การบันทึกสัญญาณโสยังพูดและเล่นกลับ

VOICE RECORDING AND PLAYBACK

โดย

นายยุทธกร	รมยเวศม์	34162166
นายสมนึก	แจ้งไพร	34162175
นายสุเทพ	ก่อเกิด	34162177

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาอุตสาหกรรมศาสตร์บัณฑิต

สาขาคอมพิวเตอร์อุตสาหกรรม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2535

ภาควิชา เทคโนโลยีการวัดคุมทางอุตสาหกรรม

สาขา เทคโนโลยีคอมพิวเตอร์อุตสาหกรรม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การบันทึกสัญญาณเสียงพูดและเล่นกลับ

VOICE RECORDING AND PLAYBACK

ผู้จัดทำ

1. นาย ยุทธกร รมยเวศม์ 34162166
2. นาย สมนึก แจ้งไพร 34162175
3. นาย สุเทพ ก่อเกิด 34162177

..... อาจารย์ที่ปรึกษา

(อาจารย์ ภากร หุตะสังกาศ)

บทคัดย่อ

ในสมัยก่อนการที่จะให้เครื่องจักรอุปกรณ์พูดได้ อาศัยหลักการของการอัดเสียงลงเทป แต่ในปัจจุบัน ราคาของชิ้นส่วนอุปกรณ์ ดิจิตอลและหน่วยความจำของไมโครคอมพิวเตอร์มีราคาถูกลงมาก การประยุกต์ใช้ อุปกรณ์ อิเลกทรอนิกส์และดิจิตอล มาใช้ในการเก็บ และรวบรวมคำพูดลงในหน่วยความจำ จึงเป็นเรื่องที่เป็นไปได้ ปริญญานิพนธ์ฉบับนี้ เป็นการแสดงถึงการเก็บเสียงพูด ไว้ในหน่วยความจำโดยใช้ไอซีเบอร์ UM5100(Voice Processor) เป็นตัวประมวลผล ซึ่งหลักการของมันก็คือ แปลงสัญญาณ จาก สัญญาณเสียง ซึ่งเป็นสัญญาณ อนาลอก โดยการ แซมปลิ้ง สัญญาณอินพุท ให้เป็นสัญญาณดิจิตอล โดยใช้ เดลต้ามอดดูเลชั่น (dalta-modulation) และนำสัญญาณ ที่ได้ ไปเก็บไว้ในหน่วยความจำ ในส่วนของการเล่นกลับก็จะเป็นลักษณะเดียวกัน

ในการนำโครงงานนี้ไปประยุกต์ใช้อื่นๆเป็นไปได้อย่างกว้างขวาง เช่นในระบบเตือนภัย แทนที่จะใช้สัญญานเตือนภัยเราก็หันมาใช้เสียงพูดแทน หรืออาจนำมาใช้ในระบบตอบรับทางโทรศัพท์โดยอัตโนมัติก็เป็นไปได้

ABSTRACT

In the part, talking magnetic tape to record the sound. Presently, the price of components of digital and memory of micro-computer has been dropped dratically. It is posible to use electronic and digital componnents in voice recording. This thesis explains voice recording and playback by using UM 5100 voice processor.

There are numerous application for the speech system to be used to good effect. For example alarm system could incorporate playback modules, with differant pre-recorded messages or warning up and alarm condition. Seat belt or vabal fuel and speed warnings could be installed in motor vehicles and perhaps a telephone answering service could on this system.

สารบัญ

บทนำ

ตั้งแต่เทคโนโลยีทางด้านสารกึ่งตัวนำได้รับการพัฒนาขึ้นมา จนสามารถสร้างเครื่องจักรที่ใช้ในการคำนวนที่มีประสิทธิภาพสูงสุดเท่าที่มนุษย์เคยรู้จักกันซึ่งเรียกว่า ดิจิตอลคอมพิวเตอร์ (digital computer) ดิจิตอลคอมพิวเตอร์ได้รับการวิวัฒนาการไปมาก และมีแนวโน้มถึงการเปลี่ยนแปลงใหม่ๆ ในอนาคต ดิจิตอลคอมพิวเตอร์ได้นำไปใช้งานในด้านต่างๆ อย่างมากมาย เช่น การนำไปใช้งานทางด้านธุรกิจ ทางอุตสาหกรรม เป็นต้น ซึ่งดิจิตอลคอมพิวเตอร์จำเป็นจะต้องควบคุมด้วยการป้อนข้อมูลเข้าทางคีย์บอร์ดหรือบัตรเจาะรู แล้วแสดงเอาท์พุทด้วยตัวอักษรบนจอ,บนกระดาษหรือทำให้อุปกรณ์อื่น ๆ ที่ควบคุมด้วยคอมพิวเตอร์ทำงาน แต่ในช่วงไม่กี่ปีมานี้คอมพิวเตอร์ได้รับการพัฒนาไปใช้ในการสร้างเสียง ซึ่งจะให้เอาท์พุทเป็นเสียงพูดของมนุษย์แทนที่จะเป็นตัวอักษรบนจอภาพหรือกระดาษในระยะแรก ๆ นั้นทำได้แต่เพียงทำให้คอมพิวเตอร์พูดได้อย่างมนุษย์ ซึ่งต้องใช้หน่วยความจำในการเก็บข้อมูลคำพูดอย่างมากมาย ซึ่งต่อไปในอนาคตด้วยเทคโนโลยีชั้นสูงจะสามารถที่จะสร้างหน่วยความจำที่มีขนาดเล็ก แต่เก็บข้อมูลได้อย่างมากมาย ในปัจจุบันนี้ผู้เชี่ยวชาญทางด้านคอมพิวเตอร์ได้มุ่งความสนใจไปที่การวิเคราะห์และสังเคราะห์เสียงพูดของมนุษย์ โดยทำให้มนุษย์สามารถที่จะสั่งให้คอมพิวเตอร์ทำงานด้วยเสียงพูดแทนที่จะป้อนข้อมูลเข้าทางคีย์บอร์ด และสามารถทำให้มนุษย์พูดจาโต้ตอบกับคอมพิวเตอร์ได้ ซึ่งจะนำเทคนิคอันนี้ไปใช้ในหุ่นยนต์ ต่อไปในอนาคตมนุษย์จะได้รับความสะดวกสบายอย่างมากมาย ในที่นี้จะกล่าวถึงการนำไอซีสำเร็จรูปมาใช้ในการสร้างเสียงพูด ซึ่งพอจะเป็นแนวทางให้ผู้สนใจได้รับประโยชน์บ้างไม่มากก็น้อย

ในการวิเคราะห์เสียงพูด จะต้องมีการเปลี่ยนสัญญาณจากอนาล็อกไปเป็นสัญญาณดิจิตอล (Analog to digital converter) ซึ่งสามารถทำได้หลายวิธี ในที่นี้ใช้ CVSD (continuous variable slope delta modulation) ทั้งนี้เนื่องจากจะประหยัดหน่วยความจำในการเก็บข้อมูล

วัตถุประสงค์ของปริญญานิพนธ์

เพื่อเป็นการแสดงให้เห็นว่าการวิเคราะห์สัญญาณเสียงพูด โดยอาศัยลักษณะธรรมชาติของสัญญาณของเสียงพูด เราสามารถใช้เทคนิคการแปลงสัญญาณทางอนาลอกเป็นดิจิตอล โดยใช้ไอซีสำเร็จรูปที่มีหน้าที่แปลงสัญญาณเป็นอนาลอกเป็นดิจิตอล แล้วแปลงกลับจากดิจิตอลเป็นอนาลอกร่วมกับไมโครคอมพิวเตอร์ซึ่งกำลังเป็นที่นิยมอย่างแพร่หลายในปัจจุบันนี้ทั้งนี้เพื่อลดความยุ่งยากและความสิ้นเปลือง หากจะใช้วงจรแปลง ADC/DAC เหมือนแต่เดิม และในที่นี้จะใช้ไอซีสำเร็จเบอร์ UM 5100 เป็นตัวแปลงสัญญาณ อนาลอกเป็นดิจิตอล และแปลงกลับจากดิจิตอลเป็นอนาลอก ในการประมวลผลสัญญาณเสียงพูดนี้เราจะใช้เทคนิคการประมวลผลสัญญาณดิจิตอล

ขอบเขตของปริญญานิพนธ์

สามารถนำสัญญาณเสียงเก็บไว้ในอุปกรณ์หน่วยความจำ สารกึ่งตัวนำประเภท SRAM หรือ EPROM โดยใช้ VICEPROCESSOR เบอร์ UM 5100 เป็นตัวประมวลผลและสามารถ เลือกเล่นกลับหรือบันทึกข้อมูล ไว้ใน SRAM หรือเลือกเล่นกลับจาก EPROM ในตำแหน่งต่างๆ ได้โดยเลือกจากคีย์บอร์ดของไมโครคอมพิวเตอร์ โดยจะแสดงรายการเป็น MENU ออกแสดงทางจอภาพของไมโครคอมพิวเตอร์อีกเช่นกัน

บทที่ 1

สัญญาณเสียงพูด

จุดมุ่งหมายของสัญญาณเสียงพูด ก็คือการติดต่อสื่อสารข้อมูล ในระบบการสื่อสารข้อมูลโดยใช้เสียงพูดนั้น สัญญาณเสียงจะถูกส่ง ถูกเก็บและประมวลผลได้มากมายหลายวิธี แต่ต้องคำนึงถึงข้อจำกัดทางเทคนิคต่าง ๆ ด้วย โดยทั่วไประบบต่าง ๆ จะคำนึงถึงเรื่องใหญ่ ๆ 2 เรื่อง ได้แก่

1. การรักษาข่าวสารที่เก็บในรูปสัญญาณเสียง

2. การแทนสัญญาณเสียงในรูปที่สะดวกต่อการส่ง การเก็บรักษา และการนำข่าวสารออกมาใช้งาน

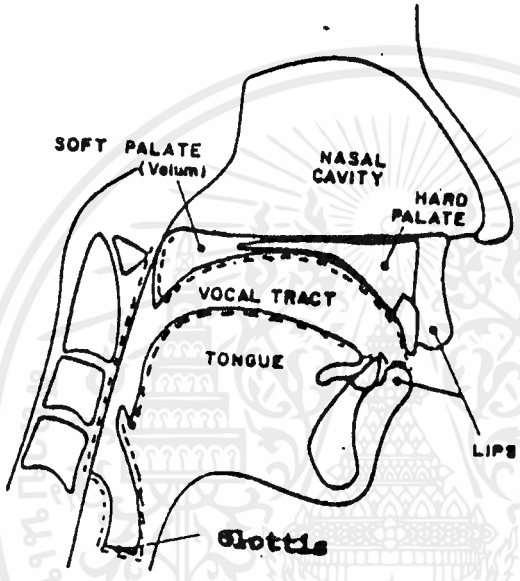
ในการแทนสัญญาณเสียงอยู่ในรูปของสัญญาณดิจิตอลนั้น ต้องมีการสุ่มค่าสัญญาณมาเก็บในหน่วยความจำโดยจะต้องคำนึงถึงความถี่สูงสุดของสัญญาณ และอัตราการสุ่มสัญญาณให้สอดคล้องกับทฤษฎีการสุ่มค่าสัญญาณ นอกจากนี้ในการเก็บค่าที่สุ่มได้นั้น เราต้องคำนึงถึงการเก็บข้อมูลให้มากที่สุด โดยใช้เนื้อที่หน่วยความจำน้อยที่สุด ทั้งนี้จะต้องรักษาเนื้อหาของสัญญาณให้ได้มากที่สุด

ในบทต่อไป จะได้กล่าวถึงแหล่งกำเนิดเสียง วิธีการสุ่มค่าสัญญาณ และวิธีการเก็บรักษาข้อมูล เพื่อนำมาวิเคราะห์ต่อไป

บทที่ 2
แหล่งกำเนิดและลักษณะของเสียง

ขั้นตอนการสร้างเสียง (speech production)

โครงสร้างของระบบสร้างเสียงของมนุษย์ ดังแสดงในรูปที่ 2.1 ส่วนที่ล้อมรอบก็คือ ช่องทางเดินของเสียง(vocal tract)

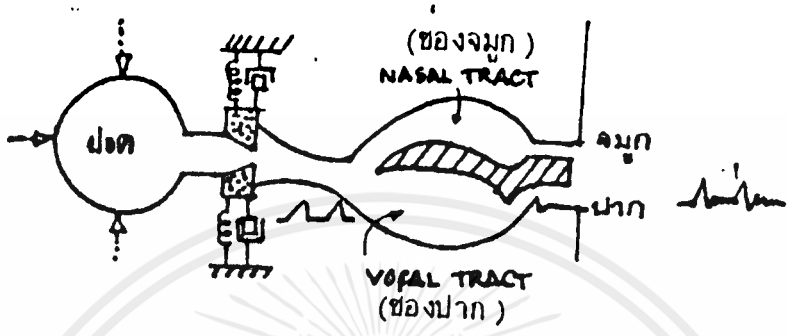


รูป 2.1 แสดงลักษณะของช่องทางเดินเสียง

ช่องทางเดินของเสียงจะเริ่มต้นจากช่องทางเปิดระหว่างเส้นเสียงในกล่องเสียง(Glottis)ไปสิ้นสุดที่ริมฝีปาก ส่วนประกอบของช่องทางเดินเสียงจะมีอยู่ 2 ส่วนใหญ่ๆ คือ ช่องคอ (pharynx)และช่องปาก(olal cavity)โดยเฉลี่ยแล้วช่องทางเดินเสียงของผู้ชายจะมีความยาวประมาน 17 ซม.ขนาดของพื้นที่หน้าตัดจะเปลี่ยนไปตามตำแหน่งของลิ้น ริมฝีปาก กราม และลิ้นไก่ ส่วนช่องทางเดินเสียงของผู้หญิงจะมีขนาดเล็กกว่าของผู้ชายเล็กน้อย

นอกจากนี้จะมีช่องจมูก (nasal tract) ซึ่งจะเริ่มตั้งแต่ลิ้นไก่(velum) ไปจนถึงตอนปลายของจมูก ลิ้นไก่ จะเป็นส่วนที่กั้นระหว่างช่องทางเดินเสียง เสียงที่เปล่งออกมาจะมีลักษณะของเสียงที่ขึ้นจมูกหรือเสียงนาสิก

ในรูปที่ 2.2 เป็นแผนผังแสดงระบบสร้างเสียงที่สมบูรณ์ ซึ่งจะเห็นส่วนของปอดและหลอดลม ซึ่งจะเป็นแหล่งพลังงานสำหรับสร้างเสียง โดยกระแสอากาศจะถูกขับออกมาจากส่วนนี้ เมื่อกระแสอากาศนี้ถูกรบกวนโดยการเคลื่อนไหวของส่วนต่างๆในช่องทางเดินเสียงหรือคลื่นอากาศแผ่กระจายออกมาจากระบบ



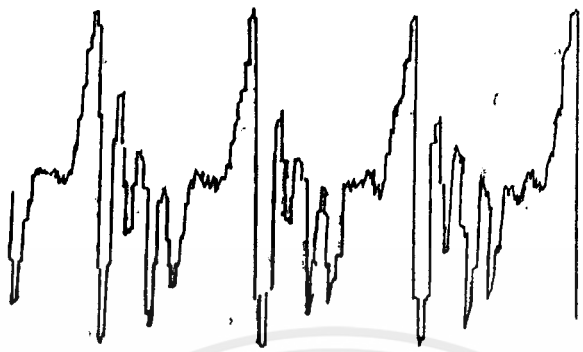
รูป 2.2 แบบจำลองระบบการสร้างเสียงพูด

เสียงพูดสามารถจะแบ่งออกเป็นกลุ่มใหญ่ๆได้ 3 กลุ่ม ตามลักษณะการกระตุ้นของเสียงดังนี้

1. เสียงที่เกิดจากลำคอ (voicad sound) เกิดจากการบังคับให้อากาศวิ่งผ่านกล่องเสียง ซึ่งมีการปรับความตึงของเสียง ทำเกิดการสั่นสะเทือนของอากาศมีลักษณะเป็นคาบไปกระตุ้นช่องทางเดินของเสียง

