โปรแกรมสื่อสารคอมพิวเตอร์ด้วยระบบมัลติมีเดีย

MULTIMEDIA COMMUNICATION SOFTWARE

โดย

นางสาววิภารัตน์ สารอภิสิทธิคาม

นางสาวอภิรพร ชีวโรจน์ฐากูร

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2538

โปรแกรมสื่อสารคอมพิวเตอร์ด้วยระบบมัลติมีเดีย

Multimedia Communication Software

โดย

นางสาววิภารัตน์ สารอภิสิทธิคาม 35104389

นางสาวอภิรพร ชีวโรจน์ฐากูร 35104531

อาจารย์ที่ปรึกษา

ผศ.ดร. สุวิพล สิทธิชีวภาค

ปริญญานิพนธ์สำหรับปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2538

ปริญญานิพนธ์ปีการศึกษา 2538

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง โปรแกรมสื่อสารคอมพิวเตอร์ด้วยระบบมัลติมีเดีย

Multimedia Communication Software

ผู้จัดทำ

1. นางสาววิภารัตน์ สารอภิสิทธิคาม 35104389
2. นางสาวอภิรพร ชีวโรจน์ฐากูร 35104531

..... อาจารย์ที่ปรึกษา

(ผศ.ดร. สุวิพล สิทธิชีวภาค)

โปรแกรมสื่อสารคอมพิวเตอร์ด้วยระบบมัลติมีเดีย
Multimedia Communication Software

โดย นางสาววิภารัตน์ สารอภิสิทธิคาม 35104389
นางสาวอภิรพร ชีวโรจน์ฐากูร 35104531

อาจารย์ที่ปรึกษา ผศ.ดร.สุวิพล สิทธิชีวภาค

บทคัดย่อ

โครงการนี้เป็นการแสดงถึงการติดต่อสื่อสารผ่านพอร์ตอนุกรมด้วยระบบมัลติมีเดีย ซึ่งมีจุดประสงค์เพื่อให้คอมพิวเตอร์สามารถติดต่อกันผ่านทางสื่อต่าง ๆ เช่น ตัวอักษร ภาพ และเสียง เพื่อให้เป็นระบบการสื่อสารที่สื่อความหมายระหว่างผู้ใช้ได้

โปรแกรมที่สร้างในโครงงานนี้มีลักษณะง่าย ๆ และเป็นพื้นฐานในการทำความเข้าใจเกี่ยวกับระบบมัลติมีเดีย ทำการส่งข้อมูลผ่านพอร์ตอนุกรมซึ่งมีอัตราเร็ว 9,600 บอด โดยจะทำให้สามารถส่งข้อมูลได้ช้า การสื่อสารแบ่งเป็นการสื่อสารเฉพาะในโหมดข้อความที่เป็นการสื่อสารแบบสองทิศทาง , การสื่อสารด้วยภาพประกอบข้อความ และการฝากข่าวสารทางเสียงด้วยการบันทึกเป็นไฟล์ แล้วส่งไปยังผู้รับ ซึ่งสามารถเลือกใช้ได้ตามต้องการ

Abstract

The program uses in this project is fundamental understanding of multimedia system. The data transfers through serial port with 9,600 baud rate. The communication is divided into many modes ; text mode two - way communication, image and text mode, and sound mode, but the sound is recoded as a file. The sound is tranmitted to the terminal, and it be chosen the proper mode for each application.

สารบัญ

สารบัญภาพ

บทที่ 1
บทนำ

มัลติมีเดีย (Multimedia)

มัลติมีเดีย หรือระบบหลายสื่อ หรือระบบหลายตัวกลาง หมายถึงตัวกลางประสาน หรือช่วยอำนวยความสะดวกให้เกิดความรู้ ความเข้าใจ ได้แก่ ตัวอักษร รูปภาพ เสียง และหมายรวมไปถึงอุปกรณ์ต่าง ๆ ที่จะบรรจุ หรือใช้แสดงสิ่งต่าง ๆ ที่กล่าวมาทั้งหมด เช่น สมุด หนังสือ กระดานดำ แผ่นเสียง แถบบันทึกเสียง เป็นต้น นอกจากนั้นยังรวมไปถึงการนำข่าวสารจากที่หนึ่งไปยังอีกที่หนึ่ง โดยอาจส่งไปในลักษณะของตัวอักษร รูปภาพ เสียง ด้วยกรรมวิธีที่ต่างกันไป มีตัวกลางที่ทำหน้าที่ให้บริการ หรือเป็นสื่อ ปัจจุบันสื่อเหล่านี้ถูกนำไปใช้มากมายหลากหลายรูปแบบ และในหลายวงการ ส่งผลให้ขอบเขตของสื่อขยายออกไปมากขึ้นเรื่อย ๆ โดยเฉพาะอย่างยิ่ง เมื่อมีการคิดค้นเทคโนโลยีใหม่ ๆ มากขึ้นเพียงใด รูปแบบของสื่อก็จะยิ่งหลากหลายมากขึ้น อีกทั้งยังสะดวก และรวดเร็วด้วย หากจะกล่าวถึงส่วนของมัลติมีเดีย อาจกล่าวได้ 5 ส่วน คือ ข่ายคอมพิวเตอร์, ข่ายการสื่อสาร, ข่ายสาธารณะ, ข่ายออดิโอ-วิดีโออิเล็กทรอนิกส์ และข่ายการถ่ายทอดภาพยนตร์หรือโทรทัศน์

ระบบมัลติมีเดียคืออะไร

ในที่นี้เป็นนิยามอย่างกว้าง ๆ ของระบบมัลติมีเดีย คือการรวบรวมสื่อต่าง ๆ ที่มีองค์ประกอบสำคัญใหญ่ ๆ 3 อย่างคือ ภาพ เสียง และข้อมูล

1. เรื่องของภาพ ภาพเป็นสื่อที่มนุษย์เรานิยมใช้กันมาตั้งแต่อดีต และนิยมใช้มาถึงปัจจุบัน ความจริงแล้วสื่อที่เป็นเสียงนั้นเป็นสื่อแรกที่มนุษย์รู้จักและใช้ แต่เสียงนั้นมีข้อจำกัดอยู่ที่ระยะทางและภาษา หากสังเกตร่องรอยที่ถูกบันทึกไว้โดยมนุษย์ในยุคแรก ๆ มักจะพบว่าเป็นภาพวาดต่าง ๆ ทั้งที่ปรากฏบนผนังถ้ำหรือบนสิ่งของเครื่องใช้ เพราะสามารถสื่อสารให้เข้าใจกันง่าย

ภาพในระบบมัลติมีเดีย แบ่งออกเป็นประเภทใหญ่ ๆ ได้ดังนี้ คือ

1.1 ตัวอักษร (Text)

1.2 รูปภาพ (Image) และสามารถแบ่งตามลักษณะของภาพเป็นประเภทใหญ่ ๆ ได้อีกคือ ภาพนิ่ง และภาพเคลื่อนไหว ถ้าหากพิจารณาถึงที่มาของภาพแล้วสามารถจำแนกออกเป็นสองลักษณะคือ

1.2.1 ภาพจากการจำลองแบบ (Imitated Image) ได้แก่ ภาพที่ถ่ายแบบมาจากของจริงโดยอาศัยเครื่องมือช่วยในการจำลองแบบ เช่น กล้องถ่ายภาพนิ่ง กล้องวิดีโอ เครื่องอ่านภาพ (Scanner) เป็นต้น

1.2.2 ภาพจากการสร้าง(Creative Images) เป็นภาพที่สร้างขึ้นจากจินตนาการมนุษย์ เช่น ภาพวาด ภาพการ์ตูน เป็นต้น

เหตุที่เราไม่แยกตัวอักษรกับภาพออกจากกันก็เพราะถ้ามองในส่วนของการใช้งานคอมพิวเตอร์ในปัจจุบัน ตัวอักษรจัดเป็นภาพจากการสร้าง โดยที่การสร้างแบบตัวอักษร (Font) จะมีลักษณะการสร้างในเชิงรูปภาพ

2. **เรื่องของเสียง** เสียงนับได้ว่าเป็นสื่อแรกที่มนุษย์ใช้ และเสียงหรือภาษาพูดก็เป็นสื่อที่มีความสำคัญไม่น้อยไปกว่าภาษาเขียน การผลิตภาษาเชิงภาพจะมีขั้นตอนยุ่งยากกว่าภาษาเสียงซึ่งอาศัยหูเป็นสื่อกลางเพียงอย่างเดียวก็ใช้งานได้แล้ว อย่างไรก็ตามเสียงก็มีข้อจำกัดที่ลดลงไปมากกว่าในอดีต เนื่องจากการพัฒนาสื่อทางเสียงได้รับการพัฒนาไปเร็วกว่าสื่อทางภาพ และที่สำคัญการพัฒนาสื่อทางเสียงจะไม่ร้ายแรงเท่ากับความผิดพลาดในเชิงภาพ ทั้งนี้เพราะความผิดพลาดของเสียงจะมีผลแค่เพียงทำให้ความชัดเจนของเสียงลดลง หรือเพี้ยนไปจากเดิม เช่นทุ้มเกินไป แหลมเกินไป เป็นต้น แต่ถ้าเป็นความผิดพลาดในลักษณะของสื่อทางภาพแล้ว อาจถึงขั้นทำให้ใช้งานไม่ได้เลยทีเดียว ระบบมัลติมีเดียจึงเป็นการประสานกันระหว่างสื่อทางภาพกับสื่อทางเสียงนั่นเอง

3. **การตอบโต้ซึ่งกันและกัน** คำว่าตอบโต้ซึ่งกันและกัน (Interactive) หมายถึง การแสดงผลตอบสนองการทำงานว่าถูกต้องหรือไม่ ใช้ได้หรือไม่ในขณะนั้น ระบบมัลติมีเดียจะต้องมีการตอบโต้ซึ่งกันและกันได้จึงจะสมบูรณ์ จากลักษณะอันนี้เองที่ทำให้ขอบเขตของมัลติมีเดียกว้างขวางออกไป จำแนกได้ดังนี้

3.1 การตอบโต้กันได้ในลักษณะตัวเลือก มีการจำลองสถานการณ์ (Simulation) จากปัญหาหนึ่งไว้หลายรูปแบบ โดยพิจารณาจากเงื่อนไขที่แตกต่างกัน หากสามารถจำลองสถานการณ์ได้มาก และซับซ้อนครอบคลุมได้มากเท่าไร จะทำให้ระบบมัลติมีเดียนั้นสมบูรณ์มากขึ้นเท่านั้น

