประสิทธิ์ β และ π ในการสเกลลิ่งนี้จะทำให้การคำนวณค่า $P(O|\lambda)$ เปลี่ยนไปเราจะไม่สามารถหาได้จากการรวมกันของเทอม $\alpha_t(i)$ แต่จะหาได้จากคุณสมบัติ

$$\prod_{t=1}^{T} c_t \sum_{i=1}^{N} \alpha_T(i) = 1 \tag{2.43}$$

ดังนั้นจะได้

$$\prod_{t=1}^{T} c_t P(O|\lambda) = 1$$

$$P(O|\lambda) = \frac{1}{\prod_{t=1}^{T} c_t} \tag{2.44}$$

ทำให้อยู่ในรูปของ log ของ P เพื่อไม่ให้เกิน dynamic range ของคอมพิวเตอร์

$$\log[p(O|\lambda)] = \sum_{t=1}^{T} \log c_t \tag{2.45}$$

2. ลำดับของค่าปรากฏหลายเหตุการณ์ (Multiple Observation Sequence)

ในการใช้แบบจำลอง Left-Right นั้น การเทรนแบบจำลองต้องใช้หลายๆ เหตุการณ์ของลำดับค่าปรากฏเข้ามาเทรน เพื่อให้ได้ค่าพารามิเตอร์ที่ถูกต้องมากขึ้น

ถ้าให้เตของ v ลำดับค่าปรากฏเป็น

$$O = \left(O^{(1)}, O^{(2)}, \ldots . O^{(k)}\right)$$

เมื่อ $O = \left(O_1^{(k)}, O_2^{(k)}, \ldots . O_{Tk}^{(k)}\right)$ เป็นลำดับค่าปรากฏของเหตุการณ์ที่ v โดยให้แต่ละเหตุการณ์เป็นอิสระต่อกันจะได้

$$P(O|\lambda) = \prod_{k=1}^{K} P(O^{(k)}|\lambda)$$
$$= \prod_{k=1}^{K} P_k$$

นำเอาจำนวนเหตุการณ์ของการเกิดค่าปรากฏแต่ละเหตุการณ์มารวมกันจะได้สูตรหา a_I , $b_j(k)$ เป็น

$$\bar{a}_{ij} = \frac{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k-1} \alpha_t^k(i) a_{ij} b_j\left(o_{t+1}^{(k)}\right) \beta_t^k(j)}{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k-1} \alpha_t^k(i) \beta_t^k(i)} \tag{2.46}$$

$$\bar{b}_j(\ell) = \frac{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k-1} \alpha_t^k(i) \beta_t^k(i) \quad s.t. O_t = v\ell}{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k-1} \alpha_t^k(i) \beta_t^k(i)} \tag{2.47}$$

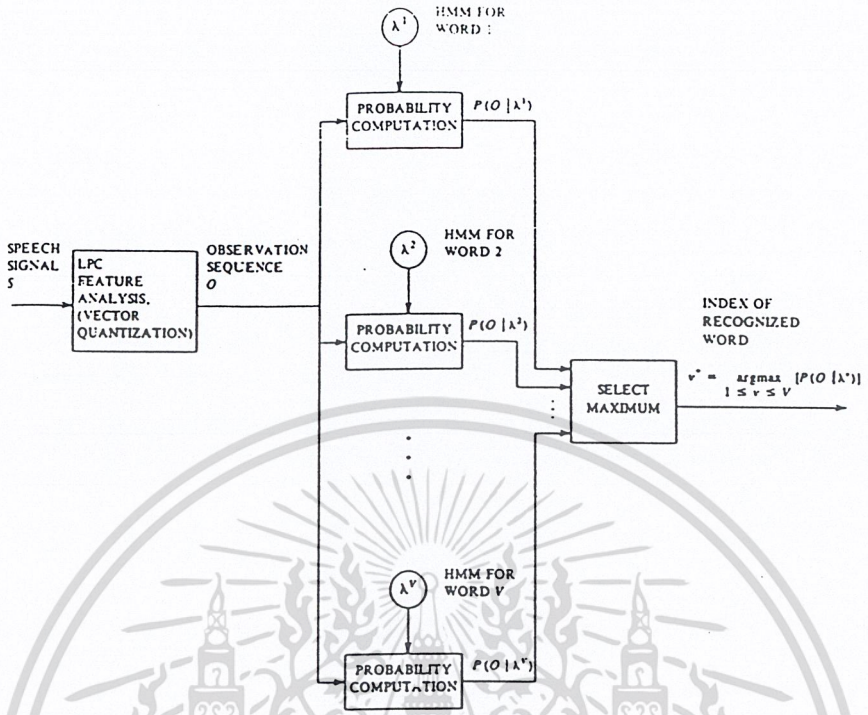
ส่วน π_i ไม่ต้องคำนวณเนื่องจาก $\pi_1 = 1, \pi_i = 0, i \neq 1$ จะได้การสเกลลิ่งที่เหมาะสมของสมการ (2.46) – (2.47)

$$\bar{a}_{ij} = \frac{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k-1} \hat{\alpha}_t^k(i) a_{ij} b_j\left(o_{t+1}^{(k)}\right) \beta_t^k(j)}{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k-1} \hat{\alpha}_t^k(i) \beta_t^k(i)} \tag{2.48}$$

$$\bar{b}_j(\ell) = \frac{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k-1} \alpha_t^k(i) \beta_t^k(i) \quad s.t. O_t = v\ell}{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k-1} \alpha_t^k(i) \beta_t^k(i)} \tag{2.49}$$

2.7 ระบบการจำลองมาร์คอฟ

เมื่อเรามีคำศัพท์อยู่ V คำ ในการทำการรู้จักได้ เราจะต้องสร้างแบบจำลองของคำ แต่ละคำที่แตกต่างกัน คำแต่ละคำจะมีลำดับเทรนนิ่งที่ได้จากคุณลักษณะเฉพาะของคำศัพท์นั้นๆ การที่เราจะรู้จักคำพูดได้ต้องทำได้ตามบล็อกไดอะแกรมดังต่อไปนี้



รูปที่ 2.18 บล็อกไดอะแกรมการรู้จำคำโดดด้วยแบบจำลองของมาร์คอฟ

1. เมื่อมีคำศัพท์อยู่ V คำ เราต้องสร้างแบบจำลองมาร์คอฟ : λ_v ของแต่ละคำนั้น นั่นคือการหาค่า (A, B, π) ที่เหมาะสมกับลำดับเทรนนิ่งของคำนั้นๆ

2. ในการจะรู้จำคำศัพท์แต่ละคำ เราจะทำการหาค่า $P(O|\lambda)$ของทุกๆ แบบจำลอง แล้วเลือกแบบจำลองที่มีค่าความน่าจะเป็นในการเกิดค่าปรากฏสูงสุดคือ

$$V^* = \arg\max P(o|\lambda_v)$$

คำศัพท์ที่สอดคล้องกับแบบจำลองดังกล่าวนี้ จะเป็นคำเดียวกับคำศัพท์ที่เราต้องการจะรู้จำนั่นเองโดยขั้นตอนการคำนวณหาค่าความน่าจะเป็น จะใช้วิธีวิทเทอร์บี ดังที่ได้กล่าวแล้วในตอนต้น

2.8 ไมโครคอนโทรลเลอร์ 8051

ไมโครคอนโทรลเลอร์ 8051 เป็นไมโครคอมพิวเตอร์ขนาดเล็ก โดยบรรจุไว้ในแผงรวม (Intrgrated Circuit) เหมาะสำหรับควบคุมอุปกรณ์อื่นๆ แบบอัตโนมัติ มีความสะดวกในการใช้งานและสามารถเขียนโปรแกรมควบคุมด้วยภาษาเบสิก หรือภาษาแอสเซมบลี้ได้ ผู้ใช้สามารถเขียนโปรแกรมควบคุมการทำวานได้ตามต้องการ และสามารถเปลี่ยนฟังก์ชั่นการทำงานของวงจรได้สะดวกโดยเปลี่ยนแปลงในส่วนของโปรแกรมเท่านั้น โดยไม่ต้องเปลี่ยนในส่วนของฮาร์ดแวร์ใดๆ เลย ไมโครคอนโทรลเลอร์ตระกูล MCS-51 เป็นอุปกรณ์ที่ออกแบบมาสนองความต้องการของผู้ใช้ คือ มีสายอินพุต เอาท์พุตบัฟเฟอร์อินเตอร์เฟส และสายควบคุมอื่นๆ ที่ใช้สำหรับแยกข้อมูลกับแอดเดรส และยังมีคำสั่งเพิ่มขึ้นเป็นพิเศษเพื่อจัดการข้อมูล นอกจากนั้นยังมีวงจรตั้งเวลาและวงจรนับด้วย MCS-51 มีด้วยกันหลายเบอร์ แต่ละเบอร์มีความสามารถพิเศษต่างกันออกไป ผู้ใช้สามารถดูได้จากคู่มือการใช้ของ MCS-51 และเลือกใช้ได้ตามสะดวก

ชื่อเบอร์	หน่วยความจำภายใน		จำนวนไทเมอร์/เคาน์เตอร์	จำนวนอินเตอร์รัปต์
	เก็บโปรแกรม	เก็บข้อมูล		
8052AH	8K x 8 ROM	256 x 8 RAM	3 x 16-Bit	6
8051AH	4K x 8 ROM	128 x 8 RAM	2 x 16-Bit	5
8051	4K x 8 ROM	128 x 8 RAM	2 x 16-Bit	5
8032AH	ไม่มี	256 x 8 RAM	3 x 16-Bit	6
8031AH	ไม่มี	128 x 8 RAM	2 x 16-Bit	5
8031	ไม่มี	128 x 8 RAM	2 x 16-Bit	5
8751H	4K x 8 EPROM	128 x 8 RAM	2 x 16-Bit	5
8751H-12	4K x 8 EPROM	128 X 8 RAM	2 X 16-Bit	5

ตารางที่ 2.1 แสดงคุณสมบัติของไมโครคอนโทรลเลอร์แต่ละเบอร์ในตระกูล MCS-51

คุณสมบัติโดยทั่วไปของไมโครคอนโทรลเลอร์ตระกูล MCS-51

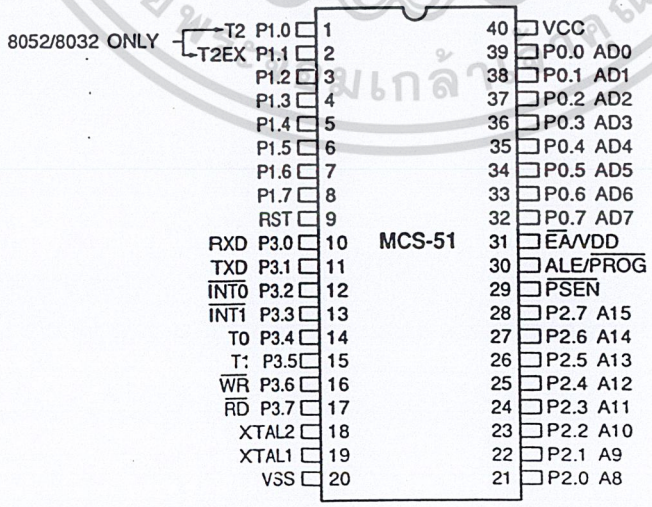
คุณสมบัติโดยทั่วไปที่สำคัญของไมโครคอนโทรลเลอร์ตระกูล MCS-51 มีดังนี้

- เป็นไมโครคอนโทรลเลอร์ขนาด 8 บิต
- มีวงจงออสซิลเลเตอร์ และวงจรผลิตสัญญาณนาฬิกาภายในไอซี
- มีขาสัญญาอินพุต เอาท์พุตจำนวน 32 บิต
- สามารถเชื่อมต่อหน่วยความจำข้อมูลภายนอก (external data memory) โดยอ้างตำแหน่งแอดเดรสได้ถึง 64 กิโลไบต์
- สามารถเชื่อมต่อหน่วยความจำโปรแกรมภายนอก (external program memory) โดยอ้างตำแหน่งแอดเดรสได้ถึง 64 กิโลไบต์

- มีหน่วยความจำโปรแกรมภายในตัว (on-chip program memory) ขนาด 4 กิโลไบต์ โดยเฉพาะเบอร์ 8052 จะมีหน่วยความจำในส่วนนี้ถึง 8 กิโลไบต์ สำหรับเบอร์ 8031 และ M8032 จะไม่มีหน่วยความจำในส่วนนี้
- มีหน่วยความจำข้อมูลภายในตัว (on-chip data memory) ขนาด 128 กิโลไบต์ โดยเฉพาะเบอร์ 8032 และเบอร์ 8052 จะมีหน่วยความจำในส่วนนี้ถึง 256 กิโลไบต์
- หน่วยความจำข้อมูลภายในบางส่วนสามารถเข้าถึงข้อมูลระดับบิตได้ด้วย ทำให้การควบคุมหรือการตรวจสอบสถานะระดับบิตทำได้ง่าย ส่งผลให้การเขียนโปรแกรมทำได้ง่ายมากขึ้น
- มีไทม์เมอร์ / เคาท์เตอร์ (timer/counter) ขนาด 16 บิต จำนวน 2 ตัว โดยเฉพาะเบอร์ 8032 หรือ 8052 จะมีไทม์เมอร์ / เคาท์เตอร์ จำนวน 3 ตัว
- การอินเตอร์รัปต์สามารถทำได้จาก 5 แหล่งกำเนิด โดยเฉพาะเบอร์ 8032 และ 8052 จะทำการอินเตอร์รัปต์ได้จาก 6 แหล่งกำเนิด โดยการอินเตอร์รัปต์ยังสามารถจัดระดับความสำคัญได้เป็น 2 ระดับ
- มีพอร์ตการสื่อสารอนุกรมภายในตัวเอง ซึ่งทำงานแบบฟูลดูเพล็กซ์ (full duplex)
- มีคำสั่งการคำนวณทางคณิตศาสตร์และทางตรรกศาสตร์
- คำสั่งส่วนใหญ่ใช้เวลาการทำงานเพียง 1 ไมโครวินาที เมื่อใช้คริสตอลความถี่ 12 เมกะเฮิรตซ์
- ต้องการแหล่งจ่ายไฟ 5 โวลท์ เพียงชุดเดียว

2.8.1 โครงสร้างของไมโครคอนโทรลเลอร์ 8051

โครงสร้างของไมโครคอนโทรลเลอร์ ตระกูล MCS-51 ทุกเบอร์ จะมีตำแหน่งขาที่เหมือนกัน มีโครงสร้างทางสถาปัตยกรรมภายในและภายนอก ดังรูป 2.19



รูปที่ 2.19 แสดงสถาปัตยกรรมภายนอกและการจัดตำแหน่งขาต่างๆ ของไมโครคอนโทรลเลอร์ตระกูล MCS-51

ตำแหน่งการใช้งานของแต่ละขาของไมโครคอนโทรลเลอร์ตระกูล 8051 มีดังนี้

- ขา Vcc

เป็นขาสำหรับต่อป้อนแรงดันไฟเลี้ยง +5 โวลท์

- ขา Vss

เป็นขาที่สำหรับต่อลงกราวด์

- ขาพอร์ต 0 (Port 0)

มี 8 ขา ได้แก่ ขา P0.0-P0.7 เป็นขาพอร์ตอินพุตเอาท์พุตแบบ 2 ทิศทางสำหรับการใช้งานโดยทั่วไป โดยถ้าใช้งานเป็นอินพุตพอร์ตต้องทำการเซ็ตค่า 1 ไปยังแต่ละบิตของพอร์ต เพื่อกำหนดให้ขาพอร์ตเหล่านั้นอยู่ในสถานะปล่อยลอย ซึ่งในสถานะนี้เองที่สามาถนำมาใช้เป็นพอร์ตอินพุตอิมพีแดนซ์สูงได้ นอกจากพอร์ตนี้จะใช้งานเป็นขาพอร์ตอินพุตเอาท์พุตแล้ว มันยังถูกใช้งานในการติดต่อกับหน่วยความจำภายนอกด้วย โดยทำหน้าที่ในการกำหนดแอดเดรสไบต์ต่ำ (A0-A7) ส่วนตำแหน่งแอดเดรสไบต์สูงจะอยู่ที่พอร์ต 2

- ขาพอร์ต 1 (Port 1)

มี 8 ขา ได้แก่ ขา P1.0-P1.7 เป็นขาพอร์ตอินพุตเอาท์พุตแบบ 2 ทิศทางสำหรับการใช้งานโดยทั่วไป โดยถ้าใช้งานเป็นอินพุตพอร์ตต้องทำการเซ็ตค่า 1 ไปยังแต่ละบิตของพอร์ต เพื่อกำหนดให้เป็นพอร์ตอินพุต นอกจากนี้สำหรับเบอร์ 8032 และ 8052 ขาพอร์ต P1.0 และ P1.1 จะถูกนำมาใช้งานเป็นขา T2 และ T2EX ตามลำดับด้วย

- ขาพอร์ต 2 (Port 2)

มี 8 ขา ได้แก่ ขา P2.0-P2.7 เป็นขาพอร์ตอินพุตเอาท์พุตแบบ 2 ทิศทางสำหรับการใช้งานโดยทั่วไป โดยถ้าใช้งานเป็นอินพุตพอร์ตต้องทำการเซ็ตค่า 1 ไปยังแต่ละบิตของพอร์ต เพื่อกำหนดให้เป็นพอร์ตอินพุต นอกจากพอร์ตนี้จะถูกใช้งานเป็นพอร์ตอินพุตแล้วมันยังถูกใช้งานในการติดต่อกับหน่วยความจำภายนอกด้วย โดยทำหน้าที่ในการกำหนดแอดเดรสไบต์สูง (A8-A15)

- ขาพอร์ต 3 (Port 3)

มี 8 ขา ได้แก่ ขา P3.0-P3.7 เป็นขาพอร์ตอินพุตเอาท์พุตแบบ 2 ทิศทางสำหรับการใช้งานโดยทั่วไป โดยถ้าใช้งานเป็นอินพุตพอร์ตต้องทำการเซ็ตค่า 1 ไปยังแต่ละบิตของพอร์ต เพื่อกำหนดให้เป็นพอร์ตอินพุต นอกจากพอร์ตนี้จะถูกใช้งานเป็นพอร์ตอินพุตแล้วมันยังถูกใช้งานในหน้าที่พิเศษต่างๆ ดังแสดงในตารางที่ 2.2

ขาพอร์ต	หน้าที่พิเศษ
$P_{3.0}$	RXD (serial input port)
$P_{3.1}$	TXD (serial output port)
$P_{3.2}$	$\overline{INT0}$ (external interrupt 0)
$P_{3.3}$	$\overline{INT1}$ (external interrupt 1)
$P_{3.4}$	T0 (Timer 0 external input)
$P_{3.5}$	T1 (Timer 1 external input)
$P_{3.6}$	$\overline{WR}$ (external data memory write strobe)
$P_{3.7}$	$\overline{RD}$ (external data menory read strobe)

ตารางที่ 2.2 แสดงหน้าที่พิเศษของแต่ละขาพอร์ต 3

- ขารีเซต (RST)

ใช้สำหรับการรีเซตการทำงานของไมโครคอนโทรลเลอร์ โดยการรีเซตต้องคงสถานะเป็นหนึ่งอย่างน้อย 2 แมทชีนไซเคิล ในขณะที่ออสซิลเลเตอร์ยังทำงานอยู่

- ขา ALE / PROG

เป็นขาสัญญาณเพื่อทำหน้าที่ควบคุมการแลตช์ (latch) ค่าตำแหน่งแอดเดรสไบท์ต่ำ (Address Latch Enable) เมื่อต้องการติดต่อกับหน่วยความจำภายนอก นอกจากนี้ยังทำหน้าที่เป็นอินพุตรับพัลส์ในการโปรแกรม (program pulse input) ในส่วนของหน่วยความจำ EPROM สำหรับไมโครคอนโทรลเลอร์ในตระกูล MCS-51 ที่มีหน่วยความจำในการโปรแกรมภายในเป็น EPROM

- ขา PSEN (Program Store Enable)

ทำหน้าที่เป็นสัญญาณสโตรบเพื่ออ่านคำสั่งจากหน่วยความจำภายนอก เมื่อไมโครคอนโทรลเลอร์ประมวลผลคำสั่งจากหน่วยความจำภายนอก ขานี้จะส่งสัญญาณสโตรบจำนวน 2 ครั้ง ในแต่ละแมทชีนไซเคิล แต่ในขณะที่ติดต่อกับหน่วยความจำภายนอกจะไม่มีการส่งสัญญาณสโตรบแต่อย่างใด

- ขา EA / VPP (External Access enable/VPP)

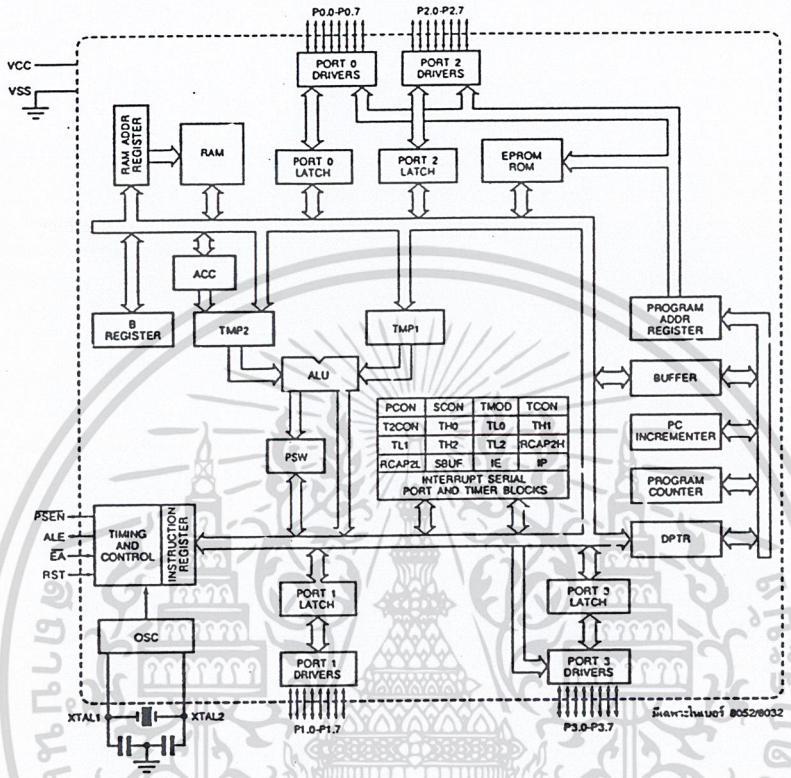
เป็นขาสำหรับเลือกใช้หน่วยความจำโปรแกรมจากภายในหรือภายนอก โดยถ้ามีสถานะเป็น 0 หมายถึง ให้ไมโครคอนโทรลเลอร์รับคำสั่งจากหน่วยความจำภายนอกที่ตำแหน่งแอดเดรส 0-0FFFH (0-1FFFH ถ้าเป็นเบอร์ 8052) อย่างไรก็ตามถ้าบิตป้องกัน (security bit) ในหน่วยความจำ EPROM ถูกโปรแกรมไว้ ไมโครคอนโทรลเลอร์จะไม่รับคำสั่งจากหน่วยความจำภายนอกเลย นอกจากนี้ขานี้ยังทำหน้าที่รับแรงดันไฟสำหรับการโปรแกรม (Vpp) ขนาด 21 โวลท์ เพื่อใช้ในระหว่างโปรแกรม EPROM

- ขา XTAL1 และขา XTAL2

เป็นขาอินพุตและเอาท์พุตของวงจร อินเวอร์ติ้งออสซิลเลเตอร์แอมปลิไฟเออร์ (inverting oscillater amplifier) สำหรับใช้ต่อร่วมกับคริสตอลภายนอก

2.8.2 โครงสร้างภายในของไมโครคอนโทรลเลอร์ 8051

โครงสร้างภายในของไมโครคอนโทรลเลอร์ตระกูล MCS-8051 แสดงในรูปที่ 2.20 โดยส่วนที่มีเครื่องหมาย (*) จะมีในเบอร์ 8032 และ 8052 เท่านั้น



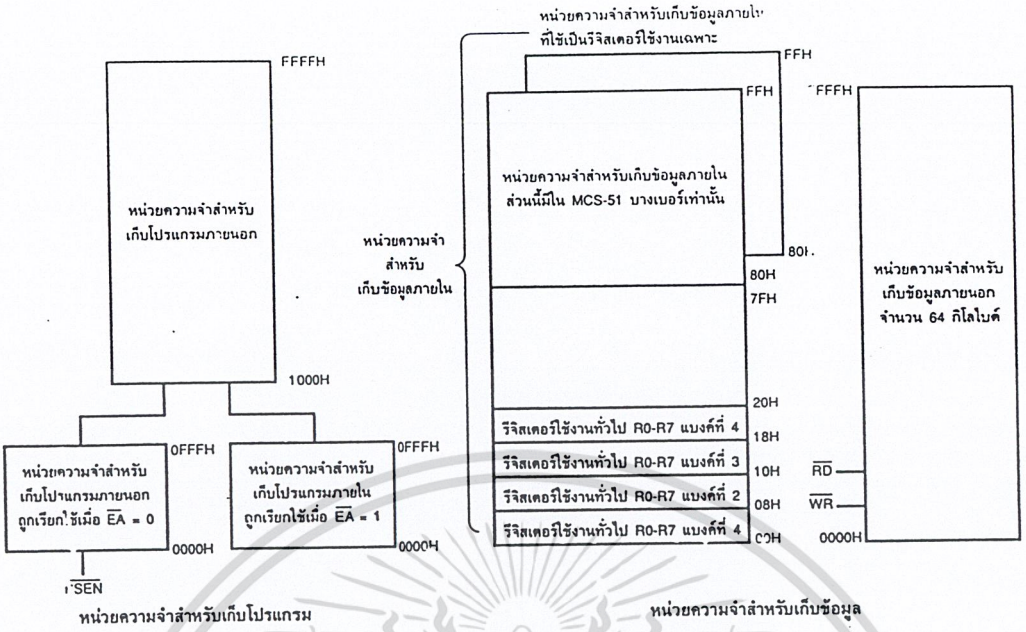
รูปที่ 2.20 แสดงโครงสร้างภายในของไมโครคอนโทรลเลอร์ตระกูล MCS-8051

2.8.3 การจัดหน่วยความจำ

ในไมโครคอนโทรลเลอร์ตระกูล MCS-8051 แบ่งชนิดหน้าที่ของหน่วยความจำออกเป็น 2 ส่วน คือ หน่วยความจำโปรแกรม (program memory) และหน่วยความจำข้อมูล (data memory)

หน่วยความจำโปรแกรมจะใช้สำหรับเก็บโปรแกรมควบคุมการทำงานของไมโครคอนโทรลเลอร์ซึ่งบางเบอร์จะมีหน่วยความจำส่วนนี้อยู่ในตัว โดยอาจจะมีขนาดไม่เท่ากันหรือเป็นหน่วยความจำต่างชนิดกัน เช่น บางเบอร์เป็น ROM และบางเบอร์อาจเป็น EPROM และบางเบอร์อาจไม่มีหน่วยความจำในส่วนนี้เลย โปรแกรมการทำงานจะถูกเก็บไว้ยังหน่วยความจำโปรแกรมภายนอกทั้งหมด

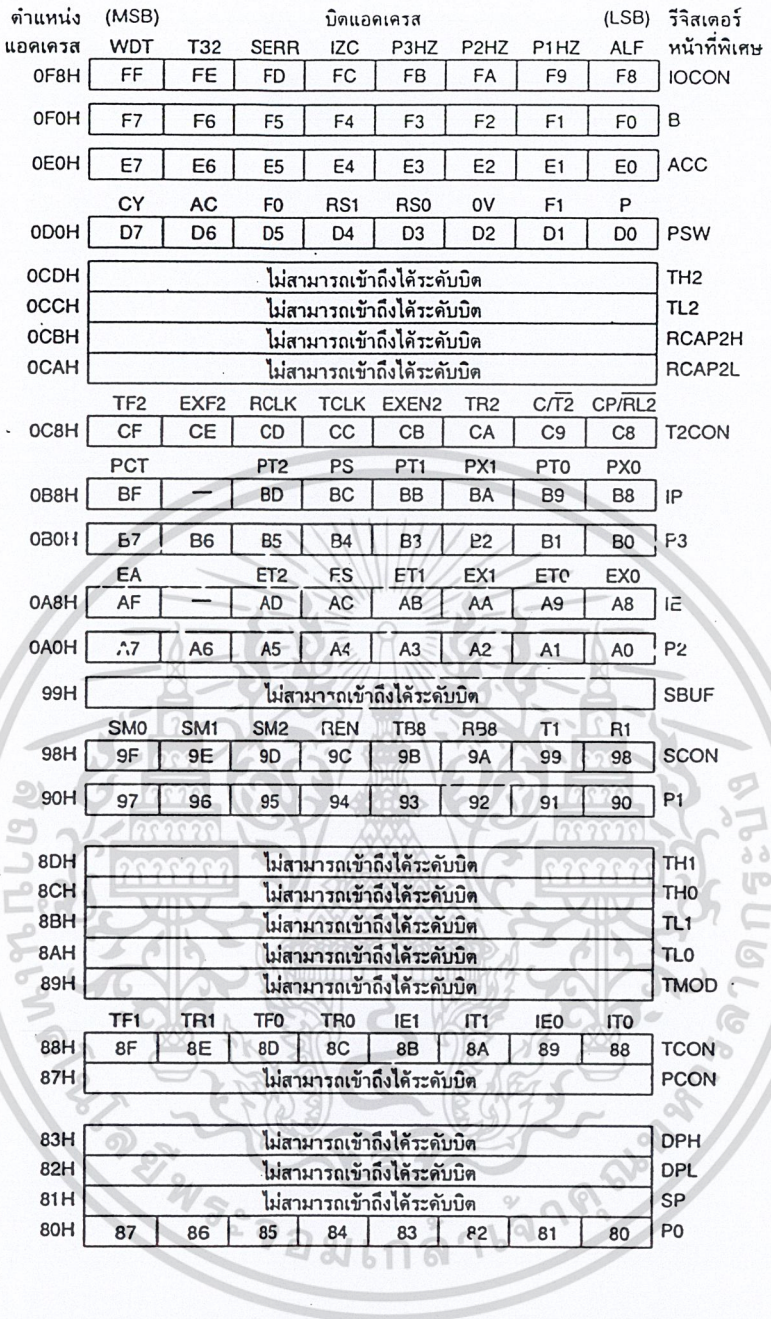
สำหรับหน่วยความจำข้อมูลจะใช้สำหรับเก็บข้อมูลหรือค่าตัวแปรต่างๆ จากการทำงานของโปรแกรม ซึ่งใน MCS-51 ทุกเบอร์จะมีหน่วยความจำในส่วนนี้อยู่จำนวนหนึ่ง แต่อาจมีขนาดมากน้อยต่างกันไปในแต่ละเบอร์ สำหรับการจัดโครงสร้างของหน่วยความจำทั้งในส่วนของหน่วยความจำโปรแกรมและหน่วยความจำข้อมูลแสดงดังรูป 2.21



รูปที่ 2.21 แสดงการจัดโครงสร้างของหน่วยความจำ
ทั้งในส่วนของหน่วยความจำโปรแกรมและหน่วยความจำข้อมูล

2.8.3 หน่วยความจำโปรแกรม

หน่วยความจำโปรแกรมสามารถแบ่งออกได้เป็น 2 ส่วน คือ หน่วยความจำโปรแกรมภายในและหน่วยความจำโปรแกรมภายนอก หน่วยความจำโปรแกรมภายในจะถูกเลือกใช้งานถ้าขาสัญญาณ EA มีค่าเป็น 1 โดยจะถูกใช้งานในช่วงแอดเดรส 0-0FFFH (หรือช่วงแอดเดรส 0-1FFFH ในเบอร์ 8052) นอกเหนือจากช่วงแอดเดรสนี้จะใช้หน่วยความจำโปรแกรมภายนอกทั้งหมด ในกรณีตรงข้ามถ้าขาสัญญาณ EA มีค่าเป็น 0 ในช่วงแอดเดรส 0-0FFFH (หรือช่วงแอดเดรส 0-1FFFH ในเบอร์ 8052) จะถูกใช้จากหน่วยความจำภายนอก หรือกล่าวได้ว่าถ้าขาสัญญาณ EA มีค่าเป็น 0 จะเป็นการเลือกใช้หน่วยความจำโปรแกรมภายนอกทั้งหมดตลอดช่วงแอดเดรส



รูปที่ 2.22 แสดงการจัดหน่วยความจำและตำแหน่งของรีจิสเตอร์หน้าที่พิเศษต่างๆ

2.8.4 หน่วยความจำข้อมูล

หน่วยความจำข้อมูลสามารถแบ่งออกได้เป็น 2 ส่วน คือ หน่วยความจำข้อมูลภายในและหน่วยความจำข้อมูลภายนอก หน่วยความจำโปรแกรมภายในยังถูกแบ่งออกได้เป็น 2 ส่วนย่อย คือ ส่วนที่ใช้เก็บข้อมูลทั่วไปและส่วนที่ใช้เป็นรีจิสเตอร์หน้าที่พิเศษหรือ SFR (Special Function Register) โดยส่วนที่ใช้เก็บข้อมูลทั่วไปจะถูกใช้สำหรับเก็บข้อมูลหรือค่าตัวแปรต่างๆ จากการทำงานของโปรแกรม ส่วนรีจิสเตอร์หน้าที่พิเศษจะถูกใช้งานเป็นรีจิสเตอร์ควบคุมการทำงาน และบอกสถานะการทำงานของไมโครคอนโทรลเลอร์

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 ทุกเบอร์จะมีหน่วยความจำข้อมูลขนาด 128 ไบต์เป็นอย่างน้อย และบางเบอร์อาจมีถึง 256 ไบต์

2.8.5 รีจิสเตอร์หน้าที่พิเศษ (SFR)

รีจิสเตอร์หน้าที่พิเศษมีบทบาทอย่างมากในการควบคุมดูแลการทำงานของไมโครคอนโทรลเลอร์ และทำให้การเขียนโปรแกรมสามารถทำได้สะดวกมากขึ้น รีจิสเตอร์หน้าที่พิเศษทำหน้าที่สำคัญคือการควบคุมการทำงานในส่วนต่างๆ ภายในไมโครคอนโทรลเลอร์ และทำหน้าที่แสดงสถานะการทำงาน ซึ่งในรีจิสเตอร์หน้าที่พิเศษบางตัวยังสามารถเข้าถึงได้ในระดับบิต (bit addressable) ด้วย ดังแสดงในรูปการจัดหน่วยความจำและตำแหน่งของรีจิสเตอร์หส้าที่พิเศษต่างๆ ในรูปที่ 2.22

2.8.6 รีจิสเตอร์ใช้งานทั่วไป

รีจิสเตอร์ใช้งานทั่วไปมีไว้สำหรับผู้ใช้งานโปรแกรมสามารถนำข้อมูลไปพักไว้ชั่วคราวหรือใช้งานทั่วไปได้ตามต้องการ ซึ่งรีจิสเตอร์ใช้งานทั่วไปนี้มีอยู่ด้วยกัน 8 ตัว ถูกจัดให้อยู่ด้วยกัน 8 ตัว ถูกจัดให้อยู่รวมกันและมีให้เลือกถึง 4 แบงค์ (bank) นั่นคือมีรีจิสเตอร์ใช้งานทั่วไปถึง 32 ตัว ให้ใช้งาน เพียงแต่การเลือกใช้รีจิสเตอร์ R0-R7 ในแบงค์ใดแบงค์หนึ่งจะถูกกำหนดจากบิต RS0,RS1 ในรีจิสเตอร์หน้าที่พิเศษ PSW ดังนั้นการเลือกใช้จึงเลือกได้เพียงแบงค์เดียวในขณะใดขณะหนึ่ง อย่างไรก็ตามค่าข้อมูลที่เก็บไว้ในรีจิสเตอร์แบงค์ใดก็ตามที่มีชื่อเดียงกันแต่อยู่คนละแบงค์จะไม่มีผลซึ่งกันและกันเลย ทำให้ผู้เขียนโปรแกรมใช้งานรีจิสเตอร์ทั่วไปนี้ได้ทั้ง 32 ตัวอย่างเต็มที่และไม่ยุ่งยากในการเขียนโปรแกรม

2.9 พอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 มีวงจรสื่อสารอนุกรมแบบฟูลดูเพล็กซ์ 1 ชุด (วงจรสื่อสารแบบฟูลดูเพล็กซ์ หมายถึง วงจรสื่อสารแบบที่สามารถทำการรับและส่งข้อมูลในลักษณะ 2 ทิศทางได้ในเวลาเดียวกัน) โดยใช้ขาสัญญาณของพอร์ต 3 คือ ขา P3.0 เป็นขารับข้อมูลหรือ RxD และขา P3.1 เป็นขาส่งข้อมูลออกหรือ TxD โดยวงจรสื่อสารข้อมูลแบบอนุกรมของไมโครคอนโทรลเลอร์ตระกูล MCS-51 แบบแฟลชเป็นแบบอะซิงโครนัส ปกติแล้วพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ตระกูล MCS-51 จะใช้ในการสื่อสารกับพอร์ตอนุกรมของคอมพิวเตอร์ โดยใช้มาตรฐาน RS-232 แต่ในปัจจุบันสามารถติดต่อกันในมาตรฐาน RS-422 หรือ RS-485 ได้แล้ว โดยใช้ไอซีพิเศษที่ทำหน้าที่ในการแปลงสัญญาณการสื่อสารดังกล่าว

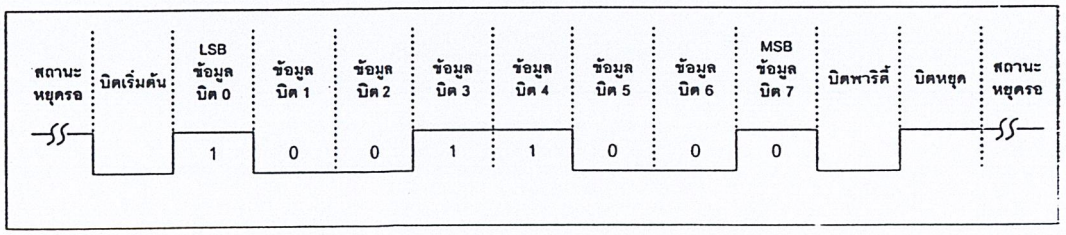
2.9.1 การสื่อสารข้อมูลแบบอะซิงโครนัส

การสื่อสารข้อมูลแบบอะซิงโครนัส คือ การสื่อสารข้อมูลโดยไม่จำเป็นต้องมีสัญญาณนาฬิการ่วมด้วย แต่จะใช้ในการกำหนดค่าอัตราเร็วในการรับและส่งข้อมูลให้มีค่าเท่ากัน ซึ่งเรียกอัตราเร็วนี้ว่า อัตราบอด หรือ บอดเรต (baud rate) มีหน่วยเป็น บิตต่อวินาที (bit per second : bps)

รูปแบบของข้อมูลที่ใช้ในการรับส่งแบบอะซิงโครนัสประกอบด้วย 4 ส่วนด้วยกัน คือ

1. บิตเริ่มต้น (start bit) มีขนาด 1 บิต
2. บิตข้อมูลอนุกรมมีขนาด 8 บิต
3. บิตตรวจสอบพาริตี้ (parity bit) มีขาด 1 บิต หรือ ไม่มี
4. บิตปิดท้ายหรือบิตหยุด (stop bit) มีขนาด 1 บิต

รูปที่ 2.23 แสดงรูปแบบของข้อมูลอนุกรมแบบอะซิงโครนัส เมื่อไม่มีการส่งข้อมูล ขา DATA จะมีสถานะลอจิก "1" เรียกสถานะนี้ว่า สถานะหยุดรอ (waiting stage) การเริ่มต้นส่งข้อมูลจะเริ่มจากการให้ขา DATA มีลอจิก "0" ด้วยช่วงระยะเวลา 1 บิต เรียกบิตนี้ว่า บิตเริ่มต้น (start bit) จากนั้นบิตข้อมูลจะถูกส่งออกไป โดยเริ่มจากบิตที่มีนัยสำคัญต่ำสุดหรือบิต LSB ก่อน ซึ่งข้อมูลที่ต้องการส่งมีจำนวน 8 บิต จากนั้นตามด้วย บิตพาริตี้ (parity bit) ซึ่งใช้ในการตรวจสอบความผิดพลาดที่เกิดขึ้นในการส่งข้อมูล บิตสุดท้ายที่ส่งคือ บิตปิดท้ายหรือบิตหยุด (stop bit) โดยจะเป็นการทำให้ขา DATA มีสถานะลอจิก "1" อีกครั้งด้วยระยะเวลาอย่างน้อย 1 บิต 1.5 บิต หรือ 2 บิต เพื่อเป็นการแสดงว่าสิ้นสุดข้อมูลแล้ว