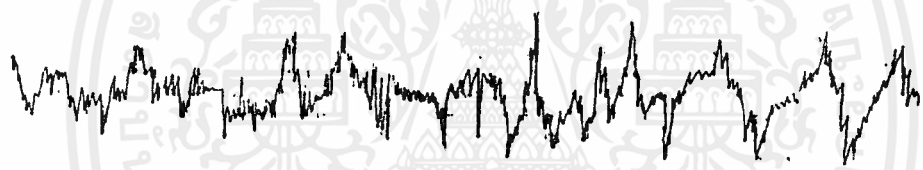
2. เสียงเสียดสี (unvoiced sound or fricative) เกิดจากการบังคับอากาศออกจากช่องปาก แล้วสร้างจุดบีบรัดช่องทางเดินเสียง โดยใช้ ลิ้น ฟัน และริมฝีปาก อากาศจะผ่านจุดรัดนี้ด้วยความเร็วสูง เกิดการปั่นป่วนของกระแสอากาศ เป็นแหล่งกำเนิดของเสียงรบกวนไปกระตุ้นช่องทางเดินเสียง

3. เสียงอัด (plosive sound) เกิดจากการปิดช่องทางเดินของเสียง ทำให้เกิดแรงดันหลังจุดนี้ เมื่อเปิดช่องนี้โดยฉับพลัน ทำให้เกิดแรงกระตุ้นอัดขึ้นมา



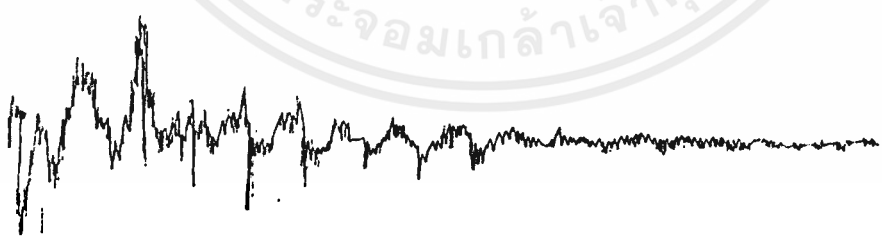
28.5 ms

รูป 2.3 ลักษณะของเสียงที่เกิดจากลำคอ



28.5 ms

รูป 2.4 ลักษณะของเสียงเสียดสี



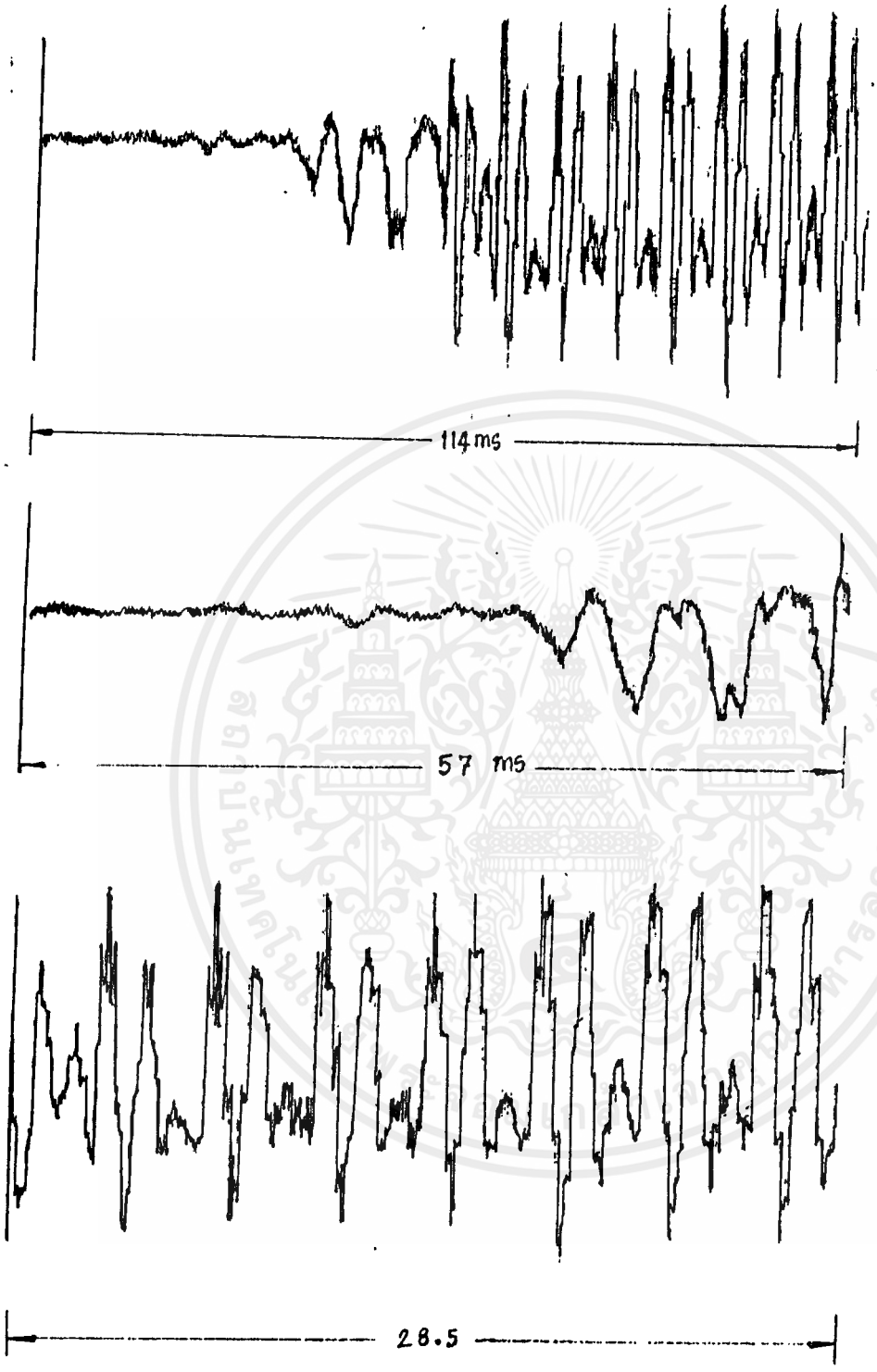
28.5ms

รูป 2.5 ลักษณะของเสียงอัด

เมื่อเสียงถูกกระตุ้นขึ้นมาตามวิธีทั้ง 3 ข้างต้น มันจะเคลื่อนที่ไปตามช่องทางเดินเสียง ซึ่งมีพื้นที่หน้าตัดไม่สม่ำเสมอ รูปร่างของสเป็คตรัม ความถี่ของช่องเสียงจะขึ้นอยู่กับรูปร่างของช่องเสียงว่าจะเข้ากับความถี่ใด ในลักษณะเดียวกับการเกิด เรโซแนนซ์ ในท่อออร์แกน แต่ในทางทฤษฎีเกี่ยวกับเสียงพูด ความถี่เรโซแนนซ์ของช่องทางเดินเสียง เรียกว่า ความถี่ Formant ดังนั้นรูปร่างและขนาดของช่องทางเดินเสียงแต่ละรูปร่างก็จะมีความ Format ชุดหนึ่ง สำหรับความยาวของช่องทางเดินเสียงโดยเฉลี่ย 17 ซม. จะมีความถี่ Format อยู่ในช่วง 250 ถึง 2800 เฮิทซ์ ส่วนช่องทางเดินเสียงของเด็กและสตรี ยิ่งมีขนาดเล็กความถี่ Format จะอยู่ในช่วง 300-3500 เฮิทซ์

คำพูดแต่ละคำ จะประกอบด้วยรูปเสียงต่าง ๆ ตั้งแต่หนึ่งรูปขึ้นไป รูปเสียง คือเสียงที่เกิดจากการจัดช่องทางเดินเสียงให้มีรูปร่างคงที่อันหนึ่ง ดังนั้นเสียงที่เปล่งออกมาสำหรับรูปเสียงหนึ่ง ๆ จะมีความถี่ที่แน่นอน คำบางคำจะประกอบด้วยรูปเสียงรูปเดียว แต่บางคำอาจจะประกอบด้วยรูปเสียงมากกว่าหนึ่งได้ รูปร่างของช่องทางเดินเสียงจะมีการเปลี่ยนแปลงในระหว่างคำนั้น ๆ ดังรูปที่ 2.6

รูปที่ 2.6 แสดงการเปลี่ยนแปลงของความถี่เสียง



รป 2.8 สัญญาณเสียงคำว่า "สิบ"

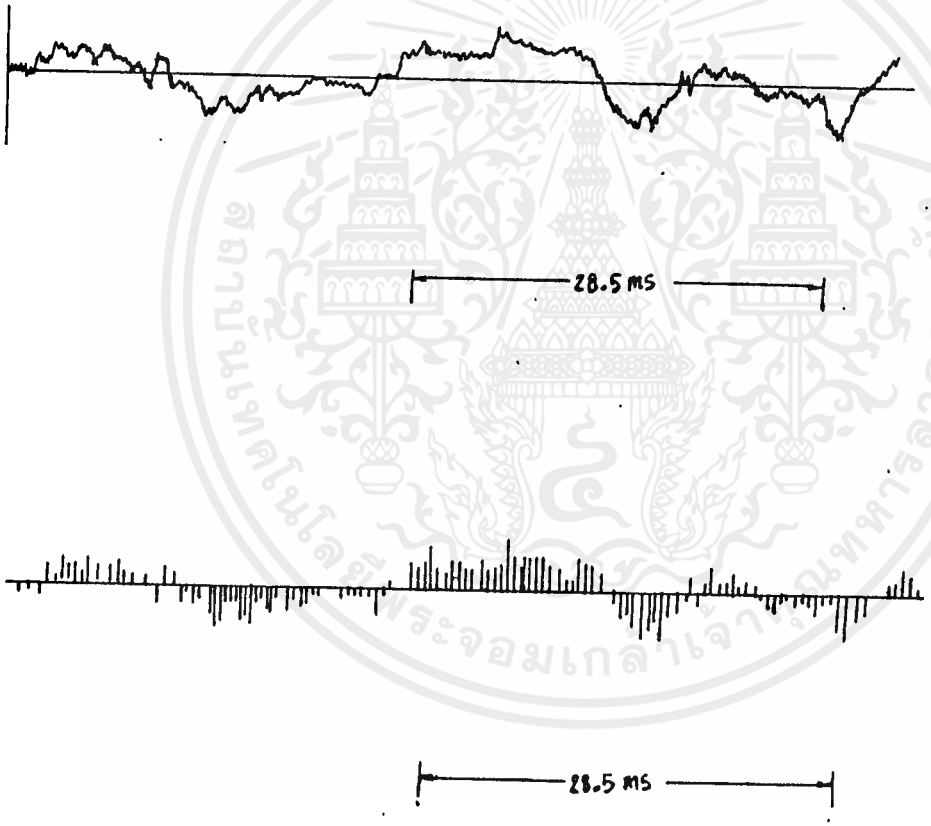
บทที่ 3
การสุ่มสัญญาณเสียง

3.1 การประมวลสัญญาณเสียงด้วยสัญญาณดิจิตอล

ในการพิจารณาเกี่ยวกับการกำเนิดการประมวลผลของสัญญาณเสียงพูด สิ่งแรกที่จะต้องพิจารณา ก็คือ รูปแบบของสัญญาณเสียงพูด ซึ่งจะแสดงได้เป็น 2 แบบ คือ

1. รูปคลื่นของสัญญาณ (Waveform Representation)
2. ค่าพารามิเตอร์ของสัญญาณ (Parametric Representation)

การแสดงรูปแบบของสัญญาณเป็นรูปคลื่น อาจจะแสดงได้เป็นลักษณะของสัญญาณต่อเนื่อง (Analog)และสัญญาณดิจิตอล (Digital) ดังรูปที่ 3.1



รูปที่ 3.1 รูปคลื่นของสัญญาณเสียงพูดแบบต่อเนื่อง (ก) และแบบดิจิตอล (ข

ส่วนการแสดงสัญญาณเสียงที่อยู่ในรูปของค่าพารามิเตอร์ หรือ ค่าที่บ่งบอกคุณสมบัติหรือลักษณะของเสียงได้ ค่าเหล่านี้จะแบ่งออกได้เป็น 2 พวก คือ

1. Excitation Parameter จะเป็นค่าพารามิเตอร์ที่เกี่ยวข้องกับแหล่งกำเนิดหรือลักษณะการกระตุ้นของเสียงพูด

2. Vocal tract response Parameter เป็นค่าพารามิเตอร์ที่เกี่ยวข้องกับลักษณะเสียงพูดแต่ละชนิด ซึ่งขึ้นอยู่กับรูปร่างของช่องทางเดินเสียง

ในการหาค่าพารามิเตอร์แทนสัญญาณนั้น เราจะต้องแทนสัญญาณให้อยู่ในรูปของสัญญาณดิจิตอลเสียก่อน นั่นคือ สัญญาณเสียงจะถูกสุ่มค่ามาจากนั้นจึงจะนำค่าที่สุ่มได้นี้ไปผ่านกระบวนการต่าง ๆ เพื่อหาค่าพารามิเตอร์ของสัญญาณเสียงนั้น

การสุ่มค่า (Sampling) ก็คือ การวัดขนาดของสัญญาณต่อเนื่อง $X_a(t)$ มาเป็นช่วง ๆ เราก็จะได้ค่าเป็นชุดของตัวเลขที่แสดงของสัญญาณที่เวลาต่าง ๆ กัน $X(n)$ และถ้าเราให้ช่วงห่างของเวลา หรือคาบของการสุ่มค่านี้เป็น T แล้ว เราจะเขียนได้ว่า

$$X(n) = X_a(nt) \qquad -\alpha < n < \alpha \quad \text{------} \quad (3.1)$$

โดยที่ n เป็นตัวเลขจำนวนเต็ม บอกลำดับที่ของค่าที่สุ่มได้

ในรูปที่ 2.1 แสดงรูปแบบของสัญญาณเสียงพูด และค่าที่สุ่มมาโดยมีคาบเวลาการสุ่ม $T = {}^{1}/_{9000}$ วินาที หรือพูดอีกอย่างหนึ่งก็คือ สัญญาณนี้ถูกสุ่มค่าในอัตราความถี่ 9000 ครั้งใน 1วินาที

3.2 ทฤษฎีการสุ่มสัญญาณ (Sampling Theory)

ถ้าสัญญาณ $X_a(t)$ เป็นสัญญาณต่อเนื่อง ซึ่งเมื่อแปลงโดยใช้ Fourier เป็น $X_a(j\Omega)$ ที่มีความถี่สูงสุดนั้นเป็น f_m นั่นคือ

$$X_a(j\Omega) = 0 \qquad \text{เมื่อ } \Omega > 2\pi f_m \qquad \text{------ (3.2)}$$

เราสามารถจะสร้างสัญญาณ $x_a(t)$ ขึ้นมาใหม่ได้จากค่าที่ได้จากการสุ่มสัญญาณ คือ

$$X_a(nt) \qquad \text{เมื่อ } -\alpha < n < \alpha \qquad \text{------ (3.3)}$$

ถ้าอัตราของการสุ่มสัญญาณมากกว่า 2 เท่าของความถี่สูงสุด

$$1/T > 2f_m \qquad \text{------ (3.4)}$$

ทฤษฎีนี้ได้มาจากความจริงที่ว่า fourier transform ของ $X_a(t)$ ซึ่งนิยามว่า

$$X_a(j\Omega) = \int_{-\alpha}^{\alpha} X_a(t)\, e^{-j\Omega t}\, dt \qquad \text{------ (3.5)}$$