3.2 การตอบโต้กันได้ในลักษณะการติดต่อสื่อสาร ใช้สมรรถนะของการคมนาคมที่ทันสมัยเข้าช่วย ผลลัพธ์ที่ได้จะปรับเปลี่ยนตามเหตุการณ์ที่เปลี่ยนไป เช่น เดิมต้องการทราบข้อมูลเพียงว่ามีโรงแรมอะไรบ้างเท่านั้น แต่ในปัจจุบันเมื่อเราเลือกโรงแรมใดแล้ว เราจะทราบถึงขนาดโรงแรมว่าโรงแรมนั้นมีห้องพักกี่ห้องและมีห้องว่างด้วยหรือไม่ เป็นต้น

ระบบมัลติมีเดียมีอะไรบ้าง

องค์ประกอบที่สำคัญในระบบมัลติมีเดียได้แก่

1. **ฮาร์ดแวร์** องค์ประกอบที่สำคัญของระบบมัลติมีเดียคือภาพ เสียง และการตอบโต้ซึ่งกันและกันได้ ส่วนที่เป็นฮาร์ดแวร์จึงต้องจำแนกออกไปตามองค์ประกอบดังกล่าวโดยมีคอมพิวเตอร์เป็นส่วนสำคัญ คอมพิวเตอร์ที่ใช้งานในระบบมัลติมีเดียต้องแยกจากกัน ระหว่างการสร้างงานจำเป็นต้องใช้คอมพิวเตอร์ที่มีสมรรถนะสูง ถ้าเป็นระดับเวิร์กสเตชัน (Workstation)จะดีมาก ซึ่งจะเป็นตัวเชื่อมโยงไปหาฮาร์ดแวร์ตัวอื่น ๆ

1.1 ฮาร์ดแวร์สำหรับงานด้านภาพ อุปกรณ์ที่สำคัญและจำเป็นนอกจากคอมพิวเตอร์ ได้แก่เครื่องอ่านภาพ (Scanner) ใช้สำหรับงานสำเนาภาพจากต้นฉบับที่เป็นภาพนิ่ง หลายแบบ หลายระดับ มีลักษณะเป็นภาพสองมิติ, กล้องถ่ายภาพเชิงตัวเลข (Digital Camera) เป็นกล้องถ่ายภาพธรรมดาที่มีการเปลี่ยนส่วนรับภาพที่เป็นฟิลม์จะเป็นตัวเปลี่ยนสัญญาณภาพมาเป็นสัญญาณเชิงตัวเลข หรือเรียกว่าซีซีดี (Charge Couple Device : CCD) ใช้ทำสำเนาภาพนิ่งได้ทั้งสองมิติ และสามมิติ, กล้องถ่ายภาพวิดีทัศน์ (Video Camera) ใช้งานถ่ายภาพเคลื่อนไหว ใช้สัญญาณเชิงตัวเลขและใช้ซีซีดีเช่นเดียวกับกล้องถ่ายภาพเชิงตัวเลข, การ์ดแปลงสัญญาณอนาลอกเป็นสัญญาณเชิงตัวเลข (Digital Card) เนื่องจากเครื่องเล่นวิดีโอและกล้อง

ถ่ายภาพวิดีทัศน์ส่วนใหญ่ให้สัญญาณแบบอนาลอก การจะนำสัญญาณจากอุปกรณ์ดังกล่าวไปใช้งานกับคอมพิวเตอร์ จึงจำเป็นต้องแปลงสัญญาณจากอนาลอกให้เป็นสัญญาณเชิงตัวเลข

1.2 ฮาร์ดแวร์สำหรับงานทางด้านเสียง ในส่วนระบบเสียงนั้นเดิมเป็นแบบอนาลอก แต่ในปัจจุบันได้มีการประยุกต์ และพัฒนาระบบเสียงให้ใช้สัญญาณเชิงตัวเลขได้ ฮาร์ดแวร์ที่ใช้สำหรับงานทางด้านเสียงจึงแทบจะประยุกต์ใช้กับระบบมัลติมีเดียได้ทันทีทั้งโดยตรงและโดยอ้อม อุปกรณ์มีดังนี้ คือเครื่องเล่นซีดี (CD - ROM) ถ้าไม่ต้องการบันทึกเสียงลงในคอมพิวเตอร์ก็สามารถจะแยกใช้ระบบเสียงจากภายนอกได้ โดยอาศัยการควบคุมการทำงานวิธีที่สะดวกที่สุดจะเป็นการใช้เครื่องเล่นซีดี-รอมที่ปัจจุบันสามารถเล่นได้ทั้งระบบเสียง และระบบอ่านข้อมูลคอมพิวเตอร์ เพียงแต่ต้องบันทึกลงในแผ่นซีดีมาก่อน จากนั้นจึงมาเปิดใช้งานโดยอาศัยคำสั่งจากตัวเครื่องคอมพิวเตอร์เป็นตัวควบคุมอีกทีหนึ่ง, เสียงสำเร็จรูป (Clip Sound) เป็นระบบเสียงสำเร็จรูปในลักษณะไฟล์สัญญาณเชิงตัวเลขที่สามารถใช้งานได้ทันที ส่วนใหญ่จะเป็นเสียงเพลงหรือเอฟเฟ็กต์ต่าง ๆ

2. ซอฟต์แวร์มัลติมีเดีย แยกออกเป็นสองตอนคือ ตอนสร้างกับตอนใช้งาน

2.1 ซอฟต์แวร์สร้างงานระบบมัลติมีเดีย แยกออกเป็นสองส่วน คือซอฟต์แวร์สำหรับสร้างภาพและเสียง กับซอฟต์แวร์จัดระบบ

2.1.1 ซอฟต์แวร์สำหรับสร้างภาพและเสียง มักมีการใช้งานและเป็นที่คุ้นเคยของผู้ใช้คอมพิวเตอร์อยู่แล้ว มีการตกแต่ง ดัดแปลง สร้างใหม่ ซึ่งแยกกันทำเป็นขั้นตอนได้

2.2.2 ซอฟต์แวร์จัดระบบมัลติมีเดีย เป็นซอฟต์แวร์ที่ควบคุมงานส่วนต่าง ๆ มาจัดลำดับเพื่อให้มีการโต้ตอบกันได้ เช่น บอกให้รู้ว่าถ้ากดตรงนี้จะมีภาพต่อไปจะมีเสียง หรือถ้ากดตรงนี้จะได้ภาพนั้นภาพนี้ออกมา หรือมีเสียงนี้เสียงนั้นออกมา เป็นต้น ซอฟต์แวร์ประเภทนี้ยังไม่เป็นที่คุ้นเคยมากนัก ทำให้ยังมีน้อย ราคาสูง และที่สำคัญยังต้องใช้กับคอมพิวเตอร์สมรรถนะสูงอีกด้วย

2.2 ซอฟต์แวร์ใช้งานระบบมัลติมีเดีย ระบบมัลติมีเดียนั้นเกิดจากการสร้างขึ้นมาโดยผ่านทางระบบคอมพิวเตอร์ ทำให้ไม่เป็นการยากนักหากจะใช้งาน แต่จะมีข้อจำกัดอยู่ที่ความง่ายในการใช้งาน

<u>เทคโนโลยีที่เกี่ยวกับมัลติมีเดีย</u>

เนื่องจากระบบคอมพิวเตอร์มัลติมีเดียเป็นการรวบรวมเทคโนโลยีหลายอย่างเข้าด้วยกัน เพื่อให้เกิดความสมบูรณ์ในการทำงาน เทคโนโลยีเหล่านั้นได้แก่

1. การพัฒนาเทคโนโลยีในการบันทึกข้อมูลการทำงานของมัลติมีเดียประกอบไปด้วยภาพและเสียง

2. การพัฒนาด้านระบบคอมพิวเตอร์เครือข่าย สิ่งที่ระบบคอมพิวเตอร์มัลติมีเดียเข้าไปมีบทบาทร่วมกับระบบคอมพิวเตอร์เครือข่าย เช่น การติดต่อสื่อสารด้วยระบบอิเล็กทรอนิกส์เมล์ (Electronic Mail) ซึ่งเดิมเป็นการติดต่อที่เป็นลักษณะฐานข้อมูล (Text Base) เท่านั้น เป็นการนำสองเทคโนโลยีมาช่วยกัน ทำให้การติดต่อสื่อสารในระบบเครือข่ายคอมพิวเตอร์ทำได้ทั้งที่เป็นภาพและเสียง

3. การพัฒนาเทคนิคการย่อขนาดข้อมูล การย่อข้อมูลที่มีประสิทธิภาพจะเป็นปัจจัยสำคัญอย่างหนึ่งในการทำงานของระบบคอมพิวเตอร์มัลติมีเดีย

4. การพัฒนาไมโครคอมพิวเตอร์ การทำงานของคอมพิวเตอร์มัลติมีเดียเป็นการทำงานที่เกี่ยวข้องกับข้อมูลในปริมาณมหาศาล กระบวนการย่อและขยายขนาดข้อมูลจะต้องเกิดอย่างรวดเร็ว และมากพอที่จะทำให้การติดต่อส่งข้อมูลระหว่างหน่วยความจำและอุปกรณ์ต่าง ๆ ไม่เกิดการหยุดชะงัก เพราะถ้าเกิดเหตุการณ์เช่นนี้จะทำให้การแสดงผลทั้งภาพและเสียงอาจเพี้ยนไปจากของจริงได้

5. การพัฒนาของจอภาพ

6. การพัฒนาอุปกรณ์ป้อนข้อมูล

7.การพัฒนาซอฟต์แวร์ส่วนหนึ่งที่ทำให้โลกของคอมพิวเตอร์มัลติมีเดียเป็นจริง ก็คือการพัฒนาซอฟต์แวร์ให้มีประสิทธิภาพสูง และมีการใช้งานได้ง่ายขึ้น

บทที่ 2
ทฤษฎีของตัวอักษร ภาพ และเสียง

2.1 ตัวอักษร (Text)

คอมพิวเตอร์จะแสดงผลตัวอักษรในเท็กซ์โหมด โดยการนำตัวอักษร ตัวเลข และเครื่องหมายต่าง ๆ ที่มีอยู่ในหน่วยความจำของคอมพิวเตอร์มาแสดงผลบนจอภาพตามคำสั่ง แบ่งหน้าจอเป็นลักษณะแถว และคอลัมน์ เช่น 25 * 80 การบอกตำแหน่งจะทำการบอกเป็นค่าของ x , y แต่เนื่องจากตัวอักษร ตัวเลข และเครื่องหมายที่มีอยู่นั้นได้ถูกกำหนดอยู่ในรูปร่างที่แน่นอนแล้ว และมีจำนวนจำกัด เพราะฉะนั้นจึงไม่สามารถนำมาประกอบกันให้เกิดเป็นภาพต่าง ๆ ได้