รูปที่ 2.23 รูปแบบข้อมูลอนุกรมแบบซิงโครนัส

อัตราความเร็วในการรับแและส่งข้อมูลของการรับส่งข้อมูลแบบอะซิงโครนัสหรืออัตราบอดหรือบอดเรตที่ใช้สำหรับพอร์ตอนุกรม RS-232 มีด้วยกันหลายค่า ตั้งแต่ 110 ถึง 19200 บิตต่อวินาที เนื่องจากอัตราบอดเรต คือ อัตราที่สามารถส่งข้อมูลได้ใน 1 วินาที สมมติว่าข้อมูลอนุกรมมีขนาด 8 บิต ไม่มีการตรวจสอบพาริตี้ มีบิตเริ่มต้น 1 บิต และบิตปิดท้าย 1 บิต ความยาวของข้อมูล 1 ไบต์ จะมีความยาวเท่ากับ 10 บิต ถ้าใช้บอดเรตในการส่งข้อมูลเท่ากับ 9600 บิตต่อวินาที ก็จะสามารถรับส่งข้อมูลได้ ด้วยความเร็ว 960 ไบต์ต่อวินาที

การตรวจสอบพาริตี้สามารถกำหนดให้เป็นแบบคี่ (odd) แบบคู่ (even) หรือไม่มีการตรวจสอบพาริตี้ก็ได้ พาริตี้คี่หรือพาริตี้คู่แสดงถึงจำนวนลอจิก "1" ทั้งหมดภายในข้อมูลที่ส่งไป 1 ไบต์รวมบิตพาริตี้ว่ามีจำนวนเป็นเลขคู่หรือเลขคี่ ตัวอย่างเช่น ข้อมูลที่จะทำการส่งมีขนาด 8 บิต มีค่าเท่ากับ 99H หรือ 10011001B จะเห็นว่าข้อมูลในไบต์นี้มีจำนวนลอจิก "1" จำนวน 4 ตัวซึ่งเป็นเลขคู่ ดังนั้นถ้ากำหนดค่าพาริตี้เป็นคู่ ค่าของบิตพาริตี้จะต้องมีค่าลอจิกเป็น "0" แต่ถ้าพาริตี้เป็นคี่ ค่าของพาริตี้เป็นคี่จะต้องเป็น "1" เพื่อให้ข้อมูล 1 ไบต์รวมทั้งบิตพาริตี้เป็นคี่

บิตพาริตี้ถูกสร้างขึ้นจากภาคส่งข้อมูลของ UART (Universal Asynchronous Receiver Transmitter : เป็นอุปกรณ์ที่ใช้ในการรับและส่งข้อมูลอนุกรม) ซึ่งทางภาครับต้องกำหนดการตรวจสอบพาริตี้ที่ตรงกันเอาไว้ว่าจะตรวจสอบพาริตี้คี่หรือคู่ จากนั้นภาครับของ UART จะทำการตรวจสอบพาริตี้ที่เกิดขึ้นว่าเป็นคู่หรือเป็นคี่ โดยการนับจำนวนลอจิก "1" ทั้งหมดรวมทั้งบิตพาริตี้ด้วย ถ้ากำหนดพาริตี้ไว้เป็นคู่แต่อ่านค่าตัวเลขในการสับออกมาได้เป็นตัวเลขคี่ ทางภาครับจะแสดงข้อผิดพลาดออกมาให้ผู้ใช้ทราบ กระบวนการดังกล่าวนี้เป็นวิธีการตรวจสอบความผิดพลาดที่เกิดขึ้นในการรับส่งข้อมูลที่ง่ายที่สุด แต่มันสามารถตรวจสอบได้เมื่อมีบิตข้อมูลที่ทำการรับส่งผิดพลาดเพียงบิตเดียวเท่านั้น ถ้าข้อมูลที่ทำการรับส่งผิดพลาดมากกว่า 1 บิต การตรวจสอบด้วยวิธีนี้จะไม่ได้ผล สำหรับการตั้งค่าพาริตี้เป็น NONE นั้น ทางภาครับและภาคส่งจะไม่มีการตรวจสอบพาริตี้

2.9.2 รีจิสเตอร์ที่เกี่ยวข้องกับการทำงานของพอร์ตอนุกรมใน MCS-51

ในการทำงานของพอร์ตอนุกรมในไมโครคอนโทรลเลอร์ MCS-51 มีรีจิสเตอร์ที่เกี่ยวข้องอยู่ 2 ตัว ดังนี้

รีจิสเตอร์บัฟเฟอร์ของพอร์ตอนุกรมหรือ SBUF (Serial data buffer register)

มีแอดเดรสอยู่ที่ 99H ในรีจิสเตอร์ฟังก์ชั่นพิเศษหรือ SFR มีขนาด 8 บิต แบ่งเป็น 2 ส่วน คือ รีจิสเตอร์บัฟเฟอร์สำหรับส่งข้อมูล (transmit buffer register) และรับข้อมูล (receive buffer register) เมื่อมีการเขียนข้อมูลเข้ามาในรีจิสเตอร์ SBUF ข้อมูลนั้นจะถูกส่งต่อไปยังบัฟเฟอร์สำหรับส่งข้อมูล เพื่อส่งออกจากไมโครคอนโทรลเลอร์ผ่านทางขา TxD หรือขา P3.1 ในกรณีที่มีการอ่านข้อมูลจากรีจิสเตอร์ SBUF ข้อมูลจะถูกส่งผ่านไปยังบัฟเฟอร์สำหรับรับข้อมูลเพื่อส่งต่อไปยังไมโครคอนโทรลเลอร์ต่อไป สำหรับการรับข้อมูลจากภายนอกนั้นผ่านทางขา RxD หรือ P3.0 ของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

รีจิสเตอร์ควบคุมการทำงานของพอร์ตอนุกรมหรือ SCON (Serial control register)

มีแอดเดรสอยู่ที่ 98H ในรีจิสเตอร์ฟังก์ชั่นพิเศษหรือ SFR มีขนาด 8 บิต สามารถเข้าถึงได้ในระดับบิต มีรายละเอียดการทำงานดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
SM0	SM1	SM2	REN	TB8	RB8	TI	RI

รูปที่ 2.24 แสดงรายละเอียดภายในของรีจิสเตอร์ควบคุมการทำงานของพอร์ตอนุกรม SCON

SM0-SM1 (Serial port mode bit 0-1) : ใช้ในการเลือกโหมดการทำงานของพอร์ตอนุกรมภายในไมโครคอนโทรลเลอร์ MCS-51 ดังมีรายละเอียดดังต่อไปนี้

SM2 : ใช้ในการเอ็นเอเบิลการสื่อสารในแบบมัลติโปรเซสเซอร์ (multiprocessor) ในการทำงานของโหมด 2 และ 3 ของพอร์ตอนุกรมในไมโครคอนโทรลเลอร์ MCS-51 ถ้าบิตนี้เป็น "1" บิต RI จะไม่แอกตีฟ ถ้าบิตที่ 9 ที่รับเข้ามาเป็น "0" (ข้อมูลบิตที่ 9 เก็บไว้ที่บิต RB8) ในการทำงานโหมด 1 ถ้าบิตนี้เซตบิต RI จะไม่แอกตีฟถ้ายังไม่ได้รับบิตหยุด ในส่วนโหมด 0 บิตนี้ไม่มีการใช้งาน

REN (Enable serial reception) : ใช้ในการเอ็นเอเบิลการรับข้อมูลของพอร์ตอนุกรม ทำการเซตและเคลียร์ด้วยกระบวนการทางซอฟต์แวร์ ถ้าต้องการให้มีการรับข้อมูลต้องเซตให้บิตนี้เป็น "1"

TB8 : ใช้สำหรับเก็บข้อมูลบิตที่ 9 ที่ต้องการส่งออกไปในการทำงานโหมด 2 และ 3 ของพอร์ตอนุกรมในไมโครคอนโทรลเลอร์ MCS-51 เซตและเคลียร์ด้วยกระบวนการทางซอฟต์แวร์

RB8 : ใช้สำหรับรับข้อมูลบิตที่ 9 ที่เข้ามาในการทำงานของโหมด 2 และ 3 ของพอร์ตอนุกรมในไมโครคอนโทรลเลอร์ MCS-51 แต่ถ้าหากพอร์ตอนุกรมทำงานอยู่ในโหมด 1 และบิต SM2 เป็น "0" ข้อมูลที่บิต RB8 คือข้อมูลของบิตหยุด (STOP bit) สำหรับในการทำงานโหมด 0 บิตนี้จะไม่ใช้งาน บิต RB8 นี้สามารถเซตและเคลียร์ด้วยกระบวนการทางซอฟต์แวร์

TI (Transmit Interrupt flag) : ใช้ในการแสดงการเกิดอินเตอร์รัปต์เมื่อมีการส่งข้อมูลออกจากพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51 สามารถเซตได้ด้วยกระบวนการทางฮาร์ดแวร์ เมื่อทำการส่งข้อมูลบิตที่ 8 ไปเรียบร้อยแล้วในการทำงานโหมด 0 ส่วนในการทำงานโหมดอื่น บิตนี้จะเซตเมื่อมีการเริ่มต้นส่งบิตหยุดออกไป การเคลียร์บิตนี้ต้องใช้กระบวนการทางซอฟต์แวร์เท่านั้น

RI (Receive Interrupt flag) : ใช้แสดงการเกิดอินเตอร์รัปต์เมื่อมีการรับข้อมูลเข้าสู่พอร์ตอนุกรม สามารถเซตได้ด้วยกระบวนการทางฮาร์ดแวร์ เมื่อทำการรับข้อมูลบิตที่ 8 เรียบร้อยแล้วในการทำงานโหมด 0 ส่วนในการทำงานโหมดอื่น บิตนี้จะเซตเมื่อสามารถรับบิตหยุดของข้อมูลอนุกรมไปได้ครึ่งทางแล้ว ยกเว้นในกรณีที่บิต SM2 มีการเซต บิตนี้จะเซตได้ก็ต่อเมื่อการรับบิตหยุดหรือบิตที่ 9เกิดขึ้นอย่างสมบูรณ์แล้ว การเคลียร์บิตนี้ต้องใช้กระบวนการทางซอฟต์แวร์เท่านั้น

2.9.3 โหมดการทำงานของพอร์ตอนุกรมใน MCS-51

พอร์ตอนุกรมในไมโครคอนโทรลเลอร์ MCS-51 สามารถเลือกการทำงานได้ถึง 4 โหมด คือ

1. โหมด 0 เป็นการกำหนดให้พอร์ตอนุกรมทำงานในลักษณะ ชิฟต์รีจิสเตอร์
2. โหมด 1 เป็นการกำหนดให้เป็น UART ขนาด 8 บิต สามารถเลือกอัตราบอดได้
3. โหมด 2 เป็นการกำหนดให้เป็น UART ขนาด 9 บิต โดยมีอัตราอัตราบอดคงที่
4. โหมด 3 เป็นการกำหนดให้เป็น UART ขนาด 9 บิต สามารถเลือกอัตราบอดได้

การเลือกโหมดทำได้โดยการกำหนดข้อมูลให้แก่บิต SM0 และ SM1 ในรีจิสเตอร์ SCON

การทำงานในโหมด 0 ของวงจรพอร์ตอนุกรม

มีไดอะแกรมการทำงานและไดอะแกรมเวลาแสดงในรูปที่ 2.25 ข้อมูลอนุกรมจะผ่านเข้าและออกทางขา RxD ส่วนขา TxD ทำหน้าที่เป็นสัญญาณนาฬิกาของการเลื่อนข้อมูล (shift clock) ในโหมดนี้มีข้อมูล 8 บิต โดยทำการรับและส่งข้อมูลในบิต LSB ก่อน อัตราในการรับส่งข้อมูลหรืออัตราบอดถูกกำหนดไว้คงที่ที่ 1/12 ของความถี่สัญญาณนาฬิกา

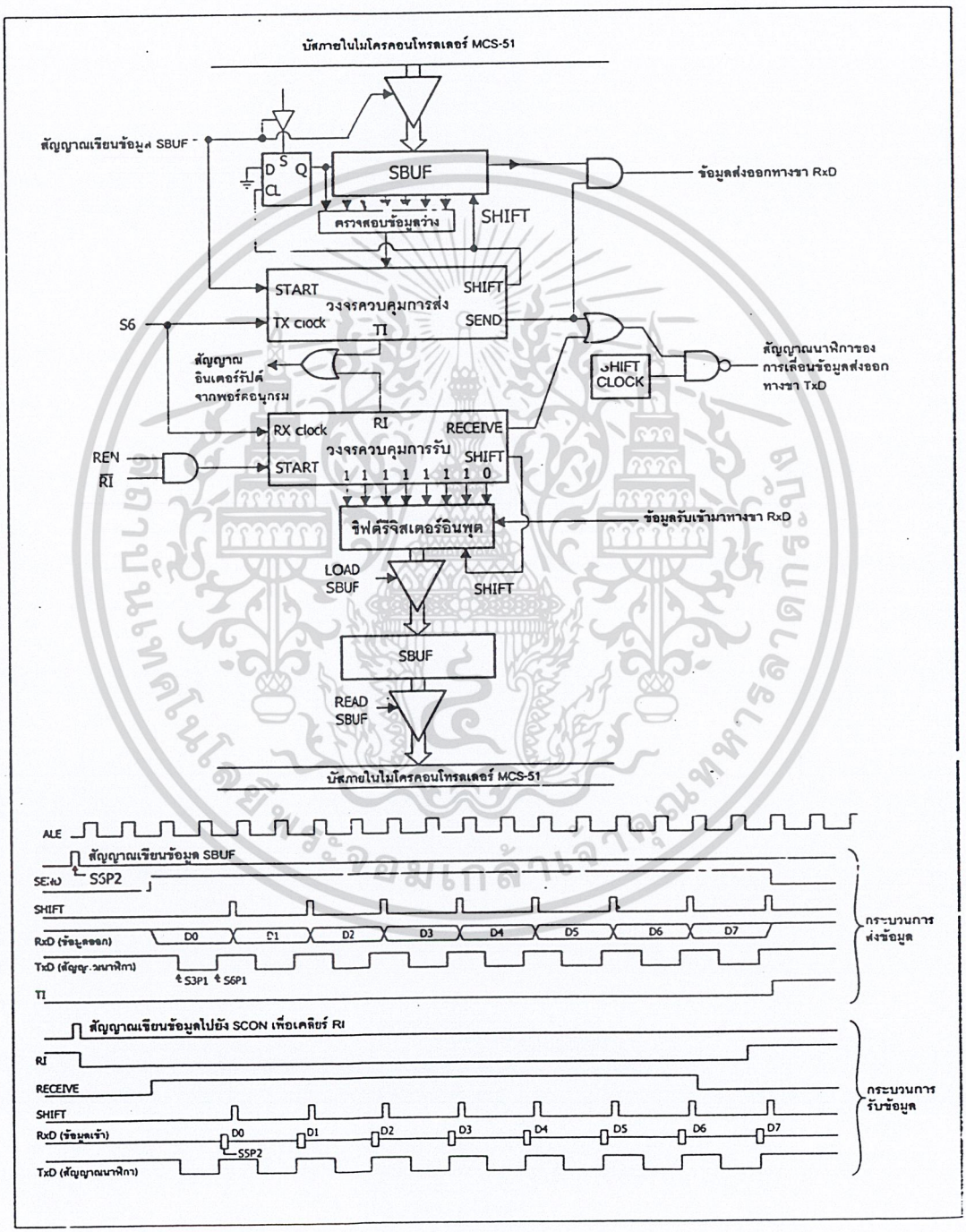
เริ่มต้นการส่งข้อมูลด้วยการเขียนข้อมูลที่ต้องการส่งมายังรีจิสเตอร์ SBUF สัญญาณเขียนข้อมูล SBUF แอคทีฟเป็น "1" ที่สเตต 6 เฟส 2 (S6P2) ของแมชีนไซเคิล ส่งมายังวงจรควบคุมการส่ง (Tx control) ทำให้วงจรควบคุมเริ่มส่งข้อมูล สัญญาณ SEND จะแอคทีฟเป็น "1" ตลอดการส่งข้อมูล

ข้อมูลจากรีจิสเตอร์ SBUF จะถูกเลื่อนออกที่ขา P3.0 หรือขา RxD ครั้งละบิต ตามจังหวะของสัญญาณนาฬิกาที่ส่งออกมาทางขา P3.1 หรือ TxD โดยสัญญาณนาฬิกาของการเลื่อนข้อมูลจะมีขอบเขตของสัญญาณขาลงที่สเตต 3 เฟส และมีขอบเขตขาขึ้นของสัญญาณที่สเตต 6 เฟส 1 ของแต่ละแมชีนไซเคิลในกระบวนการส่งข้อมูล จนกระทั่งเมื่อส่งข้อมูลมาครบ 8 บิตแล้ว บิต TI ในรีจิสเตอร์ SCON จะเกิดการเซต เป็นการแจ้งให้ทราบว่าส่งข้อมูลครบแล้ว หากการอินเตอร์รัปต์จากพอร์ตอนุกรมได้รับการเอ็นเอเบิลไว้ ก็จะเกิดการอินเตอร์รัปต์ขึ้นในระบบ เมื่อเสร็จสิ้นกระบวนการรับข้อมูล สัญญาณ SEND จะกลายเป็น "0" จนกว่าจะเริ่มกระบวนการรับข้อมูลใหม่

ในกระบวนการรับข้อมูล เริ่มต้นด้วยการเซต REN ให้เป็น "1" และเคลียร์บิต RI ในรีจิสเตอร์ SCON ก่อนที่สเตต 6 เฟส 2 ของแมชีนไซเกิลถัดไป วงจรควบคุมการรับ (RX control) จะทำการเขียนข้อมูล 11111110 ไปยังชิฟรีจิสเตอร์สำหรับรับข้อมูลและทำการแอกตีฟสัญญาณ RECEIVE ให้เป็น "1" ในสัญญาณนาฬิกาถูกถัดไป

เมื่อสัญญาณ RECEIVE แอกตีฟ ก็จะเกิดการส่งสัญญาณนาฬิกาของการเลื่อนข้อมูล (Shift clock) ขึ้นผ่านทางขา P3.1 หรือ TxD เพื่อทำการกำหนดจังหวะการรับข้อมูลครั้งละบิต โดยสัญญาณนาฬิกานี้จะเกิดขึ้นในช่วงสเตต 3 เฟส 1 ถึงสเตต 6 เฟส 1 ของแต่ละแมชีนไซเคิล การรับข้อมูลเข้ามาทางขา P3.0 หรือ RxD จะเกิดขึ้นที่สเตต 5 เฟส 2 ในแมชีนไซเกิลเกี่ยวกับสัญญาณนาฬิกาของการเลื่อนข้อมูล จนกระทั่งรับข้อมูลครบทั้ง 8 บิต บิต RI จะได้รับการเซตเพื่อแจ้งการเสร็จสิ้นกระบวนการรับข้อมูล หากการอินเตอร์รัปต์จากพอร์ตอนุกรมได้รับการเอ็นเอเบิลไว้ก็จะเกิดการอินเตอร์รัปต์ในระบบ เมื่อเสร็จสิ้นกระบวนการรับข้อมูล สัญญาณ RECEIVE จะกลายเป็น "0" จนกว่าจะเริ่มต้นกระบวนการรับข้อมูลใหม่

การทำงานในโหมดนี้ของพอร์ตอนุกรมในไมโครคอนโทรลเลอร์ MCS-51 จะใช้ในการเชื่อมต่อไปกับไอซีรีจิสเตอร์ภายนอกเพื่อทำการขยายจำนวนพอร์ตอินพุตหรือเอาต์พุต แต่ไม่เป็นที่นิยมใช้งานมากนัก เนื่องจากในไมโครคอนโทรลเลอร์ MCS-51 เองมีพอร์ตอยู่ค่อนข้างมาก และติดต่อกับพอร์ตนั้นได้ง่ายและเร็วกว่ามาก



รูปที่ 2.25 ไดอะแกรมการทำงานในโหมด 0 ของพอร์ตอนุกรมภายในไมโครคอนโทรลเลอร์ MCS-51

การทำงานในโหมด 1 ของวงจรในพอร์ตอนุกรม

มีไดอะแกรมแสดงในรูปที่ 2.26 ในโหมดนี้ใช้รับส่งข้อมูลรวม 10 บิต โดยส่งข้อมูลออกทางขา P3.1 หรือ TxD และรับข้อมูลเข้าทางขา P3.0 หรือ RxD ข้อมูลทั้ง 10 บิตประกอบด้วย บิตเริ่มต้น (มีขาเป็น "0") 1 บิต บิตข้อมูล 8 บิต โดยรับหรือส่งข้อมูลในบิต LSB ก่อน และบิตหยุดหรือบิตปิดท้าย (มีค่าเป็น "1") ในการรับข้อมูลบิตหยุดจะถูกเก็บไว้ในบิต RB8 ในรีจิสเตอร์ SCON อัตราบอดในโหมดนี้ได้รับการกำหนดโดยอัตราการเกิดโอเวอร์โฟลวของไทเมอร์ 1 ใน AT89C51 ส่วนในไมโครคอนโทรลเลอร์เบอร์ AT89C52 และในอนุกรม AT89Sxx สามารถเลือกอัตราการเกิดโอเวอร์โฟลวของไทเมอร์ 1 หรือไทเมอร์ 2 ในการกำหนดอัตราบอดได้

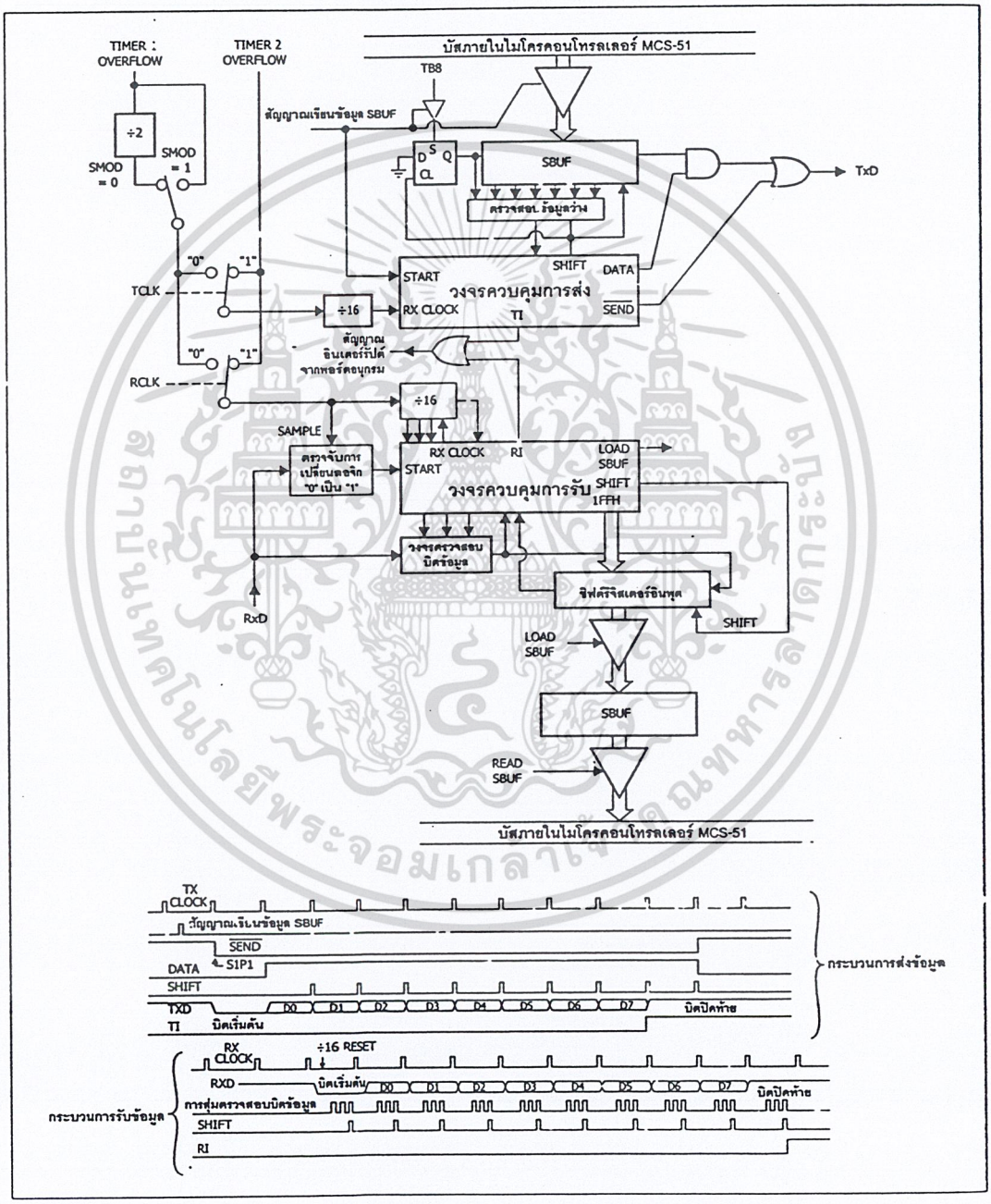
กระบวนการส่งข้อมูลเริ่มต้นด้วยการแอกตีฟสัญญาณเขียนข้อมูลมายังรีจิสเตอร์ SBUF ส่งมายังวงจรควบคุมการส่ง (TX control) จากนั้นวงจรควบคุมจะทำการแอกตีฟสัญญาณ SEND ที่สเตต 1 เฟส 1 ของแมชีนไซเกิลต่อมา โดยสัญญาณ SEND จะเป็น "0" ตลอดการส่งข้อมูล เมื่อสัญญาณ SEND แอกตีฟจะทำการส่งบิตเริ่มต้นก่อนเป็นบิตแรก โดยมีคาบเวลาของบิตเริ่มต้นเท่ากับ 1 แมชีนไซเกิล จากนั้นตามด้วยการส่งบิตข้อมูล 8 บิต เรียงลำดับจากบิต LSB โดยข้อมูลที่ทำการส่งถูกเรียกออกมาจากรีจิสเตอร์บัฟเฟอร์สำหรับการส่งข้อมูลในทุก ๆ บิตข้อมูลที่ทำการส่งข้อมูลออกไปจะเกิดสัญญาณพัลส์ SHIFT ขึ้น เพื่อให้เรียกข้อมูลในแต่ละบิตจากรีจิสเตอร์บัฟเฟอร์ การกำหนดจังหวะการส่งข้อมูลใช้สัญญาณนาฬิกาการส่ง (TX clock) เป็นตัวกำหนด โดยสัญญาณนาฬิกานี้ได้มาจากการหารสัญญาณ TCLK จากไทเมอร์ 1 ด้วย 16 หลังจากการส่งบิตข้อมูลก็จะทำการส่งบิตหยุดหรือบิตปิดท้าย 1 บิต ดังนั้นการส่งข้อมูลจะใช้สัญญาณนาฬิกาทั้งหมด 10 ลูก เมื่อทำการส่งข้อมูลครบเรียบร้อยแล้ว จะทำการเซตบิต TI ในรีจิสเตอร์ SCON หากการอินเตอร์รัปต์จากพอร์ตอนุกรมได้รับการเอ็นเอเบิลไว้ก็จะเกิดการอินเตอร์รัปต์ขึ้นในระบบ หลังจากที่ทำการบริการอินเตอร์รัปต์หรือส่งข้อมูลเรียบร้อยแล้ว ต้องทำการเคลียร์บิต TI ก่อนเป็นอันดับแรก เพื่อให้การรับข้อมูลทางพอร์ตอนุกรมดำเนินต่อไปได้

ด้านการรับข้อมูลจะทำการตรวจจับการเปลี่ยนแปลงระดับลอจิกจาก "1" เป็น "0" ที่ขา RxD โดยใช้อัตราการสุ่มเท่ากับ 1/16 เท่าของอัตราบอด เมื่อตรวจจับพบ ไทเมอร์/เคาน์เตอร์ที่ใช้ในการกำหนดอัตราบอดจะรีเซตและทำการเขียนข้อมูล 1FFH ไปยังชิฟต์รีจิสเตอร์ ข้อมูลจะเริ่มเดินทางเข้าสู่พอร์ตอนุกรมของไมโครคอนโทรลเลอร์ผ่านทางขา RxD ในการตีความว่าบิตที่เข้ามาเป็น "0" หรือ "1" จะใช้ผลการสุ่มข้างมาก โดยบิตของข้อมูลที่เข้ามาได้รับการแบ่งออกเป็น 16 สเตต การสุ่มข้อมูลจะทำการสุ่มสเตตที่ 7, 8 และ 9 หาก 2 ใน 3 ของการสุ่มพบว่าข้อมูลเป็นลอจิกใด จะตีความข้อมูลในบิตนั้นเป็นเสียงข้างมาก ยกตัวอย่าง สุ่มพบลอจิก "1" 2 ใน 3 ครั้ง จะตีความว่าบิตของข้อมูลนั้นที่รับได้นั้นเป็น "1"

ลำดับของการรับข้อมูลมีลักษณะเดียวกับการส่งข้อมูลคือ เริ่มด้วยบิตเริ่มต้นก่อน ตามด้วยบิตข้อมูล และบิตปิดท้าย ในทุก ๆ การรับข้อมูลได้ 1 บิต จะมีพัลส์ SHIFT เกิดขึ้น เพื่อทำการเลื่อนข้อมูลเข้าสู่รีจิสเตอร์บัฟเฟอร์การรับข้อมูล การกำหนดจังหวะการรับข้อมูลใช้สัญญาณนาฬิกาการรับข้อมูล (RX clock) หลังจากสัญญาณนาฬิกาลูกสุดท้าย อันหมายถึงสามารถรับข้อมูลได้ครบแล้ว วงจรควบคุมการรับข้อมูลจะทำการส่งข้อมูลจากรีจิสเตอร์บัฟเฟอร์ไปยังรีจิสเตอร์ SBUF และบิต RB8 ในรีจิสเตอร์ SCON โดยข้อมูลในบิต RB8 ก็คือข้อมูลของบิตหยุดนั่นเอง พร้อมกันนั้นยังทำการเซตบิต RI ในรีจิสเตอร์ SCON

ด้วย หากการอินเตอร์รัปต์จากพอร์ตอนุกรมได้รับการเอ็นเอเบิลไว้ก็จะเกิดการอินเตอร์รัปต์ขึ้นในระบบ หลังจากอินเตอร์รัปต์หรือรับข้อมูลเรียบร้อยแล้ว ต้องทำการเคลียร์บิต RI ก่อนเพื่อให้การรับส่งข้อมูลทางพอร์ตอนุกรมดำเนินต่อไปได้

การทำงานในโหมดนี้ได้รับความนิยมสูงสุด เนื่องจากมีกระบวนการที่ไม่ซับซ้อนและสามารถทำการรับส่งข้อมูลกับคอมพิวเตอร์ได้อย่างมีประสิทธิภาพ

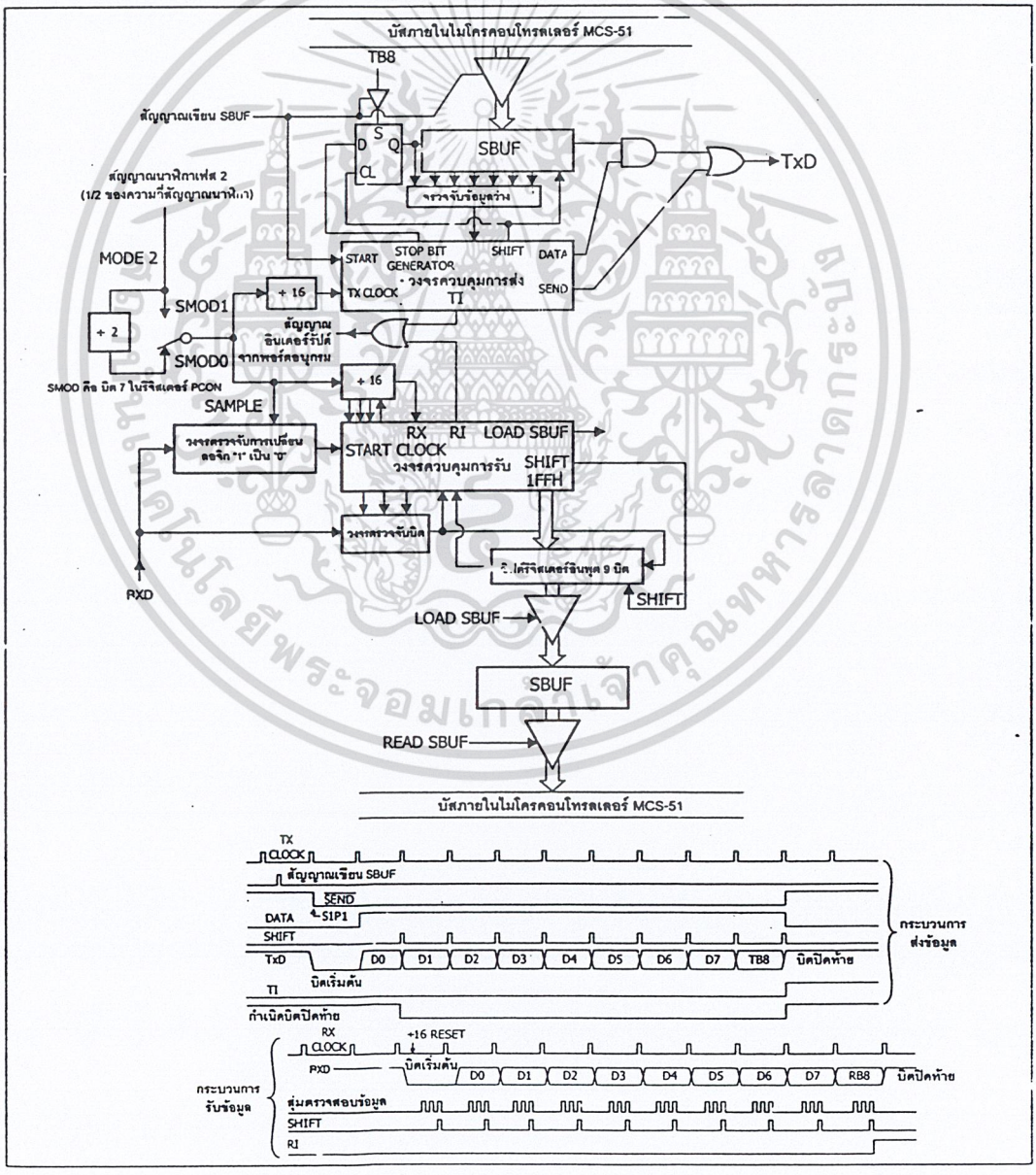


รูปที่ 2.26 ไดอะแกรมการทำงานในโหมด 1 ของพอร์ตอนุกรมภายในไมโครคอนโทรลเลอร์ MCS-51

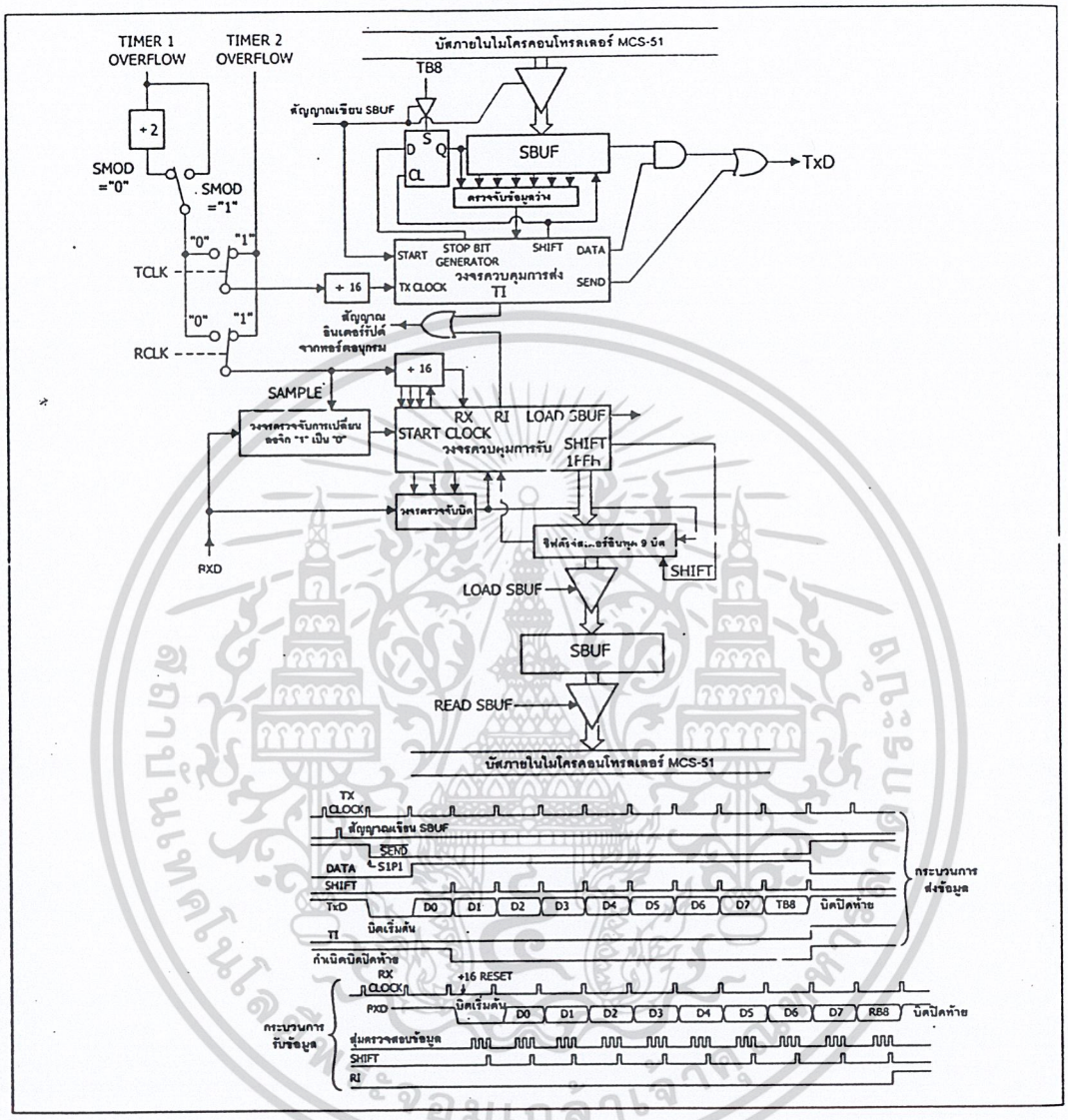
การทำงานในโหมด 2 และ 3 ของวงจรพอร์ตอนุกรม

ในทั้งสองโหมดนี้จะใช้รูปแบบข้อมูลรวม 11 บิต ประกอบด้วยบิตเริ่มต้น มีค่าเป็น “0” จำนวน 1 บิต, บิตข้อมูล 8 บิต โดยทำการรับและส่งบิต LSB ก่อน, บิตข้อมูลบิตที่ 9 และ บิตท้ายมีค่าเป็น “1” จำนวน 1 บิต ในการส่งข้อมูล ข้อมูลบิตที่ 9 จะได้รับการเก็บไว้ในบิต TB8 ในรีจิสเตอร์ SCON และในการรับข้อมูล ข้อมูลบิตที่ 9 จะนำไปเก็บไว้ในบิต RB8 ในรีจิสเตอร์ SCON สำหรับอัตราบอดในโหมด 2 จะคงที่โดยเลือกได้ 2 ค่าคือ 1/32 หรือ 1/64 ของความถี่สัญญาณนาฬิกา สำหรับในโหมด 3 อัตราบอดสามารถปรับได้เหมือนกับในโหมด 1

ในรูปที่ 2.27 และ 2.28 เป็นไดอะแกรมการทำงานและไดอะแกรมการทำงานในโหมด 2 และ 3 ของพอร์ตอนุกรม การทำงานโดยรวมจะคล้ายกับการทำงานในโหมด 1 ส่วนที่แตกต่างกันคือจำนวนบิตของข้อมูลที่ในโหมด 2 และ 3 จะมีเพิ่มมาอีก 1 บิต โดยส่วนใหญ่จะใช้เป็นบิตตรวจสอบพาริตี้



รูปที่ 2.27 ไดอะแกรมการทำงานในโหมด 2 ของพอร์ตอนุกรมภายในไมโครคอนโทรลเลอร์ MCS-51



รูปที่ 2.28 ไดอะแกรมการทำงานในโหมด 3 ของพอร์ตอนุกรมภายในไมโครคอนโทรลเลอร์ MCS-51

2.9.4 อัตราบอดของพอร์ตอนุกรมในไมโครคอนโทรลเลอร์ MCS-51

โหมด 0

อัตราบอดของโหมดนี้มีค่าคงที่ โดยสามารถคำนวณได้จากสูตร

อัตราบอดในโหมด 0 = ความถี่ของสัญญาณนาฬิกา/12 หน่วยเป็น บิตต่อวินาที

โหมด 1 และ 3

เนื่องจากทั้งสองโหมดนี้สามารถเลือกแหล่งกำเนิดอัตราบอดได้ 2 แหล่งคือ จากอัตราโอเวอร์โฟลวของไทเมอร์ 1 และ 2 สำหรับอัตราบอดเมื่อใช้โอเวอร์โฟลวของไทเมอร์ 1 จะต้องใช้ค่าของบิต SMOD ในรีจิสเตอร์ PCON (จะกล่าวถึงรายละเอียดของรีจิสเตอร์ตัวนี้ในบทที่ว่าด้วยการทำงานในโหมดประหยัดพลังงาน) มาพิจารณาประกอบด้วย สามารถคำนวณอัตราบอดได้จาก