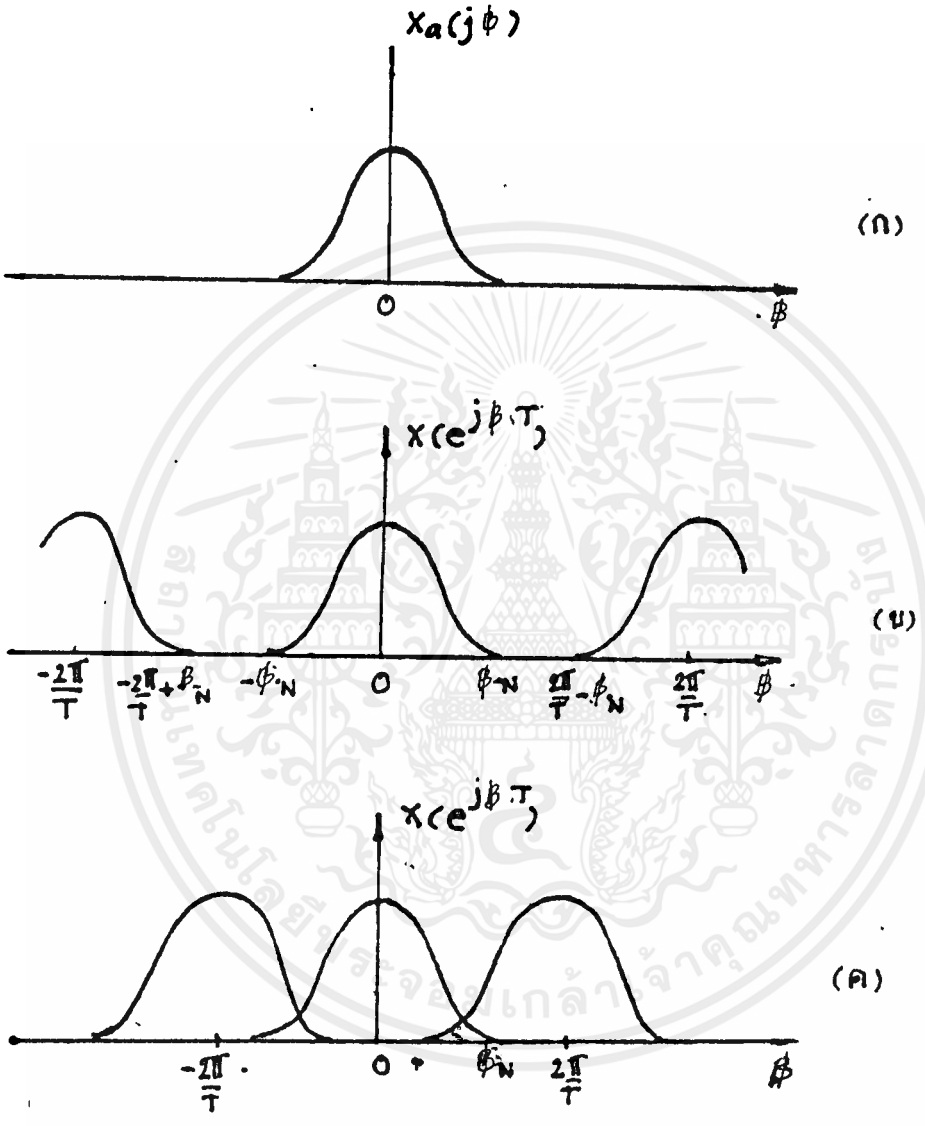
เป็นสัญญาณซึ่งมีความถี่สูงสุด คือ f_m ดังรูป ที่ 3.2 และ fourier transform ของสัญญาณไม่ต่อเนื่อง X(n) ได้แก่

$$X(e^{j\omega}) = \sum_{n=-\alpha}^{\alpha} X_a(t)\, e^{-j\omega n} \qquad \text{------ (3.6)}$$

ดังนั้นถ้าเราหาค่าของ $X(e^{j\omega})$ ที่ความถี่ $\omega=\Omega t$ เราจะได้ความสัมพันธ์ระหว่าง $X(e^{j\Omega t})$ กับ $X_a(j\Omega)$ ดังนี้

$$X(e^{j\Omega t}) = 1/T \sum_{k=-\alpha}^{\alpha} X_a\,(j\Omega + j(2\pi/T)k) \qquad \text{------ (3.7)}$$

จากสมการนี้จะเห็นว่า $X(e^{j\phi T})$ เป็นผลบวกของค่าที่ซ้ำกันของ $X_a(j\phi)$ ซึ่งมีระยะห่างกันเป็นจำนวนเท่าของ $2\pi/T$



รูป รูปแสดงแถบความถี่ของสัญญาณเมื่อผ่านการแปลงฟูเรียร์

(ก) แถบความถี่ของสัญญาณดั้งเดิม

(ข) แถบความถี่ของสัญญาณที่สุ่มด้วยความถี่มากกว่า $2\ f_n$

(ค) แถบความถี่ของสัญญาณที่สุ่มด้วยความถี่น้อยกว่า $2\ f_n$

ในรูปที่ 3.2 แสดงแถบความถี่ของสัญญาณดั้งเดิม และสัญญาณที่ถูกสุ่มมาโดยใช้ความถี่ในการสุ่มสัญญาณต่างกันดังนี้

1.รูป (ข) เป็นกรณีที่ $(1/T)>2\ f_m$ จะเห็นว่าแถบความถี่ชิ่งเป็นเงาเสมือน (imag) ของแถบความถี่แรก (base band) จะไม่ซ้อนกันแถบความถี่แรก $|\omega| > 2\pi f_m$

2.รูป (ค) เป็นกรณีที่ $(1/T)<2\ f_m$ ในกรณีนี้แถบความถี่ชิ่งมีจุดกึ่งกลางอยู่ที่ $2\pi/T$ จะซ้อนอยู่กับแถบความถี่แรก ชิ่งทำให้ความถี่สูงมีค่าเสมือนความถี่ต่ำ

เราจะหลีกเลี่ยงกรณีที่แถบความถี่ซ้อนกัน โดยให้แถบความถี่ที่ได้จากการแปลงฟูเรียมีความถี่จำกัด นั่นคือ มีความถี่สูงสุดเป็น f_m และความถี่ของการสุ่มต่ำจะมีค่าอย่างน้อย 2 เท่าของความถี่สูงสุดนี้ $(1/T > 2\ f_m)$

ภายใต้เงื่อนไข $(1/T > 2\ f_m)$ นี้ จะเห็นว่าการแปลงฟูเรียของค่าที่ได้จากการสุ่มสัญญาณ จะเป็นสัดส่วนกับการแปลงฟูเรียของสัญญาณต่อเนื่อง ในแถบความแรก

$$X(e^{j\omega T}) = (1/T)\ X_a(j\omega) \quad |\omega| < \pi/T \qquad \text{------ (3.8)}$$

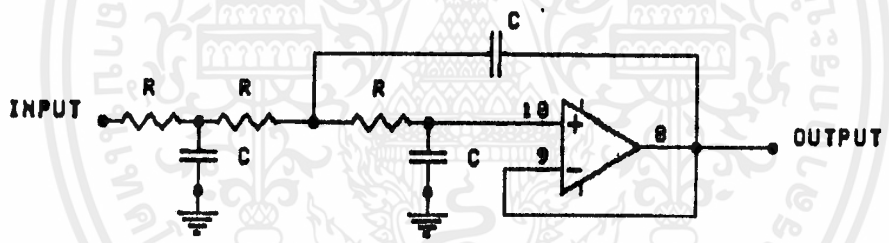
จากผลอันนี้ เราสามารถจะหาค่าของสัญญาณดั้งเดิมได้ จากค่าของสัญญาณสุ่มโดยใช้สูตรการ interpolation ดังนี้

$$X_a(t) = \sum_{n=-\alpha}^{\alpha} X_a(nt)\ [(\sin \pi\ (t-nt)/T)/\ \pi(t-nt)/T)] \qquad \text{------ (3.9)}$$

นั่นคือ ถ้าเรามีค่าสุ่มตัวอย่างของสัญญาณต่อเนื่อง ชิ่งมีความถี่ของการสุ่มอย่างน้อย 2 เท่าของความถี่สูงสุดของสัญญาณ เราสามารถจะสร้างสัญญาณต่อเนื่องนั้นคืนมาใหม่โดยใช้สมการ (3.9)

3.3 วงจรกรองความถี่ (filter circuit)

จากบทที่แล้วเราทราบว่าเสียงพูดของคนโดยเฉลี่ยจะมีความถี่อยู่ในช่วง 250 - 3500 เฮิทซ์ ซึ่งโดยทั่วไปแล้ว ความถี่สูงสุดของเสียงพูดจะไม่เกิน 4000 เฮิทซ์ แต่ก็ยังมีโอกาสที่บางส่วนของสัญญาณเสียงพูดจะมีความถี่สูงกว่านี้ ดังนั้นก่อนที่จะมีการสุ่มค่า สัญญาณนี้จะถูกนำไปผ่านวงจรกรองความถี่แบบผ่านต่ำ (low pass filter) ที่มีจุดตัดความถี่เป็น 4000 เฮิทซ์ เพื่อตัดส่วนของการสุ่มสัญญาณที่มีความถี่สูงกว่า 4000 เฮิทซ์ออก เป็นการจำกัดช่วงกว้างของแถบความถี่ ทำให้ความถี่ของการสุ่มสัญญาณไม่สูงมาก แต่โดยทั่วไปสัญญาณส่วนความถี่สูงจะมีขนาดต่ำอยู่แล้ว เมื่อถูกลดทอนขนาดลงไปอีกโดยวงจรกรองความถี่ถึงแม้จะไม่หมด แต่ส่วนที่ผ่านไปได้ก็จะมีขนาดเล็กมากจนไม่มีผลต่อการสุ่มสัญญาณ จากทฤษฎีของการสุ่มสัญญาณกำหนดให้ความถี่ของสุ่มสัญญาณมากกว่า 2 เท่าของความถี่สูงสุดของสัญญาณ ดังนั้นความถี่ของการสุ่มจะต้องมากกว่า 8000 เฮิทซ์



บทที่ 4
เดลต้ามอดูเลชั่น

เดลต้ามอดูเลชั่น (DM) เป็นวิธีการหนึ่งของ ดิจิตอลมอดูเลชั่น (Digital Modulation) ถูกนำไปใช้งานในด้านการสื่อสารดาวเทียว การส่งโทรทัศน์และระบบโทรศัพท์ เดลต้ามอดูเลชั่น เป็นวิธีการที่ลุ่งยากน้อยกว่า และค่าใช้จ่ายถูกกว่าระบบ พีซีเอ็ม (PCM) นอกจากนี้มันยังให้การผิดพลาดในการส่งข้อมูลเหมือนกับระบบพีซีเอ็ม แต่มันมีข้อเสียคือ มันความไวต่อการเปลี่ยนแปลงความชันเกินคาด (slope overload) ของวงจรอินทิเกรเตอร์ (integrator) และใช้กับระบบเวลาร่วม (time sharing) ไม่ได้

เทคนิคของเดลต้ามอดูเลชั่นจะไม่ใช้การสุ่มสัญญาณหนึ่งจุด แล้วแปลงเป็นข้อมูลดิจิตอลหนึ่งเวิร์ด ที่มีความละเอียดเป็นจำนวนบิทที่ต้องการ แต่จะใช้วิธีเปรียบเทียบความสูงหรือการเปลี่ยนของสัญญาณเสียงแทน

ข้อมูลที่ได้ก็คือ ทิศทางการเปลี่ยนแปลง ซึ่งก็มีเพียง ขึ้น หรือ ลงเท่านั้น ดังนั้นความกว้างของข้อมูลดิจิตอลจึงใช้เพียงบิทเดียวก็พอ ข้อดีของวิธีการเดลต้ามอดูเลชั่นก็คือ ใช้หน่วยความจำน้อยกว่าแบบอื่นๆ

หลักการทำงานของ เดลต้า มอดูเลชั่น

ในรูปแบบปกติ สัญญาณที่ได้จาก เดลต้ามอดูเลชั่น จะให้สัญญาณเป็นรูปขั้นบันได (stair case) ซึ่งประมาณได้จากสัญญาณอินพุทหรือสัญญาณ เบสแบนด์ (base band) ดังแสดงในรูป ความแตกต่างระหว่างอินพุทและสัญญาณที่ได้จากการประมาณ จะถูกเปลี่ยนเป็นสัญญาณสองระดับ คือ β เท่านั้น

หลักการทำงานการประมาณค่าสัญญาณนั้น ถ้าสัญญาณอินพุทมีค่าสูงกว่า สัญญาณที่ได้จากการประมาณค่าที่ตำแหน่งของการสุ่ม (sampling) ครั้งก่อนก็จะทำการเพิ่มสัญญาณ แอปโปรชิเมท (approximate) ขึ้นอีก ในทางกลับกันถ้าสัญญาณแอปโปชิเมทมีค่ามากกว่าสัญญาณอินพุทมันก็จะถูกลดลง เช่นกัน ถ้าสัญญาณเปลี่ยนแปลงไม่เร็วเกินไปเราจะพบว่า สไตร์ เคส แอปโปชิเมท (stair case approximate) จะยังคงสูงขึ้นหรือต่ำกว่าสัญญาณอินพุทไม่เกิน β

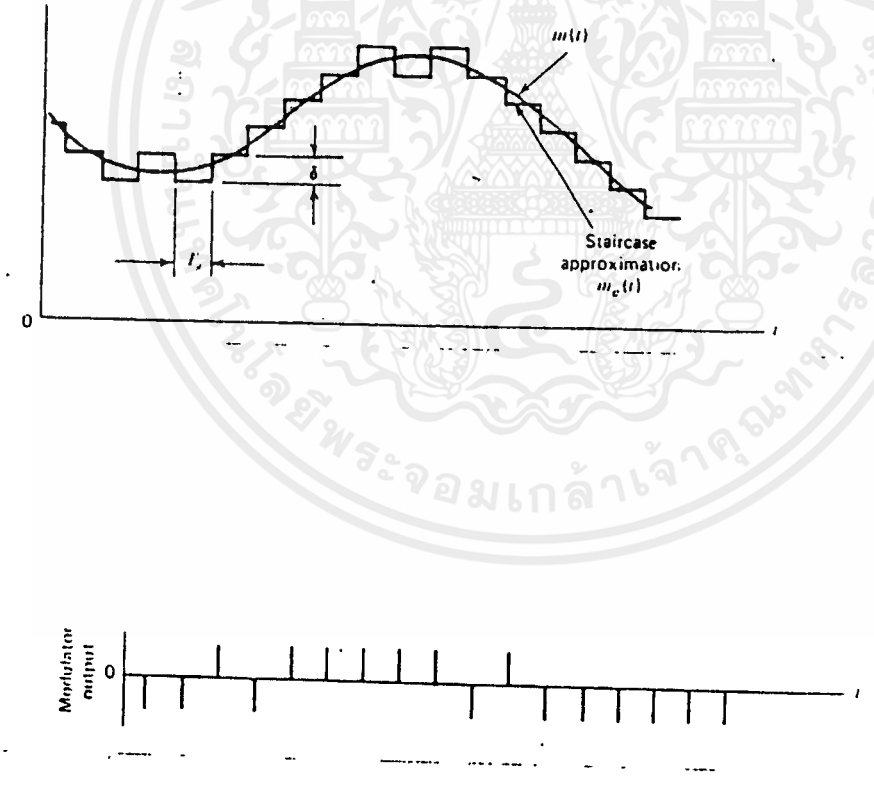
ในสัญญาณอินพุทเป็น m(t) และ สไตร์ เคส แอปโปซิเมท เป็น ma(t) แล้วหลักการเบื้องต้นของ เดลต้ามอดูเลชั่น สามารถเขียนเป็นสมการได้

$$b_n = \text{sgn}[m(n\,Ts) - ma(n\,Ts - Ts)] \quad \text{------ (5.1)}$$

$$ma(n\,Ts) = m(n\,Ts - Ts) + \beta b_n \quad \text{------ (5.2)}$$

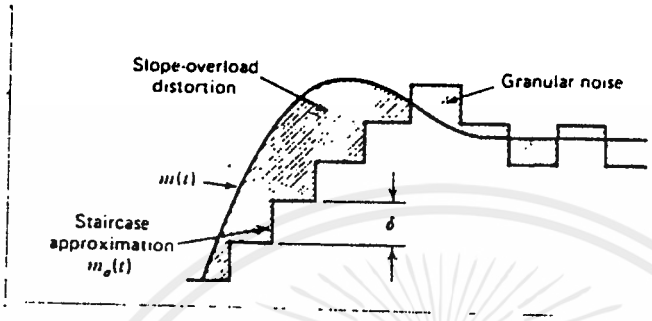
Ts เป็นเวลาของการ เซมปลิ่ง และ b_n เป็นเครื่องหมายที่ได้