รูปแบบการจัดเก็บตัวอักษรเก็บได้ใน 2 รูปแบบ

1. ไฟล์ที่ไม่มีรูปแบบ (Unformated Text) ขนาดของคาแรกเตอร์จะถูกกำหนดแน่นอน และจะมีรูปแบบในการนำเสนอเพียงแบบฟอร์ม หรือสไตล์เดียว

2. ไฟล์ที่มีรูปแบบ (Format Text) คาแรกเตอร์จะมีหลายรูปแบบ หลายขนาด

2.2 ภาพ (Image)

ภาพเป็นข้อมูลที่มีโครงสร้างเป็นแบบพิกเซล (Pixel) ซึ่งเป็นหน่วยที่เล็กที่สุด (กรณีมีการจัดเก็บแบบบิตแมพ) หรือมีโครงสร้างเป็นแบบออบเจค (กรณีที่มีการจัดเก็บแบบเวกเตอร์) โดยที่สามารถจะแสดงให้เห็นที่จอภาพ และที่อุปกรณ์กราฟฟิกอื่น ๆ ของระบบคอมพิวเตอร์ โดยจะประกอบกันเป็นจุด เส้นแบบลาย และสีของภาพ การแบ่งหน้าจออันเป็นความสามารถในการแสดงผลนั้นจะขึ้นอยู่กับคุณสมบัติของไดรเวอร์ (Driver), โหมด (Mode), ความละเอียด (Resolution), สี และเพจ (page) ของระบบกราฟฟิกอื่น ๆ ที่เลือกใช้ระบบกราฟฟิกนั้นมีการแสดงสีได้ตั้งแต่ 2 สี จนถึงล้านสี ในส่วนของกราฟฟิกโหมดนี้จะมีรูปแบบฟอร์แมตในการจัดเก็บข้อมูลอยู่ด้วยกันหลายชนิด แต่ที่เราได้นำมาใช้ในปริญญานิพนธ์นี้คือ พีซีเอ็กซ์เพนท์บรัช (PCX Paintbrush) โดยจะได้อธิบายในส่วนต่อไปของบทนี้

2.3 PCX (PC Paintbrush)

PCX (PC Paintbrush) เป็นฟอร์เมตของกราฟฟิกบิตแมพที่ออกแบบโดยบริษัท Zsoft เดิมแล้วฟอร์แมตนี้จะใช้อยู่บนเอ็มเอสดอส แต่ต่อมาได้นำมาใช้ในวินโดวส์ด้วย

โครงสร้างของ PCX ไฟล์

ไฟล์ PCX เริ่มต้นด้วยข้อมูล 128 ไบต์ ซึ่งจะทำการเก็บข้อมูลต่าง ๆ เพื่อนำไปใช้ในการเรียกคืน (Restore) ภาพดังรูปที่ 2.1 ข้อมูลคัลเลอร์แมพ (Color Map) ถูกใช้ในการแทนค่าเป็นแพลตเลต (Palette Register) 1 ตัว ของระบบแสดงผลอีจีเอจะบรรจุข้อมูล 6 บิตเท่านั้นโดยแต่ละ 2 บิตจะแทนสีหลักคือ แดง, น้ำเงิน, เขียว ดังรูปที่ 2.2 ตัวอักษรตัวใหญ่จะแทนความเข้มของสี 75 % ตัวอักษรเล็กแทน 25 % แต่ละสี

หลักจะมี 4 ระดับของการแสดงสีคือ 0%, 25%, 75%, 100% (ค่าบิตของตัวอักษรตัวเล็กและตัวอักษรตัวใหญ่เป็น ' 1 ' หมด) ดังรูปที่ 3.3 เช่น

- ค่ารีจีสเตอร์เป็น ' 00000000 ' หมายถึง ไม่แสดงสีใดเลย
- ค่ารีจีสเตอร์เป็น ' 00100101 ' หมายถึง แสดงสีแดงที่มีความเข้ม 100% และแสดงสีน้ำเงินที่มีความเข้ม 75%
- ค่ารีจีสเตอร์เป็น ' 00010001 ' หมายถึง แสดงสีเขียวที่มีความเข้ม 75% และแสดงสีน้ำเงินที่มีความเข้ม 25%

ข้อมูลคัลเลอร์แมพในตอนต้นไฟล์ (Header File)ประกอบด้วยข้อมูล 16 ชุด ชุดละ 3 ไบต์ ไบต์ที่ 1 ของแต่ละชุดจำนวน 3 ไบต์ย่อยนั้นเป็นค่าสำหรับแทนสีแดงโดยถูกนำมาสร้างเป็นเลข 4 จำนวนตั้งแต่ 0 - 3 (00, 01, 10, 11) และตัวเลขนี้จะต้องคูณกับ 85 เพื่อจะทำการเก็บลงไฟล์ในส่วนเฮดเดอร์ไฟล์ ในกรณีเดียวกันสีเขียวคือไบต์ที่ 2 และสีน้ำเงินคือไบต์ที่ 3 วิธีการนี้จะถูกทำซ้ำ ๆ กันจนครบ16 รอบเพื่อจะได้ค่าของสีแดง, สีน้ำเงิน และสีเขียวจนครบ 16 ชุด

ไบต์ที่	ขนาด (ไบต์)	ชื่อข้อมูล	รายละเอียด
0	1	Password	ค่า 0aH = ไฟล์.PCX
1	1	Version	เก็บค่าเวอร์ชัน(รุ่น)ของ PC Paintbrush ดังนี้ 0 = เวอร์ชัน 2.5 2 = เวอร์ชัน 2.8 with Palatte 3 = เวอร์ชัน 2.8 without Palatte 5 = เวอร์ชัน 3.0
2	1	Encoding	ค่า = 1
3	1	Bits per Pixel	จำนวนของบิตที่ใช้แสดง 1 พิกเซลจาก 1 เพลน 1 = EGA,VGA, or HERC 4 = CGA
4	8	Windows Dimensions	ค่า Integer 4 ค่า(ค่าละ 2 ไบต์) ให้ค่ามุมบนซ้าย ขวาของภาพ กำหนดรูปแบบ x1,y1,x2,y2
12	2	Horizontal Resolution	ความละเอียดของการแสดงภาพในแนวนอน 640 = EGA,VGA 320 = CGA 720 = HERCULES

ไบต์ที่	ขนาด (ไบต์)	ชื่อข้อมูล	รายละเอียด
14	2	Vertical Resolution	ความละเอียดของการแสดงภาพในแนวตั้ง 480 = VGA 350 = EGA 200 = CGA 348 = HERC
16	48	Color Map	ข้อมูล Color Pallete
64	1	Reserved	
65	1	No. of Plane	จำนวนของเพลนที่ใช้แสดงภาพ
66	1	Byte per Line	จำนวนไบต์ต่อการสแกน 1 บรรทัด
68	2	Palette Info	How to interpret the palette
70	8	Maximum x value	Special for fractal files
78	8	Minimum x value	Special for fractal files
86	8	Maximum y value	Special for fractal files
94	8	Minimum y value	Special for fractal files
102	8	P Value	Special for fractal files
110	8	Q Value	Special for fractal files
118	10	Not used	

รูปที่ 2.1 ตารางโครงสร้าง PCX เฮดเดอร์

ไบต์ที่	Palette	ไบต์ที่	Palette
16,17,18	0	40,41,42	8
19,20,21	1	43,44,45	9
22,23,24	2	46,47,48	10
25,26,27	3	49,50,51	11
28,29,30	4	52,53,54	12
31,32,33	5	55,56,57	13
34,35,36	6	58,59,60	14
37,38,39	7	61,62,63	15

รูปที่ 2.2 แสดงตำแหน่งในการเก็บสี จำนวน 3 ไบต์ คือสีแดง, สีเขียว และสีน้ำเงิน

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
-	-	R	G	B	r	g	b
x	x	75%	75%	75%	25%	25%	25%

รูปที่ 2.3 ระดับของการแสดงสี

ถ้าแพลตเลตรีจีสเตอร์มีค่า XX100001 แล้ว (ในแพลตเลตใด ๆ (0 - 15))

บิตที่ 5, 2 เป็นของสีแดง

บิตที่ 4, 1 เป็นของสีเขียว

บิตที่ 3, 0 เป็นของสีน้ำเงิน

แดง คือ ' 10 ' = 2 , เขียว คือ ' 00 ' = 0, น้ำเงิน คือ ' 01 ' = 1

ดังนั้นค่าที่จะเก็บในคัลเลอร์แมพของแพลตเลตใด ๆ (0 - 15) คือ (* 85)

สีแดง	สีเขียว	สีน้ำเงิน
170	0	85

หรือค่าแพลตเลตรีจีสเตสอร์มีค่า 'xx111111'แล้ว ค่าที่จะเก็บคือ

สีแดง	สีเขียว	สีน้ำเงิน
255	255	255

<u>ข้อแตกต่างในจอวีจีเอ</u>

ในวีจีเอจะมีสถานะสี 256 สีที่แตกต่างกัน รูปแบบการใช้งานเหมือนกับการแสดง 16 สี ข้อมูลในแพลตเลตจะยาวกว่า 16 สี ข้อมูลนี้จะอยู่ที่ส่วนท้ายของไฟล์.PCX การเข้าถึงข้อมูลเป็นดังนี้ต้องตรวจไบต์ตรงตำแหน่งเวอร์ชันของไฟล์ว่าเป็น 5 (เวอร์ชัน 3.0) หรือไม่ และข้อมูลแพลตเลตจะต้องนับถอยหลังจากไฟล์ไป 769 ไบต์ ในไบต์นั้นจะต้องเป็น 0CH (12 Decimal) จากนั้นข้อมูลที่ท้ายไฟล์จะเป็นข้อมูลแพลตเลต 256 สี

จอภาพวีจีเอจะมีการควบคุมสีที่แตกต่างกับอีจีเอ คือในแต่ละแพลตเลตรีจีเตอร์จะเก็บลำดับที่ของคัลเลอร์รีจีสเตอร์ไว้ 1 ตัว และตัวของคัลเลอร์รีจีสเตอร์จะใช้เนื้อที่จำนวน 6 บิต ดังนั้นค่าสีที่เป็นไปได้จะเป็น 64 ค่า (หมายถึง Shade ของสีนั้นเอง) ของแต่ละสี ในวีจีเอสามารถอ่านค่า (ค่า 6 บิตของสีแดง, สีน้ำเงิน, สีเขียว)ข้อมูลในแพลตเลตรีจีสเตอร์ในคัลเลอร์รีจีสเตอร์เพื่อทำการอ่านคัลเลอร์รีจีสเตอร์อีกที ทำการคูณสีแดง , สีน้ำเงิน และสีเขียว ด้วย 4 และเก็บผลลงในกลุ่ม 3 ไบต์ของแพลตเลตต์นั้น ๆ