อัตราบอด = $(2^{\text{ค่าของบิต SMOD}}/32)$ X อัตราโอเวอร์โฟลวของไทเมอร์ 1

ถ้าหากในไทเมอร์ 1 ไม่ได้เอ็นเอเบิลการอินเตอร์รัปต์ไว้ สามารถคำนวณค่าอัตาบอดได้จาก

อัตราบอด = $(2^{\text{ค่าในรีจิสเตอร์ SMOD}}/32)$ X (ความถี่สัญญาณนาฬิกา/{12X[256-(TH1)]})

ในตารางที่ 2.3 แสดงการกำหนดอัตราบอดโดยใช้ไทเมอร์ 1

ในกรณีที่ใช้ไทเมอร์ 2 ในการกำหนดอัตราบอด โดยกำหนดให้ไทเมอร์ 2 ทำงานในโหมดกำเนิดอัตราบอด (baud rate generator) สามารถคำนวณหาค่าอัตราบอดได้จาก

อัตราบอด = อัตราโอเวอร์โฟลวของไทเมอร์ 2/16 หน่วยเป็น บิตต่อวินาที

ถ้าหากกำหนดให้ไทเมอร์ 2 ทำงานในโหมดปกติ สามารถคำนวณหาค่าอัตรบอดได้จาก

อัตราบอด = ความถี่ของสัญญาณนาฬิกา/(32 X (65536-(RCAP2H,RCAP2L)))

โดยที่ (RCAP2H, RCAP2L) เป็นค่าของรีจิสเตอร์ RCAP2H และ RCAP2L มีขนาด 16 บิต ไม่คิดเครื่องหมาย

โหมด 2

ในโหมดนี้อัตราบอดจะขึ้นอยู่กับค่าของบิต SMOD ในรีจิสเตอร์ PCON ถ้า SMOD เป็น "0" อัตราบอดจะเท่ากับ 1/64 ของความถี่สัญญาณนาฬิกา ในกรณีที่ SMOD เป็น "1" อัตราบอดจะเท่ากับ 1/32 ของความถี่สัญญาณนาฬิกา สามารถแสดงเป็นสูตรคำนวณทางคณิตศาสตร์ได้ดังนี้

อัตราบอด = $(2^{\text{ค่าของบิต SMOD}}/64)$ X ความถี่สัญญาณนาฬิกา

2.9.5 การกำหนดค่าของไทเมอร์เพื่อเลือกอัตราบอด

ในการใช้งานพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51 สิ่งที่ต้องให้ความสนใจมากที่สุดประการหนึ่งคือ อัตราการถ่ายทอดข้อมูล หรือ อัตราบอด ซึ่งการกำหนดอัตราบอดนั้นจะขึ้นอยู่กับค่าความถี่ของสัญญาณนาฬิกาเป็นหลัก สำหรับโหมดการทำงานของพอร์ตอนุกรมที่สามารถเลือกอัตราบอดได้อย่างอิสระคือ ในโหมด 1 และ 3 โดยกำหนดได้จากอัตราการเกิดโอเวอร์โฟลวของไทเมอร์ 1 ถ้าหากไทเมอร์ 1 มีการเกิดโอเวอร์โฟลวในอัตราที่สูงมากเท่าใด อัตราบอดก็จะมีอัตราสูงมากขึ้นตาม นั่นหมายความว่า อัตราในการถ่ายทอดข้อมูลจะสูงมาก สามารถถ่ายทอดข้อมูลได้อย่างรวดเร็ว

ในการใช้ไทเมอร์ 1 เพื่อกำหนดอัตราการบอดในโหมด 1 และ 3 ของพอร์ตอนุกรมจะต้องกำหนดให้ไทเมอร์ 1 ทำงานในโหมด 2 หรือโหมด 8 บิตแบบตั้งค่าการนับอัตโนมัติ และการกำหนดค่ารีโหลดให้แก่รีจิสเตอร์ TH1 จึงเป็นตัวแปรหลักที่ใช้ในการกำหนดอัตราบอดให้แก่พอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51

เริ่มต้นด้วยการเคลียร์บิต SMOD ซึ่งเป็นบิต 7 ของรีจิสเตอร์ PCON ให้เป็น "0" ค่าของการรีโหลดให้แก่ TH1 สามารถคำนวณได้จาก

TH1 = 256-((ค่าความถี่ของคลิสตอล/384)/อัตราบอด)

แต่ถ้าบิต SMOD เกิดจากเซต จะเป็นการเอ็นเอเบิลการทวีคูณของอัตราบอด ดังนั้นการกำหนดค่าให้แก่ TH1 จึงต้องคำนวณจาก

TH1 = 256-((ค่าความถี่ของคลิสตอล/192)/อัตราบอด)

ยกตัวอย่าง ถ้าหากในไมโครคอนโทรลเลอร์ AT89C51 ใช้คลิสตอล 11.0592 MHz ต้องการกำหนดอัตราบอดของพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ไว้ที่ 19,200 บิตต่อวินาที ในกรณีที่ไม่เอ็นเอเบิลการทวีคูณของอัตราบอด ค่ารีโหลดของไมโครคอนโทรลเลอร์จะเท่ากับ

TH1 = 256-((ค่าความถี่ของคลิสตอล/384)/อัตราบอด)

= 256-((11059200/384)/19200)

= 256/(28800/19200)

= 256-1.5 = 254.5

เนื่องจากผลลัพธ์ที่ได้เป็นค่าที่ไม่ได้เป็นจำนวนเต็ม ถ้าหากกำหนดค่า TH1 เป็น 254 เมื่อทำการแทนค่าเพื่อคำนวณหาอัตราบอด จะได้อัตราบอดเท่ากับ 14,400 บิตต่อวินาที และถ้าหากกำหนดค่าของ TH1 เป็น 255 อัตราบอดจะมีค่าเท่ากับ 28,800 บิตต่อวินาที ดังนั้นจะเห็นได้ค่าของ TH1 ที่ไม่เป็นจำนวนเต็มจะไม่สามารถทำให้เกิดอัตราบอดตามที่ต้องการได้

ทางแก้ไขคือ ทำให้การเอ็นเอเบิลการทวีคูณอัตราบอด โดยการเซตบิต SMOD ในรีจิสเตอร์ PCON ให้เป็น "1" จากนั้นแทนค่าลงในสมการหาค่า TH1 เมื่อมีการเซตบิต SMOD ได้ผลดังนี้

TH1 = 256-((ค่าความถี่ของคลิสตอล/192)/อัตราบอด)

= 256-((11059200/192)/19200)

= 256-57600/19200)

= 256-3 = 253

นำค่าของ TH1 ที่ได้ทำการแทนค่าคำนวณหาค่าอัตราบอดจะได้เท่ากับ 19,200 บิตต่อวินาที

สามารถสรุปขั้นตอนในการเลือกอัตราบอดโดยการกำหนดค่าของไทเมอร์ 1 ได้ดังนี้

1. กำหนดให้พอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51 ทำงานในโหมด 1 หรือ 3

2. กำหนดให้ไทเมอร์ 1 ทำงานในโหมด 2 หรือโหมด 8 บิตตั้งค่าอัตโนมัติ

3. กำหนดข้อมูลให้แก่ TH1 เท่ากับ 253 เพื่อให้สามารถกำเนิดอัตราบอดได้ 19,200 บิตต่อวินาทีตามที่ต้องการ

4. ทำการเซตบิต SMOD ซึ่งเป็นบิต 7 ของรีจิสเตอร์ PCON เพื่อเอ็นเอเบิลการทวีคูณอัตราบอด

อัตราบอด (บิตต่อวินาที : bps)	ความถี่สัญญาณนาฬิกา	SMOD	ไทเมอร์ 1		
			C/T	โหมด	ค่ารีโหลด
โหมด 0 : สูงสุด 1 MHz	12 MHz	X	X	X	X
โหมด 2 : สูงสุด 375K	12 MHz	1	X	X	X
โหมด 1,3 : 62.5K	12 MHz	1	0	2	FFH
19.2K (19,200)	11.0592 MHz	1	0	2	FDH
9.6K (9,600)	11.0592 MHz	0	0	2	FDH
4.8K (4,800)	11.0592 MHz	0	0	2	FAH
2.4K (2,400)	11.0592 MHz	0	0	2	F4H
1.2K (1,200)	11.0592 MHz	0	0	2	E8H
137.5	11.0592 MHz	0	0	2	1DH
110	6 MHz	0	0	2	72H
110	12 MHz	0	0	1	FEEBH

ตารางที่ 2.3 การเลือกอัตราบอดของวงจรพอร์ตอนุกรมภายในไมโครคอนโทรลเลอร์ MCS-51

2.9.6 การเขียนหรือส่งข้อมูลออกจากพอร์ตอนุกรม

ข้อมูลที่ต้องการส่งออกทุกค่าต้องนำไปเก็บในรีจิสเตอร์บัฟเฟอร์ของพอร์ตอนุกรมซึ่งก็คือรีจิสเตอร์ SBUF ดังตัวอย่าง

```
MOV SBUF, #'A'
```

จากคำสั่งข้างต้นเป็นการส่งข้อมูลของตัวอักษร A ออกไปยังพอร์ตอนุกรมของไมโครคอนโทรลเลอร์อย่างไรก็ตามก่อนทำการส่งข้อมูลทุกครั้งต้องแน่ใจว่าบิต TI เคลียร์หรือมีค่าเป็น "0" และเมื่อทำการส่งข้อมูลเรียบร้อยแล้ว ก็จะเกิดการเซตบิต TI เพื่อแจ้งให้ทราบ ดังตัวอย่างโปรแกรมต่อไปนี้

```
CLR TI              ; เคลียร์บิต TI เพื่อเตรียมการส่งข้อมูลออก
MOV SBUF , #'A'     ; ส่งข้อมูลของตัวอักษร A ไปยังพอร์ตอนุกรม
JNB TI , $          ; รอการเซตของบิต TI เพื่อแจ้งการส่งข้อมูลที่เสร็จสมบูรณ์
```

2.9.7 การอ่านหรือรับข้อมูลจากพอร์ตอนุกรม

การรับข้อมูลจากพอร์ตอนุกรมสามารถกระทำได้ง่ายมาก เพียงทำการตรวจสอบว่าบิต RI เกิดการเซตขึ้นหรือไม่ ถ้าพบว่ามีการเซตเกิดขึ้นแล้ว ให้ทำการอ่านจากรีจิสเตอร์ SBUF โดยต้องทำการโอนย้ายข้อมูลผ่านทางแอกคิวมูเลเตอร์หรือรีจิสเตอร์ A ดังตัวอย่าง

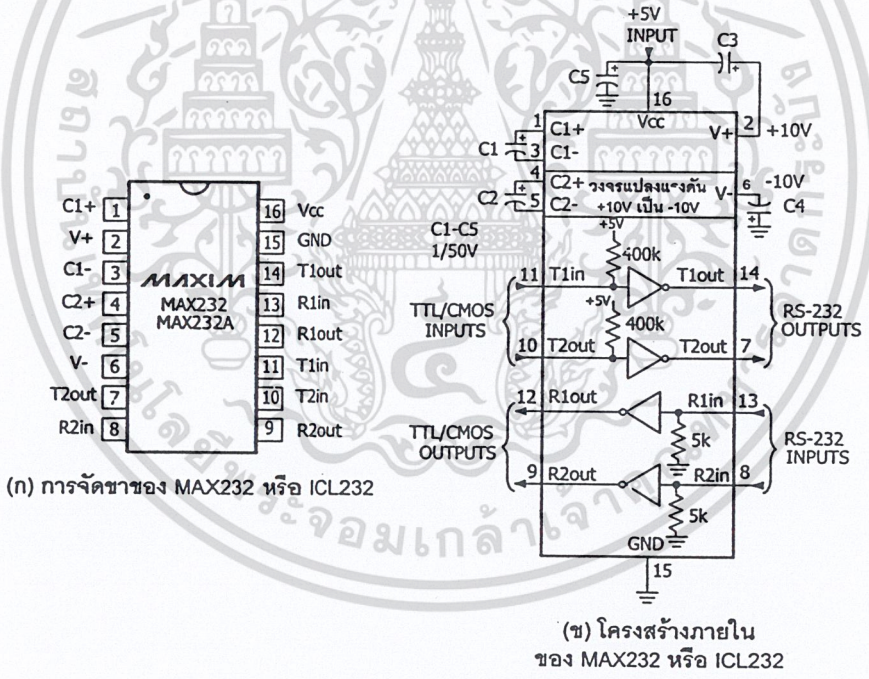
```
CLR RI              ; เคลียร์บิต RI เพื่อเตรียมการส่งข้อมูลออก
```

```
JNB RI , $          ; รอคอยการเซตของบิต RI อันเป็นการแจ้งให้ทราบว่า การรับ
                    ; ข้อมูลเสร็จสมบูรณ์และมีข้อมูลเกิดขึ้นที่รีจิสเตอร์ SBUF
MOV A, SBUF         ; อ่านค่าจากรีจิสเตอร์ โดยการโอนย้ายข้อมูลผ่านทางรีจิสเตอร์ A
CLR RI              ; หลังจากทำการอ่านข้อมูลเรียบร้อยแล้ว ต้องทำการเคลียร์บิต RI
```

!สมอ

2.9.8 การเชื่อมต่อกับพอร์ตอนุกรมของคอมพิวเตอร์

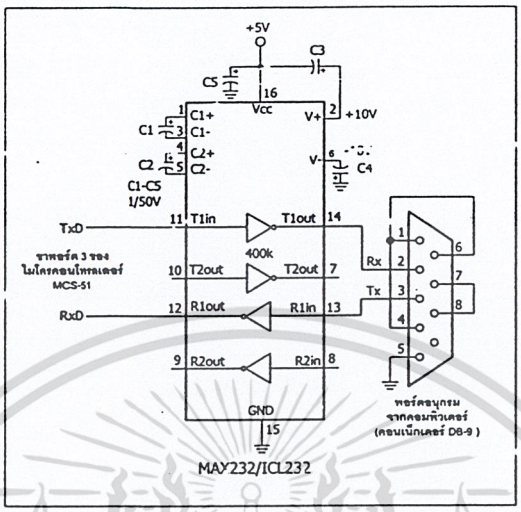
การใช้งานวงจรพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51 มักนิยมใช้ในการติดต่อเพื่อแลกเปลี่ยนข้อมูลกับคอมพิวเตอร์ผ่านทางพอร์ตอนุกรมในมาตราฐาน RS-232 เป็นส่วนใหญ่ แต่เนื่องจากระดับสัญญาณของพอร์ตอนุกรม RS-232 มีระดับตั้งแต่ +-3 ถึง +-12V ในขณะที่ระดับสัญญาณของไมโครคอนโทรลเลอร์ MCS-51 อยู่ในระดับทีทีแอล ดังนั้นจึงไม่สามารถเชื่อมต่อพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51 เข้ากับพอร์ตอนุกรมของคอมพิวเตอร์ได้โดยตรง จึงต้องอาศัยการเชื่อมต่อผ่านไอซีพิเศษที่ทำหน้าที่ในการแปลงระดับสัญญาณ



รูปที่ 2.29 รายละเอียดเบื้องต้นของไอซีแปลงสัญญาณเพื่อเชื่อมต่อกับพอร์ตอนุกรมของคอมพิวเตอร์

ไอซีที่ทำหน้าที่ในการแปลงระดับสัญญาณนี้ ต้องทำการแปลงข้อมูลส่งของไมโครคอนโทรลเลอร์ MCS-51 จากระดับทีทีแอลไปเป็นระดับของ RS-232 และทำการแปลงข้อมูลรับจากคอมพิวเตอร์จากระดับของ RS-232 เป็นระดับทีทีแอล เพื่อให้สามารถถ่ายทอดไปยังไมรคอนโทรลเลอร์ MCS-51 ได้อย่างสมบูรณ์ ไอซีดังกล่าวมีด้วยกันหลายเบอร์จากหลายผู้ผลิต อาทิ MAX232 จาก MAXIM

หรือ ICL232 จาก HARRIS เป็นต้น ในรูปที่ 2.29 แสดงการจัดขาของไอซี ICL232 ซึ่งใช้ในการแปลงสัญญาณ RS-232 ส่วนวงจรของการต่อกับไมโครคอนโทรลเลอร์ MCS-51 แสดงในรูป 2.30

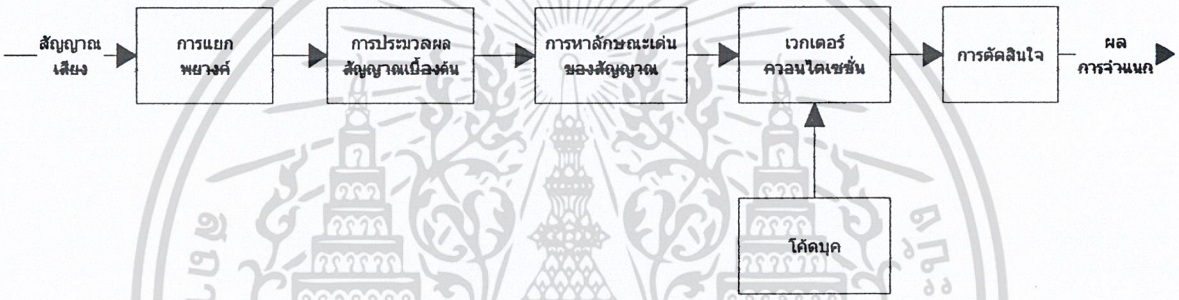


รูปที่ 2.30 วงจรเชื่อมต่อ MAX232 หรือ ICL232 เข้ากับพอร์ตอนุกรมของคอมพิวเตอร์และไมโครคอนโทรลเลอร์ MCS-51

บทที่ 3
การคำนวณและการสร้าง

จากทฤษฎีและหลักการสามารถแบ่งการทำงานของการควบคุมเครื่องใช้ไฟฟ้าด้วยเสียงพูดได้เป็น 2 ส่วน คือ ส่วนของโปรแกรมจำแนกเสียงพูด (Software) และส่วนของวงจรจำลองการควบคุมอุปกรณ์ไฟฟ้า (Hardware)

ปริญญานิพนธ์นี้ได้เน้นไปที่ส่วนของการเขียนโปรแกรมจำแนกเสียงพูด โดยแบ่งการทำงานของโปรแกรมออกเป็นส่วนต่างๆ ได้ตามบล็อกไดอะแกรมดังรูปที่ 3.1 ซึ่งประกอบด้วย ภาคการประมวลผลเบื้องต้น (Preprocessing) ภาคการวิเคราะห์หาลักษณะเด่นของสัญญาณ (Feature Extraction) ภาคเวกเตอร์ควอนไตเซชั่น (Vector Quantization) ภาคโค้ดบุค (Codebook)และภาคการตัดสินใจ (Decision rule) โดยก่อนทำการการประมวลผลเบื้องต้น จะทำการแยกพยางค์ของสัญญาณเสียง และทำการหาขอบเขตของคำที่แท้จริงก่อนที่จะทำการส่งไปยังภาคอื่น ๆ ต่อไป



รูปที่ 3.1 แสดงบล็อกไดอะแกรมการจำแนกเสียงพูดโดยทั่วไป

โครงงานที่ได้ทำการศึกษาและทดลอง ในส่วนของการรู้จำเสียงมีวิธีการทำงาน ดังแสดงในบล็อกไดอะแกรมดังรูปที่ 3.1 ซึ่งอธิบายได้ดังนี้

3.1 การแยกพยางค์ของสัญญาณเสียงคำสั่ง

สัญญาณเสียงคำสั่งที่ใช้ กำหนดให้ใช้สัญญาณเสียงคำสั่งความยาว 3 พยางค์ โดยกำหนดให้มีระยะเวลาในการสั่งงานไม่เกิน 4 วินาที และ การสั่งงานจะต้องมีการเว้นระยะเวลาระหว่างแต่ละพยางค์เพื่อให้โปรแกรมสามารถทำการแยกพยางค์ได้ การแบ่งสัญญาณเสียงคำสั่งออกเป็นคำเดี่ยวมีกระบวนการดังต่อไปนี้

1. รับสัญญาณเสียงคำสั่งจากไมโครโฟน คำนวณหาค่าพลังงานของสัญญาณในช่วงเวลาที่ถูกบันทึกทั้งหมด โดยคำนวณหาค่าพลังงานของสัญญาณทีละเฟรม กำหนดให้เฟรมหนึ่งที่ขนาด 200 แซมเปิ้ล และทำการคำนวณไปเรื่อยๆ จนครบทุกเฟรม

2. ทำการเปรียบเทียบพลังงานของสัญญาณเสียงทีละเฟรมกับระดับพลังงานอ้างอิงที่กำหนดไว้ จะได้ช่วงที่คาดว่าเป็นสัญญาณเสียงออกมา

3. หาค่าพลังงานสูงสุดของแต่ละช่วงที่คาดว่าเป็นสัญญาณเสียงที่ได้จากขั้นตอนที่แล้ว นำค่าพลังงานที่ได้มาเปรียบเทียบกันและเลือกช่วงที่ที่ค่าพลังงานที่มากที่สุดจำนวน 3 ช่วงออกมา

4. เมื่อได้ช่วงของพลังงานแล้ว ก็สามารถแบ่งสัญญาณเสียงคำสั่งออกเป็น 3 พยางค์ได้

5. นำสัญญาณเสียงคำสั่งไปทำการรู้จำเสียงทีละพยางค์

3.2 การประมวลผลสัญญาณเบื้องต้น (Preprocessing)

3.2.1 การหาขอบเขตของคำที่แท้จริง

ขั้นตอนในภาคนี้ จะใช้หลักการวิเคราะห์ค่าพลังงานของสัญญาณเสียงในโดเมนเวลา ซึ่งมีวิธีการดังต่อไปนี้

1. คำนวณหาค่าพลังงานของสัญญาณในช่วงเวลาที่ถูกบันทึกทั้งหมด โดยการคำนวณหาค่าพลังงานของสัญญาณจะทำการคำนวณเป็นเฟรม กำหนดให้เฟรมหนึ่งที่ขนาด 100 แซมเปิ้ล และทำการคำนวณไปเรื่อย ๆ จนครบทุกเฟรม

2. พิจารณาหาค่าพลังงานสูงสุดของสัญญาณ (Maximum Energy) หาเฟรมที่มีค่าพลังงานสูงสุดแล้วกำหนดให้เฟรมนั้นเป็นเฟรมอ้างอิง

3. กำหนดระดับอ้างอิง (Reference) โดยระดับอ้างอิงจะบอกว่า จุดเริ่มต้นและจุดสุดท้ายของคำจะอยู่ในช่วงใด คำนวณได้จาก (Cutfactor*Emax) ในที่นี้จะใช้ค่า Cutfactor = 0.2

4. การหาจุดเริ่มต้นของคำ เมื่อกำหนดระดับอ้างอิงเป็นที่เรียบร้อยแล้ว พิจารณาเปรียบเทียบค่าพลังงานของสัญญาณกับระดับอ้างอิง โดยเฟรมแรกเริ่มที่เฟรมที่มีค่าระดับพลังงานสูงสุดก่อน แล้วเลื่อนเฟรมที่จะเปรียบเทียบพลังงานกับระดับพลังงานอ้างอิงไปด้านหลังเรื่อยๆ จนกว่าจะพบเฟรมที่มีค่าพลังงานน้อยกว่าเฟรมอ้างอิง

5. การหาจุดสุดท้ายของคำ จะทำการพิจารณาค่าพลังงานไปข้างหน้า เช่นเดียวกับการหาจุดเริ่มต้นของคำ จนกว่าจะพบเฟรมที่มีค่าพลังงานน้อยกว่าเฟรมอ้างอิง จะได้เฟรมที่เป็นจุดสุดท้ายของคำ

3.2.2 การนอร์มอลไลซ์ (Normalization)

เนื่องจากสัญญาณเสียงพูดแต่ละคำมีความยาวไม่เท่ากัน จึงต้องมีการนอร์มอลไลซ์สัญญาณเสียงพูดที่ได้จากการตัดคำให้มีความยาวเท่ากันเสียก่อน โดยจะใช้หลักการ Interpolation จาก Numerical Mathematics คำนวณจากสมการ

$$g(x) = \frac{b-x}{b-a} f(a) + \frac{x-a}{b-a} f(b)$$

3.2.3 การกรองแบบพรีเอมฟาซีส (Peemphasis)

ใช้วงจรกรองความถี่อันดับหนึ่ง โดยมีฟังก์ชั่นถ่ายโอนดังนี้

$$H(z) = 1 - az^{-1}$$

ค่า a ที่ใช้คือ $15/16 = 0.9375$

3.2.4 การแบ่งช่วงสัญญาณ

ขนาดของช่วงสัญญาณมีเงื่อนไขในการเลือกคือ

- ค่า M ต้องสั้นพอที่คุณสมบัติของเสียงไม่เปลี่ยนแปลง
- ค่า N ต้องยาวพอที่จำนวนของตัวอย่างมีเพียงพอสำหรับการหาสัมประสิทธิ์
- การเลื่อนในการวิเคราะห์ (ค่า M) ต้องไม่ข้ามข้อมูล

ดังนั้นค่า M จะต้องน้อยกว่าค่า N แต่ถ้าค่า M มีขนาดเล็กเกินไป จะทำให้การคำนวณช้าลง จึงเลือกค่า M = 80 แซมเปิ้ล และค่า N = 240 แซมเปิ้ล

ความถี่ที่ใช้ในการสุ่มค่าสัญญาณ

เนื่องจากความถี่ที่ใช้ในการแซมเปิ้ล ต้องมีค่ามากกว่า หรือ เท่ากับสองเท่าของความถี่เสียง $f_s \geq f_{max}$ เนื่องจากความถี่ของเสียงพูดมีความถี่อยู่ในช่วง 300 – 3400 เฮิรตซ์ จึงใช้ความถี่แซมเปิ้ลที่ 8 กิโลเฮิรตซ์

ดังนั้น ช่วงเวลาในการวิเคราะห์แต่ละเฟรม คือ 240/8000 = 30 ms และระยะที่ใช้ในการเลื่อนแต่ละเฟรม คือ 80/8000 = 10 ms

การเลือกวินโดว์ที่เหมาะสมสำหรับการวิเคราะห์เสียง

โดยพิจารณาจากลักษณะสเปคตรัม คือ

- ความถี่รีโซลูชั่นสูง (high frequency resolution) คือ มีโลบหลักแคบและแหลม
- การลดทอน (Attenuation) นอกช่วงความถี่ที่ผ่านได้ต่ำ คือ ไซด์โลบมีค่าน้อย ฟังก์ชั่นวินโดว์มีหลายชนิด แต่ชนิดที่เหมาะสมที่สุดที่นำมาใช้ได้แก่ แฮมมิ่งวินโดว์ (Hamming Window) ซึ่งมีค่าไซด์โลบต่ำ (-40 dB) และเมนโลบแคบพอใช้ มีสมการดังนี้

$$w(n) = 0.54 - 0.46 \cos\left(\frac{2\pi n}{N-1}\right) \quad \text{เมื่อ } n = 0, 1, 2, \ldots, N\text{-}1$$

3.3 การวิเคราะห์หาลักษณะเด่นของสัญญาณ (Feature Extraction)

ใช้ในการประมาณพันธะเชิงเส้นในการวิเคราะห์หาค่าสัมประสิทธิ์ LPC ซึ่งการประมาณเชิงเส้นที่เลือกใช้คือ วิธีอัตตสัมพันธ์ (autocorrelation method) ซึ่งเป็นวิธีที่ง่ายในการคำนวณมากกว่าวิธีอื่น ๆ และมีความแน่นอนด้านเสถียรภาพ อีกทั้งมีการเก็บข้อมูลที่น้อยกว่า

เนื่องจากการวิเคราะห์โดยวิธีอัตตสัมพันธ์ อันดับการประมาณเชิงเส้น (P) ที่มากจะทำให้การประมาณเสียงมีความใกล้เคียงมากยิ่งขึ้น แต่ถ้าอันดับ P มากเกินไปจะทำให้การคำนวณมีความยุ่งยากและใช้เวลานาน ดังนั้นเพื่อความเหมาะสมค่าอันดับ P ที่ใช้คือ 12 โดยผลที่ได้อยู่ในรูปของสัมประสิทธิ์เซปสตรัมที่ถ่วงน้ำหนักแล้ว

ก่อนที่จะเข้าสู่ภาคเวกเตอร์ควอนไตเซชันและภาคการตัดสินใจ ต้องมีการสร้างโค้ดบุคและแบบจำลอง HMM ของเสียงเสียก่อน ซึ่งเรียกว่าขั้นตอนการเรียนรู้ ดังนี้

3.4 ขั้นตอนการเรียนรู้

3.4.1 การสร้างโค้ดบุค (Codebook)

จากการทดสอบ (L.R.Labiner S.E.Levinson และ Sondhi. 1982) จะได้ว่าที่ขนาดโค้ดบุคเท่ากับ 128 จะมีค่าความคลาดเคลื่อนน้อยที่สุด แต่ในการทดลองผู้จัดทำได้เลือกใช้โค้ดบุค 64 ซึ่งช่วยลดเวลาในการคำนวณได้มากและมีค่าความผิดพลาดที่สามารถยอมรับได้ โดยวิธีการคำนวณแบบในการหาค่าระยะทางเนื่องจากเป็นวิธีที่ง่ายและรวดเร็ว

การสร้างโค้ดบุค โดยนำเทรนนิ่งเซตที่ได้จากการประมาณเชิงเส้นมาผ่านกระบวนการดังนี้

1. สุ่มค่าโค้ดบุคเริ่มต้นโดยเลือกจากเทรนนิ่งเซตมา 64 ตัว ตัวละ 12 มิติ
2. หาระยะทางระหว่างโค้ดบุคกับเทรนนิ่งเซตแต่ละตัว โดยใช้ค่าความคลาดเคลื่อนกำลังสอง
3. จัดกลุ่มของเวกเตอร์อินพุต โดยพิจารณาจากระยะทางที่น้อยที่สุด
4. หาจุดศูนย์กลางร่วมของกลุ่มโดยเฉลี่ยจากเวกเตอร์ทั้งหมดที่อยู่ในกลุ่ม
5. ทำขั้นตอน 3 และ 4 จนกว่าค่าความคลาดเคลื่อนรวมจะน้อยกว่า 0.0001 ซึ่งจุดศูนย์กลางร่วมที่ได้คือโค้ดบุคนั่นเอง

3.4.2 การสร้างแบบจำลอง HMM ของเสียง มีขั้นตอนดังนี้

1. สุ่มค่าเริ่มต้น a , b และกำหนดให้ π = [100000] ตามเงื่อนไขในการใช้แบบจำลองแบบ Left-Right Model
2. หาค่า α,β จากค่า a , b เริ่มต้น และลำดับค่าปรากฏ $O=\{o_1o_2o_3...o_T\}$ ซึ่งเรียกว่าลำดับเทรนนิ่ง ตามวิธีของ Forward-Backward Procedure โดยใช้ลำดับของค่าปรากฏหลาย ๆ ลำดับเข้ามาเทรนเพื่อความถูกต้องมากขึ้น
3. การทำสเกลลิ่ง เพื่อให้ค่าอยู่ในย่านที่คอมพิวเตอร์สามารถคำนวณได้อย่างถูกต้อง
4. หาค่าพารามิเตอร์ λ ที่ให้ค่าความน่าจะเป็นสูงสุด ที่จะเป็นแบบจำลอง λ ที่เหมาะสมของคำ
5. ตรวจสอบค่าพารามิเตอร์ a , b ของแบบจำลองที่ได้ว่าลู่เข้าหรือยัง โดยใช้วิธีการคำนวณค่า a ,b ซ้ำ ประมาณ 50 รอบ เมื่อมีการเปลี่ยนแปลงน้อยมากจนเป็นที่พอใจตามระดับค่าที่ตั้งไว้ ในที่นี้ให้เท่ากับ 10^{-5} ก็จะหยุด และจะได้ค่าพารามิเตอร์ a ,b และ π แบบจำลองที่ต้องการ
6. เก็บค่าพารามิเตอร์ a , b , π ที่ได้จากข้อ 5 เป็นพารามิเตอร์ของแบบจำลองไว้

3.5 ขั้นตอนการตัดสินใจ

3.5.1 การหาดัชนีโค้ดบุค

โดยการนำเวกเตอร์เสียงจากการประมาณเชิงเส้นทีละเสียง แล้วเปรียบเทียบกับโค้ดบุคที่ได้จากการสร้างในขั้นตอนการเรียนรู้ทีละเฟรม โดยวิธีความคลาดเคลื่อนกำลังสอง เวกเตอร์เสียงห่างจากโค้ด

บุคใดน้อยที่สุด จะได้ว่าเป็นดัชนีโค้ดบุคของเฟรมเสียงนั้น และเก็บดัชนีโค้ดบุคของแต่ละเฟรมในแต่ละเสียงไว้เป็นลำดับค่าปรากฏ (Observation sequence) สำหรับการสร้างแบบจำลองต่อไป

3.5.2 การรู้จำเสียง

หลังจากได้แบบจำลอง HMM ของแต่ละคำแล้ว เมื่อมีลำดับของค่าปรากฏ $O = \{o_1 o_2 o_3 \ldots o_T\}$ ของเสียง Unknown ซึ่งเป็นเสียงที่ต้องการทดสอบเข้ามา หลังจากนั้นจะทำการหาค่าความน่าจะเป็น $P(o|\lambda)$ ทุกแบบจำลองแต่ละศัพท์โดยใช้วิธี Viberti Algolithm แล้วเลือกเอาคำที่มีค่าความน่าจะเป็นสูงสุด ซึ่งก็คือคำศัพท์ที่แบบจำลองการจำได้นั่นเอง

3.6 การออกแบบวงจรไมโครคอนโทรลเลอร์ MCS-51 เพื่อเชื่อมต่อพอร์ตอนุกรม

1. ทำการต่อวงจรไมโครคอนโทรลเลอร์ MCS-51 เพื่อเชื่อมต่อพอร์ตอนุกรมดังรูปที่ 3.2
2. สำหรับการโปรแกรมอุปกรณ์ไมโครคอนโทรลเลอร์ MCS-51 เพื่อให้สามารถติดต่อสื่อสารกับพอร์ตอนุกรมได้ ต้องทำการตั้งค่ารูปแบบ (format) ของข้อมูลและอัตราบอด (baud rate) ให้ตรงกันเสียก่อน โดยในปริญญานิพนธ์นี้ ได้ทำการกำหนดค่าต่างๆ ดังนี้

   อัตราบอด (baud rate) = 9600 บิต / วินาที

   จำนวนข้อมูล = 8 บิต

   การตรวจสอบพาริตี้ = none

   บิตหยุด (stop bit) = 1 บิต
3. ทำการเขียนโปรแกรมและโปรแกรมลงในอุปกรณ์ไมโครคอนโทรลเลอร์ MCS-51 เพื่อทำทำการตรวจสอบข้อมูลที่ส่งออกจากจากพอร์ตอนุกรม ดังนี้

   ถ้าข้อมูลที่ส่งจากพอร์ตอนุกรมเป็น “L” ให้ทำการเซตบิต P0.0 เป็นการจำลองเพื่อบอกให้ทราบว่า ผู้ใช้ได้สั่งงานให้ “เปิด-ไฟ-หนึ่ง”

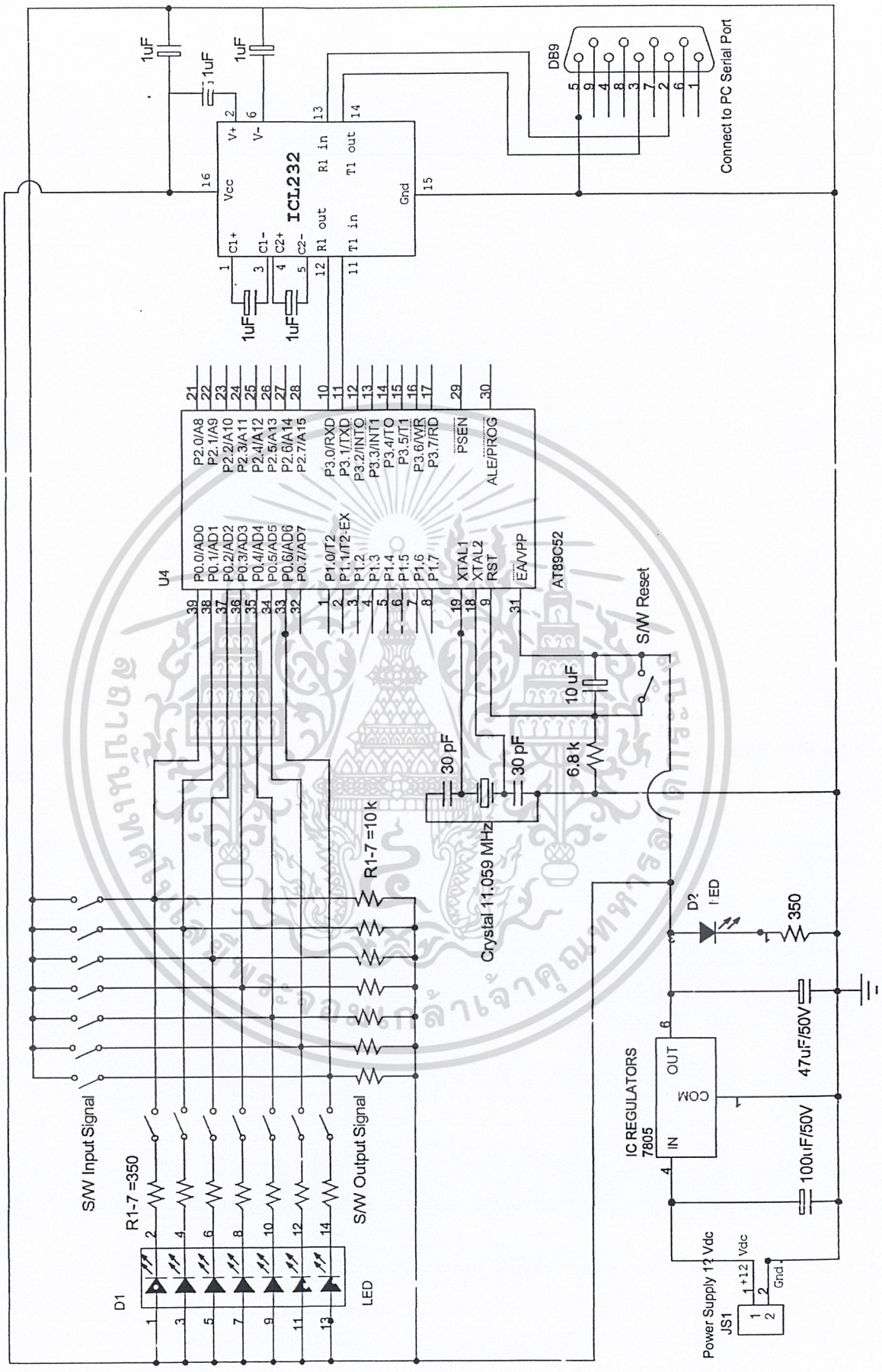
   ถ้าข้อมูลที่ส่งจากพอร์ตอนุกรมเป็น “M” ให้ทำการเซตบิต P0.1 เป็นการจำลองเพื่อบอกให้ทราบว่า ผู้ใช้ได้สั่งงานให้ “เปิด-ไฟ-สอง”

   ถ้าข้อมูลที่ส่งจากพอร์ตอนุกรมเป็น “N” ให้ทำการเซตบิต P0.2 เป็นการจำลองเพื่อบอกให้ทราบว่า ผู้ใช้ได้สั่งงานให้ “เปิด-ไฟ-สาม”

   ถ้าข้อมูลที่ส่งจากพอร์ตอนุกรมเป็น “X” ให้ทำการเคลียร์บิต P0.0 เป็นการจำลองเพื่อบอกให้ทราบว่า ผู้ใช้ได้สั่งงานให้ “ปิด-ไฟ-หนึ่ง”

   ถ้าข้อมูลที่ส่งจากพอร์ตอนุกรมเป็น “Y” ให้ทำการเคลียร์บิต P0.1 เป็นการจำลองเพื่อบอกให้ทราบว่า ผู้ใช้ได้สั่งงานให้ “ปิด-ไฟ-สอง”

   ถ้าข้อมูลที่ส่งจากพอร์ตอนุกรมเป็น “Z” ให้ทำการเคลียร์บิต P0.2 เป็นการจำลองเพื่อบอกให้ทราบว่า ผู้ใช้ได้สั่งงานให้ “ปิด-ไฟ-สาม”