สำหรับแต่ละแซมเปิล การส่งจะส่งเป็นข้อมูล 1 bit คือ b_n และอัตราของการส่งผ่านข้อมูลจะเท่ากับอัตราการแซมปลิ่ง 1/Ts ดังแสดงในรูปที่ 4.1



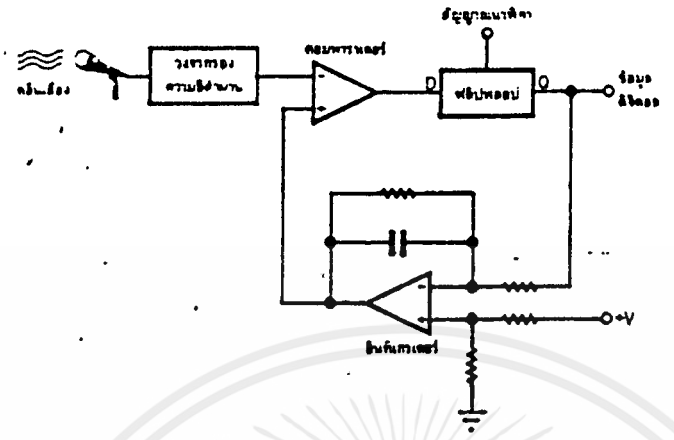
รูปที่ 4.1

ระบบเดลต้ามอด จะเกิดความผิดพลาดหรือที่คลาดเคลื่อนที่เรียกว่า ความผิดพลาดควอนติซิ่ง (Quantizing error) ซึ่งมี 2 อย่าง คือ ความผิดพลาดที่เกิดจาก สโลป โอเวอร์โหลด ดีสทอร์ชั่น (slope overload distortion) และความผิดพลาดที่เรียกว่า เจนนูลาร์ นอยซ์ (granular noise) ดังแสดงในรูปที่ 4.2



รูปที่ 4.2 แสดงการเกิด quantizing error ใน เดลต้ามอด

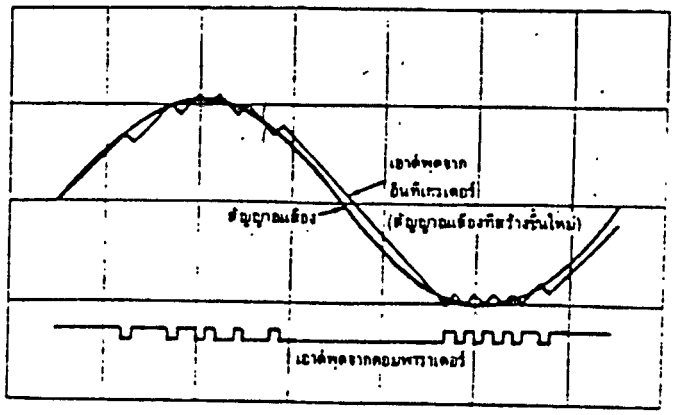
สโลป โอเวอร์โหลด ดีสทอร์ชั่น เกิดขึ้นเมื่อขนาดของ สเตป δ เล็กเกินไปที่จะทำให้สไตร์ เคส แอปโปซิเมท ma(t)ตามสัญญาณอินพุท m(t)ได้ทัน และเจนนูล่าร์ นอยซ์ เกิดขึ้นเมื่อขนาดของสเตป ใหญ่เกินไปสัญญาณอินพุทซึ่งเป็นผลมาจากสัญญาณ ma(t) สเตปขึ้นและลงระหว่างส่วนที่ค่อนข้างจะราบเรียบของอินพุท เจนนูล่าร์ นอยซ์ ในระบบเดลต้ามอดนี้เทียบเท่ากับ ควอนติซิ่ง นอยซ์ ในระบบ พีซีเอ็ม (PCM) ดังนั้นสำหรับสัญญาณที่มีค่าสโลปอันหนึ่ง ถ้าขนาดของสเตปเล็กก็จะทำให้เกิด สโลป โอเวอร์โหลด ดีสทอร์ชั่น ในขณะที่ขนาดของสเตปใหญ่จะทำให้เกิด เจนนูล่าร์ นอยซ์



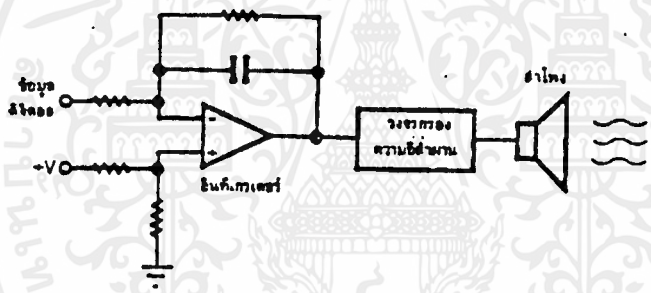
รูปที่ 4.3 วงจรเบื้องต้นของเดลต้ามอดูเลชั่นในส่วนการแปลงจากสัญญาณเสียงเป็นดิจิตอล

รูปที่ 4.3เป็นวงจรเบื้องต้นของเดลต้ามอดูเลชั่น คอมพาราเตอร์จะทำหน้าที่เปรียบเทียบสัญญาณอินพุทปัจจุบันกับอินพุทก่อนหน้า ซึ่งได้จากการป้อนกลับมายังอินทิเกรเตอร์ เอาท์พุทจากการเปรียบเทียบถูกป้อนผ่านฟลิบฟลอบ ที่ควบคุมด้วยสัญญาณนาฬิกาเพื่อให้ได้เป็นข้อมูลดิจิตอล ซึ่งก็คือการกำหนดอัตราการสุ่มสัญญาณนั่นเอง สัญญาณที่ได้รับจากตัวเปรียบเทียบและจากอินทิเกรเตอร์เมื่อเปรียบเทียบกับสัญญาณอินพุทแสดงในรูปที่ 5.4

ลักษณะเช่นนี้จะพบว่า ยิ่งความถี่ของสัญญาณนาฬิกามีค่าสูงก็ยิ่งสามารถ บันทึกการเปลี่ยนแปลงที่แคบได้มากขึ้นทำให้ได้คุณภาพเสียงที่ดีขึ้น แต่ก็สิ้นเปลืองหน่วยความจำมากขึ้นตามไปด้วย ความถี่เท่าใดจึงจะเพียงพอนั้น คงต้องทดลองโดยการนำเอาท์พุทสุดท้ายที่เป็นข้อมูลดิจิตอลผ่านวงจรแปลงกลับดังในรูปที่ 4.5 แล้วฟังเสียงก็ได้ หากฟังแล้วปรากฏว่าเสียงชัดเจนดีก็เลือกเอาความถี่นั้น



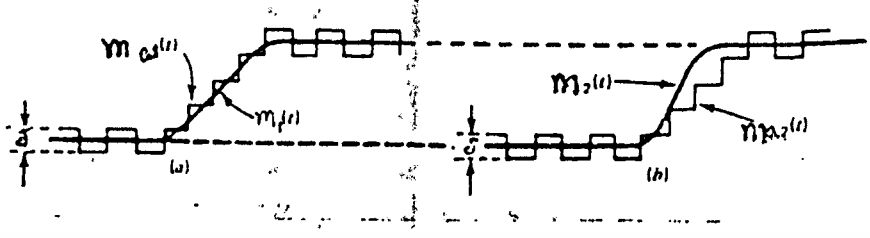
รูปที่ 4.4 เปรียบเทียบสัญญาณอินพุทที่ได้สัญญาณอนาลอกจากอินทีเกรเตอร์



รูปที่ 4.5 วงจรที่ใช้แปลงกลับข้อมูลดิจิตอลเป็นสัญญาณเสียง

slope overloading

ปัญหาใหญ่ในระบบเดลต้ามอด คือการเกิดสโลป โอเวอร์โหลด เมื่อสัญญาณอินพุท $m(t)$ เปลี่ยนแปลง สัญญาณ แอปโปซิเมท $ma(t)$ จะสเตป ตามสัญญาณอินพุทไปตลอด ตราบเท่าที่สัญญาณอินพุท แซมเปิล ยังมากหรือน้อยกว่าสัญญาณ $a(t)$ โดยมีความแตกต่างไม่มากนัก แต่เมื่อความแตกต่างเกินกว่า $a\ ma(t)$ ซึ่ง สเตป ตามสัญญาณ $m(t)$ ไม่ทันนั้น ก็จะเกิดเป็นความผิดพลาดที่เรียกว่า โอเวอร์โหลด นี้เกิดขึ้นเนื่องจาก สโลป ของสัญญาณ $m(t)$ เราจึงเรียกว่า สโลป โอเวอร์โหลด ดังแสดงในรูปที่ 4.6



รูปที่ 4.6 การเกิด slope overload

เพื่อที่จะหาเงื่อนไขสำหรับป้องกัน สโลป โอเวอร์โหลด ในระบบ เดลต้ามอด เราสมมุติว่าอินพุท $m(t) = A \cos (2 \pi f_m t)$ ดังนั้น สโลป สูงสุดของสัญญาณคือ

$$\left[dm(t)/dt \right]_{max} = A\, 2\pi\, f_m$$

การเปลี่ยนแปลงมากที่สุดในช่วงระหว่าง แซมเปิล ของสัญญาณอินพุท คือ $A\, 2\pi\, f_m\, Ts$ เพื่อหลีกเลี่ยงการเกิด สโลป โอเวอร์โหลด การเปลี่ยนแปลงของ แซมเปิล นี้ต้องน้อยกว่า นั่นคือ

$$2\pi\, f_m\, Ts\, A < \beta$$

หรือ แอมปริจูด ที่ซึ่ง สโลป โอเวอร์โหลด จะเกิด คือ

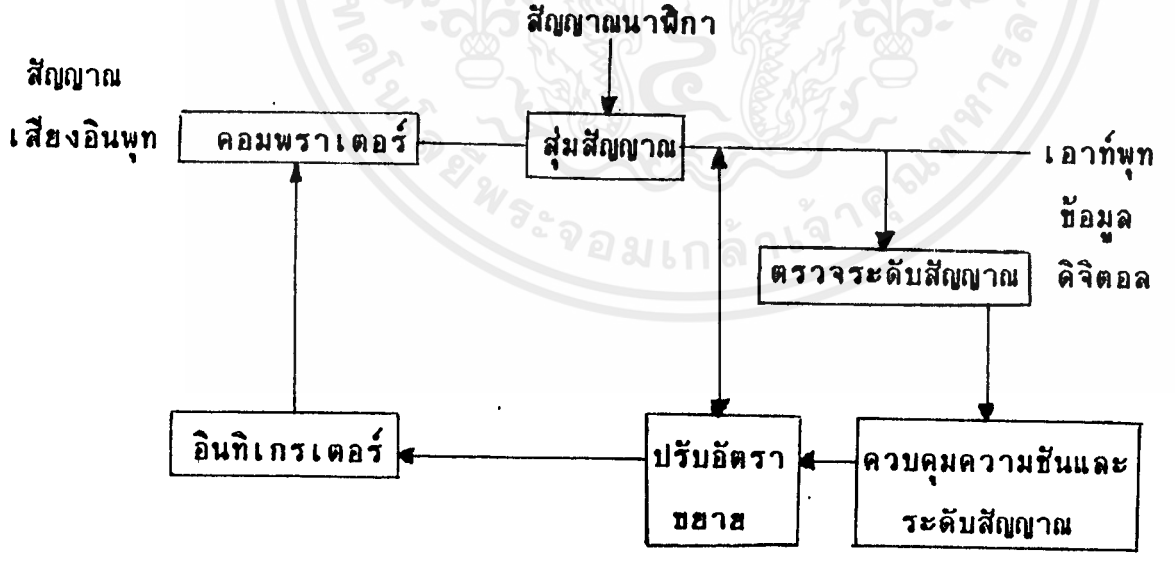
$$A = \beta / 2\pi * (f_s / f_m)$$

ซึ่ง $fs = 1/Ts$ เป็นอัตราการ แซมปลิ่ง ของระบบ เดลต้ามอด ได้มีการตรวจสอบโดยการทดลองพบว่า เดลต้า มอดูเลชั่น จะส่งสัญญาณโดยปราศจากการเกิด สโลป โอเวอร์โหลด นั้น แอมปริจูด ของสัญญาณจะต้องไม่เกินกว่า แอมปริจูด มากที่สุดของสัญญาณในสมการ ซึ่งใช้กับ $fm = 800$ Hz

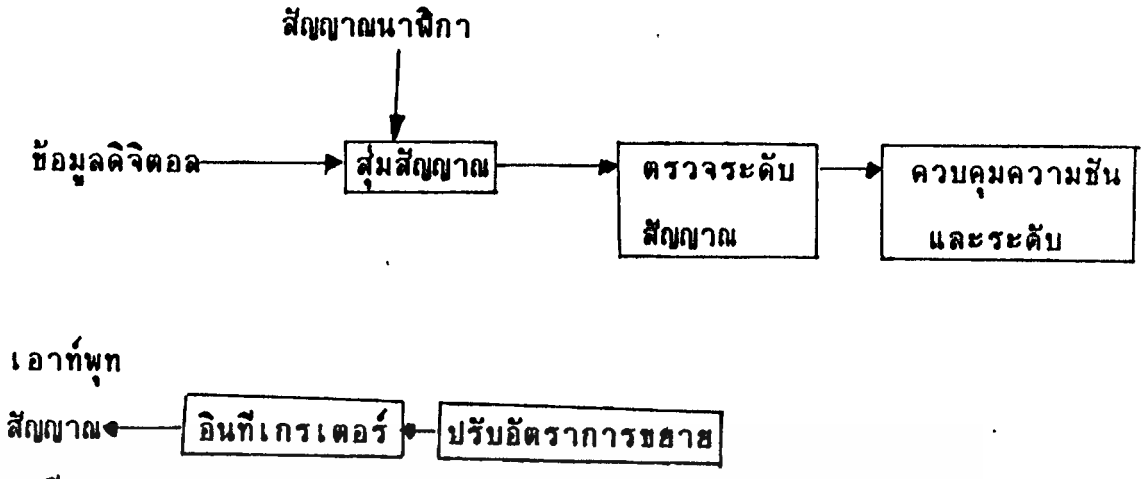
ปัญหาของการเกิด สโลป โอเวอร์โหลด ในระบบ เดลต้า มอดูเลชั่น สามารถแก้ไขโดยการ ฟิลเตอร์ หรือโดยสัญญาณเพื่อจำกัดอัตราสูงสุดของการเปลี่ยนแปลงการเพิ่มขนาดของ สเตป หรือเพิ่มอัตราการ แซมปลิ่ง การฟิลเตอร์ สัญญาณและการเพิ่มขนาดของ สเตป ทำให้เกิด เจนนูลาร์ นอยซ์ มาก และการเพิ่มอัตราการ แซมปลิ่ง จะทำให้ต้องใช้ แบนด์วิดท์ มากขึ้น ทางที่ดีที่สุดที่จะหลีกเลี่ยง สโลป โอเวอร์โหลด โดยการตรวจสอบเงื่อนไขการเกิดโอเวอร์โหลด และทำให้ สเตป มีขนาดมากขึ้นเพื่อตรวจสอบได้ว่าเกิด โอเวอร์โหลด