การเข้ารหัส

ภาพที่จะอ่านจะถูกอ่านตามแนวของจอภาพในแนวนอนจากซ้ายไปขวา เริ่มที่พิกเซลที่ตำแหน่งบนซ้ายไปทางขวาจนสุดภาพแล้วจึงอ่านในแถวถัดไปโดยในวีจีเอและอีจีเอซึ่งมีหน่วยความจำหลาย ๆ เพลน ดังนั้นในหนึ่งแถว (Scanline) จะอ่านข้อมูลในทุกเพลน เริ่มที่เพลน 0, 1, 2, 3 ตามลำดับจนครบแล้วจึงอ่านแถวถัดไปจนหมดภาพที่ต้องการเก็บ

มีการเข้ารหัสแบบ Run - Length Encoding (RLE) ซึ่งเป็นวิธีการลดขนาดไฟล์แบบไม่มีการสูญเสีย โดยเปรียบเทียบกับไบต์ข้างเคียงใน 1 แถว ถ้าเหมือนกันก็จะให้ค่า a flag (C0h) และ a count (เช่น C0h : count) ไบต์ต่อมาจะบอกข้อมูลที่ซ้ำ เช่นมี 5 ไบต์ที่มีค่าเหมือนกัน จะได้ 01 01 01 01 01 = C5 01 = flag c with a count 5 และ 01 คือข้อมูลในตำแหน่งถัดไป หรือ 01 01 01 01 01 04 01 01 = C5 01 04 C5 01

หมายเหตุ 1. ถ้าค่าของไบต์ < C0h และไม่เหมือนไบต์อื่น จะเก็บลงไฟล์เลย

2. ถ้าค่าของไบต์ > C0h และไม่เหมือนไบต์อื่น จะเก็บ "C1h" ลงไฟล์ แล้วตามด้วยค่าของไบต์นั้น

3. ถ้าค่าของไบต์เหมือนไบต์เหมือนไบต์อื่น นับ count=1

4. ถ้า count > 63 จะให้ count = 1, เก็บค่า "Ffh" และค่าของไบต์ลงไฟล์

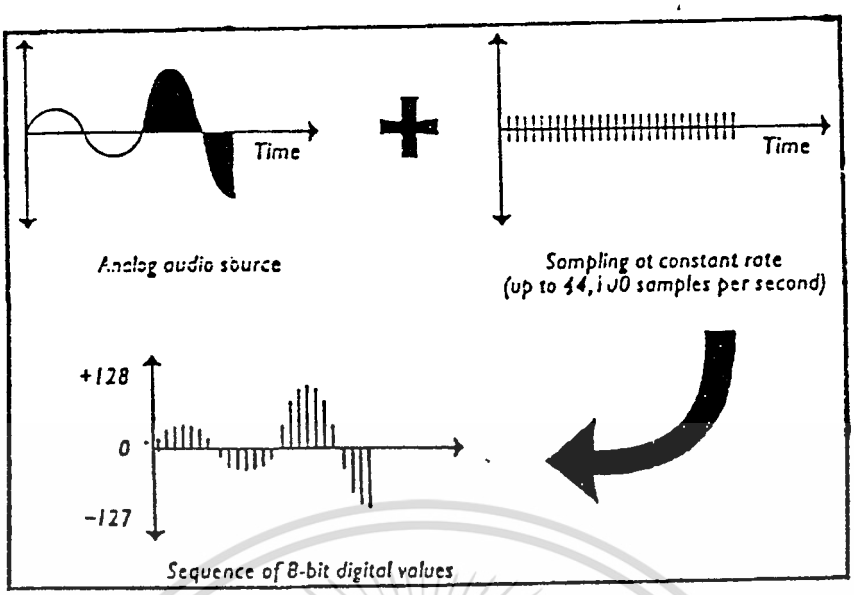
การถอดรหัส

การถอดรหัสใช้วิธีการตรงกันข้ามกับการเข้ารหัส ซึ่งสรุปได้คือ จะทำการอ่านข้อมูลมา 1 ไบต์ แล้วตรวจดูว่ามากกว่า ' C0h ' หรือไม่ (ค่า 2 บิตบนเป็น 11) ถ้าใช่ค่านี้จะเป็นค่าตัวนับทันที (XOR ด้วย C0h จะได้ค่าจริงที่น้อยกว่า 63 ออกมา) ซึ่งไบต์ต่อไปก็จะเป็นค่าของไบต์ (Value) แล้วทำการขยายข้อมูลออกมาตามจำนวนตัวนับนั้น ๆ แล้วเก็บในหน่วยความจำ ถ้าค่าไบต์ที่อ่านน้อยกว่า C0h จะทำการเก็บข้อมูลไบต์นั้นตัวเดียวลงในหน่วยความจำที่มีอยู่ได้เลยโดยไม่ต้องขยายข้อมูล ทำเช่นนี้จนจบไฟล์ก็จะได้ข้อมูลของไฟล์ทั้งหมด จากนั้นนำข้อมูลในหน่วยความจำที่ได้มาแสดงบนจอภาพ โดยใช้ฟังก์ชัน POKEB ( ) ในการนำเอาข้อมูลไปไว้ในหน่วยความจำแสดงผล (Display memory) หรืออาจจะใช้ฟังก์ชัน PUTPIXEL ( ) แสดงภาพออกมาก็ได้

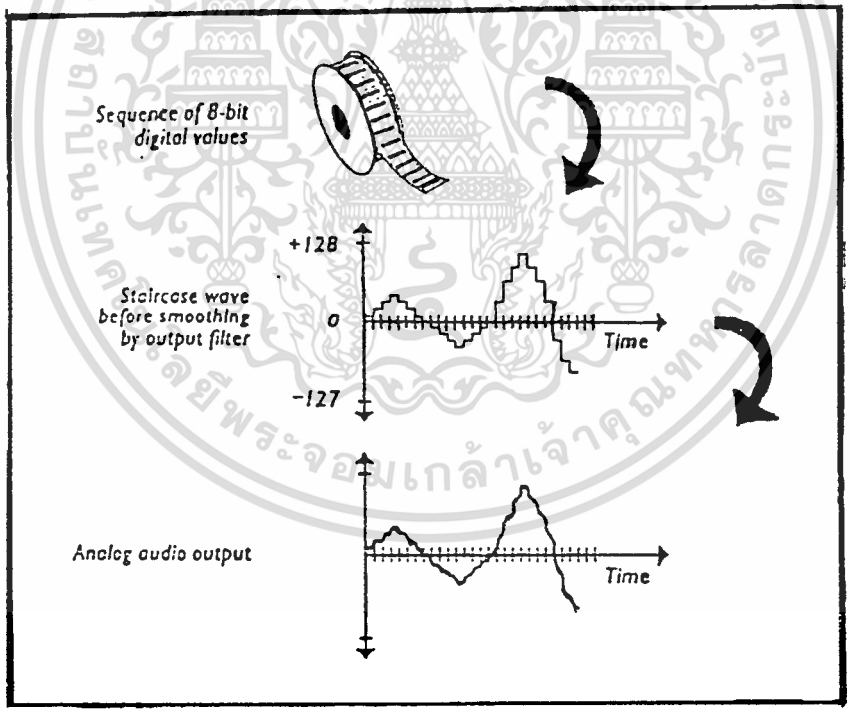
2.3 เสียง (Sound)

หลักการของเสียงและการแปลงสัญญาณเสียง

เสียงในที่นี้หมายถึงเสียงคำพูดเท่านั้น และเราจะทำการพิจารณาในสองขั้นตอนคือ ขั้นตอนการแปลงเสียงไปเป็นสัญญาณดิจิตอล และการแปลงกลับ โดยทั่วไปในการแปลงสัญญาณอนาลอกให้เป็นสัญญาณดิจิตอล สามารถทำได้โดยกระบวนการที่เรียกว่าแซมปลิ้ง (Sampling) ในซาวน์บลาสเตอร์ (Sound Blaster) ก็เช่นเดียวกัน กระบวนการแซมปลิ้งจะถูกทำโดยตัวแปลงสัญญาณจากอนาลอกเป็นสัญญาณดิจิตอล (Analog to Digital Converter : ADC) ดังรูปที่ 2.4



รูปที่ 2.4 แสดงถึงตัวแปลงสัญญาณอนาลอกเป็นสัญญาณดิจิตอล



รูปที่ 2.5 แสดงถึงตัวแปลงสัญญาณดิจิตอลเป็นสัญญาณอนาลอก

ตัวแปลงสัญญาณอนาลอกเป็นสัญญาณดิจิตอลนั้น จะทำการสุ่มสัญญาณเสียงด้วยอัตราสุ่ม (Sampling rate) ที่กำหนดไว้ล่วงหน้า และแปลงผลลัพธ์ที่ได้จากการสุ่มให้เป็นค่าตัวเลข (Digit) อัตราการสุ่มอาจเรียกได้อีกอย่างว่าความถี่ในการสุ่ม ซึ่งมีหน่วยเป็นเฮิร์ต (Hertz) ความถี่ที่เหมาะสมในการสุ่ม

สัญญาณตามทฤษฎีแซนนอน (Shannon Theorem) กำหนดไว้ว่าต้องมีค่าอย่างน้อยเท่ากับสองเท่าของความถี่ของสัญญาณที่ถูกสุ่ม ในกรณีของเสียงพูด สัญญาณจะมีความถี่สูงสุดประมาณ 4000 เฮิร์ต เพราะฉะนั้นความถี่ในการสุ่มต้องมีค่าอย่างน้อย 8000 เฮิร์ต

นอกจากความถี่ในการสุ่มแล้ว ยังมีพารามิเตอร์ (Parameter) อีกตัวหนึ่งที่เกี่ยวข้องกับการสุ่มนั่นคือขนาดในการสุ่ม (Sampling Size) ขนาดในการสุ่มจะเป็นตัวกำหนดช่วงกว้าง (Range) ของสัญญาณ ในซาวน์บลาสเตอร์ทั่วไปนั้นจะมีขนาด 8 บิต แต่ในรุ่นใหม่ เช่น ซาวน์บลาสเตอร์16 จะมีขนาด 16 บิต การสุ่มหนึ่งครั้งจะถูกแปลงเป็นเลขฐานสองมีขนาดตามขนาดในการสุ่ม