รูปที่ 3.2 วงจรไมโครคอนโทรลเลอร์ MCS-51 ที่ใช้ในการทดลอง

บทที่ 4
การทดลองและผลการทดลอง

4.1 การทดลองในส่วนของโปรแกรมการรู้จำเสียง

4.1.1 ขั้นตอนการเรียนรู้

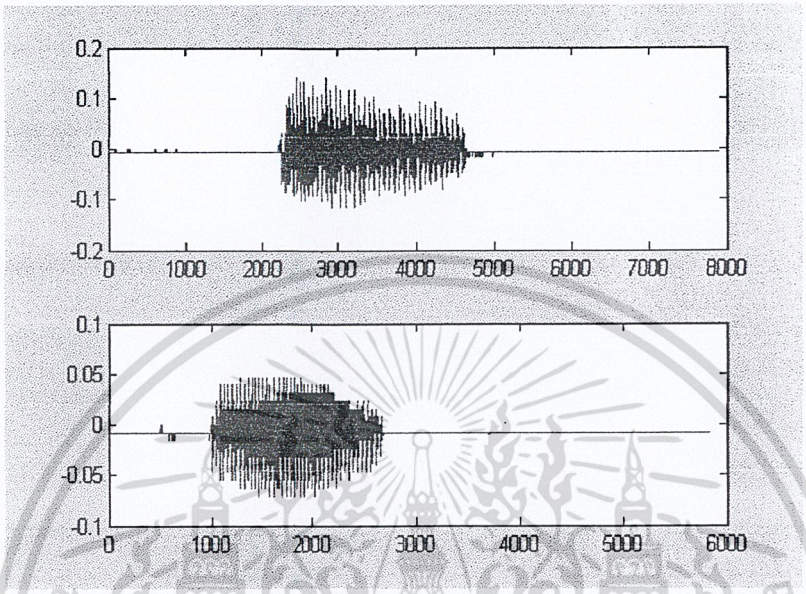
1) ทำการบันทึกเสียงที่ต้องการสร้างโมเดลซึ่งการบันทึกจะเลือกรูปแบบการบันทึกแบบ PCM ขนาด 8 บิต 8 กิโลเฮิรตซ์ โดย

บันทึกเสียง	"เปิด"	จากผู้ชาย 5 คนและผู้หญิง 5 คน คนละ 5 ครั้ง รวม 50 เสียง
บันทึกเสียง	"ปิด"	จากผู้ชาย 5 คนและผู้หญิง 5 คน คนละ 5 ครั้ง รวม 50 เสียง
บันทึกเสียง	"ไฟ"	จากผู้ชาย 5 คนและผู้หญิง 5 คน คนละ 5 ครั้ง รวม 50 เสียง
บันทึกเสียง	"หนึ่ง"	จากผู้ชาย 5 คนและผู้หญิง 5 คน คนละ 5 ครั้ง รวม 50 เสียง
บันทึกเสียง	"สอง"	จากผู้ชาย 5 คนและผู้หญิง 5 คน คนละ 5 ครั้ง รวม 50 เสียง
บันทึกเสียง	"สาม"	จากผู้ชาย 5 คนและผู้หญิง 5 คน คนละ 5 ครั้ง รวม 50 เสียง

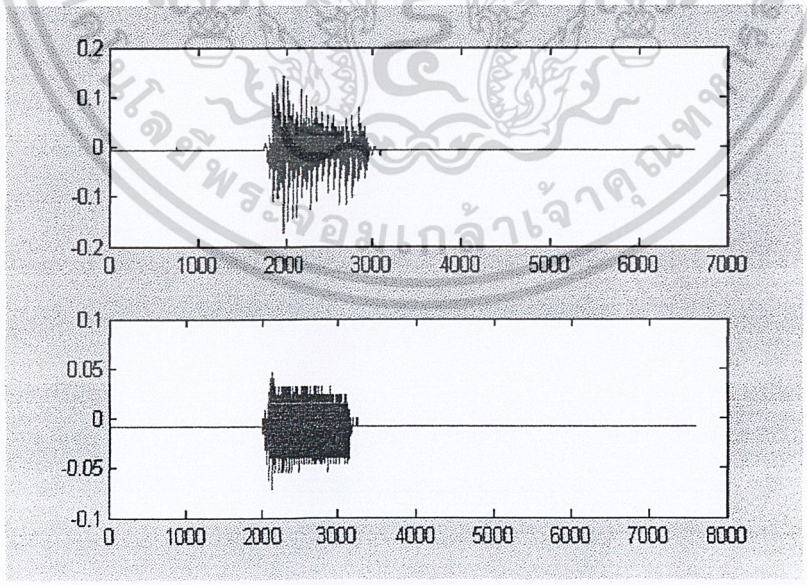
2) นำเสียงที่ทำการบันทึกมาทำการหาค่าพารามิเตอร์ที่ใช้เป็นตัวแทนของสัญญาณเสียง โดยวิธีหาค่าสัมประสิทธิ์แอลพีซี แล้วเปลี่ยนเป็นสัมประสิทธิ์เซปสตรัม
3) สร้างโค้ดบุคจากพารามิเตอร์เสียงทั้งหมดจากทุกกลุ่ม โดยจะได้เวกเตอร์ตัวแทนของเสียงที่ระบบสามารถรับรู้ได้ออกมาจำนวน 64 ค่า (โค้ดบุค 64)
4) นำพารามิเตอร์ของเสียงแต่ละคำมาเปรียบเทียบกับโค้ดบุคที่สร้างไว้ข้างต้น โดยแต่ละเสียงจะได้ค่าของ Index ออกมาจำนวน 25 ค่า เรียกว่าลำดับค่าปรากฏของเสียง
5) สร้างโมเดลของเสียง โดยการนำลำดับค่าปรากฏของแต่ละกลุ่มเสียงมาผ่านขั้นตอนการสร้างโมเดลด้วยเสียง ด้วยวิธี Hidden Markov Model (HMM) เมื่อทำจนครบทุกกลุ่มเสียงจะได้โมเดลเสียงทุกเสียงและพร้อมที่จะนำไปใช้งานในขั้นตอนการวิเคราะห์ต่อไป

4.1.2 ผลการทดลองในส่วนของการเรียนรู้

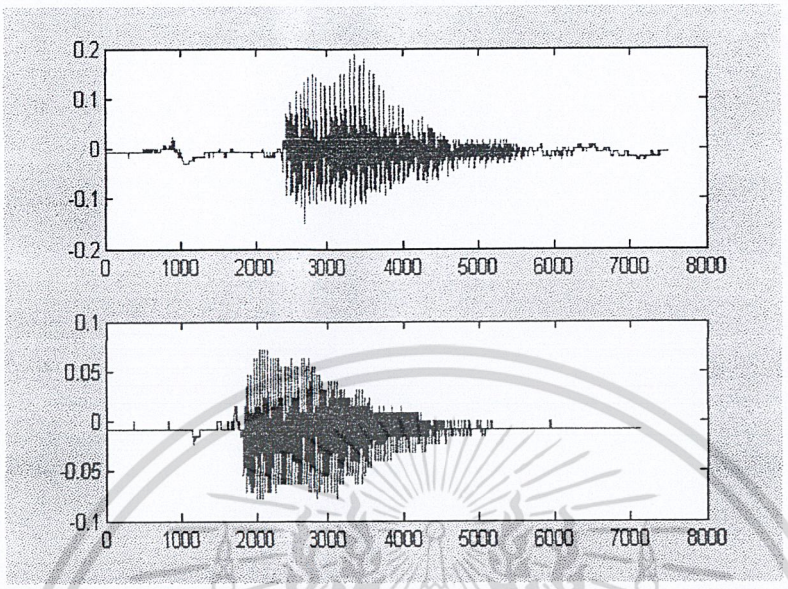
1) บันทึกเสียงที่จะใช้สร้างโค้ดบุคและโมเดลอ้างอิงจากกลุ่มตัวอย่างประกอบด้วย ผู้ชาย 5 คน ผู้หญิง 5 คน โดยมีรูปแบบการบันทึกเป็นโมโนสเตอริโอ PCM 8 บิต 8 กิโลเฮิรตซ์ แต่ละคนบันทึกเสียง "เปิด", "ปิด", "ไฟ", "หนึ่ง", "สอง" และ "สาม" คำละ 5 ครั้งรวมได้คำละ 50 เสียง



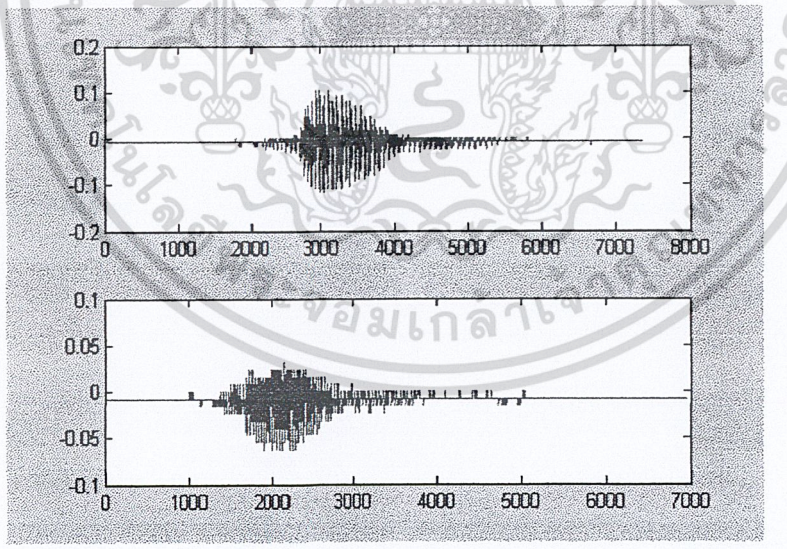
รูปที่ 4.1 (a) ตัวอย่างสัญญาณเสียง "เปิด" ที่บันทึกไว้จากกลุ่มตัวอย่างชาย
(b) ตัวอย่างสัญญาณเสียง "เปิด" ที่บันทึกไว้จากกลุ่มตัวอย่างหญิง



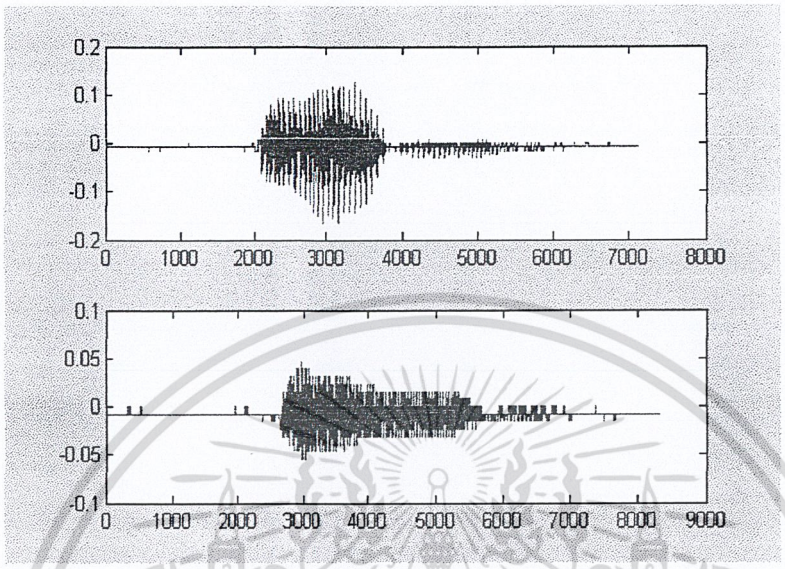
รูปที่ 4.2 (a) ตัวอย่างสัญญาณเสียง "ปิด" ที่บันทึกไว้จากกลุ่มตัวอย่างชาย
(b) ตัวอย่างสัญญาณเสียง "ปิด" ที่บันทึกไว้จากกลุ่มตัวอย่างหญิง



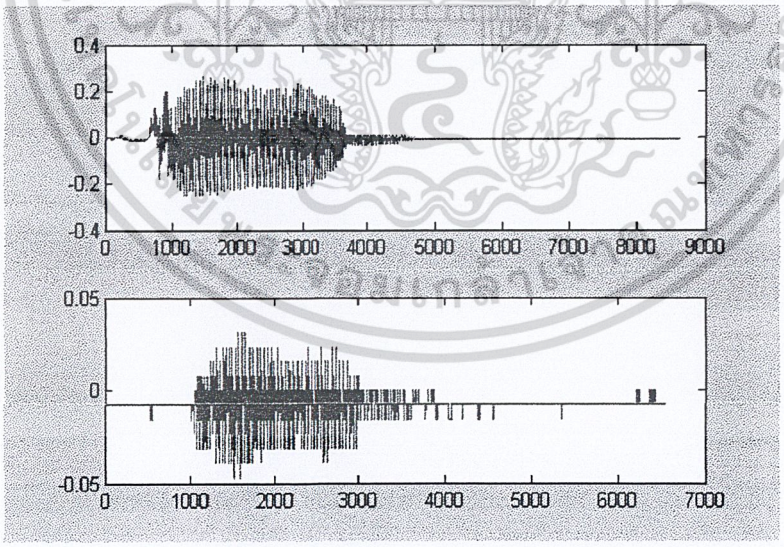
รูปที่ 4.3 (a) ตัวอย่างสัญญาณเสียง "ไฟ" ที่บันทึกไว้จากกลุ่มตัวอย่างชาย
(b) ตัวอย่างสัญญาณเสียง "ไฟ" ที่บันทึกไว้จากกลุ่มตัวอย่างหญิง



รูปที่ 4.4 (a) ตัวอย่างสัญญาณเสียง "หนึ่ง" ที่บันทึกไว้จากกลุ่มตัวอย่างชาย
(b) ตัวอย่างสัญญาณเสียง "หนึ่ง" ที่บันทึกไว้จากกลุ่มตัวอย่างหญิง

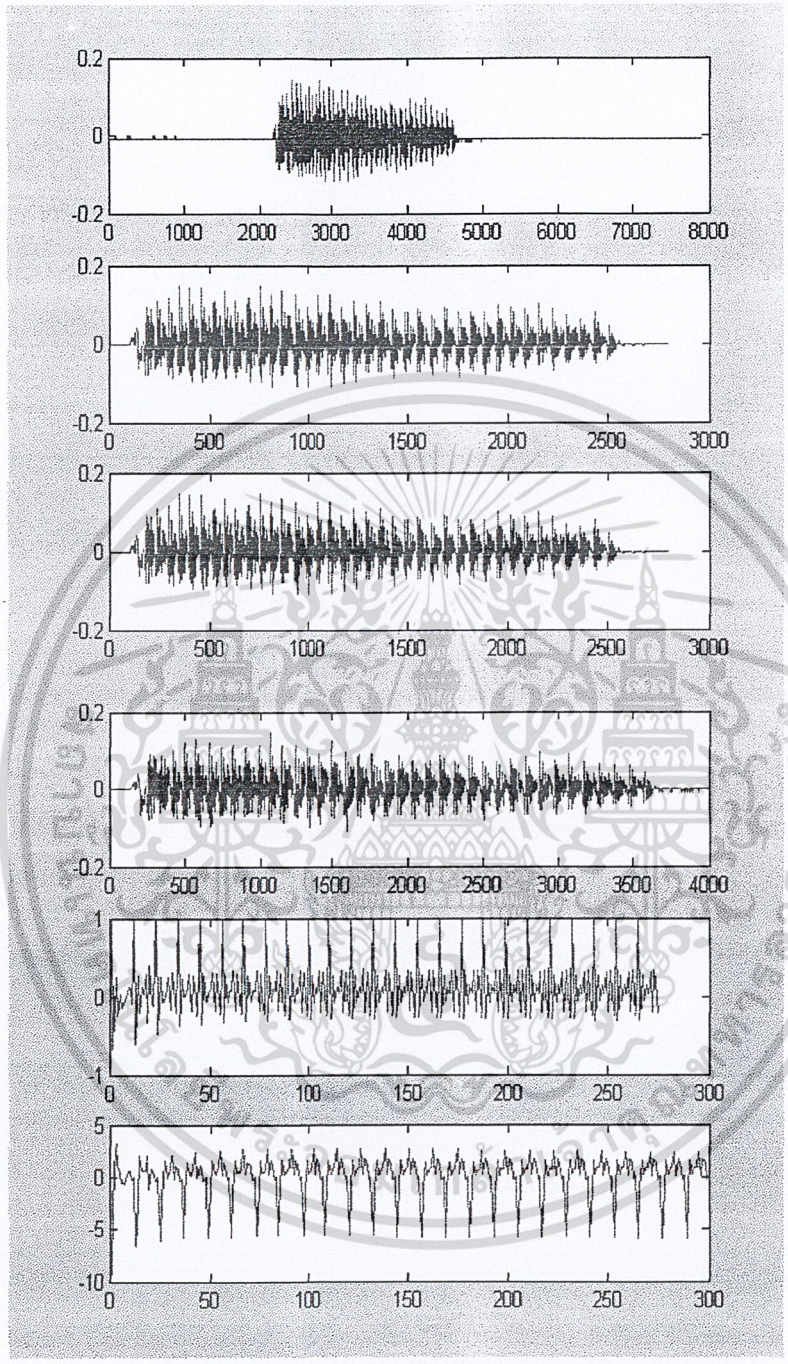


รูปที่ 4.5 (a) ตัวอย่างสัญญาณเสียง "สอง" ที่บันทึกไว้จากกลุ่มตัวอย่างชาย
(b) ตัวอย่างสัญญาณเสียง "สอง" ที่บันทึกไว้จากกลุ่มตัวอย่างหญิง

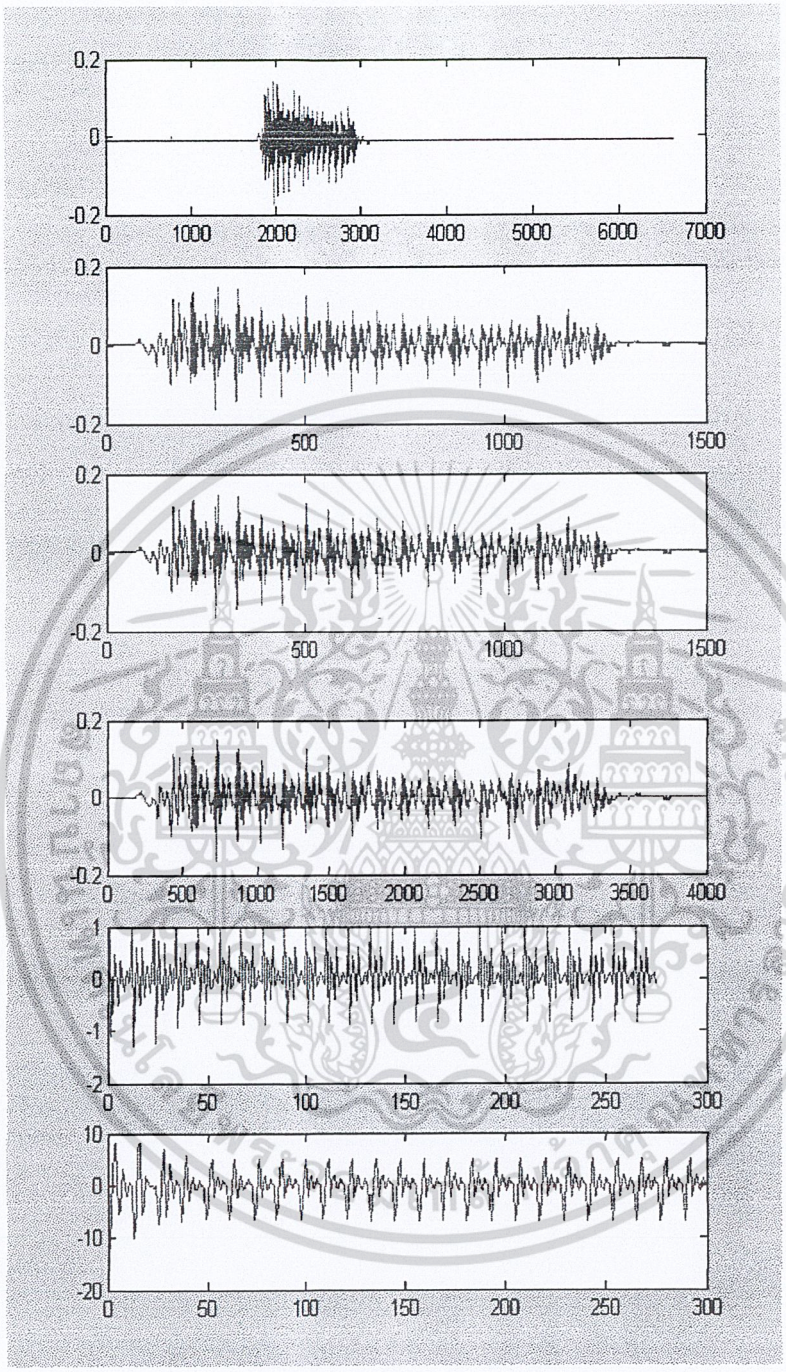


รูปที่ 4.6 (a) ตัวอย่างสัญญาณเสียง "สาม" ที่บันทึกไว้จากกลุ่มตัวอย่างชาย
(b) ตัวอย่างสัญญาณเสียง "สาม" ที่บันทึกไว้จากกลุ่มตัวอย่างหญิง

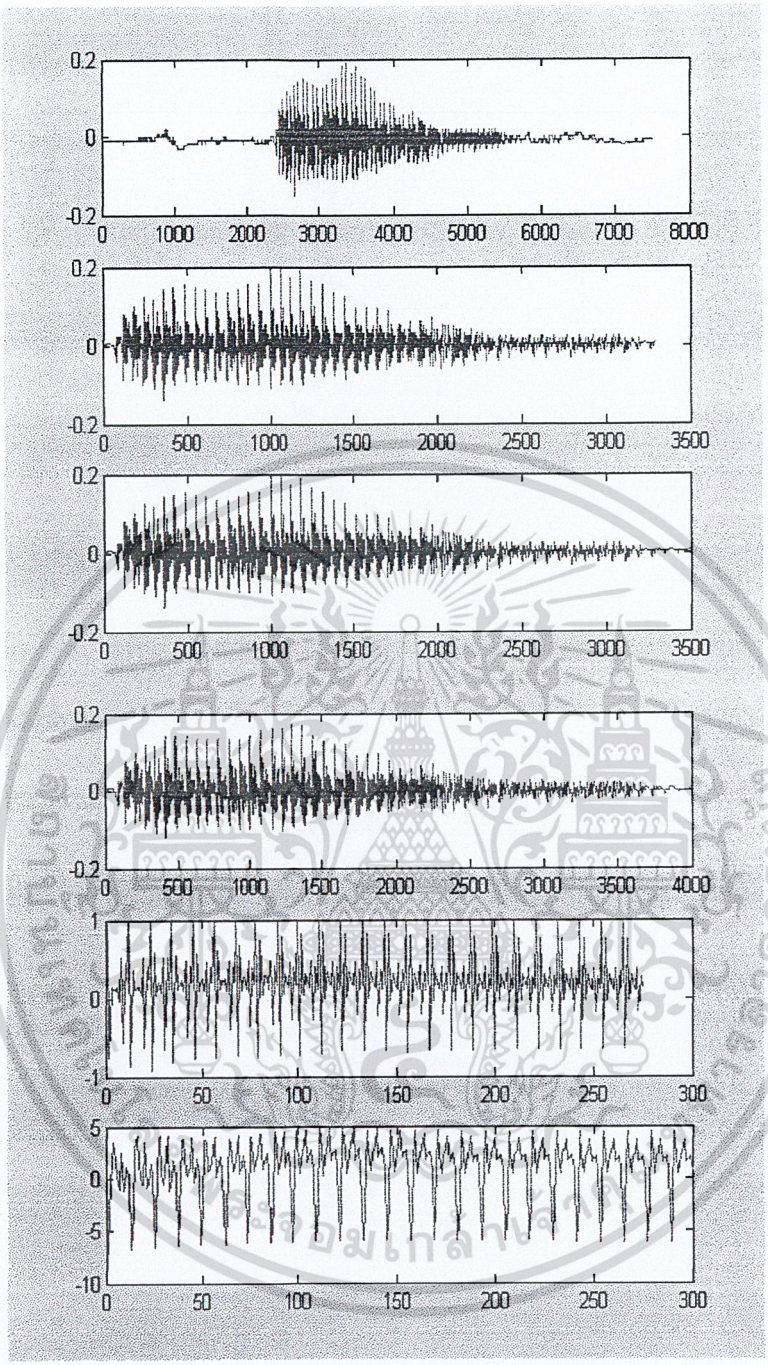
2) นำเสียงจากกลุ่มตัวอย่างทุกเสียงมาหาค่าสัมประสิทธิ์เซปสตรัมที่ใช้เป็นพารามิเตอร์ตัวแทนของสัญญาณเสียง



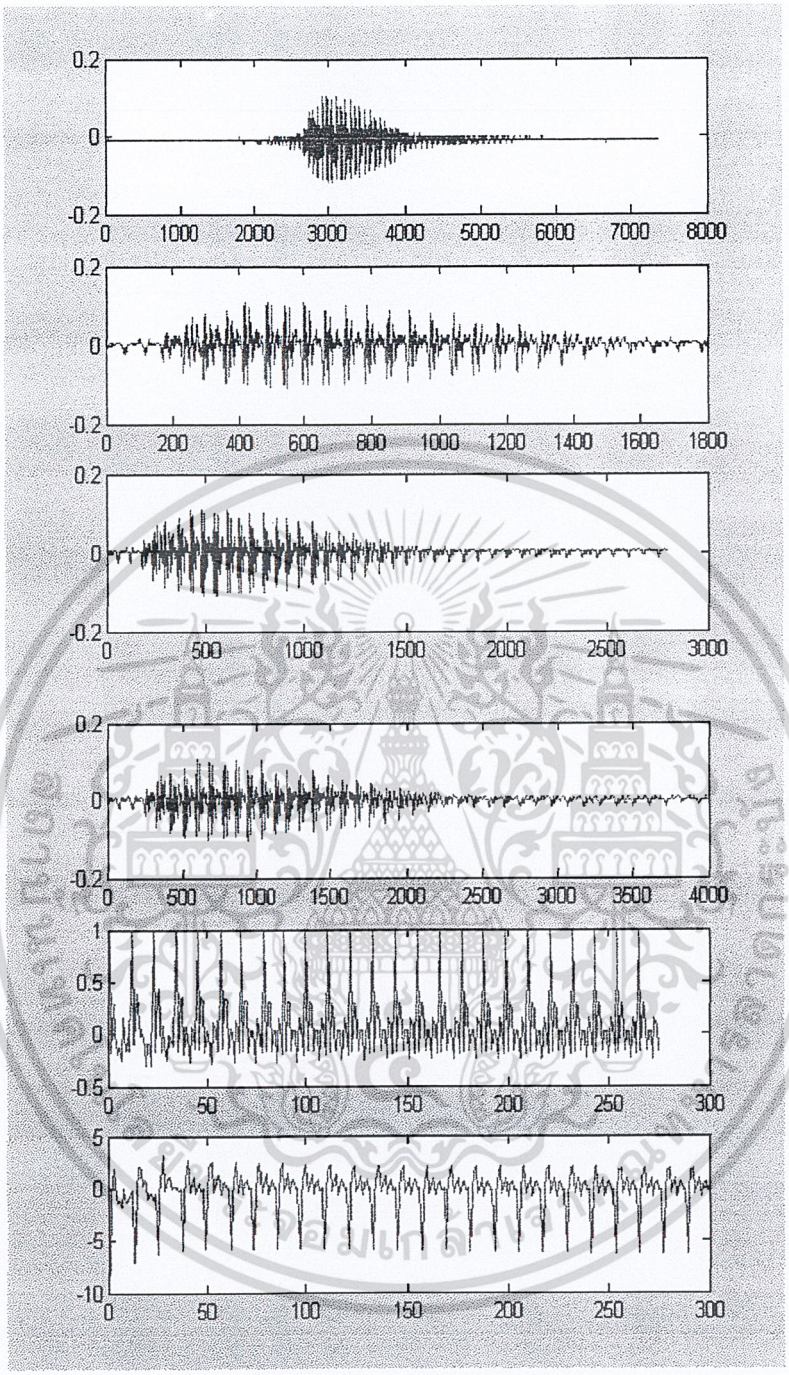
รูปที่ 4.7 (a) ตัวอย่างสัญญาณเสียง "เปิด" ที่บันทึกไว้จากกลุ่มตัวอย่าง
(b) การหาขอบเขตของคำว่า "เปิด" โดยการหาจากค่าพลังงาน
(c) การหาขอบเขตของคำว่า "เปิด" โดยการหาจากอัตราการตัดศูนย์
(d) การนอมอลไลซ์สัญญาณเสียง "เปิด" ที่หาขอบเขตแล้วให้มีจำนวนข้อมูล 4000 ค่า
(e) การหาสัมประสิทธิ์แอลพีซีจากตัวอย่างสัญญาณเสียง "เปิด" ที่นอมอลไลซ์แล้ว
(f) สัมประสิทธิ์เซปสตรัมที่คำนวณจากสัมประสิทธิ์แอลพีซีของเสียง "เปิด"



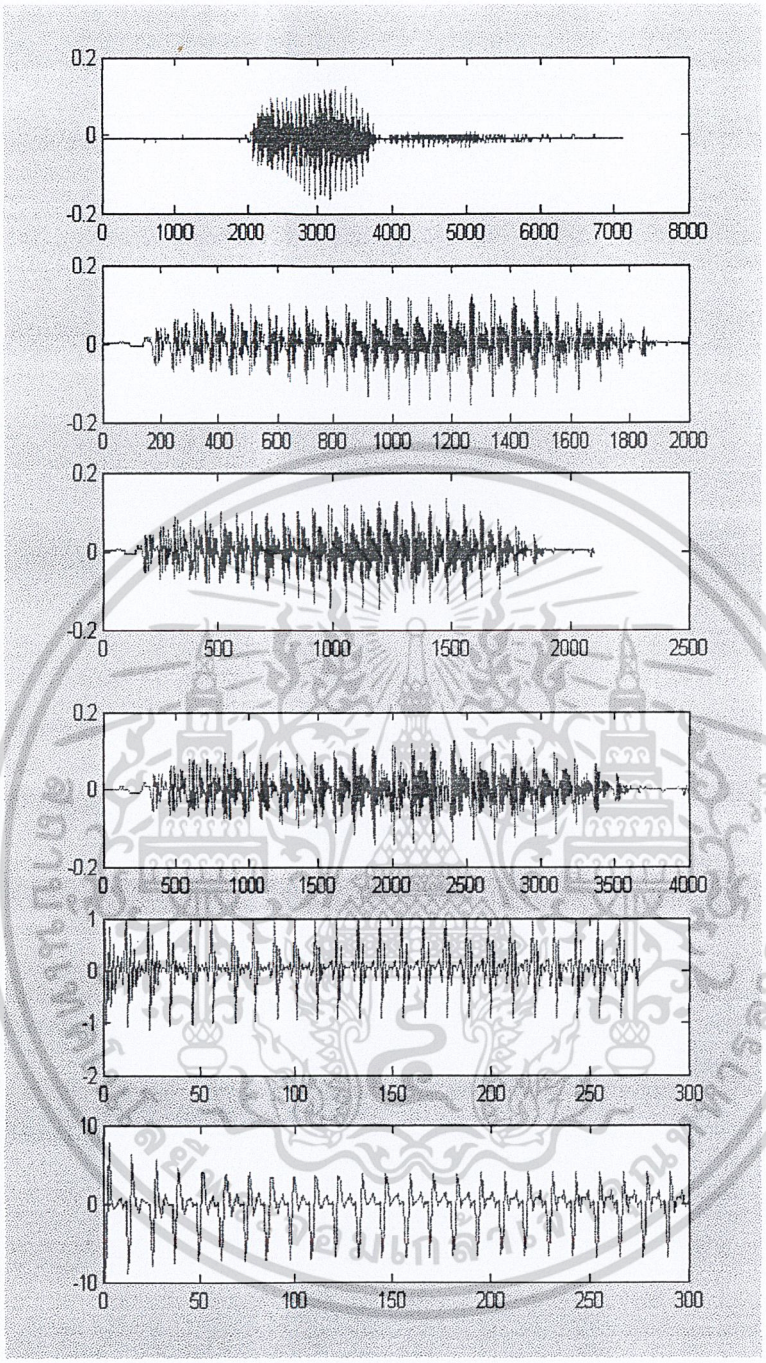
รูปที่ 4.8 (a) ตัวอย่างสัญญาณเสียง "ปิด" ที่บันทึกไว้จากกลุ่มตัวอย่าง
(b) การหาขอบเขตของคำว่า "ปิด" โดยการหาจากค่าพลังงาน
(c) การหาขอบเขตของคำว่า "ปิด" โดยการหาจากอัตราการตัดศูนย์
(d) การนอมอลไลซ์สัญญาณเสียง "ปิด" ที่หาขอบเขตแล้วให้มีจำนวนข้อมูล 4000 ค่า
(e) การหาสัมประสิทธิ์แอลพีซีจากตัวอย่างสัญญาณเสียง "ปิด" ที่นอมอลไลซ์แล้ว
(f) สัมประสิทธิ์เซปสตรัมที่คำนวณจากสัมประสิทธิ์แอลพีซีของเสียง "ปิด"



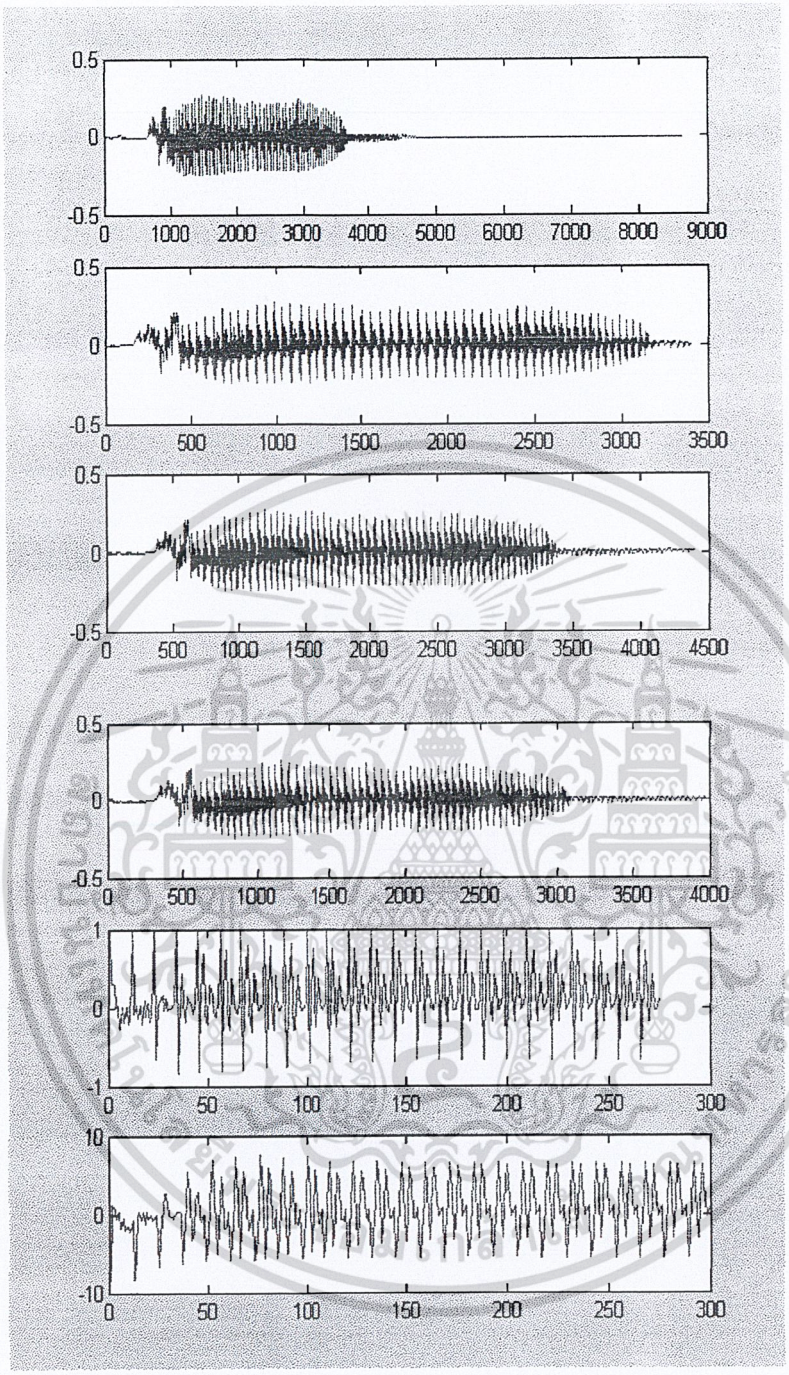
รูปที่ 4.9 (a) ตัวอย่างสัญญาณเสียง "ไฟ" ที่บันทึกไว้จากกลุ่มตัวอย่าง
(b) การหาขอบเขตของคำว่า "ไฟ" โดยการหาจากค่าพลังงาน
(c) การหาขอบเขตของคำว่า "ไฟ" โดยการหาจากอัตราการตัดศูนย์
(d) การนอมอลไลซ์สัญญาณเสียง "ไฟ" ที่หาขอบเขตแล้วให้มีจำนวนข้อมูล 4000 ค่า
(e) การหาสัมประสิทธิ์แอลพีซีจากตัวอย่างสัญญาณเสียง "ไฟ" ที่นอมอลไลซ์แล้ว
(f) สัมประสิทธิ์เซปสตรัมที่คำนวณจากสัมประสิทธิ์แอลพีซีของเสียง "ไฟ"



รูปที่ 4.10 (a) ตัวอย่างสัญญาณเสียง “หนึ่ง” ที่บันทึกไว้จากกลุ่มตัวอย่าง
(b) การหาขอบเขตของคำว่า “หนึ่ง” โดยการหาจากค่าพลังงาน
(c) การหาขอบเขตของคำว่า “หนึ่ง” โดยการหาจากอัตราการตัดศูนย์
(d) การนอมอลไลซ์สัญญาณเสียง “หนึ่ง” ที่หาขอบเขตแล้วให้มีจำนวนข้อมูล 4000 ค่า
(e) การหาสัมประสิทธิ์แอลพีซีจากตัวอย่างสัญญาณเสียง “หนึ่ง” ที่นอมอลไลซ์แล้ว
(f) สัมประสิทธิ์เซปสตรัมที่คำนวณจากสัมประสิทธิ์แอลพีซีของเสียง “หนึ่ง”

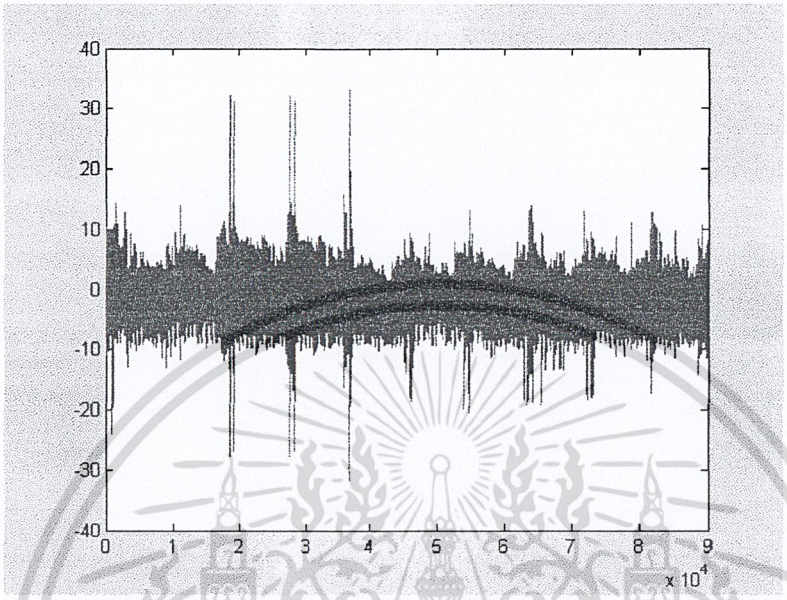


รูปที่ 4.11 (a) ตัวอย่างสัญญาณเสียง “สอง” ที่บันทึกไว้จากกลุ่มตัวอย่าง
(b) การหาขอบเขตของคำว่า “สอง” โดยการหาจากค่าพลังงาน
(c) การหาขอบเขตของคำว่า “สอง” โดยการหาจากอัตราการตัดศูนย์
(d) การนอมอลไลซ์สัญญาณเสียง “สอง” ที่หาขอบเขตแล้วให้มีจำนวนข้อมูล 4000 ค่า
(e) การหาสัมประสิทธิ์แอลพีซีจากตัวอย่างสัญญาณเสียง “สอง” ที่นอมอลไลซ์แล้ว
(f) สัมประสิทธิ์เซปสตรัมที่คำนวณจากสัมประสิทธิ์แอลพีซีของเสียง “สอง”