CVSD

ข้อจำกัดของวิธีการเดลต้ามอดูเลชั่นก็คือ แถบกว้างของความถี่ใช้งาน ซึ่งถูกจำกัดโดย-ความถี่สัญญาณนาฬิกา และจะสูงกว่าความถี่สูงสุดของสัญญาณอินพุทมากกว่า 2 เท่าขึ้นไป อีกอันหนึ่งคือความเร็วของการเปลี่ยนแปลงความสูงของสัญญาณหรือไดนามิกเรนจ์ ระบบเดลต้า-มอดูเลชั่นมีค่าไดนามิกเรนจ์ที่แคบ จำเป็นต้องมีส่วนขยายเพิ่มเติมทำหน้าที่ขยายไดนามิกเรนจ์ให้กว้าง โดยการควบคุมอัตราการขยายของอินทิเกรเตอร์ เพื่อให้สนองต่อสัญญาณที่มีความชันมากๆได้ทัน ระบบใหม่นี้เรียกว่า ระบบเดลต้ามอดูเลชั่นแบบเปลี่ยนแปลงความชันต่อเนื่องหรือ CVSD (continuous variable slope delta modulation)



รูปที่ 4.แผนผังการทำงานของระบบ CVSD ในส่วนการเปลี่ยนแปลงจากสัญญาเป็นข้อมูลดิจิตอล



รูปที่ 4. แผนผังการทำงานของระบบ CVSD ในส่วนการเปลี่ยนแปลงจากเป็นข้อมูลดิจิตอลเป็นสัญญาณเสียง

ระบบ CVSD ทั้งส่วนการเปลี่ยนแปลงจากอนาลอกดิจิตอล และส่วนแปลงกลับจากดิจิตอลเป็นอนาลอก แสดงในรูปที่ 4 และ 5 ตามลำดับ

วิธีการ CVSD ก็คือมีการตรวจระดับสัญญาณ โดยอาจใช้วิธีการจัดให้มีรีจิสเตอร์สำหรับเก็บข้อมูลล่าสุดจำนวน 3 ถึง 4 บิทแล้วตรวจดูว่าเป็น "0" หรือ "1" หมดหรือไม่ ถ้าใช่แสดงว่าขณะนี้อัตราขยายของอินทีเกรเตอร์ต่ำเกินไปและตอบสนองต่อความชันเสียงไม่ทัน ก็จะทำการเพิ่มอัตราการขยายให้สูงขึ้นเฉพาะในช่วงนั้น

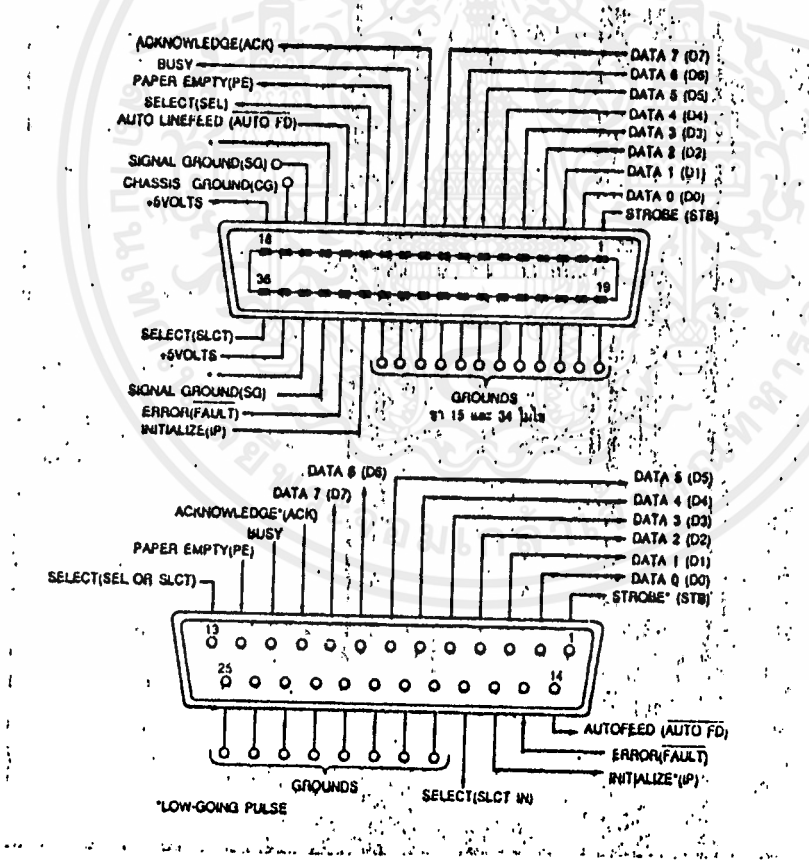
ในส่วนการแปลงกลับก็ต้องมีการทำงานในลักษณะเดียวกัน คือมีรีจิสเตอร์สำหรับตรวจดูข้อมูลว่าเป็น
"0" หรือ "1" หมดหรือไม่ แล้วจัดการควบคุมอัตราขยายของอินทีเกรเตอร์ให้สอดคล้องกัน

ในปัจจุบันนี้มีการคิดค้นและออกแบบวงจรอิเลคทรอนิคส์ขึ้นมาทำหน้าที่บันทึก และถ่ายเทข้อมูลที่เป็นเสียงแทนเทป ซึ่งให้ความสดวกและคล่องตัวกว่า ถึงแม้ว่าวงจรอิเลคทรอนิคส์จะบันทึกเสียงไว้ไม่ได้มากกว่าเทป แต่เมื่อเทียบประโยชน์ในงานบางอย่างแล้วนับว่าคุ้มค่ามาก และในโครงงานนี้ได้ใช้ไอซีสำเร็จรูป เบอร์ UM5100 เป็นตัวจัดการในการบันทึกและถ่ายเทข้อมูลที่เป็นเสียง ซึ่งรายละเอียดการทำงานของไอซีเบอร์ UM 5100 จะกล่าวในบทที่ 7

บทที่ 5
พอร์ทขนาน

อุปกรณ์ขนานส่วนใหญ่จะมีจุดต่อเป็นคอนเน็กเตอร์ตัวผู้หรือเซ็นทรอนิกส์ (centronice) มีทั้งหมด 36 ขาดังแสดงในรูปที่ 7.1(ก) แต่คอมพิวเตอร์ที่เป็น IBM คอมแพตติเบิลนั้นจะเป็นคอนเน็กเตอร์แบบ DB-25 ดังแสดงในรูปที่ 7.1(ข) ซึ่งจะมีลักษณะเป็นคอนเน็กเตอร์ตัวเมีย จำนวน 25 ขา

จากรูปที่ 7.1 แสดงถึงสัญญาณที่จะเดินทางเข้าสู่คอนเน็กเตอร์ ซึ่งสัญญาณเข้าจะแสดงด้วยลูกศรที่ชี้เข้าหาคอนเน็กเตอร์ และสัญญาณออกจะแสดงด้วยลูกศรที่ชี้ออกจากคอนเน็กเตอร์ ส่วนกราวด์นั้นจะแสดงด้วยวงกลม และอักษรย่อที่เป็นมาตราฐานของสัญญาณแสดงไว้ในวงเล็บหลังชื่อสัญญาณต่าง ๆ



รูปที่ 5.1 แสดงรูปร่างและลักษณะของคอนเน็กเตอร์

(ก) คอนเน็กเตอร์แบบเซ็นทรอนิกส์กับสัญญาณ

(ข) คอนเน็กเตอร์แบบ DB-25 กับสัญญาณ

เพื่อให้สะดวกและง่ายขึ้นสามารถสรุปที่มาของตำแหน่งขา และสัญญาณที่อยู่บนคอนเน็กเตอร์แบบเซ็นทรอนิกส์ไว้ดังตาราง ทุก ๆ สัญญาณจะใช้ระดับสัญญาณแบบ TTL คือสัญญาณที่อยู่ระหว่าง 2.4-5 โวลท์จะเป็น high หรือลอจิก "1" และสัญญาณที่อยู่ระหว่าง 0-0.8 โวลท์จะเป็น low หรือลอจิก "0" ส่วนสัญญาณที่อยู่ระหว่าง 0.8-2.4 โวลท์จะถูกพิจารณาเป็นโมฆะหรือใช้การไม่ได้ เริ่มต้นที่สัญญาณสโตบ (data strobe) เป็นสัญญาณแรก คอมพิวเตอร์จะสร้างพัลส์ low ที่สายเส้นนี้เพื่อบอกสถานะแก่อุปกรณ์ร่วมว่าข้อมูลพร้อมแล้ว และทำการเตียมพร้อมที่จะรับข้อมูลเข้า (data 0 - data 7)

สายข้อมูล (ขา2-9)มีขนาด 8 บิตเป็นตัวที่จะรับข้อมูลได้ 1ไบต์ที่ส่งมาอาจเป็นตัวอักษรก็ได้ ซึ่งอุปกรณ์บางตัวจะรับข้อมูลได้ทีละ 7 บิตเท่านั้นในกรณีนี้ขาดาต้า 7 (D7) ไม่ได้ถูกนำมาใช้ หรือในบางทีอาจใช้ในการตรวจสอบพาริตี้ก็ได้ ดังนั้นควรที่จะดีสเอเบิล (disable) การตรวจสอบพาริตี้ไว้เลย

วิธีนี้สามารถสั่งให้คอมพิวเตอร์รอจนกว่าจะพร้อมที่จะรับข้อมูลในครั้งต่อไปได้ อุปกรณ์จะสามารถหยุดรับข้อมูลโดยการส่งสัญญาณ busy เป็น high เมื่ออุปกรณ์ร่วมตรวจสอบพัลส์สโตรบมันจะคงค่า busy เป็น hihg จนกว่ามันจะประมวลผลข้อมูลเดิมเสร็จ และเมื่อพร้อมที่จะรับข้อมูลก็จะทำให้สัญญาณ busy เป็น low ซึ่งคอมพิวเตอร์สามารถส่งข้อมูลมายังอุปกรณ์ได้ สัญญาณ busy บางครั้งอาจใช้สำหรับหยุดคอมพิวเตอร์ ในกรณีอื่นก็ได้เช่น ในกรณี out of paper หรือ off line เป็นต้น ข้อควรระวังอย่างหนึ่งก็คือสัญญาณ busy นี้บางครั้งสามารถเปลี่ยนขั้วได้ โดยกำหนดดีพสวิทช์ที่อุปกรณ์ร่วม

สัญญาณ acknowledge และ busy จะทำงานในลักษณะเดียวกันแต่คนละหน้าที่ โดยอุปกรณ์ร่วมบางตัวอาจจะสนับสนุนสัญญาณ handshaking เพียงสัญญาณใดสัญญาณหนึ่งเท่านั้น ยังมีสาย handshaking อื่นๆ อีกที่ใช้สำหรับแสดงหรือควบคุมสถานะของอุปกรณ์ เช่นต้องการจะบอกกับคอมพิวเตอร์ว่ามันไม่มีกระดาษก็จะสามารถกระทำได้โดยการทำให้สาย paper empty เป็น high จนกว่ามันได้รับกระดาษเข้าไป คุณสมบัตินี้มีอยู่ในพอร์ทขนานของ IBM แต่คอมพิวเตอร์อื่นอาจไม่มีคุณสมบัตินี้ก็ได้

อุปกรณ์ร่วมสามารถที่จะบอกคอมพิวเตอร์ขณะนี้ power up หรือ online ได้โดยสาย select(ขา13)เป็น high วิ่งสัญญาณนี้มีความจำเป็นในบางครั้งเท่านั้นเพราะอุปกรณ์ร่วมบางตัวสามารถทำ powerup ได้เองแต่จะถูก offline โดยการส่งสัญญาณในลักษณะพิเศษที่เรียกว่า deselect(ในคู่มือ printer คือ DC3 หรือ XOFF ซึ่งมีค่า ASCII เท่ากับ 19) ดังนั้นอุปกรณ์สามารถที่จะ online ได้อีกครั้งโดยการส่งสัญญาณ select (คือ DC1หรือ XON 17)

ตารางสัญญาณทั้งหมดที่นำไปใช้งาน

ชื่อสัญญาณ	ขาที่	แหล่งจ่าย	ใช้	รายระเอียด
Data strobe	1	คอมพิวเตอร์	ควบคุม	จะให้พัลส์เป็นlowไปควบคุมสัญญาณข้อมูล
Data 0-7	2-9	คอมพิวเตอร์	ข้อมูล	จะส่งสัญญาณ 8 บิตไปสร้างตัวอักษร
Acknowledge	10	อุปกรณ์ร่วม	ควบคุม	จะให้พัลส์เป็นlowเมื่อรับข้อมูลมาแล้ว
Busy	11	อุปกรณ์ร่วม	ควบคุม	จะให้พัลส์เป็นhighเมื่ออุปกร่วมไม่พร้อมที่จะทำงาน
Paper Empty	12	อุปกรณ์ร่วม	ควบคุม	จะให้พัลส์เป็นhighเมื่อต้องการกระดาษเพิ่ม
Select	13	อุปกรณ์ร่วม	ควบคุม	จะให้พัลส์เป็นhighเมื่ออุปกร่วม online
Auto Linefeed	14	คอมพิวเตอร์	ควบคุม	จะให้พัลส์เป็นlow เมื่ออุปกร่วมต้องการ linefeed
Signal Ground	16,33	-	กราวด์	สัญญาณอ้างอิง 0 โวลท์
Chassis Ground	17	อุปกรณ์ร่วม	กราวด์	กราวด์แท่น
+5 Volt	18,35	อุปกรณ์ร่วม	-	เป็นแรงดัน pull-up
Ground Returns	19-30	-	-	เป็นสายชีลด์ระหว่างขา 1-
Input Prime	31	คอมพิวเตอร์	ควบคุม	จะให้พัลส์เป็น low เพื่อรีเ อุปกรณ์ร่วม
Fault	32	อุปกรณ์ร่วม	ควบคุม	จะให้พัลส์เป็น low เพื่อแส สถานะของการผิดพลาด
Select	36	คอมพิวเตอร์	-	อีนาเบิล DC1/DC3 เมื่อ protocol เป็น high
Notused	15,36	-	-	

การกำหนดให้สาย autofeed เป็น low นั้นคอมพิวเตอร์จะสามารถบอกอุปกรณ์ร่วมในการ carriage return ในแต่ละครั้งโดยการ linefeed เช่นคอมพิวเตอร์จะบอกอุปกรณ์ร่วมว่าไม่สามารถส่ง linefeed มาได้ การเชื่อมโยงแบบเซ็นทรอนิกส์ จะมีกราวด์อยู่ 3 แบบด้วยกันได้แก่ กราวด์สัญญาณอยู่ที่ขา 16 และขา 33 ซึ่งเป็นกราวด์สัญญาณของอุปกรณ์ทั้ง 2 ซึ่งจะถูกต่อไปยังกราวด์ร่วม หรือจุดอ้างอิงสัญญาณ ส่วนกราวด์แท่น จะอยู่ที่ขา 17 ซึ่งบางครั้งจะถูกต่อกับ สายกราวด์กำลังในคอมพิวเตอร์หรืออุปกรณ์ เพื่อไม่ให้เกิดอันตรายกับอุปกรณ์ กราวด์ที่เหลืออีกหนึ่งคือ กราวด์รีเทิร์น ซึ่งใช้สำหรับการ ชีลด์สายสำคัญ จากสายอื่นๆ ไปยังสายแพ กราวด์รีเทิร์นจะอยู่ระหว่างสายสัญญาณ 2 เส้น เช่นสายกราวด์รีเทิร์นที่ขาที่ 19 ของคอนเน็กเตอร์เซ็นทรอนิกส์ จะอยู่ระหว่างสายที่ 1 กับขา 2 ซึ่งจะเป็นการเว้นระยะระหว่างสัญญาณทั้ง 2 โดยจะช่วยลดการคัปปลิ้งลง สายที่มีคุณภาพดีจะใช้สายชีลด์ที่มีการห่อหุ้มกราวด์รีเทิร์นรอบ ๆ สายสัญญาณแต่ละเส้น