ในขั้นตอนการแปลงสัญญาณกลับจากสัญญาณดิจิตอลเป็นสัญญาณเสียง จะถูกทำโดยตัวแปลงสัญญาณจากสัญญาณดิจิตอลเป็นสัญญาณอนาลอก (Digital to Analog Converter : DAC) ดังรูปที่ 2.5

คุณภาพของเสียงที่ได้จากการสุ่ม เมื่อนำมาแปลงกลับจะขึ้นอยู่กับความถี่ ในการสุ่มนั้นยิ่งความถี่ในการสุ่มมากคุณภาพของสัญญาณเสียงก็จะยิ่งดีมากขึ้น แต่ปริมาณของข้อมูลก็จะต้องมากขึ้นด้วย ดังนั้นการเลือกความถี่ในการสุ่มจึงพิจารณาได้จากความต้องการในคุณภาพของงาน และเนื้อที่ที่จะใช้ในการเก็บข้อมูล

<u>คุณสมบัติของซาวน์บลาสเตอร์ และการนำมาใช้งาน</u>

การ์ดซาวน์บลาสเตอร์ (Sound Blaster Card) เป็นอุปกรณ์ที่ช่วยเพิ่มประสิทธิภาพการทำงานเกี่ยวกับเสียงในคอมพิวเตอร์ส่วนบุคคลหรือพีซี (Personal Computer : PC) การพัฒนาทางเทคโนโลยีในยุคปัจจุบันทำให้คอมพิวเตอร์ส่วนบุคคลมีความสามารถในการทำงานด้านกราฟฟิกและเสียงมากขึ้น ความสามารถที่เพิ่มขึ้นเหล่านี้กำลังจะกลายเป็นมาตรฐานการทำงานของเครื่องคอมพิวเตอร์ส่วนบุคคลทั่วไป รู้จักกันในนามของศัพท์ที่เรียกว่ามัลติมีเดีย รายละเอียดของคุณสมบัติของซาวน์บลาสเตอร์และการนำมาใช้งานจะมีดังนี้คือ

1. สร้างและเลียนแบบเสียงดนตรีหรือเสียงธรรมชาติอื่น ๆ ด้วยทฤษฎีการสังเคราะห์เอฟเอ็ม (FM Synthesis)

2. บันทึก (Record) สัญญาณเสียงด้วยทฤษฎีการแซมปลิ้ง และเก็บข้อมูลอยู่ในรูปของสัญญาณดิจิตอล รวมทั้งเล่นกลับ (Play Back) ข้อมูลดิจิตอลให้อยู่ในรูปสัญญาณเสียงดั้งเดิม

3. มีการนำเทคโนโลยีด้านการบีบอัดข้อมูล (Data Compression) มาใช้

นอกจากนี้ ซาวน์บลาสเตอร์ยังมีความสามารถในการติดต่อกับหน่วยความจำของคอมพิวเตอร์ได้โดยตรง โดยไม่ต้องผ่านซีพียู (CPU) โดยใช้ดีเอ็มเอ (Direct Memory Access : DMA) ทำให้ประหยัดเวลาในการประมวลผลสัญญาณ

ในซาวน์บลาสเตอร์แต่ละตัวจะมีชิป (Chip) ประมวลผลสัญญาณเป็นของตัวเอง ทำให้การประมวลเสียงสามารถทำได้พร้อม ๆ กับที่เครื่องคอมพิวเตอร์กำลังทำงานอื่นโดยใช้ซีพียู

เอาท์พุทของซาวนด์บลาสเตอร์คือลำโพง ส่วนอินพุทของซาวนด์บลาสเตอร์มีได้หลายอย่าง แต่สิ่งที่เป็นพื้นฐานก็คือไมโครโฟน นอกจากนี้อาจเป็นซ่องซีดี (CD) หรือลายน์อิน (Line - in)

รูปแบบของไฟล์เสียงที่ใช้ในซาวนด์บลาสเตอร์มีหลายรูปแบบเช่นครีเอทีฟวอยซ์ไฟล์ฟอร์แมต (Creative Voice File format : .VOC), ไมโครซอฟท์เวฟฟอร์มออดิโอไฟล์ฟอร์แมต (Microsoft Waveform Audio File format : .WAV), ครีเอทีฟมิวสิคไฟล์ฟอร์แมต (Creative Music File format : .CMF)

<u>ในส่วนของรูปแบบไฟล์.VOC</u>

การเขียนโปรแกรมสั่งงานซาวนด์บลาสเตอร์จะเรียกใช้ฟังก์ชันของไดรเวอร์ที่ชื่อ CT - VOICE . DRV เพื่อให้ซาวนด์บลาสเตอร์ทำหน้าที่ตามต้องการ ฟังก์ชันที่สำคัญของไดรเวอร์ที่นำมาใสช้งานก็เช่น ฟังก์ชันที่เชคสถานะการทำงานเริ่มต้นของซาวนด์การ์ด ฟังก์ชันบันทึกสัญญาณเสียง และฟังก์ชันเล่นย้อนกลับของข้อมูลสัญญาณเสียง เป็นต้น ข้อมูลที่ได้จากการทำงานของฟังก์ชันของไดรเวอร์จะอยู่ในรูปแบบของ VOC format

<u>โครงสร้างของซีทีวอยซ์ไฟล์ฟอร์แมต(Creative Voice File format : CT- Voice format)</u>

ซีทีวอยซ์ไฟล์ฟอร์แมต เป็นรูปแบบของไฟล์ข้อมูลดิจิตอลที่ได้จากการสุ่มสัญญาณเสียงอนาลอกและถูกแปลงให้เป็นสัญญาณตัวเลขแล้ว และรูปแบบของไฟล์ชนิดนี้กำหนดโดยบริษัท Creative Labs โดยไฟล์ชนิดนี้จะถูกระบุสกุลว่า .VOC ไฟล์วีโอซีจะแบ่งออกเป็น 2 บล็อค คือส่วนหัว (Header) และส่วนข้อมูล (Actual Data)

1. ซีทีวอยซ์เฮดเดอร์บล็อค (CT - Voice Header Block) เป็นบล็อคที่ใช้บ่งลักษณะของไฟล์ว่าเป็นไฟล์ซีทีฟอร์แมต แยกเป็นส่วนย่อย ๆ ดังนี้ คือ

- ไบต์ $00-$13 (0 - 19) บรรจุข้อความ "Creative Voice File"
- ไบต์ $14-$15 (20 - 21) บรรจุออฟเซตแอดเดรส (Offset Address) ของข้อมูล
- ไบต์ $16-$17 (22 - 23) บรรจุหมายเลขเวอร์ชันของซีทีวอยซ์ฟอร์แมต
- ไบต์ $18-$18 (24 - 25) บรรจุค่าคอมพลีเมนต์ของหมายเลขเวอร์ชันซึ่งบวกกับค่า $1234 เป็นไบต์สูงและไบต์ต่ำ

2. ดาต้าบล็อค (Data Block) ประกอบด้วยบล็อคย่อย ๆ 8 บล็อค แต่ละบล็อคมีโครงสร้างคล้าย ๆ กัน

โดยที่ Block 0 - End Block

เป็นบล็อคที่ใช้บ่งว่าหมดข้อมูลแล้ว จะอยู่ที่ท้ายของวีโอซีไฟล์เท่านั้น

block 0

Structure of the End Block	
Block Type	1 byte = 0
Block Length	none
Data Bytes	none

Block 1 - New Voice Block

บรรจุข้อมูลแท้ ๆ ที่สามารถนำมาเล่นย้อนกลับ (Play Back) ได้ SR คืออัตราการแซมปลิ้งที่ใช้ในการบันทึกเสียง ส่วนแพคไบต์ (Pack Byte) เป็นไบต์ที่ใช้ระบุว่าข้อมูลที่บันทึกมีการแพค (Pack) หรือไม่ ถ้ามีก็บอกให้ทราบว่าเป็นการแพคชนิดไหน

block 1

Structure of the New Voice Block	
Block Type	1 byte = 1
Block Length	3 bytes
SR Byte	1 byte
Pack Byte	1 byte = 0,1,2,3
Data Byte	x bytes

Block 2 - Sequent Voice Block

เป็นบล็อคที่ใช้ในกรณีที่มีข้อมูลขนาดใหญ่เกินกว่าที่จะโหลด (Load) ลงหน่วยความจำภายในครั้งเดียว บล็อคนี้จะทำให้สามารถแบ่งข้อมูลออกเป็นบล็อคเล็ก ๆ หลายบล็อคได้

block 2

Structure of the Subsequent Voice Block	
Block Type	1 byte = 2
Block Length	3 bytes
Data Byte	x bytes

Block 3 - Silence Block

เป็นบล็อคที่ใช้ในการลดขนาดของข้อมูล โดยจะแทนที่ข้อมูลที่ไม่มีเสียง(เงียบ)ด้วยบล็อคนี้

block 3

Structure of the Subsequent Voice Block	
Block Type	1 byte = 2
Block Length	3 bytes
Duration	2 bytes

Block 4 - Marker Block

ใช้ในการมาร์ค (Mark) ตำแหน่งของข้อมูลที่ต้องการใช้งานแบบเล่นย้อนกลับ

block 4

Structure of the Marker Voice Block	
Block Type	1 byte = 4
Block Length	3 bytes = 2
Marker	2 bytes

Block 5 - Message Block

ใช้ในการแทรกตัวอักษรแอสกีลงไปในวีโอซีไฟล์

block 5

Structure of the Message Voice Block	
Block Type	1 byte = 5
Block Length	3 bytes
Ascii Data	x bytes
End Character	1 byte

Block 6 - Repeat Block

เป็นบล็อคที่ใช้กำหนดให้นำข้อมูลที่เริ่มตั้งแต่จุดนั้นไปเล่นกลับซ้ำตามจำนวนที่กำหนด

block 6

Structure of the Repeat Block	
Block Type	1 byte = 6
Block Length	3 bytes = 2
Count	2 bytes

Block 7 - Repeat End Block

เป็นบล็อคที่ใช้กำหนดจุดสิ้นสุดของข้อมูลที่จะให้เล่นซ้ำ

block 7

Structure of the End Repeat Block	
Block Type	1 byte = 7
Block Length	3 bytes = 0

Block 8 - Extended Header Block

ใช้ใน SB Pro เพราะว่าในดาต้าบล็อค 1 ได้ให้รายละเอียดของข้อมูลไม่ได้ครบถ้วน ในกรณีที่เป็นข้อมูลเสียงแบบสเตอริโอ

block 8

Structure of the Extended Header Block	
Block Type	1 byte = 8
Block Length	3 bytes = 4
Sample rate	2 bytes
Data Mode	1 byte

ฟังก์ชันในการใช้งานของซีทีไดรเวอร์ (CT - driver : CT - Voice . DRV)