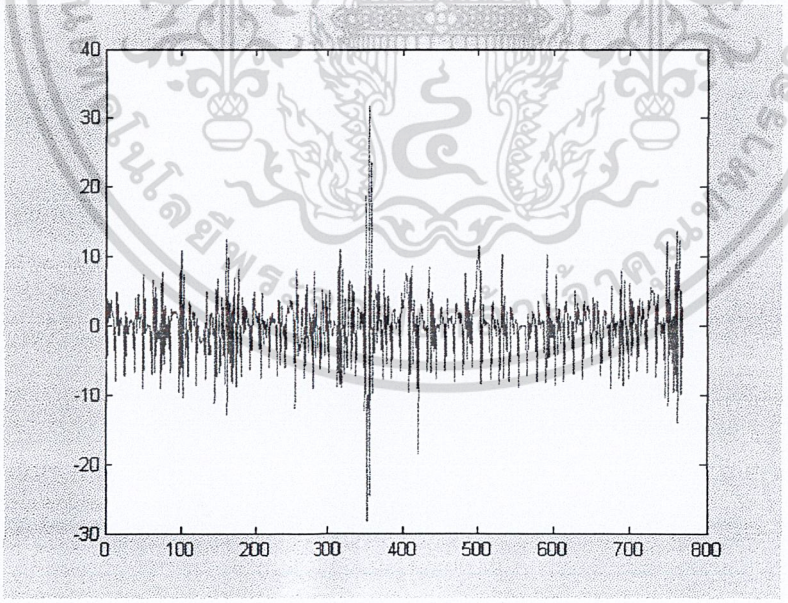


รูปที่ 4.12 (a) ตัวอย่างสัญญาณเสียง “สาม” ที่บันทึกไว้จากกลุ่มตัวอย่าง
(b) การหาขอบเขตของคำว่า “สาม” โดยการหาจากค่าพลังงาน
(c) การหาขอบเขตของคำว่า “สาม” โดยการหาจากอัตราการตัดศูนย์
(d) การนอมอลไลซ์สัญญาณเสียง “สาม” ที่หาขอบเขตแล้วให้มีจำนวนข้อมูล 4000 ค่า
(e) การหาสัมประสิทธิ์แอลพีซีจากตัวอย่างสัญญาณเสียง “สาม” ที่นอมอลไลซ์แล้ว
(f) สัมประสิทธิ์เซปสตรัมที่คำนวณจากสัมประสิทธิ์แอลพีซีของเสียง “สาม”

3) นำพารามิเตอร์ตัวแทนเสียงของเสียงทุกเสียงที่บันทึกไว้มาเรียงต่อกันเป็นเทรนนิ่งเซต และนำเทรนนิ่งเซตที่ได้ไปคำนวณกลุ่มของข้อมูลจำนวน 64 กลุ่ม กลุ่มละ 12 ค่า ที่ใช้เป็นตัวแทนของข้อมูลทั้งหมด (โค้ดบุค 64)

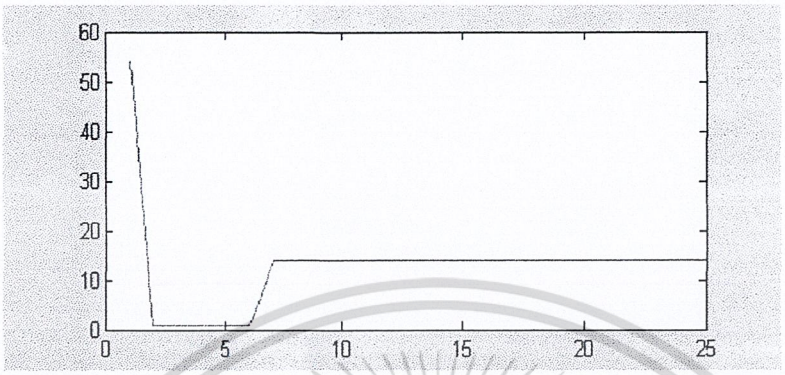


รูปที่ 4.13 แสดงเทรนนิ่งเซตที่ได้จากเสียงที่บันทึกจากกลุ่มตัวอย่าง

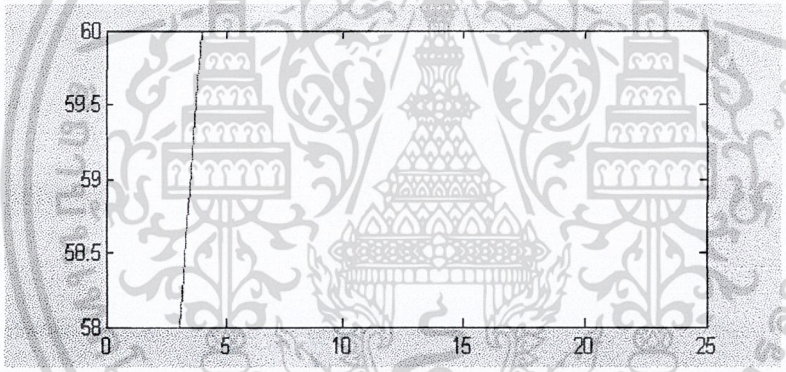


รูปที่ 4.14 แสดงโค้ดบุค 64 ที่คำนวณได้จากเทรนนิ่งเซต

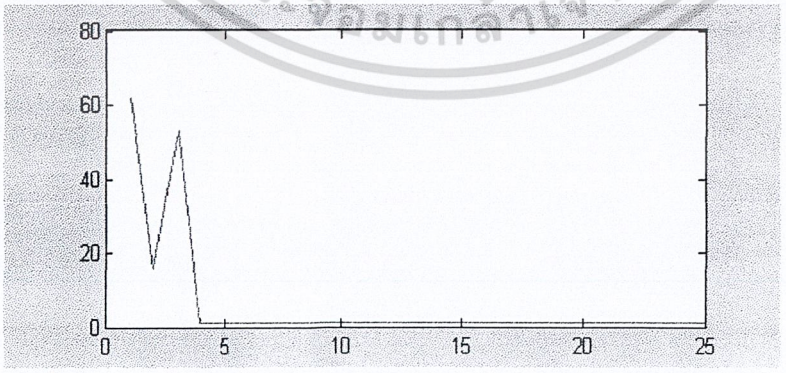
4) นำพารามิเตอร์ตัวแทนของเสียงที่บันทึกไว้ทุกคำมาเปรียบเทียบกับโค้ดบุคที่สร้างไว้โดยวิธี Vector Quantization (VQ) ซึ่งจะได้ Index ออกมาจำนวน 25 ค่า ในที่นี้คือลำดับค่าปรากฏของแต่ละเสียงออกมา



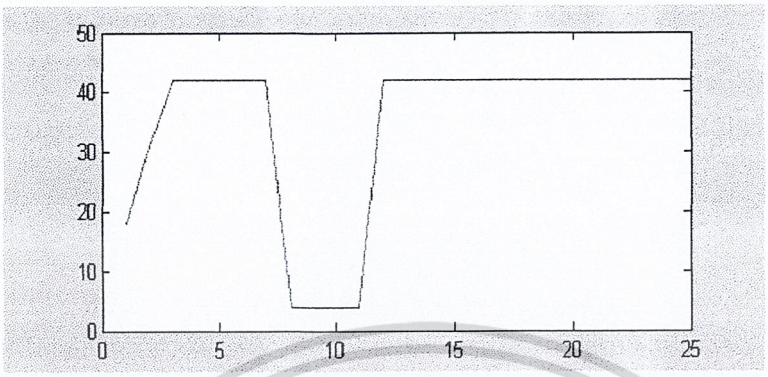
รูปที่ 4.15 แสดงตัวอย่าง ลำดับค่าปรากฏของสียง "เปิด" ที่บันทึกไว้จากกลุ่มตัวอย่าง



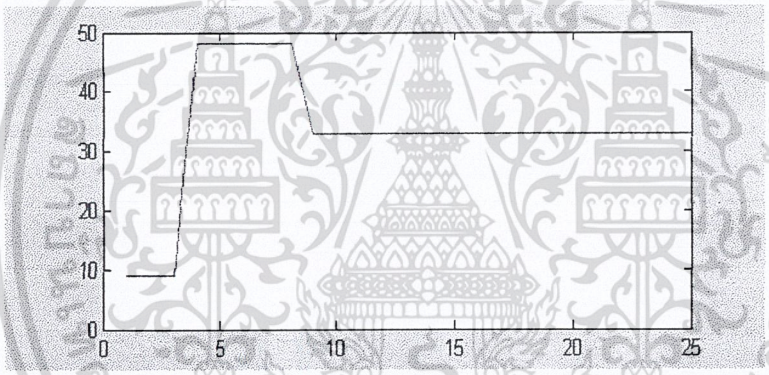
รูปที่ 4.16 แสดงตัวอย่างลำดับค่าปรากฏของสียง "ปิด" ที่บันทึกไว้จากกลุ่มตัวอย่าง



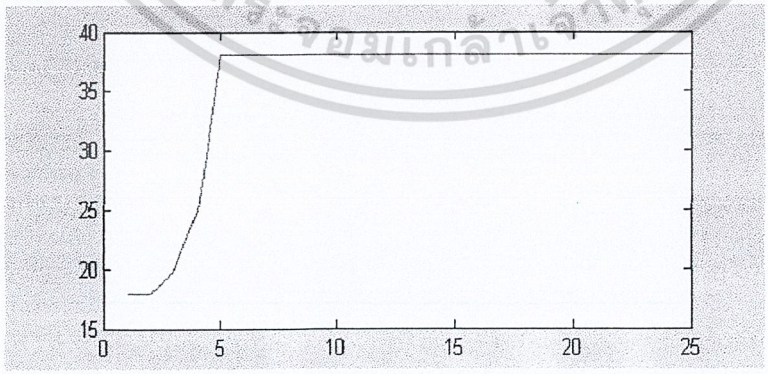
รูปที่ 4.17 แสดงตัวอย่างลำดับค่าปรากฏของสียง "ไฟ" ที่บันทึกไว้จากกลุ่มตัวอย่าง



รูปที่ 4.18 แสดงตัวอย่างลำดับค่าปรากฏของสียง "หนึ่ง" ที่บันทึกไว้จากกลุ่มตัวอย่าง



รูปที่ 4.19 แสดงตัวอย่างลำดับค่าปรากฏของสียง "สอง" ที่บันทึกไว้จากกลุ่มตัวอย่าง



รูปที่ 4.20 แสดงตัวอย่างลำดับค่าปรากฏของสียง "สาม" ที่บันทึกไว้จากกลุ่มตัวอย่าง

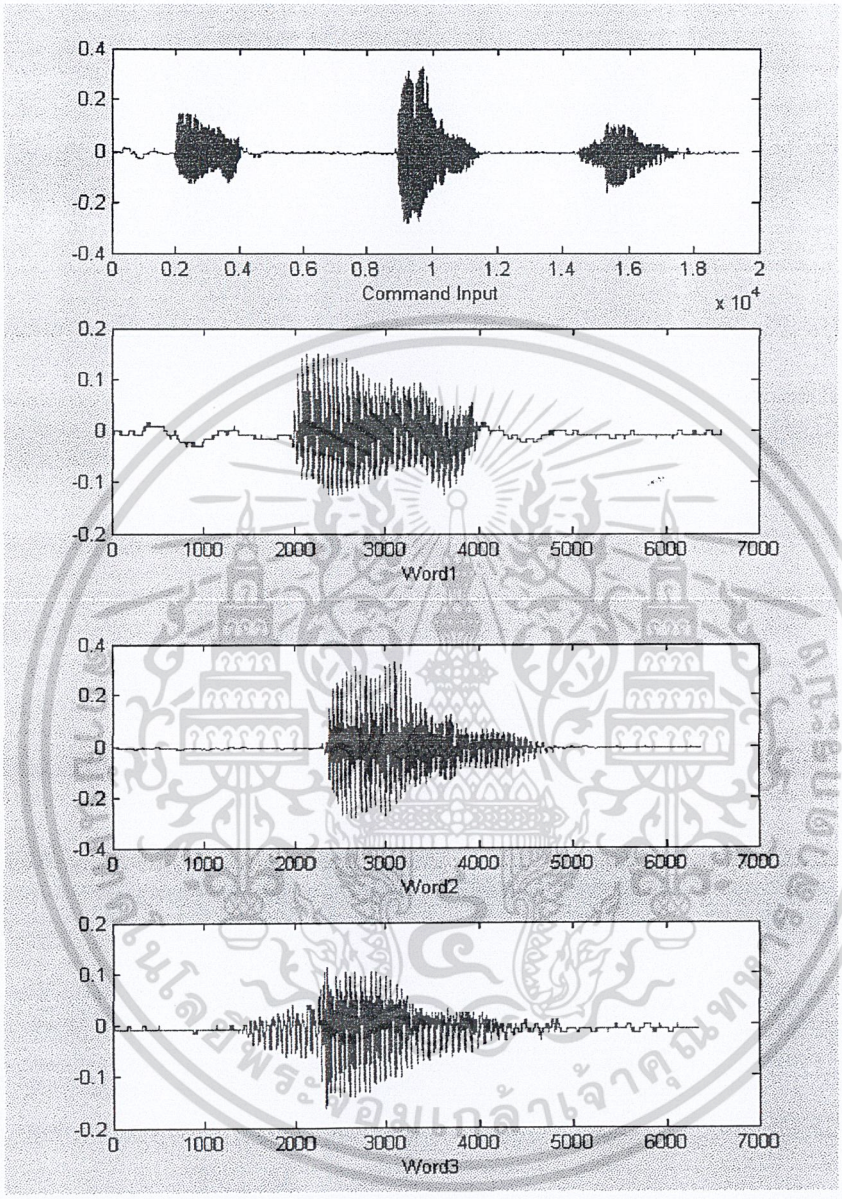
4.1.3 ขั้นตอนการวิเคราะห์

เป็นขั้นตอนที่จะทำการวิเคราะห์และรู้จำเสียงคำสั่งที่ไม่ทราบว่าเป็นคำใด โดยจะทำการแยกเสียงคำสั่งออกเป็น 3 พยางค์ นำเสียงไปทำการวิเคราะห์และรู้จำว่าเป็นเสียงใดทีละ 1 พยางค์ จากนั้นทำการเปรียบเทียบกับเสียงคำสั่งว่าเหมือนหรือต่างกันอย่างไร โดยมีขั้นตอนดังนี้

1) นำสัญญาณเสียงคำสั่งมาทำการแยกออกเป็นคำเดี่ยว 3 พยางค์
2) นำเสียงคำสั่งพยางค์แรก มาผ่านขั้นตอนการหาสัมประสิทธิ์แอลซีพี และหาค่า Weight จะได้พารามิเตอร์ของเสียงนั้น และนำพารามิเตอร์มาเปรียบเทียบกับ Codebook จะได้ Codebook Index
3) นำ Codebook Index มาหาค่าความน่าจะเป็นกับทุกโมเดลที่เก็บไว้ทั้งหมด โดยวิธี Viterbi Algolithm ซึ่งจะได้ค่าความน่าจะเป็นของแต่ละโมเดลออกมา ค่าความน่าจะเป็นเมื่อเปรียบเทียบกับโมเดลใดมีค่าสูงที่สุด ผลจะได้ว่าเสียงที่นำมาทดสอบตรงกับโมเดลนั้น แต่ถ้าค่าความน่าจะเป็นที่ได้มีค่าเป็นศูนย์ จะให้โปรแกรมบอกว่ามีความผิดพลาดเกิดขึ้น
4) ทำการวิเคราะห์และรู้จำเสียงของเสียงคำสั่งพยางค์ที่ 2 และ 3 ตามลำดับโดยทำซ้ำกระบวนการที่สองและสาม
5) เมื่อทำการรู้จำเสียงคำสั่งครบทั้ง 3 พยางค์แล้ว เปรียบเทียบผลจากการรู้จำกับเสียงคำสั่งและหาค่าเปอร์เซ็นต์ความถูกต้องออกมา

4.1.4 ผลการทดลองในส่วนของขั้นตอนการวิเคราะห์

แสดงสัญญาณเสียงคำสั่งขนาดความยาว 3 พยางค์



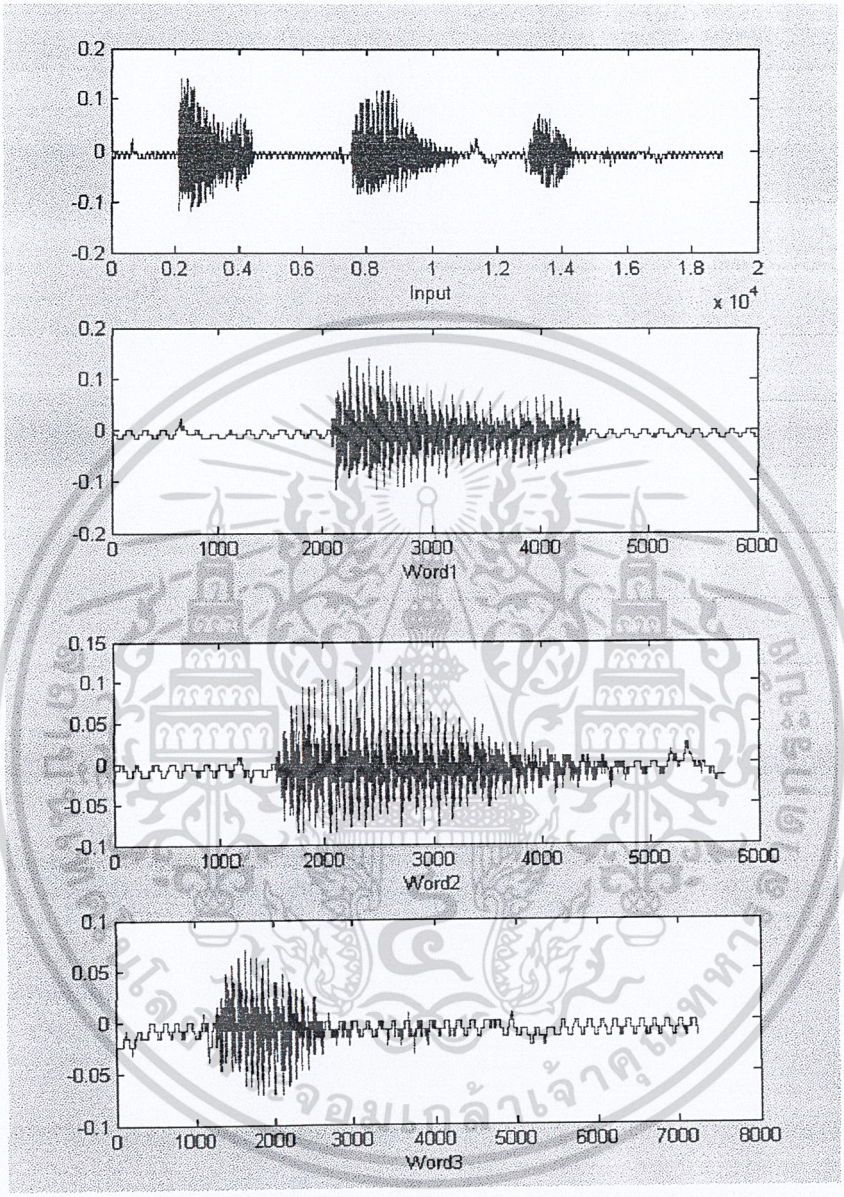
(a)

(b)

(c)

(d)

รูปที่ 4.21 (a) แสดงสัญญาณเสียงของเสียงคำสั่ง เปิด-ไฟ-หนึ่ง
(b) แสดงสัญญาณสียงของคำว่า เปิด
(c) แสดงสัญญาณสียงของคำว่า ไฟ
(d) แสดงสัญญาณสียงของคำว่า หนึ่ง

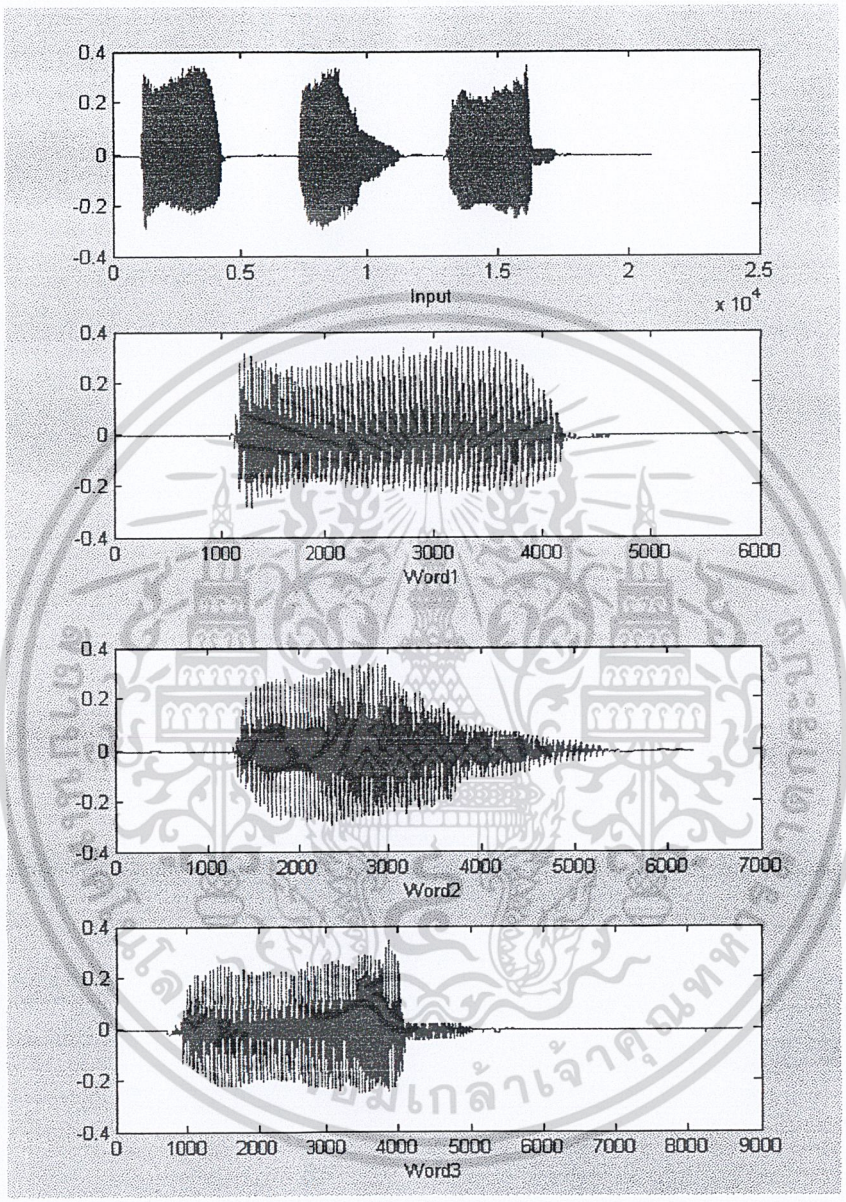


รูปที่ 4.22 (a) แสดงสัญญาณเสียงของเสียงคำสั่ง เปิด-ไฟ-สอง

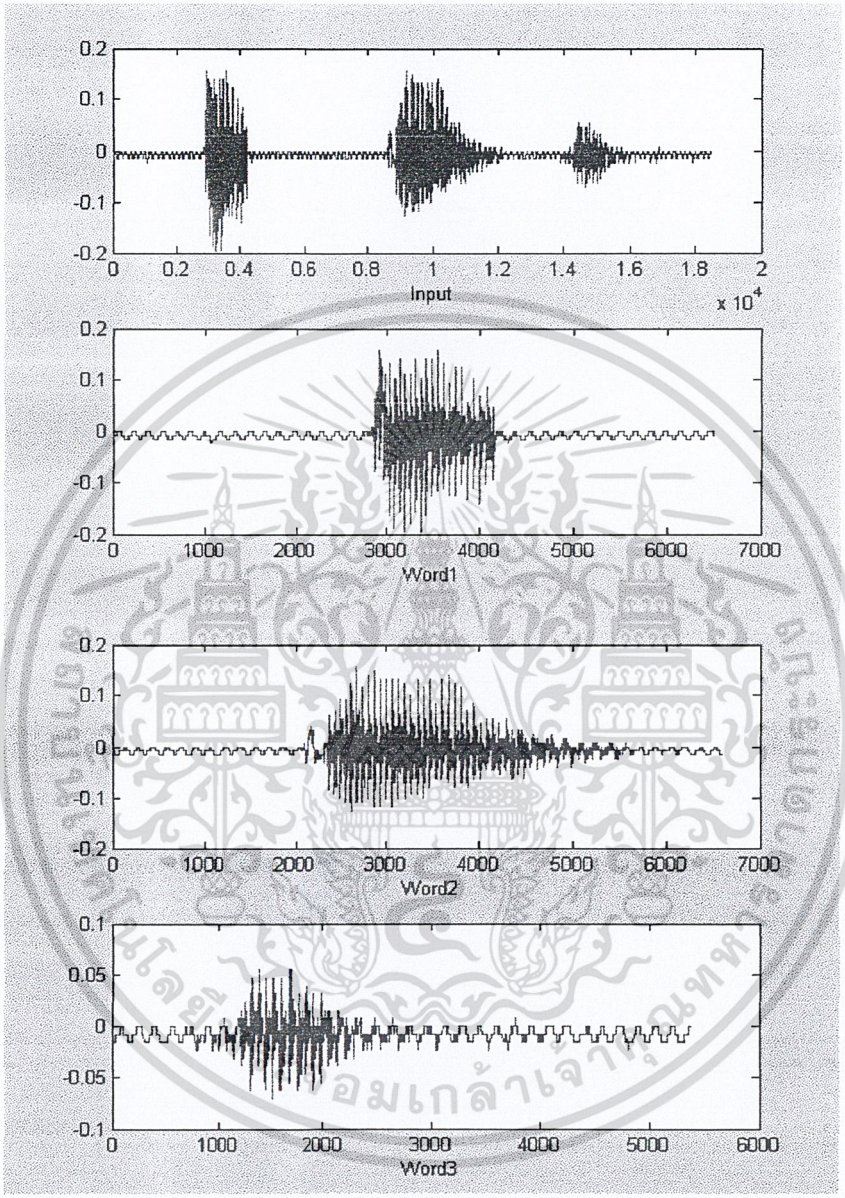
(b) แสดงสัญญาณสียงของคำว่า เปิด

(c) แสดงสัญญาณสียงของคำว่า ไฟ

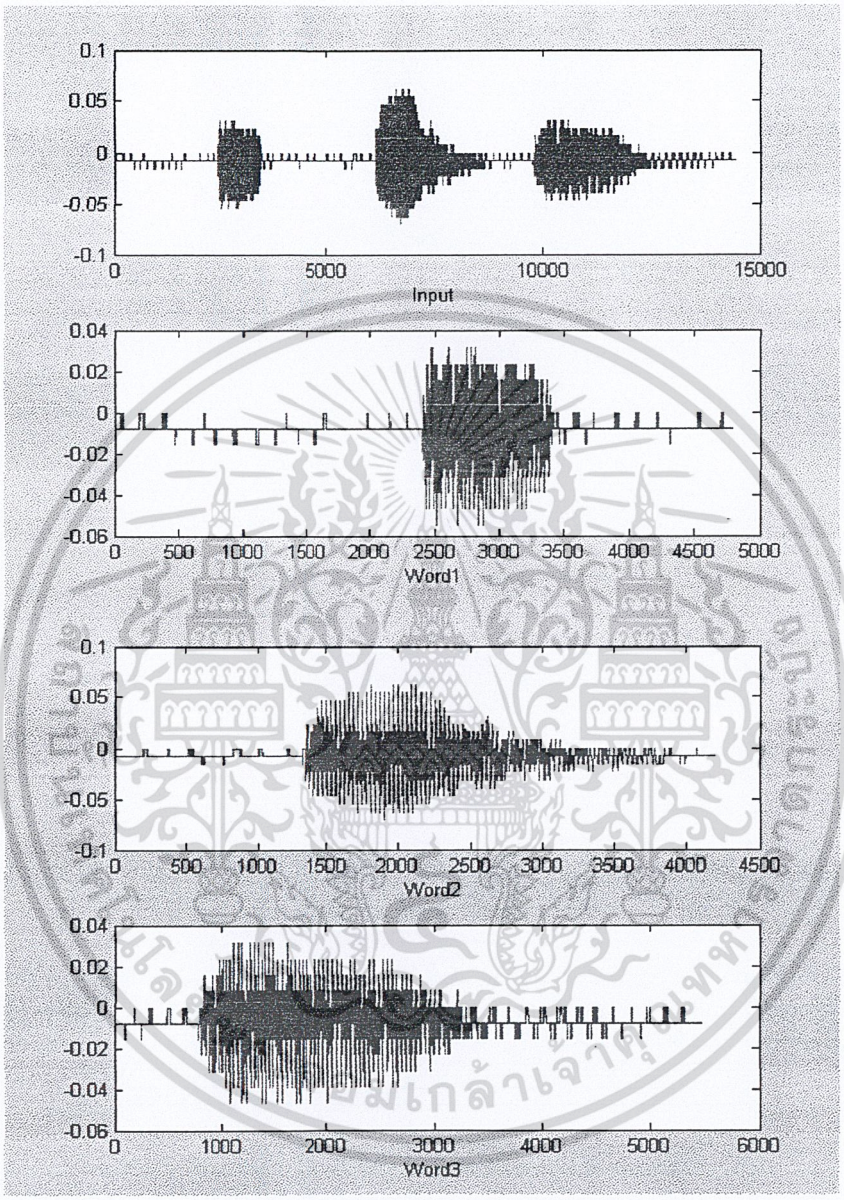
(d) แสดงสัญญาณสียงของคำว่า สอง



รูปที่ 4.23 (a) แสดงสัญญาณเสียงของเสียงคำสั่ง เปิด-ไฟ-สาม
(b) แสดงสัญญาณสียงของคำว่า เปิด
(c) แสดงสัญญาณสียงของคำว่า ไฟ
(d) แสดงสัญญาณสียงของคำว่า สาม



รูปที่ 4.24 (a) แสดงสัญญาณเสียงของเสียงคำสั่ง ปิด-ไฟ-หนึ่ง
(b) แสดงสัญญาณสียงของคำว่า ปิด
(c) แสดงสัญญาณสียงของคำว่า ไฟ
(d) แสดงสัญญาณสียงของคำว่า หนึ่ง

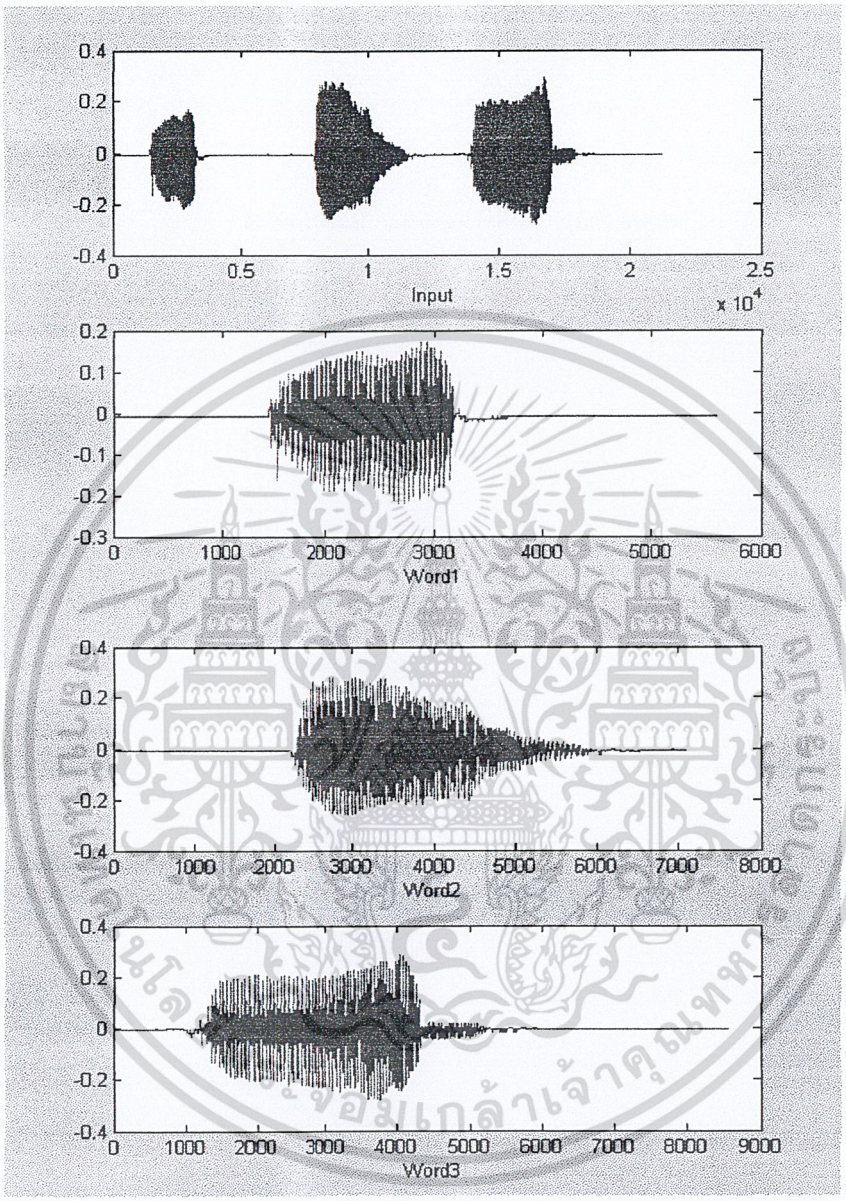


รูปที่ 4.25 (a) แสดงสัญญาณเสียงของเสียงคำสั่ง ปิด-ไฟ-สอง

(b) แสดงสัญญาณสียงของคำว่า ปิด

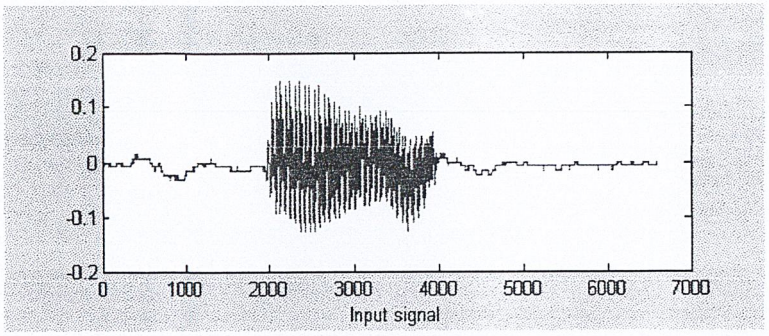
(c) แสดงสัญญาณสียงของคำว่า ไฟ

(d) แสดงสัญญาณสียงของคำว่า สอง

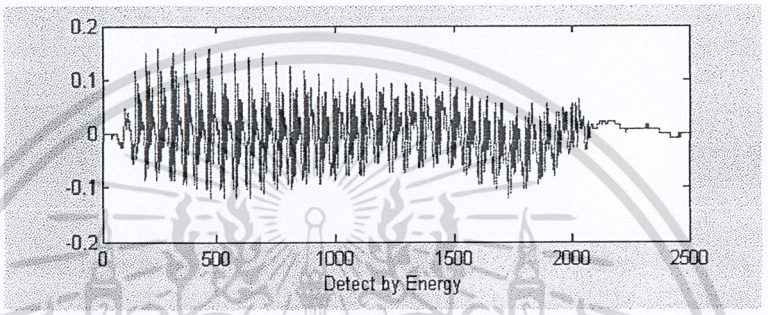


รูปที่ 4.26 (a) แสดงสัญญาณเสียงของเสียงคำสั่ง ปิด-ไฟ-สาม
(b) แสดงสัญญาณเสียงของคำว่า ปิด
(c) แสดงสัญญาณเสียงของคำว่า ไฟ
(d) แสดงสัญญาณเสียงของคำว่า สาม

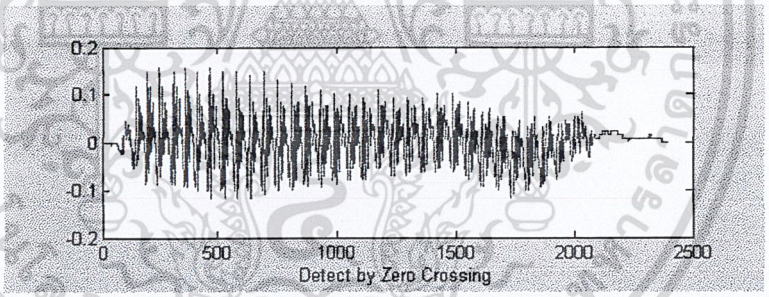
แสดงสัญญาณเสียงของเสียงคำสั่งแต่ละพยางค์ และการหาค่าพารามิเตอร์ต่างๆ



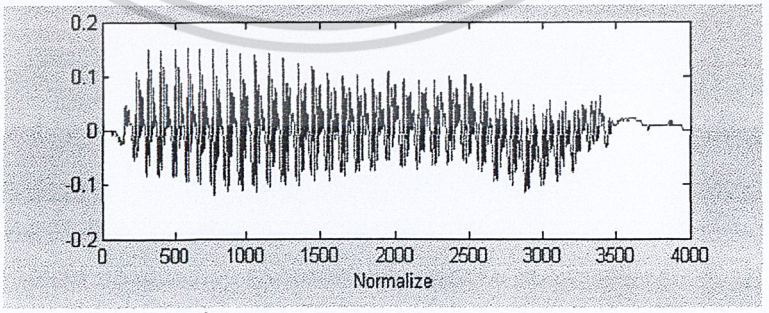
-กราฟแสดงข้อมูลของไฟล์เสียงคำว่า “เปิด”



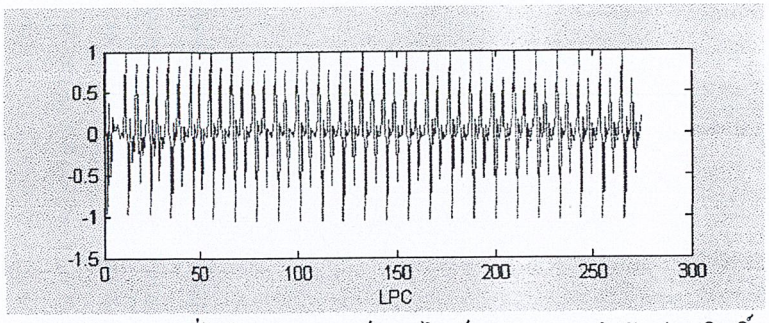
-ทำการหาขอบเขตของคำจากค่าพลังงานของเฟรมอ้างอิง โดยเฟรมดังกล่าวมีค่าพลังงาน 0.2 เท่าของเฟรมที่มีค่าพลังงานสูงสุด



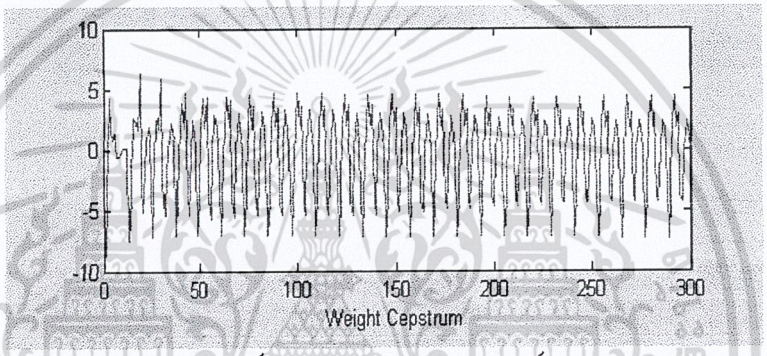
-ทำการหาขอบเขตของคำจากอัตราการตัดศูนย์เป็นค่าอ้างอิง โดยขอบเขตของข้อมูลมีค่าอัตราการตัดศูนย์เป็น 0.2 เท่าของเฟรมที่มีอัตราการตัดศูนย์สูงสุด



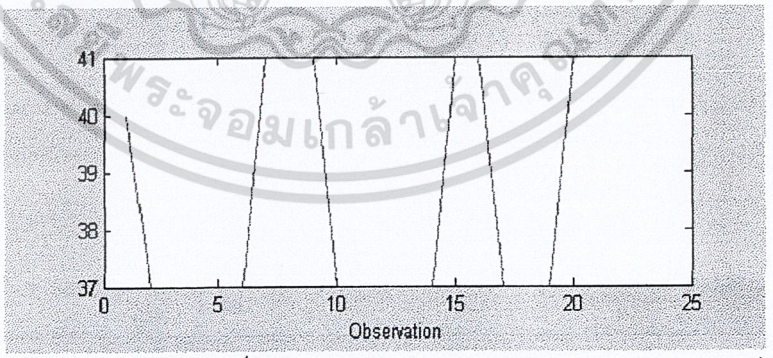
-นำสัญญาณที่หาขอบเขตแล้วมานอร์มอลไลซ์ให้มีจำนวนข้อมูล 4000 ค่า