ถ้าคอมพิวเตอร์ส่งพัลส์ low-going เข้าไปในสาย input initialize อุบกรณ์ร่วมจะสนับสนุนการกระทำสายนี้โดยการ resetค่าพารามิเตอร์ของมันไปยังค่า default ซึ่งเป็นโครงสร้างเริ่มแรกของเครื่อง หรือเป็นสถานะของ power up ที่อุปกรณ์นั้นเอง

อุปกรณ์สามารถแสดงค่าผิดพลาดทั่วไปได้โดยทำให้สาย faultเป็น low ซึ่งจะมีลักษณะคล้ายกับสาย busy บางครั้ง อุปกรณ์ร่วมจะใช้สายนี้ ในการแสดงว่ามัน offline หรือ out of paper ซึ่งก็เพียงพอแล้วสำหรับสัญญาณที่พอร์ตขนาน

บทที่ 6

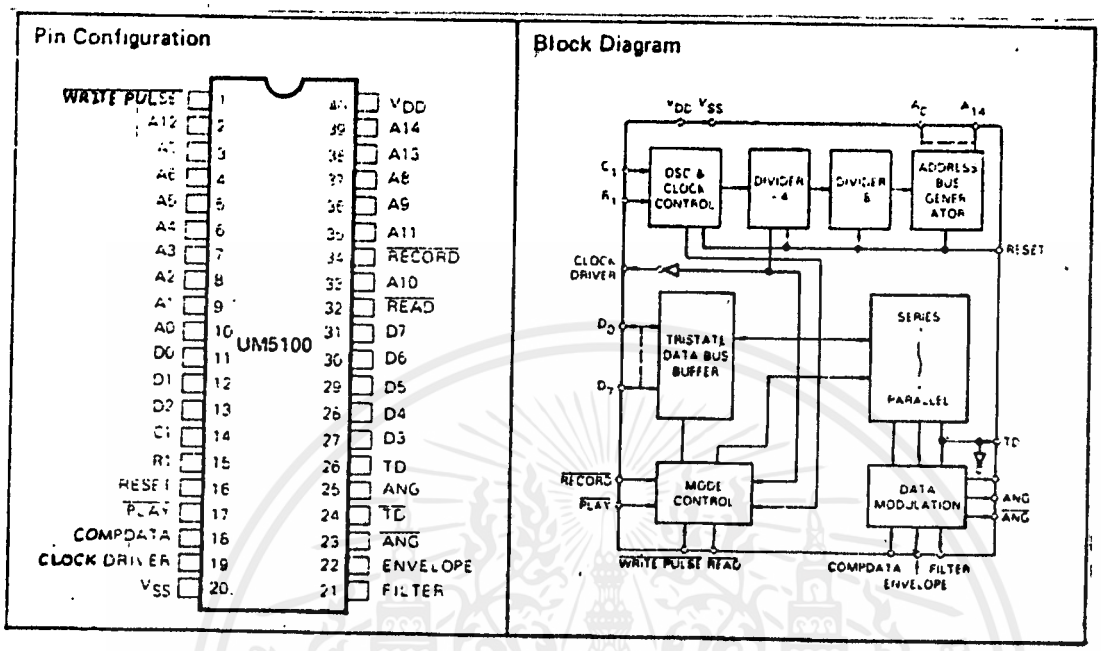
UM.5100 Voice Processor และการทำงานของวงจร

ลักษณะสำคัญ

- บันทึกเสียงและคัดลอกเสียงจะใช้หน่วยความจำประเภท SRAM หรือ EPROM
- ใช้หลักการของ Delta Modulation.
- การอ้างแอดเดรสสูงสุด 8 x 32 kb.
- แหล่งจ่ายไฟประมาณ 3 V-6 V.
- ใช้พลังงานต่ำเพราะโครงสร้างเป็น CMOS.
- มีแหล่งผลิตสัญญานาฬิการาคาถูก (RC-Osciltaion)
- ปรับอัตราการส่งข้อมูล (bit rates) จาก 10 K ถึง 28 kbps.
- เสียงที่บันทึกไว้มีคุณภาพสูง
- นำมาประยุกต์ใช้งานกับระบบรักษาความปลอดภัย การติดต่อสื่อสาร และการประยุกต์ใช้งานอื่น.

ลักษณะทั่วไป

UM 5100 เป็นชิพ (CMOS LSI ใช้สำหรับบันทึกเสียงไว้ใน SRAM และคัดลอกเสียงจาก EPROM หรือ ROM โดยมีหลักการคือแปลงสัญญาณจากสัญญาณเสียงซึ่งเป็นสัญญานอนาลอก โดยใช้ความถี่จาก RC oscillator แซมปลิ้งสัญญานจากอินพุทมาทำการมอดูเลชั่นโดยมีหลักการของ Delta Modutation ทำให้ได้สัญญาณแบบดิจิตอล และนำสัญญาที่ได้ไปเก็บไว้ในหน่วยความจำ สำหรับการนำสัญญาณที่เก็บไว้ในหน่วยความจำมาได้ ก็จะแปลงจากสัญญาณแบบไม่ต่อเนื่องมาแปลงเป็นสัญญาณแบบต่อเนื่อง ซึ่งก็จะได้สัญญาณเสียงที่เก็บไว้ได้นั่นเอง



รูปที่ 6.1 แสดงขาไอซีและ BLOCK DIAGRAM

โดยขาต่าง ๆ ทำหน้าที่ดังนี้

Vdd ทำหน้าที่เป็นขาที่รับไฟเลี้ยงวงจรในตัวไอซีโดยใช้ไฟเลี้ยงในช่วง 3-6 โวลท์

Vss ทำหน้าที่เป็นขากราวน์

A0-A7 ทำหน้าที่เป็นขาแอดเดรส

D0-D7 ทำหน้าที่เป็นขาดาต้า

RECORD ทำหน้าที่เป็นขา I/P ทำงานที่ "LOW"ใช้สำหรับบันทึกข้อมูลในโหมด"Speech analysis"

WRITE PULSE ทำหน้าที่เป็นขา O/Pทำงานที่ "LOW" ใช้สำหรับบันทึกข้อมูลในโหมด"Speech analysis"

READ ทำหน้าที่เป็นขา O/Pทำงานที่ "LOW"ใช้ในโหมดการอ่าน ซึ่งจะควบคุมหน่วยความจำภายนอก

PLAY ทำหน้าที่เป็นขา I/P ทำงานที่ "LOW" ใช้สำหรับกระตุ้นเมื่อเข้าในโหมด "Speech analysis"

RESET ทำหน้าที่เป็นขา I/Pทำงานที่ "HIGH"ใช้สำหรับการกลับเข้าสู่สภาวะเริ่มต้น

ANG และ ANG\ ทำหน้าที่เป็นขา ที่ใช้เปรียบเทียบสัญญาณ เพราะสัญญาณที่ขาทั้งสองจะ OUT OF PHASE

FILTER ทำหน้าที่เป็นขา O/P เป็นขาที่รับสัญญาณที่สร้างจากวงจร อินทีเกรเตอร์ภายนอก

ENVERLOPE ทำหน้าที่เป็นขาI/Pเป็นขาที่รับสัญญาณเข้ามาทำการ MODULATE VOICE AMPLITUDE

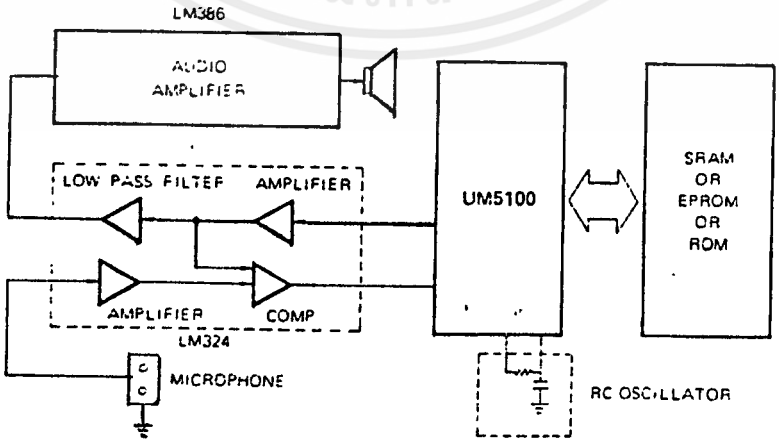
TD และ TD\ ทำหน้าที่เป็นขา O/P ที่ใช้สำหรับ LOW FREQUENCY

COMPDATA ทำหน้าที่เป็นขา I/P มีหน้าที่ในการ Detect delta slope โดยจะสร้างสัญญาณจากการเทียบสัญญาณอินพุทกับสัญญาณที่ป้อนกลับ

C,R ทำหน้าที่เป็นขา ที่ใช้ในการสร้าง RC Oscilator โดยมีความถี่เท่ากับ 40 KH$_z$

CLOCK DRIVER ทำหน้าที่เป็นขา O/P ใช้ในการสร้างแรงดันที่เป็นค่าลบ

Talk Back Application Block Diagram



การทำงานของวงจร speech recording and playback

การทำงานจะแบ่งเป็น 2 ขั้นตอนคือ

1. ช่วงบันทึกข้อมูล(recording)

เริ่มแรกไมโครโฟนจะรับสัญญาณเสียงแปลงเป็นสัญญาณทางไฟฟ้า ผ่านวงจรปรีแอมป์ (Pre amp) เพื่อขยายสัญญาณให้แรงขึ้นและส่งสัญญาณที่ได้นี้ไปยังวงจรเปรียบเทียบ(Compare circuit) โดยจะเปรียบเทียบสัญญาณที่ได้จากวงจรเพื่อหาผลต่างของสัญญาณเอาท์พุท สัญญาณที่ผ่านการเปรียบเทียบแล้วจะส่งค่าให้ขาที่ 18(compdata) เพื่อเอาสัญญาณที่ได้นี้ไปทำการ Modulate กับ Envellope signal และสัญญาณที่ได้จะถูกนำไปแปลงจาก Series ไปเป็น Parallel แล้วจะถูกนำไปเก็บที่ Tristate Data buffer (D0 - D7)

ถ้าหากมีการบ่งแอดเดรส(A0-A14) ไปที่ตำแหน่งของหน่วยความจำชั่วคราว (RAM) ที่ต้องการและที่ขา 34(record) เป็น "LOW" และขา 1 (write pulse)มีพัลซ์ส่งไปให้ ram เพื่อบอกว่าจะทำการติดต่อด้วย ข้อมูลที่ tristate data bus buffer (D0-D7) ก็จะถูกนำไปเขียนลงในหน่วยความจำชั่วคราว

2. ช่วงเล่นกลับ(Playback)

ในช่วงนี้ UM 5100 จะมีการบ่งแอดเดรสของหน่วยความจำอาจเป็น SRAM,EPROM หรือ ROM ที่ต้องการจะนำข้อมูลมา เล่นกลับโดยที่ขา PLAY และ READ ต้องเป็น "LOW" ข้อมูลจากหน่วยความจำจะถูกนำมาเก็บที่ tristate data bus buffer เพื่อนำมาแปลงกลับจาก paralel เป็นแบบ series แล้วนำมาผ่านเดลต้า มอดูเลชั่น เพื่อแปลงจากสัญญาณแบบดิจิตอลให้เป็นสัญญาณอนาลอก สัญญาณที่ได้จะนำมาผ่านวงจรดิฟเฟอร์เรนเชียนแอมป์ เพื่อหาผลต่างของสัญญาณสองจุด คือ TD,ANG และ TD,ANG สัญญาณที่ได้นี้จะถูกนำมาผ่านวงจรกรองสัญญาณความถี่ต่ำ เพื่อกรองเอาเฉพาะความถี่เสียงเท่านั้น แล้วนำสัญญาณที่ได้ไปทำการขยายอีกครั้งเพื่อป้อนให้ลำโพง

บทที่ 7

การออกแบบและการสร้าง

โครงงานนี้เป็นลักษณะการบันทึกเสียงพูดเก็บไว้ในหน่วยความจำโดยการเก็บข้อมูลแบบดิจิตอล และสามารถเรียกข้อมูลที่เป็นดิจิตอลนั้นกลับมาสังเคราะห์เป็นเสียงพูดได้ดังเดิม โดยใช้หลักการแปลงข้อมูล เดลต้ามอดูเลชั่นแบบความชันต่อเนื่อง(continous slope delta modulation) ข้อมูลที่ได้จากการแปลงนี้จะมีขนาด 8 บิต (D0-D7)

ข้อมูลที่เก็บไว้ในหน่วยความจำนั้นจะแบ่งเป็น 2 ส่วนคือส่วนของหน่วยความจำ EPROM และ หน่วยความจำ RAM หน่วยความจำ EPROM นั้นจะถูกบันทึกข้อมูลไว้เป็นที่แน่นอนแล้ว ซึ่งสามารถเล่นกลับได้อย่างเดียว โดยการเลือกตำแหน่งที่จะเล่นกลับนั้นจะถูกเลือกผ่านไมโครคอมพิวเตอร์ ซึ่งจะติดต่อกันทางพอร์ทขนาน ส่วนหน่วยความจำ RAM จะเป็นการบันทึกเสียงและเล่นกลับโดยข้อมูลในขณะนั้น ซึ่งเมื่อปิดเครื่องแล้วข้อมูลส่วนนี้ก็จะหายไปด้วย การบันทึกหรือเล่นกลับนี้ก็จะถูกเลือกโดยผ่านพอร์ทขนานของไมโครคอมพิวเตอร์อีกเช่นกัน

ลักษณะการออกแบบ

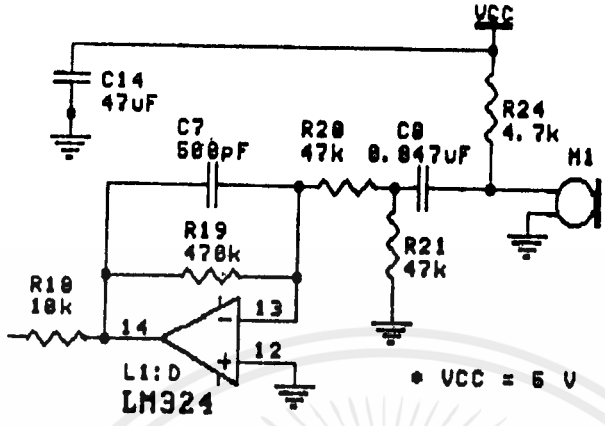
โครงงานนี้ประกอบไปด้วยส่วนที่คัญพอที่จะแยกได้เป็น 2 ส่วนใหญ่ๆคือส่วนที่เป็น voice processor และ ส่วนที่เป็นส่วนควบคุมฟังก์ชั่นการทำงาน

1. ส่วน voice processor เป็นส่วนที่ทำการบันทึกเสียงและเล่นกลับ โดยใช้ IC ประมวลผลสัญญาณเสียงเบอร์ UM 5100 เป็นตัวแปลงสัญญาณเสียงซึ่งเป็นสัญญาณ analog ให้เป็น digital ขนาด 8 bit และแปลงกลับจาก digital เป็น analog ตามเดิม ในส่วนนี้จะประกอบไปด้วยวงจรส่วนต่าง ๆ อีกหลายส่วนได้แก่ วงจร comparator , low pass filter,band pass filter,signal amplifier เป็นต้น ซึ่งจะแยกอธิบายการทำงานในส่วนย่อย ๆ ถึงรายระเอียดอีกต่อไป

2. ส่วนควบคุมเลือกฟังก์การทำงาน ในส่วนนี้จะถูกควบคุมโดยไมโครคอมพิวเตอร์ผ่านทางพอร์ทขนานโดยจะมีการเลือกให้ EPROM ตัวใดทำงานเลือกให้ SRAM ทำการอ่านหรือเขียนข้อมูล