ฟังก์ชัน 0 (BX = 0) : กำหนดเวอร์ชันของไดรเวอร์

function 0

Determine driver version	
Input	BX = 00
Output	AH = Main nymber
	AL = Sub-number
Remarks	none

ฟังก์ชัน 1 (BX = 1) : ตั้งค่าพอร์ตแอดเดรส

function 1

Set Port Address	
Input	BX = 01
	AX = Port Address
Output	none
Remarks	can only be called before
	function 3

ฟังก์ชัน 2 (BX = 2) : ตั้งค่าอินเตอร์รัพต์

function 2

Set Interrupt	
Input	BX = 02
	AX = Interrupt Address
Output	none
Remarks	can only be called before
	function 3

ฟังก์ชัน 3 (BX = 3) : เช็คสถานะต่าง ๆ ของซาวน์การ์ด ให้อยู่ในสภาวะที่ถูกต้อง

function 3

Initialize driver	
Input	BX = 03
Output	AX = 0 Successful
	AX = 1 SB not found
	AX = 2 Port address error
	AX = 3 Interrupt error
Remarks	none

ฟังก์ชัน 4 (BX = 4) : เปิด - ปิดลำโพง

function 4

Loudspeaker on/off	
Input	BX = 04
	AL = 0 off
	AL = 1 on
Output	none
Remarks	none

ฟังก์ชัน 5 (BX = 5) : ตั้งค่า Statusword address

function 5

Set Status Address	
Input	BX = 05
	ES : DI = Sample address
Output	none
Remarks	none

ฟังก์ชัน 6 (BX = 6) : เล่นกลับ (Play Back) ข้อมูล (แปลงข้อมูลดิจิตอลเป็นสัญญาณเสียงดั้งเดิม)

function 6

Sample Playback	
Input	BX = 06
	FS : DI = Sample Address
Output	none
Remarks	Statusword is changed
	according to encountered
	marker block

ฟังก์ชัน 7 (BX = 7) : บันทึกข้อมูล (แปลงสัญญาณเสียงเป็นสัญญาณดิจิตอล และเก็บไว้)

function 7

Record Sample	
Input	BX = 07
	AX = Sampling rate
	DX : CX = Length
	ES : DI = Sample Address
Output	none
Remarks	none

ฟังก์ชัน 8 (BX = 8) : ยกเลิกการทำงานทุกอย่างของซาวน์การ์ด

function 8

Abort Sample	
Input	BX = 08
Output	none
Remarks	Statusword = 0

ฟังก์ชัน 9 (BX = 9) : ยกเลิกการติดตั้งไดรเวอร์เดิม และทำการติดตั้งใหม่

function 9

De-install driver	
Input	BX = 09
Output	none
Remarks	none

ฟังก์ชัน 10 (BX = 10) : หยุดการทำงานของฟังก์ชันเล่นย้อนกลับชั่วขณะ

function 10

Pulse Sample	
Input	BX = 10
Output	AX = 0 Successful
	AX = 1 Not Successful
Remarks	none

ฟังก์ชัน 11 (BX = 11) : ใช้ในการสั่งให้ฟังก์ชันเล่นย้อนกลับทำงานต่อไป หลังจากสั่งให้หยุดด้วยฟังก์ชันที่ 10 แล้ว

function 11

Cotinue Sample	
Input	BX = 11
Output	AX = 0 Successful
	AX = 1 Not Successful
Remarks	none

ฟังก์ชัน 12 (BX = 12) : ทำหน้าที่หยุดการทำงานเล่นย้อนกลับแบบซ้ำ ๆ

function 12

Interrupt Loop	
Input	BX = 12
	AX = 0 at end of loop
	AX = 1 immediatly
Output	AX = 0 Successful
	AX = 1 no loop being executed
Remarks	none

ฟังก์ชัน 13 (BX = 13) : เป็นฟังก์ชันที่เพิ่มขึ้นมาให้ผู้ใช้กำหนดหน้าที่และการใช้งานเอง

function 13

User-defined function	
Input	BX = 13
	DX : AX = Function Address
Output	ES : BX = Address of the
	current data block
Remarks	none

บทที่ 3
การสื่อสารข้อมูลแบบอนุกรม

3.1 การสื่อสารข้อมูลแบบอนุกรม

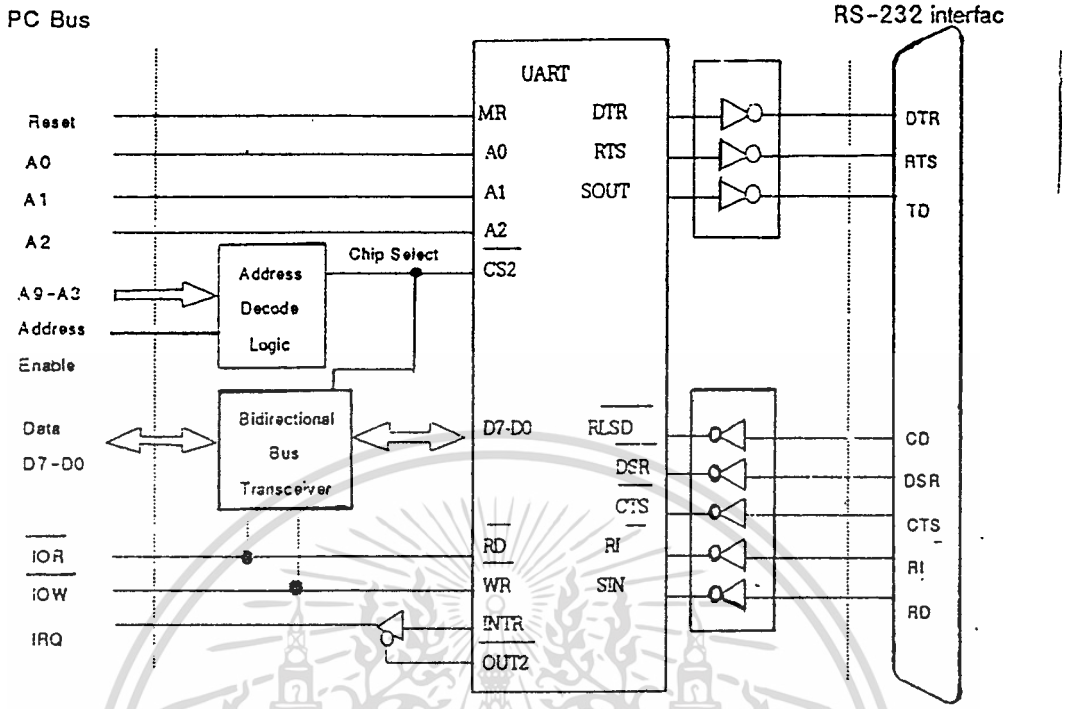
การส่งข้อมูลจะส่งแบบอนุกรม คือข้อมูลจะถูกทำการถ่ายโอนหรือส่งออกมาทีละบิต ระหว่างจุดส่งและจุดรับ ตัวกลางการสื่อสารต้องการเพียงช่องเดียวหรือสายเพียงคู่เดียวในการส่งข้อมูล ส่วนสายสัญญาณที่เหลือจะเป็นสายส่งสัญญาณควบคุม และสายกราวน์ โดยลักษณะของการส่งแบบอนุกรมนั้นจะสามารถส่งได้ในระยะสั้น ๆ จนถึงระยะทางเป็นไมล์ อัตราความเร็วของข้อมูลที่ใช้กันอยู่ทั่วไปจะอยู่ในช่วงตั้งแต่ 0 ถึง 2 ล้านบิตต่อวินาที มีการใช้มาตรฐานของ EIA RS - 232C คือมีระดับสัญญาณไฟฟ้าขนาด 12 โวลต์ หรืออาจจะใช้มาตรฐาน 20 มิลลิแอมป์เคอร์เรนต์ลูป หรืออาจจะใช้ระดับสัญญาณของ TTL ก็ได้ ซึ่งค่าใช้จ่ายก็น้อยกว่าการส่งข้อมูลแบบขนาน รูปแบบของการติดต่อสื่อสารแบบอนุกรมอาจเป็นไปได้ 3 รูปแบบ คือแบบซิมเพล็กซ์ (Simplex) คือส่งข้อมูลได้ในทางเดียวเท่านั้น บางครั้งก็เรียกว่า การส่งทิศทางเดียว (Unidirection Data Bus) แบบที่สองคือการส่งแบบฮาล์ฟดูเพล็กซ์ (Half Duplex) ซึ่งข้อมูลสามารถส่งได้ทั้งสองสถานี แต่จะต้องผลัดกันส่งและผลัดกันรับ จะทำการส่งและรับพร้อมกันไม่ได้ แบบที่สามก็คือแบบฟูลดูเพล็กซ์ (Full Duplex) ทั้งสองสถานีจะสามารถทำการรับและส่งได้ในเวลาเดียวกัน ความเร็วในการถ่ายโอนข้อมูลแบบอนุกรมนี้จะมีหน่วยวัดเป็นบิตต่อวินาที (BPS : Bit Per Second) หน่วยที่แสดงถึงการเปลี่ยนแปลงของสัญญาณใน 1 วินาที เราเรียกว่า บอดเรต (Baud Rate) หรืออัตราบอด โดยที่อัตราบิตจะเท่ากับอัตราบอดคูณกับบิตในหนึ่งบอด การส่งข้อมูลจะส่งแบบอะซิงโครนัส ซึ่งประกอบด้วยบิตเริ่มต้นหรือบิตสตาร์ต (Start Bit) และบิตสิ้นสุด (Stop Bit) ข้อมูลถูกส่งออกไปอย่างไม่มีกำหนดเวลาแน่นอน

3.2 พอร์ตสื่อสารแบบอนุกรม

ในเครื่องคอมพิวเตอร์ทั่วไปจะมีอะแดปเตอร์สื่อสารแบบอะซิงโครนัส(Asynchronous) เป็นการ์ดอะแดปเตอร์แบบอนุกรมและขนานในการ์ดอันเดียวกัน สำหรับอะแดปเตอร์แบบอนุกรมจะแบ่งได้เป็น 3 ประเภทดังนี้