-สัญญาณที่ได้จากการนอร์มอลไลซ์จะนำมาหาค่าสัมประสิทธิ์แอลพีซี โดยสัมประสิทธิ์แอลพีซี 10 ค่า หาได้จากสัญญาณ 240 ค่า โดยทำการเลื่อนเพื่อหาสัมประสิทธิ์ทีละ 160 ค่า จนครบทั้งหมด

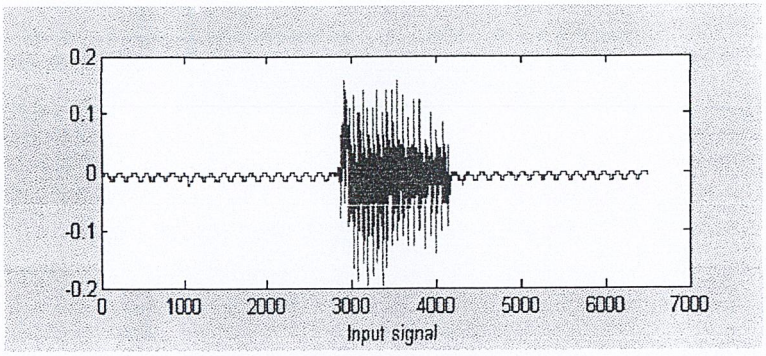


-หาค่าสัมประสิทธิ์เซปสตรัม โดยสัมประสิทธิ์ แอลพีซี 10 ค่า หาค่าสัมประสิทธิ์เซปสตรัมได้ 12 ค่า ดังนั้นจะสามารถหาค่าสัมประสิทธิ์เซปสตรัมได้ 300 ค่า จากสัมประสิทธิ์แอลพีซี 250 ค่า จากนั้นนำสัมประสิทธิ์เซปสตรัมที่ได้มาถ่วงน้ำหนักเพื่อลดความคลาดเคลื่อน

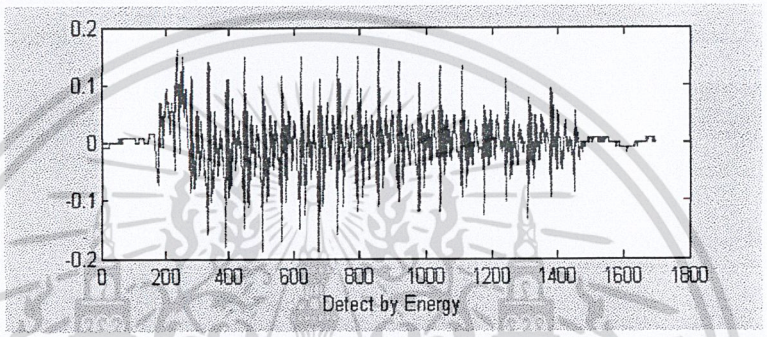


-นำสัมประสิทธิ์ เซปสตรัมทีละ 12 ค่า ไปผ่านกระบวนการเวกเตอร์ควอนไตเซชั่นจะได้ตัวแทนของข้อมูล (เวกเตอร์อินเดกซ์) 1 ค่า ดังนั้นจะได้ตัวแทนของข้อมูลออกมาทั้งหมด 25 ค่า ตัวแทนข้อมูลที่ได้ทั้งหมดจะใช้เป็นลำดับค่าปรากฏ (Observation) ในการรู้จำเสียงต่อไป

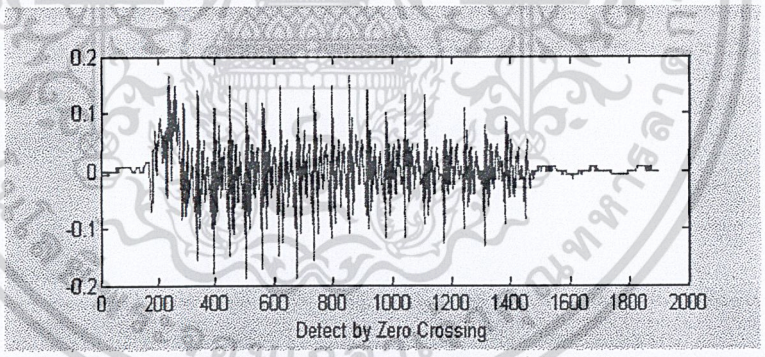
รูปที่ 4.27 แสดงเสียงคำว่า เปิด และการหาค่าพารามิเตอร์ต่างๆ ของเสียง



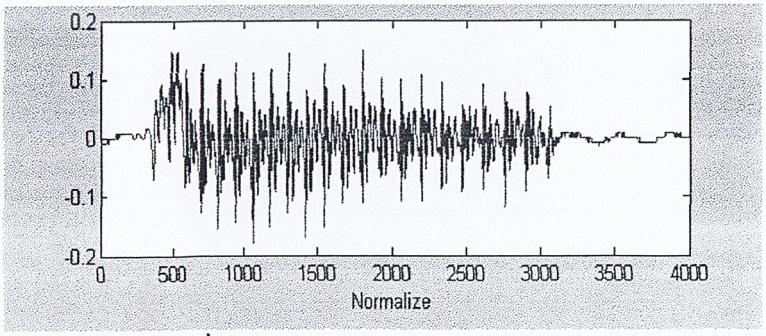
-กราฟแสดงข้อมูลของไฟล์เสียงคำว่า "ปิด"



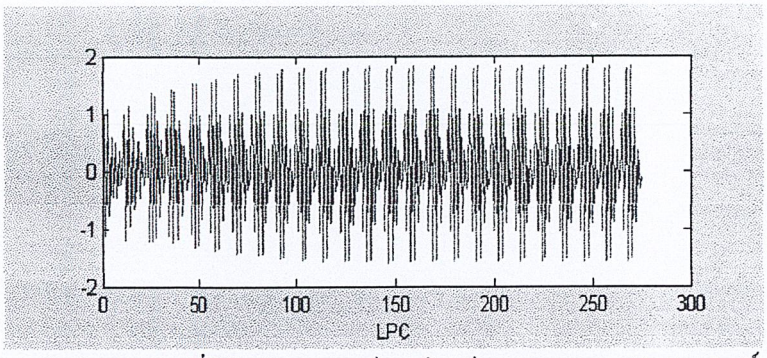
-ทำการหาขอบเขตของคำจากค่าพลังงานของเฟรมอ้างอิง โดยเฟรมดังกล่าวมีค่าพลังงาน 0.2 เท่าของเฟรมที่มีค่าพลังงานสูงสุด



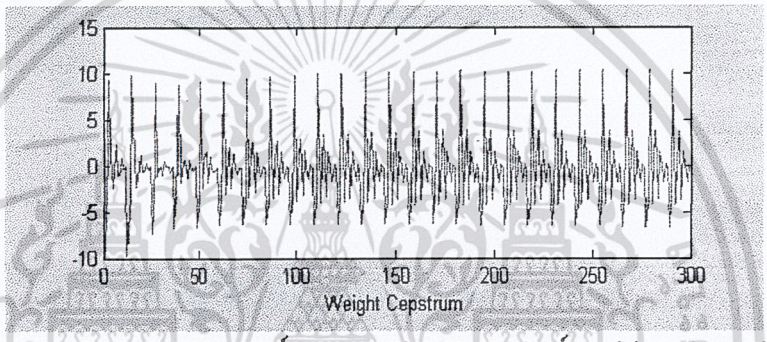
-ทำการหาขอบเขตของคำจากอัตราการตัดศูนย์เป็นค่าอ้างอิง โดยขอบเขตของข้อมูลมีค่าอัตราการตัดศูนย์เป็น 0.2 เท่าของเฟรมที่มีอัตราการตัดศูนย์สูงสุด



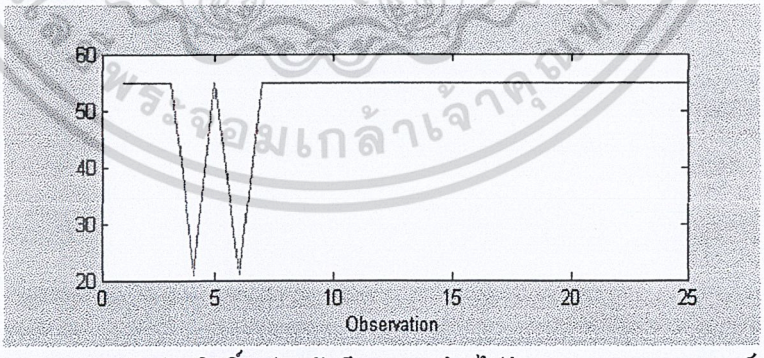
-นำสัญญาณที่หาขอบเขตแล้วมานอร์มอลไลซ์ให้มีจำนวนข้อมูล 4000 ค่า



-สัญญาณที่ได้จากการนอร์มอลไลซ์จะนำมาหาค่าสัมประสิทธิ์แอลพีซี โดยสัมประสิทธิ์แอลพีซี 10 ค่า หาได้จากสัญญาณ 240 ค่า โดยทำการเลื่อนเพื่อหาสัมประสิทธิ์ทีละ 160 ค่า จนครบทั้งหมด

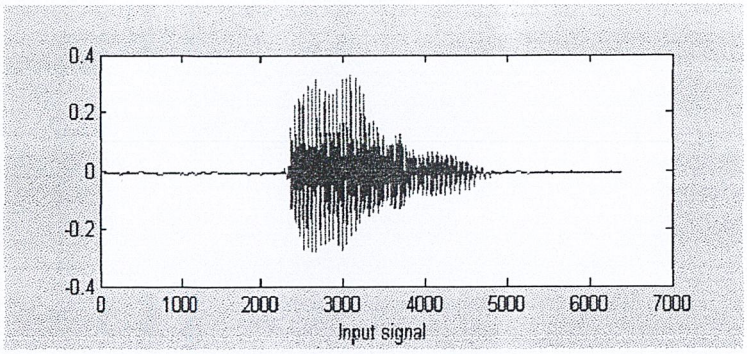


-หาค่าสัมประสิทธิ์เซปสตรัม โดยสัมประสิทธิ์แอลพีซี 10 ค่า หาค่าสัมประสิทธิ์เซปสตรัมได้ 12 ค่า ดังนั้นจะสามารถหาค่าสัมประสิทธิ์เซปสตรัมได้ 300 ค่าจากสัมประสิทธิ์แอลพีซี 250 ค่า จากนั้นนำสัมประสิทธิ์เซปสตรัมที่ได้มาถ่วงน้ำหนักเพื่อลดความคลาดเคลื่อน

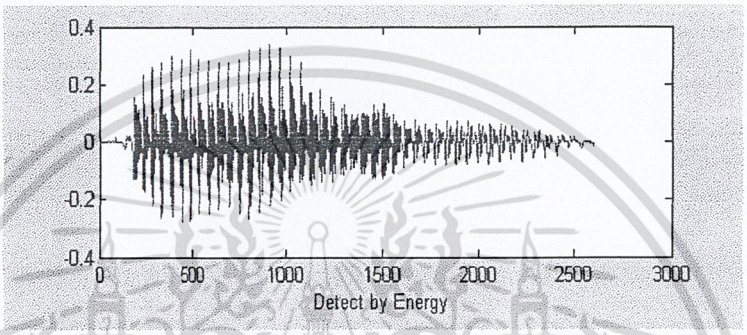


-นำสัมประสิทธิ์เซปสตรัมทีละ 12 ค่า ไปผ่านกระบวนการเวกเตอร์ควอนไตเซชั่นจะได้ตัวแทนของข้อมูล (เวกเตอร์อินเดกซ์) 1 ค่า ดังนั้นจะได้ตัวแทนของข้อมูลออกมาทั้งหมด 25 ค่า ตัวแทนข้อมูลที่ได้ทั้งหมดจะใช้เป็นลำดับค่าปรากฏ (Observation) ในการรู้จำเสียงต่อไป

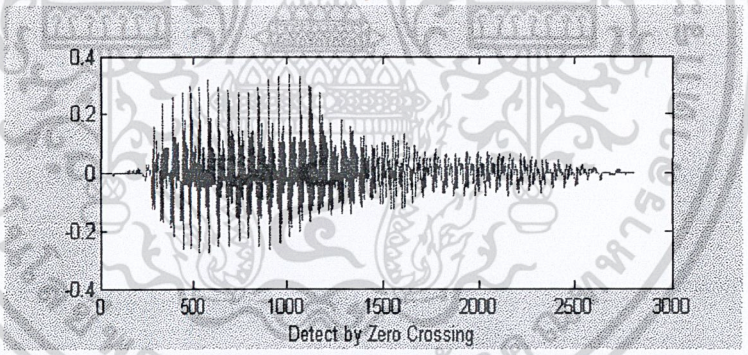
รูปที่ 4.28 แสดงเสียงคำว่า ปิด และการหาค่าพารามิเตอร์ต่างๆ ของเสียง



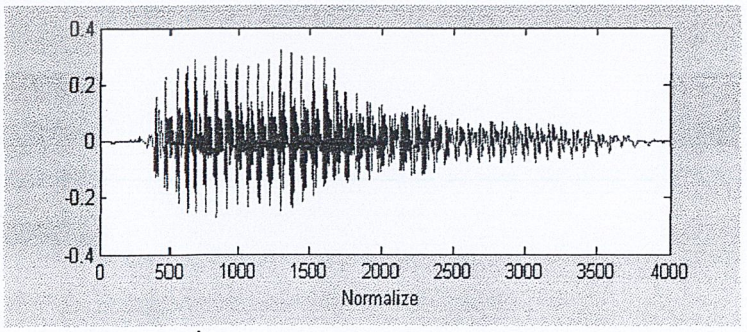
-กราฟแสดงข้อมูลของไฟล์เสียงคำว่า “ไฟ”



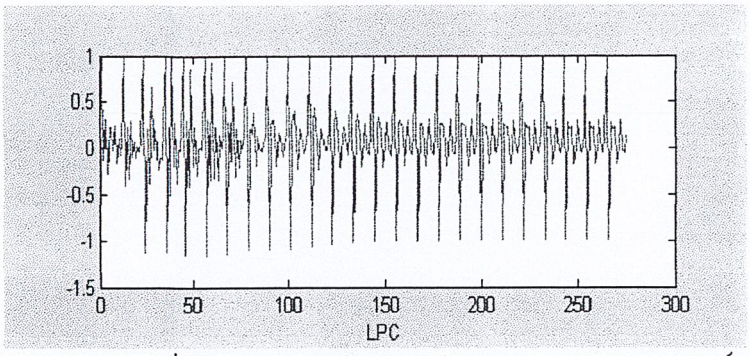
-ทำการหาขอบเขตของคำจากค่าพลังงานของเฟรมอ้างอิง โดยเฟรมดังกล่าวมีค่าพลังงาน 0.2 เท่าของเฟรมที่มีค่าพลังงานสูงสุด



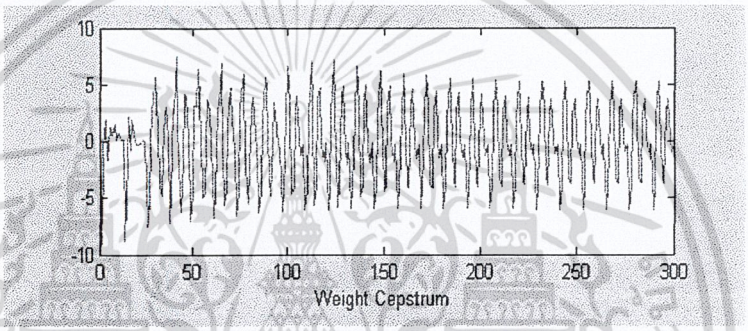
-ทำการหาขอบเขตของคำจากอัตราการตัดศูนย์เป็นค่าอ้างอิง โดยขอบเขตของข้อมูลมีค่าอัตราการตัดศูนย์เป็น 0.2 เท่าของเฟรมที่มีอัตราการตัดศูนย์สูงสุด



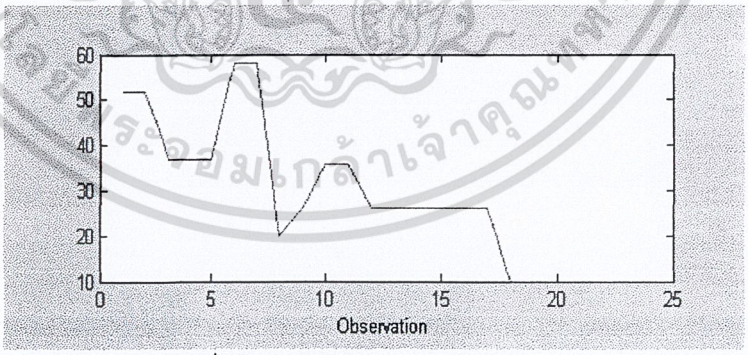
-นำสัญญาณที่หาขอบเขตแล้วมานอร์มอลไลซ์ให้มีจำนวนข้อมูล 4000 ค่า



-สัญญาณที่ได้จากการนอร์มอลไลซ์จะนำมาหาค่าสัมประสิทธิ์แอลพีซี โดยสัมประสิทธิ์แอลพีซี 10 ค่า หาได้จากสัญญาณ 240 ค่า โดยทำการเลื่อนเพื่อหาสัมประสิทธิ์ทีละ 160 ค่า จนครบทั้งหมด

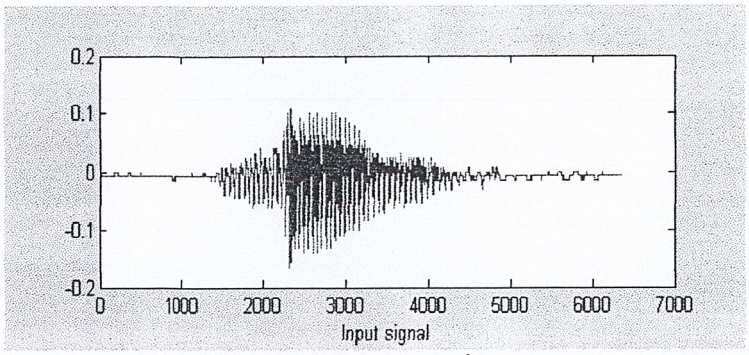


-หาค่าสัมประสิทธิ์เซปสตรัม โดยสัมประสิทธิ์แอลพีซี 10 ค่า หาค่าสัมประสิทธิ์เซปสตรัมได้ 12 ค่า ดังนั้นจะสามารถหาค่าสัมประสิทธิ์เซปสตรัมได้ 300 ค่า จากสัมประสิทธิ์แอลพีซี 250 ค่า จากนั้นนำสัมประสิทธิ์เซปสตรัมที่ได้มาถ่วงน้ำหนักเพื่อลดความคลาดเคลื่อน

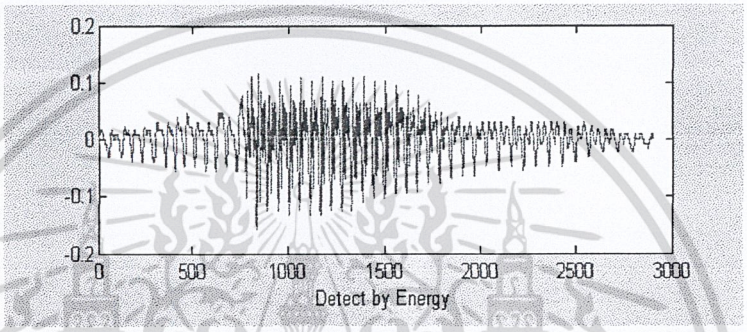


-นำสัมประสิทธิ์เซปสตรัมทีละ 12 ค่า ไปผ่านกระบวนการเวกเตอร์ควอนไตเซชั่นจะได้ตัวแทนของข้อมูล (เวกเตอร์อินเดกซ์) 1 ค่า ดังนั้นจะได้ตัวแทนของข้อมูลออกมาทั้งหมด 25 ค่า ตัวแทนข้อมูลที่ได้ทั้งหมดจะใช้เป็นลำดับค่าปรากฏ (Observation) ในการรู้จำเสียงต่อไป

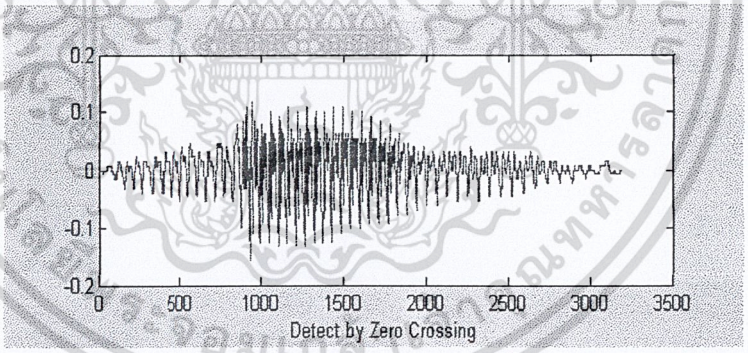
รูปที่ 4.29 แสดงเสียงคำว่า ไฟ และการหาค่าพารามิเตอร์ต่างๆ ของเสียง



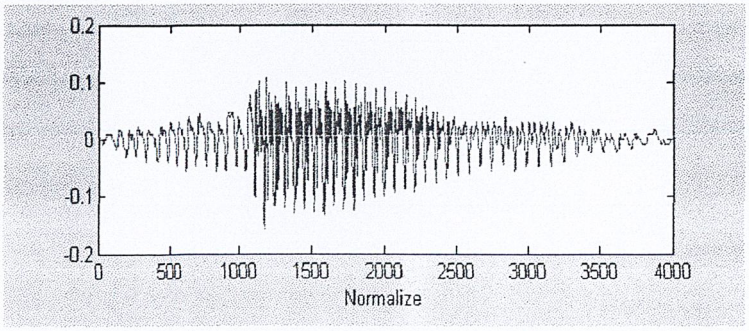
-กราฟแสดงข้อมูลของไฟล์เสียงคำว่า “หนึ่ง”



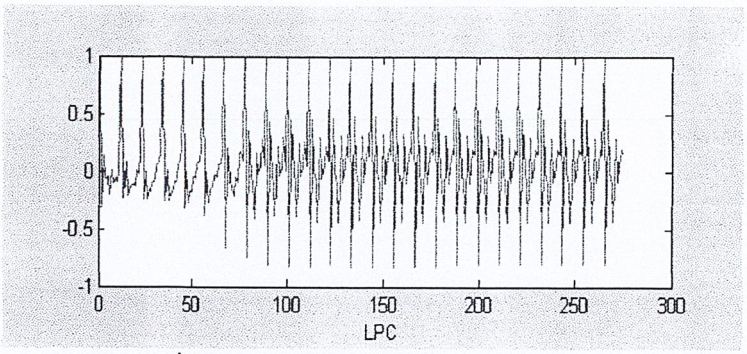
-ทำการหาขอบเขตของคำจากค่าพลังงานของเฟรมอ้างอิง โดยเฟรมดังกล่าวมีค่าพลังงาน 0.2 เท่าของเฟรมที่มีค่าพลังงานสูงสุด



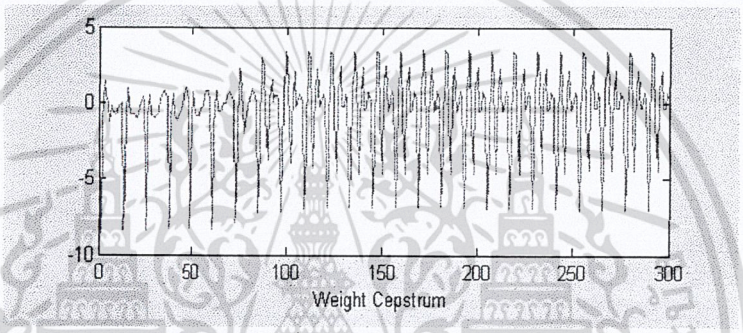
-ทำการหาขอบเขตของคำจากอัตราการตัดศูนย์เป็นค่าอ้างอิง โดยขอบเขตของข้อมูลมีค่าอัตราการตัดศูนย์เป็น 0.2 เท่าของเฟรมที่มีอัตราการตัดศูนย์สูงสุด



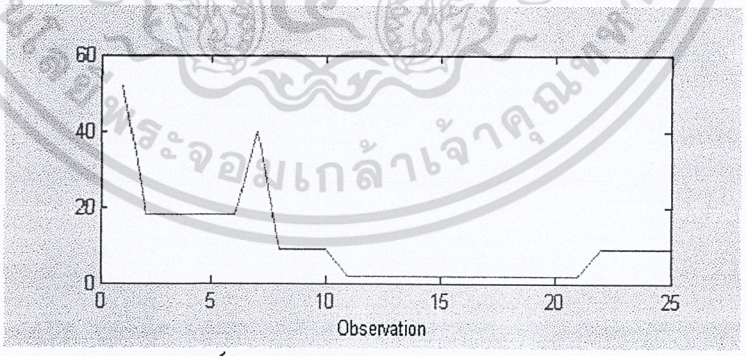
-นำสัญญาณที่หาขอบเขตแล้วมานอร์มอลไลซ์ให้มีจำนวนข้อมูล 4000 ค่า



-สัญญาณที่ได้จากการนอร์มอลไลซ์จะนำมาหาค่าสัมประสิทธิ์ แอลพีซี โดยสัมประสิทธิ์แอลพีซี 10 ค่า หาได้จากสัญญาณ 240 ค่า โดยทำการเลื่อนเพื่อหาสัมประสิทธิ์ทีละ 160 ค่า จนครบทั้งหมด

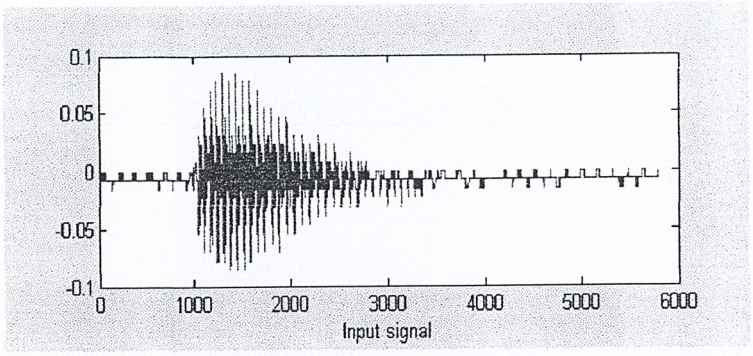


-หาค่าสัมประสิทธิ์เซปสตรัม โดยสัมประสิทธิ์แอลพีซี 10 ค่า หาค่าสัมประสิทธิ์เซปสตรัมได้ 12 ค่า ดังนั้นจะสามารถหาค่าสัมประสิทธิ์เซปสตรัมได้ 300 ค่า จากสัมประสิทธิ์แอลพีซี 250 ค่า จากนั้นนำสัมประสิทธิ์ เซปสตรัมที่ได้มาถ่วงน้ำหนักเพื่อลดความคลาดเคลื่อน

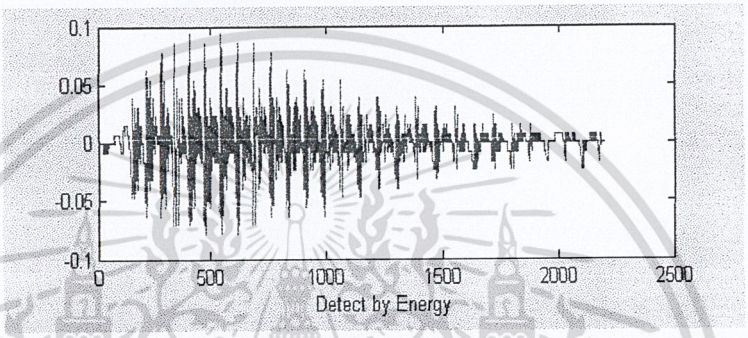


-นำสัมประสิทธิ์ เซปสตรัมทีละ 12 ค่า ไปผ่านกระบวนการเวกเตอร์ควอนไตเซชั่น จะได้ตัวแทนของข้อมูล (เวกเตอร์อินเดกซ์) 1 ค่า ดังนั้นจะได้ตัวแทนของข้อมูลออกมาทั้งหมด 25 ค่า ตัวแทนข้อมูลที่ได้ทั้งหมดจะใช้เป็นลำดับค่าปรากฏ (Observation) ในการรู้จำเสียงต่อไป

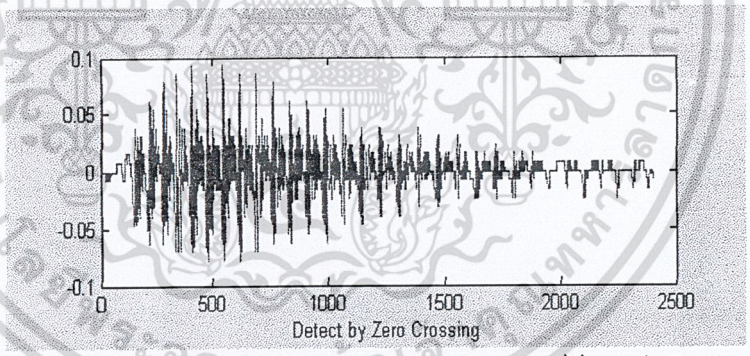
รูปที่ 4.30 แสดงเสียงคำว่า หนึ่ง และการหาค่าพารามิเตอร์ต่างๆ ของเสียง



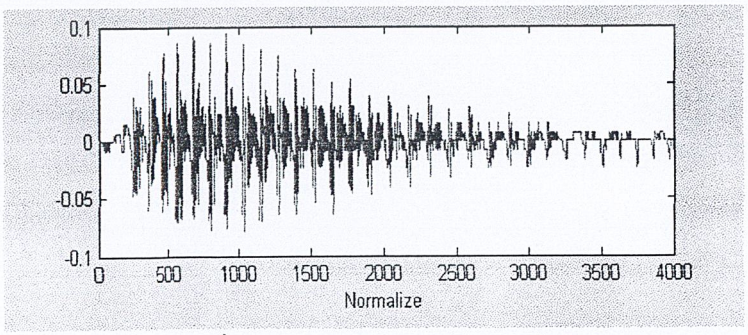
-กราฟแสดงข้อมูลของไฟล์เสียงคำว่า “สอง”



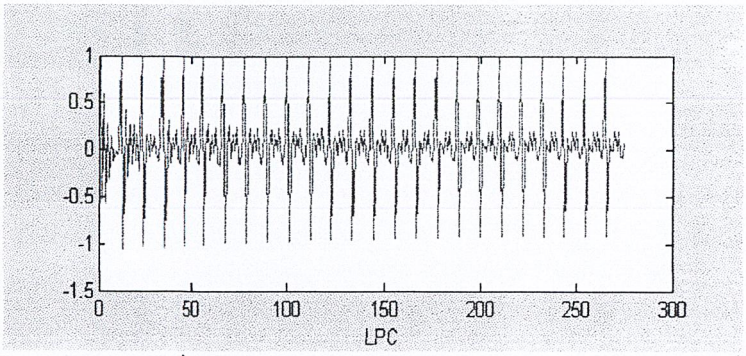
-ทำการหาขอบเขตของคำจากค่าพลังงานของเฟรมอ้างอิง โดยเฟรมดังกล่าวมีค่าพลังงาน 0.2 เท่าของเฟรมที่มีค่าพลังงานสูงสุด



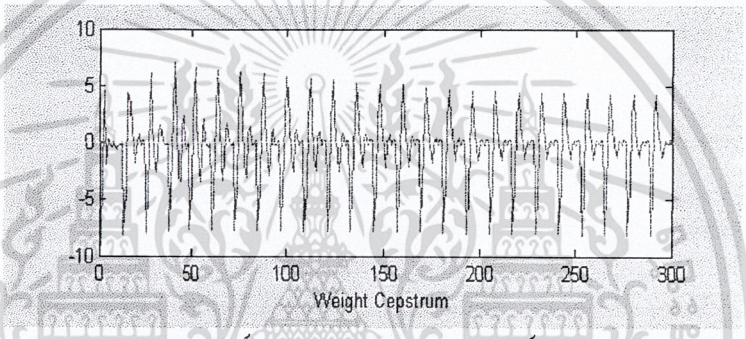
-ทำการหาขอบเขตของคำจากอัตราการตัดศูนย์เป็นค่าอ้างอิง โดยขอบเขตของข้อมูลมีค่าอัตราการตัดศูนย์เป็น 0.2 เท่าของเฟรมที่มีอัตราการตัดศูนย์สูงสุด



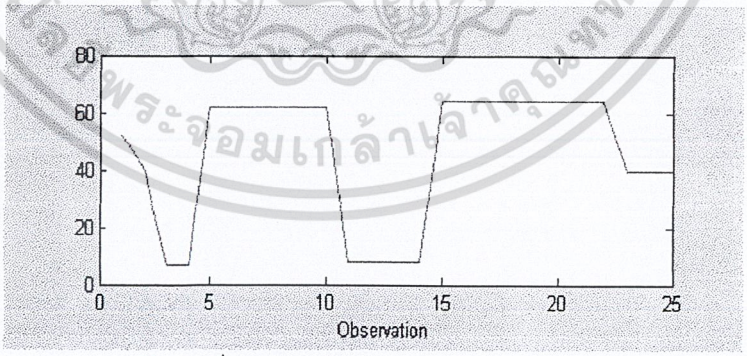
-นำสัญญาณที่หาขอบเขตแล้วมานอร์มอลไลซ์ให้มีจำนวนข้อมูล 4000 ค่า



-สัญญาณที่ได้จากการนอร์มอลไลซ์จะนำมาหาค่าสัมประสิทธิ์แอลพีซี โดยสัมประสิทธิ์แอลพีซี 10 ค่า หาได้จากสัญญาณ 240 ค่า โดยทำการเลื่อนเพื่อหาสัมประสิทธิ์ทีละ 160 ค่า จนครบทั้งหมด

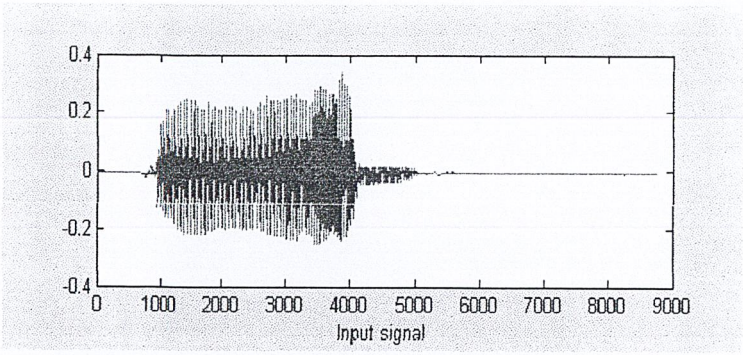


-หาค่าสัมประสิทธิ์เซปสตรัม โดยสัมประสิทธิ์แอลพีซี 10 ค่า หาค่าสัมประสิทธิ์เซปสตรัมได้ 12 ค่า ดังนั้นจะสามารถหาค่าสัมประสิทธิ์เซปสตรัมได้ 300 ค่า จากสัมประสิทธิ์แอลพีซี 250 ค่า จากนั้นนำสัมประสิทธิ์เซปสตรัมที่ได้มาถ่วงน้ำหนักเพื่อลดความคลาดเคลื่อน

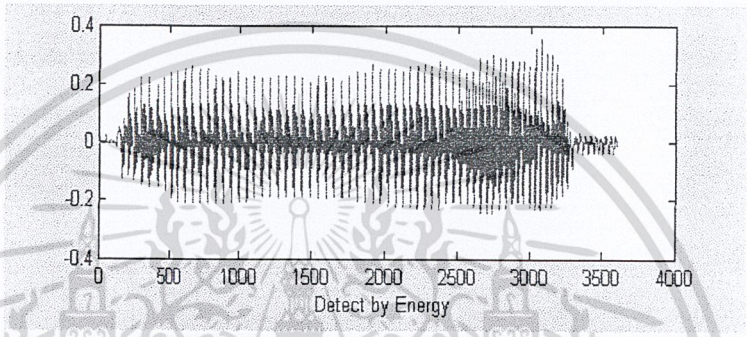


-นำสัมประสิทธิ์เซปสตรัมทีละ 12 ค่า ไปผ่านกระบวนการเวกเตอร์ควอนไตเซชั่นจะได้ตัวแทนของข้อมูล (เวกเตอร์อินเดกซ์) 1 ค่า ดังนั้นจะได้ตัวแทนของข้อมูลออกมาทั้งหมด 25 ค่า ตัวแทนข้อมูลที่ได้ทั้งหมดจะใช้เป็นลำดับค่าปรากฏ (Observation) ในการรู้จำเสียงต่อไป

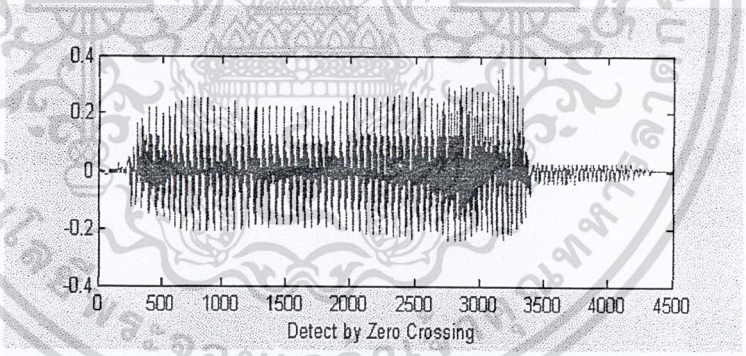
รูปที่ 4.31 แสดงเสียงคำว่า สอง และการหาค่าพารามิเตอร์ต่างๆ ของเสียง



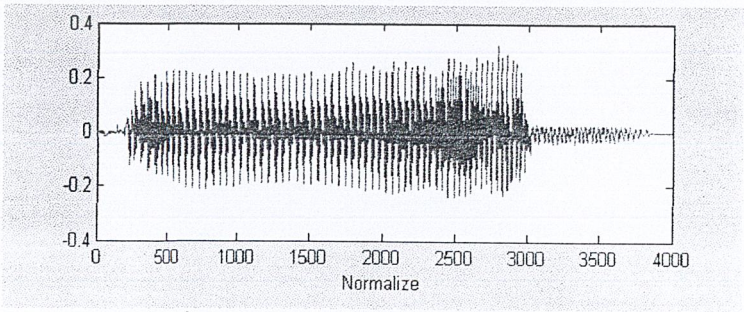
-กราฟแสดงข้อมูลของไฟล์เสียงคำว่า "สาม"



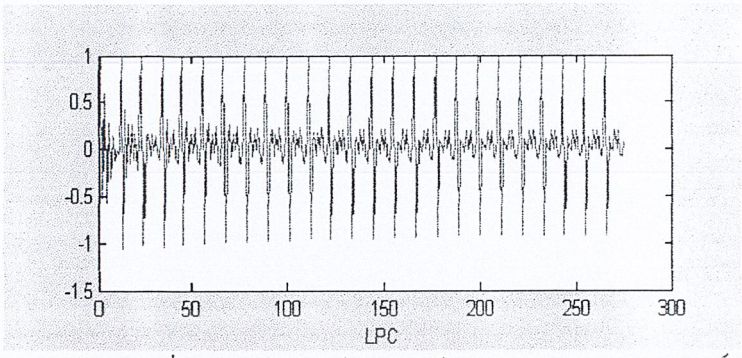
-ทำการหาขอบเขตของคำจากค่าพลังงานของเฟรมอ้างอิง โดยเฟรมดังกล่าวมีค่าพลังงาน 0.2 เท่าของเฟรมที่มีค่าพลังงานสูงสุด



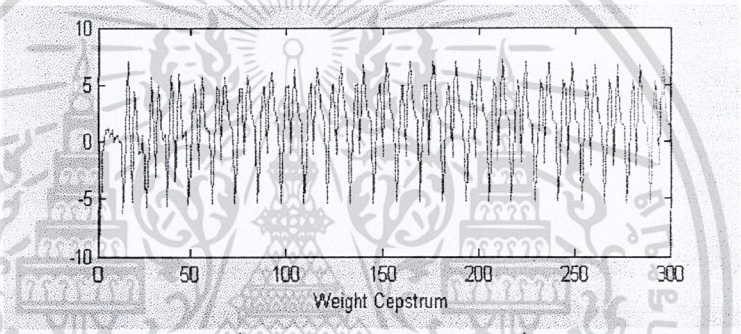
-ทำการหาขอบเขตของคำจากอัตราการตัดศูนย์เป็นค่าอ้างอิง โดยขอบเขตของข้อมูลมีค่าอัตราการตัดศูนย์เป็น 0.2 เท่าของเฟรมที่มีอัตราการตัดศูนย์สูงสุด



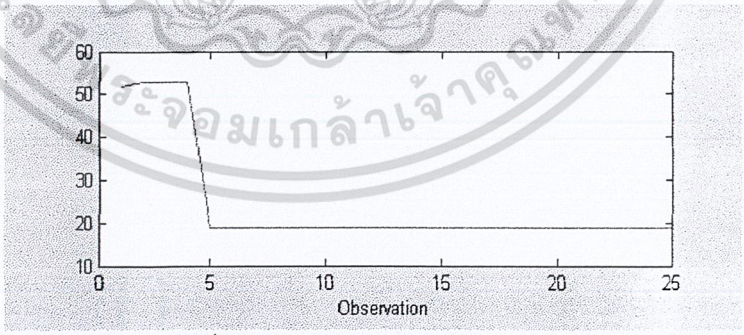
-นำสัญญาณที่หาขอบเขตแล้วมานอร์มอลไลซ์ให้มีจำนวนข้อมูล 4000 ค่า