1.ส่วน INPUT

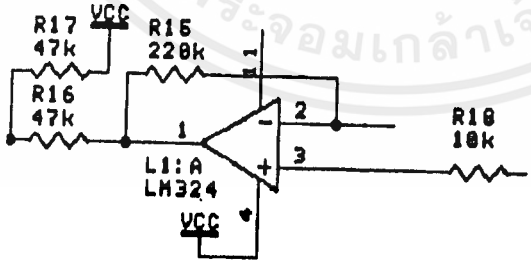
จะประกอบไปด้วยชุด microphone และ amplifier ดังรูป



สัญญาณเสียงจะถูกรับเข้ามาโดยผ่าน microphone แต่สัญญาณที่ได้มีขนาดที่เล็ก ฉะนั้นจึงต้องผ่านวงจร op-amp ที่ต่อเป็นวงจรอินทีเกรเตอร์ เพื่อให้สัญญาณมีขนาดใหญ่ขึ้น

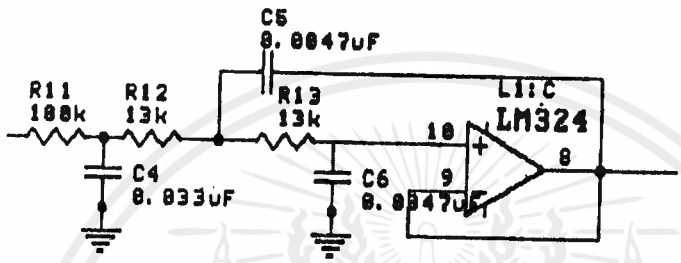
2.วงจร comparator

จะเป็นวงจรที่รับสัญญาณทาง input ซึ่งเป็น input ปัจจุบันแบบ analog เพื่อมาเปรียบเทียบกับสัญญาณ output ก่อนหน้าซึ่งเป็น analog เช่นเดียวกันก่อนที่จะผ่านไปยังวงจร Delta modulation ซึ่งวงจรจะเป็นดังรูป



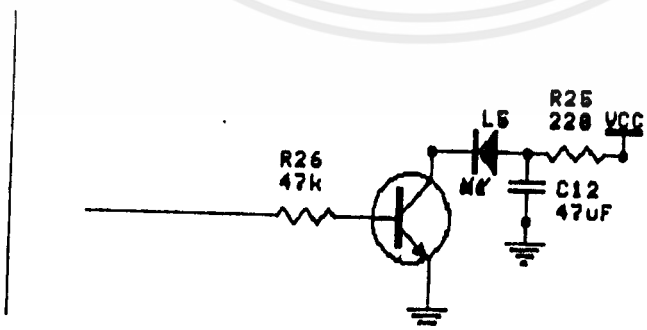
3.วงจร Low pass filter

จะเป็นวงจรที่กรองเอาความถี่สูงออกไป ให้เหลือแต่เพียงความถี่ต่ำออกไปยังวงจร AUDIO AMPLIFIER ต่อไป ซึ่งวงจร Low pass filter ในโครงงานนี้จะเป็นแบบอันดับ 2 (2 order) การที่ใช้แบบอันดับ 2 ก็เพื่อที่จะให้ประสิทธิภาพในการ filter ดีขึ้นนั่นเอง วงจรจะเป็นดังรูป



4.ส่วนแสดงผล

-วงจรนี้เป็นการแสดงสถานะการทำงานของวงจรโดยใช้ LED เป็นตัวแสดงผล โดยเมื่อมีการบันทึกหรือเล่นกลับของเสียงนั้น ความสว่างของ LED เป็นตัวบอกความแรงของสัญญาณที่บันทึกและเล่นกลับ คือหากมีสัญญาณดังมาก LED ก็จะสว่างมาก และสัญญาณที่ค่อย LED ก้จะสว่างน้อย โดยจะกระพริบตามสัญญาณ



2. ส่วนควบคุมการเลือกฟังก์ชั่นการทำงาน

ในส่วนนี้การเลือกฟังก์ชั่นการทำงานจะเลือกได้โดยผ่านไมโครคอมพิวเตอร์ โดยจะส่งข้อมูลออกทางพอร์ทขนานทางขา DATA (D0-D7) เพื่อเอาสัญญาณนี้ไปควบคุมการเลือกฟังก์ชั่นการทำงานต่าง ๆ ดังนี้

ขา D0 นำไปใช้ select ให้ EPROM ตัวที่ 4 ทำงาน

ขา D1 นำไปใช้ select ให้ EPROM ตัวที่ 3 ทำงาน

ขา D2 นำไปใช้ select ให้ EPROM ตัวที่ 2 ทำงาน

ขา D3 นำไปใช้ select ให้ EPROM ตัวที่ 1 ทำงาน

ขา D4 นำไปทริกให้ SRAM ทำงาน

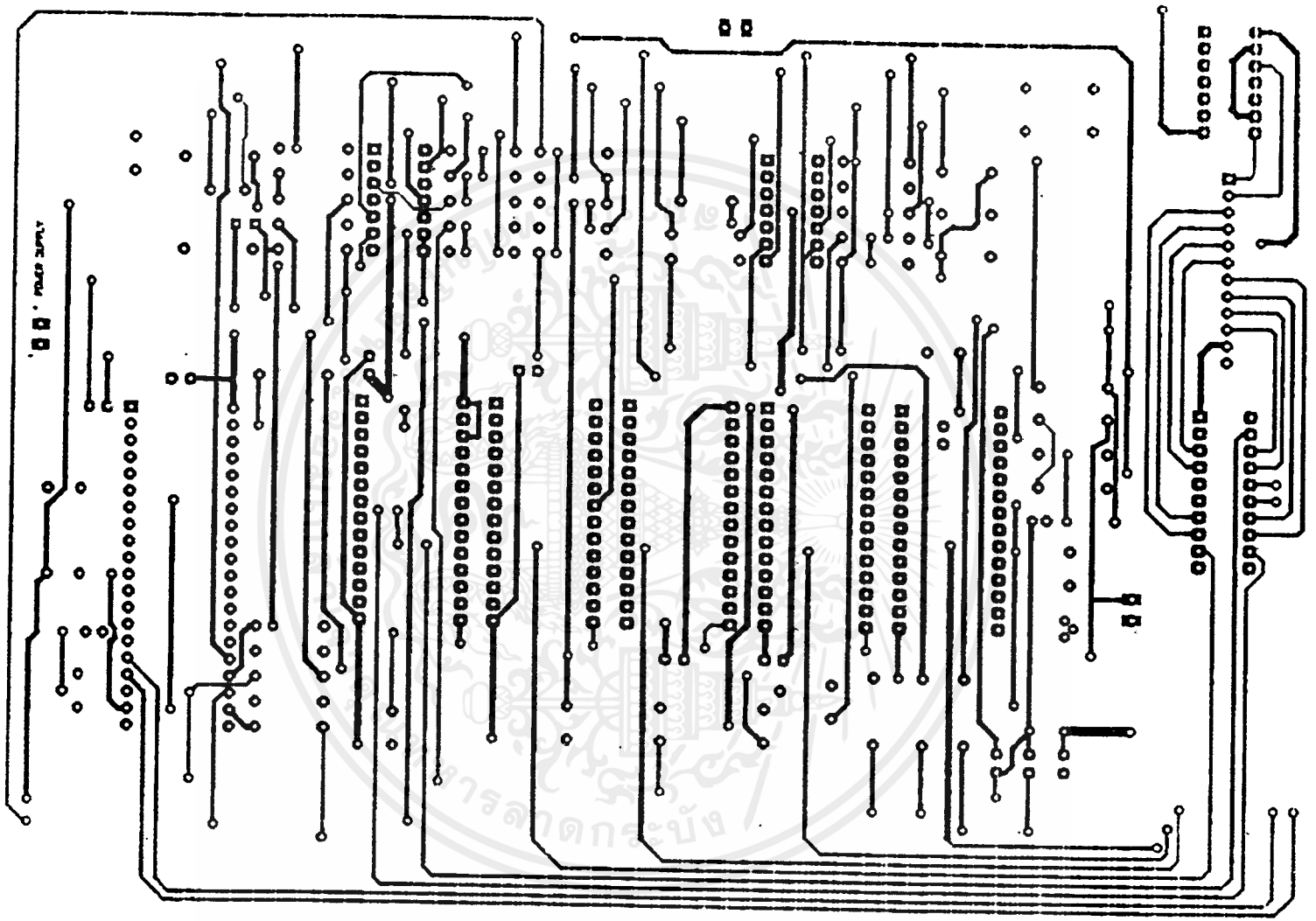
ขา D5 นำไปทริกขา RESET ของ # UM5100

ขา D6 นำไปทริกขา PLAY ของ # UM5100 เพื่อให้วงจรทำงานในสภาวะเล่นกลับ

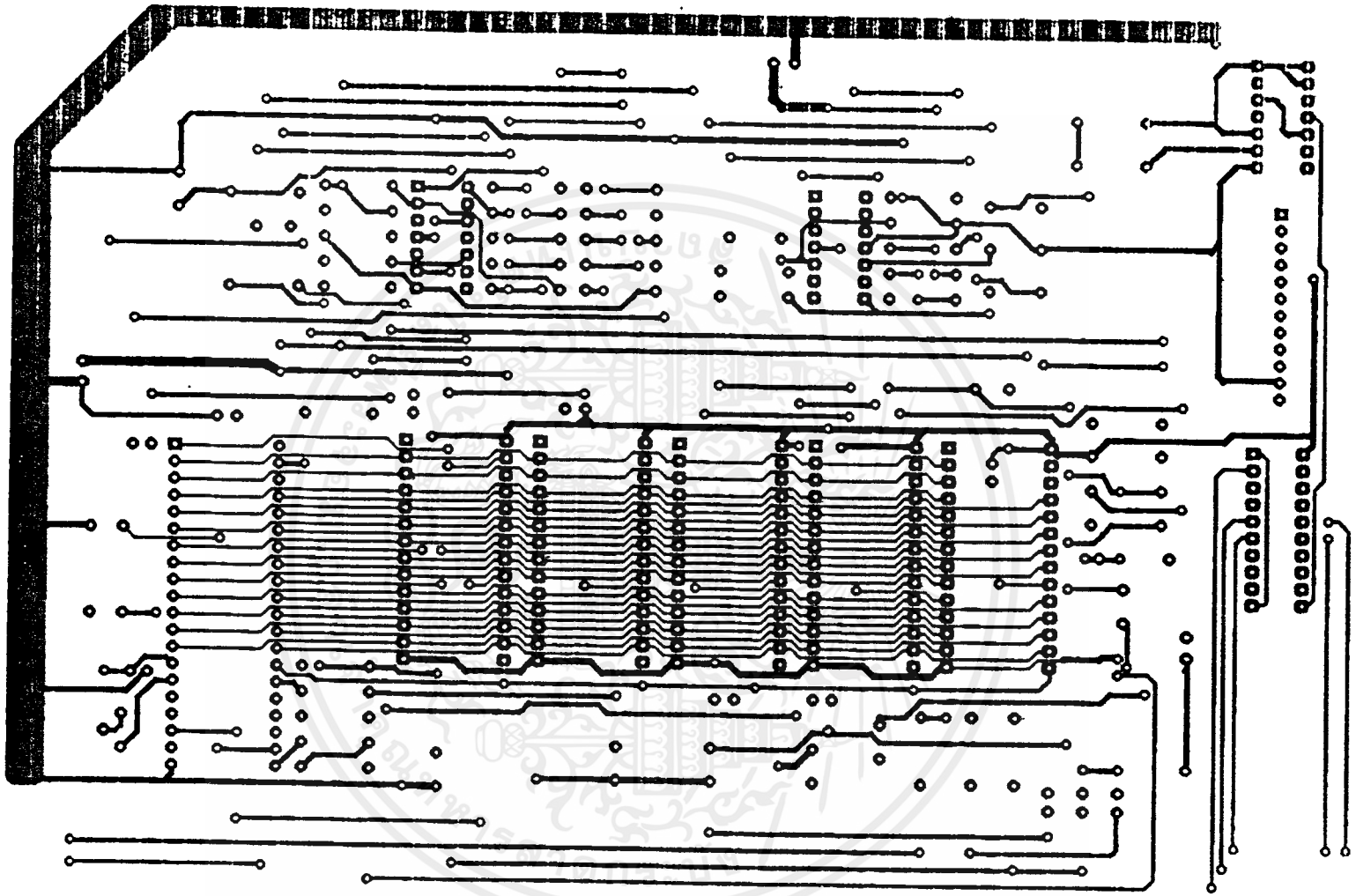
ขา D7 นำไปทริกขา RECORD ของ # UM5100 เพื่อให้วงจรทำงานในสภาวะบันทึกสัญญาณข้อมูล

เพื่อให้ง่ายต่อการเลือกการทำงาน การเลือกนี้จะเป็นลักษณะ menu แสดงออกที่จอภาพของคอมพิวเตอร์ ซึ่ง menu นี้จะถูกเขียนโดยโปรแกรมภาษา c รวมทั้งการส่งข้อมูลออกพอร์ทตามที่เลือกจาก menu ก็จะถูกเขียนรวมอยู่ในโปรแกรมนี้ด้วย ดังที่แสดงอยู่ในภาคผนวก