- ประเภทแรกเป็นอะแดปเตอร์ที่ใช้ชิพ (Chip) เบอร์ 8250 หรือ 16450 UART เป็นอะแดปเตอร์ที่พื้นฐานที่สุดสำหรับการสื่อสารโดยทั่วไป
- ประเภทที่สองคล้าย ๆ กับประเภทแรกแต่จะใช้ชิพเบอร์ 16550 และ FIFO (First In First Out) บัฟเฟอร์ ซึ่งมีประสิทธิภาพมากกว่า ใช้สำหรับการสื่อสารโมเด็มความเร็วสูงและสภาวะแวดล้อมการทำงานแบบหลาย ๆ งาน (Multitasking Environments) แต่จะมีราคาแพงกว่าประเภทแรก
- ประเภทที่สามประกอบด้วยอะแดปเตอร์แบบพิเศษคือ ตัวควบคุมพอร์ตอนุกรม(Serial Port Controller) และเฮส์เอนฮานซด์ซีเรียลอินเตอร์เฟซ (Hayes Enhanced Serial Interface : HESI) สำหรับประเภทที่สามนี้จะสนับสนุนการถ่ายเทข้อมูลแบบ DMA (Direct Memory Access) และมีโฟลว์คอนโทรล (Flow Control) ของตัวเอง HESI เป็นโคโพรเซสเซอร์ (Coprocessor) ทางการสื่อสารที่สมบูรณ์สามารถใช้ไมโครโพรเซสเซอร์ของตนเองในการจัดการสื่อสารโดยไม่ขึ้นอยู่กับประเภทของคอมพิวเตอร์

พอร์ตอนุกรมประกอบไปด้วยโครงสร้างหลายอย่างด้วยกัน คือ



รูปที่ 3.1 แสดงบล็อคไดอะแกรมพอร์ตอนุกรม

1. ไอโอแอดเดรส (I /O Address) ในคอมพิวเตอร์จะมีไอโอแอดเดรสสำหรับใช้อ้างถึงอุะแดปเตอร์แบบอนุกรมอยู่ 4 ช่วง ดังรูปที่ 3.2 ไอโอแอดเดรสดังกล่าวเชื่อมต่อกับ UART (Universal Asyncronous Recerver Transmitter) ซึ่งทำหน้าที่ในการติดต่อสื่อสารแบบอะซิงโครนัสและอินเตอร์เฟสแบบ RS-232 เมื่อเราทำการติดตั้งพอร์ตอนุกรมอันแรก (COM1) จะมีฐานแอดเดรสอยู่ที่ 03F8H โดยเราสามารถอ้างถึงรีจิสเตอร์ใน UART จากฐานช่องแอดเดรสนี้ เช่น รีจีสเตอร์ตัวที่สี่ของ COM1 จะมีแอดเดรสตรงกับพีซี I/O แอดเดรสที่ 03F8H + 4 = 03FCH

พอร์ตอนุกรม	ฐานไอโอแอดเดรส	IRQ มาตรฐาน
COM1	03F8H-03FFH	4
COM2	02F8H-02FFH	3
COM3	03E8H-03EFH	2
COM4	02E8H-02EFH	1

รูปที่ 3.2 ตารางแสดงความสัมพันธ์ระหว่างเบอร์พอร์ตอนุกรม และไอโอแอดเดรส

<u>หมายเหตุ</u> COM1, COM2, COM3 และ COM4 เป็นชื่อของอุปกรณ์ (Device) ที่ตั้งโดยระบบปฏิบัติการ (MS-DOS) สำหรับพอร์ตอนุกรมอันที่ 1 ถึงอันที่ 4 ตามลำดับที่ติดตั้งอยู่ในระบบ เพราะโปรแกรมส่วนใหญ่จะติดต่อกับฮาร์ดแวร์โดยตรง ดังนั้นชื่อเหล่านี้จึงเป็นที่รู้จักโดยทั่วไปเมื่ออ้างถึงการสื่อสารแบบอะซิงโครนัส

2. การเชื่อมต่อกับพีซีบัส (Bus Interface) การอ่านและการเขียนข้อมูลผ่านรีจิสเตอร์ของ UART ไอโอแอดเดรสของรีจิสเตอร์นั้นจะวางอยู่บนพีซีแอดเดรสบัส พีซีจะสโตรป (Storbe) สัญญาณ IOR (I /O read) หรือ (I /O write) เมื่อส่วนวงจรถอดรหัสแอสเดรส (Address Decoding Logic) ที่อยู่บนการ์ดอนุกรมตรวจสอบพบว่าเป็นแอดเดรสของตัวมันเอง มันจะทำการกระตุ้นสัญญาณเลือกชิป (Chip Select) ของ UART (CS2) การอ้างถึงรีจิสเตอร์ภายในของUART สามารถทำได้โดยการถอดรหัส(Decode Address) จากเส้นแอดเดรส A2 - A0 ซึ่งเป็น 3 บิตต่ำของไอโอแอดเดรส

สัญญาณเลือกชิปจะทำให้สัญญาณข้อมูลของ UART เชื่อมต่อกับบัสข้อมูลของพีซีโดยการควบคุมแบบรับส่งสองทิศทาง (Bidirection Bus Transceiver) ทิศทางการไหลของข้อมูลที่ผ่านตัวรับส่ง (Tranceiver) จะได้จากการตรวจสอบสัญญาณ IOR และ IOW ของพีซี ซึ่งสัญญาณเหล่านี้จะต่อกับสัญญาณ RD และ WR ของ UART

3. การเชื่อมต่ออินเตอร์รัพต์ (Interrupt Interface) สำหรับแต่ละพอร์ตอนุกรมจะมีสัญญาณอินเตอร์รัพต์โดยต่อสัญญาณ INTR เข้ากับบัสของพีซี โดยได้กำหนดให้ IRQ4 ใช้ร่วมกันระหว่าง COM1 และ COM3 ส่วน IRQ3 ใช้ร่วมกันระหว่าง COM2 และ COM4 หมายความว่าเราไม่สามารถที่จะใช้ COM1, COM3 โดยใช้แบบอินเตอร์รัพต์ได้ในเวลาเดียวกัน

จากบล็อคไดอะแกรมโครงสร้างพอร์ตอนุกรมข้างต้น จะเห็นว่าสัญญาณ INTR. ไม่ได้ต่อโดยตรงกับสัญญาณ IRQ ที่อยู่บนบัสของพีซี แต่จะมีลอจิกเกต (Logic Gate) คั่นกลาง ซึ่งควบคุมการเปิดปิดของเกตนี้โดยใช้สัญญาณ OUT2 ของ UART ดังนั้นในการที่จะทำให้เกิดการอินเตอร์รัพต์ขึ้นจึงต้องมีการโปรแกรมให้เซตบิต OUT2 ที่อยู่ในรีจีสเตอร์ควบคุมโมเด็ม (Modem Control Register) เป็น 1 เสียก่อน

4. อินเตอร์เฟส RS - 232 ขา DTR, RTS และ SOUT ของ UART จะต่อกับอินเตอร์เฟสแบบ RS-232 โดยผ่านตัวขับสัญญาณ (Line Drivers) สำหรับตัวขับสัญญาณนี้จะแปลงสัญญาณ 0 โวลต์ และ 5 โวลต์ ของ UART ไปเป็น -12 โวลต์ และ +12 โวลต์ ตามมาตรฐานของ RS-232 เช่นเดียวกันกับขาสัญญาณอินพุท CD, DSR, CTS, RI และ RD ของ RS-232 ก็จะต่อกับขาของ UART ที่สอดคล้องกัน ในการออกแบบตัวขับสัญญาณของ RS-232 นั้นจะใช้ตัวขับสัญญาณที่มีลอจิกตรงข้าม คือมันจะทำการกลับค่าลอจิกอินพุทที่มีมาจาก UART

5. รายละเอียดของรีจีสเตอร์ของ UART ในรายละเอียดของรีจีสเตอร์จะเป็นการกล่าวถึงหน้าที่ของแต่ละบิตที่อยู่ภายในรีจีสเตอร์ ซึ่งจะกล่าวถึงโดยอ้างจากฐานแอดเดรสของพอร์ต COM1 (03F8h - 03FFh) ส่วนพอร์ตอนุกรม

มาตรฐาน RS-232

มาตรฐาน RS-232 ได้จัดพิมพ์ขึ้นเมื่อปีค.ศ.1969 เป็นมาตรฐานที่กำหนดขึ้นโดยสมาคมของโรงงานอุตสาหกรรมผู้ผลิตอุปกรณ์อิเล็กทรอนิกส์แห่งอเมริกา (EIA : The Electronics Industries Association) ซึ่งใช้กันแพร่หลายในระบบการสื่อสารข้อมูลคอมพิวเตอร์ RS ย่อมาจาก Recommened Standard ส่วน 232 เป็นหมายเลขบ่งบอกของมาตรฐานตัวนี้ C เป็นหมายเลขของฉบับท้ายสุดของมาตรฐาน โดยจะกล่าวถึงมาตรฐานของลักษณะสัญญาณไฟฟ้าในการอินเตอร์เฟสเทอร์มินอลเข้ากับไมโครคอมพิวเตอร์ หรืออินเตอร์เฟสเครื่องพิมพ์เข้ากับคอมพิวเตอร์เป็นต้น โดยจะทำการส่งข้อมูลแบบอนุกรม จุดประสงค์ของมาตรฐานนี้ก็เพื่อบรรยายคุณลักษณะของการเชื่อมต่ออุปกรณ์รับ-ส่งข้อมูลปลายทาง (Data Terminal Equipment : DTE) กับอุปกรณ์สื่อสารข้อมูล (Data Communication Equipment : DCE) สำหรับผู้ใช้ไมโครคอมพิวเตอร์ DTE หมายถึงตัวไมโครคอมพิวเตอร์ และ DCE หมายถึงโมเด็ม อุปกรณ์ อื่น ๆ เช่นเครื่องพิมพ์ที่รับสัญญาณแบบอนุกรมอาจจะเป็นได้ทั้ง DTE และ DCE ขึ้นอยู่กับผู้ผลิต ซึ่งความเร็วและระยะทางของการเชื่อมต่อของ RS-232 สามารถเชื่อมต่อการถ่ายโอนข้อมูลได้จาก 0-20,000 บิตต่อวินาที ซึ่งเพียงพอสำหรับไมโครคอมพิวเตอร์ที่มีอัตราบอด 110 ถึง 9600 บอด ความยาวของสายเชื่อมต่อโดยสัญญาณตามมาตรฐานของ RS-232 จำกัดอยู่ประมาณ 50 ฟุต

สักษณะของสัญญาณ RS-232

ถ้าแรงดันไฟฟ้าเป็นบวกสภาพลอจิกจะเป็นศูนย์สถานภาพของสัญญาณจะเป็นสเปซและฟังก์ชันในการควบคุมจะเป็นออน แต่เมื่อแรงดันไฟฟ้าเป็นลบ สภาพลอจิกจะเป็นหนึ่ง ส่วนสภานภาพของสัญญาณจะเป็นมาร์คและฟังก์ชันในการควบคุมจะเป็นออฟ ลักษณะของสัญญาณที่ขาต่าง ๆ จะเป็นดังนี้