-สัญญาณที่ได้จากการนอร์มอลไลซ์จะนำมาหาค่าสัมประสิทธิ์แอลพีซี โดยสัมประสิทธิ์แอลพีซี 10 ค่า หาได้จากสัญญาณ 240 ค่า โดยทำการเลื่อนเพื่อหาสัมประสิทธิ์ทีละ 160 ค่า จนครบทั้งหมด



-หาค่าสัมประสิทธิ์เซปสตรัม โดยสัมประสิทธิ์แอลพีซี 10 ค่า หาค่าสัมประสิทธิ์เซปสตรัมได้ 12 ค่า ดังนั้นจะสามารถหาค่าสัมประสิทธิ์เซปสตรัมได้ 300 ค่า จากสัมประสิทธิ์แอลพีซี 250 ค่า จากนั้นนำสัมประสิทธิ์เซปสตรัมที่ได้มาถ่วงน้ำหนักเพื่อลดความคลาดเคลื่อน

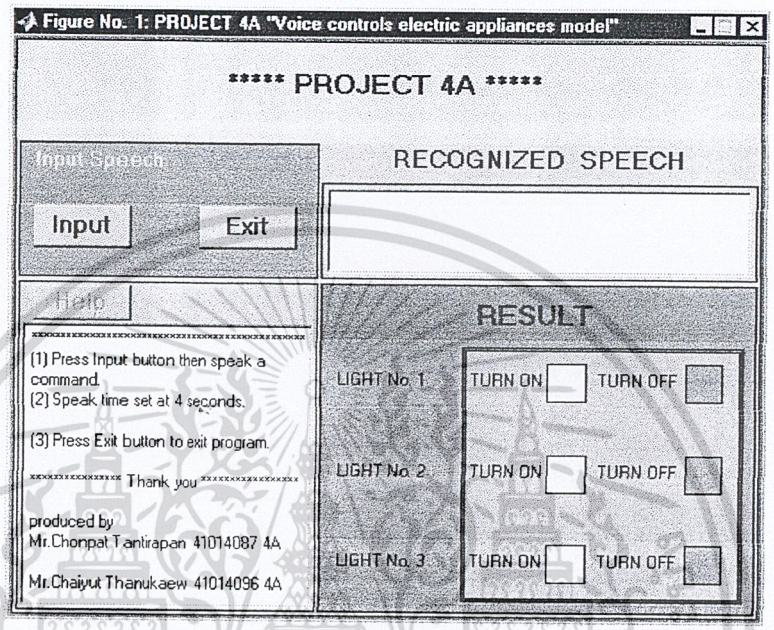


-นำสัมประสิทธิ์เซปสตรัมทีละ 12 ค่า ไปผ่านกระบวนการเวกเตอร์ควอนไตเซชั่น จะได้ตัวแทนของข้อมูล (เวกเตอร์อินเดกซ์) 1 ค่า ดังนั้นจะได้ตัวแทนของข้อมูลออกมาทั้งหมด 25 ค่า ตัวแทนข้อมูลที่ได้ทั้งหมดจะใช้เป็นลำดับค่าปรากฏ (Observation) ในการรู้จำเสียงต่อไป

รูปที่ 4.32 แสดงเสียงคำว่า สาม และการหาค่าพารามิเตอร์ต่างๆ ของเสียง

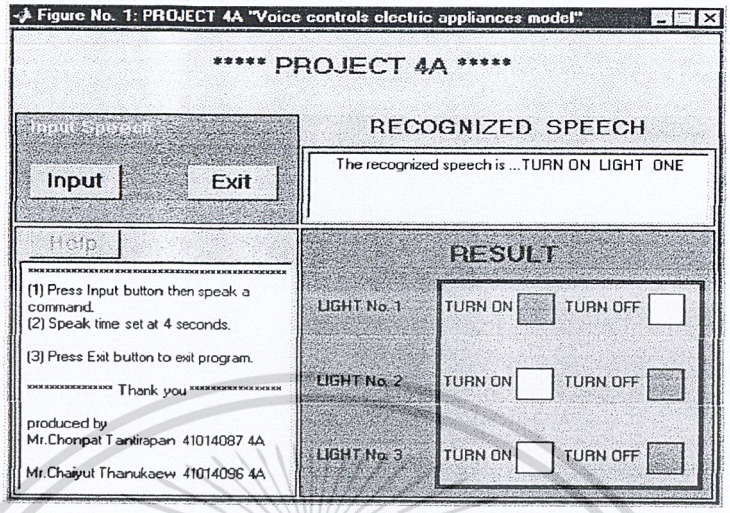
4.1.5 ขั้นตอนการทดลองโปรแกรมการรู้จำเสียงพูด

1) จากหน้าต่าง GUI ของโปรแกรมการควบคุมเครื่องใช้ไฟฟ้าด้วยเสียง ทำการบันทึกเสียงที่ใช้ในการสั่งงาน โดยคำสั่งที่ใช้ในการสั่งงานต้องเป็นคำสั่งเสียง 3 พยางค์ และมีระยะเวลาในการบันทึกสัญญาณเสียง 4 วินาที เสียงคำสั่งที่จะถูกบันทึกในรูปของไฟล์ข้อมูล (ไฟล์.mat) ของโปรแกรม MATLAB

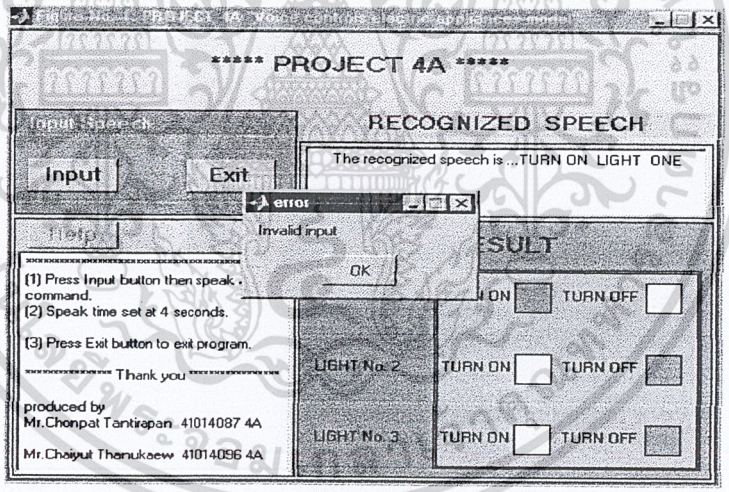


รูปที่ 4.33 แสดงหน้าต่าง GUI ของโปรแกรมการควบคุมเครื่องใช้ไฟฟ้าด้วยเสียงพูด

2) เสียงคำสั่งขนาด 3 พยางค์ จะถูกนำมาแบ่งเพื่อประมวลผลทีละพยางค์ สัญญาณแต่ละพยางค์จะนำมาหาขอบเขตของคำโดย การหาค่าพลังงาน การหาอัตราตัดศูนย์ และทำการนอร์มอลไลซ์ให้มีความยาวในแต่ละสัญญาณเท่ากัน ก่อนที่จะนำสัญญาณเสียงดังกล่าวมาทำการรู้จำเสียง
3) สัญญาณเสียงแต่ละพยางค์ที่ทำการนอร์มอลไลซ์แล้ว จะนำมาผ่านวงจรกรองแบบพรีเอมพาซีส ผ่านเข้าสู่กระบวนการแบ่งช่วงสัญญาณ ก่อนที่จะทำการหาค่าพารามิเตอร์ต่างๆ ได้แก่ สัมประสิทธิ์แอลพีซี สัมประสิทธิ์เซปสตรัมและสัมประสิทธิ์เซปสตรัมถ่วงน้ำหนัก
4) ค่าพารามิเตอร์ที่ได้จะนำไปเปรียบเทียบกับโค้ดบุค 64 จากเวกเตอร์เสียงเทรนนิ่ง และจะหาค่าความน่าจะเป็นสูงสุดที่เป็นไปได้ จากนั้นหน้าต่าง GUI จะแสดงผลการทดสอบการรู้จำเสียงพูดออกมา ทำการทดสอบในกรณีที่เสียงสัญญาณอินพุตเป็นเสียงคำสั่ง ต่างๆ กัน



รูปที่ 4.34 แสดงตัวอย่างของหน้าต่าง GUI เมื่อพูดคำสั่ง เปิด ไฟ หนึ่ง



รูปที่ 4.35 แสดงตัวอย่างของหน้าต่าง GUI เมื่อพูดเสียงคำสั่งที่พูดใช้งานไม่ได้
โดยหน้าต่าง GUI จะให้ป๊อนอินพุตใหม่

4.1.6 ผลการทดลองโปรแกรมรู้จำเสียงพูด

1) ทำการทดสอบโปรแกรมรู้จำเสียงพูดโดยใช้เสียงจากผู้พูดต้นแบบที่นำมาสร้างโค้ดบุค เป็นผู้ชายจำนวน 5 คนและผู้หญิง 5 คน โดยแบ่งการทดลองออกเป็นเสียงผู้ชายและเสียงผู้หญิง แต่ละคนกำหนดให้พูด

- เสียง "เปิดไฟหนึ่ง" จำนวน 5 เสียง
- เสียง "เปิดไฟสอง" จำนวน 5 เสียง
- เสียง "เปิดไฟสาม" จำนวน 5 เสียง
- เสียง "ปิดไฟหนึ่ง" จำนวน 5 เสียง
- เสียง "ปิดไฟสอง" จำนวน 5 เสียง
- เสียง "ปิดไฟสาม" จำนวน 5 เสียง

ผลการทดลองโปรแกรมรู้จำเสียงพูดกับเสียงต้นแบบของผู้ชายที่นำมาสร้างโค้ดบุค ดังตารางที่ 4.1

เสียงอินพุตจากผู้ชาย	รูปแบบเสียงที่รู้จำได้(เสียง)							เปอร์เซ็นต์ความถูกต้อง(%)
	เปิดไฟหนึ่ง	เปิดไฟสอง	เปิดไฟสาม	ปิดไฟหนึ่ง	ปิดไฟสอง	ปิดไฟสาม	ผิดพลาด	
เปิดไฟหนึ่ง	11	1					8	60
เปิดไฟสอง		8	6				6	40
เปิดไฟสาม		4	12				4	60
ปิดไฟหนึ่ง				15			5	75
ปิดไฟสอง					14	2	4	70
ปิดไฟสาม						11	9	55

ตารางที่ 4.1 แสดงผลการทดลองของโปรแกรมรู้จำเสียงคำสั่งเมื่อเสียงคำสั่งเป็นเสียงของผู้ชายที่นำมาสร้างโค้ดบุค

และผลการทดลองโปรแกรมรู้จำเสียงพูดกับเสียงต้นแบบของผู้หญิงที่นำมาสร้างโค้ดบุค ดังตารางที่ 4.2

เสียงอินพุตจากผู้หญิง	รูปแบบเสียงที่รู้จำได้(เสียง)							เปอร์เซ็นต์ความถูกต้อง(%)
	เปิดไฟหนึ่ง	เปิดไฟสอง	เปิดไฟสาม	ปิดไฟหนึ่ง	ปิดไฟสอง	ปิดไฟสาม	ผิดพลาด	
เปิดไฟหนึ่ง	17						3	85
เปิดไฟสอง		10	5				5	50
เปิดไฟสาม		5	8				7	40
ปิดไฟหนึ่ง				14			6	70
ปิดไฟสอง					8	3	9	40
ปิดไฟสาม					4	12	4	60

ตารางที่ 4.2 แสดงผลการทดลองของโปรแกรมรู้จำเสียงคำสั่งเมื่อ
เสียงคำสั่งเป็นเสียงของผู้หญิงที่นำมาสร้างโค้ดบุค

2) ทำการทดสอบโปรแกรมรู้จำเสียงพูดโดยใช้เสียงจากผู้พูดที่ไม่ได้นำมาสร้างโค้ดบุค เป็นผู้ชายจำนวน 5 คนและผู้หญิง 5 คน โดยแบ่งการทดลองออกเป็นเสียงผู้ชายและเสียงผู้หญิงเหมือนกับการทดลองในข้อที่แล้ว

ผลการทดลองโปรแกรมรู้จำเสียงพูดกับเสียงของผู้ชายที่ไม่ได้นำมาสร้างโค้ดบุค ดังตารางที่ 4.3

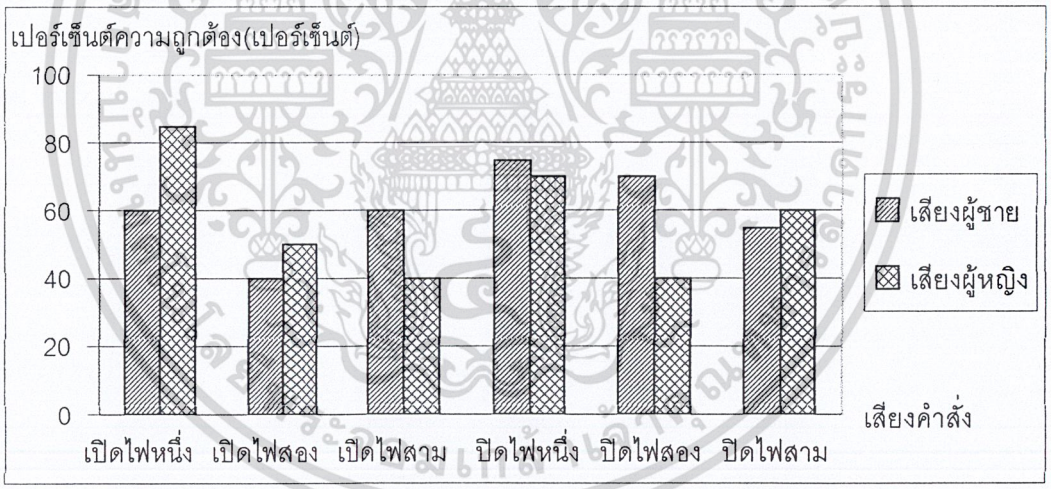
เสียงอินพุตจากผู้ชาย	รูปแบบเสียงที่รู้จำได้(เสียง)							เปอร์เซ็นต์ความถูกต้อง(%)
	เปิดไฟหนึ่ง	เปิดไฟสอง	เปิดไฟสาม	ปิดไฟหนึ่ง	ปิดไฟสอง	ปิดไฟสาม	ผิดพลาด	
เปิดไฟหนึ่ง	11	1					8	55
เปิดไฟสอง		6	6				8	30
เปิดไฟสาม		8	8				4	40
ปิดไฟหนึ่ง				12			8	60
ปิดไฟสอง					12		8	60
ปิดไฟสาม					1	10	9	50

ตารางที่ 4.3 แสดงผลการทดลองของโปรแกรมรู้จำเสียงคำสั่งเมื่อ
เสียงคำสั่งเป็นเสียงของผู้ชายที่ไม่ได้นำมาสร้างโค้ดบุค

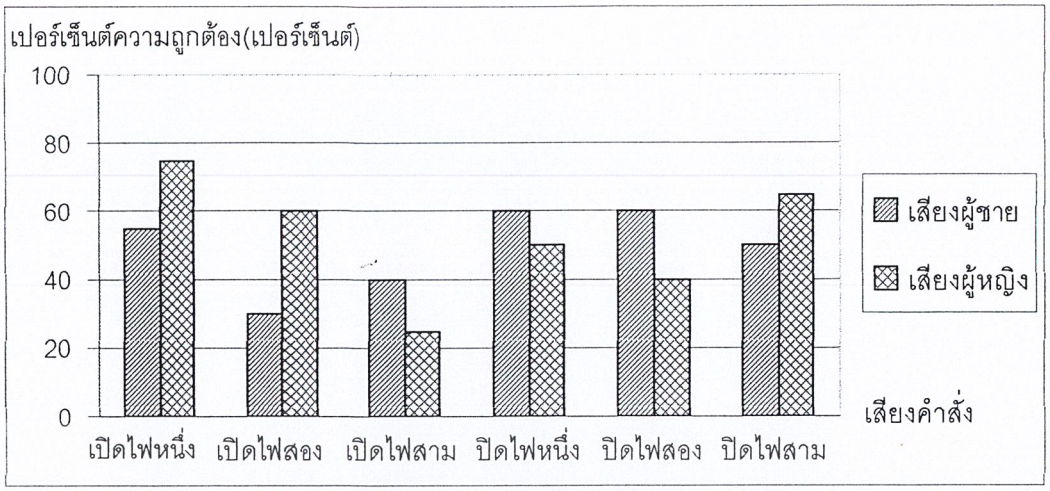
และผลการทดลองโปรแกรมรู้จำเสียงพูดกับเสียงต้นแบบของผู้หญิงที่ไม่ได้นำมาสร้างโค้ดบุค ดัง ตารางที่ 4.4

เสียงอินพุตจากผู้หญิง	รูปแบบเสียงที่รู้จำได้(เสียง)							เปอร์เซ็นต์ความถูกต้อง(%)
	เปิดไฟหนึ่ง	เปิดไฟสอง	เปิดไฟสาม	ปิดไฟหนึ่ง	ปิดไฟสอง	ปิดไฟสาม	ผิดพลาด	
เปิดไฟหนึ่ง	15						5	75
เปิดไฟสอง		10	2				8	50
เปิดไฟสาม		7	5				8	25
ปิดไฟหนึ่ง				12			8	60
ปิดไฟสอง					8	1	11	40
ปิดไฟสาม					1	15	4	75

ตารางที่ 4.4 แสดงผลการทดลองของโปรแกรมรู้จำเสียงคำสั่งเมื่อ
เสียงคำสั่งเป็นเสียงของผู้หญิงที่ไม่ได้นำมาสร้างโค้ดบุค



รูปที่ 4.36 กราฟเปรียบเทียบเปอร์เซ็นต์ความถูกต้องของการรู้จำเสียงคำสั่ง
โดยใช้เสียงผู้พูดที่นำมาสร้างโค้ดบุค



รูปที่ 4.37 กราฟเปรียบเทียบเปอร์เซ็นต์ความถูกต้องของการรู้จำเสียงคำสั่ง
โดยใช้เสียงผู้พูดที่ไม่ได้นำมาสร้างโค้ดบุค

บทที่ 5
สรุปและวิจารณ์

จากการทดสอบโปรแกรมการรู้จำเสียงพูดกับเสียงคำสั่งทีละ 3 พยางค์ตามที่กำหนดไว้ โดยป้อนสัญญาณเสียงคำสั่งจากไมโครโฟน บันทึกแบบ 8 บิต สุ่มด้วยความถี่ 8 กิโลเฮิรตซ์ ทดสอบกับเสียงของผู้ชายและผู้หญิงอย่างละ 5 คน สามารถสรุปได้ดังนี้

1. การทดลองได้นำกระบวนการรู้จำเสียงคำโดดมาประยุกต์ใช้ ดังนั้นในการใช้งานจริงผู้ใช้ต้องเว้นช่วงระหว่างคำในแต่ละพยางค์ที่พูดเพื่อให้โปรแกรมสามารถแยกเสียงคำสั่งหลายคำออกเป็นคำโดดได้
2. ในการใช้งานโปรแกรมรู้จำเสียง การบันทึกเสียงต้องมีระดับสัญญาณรบกวนที่ต่ำ เพื่อให้โปรแกรมสามารถรับรู้ได้ว่าสัญญาณที่บันทึกได้ ส่วนใดเป็นเสียงคำสั่ง ส่วนใดเป็นสัญญาณรบกวน
3. เนื่องจากสัญญาณเสียงที่แยกได้ในแต่ละพยางค์มีความยาวไม่เท่ากัน ทำให้การหาค่าเฉลี่ยของสัญญาณที่นำมาสร้างแบบอ้างอิงทำได้ไม่แน่นอน เพื่อความถูกต้องของการสร้างแบบอ้างอิง จึงทำการนอร์มอลไลซ์ให้สัญญาณมีความยาวเท่ากันก่อนที่จะนำมาหาค่าพารามิเตอร์ต่างๆ
4. จากวิธีการของฮิดเดนมาร์คอฟนั้น เมื่อค่าความน่าจะเป็นของค่าปรากฏมีค่าเป็นศูนย์ จะทำให้ความน่าจะเป็นรวมมีค่าเป็นศูนย์ด้วย
5. การใช้เสียงจากผู้พูดจำนวนมากและโค้ดบุคที่มีขนาดใหญ่ ทำให้ฐานข้อมูลมีขนาดใหญ่ จึงใช้เวลามากในการหาโมเดลเสียง

ปัญหาที่เกิดขึ้นในการทดลอง

1. การหาค่าพารามิเตอร์ของสัญญาณเสียงนั้นได้ทำการคำนวณในโดเมนเวลาเนื่องจากวิธีการไม่ยุ่งยากเมื่อเทียบกับการหาค่าพารามิเตอร์ในโดเมนของความถี่ซึ่งมีวิธีซับซ้อนกว่าแต่มีความถูกต้องมากกว่า ดังนั้นจึงควรพัฒนาโดยการหาค่าพารามิเตอร์ในโดเมนของความถี่เพื่อการจดจำที่ดีขึ้น
2. ปัจจุบันได้มีการพัฒนาวิธีการเปรียบเทียบความคล้ายคลึงของสัญญาณ เช่น นิวรอลเน็ตเวิร์ค ซึ่งจะเหมาะกับการหาพารามิเตอร์ในโดเมนของความถี่ทำให้เราสามารถเลือกใช้วิธีต่างๆ เหล่านี้เพื่อใช้ในการจดจำเสียงที่มีประสิทธิภาพสูงมากๆได้

ภาคผนวก

โปรแกรมการสร้างเมนู

Graphic User Interface : GUI

```
function user(action);
global recog stat1 stat2 stat3
load state;
if nargin==0
  action='initialize';
end
if strcmp(action,'initialize')
  a = figure('Color',[0.35 0.35 0.35], ...
    'MenuBar','none', ...
    'Name','PROJECT 4A "Voice controls electric appliances model"', ...
    'Position',[140 100 520 400], ...  %390,300
    'Resize','off', ...
    'Tag','Fig1');
  recog1 = uicontrol('Parent',a, ...
    'Units','points', ...
    'BackgroundColor',[0.85 0.85 0.85], ...
    'Position',[1.5 1.5 387 297], ...
    'Style','frame', ...
    'Tag','frame1');
  recog1 = uicontrol('Parent',a, ...
    'Units','points', ...
    'BackgroundColor',[0.5 0.5 0.7], ...
    'Position',[4 176 157 70], ...
    'Style','frame', ...
    'Tag','frame2');
  recog1 = uicontrol('Parent',a, ...
    'Units','points', ...
    'BackgroundColor',[0.75 0.75 0.75], ...
    'Position',[163 176 223 50], ...
    'Style','frame', ...
    'Tag','frame3');
  recog1 = uicontrol('Parent',a, ...
    'Units','points', ...
    'BackgroundColor',[0.75 0.8 0.75], ...
    'Position',[4 4 157 170], ...
    'Style','frame', ...
    'Tag','frame4');
  recog1 = uicontrol('Parent',a, ...
    'Units','points', ...
    'BackgroundColor',[0.05 0.75 0.75], ...
    'Position',[163 4 223 170], ...
    'Style','frame', ...
    'Tag','frame5');
  recog1 = uicontrol('Parent',a, ...
    'Units','points', ...
    'BackgroundColor',[0.3 0.4 0.25], ...
    'Position',[236 10 144 132], ...
    'Style','frame', ...
    'Tag','frame6');
  recog1 = uicontrol('Parent',a, ...
```

```
    'Units','points', ...
    'BackgroundColor',[0.7 0.7 0.7], ...
    'Position',[237.5 11.5 141 129], ...
    'Style','frame', ...
    'Tag','frame7');
recog1 = uicontrol('Parent',a, ...
    'Units','points', ...
    'BackgroundColor',[0.85 0.85 0.85], ...
    'ForegroundColor',[0 0 1], ...
    'FontSize',14, ...
    'FontWeight','bold', ...
    'Position',[50 266 290 20], ...
    'String','****** PROJECT 4A ******', ...
    'Style','text', ...
    'Tag','statictext1');
recog1 = uicontrol('Parent',a, ...
    'Units','points', ...
    'BackgroundColor',[0.5 0.5 0.7], ...
    'ForegroundColor',[0.85 0.85 0.6], ...
    'FontSize',10, ...
    'FontWeight','bold', ...
    'HorizontalAlignment','left', ...
    'Position',[12 225 80 18], ...
    'String','Input Speech', ...
    'Style','text', ...
    'Tag','statictext2');
recog1 = uicontrol('Parent',a, ...
    'Units','points', ...
    'BackgroundColor',[0.75 0.75 0.75], ...
    'ForegroundColor',[0 0 0], ...
    'FontSize',12, ...
    'FontWeight','bold', ...
    'Position',[163 226 223 20], ...
    'String','RECOGNIZED  SPEECH', ...
    'Style','text', ...
    'Tag','statictext3');
recog(1) = uicontrol('Parent',a, ...
    'Units','points', ...
    'BackgroundColor',[0.5 0.5 0.7], ...
    'ForegroundColor',[0 0 0], ...
    'FontSize',12, ...
    'FontWeight','bold', ...
    'Position',[100 225 60 18], ...
    'String','', ...
    'Style','text', ...
    'Tag','statictext4');                    %shoe "speak" & "process"
recog(2) = uicontrol('Parent',a, ...
    'Callback','user("INPUT")', ...
    'BackgroundColor',[0.75 0.75 0.75], ...
    'ForegroundColor',[0 0 0], ...
```

```
    'Units','points', ...
    'FontSize',12, ...
    'FontWeight','bold', ...
    'Position',[12 191 50 22], ...
    'String','Input', ...
    'userdata',1, ...
    'Tag','pushbutton1');                    %button Input
recog(3) = uicontrol('Parent',a, ...
    'Callback','close(gcbf)', ...
    'BackgroundColor',[0.75 0.75 0.75], ...
    'ForegroundColor',[0 0 0], ...
    'Units','points', ...
    'FontSize',12, ...
    'FontWeight','bold', ...
    'Position',[100 191 50 22], ...
    'String','Exit', ...
    'userdata',1, ...
    'Tag','pushbutton2');                    %button Exit
recog(4) = uicontrol('Parent',a, ...
    'Units','points', ...
    'BackgroundColor',[1 1 1], ...
    'HorizontalAlignment','center', ...
    'Max',5, ...
    'Position',[166 179 217 44], ...
    'Style','edit', ...
    'Tag','edit1');                          %edit1 (show output)
recog(5) = uicontrol('Parent',a, ...
    'Callback','user("HELP")', ...
    'BackgroundColor',[0.75 0.75 0.75], ...
    'ForegroundColor',[0 0 0], ...
    'Units','points', ...
    'FontSize',12, ...
    'FontWeight','bold', ...
    'Position',[12 154 50 18], ...
    'String','Help', ...
    'userdata',1, ...
    'Tag','pushbutton3');                    %button Help
recog(6) = uicontrol('Parent',a, ...
    'Units','points', ...
    'BackgroundColor',[1 1 1], ...
    'FontSize',4, ...
    'HorizontalAlignment','left', ...
    'Max',5, ...
    'Position',[6 6 153 146], ...
    'Style','edit', ...
    'Tag','edit2');                          %edit2 (show help)
recog1 = uicontrol('Parent',a, ...
    'Units','points', ...
    'BackgroundColor',[0.05 0.75 0.75], ...
    'ForegroundColor',[0 0 1], ...
```

```
    'FontSize',14, ...
    'FontWeight','bold', ...
    'Position',[173 147 200 20], ...
    'String',' RESULT ', ...
    'Style','text', ...
    'Tag','statictext5');
recog1 = uicontrol('Parent',a, ...
    'Units','points', ...
    'BackgroundColor',[0.05 0.75 0.75], ...
    'ForegroundColor',[0 0 0], ...
    'FontSize',8, ...
    'FontWeight','normal', ...
    'HorizontalAlignment','left', ...
    'Position',[172 16.5 60 20], ...
    'String','LIGHT No. 3', ...
    'Style','text', ...
    'Tag','statictext6');
recog1 = uicontrol('Parent',a, ...
    'Units','points', ...
    'BackgroundColor',[0.05 0.75 0.75], ...
    'ForegroundColor',[0 0 0], ...
    'FontSize',8, ...
    'FontWeight','normal', ...
    'HorizontalAlignment','left', ...
    'Position',[172 63 60 20], ...
    'String','LIGHT No. 2', ...
    'Style','text', ...
    'Tag','statictext7');
recog1 = uicontrol('Parent',a, ...
    'Units','points', ...
    'BackgroundColor',[0.05 0.75 0.75], ...
    'ForegroundColor',[0 0 0], ...
    'FontSize',8, ...
    'FontWeight','normal', ...
    'HorizontalAlignment','left', ...
    'Position',[172 110.5 60 20], ...
    'String','LIGHT No. 1', ...
    'Style','text', ...
    'Tag','statictext8');
recog1 = uicontrol('Parent',a, ...
    'Units','points', ...
    'BackgroundColor',[0.7 0.7 0.7], ...
    'ForegroundColor',[0 0 0], ...
    'FontSize',8, ...
    'FontWeight','normal', ...
    'HorizontalAlignment','left', ...
    'Position',[240 16.5 40 20], ...
    'String','TURN ON', ...
    'Style','text', ...
    'Tag','statictext9');
```

```
recog1 = uicontrol('Parent',a, ...
    'Units','points', ...
    'BackgroundColor',[0.7 0.7 0.7], ...
    'ForegroundColor',[0 0 0], ...
    'FontSize',8, ...
    'FontWeight','normal', ...
    'HorizontalAlignment','left', ...
    'Position',[240 63 40 20], ...
    'String','TURN ON', ...
    'Style','text', ...
    'Tag','statictext10');
recog1 = uicontrol('Parent',a, ...
    'Units','points', ...
    'BackgroundColor',[0.7 0.7 0.7], ...
    'ForegroundColor',[0 0 0], ...
    'FontSize',8, ...
    'FontWeight','normal', ...
    'HorizontalAlignment','left', ...
    'Position',[240 110.5 40 20], ...
    'String','TURN ON', ...
    'Style','text', ...
    'Tag','statictext11');
recog1 = uicontrol('Parent',a, ...
    'Units','points', ...
    'BackgroundColor',[0.7 0.7 0.7], ...
    'ForegroundColor',[0 0 0], ...
    'FontSize',8, ...
    'FontWeight','normal', ...
    'HorizontalAlignment','left', ...
    'Position',[305 16.5 50 20], ...
    'String','TURN OFF', ...
    'Style','text', ...
    'Tag','statictext12');
recog1 = uicontrol('Parent',a, ...
    'Units','points', ...
    'BackgroundColor',[0.7 0.7 0.7], ...
    'ForegroundColor',[0 0 0], ...
    'FontSize',8, ...
    'FontWeight','normal', ...
    'HorizontalAlignment','left', ...
    'Position',[305 63 50 20], ...
    'String','TURN OFF', ...
    'Style','text', ...
    'Tag','statictext13');
recog1 = uicontrol('Parent',a, ...
    'Units','points', ...
    'BackgroundColor',[0.7 0.7 0.7], ...
    'ForegroundColor',[0 0 0], ...
    'FontSize',8, ...
    'FontWeight','normal', ...
```

```
    'HorizontalAlignment','left', ...
    'Position',[305 110.5 50 20], ...
    'String','TURN OFF', ...
    'Style','text', ...
    'Tag','statictext14');
recog(7) = uicontrol('Parent',a, ...
    'Units','points', ...
    'BackgroundColor',[1 1 1], ...
    'Position',[280 19.5 20 20], ...
    'Style','frame', ...
    'Tag','frame8');                    %open light 1
recog(8) = uicontrol('Parent',a, ...
    'Units','points', ...
    'BackgroundColor',[1 1 1], ...
    'Position',[280 66 20 20], ...
    'Style','frame', ...
    'Tag','frame9');                    %open light 2
recog(9) = uicontrol('Parent',a, ...
    'Units','points', ...
    'BackgroundColor',[1 1 1], ...
    'Position',[280 113.5 20 20], ...
    'Style','frame', ...
    'Tag','frame10');                   %open light 3
recog(10) = uicontrol('Parent',a, ...
    'Units','points', ...
    'BackgroundColor',[1 1 1], ...
    'Position',[350 19.5 20 20 ], ...
    'Style','frame', ...
    'Tag','frame11');                   %close light 1
recog(11) = uicontrol('Parent',a, ...
    'Units','points', ...
    'BackgroundColor',[1 1 1], ...
    'Position',[350 66 20 20], ...
    'Style','frame', ...
    'Tag','frame12');                   %close light 2
recog(12) = uicontrol('Parent',a, ...
    'Units','points', ...
    'BackgroundColor',[1 1 1], ...
    'Position',[350 113.5 20 20], ...
    'Style','frame', ...
    'Tag','frame13');                   %close light 3

set(a,'vis','on')
drawnow
if stat1==1
    set(recog(9),'BackgroundColor',[1 0 0]);
else
    set(recog(12),'BackgroundColor',[1 0 0]);
end
if stat2==1
```

```
    set(recog(8),'BackgroundColor',[1 0 0]);
  else
    set(recog(11),'BackgroundColor',[1 0 0]);
  end
  if stat3==1
    set(recog(7),'BackgroundColor',[1 0 0]);
  else
    set(recog(10),'BackgroundColor',[1 0 0]);
  end
  %recog(1) == speak&process      recog(2) == input button      recog(3) == exit button
  %recog(4) == show output       recog(5) == help button       recog(6) == show help
  %recog(7) == open light 3      recog(8) == open light 2      recog(9) == open light 1
  %recog(10)== close light 3     recog(11)== close light 2     recog(12)== close light 1

elseif strcmp(action,'INPUT');
    set(recog(1),'string','Speak');
    set(recog(2),'enable','off');
    set(recog(3),'enable','off');
    format long e
    Fs = 8000;
    y = audiorecorder(Fs,8,1);
    t=4;
    recordblocking(y,t);
    stop(y);
    speech = getaudiodata(y,'double');
    set(recog(1),'string','Process');

    [check,text,a]=recsound2(speech);

    set(recog(2),'enable','on');
    set(recog(3),'enable','on');
    set(recog(1),'string','');
    if check==1
      uiwait(msgbox('Please re-input again','error'));
      clear all;
      close all;
      user;
    elseif check==2
      msgbox('Input must contain 3 words','error');
    elseif check==3
      msgbox('Input must contain 3 words','error');
    elseif check==0
      if a(1)==0 | a(2)==0 | a(3)==0
        msgbox('Invalid input','error');
      else
        set(recog(4),'string',strcat('The recognized speech is ... ',text));
      end
    end
```

```
if a==[2 3 4]  %open light one
   if stat1==0
      set(recog(9),'BackgroundColor',[1 0 0]);
      set(recog(12),'BackgroundColor',[1 1 1]);
      portout(1);
      stat1=1;
      save state stat1 stat2 stat3;
   end
elseif a==[2 3 5]  %open light two
   if stat2==0;
      set(recog(8),'BackgroundColor',[1 0 0]);
      set(recog(11),'BackgroundColor',[1 1 1]);
      portout(2);
      stat2=1;
      save state stat1 stat2 stat3;
   end
elseif a==[2 3 6]  %open light three
   if stat3==0;
      set(recog(7),'BackgroundColor',[1 0 0]);
      set(recog(10),'BackgroundColor',[1 1 1]);
      portout(3);
      stat3=1;
      save state stat1 stat2 stat3;
   end
elseif a==[1 3 4]  %close light one
   if stat1==1;
      set(recog(9),'BackgroundColor',[1 1 1]);
      set(recog(12),'BackgroundColor',[1 0 0]);
      portout(4);
      stat1=0;
      save state stat1 stat2 stat3;
   end
elseif a==[1 3 5]  %close light two
   if stat2==1
      set(recog(8),'BackgroundColor',[1 1 1]);
      set(recog(11),'BackgroundColor',[1 0 0]);
      portout(5);
      stat2=0;
      save state stat1 stat2 stat3;
   end
elseif a==[1 3 6]  %close light three
   if stat3==1;
      set(recog(7),'BackgroundColor',[1 1 1]);
      set(recog(10),'BackgroundColor',[1 0 0]);
      portout(6);
      stat3=0;
      save state stat1 stat2 stat3;
   end
end
elseif strcmp(action,'HELP');
```

```
    text=str2mat('*************************************************', ...
                 '(1) Press Input button then speak a command.', ...
                 '(2) Speak time set at 4 seconds.', ...
                 '(3) Press Exit button to exit program.', ...
                 '***************** Thank you ******************', ...
                 'produced by', ...
                 'Mr.Chonpat Tantirapan  41014087 4A', ...
                 'Mr.Chaiyut Thanukaew  41014096 4A');
    set(recog(6),'string',text);
    set(recog(5),'enable','off');
end
```

โปรแกรมแยกพยางค์ของสัญญาณเสียงคำสั่ง

-ฟังก์ชั่น scanspeech:การแยกพยางค์ของสัญญาณเสียงคำสั่ง

แสดงผังงานในการแยกเสียงคำสั่งออกเป็นแต่ละพยางค์

```
function [check,text2,a]=scanspeech(speech);
[P,check]=noisedevice(speech);
if check==0
   P1=(P(2)+P(3))*100;
   P2=(P(4)+P(5))*100;
   for i=1:P1
      word1(i)=speech(i);
   end
   for i=(P1+1):P2
      word2(i-P1)=speech(i);
   end
   for i=(P2+1):length(speech)
      word3(i-P2)=speech(i);
   end
   numword=7;
   a(1)=prob(word1,numword);
   numword=2;
   a(2)=prob(word2,numword);
   numword=9;
   a(3)=prob(word3,numword);
   disp(a);
   if a~=[0 0 0];
      texta1=tex(a(1));
      texta2=tex(a(2));
      texta3=tex(a(3));
      text2=[texta1,' ',texta2,' ',texta3];
   else   a=[0 0 0];
      text2='ERROR! please re-input again';
   end
elseif check==1    text2='ERROR! please re-input again';      a=0;
elseif check==2    text2='ERROR! input must contain 3 words';  a=0;
elseif check==3    text2='ERROR! Input must contain 3 words';  a=0;
end

%=============================================================
function text2=tex(a);
if a==1
   text2='CLOSE ';
elseif a==2
   text2='OPEN ';
elseif a==3
   text2='LIGHT ';
elseif a==4
   text2='ONE ';
elseif a==5
   text2='TWO ';
elseif a==6
   text2='THREE ';
elseif a==0
   text2='Error ';
```

```
end
%=================================================================
%============= device speech by noise ============================
function [P,check]=noisedevice(speech);
inpt=speech-mean(speech);
for i=1:500
   x(i)=abs(inpt(i));
end
th=max(x)*1.4;
if th<=0.02
   th=0.02;
end
inp=abs(inpt)-th;
for i=1:length(inp);
   if inp(i)<0
      inp(i)=0;
   end
end
sample = 200;
vnoloop=floor(length(inpt)/sample);
for k=1:vnoloop
  sum=0;
  for k1=1:sample
    sum=sum + inp( (k-1)*sample+k1 );
  end
  Sum(k)=sum;  %=== Sum is absolute-input that get out noise & windowing(200)
end
num=0;
for i=1:(length(Sum)-1)
   if (Sum(i)==0 & Sum(i+1)>0)
      num=num+1;
      r1(num)=i;
   end
end
num=0;
for i=length(Sum):-1:2
   if Sum(i)==0 & Sum(i-1)>0
      num=num+1;
      r2(num)=i;
   end
end
if isempty(r1) | isempty(r2)
   check=1;  %error input
else
   r=[r1 r2];
   r=sort(r);
   if rem(length(r),2)~=0 | length(r)<6
      P=0;   check=1;   %error input
   elseif length(r)==4
      P=0;   check=2;   %input has 2 word
```

```
elseif length(r)==2
   P=0;  check=3;   %input has 1 word
else
   for i=2:2:length(r)
      n=0;
      for j=r(i-1):r(i)
         n=n+1;
         v(n)=Sum(j);
      end
      ref(i/2)=max(v);
      p1(i/2)=r(i-1);
      p2(i/2)=r(i);
      clear v;
   end
   w=sort(ref);
   n=0;
   for i=length(w):-1:(length(w)-2)
      for j=1:length(ref)
         if w(i)==ref(j)
            n=n+1;
            c1(n)=p1(j);
            c2(n)=p2(j);
         end
      end
   end
   P=[c1 c2];
   P=sort(P);
   check=0;
end
end
```