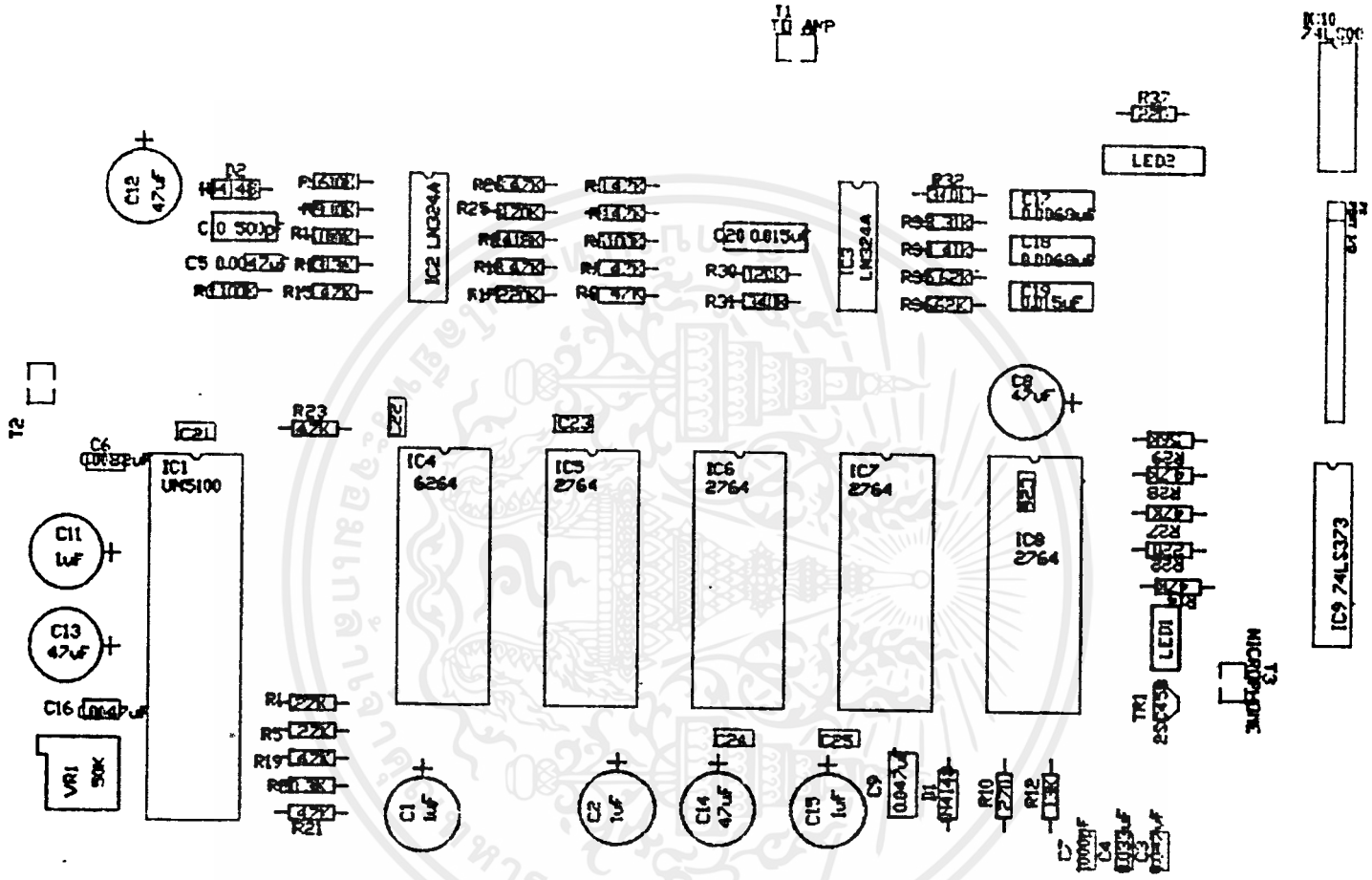
ภาคผนวก



NOY Top Layer



NOY Bottom Layer



NOY Too Overlay

```
/*  Program Pull-down menu for project      */
/*  This program is used for uot of data    */
/*  from printer port to Control device     */
/*  on circuit in Voice Synthesizer Project */

#include <stdio.h>
#include <dos.h>
#include <stdlib.h>
#include <bios.h>
#include <ctype.h>
#include <string.h>

#define BORDER     1
#define ESC        27
#define MAX_FRAME  10
#define REV_VID    0x50
#define NORM_VID   11

#define port       0x378 /* Number of printer port */

void save_video(int num);
void restore_video(int num);
void goto_xy(int x, int y), cls(void);
void write_video(int x, int y, char *p, int attrib);
void display_menu(int num);
void write_string(int x, int y, char *p, int attrib);
void write_char(int x, int y, char ch, int attrib);
void pd_driver(void);
```

```
int make_menu(int num, char *menu[], char *keys,
              int count, int x, int y, int border);
int get_resp(int num), pulldown(int num);
int video_mode(void);
void display_menu(int num), draw_border(int num);
char far *vid_mem;
int readkey(void);

struct menu_frame {
  int startx, endx, starty, endy;
  unsigned char *p;
  char **menu;
  char *keys;
  int border, count;
  int active;
} frame[MAX_FRAME], i;

char *Main[]= {
  "   Main Menu for operating. ",
  " Menu1.Read voice from EPROM",
  " Menu2.Read voice from SRAM ",
  " Menu3.Record voice to SRAM ",
  " Exit                       ",
};
char *Sub1[]= {
  " Menu1.Read voice from EPROM",
  "-----------------------------",
  " Read voice from EPROM #1   ",
  " Read voice from EPROM #2   ",
  " Read voice from EPROM #3   ",
```

```
    " Read voice from EPROM #4    ",
    " Return To Main Menu         ",
};

char *Sub2[]= {
    " Menu2. Read voice from SRAM ",
    "_____________________________",
    "             OK              ",
    " Return To Main Menu         ",
};

char *Sub3[]= {
    " Menu3. Recordd voice from SRAM ",
    "_____________________________",
    "             OK              ",
    " Return To Main Menu         ",
};

main(void)
  {
  cls();
  goto_xy(0, 0);
  /* first, create the menu frames */
  make_menu(0, Main, " ", 6, 20, 3, BORDER);
  make_menu(1, Sub1, " ", 7, 20, 12, BORDER);
  make_menu(2, Sub2, " ", 4, 5, 12, BORDER);
  make_menu(3, Sub3, " ", 4, 35, 12, BORDER);

  pd_driver(); /* activate the menu system */
```

```
                    delay(100);
                    outport(port,0);
                    break;
                case 4:     /* Read voice from EPROM #3 */
                    outport(port,157);
                    delay(100);
                    outport(port,0);
                    break;
                case 5:                 /* Read voice from EPROM
                    outport(port,158);
                    delay(100);
                    outport(port,0);
                    break;
                case 6:
                    return -1;
                    break;
                }
            }
            restore_video(1);
            break;

case 2:
            if((choice2= pulldown(2)) != -1)
               {
                 switch(choice2)
                    {
                       case 0:
                           break;
                       case 1:
                           break;
```

```
                    case 2:;
                    case 3:          /* Read voice from SRAM +
                        outport(port,143);
                        delay(100);
                        outport(port,0);
                        break;
                    case 4:
                        return -1;
                        break;
                }
            }
        selection = pulldown(2);
        restore_video(2);
        break;
case 3:
       if((choice2= pulldown(3)) != -1)
        {
          switch(choice2)
            {
              case 0:
                  break;
              case 1:;
              case 2:          /* Record voice to SRAM +
                  outport(port,79);
                  delay(100);
                  outport(port,0);
                  break;
             case 3:
                  return -1;
                  break;
```

```
                        }
                    }
                    selection = pulldown(3);
                    restore_video(3);
                    break;
            case 4:
                    return -1;
                    break;
        }
    }
    restore_video(0);
}

/* Display a pull-down menu and return selection. */
int pulldown(int num)
{
    int  vmode;

    vmode = video_mode();
    if((vmode!=2) && (vmode!=3) && (vmode!=7))
     {
      printf("video must be in 80 column text mode");
      exit(1);
     }
    /* set proper address of video RAM */
    if(vmode==7) vid_mem = (char far *) 0xB0000000;
    else vid_mem = (char far *) 0xB8000000;

    /* get active window */
    if(!frame[num].active)  /* not currently in use */
```

```
    { save_video(num);      /* save the current screen */
      frame[num].active = 1; /* set active flag */
    }

  if(frame[num].border) draw_border(num);

  display_menu(num);     /* display the menu */
  return get_resp(num); /* return response */
}

/* Construct a pull down menu frame.
   It returns 1 if the menu frame can be constructed;
   otherwise 0 is returned.
*/
make_menu(
  int num,       /* menu number */
  char *menu[], /* menu text */
  char *keys,    /* hot keys */
  int count,     /* number of menu items */
  int x, int y, /* X,Y coordinates of left hand corner */
  int border     /* no border if 0 */
)
{
  register int i, len;
  int endx, endy;
  unsigned char *p;

  if(num>MAX_FRAME) {
    printf("Too many menus\n");
```

```
    return 0;
  }

  if((y>24) || (y<0) || (x>79) || (x<0)) {
    printf("range error");
    return 0;
  }

  /* compute the size */
  len = 0;
  for(i=0; i<count; i++)
   if(strlen(menu[i]) > len) len = strlen(menu[i]);
  endx = len + 2 + x;
  endy = count + 1 + y;
  if((endy+1>24) || (endx+1>79)) {
    printf("menu won't fit");
    return 0;
  }

  /* allocate enough memory to hold current contents
     of the screen */
  p = (unsigned char *) malloc(2 * (endx-x+1) * (endy-y+1));
  if(!p) exit(1); /* put your own error handler here */

  /* construct the frame */
  frame[num].startx = x; frame[num].endx = endx;
  frame[num].starty = y; frame[num].endy = endy;
  frame[num].p = p;
  frame[num].menu = (char **) menu;
  frame[num].border = border;
```

```
    frame[num].keys = keys;
    frame[num].count = count;
    frame[num].active = 0;
    return i;
}

/* Display the menu in its proper location. */
void display_menu(int num)
{
    register int i, y;
    char **m;

    y = frame[num].starty+1;
    m = frame[num].menu;

    for(i=0; i<frame[num].count; i++, y++)
        write_string(frame[num].startx+1, y, m[i], NORM_VID);
}

/* Draw a border around the menu. */
void draw_border(int num)
{
    register int i;
    char far *v, far *t;

    v = vid_mem;
    t = v;

    /* draw vertical lines */
    for(i=frame[num].starty+1; i<frame[num].endy; i++) {
```

```
    v += (i*160) + frame[num].startx*2;
    *v++ = 179;
    *v = NORM_VID;
    v = t;
    v += (i*160) + frame[num].endx*2;
    *v++ = 179;
    *v = NORM_VID;
    v = t;
}

/* draw horizontal lines */
for(i=frame[num].startx+1; i<frame[num].endx; i++) {
    v += (frame[num].starty*160) + i*2;
    *v++ = 196;
    *v = NORM_VID;
    v = t;
    v += (frame[num].endy*160) + i*2;
    *v++ = 196;
    *v = NORM_VID;
    v = t;
}

/* draw corners */
write_char(frame[num].startx, frame[num].starty,
            (char) 218, NORM_VID);
write_char(frame[num].startx, frame[num].endy,
            (char) 192, NORM_VID);
write_char(frame[num].endx, frame[num].starty,
            (char) 191, NORM_VID);
write_char(frame[num].endx, frame[num].endy,
```

```
}


/* Clear the screen. */
void cls(void)
{
  union REGS r;

  r.h.ah = 6; /* screen scroll code */
  r.h.al = 0; /* clear screen code */
  r.h.ch = 0; /* start row */
  r.h.cl = 0; /* start column */
  r.h.dh = 24; /* end row */
  r.h.dl = 79; /* end column */
  r.h.bh = 7;  /* blank line is black */
  int86(0x10, &r, &r);
}

/* Send the cursor to x,y. */
void goto_xy(int x, int y)
{
  union REGS r;

  r.h.ah = 2; /* cursor addressing function */
  r.h.dl = x; /* column coordinate */
  r.h.dh = y; /* row coordinate */
  r.h.bh = 0; /* video page */
  int86(0x10, &r, &r);
}
```

```
        /* see if it is a hot key */
        key_choice = strchr(frame[num].keys, tolower(c.ch[0]));
        if(key_choice) return key_choice-frame[num].keys;

        /* check for ENTER or space bar */
        switch(c.ch[0]) {
          case '\r': return arrow_choice;
          case ' ' : arrow_choice++;
            break;
          case ESC : return -1; /* cancel */
        }
      }
      else {  /* is special key */
        switch(c.ch[1]) {
          case 72: arrow_choice--; /* up arrow */
            break;
          case 80: arrow_choice++; /* down arrow */
            break;
        }
      }
      if(arrow_choice==frame[num].count) arrow_choice=0;
      if(arrow_choice<0) arrow_choice = frame[num].count-1;

      /* highlight the next selection */
      goto_xy(x, y+arrow_choice);
      write_string(x, y+arrow_choice,
        frame[num].menu[arrow_choice], REV_VID);
    }
}
```

```
      /* see if it is a hot key */
      key_choice = strchr(frame[num].keys, tolower(c.ch[0]));
      if(key_choice) return key_choice-frame[num].keys;

      /* check for ENTER or space bar */
      switch(c.ch[0]) {
        case '\r': return arrow_choice;
        case ' ' : arrow_choice++;
          break;
        case ESC : return -1; /* cancel */
      }
    }
    else {  /* is special key */
      switch(c.ch[1]) {
        case 72: arrow_choice--; /* up arrow */
          break;
        case 80: arrow_choice++; /* down arrow */
          break;
      }
    }
    if(arrow_choice==frame[num].count) arrow_choice=0;
    if(arrow_choice<0) arrow_choice = frame[num].count-1;

    /* highlight the next selection */
    goto_xy(x, y+arrow_choice);
    write_string(x, y+arrow_choice,
      frame[num].menu[arrow_choice], REV_VID);
  }
}
```

```
  buf_ptr = frame[num].p;
  v = vid_mem;
  for(i=frame[num].startx; i<frame[num].endx+1; i++)
    for(j=frame[num].starty; j<frame[num].endy+1; j++) {
      t = (v + (j*160) + i*2);
      *buf_ptr++ = *t++;
      *buf_ptr++ = *t;
      *(t-1) = ' ';  /* clear the window */
    }
}

/* Restore a portion of the screen. */
void restore_video(int num)
{
  register int i,j;
  char far *v, far *t;
  char *buf_ptr;

  buf_ptr = frame[num].p;
  v = vid_mem;
  t = v;
  for(i=frame[num].startx; i<frame[num].endx+1; i++)
    for(j=frame[num].starty; j<frame[num].endy+1; j++) {
      v = t;
      v += (j*160) + i*2; /* compute the address */
      *v++ = *buf_ptr++;  /* write the character */
      *v = *buf_ptr++;    /* write the attribute */
  }
  frame[num].active = 0; /* deactivate */
```

```
    v += (i*160) + frame[num].startx*2;
    *v++ = 179;
    *v = NORM_VID;
    v = t;
    v += (i*160) + frame[num].endx*2;
    *v++ = 179;
    *v = NORM_VID;
    v = t;
}

/* draw horizontal lines */
for(i=frame[num].startx+1; i<frame[num].endx; i++) {
    v += (frame[num].starty*160) + i*2;
    *v++ = 196;
    *v = NORM_VID;
    v = t;
    v += (frame[num].endy*160) + i*2;
    *v++ = 196;
    *v = NORM_VID;
    v = t;
}

/* draw corners */
write_char(frame[num].startx, frame[num].starty,
            (char) 218, NORM_VID);
write_char(frame[num].startx, frame[num].endy,
            (char) 192, NORM_VID);
write_char(frame[num].endx, frame[num].starty,
            (char) 191, NORM_VID);
write_char(frame[num].endx, frame[num].endy,
```

```
/* returns the current video mode */
video_mode(void)
{
  union REGS r;

  r.h.ah = 15;  /* get video mode */
  return int86(0x10, &r, &r) & 255;
}

/* Return the 16-bit scan code from the keyboard. */
readkey(void)
{
  union REGS r;

  r.h.ah = 0;
  return int86(0x16, &r, &r);
}
```

กิตติกรรมประกาศ

ขอขอบพระคุณ อาจารย์ ภากร หุตะสังกาศ เป็นอย่างสูง ที่ได้ให้การประสิทธิ์ประสาทวิชา ให้คำแนะนำปรึกษาในเรื่องต่าง ๆ แก่คณะผู้จัดทำ และขอขอบพระคุณอาจารย์ วิริยะ กองรัตน์ ที่ได้ให้คำแนะนำเกี่ยวกับการโปรแกรม EPROM และขอบคุณเพื่อน ๆ ที่ได้ช่วยเหลือ และช่วยเป็นกำลังใจ ในการทำปริญญานิพนธ์นี้จนสำเร็จลุล่วงได้ด้วยดี

P1
12PIN
SPARE
GND
D7
D6
D5
D4
D3
D2
D1
D0
STOBE
BUSY

IC9
74LS373
D7 D6 D5 D4 D3 D2 D1 D0
Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0
LE
OE

IC10:D
74LS08

IC10:C
74LS08

IC10:A
74LS08

IC10:B
74LS08

R37
220

LED2

VCC

C15
1uF

VR1
50K

C16
0.0047uF

R25
47K

IC1
UM6800

VDD
CLOCK
READ
WRITE
A13
A12
A11
A9
A8
A7
A6
A5
A4
A3
A2
A1
A0
D7
D6
D5
D4
D3
D2
D1
D0
VSS

C13
47uF

D1
1N4148

D2
1N4148

C12
47uF

LED1

R22
220

R21
47K

TR1
2SC458

C14
47uF

R20
3K

C11
1uF

R17
220K

R19
47K

R18
47K

IC2:A
LM324A

C10
500pF

R25
470K

C8
47uF

R24
18K

IC2:D
LM324A

R26
47K

C9
0.047uF

R27
47K

R23
4.7K

M1

C2
1uF

R6
100K

R7
47K

R5
27K

R4
47K

R3
47K

R1
27K

R2
100K

C1
1uF

IC2:B
LM324A

R16
18K

R9
18K

R10
270

C3
0.047uF

R8
47K

C5
0.0047uF

R11
100K

R12
13K

R13
13K

C4
0.033uF

IC2:C
LM324A

C17
0.0068uF

R14
47K

C6
0.00824uF

R29
56K

R30
120K

IC3:C
LM324A

R15
47K

C7
1000pF

C18
0.0068uF

R32
340K

R31
340K

R34
41K

C20
0.015uF

IC3:B
LM324A

C19
0.015uF

R33
31K

R36
62K

R35
62K

P2
2PIN
SIG
GND

TO AUDIO AMPLIFIER

IC4
6264

IC5
2764

IC6
2764

IC7
2764

IC8
2764