โปรแกรมการรู้จำเสียงพยางค์เดียว

-ฟังก์ชั่น prob:การรู้จำเสียงพยางค์เดียวโดยวิธี Hidden Markov Model

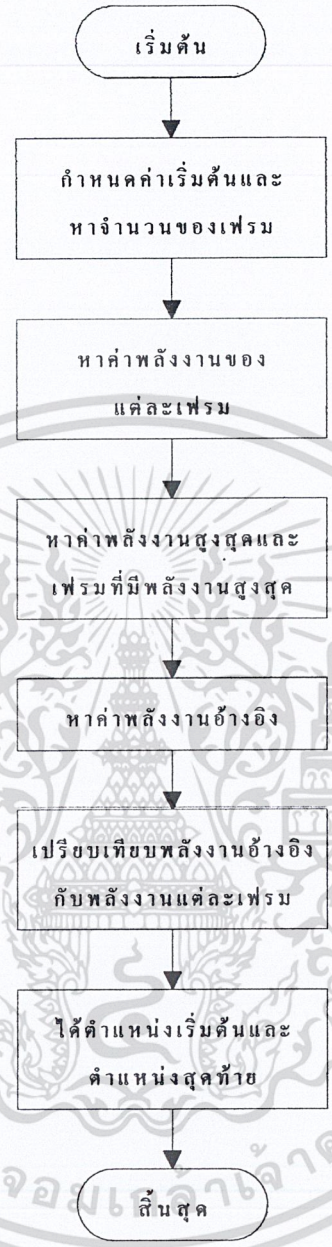
```
function [a]=prob(data,numword);
format long e;
[S,s,w,numberlength,numberweight,g,p,q,E,normS]=analysis(data);
if numword==7
   [bO]=Quantize(w,numword);
   [aN,bN,pil1,N]=model(bO);
   zz=N;
   %load modelclose;
   load 3CLOSE;
   [Pbw] = hmmt(bO,pil1,aN,bN,zz);
   PBW(1)=Pbw;
   %load modelopen;
   load 3OPEN;
   [Pbw] = hmmt(bO,pil1,aN,bN,zz);
   PBW(2)=Pbw;
   if (PBW(1)+PBW(2))~=0
      if max(PBW)==PBW(1);
         a=1;
      elseif max(PBW)==PBW(2);
         a=2;
      end
   else
      a=0;
   end
elseif numword==2
   [bO]=Quantize(w,numword);
   [aN,bN,pil1,N]=model(bO);
   zz=N;
   %load modellight;
   load LIGHT;
   [Pbw] = hmmt(bO,pil1,aN,bN,zz);
   if Pbw~=0
      a=3;
   else
      a=3;
   end
elseif numword==9
   [bO]=Quantize(w,numword);
   [aN,bN,pil1,N]=model(bO);
   zz=N;
   %load modelone;
   load 3ONE;
   [Pbw] = hmmt(bO,pil1,aN,bN,zz);
   PBW(1)=Pbw;
   %load modeltwo;
   load 3TWO;
   [Pbw] = hmmt(bO,pil1,aN,bN,zz);
   PBW(2)=Pbw;
   %load modelthree;
   load 3THREE;
```

```
    [Pbw] = hmmt(bO,pil1,aN,bN,zz);
    PBW(3)=Pbw;
    if (PBW(1)+PBW(2)+PBW(3))~=0
      if max(PBW)==PBW(1);
        a=4;
      elseif max(PBW)==PBW(2);
        a=5;
      elseif max(PBW)==PBW(3);
        a=6;
      end
    else
      a=0;
    end
end
function [Pbw] = hmmt(bO,pil1,aN,bN,N);
Pbw=pil1(N(1))*bN(N(1),bO(1));
for n=1:1:24
  Pbw=Pbw*aN(N(n),N(n+1))*bN(N(n+1),bO(n+1));
end;
```

```
function [S,s,w,numberlength,numberweight,g,p,q,E,normS] = analysis(data);
%g          : gain
%p          : order of lpc
%q          : order of weight cepstrum
%normS   : speech from normalization
normsize=4000;  p=10;  q=12;
[E,a1,b1,s]=detect(data);
[S,a2,b2]=zerocrs(data,a1,b1);
normS=normalize(S,normsize);
data_speech=normS;
[b,c,w,g]=levin_Durbin(normS,p,q);
numberlength=length(S);
numberweight=length(w);
```

โปรแกรมการหาขอบเขตของคำ

- ฟังก์ชั่น detect : การเปรียบเทียบโดยการหาค่าพลังงาน



แสดงผังงานการหาจุดเริ่มต้นและจุดสิ้นสุดของสัญญาณเสียง
โดยการเปรียบเทียบพลังงาน

```
Function [E,a1,b1,s]=detect(data);
a1=1; b1=floor(length(data)/100);
inpt=data-mean(data);
sample = 100;
vnoloop=floor(length(inpt)/sample);
for k=1:vnoloop
  sum=0;
  for k1=1:sample
    sum=sum + abs(inpt( (k-1)*sample+k1 ) );
  end
  E(k)=sum;
end
Emax=max(E);
ref=(1/5)*Emax;
for k=1:vnoloop
  if E(k)==Emax;
    fixenergy=k;
    break;
  end
end
for k=fixenergy:-1:1
  if E(k)<=ref
    a1=k;
    break;
  end
end
if isempty(a1)==1|a1==0
   a1=1;
end
for k1=fixenergy:vnoloop
  if E(k1)<=ref
    b1=k1;
    break;
  end
end
if isempty(b1)==1|b1==vnoloop
   b1=vnoloop-1;
end
num=0;
for i=(a1-1):b1
   for i1=1:sample
      num=num+1;
      s(num)=inpt(i*sample+i1);
   end
end
```

- ฟังก์ชั่น zerocrs : การเปรียบเทียบโดยการหาอัตราการตัดศูนย์

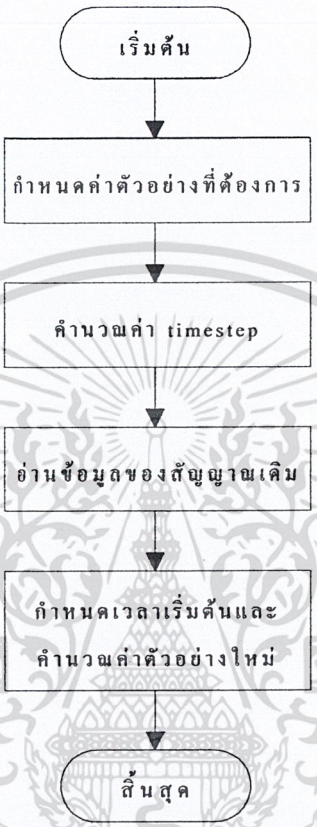
แสดงผังงานการหาจุดเริ่มต้นและจุดสิ้นสุดของสัญญาณเสียง
โดยการหาอัตราการตัดศูนย์

```
function[S,a2,b2]=zerocrs(data,a1,b1);
inpt=data-mean(data);
sample=100;
vnoloop1=floor(length(inpt)/sample);
for k=1:vnoloop1
  sum=0;
  for k1=1:sample
    if k1==1
      sum = sum+abs( sg_n( inpt((k-1)*sample+k1) ) );
    else
      sum = sum+abs( sg_n( inpt((k-1)*sample+k1) ) -...
        sg_n( inpt((k-1)*sample+(k1-1) ) ));
    end
  end
  z(k)=sum/(2*sample);
end
Zmax=max(z);
a=1/5;
for k=1:vnoloop1
  if z(k)==Zmax
    fixzero=k;
    break;
  end
end
refzro=a*Zmax;
%process find bound signal
for k1=a1:-1:1
  if z(k1)<=refzro
    a2= k1;
    break;
  end
end
if isempty(a2)==1
   a2=1;
end
for k2=b1:vnoloop1
  if z(k2)<=refzro
    b2= k2;
    break;
  end
end
n=0;
if isempty(b2)==1|b2==vnoloop1
   b2=vnoloop1-1;
   for k3=a2-1:b2
      for k4=1:sample
        n=n+1;
        S(n)=inpt( (k3*sample)+k4 );
      end
   end
```

```
else
  for k5=a2-1:b2
    for k6=1:sample
      n=n+1;
      S(n)=inpt( (k5*sample)+k6 );
    end
  end
end
%*********************************
%****** function sign *********
%*********************************
function zerc=sg_n(s);
for i=1:length(s)
  if s(i) >0
    zerc=1;
  elseif s(i)==0
    zerc=0;
  else zerc=-1;
  end
end
```

โปรแกรมการนอร์มอลไลซ์สัญญาณ

- ฟังก์ชั่น normalize : การนอร์มอลไลซ์สัญญาณให้ได้จำนวนตัวอย่าง 4000 ค่า



แสดงขั้นตอนการนอมอลไลซ์สัญญาณ

```
function normS = normalize(S,normsize);
normS(1) =S(1);
timestep = (length(S))/(normsize);
time = 1;
j=2;
while j < normsize
  time = time + timestep;
  lowtime = floor(time);
  residue = time-lowtime;
  if (lowtime+1) > length(S)
    S(lowtime+1)=0;
  end
  normS(j)=S(lowtime)+residue*(S(lowtime+1)-S(lowtime));
  j=j+1;
end;
normS(j)=S(length(S));
```

โปรแกรมการหาค่าพารามิเตอร์

- ฟังก์ชั่น levin_Durbin : การหาค่าสัมประสิทธิ์การทำนายเชิงเส้น

- ฟังก์ชั่น ceps : การหาค่าสัมประสิทธิ์เซปสตรัม

- ฟังก์ชั่น cepsw : การหาค่าสัมประสิทธิ์เซปสตรัมถ่วงน้ำหนัก

แสดงผังงานขั้นตอนการเตรียมสัญญาณในการวิเคราะห์

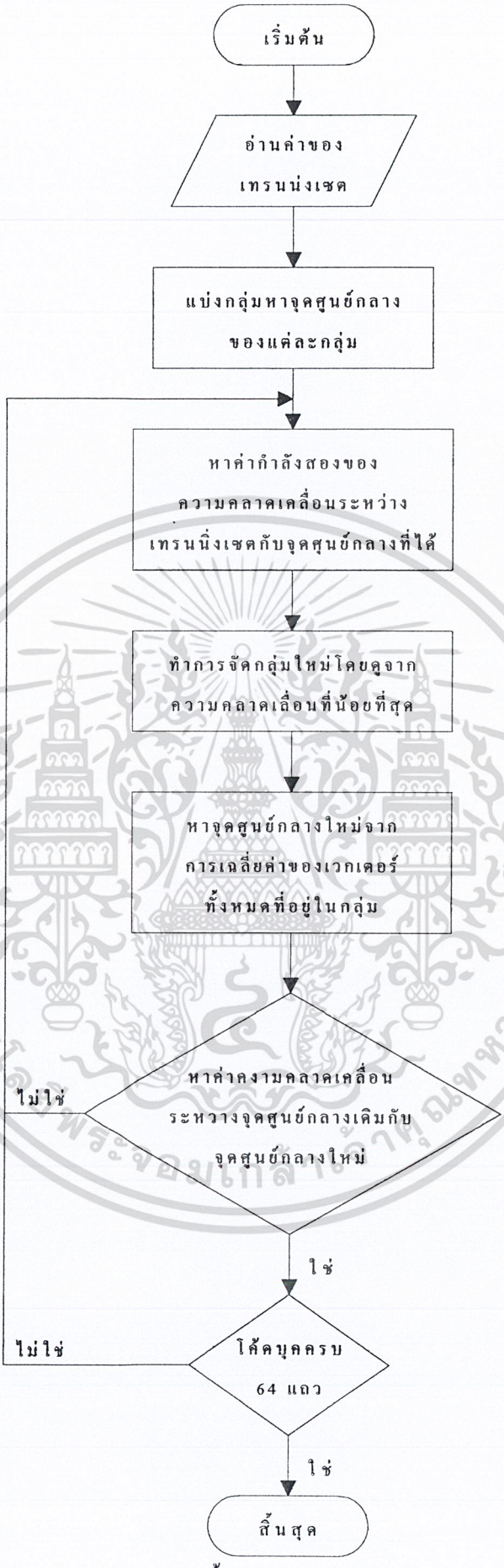
```
function [b,c,w,G]=levin_Durbin(S,N,Q);
%******************** Input **********
%when S = speech waveform
%    N = Order of lpc11
%    Q = Order of CEPSTRUM
%******************** Output *********
%when b = lpc11 Parameter
%    c = Cepstrum
%    w = Weight ceptrum Parameter
%    G = Gain
fs=8000;
sizf=length(S);
format long;
a1=0.9375; as=0; df=0; gh=0;
for i=1:sizf
  if i==1
    s(i)=S(i);
  else
    s(i)=S(i)-a1*S(i-1);
  end
end
fram=(240/fs);   %0.03
t=(160/fs);      %0.02
shift=(80/fs);   %0.01
z=length(s)/160;
le=length(s)-((floor(z)-1)*160);
if le==fram*fs   %fram*fs=240
  frame=floor(z);
else
  frame=floor(z)-1;
end
for l=1:frame
  v=0;
  for n=(((l-1)*160)+1):(((l-1)*160)+240)
    v=v+1;
    as=as+1;
    x(l,v)=s(n)*(0.54-(0.46*cos(2*pi*v/(fram*fs))));
    h(as)=x(l,v);
  end
  [a,g,R]=lpc11(h,N);  %N: Order of lpcl1
  for j=1:N+1
    df=df+1;
    b(df)=a(j);
    r(l,1)=R(1);
    G(l)=g;
  end

  [ce,cw]=ceps(a,g,N,Q);
  for i1=1:Q
    gh=gh+1;
```

```
    c(gh)=ce(i1);
    w(gh)=cw(i1);
  end
end
if le < 240
  v=0;
  for n=(frame*160)+1: (frame*160)+le
    v=v+1; as=as+1;
    x(frame+1,v)=s(n)*(0.54-(0.46*cos(2*pi*v/le)));
    h(as)=x(frame+1,v);
  end
  frame=frame+1;
  l=l+1;
  [a,g,R]=lpc11(h,N);
  for j=1:N+1;
    A(l,j)=a(j);
    df=df+1;
    b(df)=a(j);
    r(l,1)=R(1);
    G(l)=g;
  end
  [ce,cw]=ceps(a,g,N,Q);
  for i=1:Q
    gh=gh+1;
    c(gh)=ce(i);
    w(gh)=cw(i);
  end
end
%*************** lpc Function ************
function [a,g,R]=lpc11(h,N);
error(nargchk(1,2,nargin))
h=h;
[r,c]=size(h);
if (c>1)&(r==1)
  h=h(:);
end
numsigs = size(h,2);
if nargin<2   N=size(h,1)-1;  end
if (N>size(h,1)-1),
  h(N+1,:)=zeros(1,numsigs);
end
R = flipud(fftfilt(h,conj(flipud(h))))/length(h);
a = levinson(R,N);
g = real(sqrt( sum((a.').*R(1:N+1,:)) ));
```

```
function [ce,cw]=ceps(a,g,N,Q);
ce(1) = 0.5*log(g);
ce(2) = a(2);
for i=N+2:Q
  a(i)=0;
end
for m=3:Q
  sum=0;
  for k=1:m-1
    sum=sum+(k*ce(k)*a(m-k));
  end
  cep_sum(m) = (1/m)*sum;

  if m <= N+1
    ce(m) = a(m)+cep_sum(m);
  else
    ce(m) = cep_sum(m);
  end
end
cw = cepsw(Q,ce);
%=================================================================
function cw = cepsw(Q,ce);
for m = 1:Q
  cw(m) = ce(m)*(1+( (Q/2) * sin (pi* (m/Q) ) ));
end
```

โปรแกรมการหาโค้ดบุค

- ฟังก์ชั่น codebook64 : การหาโค้ดบุค 64



ผังงานแสดงขั้นตอนการสร้างโค้ดบุค

```
function [cb64]=codebook64(tset);
%load tset;
TrainingSetFile=tset;

format long;
VectorDimension = 12;
CodeBookNumber = 64;
RejectValue=0.0001;
VTD=2*RejectValue;
%================ Find Number of Frame ================
NumberOfFrame = ((length(TrainingSetFile)/VectorDimension));
%======== Ramdom Cenctoid From TrainingSetFile ========
x = floor(NumberOfFrame.*rand(1,CodeBookNumber))+1;
for nc=1:CodeBookNumber
  for nv=1:VectorDimension
    OldCentroid(((nc-1).*VectorDimension)+nv)=...
      TrainingSetFile(((x(nc)-1).*VectorDimension)+nv);
  end
end
k=0;
%======================== Loop ========================
while VTD>RejectValue
%==================== Find Codebook ====================
  for nf=1:NumberOfFrame
    for nc=1:CodeBookNumber
      TotalDistance=0;
      distance=0;
      for nv=1:VectorDimension
        distance=(TrainingSetFile(((nf-1).*VectorDimension)+nv)-...
          OldCentroid(((nc-1).*VectorDimension)+nv)).^2;
        TotalDistance=TotalDistance+distance;
      end
      VectorDistance(nc)=TotalDistance;
    end
    [MinDistance(nf),MinIndex(nf)]=min(VectorDistance);
  end
%=================== Find New Centroid ===================
  for nc=1:CodeBookNumber
    Cnum=0;
    for nv=1:VectorDimension
      NewCentroid(((nc-1)*VectorDimension)+nv)=0;
    end
    for nf=1:NumberOfFrame
      if nc==MinIndex(nf)
        Cnum=Cnum+1;
        for nv=1:VectorDimension
          NewCentroid(((nc-1)*VectorDimension)+nv)=...
            NewCentroid(((nc-1)*VectorDimension)+nv)+...
            TrainingSetFile(((nf-1)*VectorDimension)+nv);
        end
```

```
        end
      end
      if Cnum~=0
        for nv=1:VectorDimension
          NewCentroid(((nc-1)*VectorDimension)+nv) =...
            NewCentroid(((nc-1)*VectorDimension)+nv)./Cnum;
        end
      else
        for nv=1:VectorDimension
          NewCentroid(((nc-1)*VectorDimension)+nv) =...
            NewCentroid(((nc-1)*VectorDimension)+nv);
        end
      end
    end
%=================== Check Distance ======================
    d=NewCentroid-OldCentroid;
    e=d.^2;
    VTD=sum(e)./CodeBookNumber;
    k=k+1;
    CVTD64(k)=VTD;
%=========== Copy Old Centroid to New Centroid ===========
    OldCentroid=NewCentroid;
end
cb64=NewCentroid;
%save codebook64 cb64
```

โปรแกรมการควอนไตซ์เซชั่น

- ฟังก์ชั่น quantize : การจัดระดับค่าพารามิเตอร์

แสดงขั้นตอนการควอนไตซ์เซชัน

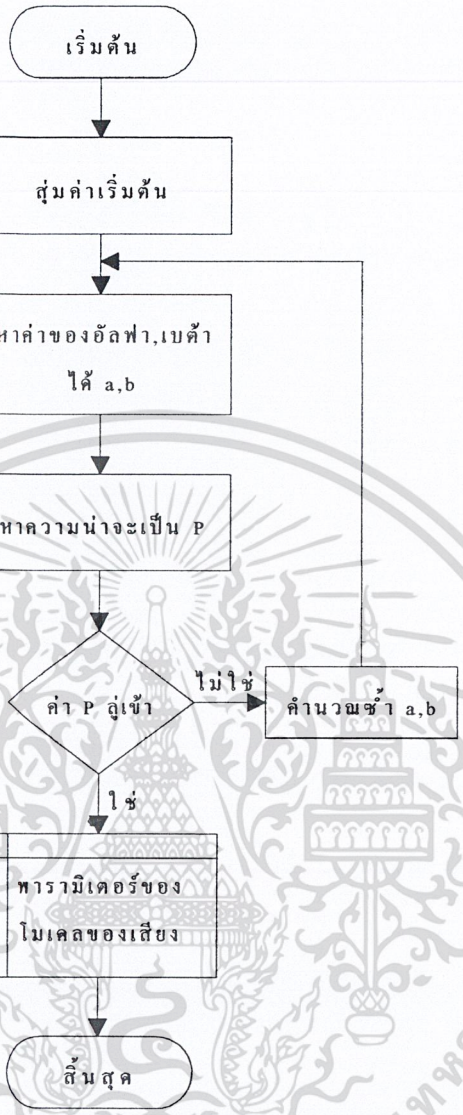
```
function [bO]=Quantize(w,numword);
if numword==2
   load ('cblight.mat','cb64');
elseif numword==7
   load ('cbword7.mat','cb64');
elseif numword==9
   load ('cbword9.mat','cb64');
end
i=1;
for n=1:25
  k=1;
  j=1;
  for m=1:64
    d(1:12)=w(i:i+11)-cb64(j:j+11);
    d=sum(d.^2)/length(d);
    f(1,k)=d;
    k=k+1;
    j=j+12;
  end
  c=find(f==min(f));
  if length(c)~=0
     bO(1,n)=c(1);
     i=i+12;
  else
     bO(1,n)=c;
     i=i+12;
  end
end
clear cb64;
```

โปรแกรมการหาแบบจำลองอ้างอิงโดยวิธีฮิดเดนมาร์คอฟ

- ฟังก์ชั่น model : การหาแบบจำลองอ้างอิงของเสียง

โปรแกรมการหาลำดับสเตทที่ดีที่สุดของการใช้วิธีฮิดเดนมาร์คอฟ

- ฟังก์ชั่น veterbi : การหาลำดับสเตทที่ดีที่สุดของการใช้วิธีฮิดเดนมาร์คอฟ



แสดงขั้นตอนการสร้างแบบจำลองอ้างอิง

```
function [aN,bN,pil1,N]=model(bO);
%function find model
format long e;
%       Hidden markov model section
%find pil1 a1 a2 a3 a4 a5 a6 b1 b2 b3 b4 b5 b6
pil1 = [1 0 0 0 0 0];
a1 = [0.3 0.5 0.2 0 0 0];
a2 = [0 0.4 0.3 0.3 0 0];
a3 = [0 0 0.4 0.2 0.4 0];
a4 = [0 0 0 0.7 0.2 0.1];
a5 = [0 0 0 0 0.5 0.5];
a6 = [0 0 0 0 0 1];
b1 = [1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 ...
  1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 ...
  1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 ...
  1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64];
b2 = [1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 ...
  1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 ...
  1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 ...
  1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64];
b3 = [1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 ...
  1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 ...
  1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 ...
  1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64];
b4 = [1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 ...
  1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 ...
  1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 ...
  1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64];
b5 = [1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 ...
  1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 ...
  1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 ...
  1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64];
b6 = [1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 ...
  1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 ...
  1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 ...
  1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64 1/64];
n = [1];
%save a;
[pil1,aN,a1,a2,a3,a4,a5,a6,bN,b1,b2,b3,b4,b5,b6,Pbw] = ...
hmm(bO,pil1,a1,a2,a3,a4,a5,a6,b1,b2,b3,b4,b5,b6);
PBW(n)=Pbw;
[pil1,aN,a1,a2,a3,a4,a5,a6,bN,b1,b2,b3,b4,b5,b6,Pbw] = ...
hmm(bO,pil1,a1,a2,a3,a4,a5,a6,b1,b2,b3,b4,b5,b6);
PBW(n+1)=Pbw;
while (PBW(n)/PBW(n+1))<=0.9999
%if (PBW(n)/PBW(n+1))>0.9999
  n=n+1;
  [pill,aN,a1,a2,a3,a4,a5,a6,bN,b1,b2,b3,b4,b5,b6,Pbw] = ...
  hmm(bO,pil1,a1,a2,a3,a4,a5,a6,b1,b2,b3,b4,b5,b6);
  PBW(n+1)=Pbw;
```

```
end;
%save z n PBW;
%find best state sequence
[N] = veterbi(bO,pil1,aN,bN);

%==================== start begin model ====================
%============================================================
function [pil1,aN,a1,a2,a3,a4,a5,a6,bN,b1,b2,b3,b4,b5,b6,Pbw] = ...
hmm(bO,pil1,a1,a2,a3,a4,a5,a6,b1,b2,b3,b4,b5,b6);

% function find model
% part calculator value begin Alpha forward 3.4.6
A1(1) = pil1(1)*b1(bO(1));
A2(1) = pil1(2)*b2(bO(1));
A3(1) = pil1(3)*b3(bO(1));
A4(1) = pil1(4)*b4(bO(1));
A5(1) = pil1(5)*b5(bO(1));
A6(1) = pil1(6)*b6(bO(1));

for n = 1:1:24
%part calculator Alpha forward 3.4.5
  A1(n+1) = ((A1(n)*a1(1))+(A2(n)*a2(1))+(A3(n)*a3(1))+(A4(n)*a4(1))+...
        (A5(n)*a5(1))+(A6(n)*a6(1)))*b1(bO(n+1));
  A2(n+1) = ((A1(n)*a1(2))+(A2(n)*a2(2))+(A3(n)*a3(2))+(A4(n)*a4(2))+...
        (A5(n)*a5(2))+(A6(n)*a6(2)))*b2(bO(n+1));
  A3(n+1) = ((A1(n)*a1(3))+(A2(n)*a2(3))+(A3(n)*a3(3))+(A4(n)*a4(3))+...
        (A5(n)*a5(3))+(A6(n)*a6(3)))*b3(bO(n+1));
  A4(n+1) = ((A1(n)*a1(4))+(A2(n)*a2(4))+(A3(n)*a3(4))+(A4(n)*a4(4))+...
        (A5(n)*a5(4))+(A6(n)*a6(4)))*b4(bO(n+1));
  A5(n+1) = ((A1(n)*a1(5))+(A2(n)*a2(5))+(A3(n)*a3(5))+(A4(n)*a4(5))+...
        (A5(n)*a5(5))+(A6(n)*a6(5)))*b5(bO(n+1));
  A6(n+1) = ((A1(n)*a1(6))+(A2(n)*a2(6))+(A3(n)*a3(6))+(A4(n)*a4(6))+...
        (A5(n)*a5(6))+(A6(n)*a6(6)))*b6(bO(n+1));
end;

%part calculator Baum-Welch forward 3.4.3
Pbw = A1(25) + A2(25) + A3(25) + A4(25) + A5(25) + A6(25);

%part calculator Backward Probability forward 3.4.13
B1(25)=1; B2(25)=1; B3(25)=1; B4(25)=1; B5(25)=1; B6(25)=1;
for n=25:-1:2
  B6(n-1) = (a6(1)*b1(bO(n))*B1(n)) + (a6(2)*b2(b0(n))*B2(n)) + (a6(3)*b3(bO(n))*B3(n)) +...
        (a6(4)*b4(bO(n))*B4(n)) + (a6(5)*b5(bO(n))*B5(n)) + (a6(6)*b6(bO(n))*B6(n));
  B5(n-1) = (a5(1)*b1(bO(n))*B1(n)) + (a5(2)*b2(b0(n))*B2(n)) + (a5(3)*b3(bO(n))*B3(n)) +...
        (a5(4)*b4(bO(n))*B4(n)) + (a5(5)*b5(bO(n))*B5(n)) + (a5(6)*b6(bO(n))*B6(n));
  B4(n-1) = (a4(1)*b1(bO(n))*B1(n)) + (a4(2)*b2(b0(n))*B2(n)) + (a4(3)*b3(bO(n))*B3(n)) +...
        (a4(4)*b4(bO(n))*B4(n)) + (a4(5)*b5(bO(n))*B5(n)) + (a4(6)*b6(bO(n))*B6(n));
  B3(n-1) = (a3(1)*b1(bO(n))*B1(n)) + (a3(2)*b2(b0(n))*B2(n)) + (a3(3)*b3(bO(n))*B3(n)) +...
        (a3(4)*b4(bO(n))*B4(n)) + (a3(5)*b5(bO(n))*B5(n)) + (a3(6)*b6(bO(n))*B6(n));
  B2(n-1) = (a2(1)*b1(bO(n))*B1(n)) + (a2(2)*b2(b0(n))*B2(n)) + (a2(3)*b3(bO(n))*B3(n)) +...
```

```
        (a2(4)*b4(bO(n))*B4(n)) + (a2(5)*b5(bO(n))*B5(n)) + (a2(6)*b6(bO(n))*B6(n));
  B1(n-1) = (a1(1)*b1(bO(n))*B1(n)) + (a1(2)*b2(bO(n))*B2(n)) + (a1(3)*b3(bO(n))*B3(n)) +...
        (a1(4)*b4(bO(n))*B4(n)) + (a1(5)*b5(bO(n))*B5(n)) + (a1(6)*b6(bO(n))*B6(n));
end;

%part calculator State Transition forward 3.4.12
a11s = 0; a12s = 0; a13s = 0; a1t = 0;
a22s = 0; a23s = 0; a24s = 0; a2t = 0;
a33s = 0; a34s = 0; a35s = 0; a3t = 0;
a44s = 0; a45s = 0; a46s = 0; a4t = 0;
a55s = 0; a56s = 0; a5t = 0;
a66s = 0; a6t = 0;

for n=2:1:25
  a11s = a11s + (A1(n-1) * a1(1) * b1(bO(n)) * B1(n));
  a12s = a12s + (A1(n-1) * a1(2) * b2(bO(n)) * B2(n));
  a13s = a13s + (A1(n-1) * a1(3) * b3(bO(n)) * B3(n));
  a1t = a1t + (A1(n-1) * B1(n-1));

  a22s = a22s + (A2(n-1) * a2(2) * b2(bO(n)) * B2(n));
  a23s = a23s + (A2(n-1) * a2(3) * b3(bO(n)) * B3(n));
  a24s = a24s + (A2(n-1) * a2(4) * b4(bO(n)) * B4(n));
  a2t = a2t + (A2(n-1) * B2(n-1));

  a33s = a33s + (A3(n-1) * a3(3) * b3(bO(n)) * B3(n));
  a34s = a34s + (A3(n-1) * a3(4) * b4(bO(n)) * B4(n));
  a35s = a35s + (A3(n-1) * a3(5) * b5(bO(n)) * B5(n));
  a3t = a3t + (A3(n-1) * B3(n-1));

  a44s = a44s + (A4(n-1) * a4(4) * b4(bO(n)) * B4(n));
  a45s = a45s + (A4(n-1) * a4(5) * b5(bO(n)) * B5(n));
  a46s = a46s + (A4(n-1) * a4(6) * b6(bO(n)) * B6(n));
  a4t = a4t + (A4(n-1) * B4(n-1));

  a55s = a55s + (A5(n-1) * a5(5) * b5(bO(n)) * B5(n));
  a56s = a56s + (A5(n-1) * a5(6) * b6(bO(n)) * B6(n));
  a5t = a5t + (A5(n-1) * B5(n-1));

  a66s = a66s + (A6(n-1) * a6(6) * b6(bO(n)) * B6(n));
  a6t = a6t + (A6(n-1) * B6(n-1));
end;
a1 = [a11s/a1t  a12s/a1t  a13s/a1t 0      0      0    ];
a2 = [0       a22s/a2t  a23s/a2t  a24s/a2t 0      0    ];
a3 = [0       0       a33s/a3t  a34s/a3t  a35s/a3t 0    ];
a4 = [0       0       0       a44s/a4t  a45s/a4t  a46s/a4t];
a5 = [0       0       0       0       a55s/a5t  a56s/a5t];
a6 = [0       0       0       0       0       a66s/a6t];
aN = [a1; a2; a3; a4; a5; a6];

% part find value forward 3.4.15
```

```
for n=1:64
  b1N(n)=0; b2N(n)=0; b3N(n)=0; b4N(n)=0; b5N(n)=0; b6N(n)=0;
end
b1Total=0; b2Total=0; b3Total=0; b4Total=0; b5Total=0; b6Total=0;
for n=1:25
  b1Total = b1Total + (A1(n) * B1(n));
  b2Total = b2Total + (A2(n) * B2(n));
  b3Total = b3Total + (A3(n) * B3(n));
  b4Total = b4Total + (A4(n) * B4(n));
  b5Total = b5Total + (A5(n) * B5(n));
  b6Total = b6Total + (A6(n) * B6(n));
  for nn=1:64
    if bO(n)==nn;
      b1N(nn) = b1N(nn) + (A1(n) * B1(n));
      b2N(nn) = b2N(nn) + (A2(n) * B2(n));
      b3N(nn) = b3N(nn) + (A3(n) * B3(n));
      b4N(nn) = b4N(nn) + (A4(n) * B4(n));
      b5N(nn) = b5N(nn) + (A5(n) * B5(n));
      b6N(nn) = b6N(nn) + (A6(n) * B6(n));
    end;
  end;
end;
b1 = b1N/b1Total;
b2 = b2N/b2Total;
b3 = b3N/b3Total;
b4 = b4N/b4Total;
b5 = b5N/b5Total;
b6 = b6N/b6Total;
bN = [b1;b2;b3;b4;b5;b6];
```

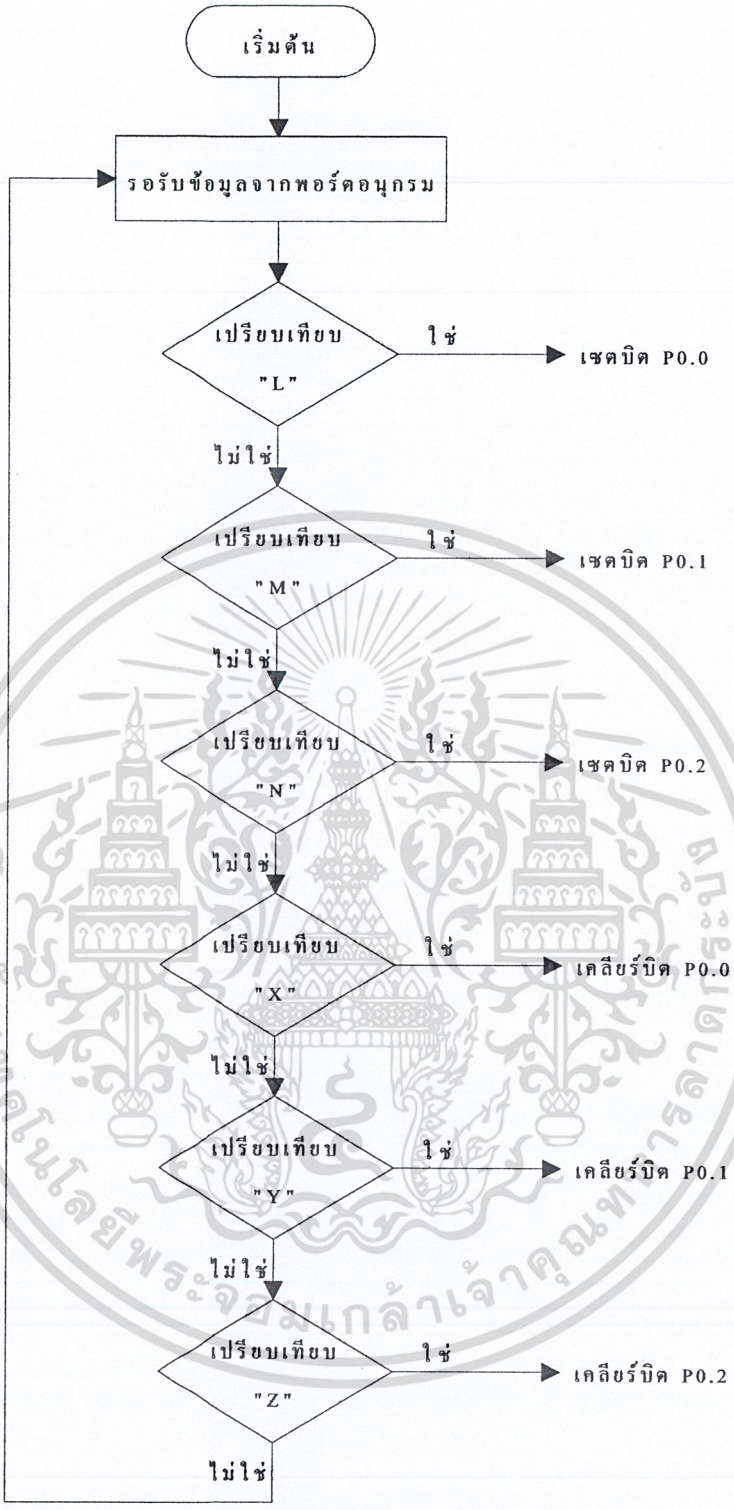
```
function [N] = veterbi(bO,pil1,aN,bN);
% function find best state sequence
% N   : best state sequence
% bO  : observation
% pil1: probability start
% aN  : state sequence probability
% bN  : observation probability

R1(1)=pil1(1)*bN(1,(bO(1)));
R2(1)=0;
R3(1)=0;
R4(1)=0;
R5(1)=0;
R6(1)=0;
R=[R1 R2 R3 R4 R5 R6];
T=max(R);
v=find(R==T);
N(1)=v;

for n=1:24
  r1=R1*aN(v,1)*bN(1,(bO(n+1)));
  r2=R1*aN(v,2)*bN(2,(bO(n+1)));
  r3=R1*aN(v,3)*bN(3,(bO(n+1)));
  r4=R1*aN(v,4)*bN(4,(bO(n+1)));
  r5=R1*aN(v,5)*bN(5,(bO(n+1)));
  r6=R1*aN(v,6)*bN(6,(bO(n+1)));
  R=[r1 r2 r3 r4 r5 r6];
  T=max(R);
  if (R(1)+R(2)+R(3)+R(4)+R(5)+R(6))==0
    N(n+1)=N(n);
  else
    v=find(R==T);
    N(n+1)=v;
  end
  R1=T;
end
```

โปรแกรมส่งข้อมูลไปยังพอร์ตอนุกรม

-ฟังก์ชั่น portout:การส่งข้อมูลไปยังพอร์ตอนุกรม

```
function portout(code);
s = serial('COM1');
set(s,'BaudRate',9600);%usually set at 9600
fopen(s)
if code==1
   fwrite(s,'L')
elseif code==2
   fwrite(s,'M')
elseif code==3
   fwrite(s,'N')
elseif code==4
   fwrite(s,'X')
elseif code==5
   fwrite(s,'Y')
elseif code==6
   fwrite(s,'Z')
end
fclose(s)
delete(s)
clear s
```

โปรแกรมการแสดงผลของไมโครคอนโทรลเลอร์ MCS-51



ผังงานแสดงการทำงานของไมโครคอนโทรลเลอร์ MCS-51

```
                ORG 0000H
                JMP MAIN
; SET BAUD RATE AND SET DATA FORMAT
; BAUD RATE  9600  bps
; DATA BIT  8 bit
; PARITY CHECK  none
; STOP BIT  1 bit
        MAIN:   MOV PCON,#00H
                MOV SCON,#50H
                MOV TH1,#0FDH
                MOV TMOD,#20H
                SETB TR1
                MOV P0,#0FFH
        SRART:  CALL RECEIVE
                MOV A,SBUF
        TURN_ON1:CJNE A,#'L',TURN_ON2       ; TURN ON LED P0.0
                CLR P0.0
        TURN_ON2:CJNE A,#'M',TURN_ON3       ; TURN ON LED P0.1
                CLR P0.1
        TURN_ON3:CJNE A,#'N',TURN_OFF1      ; TURN ON LED P0.2
                CLR P0.2
        TURN_OFF1:CJNE A,#'X',TURN_OFF2     ; TURN OFF LED P0.0
                SETB P0.0
        TURN_OFF2:CJNE A,#'Y',TURN_OFF3     ; TURN OFF LED P0.1
                SETB P0.1
        TURN_OFF3:CJNE A,#'Z',START         ; TURN OFF LED P0.2
                SETB P0.2
                JMP START

; CHECK DATA FROM RS232
        RECEIVE:JNB RI,$
                CLR RI
                RET

                END
```

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้คงจะสำเร็จลุล่วงไปได้ด้วยดีไม่ได้ ถ้าไม่ได้คำแนะนำและความช่วยเหลือในการอำนวยความสะดวกในการทดลองจากอาจารย์ที่ปรึกษาคือ ศ.ดร.วิวัฒน์ กิรานนท์ และ ผศ.วิภา แสงพิสิทธิ์ ผู้จัดทำขอขอบพระคุณอาจารย์ทั้ง 2 ท่าน ไว้ ณ ที่นี้ด้วย

ขอบพระคุณคุณพ่อ คุณแม่ที่สนับสนุนทุนการศึกษาและคอยให้กำลังใจตลอดมา ขอบคุณเพื่อนๆ และน้องๆ มีรายชื่อดังต่อไปนี้ เพื่อนจี๊ด เพื่อนแหม่ม เพื่อนต้า เพื่อนติ๊ก เพื่อนถัง เพื่อนป้อก เพื่อนหลิน เพื่อนนุ๊ก น้องอิ๋ว น้องมน น้องปุ๊บ น้องปู ที่มาบันทึกเสียงให้เพื่อใช้ในการทดลอง หากขาดเพื่อนและน้องเหล่านี้การทดลองคงจะสำเร็จไม่ได้ ขอบคุณจากใจจริง

บรรณานุกรม

[1.] รศ.ดร.มนัส สังวรศิลป์ และ วรรัตน์ ภัทรอมรกุล, "คู่มือการใช้งาน MATLAB ฉบับสมบูรณ์", สำนักพิมพ์อินโฟร์เพลส 2543.

[2.] Lawrene Rabiner and Biing-Hwang Jung, "Fundamntal of Speech Recognition", New Jersey:Prentice Hall 1993.