- สัญญาณ TRANSMIT DATA เป็นสัญญาณที่ส่งออกจาก DTE หรือไมโครคอมพิวเตอร์ ไปยังโมเด็มหรือต่อเข้าโดยตรงกับไมโครคอมพิวเตอร์ตัวอื่น หรือเครื่องพิมพ์ เมื่อไม่มีสัญญาณส่งออกสถานภาพของลอจิกที่ขานี้จะค่าเท่ากับหนึ่ง
- สัญญาณ RECEIVE DATA เป็นทางเข้าของสัญญาณไปยัง DTE หรือไมโครคอมพิวเตอร์ เมื่อไม่มีสัญญาณรับเข้ามา ขานี้จะมีสถานภาพของลอจิกเท่ากับหนึ่ง
- สัญญาณ REQUEST TO SEND ใช้สำหรับส่งสัญญาณไปยังโมเด็มหรือเครื่องพิมพ์เป็นการเรียกร้องที่ส่งสัญญาณมาทางขา 2 สัญญาณนี้ใช้คู่กับ CTS หรือ Clear To Send อุปกรณ์รับหากได้รัสัญญาณ RTS ก็จะทราบว่าพร้อมจะรับสัญญาณได้หรือยัง หากพร้อมที่จะรับก็จะส่งสัญญาณออกไปที่สาย CTS
- สัญญาณ CLEAR TO SEND เมื่ออยู่ในสภานะออฟหรือลอจิกหนึ่ง หมายความว่าพร้อมจะรับข้อมูลแล้ว
- สัญญาณ DATA SET READY เมื่อสัญญาณสายนี้อยู่ในสถานะออนหรือลอจิกศูนย์ เป็นการบอกไมโครคอมพิวเตอร์ หรือฝ่ายส่งว่าโมเด็มต่อเข้ากับสายโทรศัพท์เรียบร้อยแล้ว และพร้อมที่จะส่งแล้ว โมเด็มที่มีการหมุนหมายเลขอัตโนมัติจะส่งสัญญาณสายนี้ไปบอกให้คอมพิวเตอร์รู้ว่าต่อโทรศัพท์ได้สำเร็จแล้ว

- สัญญาณ SIGNAL GROUND ทำหน้าที่เป็นระดับแรงดันอ้างอิง สำหรับทุก ๆ สายสัญญาณจะมีแรงดันเป็นศูนย์เมื่อเทียบกับสัญญาณตัวอื่น
- สัญญาณ CARRIER DETECT โมเด็มจะส่งสัญญาณที่อยู่ในสถานะออนหรือลอจิกศูนย์ เมื่อพร้อมที่จะติดต่อกับโมเด็ม โมเด็มส่วนมากจะไม่รายงานสถานภาพของตัวเอง (CD, USR, CTS) ให้คอมพิวเตอร์รู้หากคอมพิวเตอร์ไม่เปิดสัญญาณ DTR
- สัญญาณ RING INDICATOR สัญญาณนี้ใช้ในโมเด็มที่เป็นระบบตอบรับอัตโนมัติ สัญญาณนี้จะออนเมื่อมีสัญญาณกระดิ่งมา และออฟระหว่างเสียงดังของกระดิ่ง

3.3 การอินเตอร์รัพต์ (Interrupt) หรือการขัดจังหวะ

การอินเตอร์รัพต์ หรือการขัดจังหวะ คือการที่โปรเซสหนึ่งที่กำลังทำงานอยู่ ถูกสั่งให้หยุดการทำงาน ซึ่งอาจจะสั่งมาจากโปรเซสอื่นจากฮาร์ดแวร์ หรือจากโปรเซสตัวเองก็ได้ เพื่อทำการเริ่มต้นทำงานโปรเซสอีกโปรเซสหนึ่ง และเมื่อโปรเซสนี้ทำงานเสร็จสิ้นแล้วโปรเซสที่ถูกขัดจังหวะก็จะทำงานต่อไปโดยที่โปรแกรมอินเตอร์รัพต์ (Interrupt Service Routine : ISR) คือโปรเซสที่จะถูกเรียกขึ้นมาทำงานเมื่อเกิดการร้องขออินเตอร์รัพต์ ซึ่งตารางอินเตอร์รัพต์เวกเตอร์ (Interrupt Vector Table : IVT) คือตารางเก็บค่าตำแหน่งหน่วยความจำเริ่มต้นของฟังก์ชันที่จะเป็นโปรเซสทำงานตอบสนองการร้องขออินเตอร์รัพต์ และมีอินเตอร์รัพต์ฟังก์ชัน (Interrupt Function) เป็นชื่ออีกชื่อของโปรแกรมบริการอินเตอร์รัพต์ ในมุมมองฟังก์ชันตามความหมายของภาษาซีหมายถึงกลุ่มของคำสั่งที่ทำหน้าที่ใดหน้าที่หนึ่ง ซึ่งจะเริ่มทำงานเมื่อได้รับการร้องขออินเตอร์รัพต์ (ตามหมายเลขที่ได้ติดตั้งอินเตอร์รัพต์ฟังก์ชันไว้)

ในขณะที่มีโปรเซสหนึ่งกำลังทำงานอยู่ในระบบ โปรเซสนั้นอาจถูกขัดจังหวะการทำงานได้จากความจำเป็นใด ๆ ก็ตาม ในระบบที่ต้องการการจัดการสิ่งใดสิ่งหนึ่งเร่งด่วนกว่าโปรเซสที่กำลังทำงานอยู่ในปัจจุบัน การขัดจังหวะที่เกิดขึ้นจะทำให้โปรเซสที่กำลังทำงานอยู่นั้นหยุดการทำงานลงชั่วคราว เพื่อให้ระบบหันไปทำงานในโปรเซสที่จะใช้จัดการงานเร่งด่วนนั้น เมื่องานที่เร่งด่วนกว่าเสร็จสิ้นลงโปรเซสที่ถูกหยุดค้างไว้ก็จะทำงานต่อไป การถูกขัดจังหวะเพื่อไปกระทำงานที่เร่งด่วนกว่านี้ เรียกว่าการอินเตอร์รัพต์ฟังก์ชัน หรือโปรแกรมบริการอินเตอร์รัพต์

สำหรับเครื่องพีซี แหล่งที่จะกำเนิดสัญญาณที่จะบอกการเกิดการอินเตอร์รัพต์นี้อาจมาจากฮาร์ดแวร์ เช่นเมื่อเกิดการหารด้วยศูนย์ซีพียูจะส่งสัญญาณอินเตอร์รัพต์ เพื่อให้ระบบปฏิบัติการทราบว่าค่าที่ได้จากการคำนวณนั้นเป็นอนันต์ เป็นต้น นอกจากนี้สัญญาณอินเตอร์รัพต์อาจได้จากโปรเซสที่กำลังทำงานอยู่สั่งขัดจังหวะตัวเอง อันเป็นวิธีที่เราจะใช้ในการเรียกอินเตอร์รัพต์ฟังก์ชันของระบบปฏิบัติการ และในกรณีสุดท้ายสัญญาณอินเตอร์รัพต์อาจมาจากโปรเซสอื่นที่อยู่ในระบบ ซึ่งอาจจะกำลังทำงานไปพร้อมกับโปรเซสปัจจุบัน หรืออาจจะถูกสั่งให้เริ่มทำงานเมื่อมีสัญญาณร้องขออินเตอร์รัพต์ หรือมีการเรียกใช้โปรเซสในนามของฟังก์ชัน เราอาจแยกประเภทของการอินเตอร์รัพต์ตามแหล่งที่มาได้สองชนิดคือ ฮาร์ดแวร์อินเตอร์รัพต์ (Hardware Interrupt) เป็นอินเตอร์รัพต์ที่ถูกร้องขอโดยขอจากโปรเซสใดโปรเซสหนึ่งที่กำลังทำงานภายในระบบ

เมื่อมีการร้องขออินเตอร์รัพต์ ไม่ว่าจะมาจากแหล่งใดก็ตามจะมีการแจ้งหมายเลขอินเตอร์รัพต์มาด้วยเสมอ ค่าหมายเลขอินเตอร์รัพต์นี้จะถูกนำมาเปิดตารางอินเตอร์รัพต์เวกเตอร์ และนำค่าในตารางที่ได้มาเป็นค่าตำแหน่งหน่วยความจำเริ่มต้นของอินเตอร์รัพต์ฟังก์ชัน เพื่อใช้เป็นจุดกระโดดไปทำงานของระบบ

หลังจากที่ระบบทำงานตามอินเตอร์รัพต์ฟังก์ชันเสร็จแล้ว ที่คำสั่งสุดท้ายของอินเตอร์รัพต์ฟังก์ชันจะมีคำสั่งที่ใช้กระโดดกลับไปยังจุดที่เรียกมา คำสั่งนี้จะทำให้ระบบกระโดดกลับไปทำงานในโปรเซสที่ค้างอยู่ต่อไป โดยปกติเมื่อฟังก์ชันใด ๆ ทำงานเสร็จสิ้นก็จะกระโดดไปยังฟังก์ชันที่เรียกมาด้วยคำสั่งRET(Return) คอมไพเลอร์จะเติมคำสั่ง RET นี้ให้โดยอัตโนมัติในขณะแปล แต่สำหรับการกระโดดกลับจากอินเตอร์รัพต์ฟังก์ชันจะใช้คำสั่ง RETI (Return from Interrupt)

ในตารางอินเตอร์รัพต์จะมีอยู่ 256 หน่วย แต่ละหน่วยเป็นค่าการชี้ฟังก์ชันแบบไกล ด้วยเหตุนี้เครื่องพีซีจึงมีหมายเลขอินเตอร์รัพต์ฟังก์ชันได้ตั้งแต่หมายเลข 0 ถึง 255 การส่งค่าไปยังอินเตอร์รัพต์ฟังก์ชันหรือส่งค่ากลับจะใช้การส่งผ่านค่าทางรีจีสเตอร์

<u>กรรมวิธีในการเขียนโปรแกรมที่มีอินเตอร์รัพต์ฟังก์ชัน</u>

มีขั้นตอนดังต่อไปนี้อยู่ภายในลำดับขั้นตอนการทำงานของโปรแกรมคือ

1. กำหนดหมายเลขอินเตอร์รัพต์ที่ต้องการใช้สำหรับอินเตอร์รัพต์ฟังก์ชันที่เขียนขึ้น ซึ่งอาจจะใช้หมายเลขที่ยังไม่มีอินเตอร์รัพต์ฟังก์ชันใดใช้ หรืออาจจะใช้หมายเลขที่มีอินเตอร์รัพต์ใช้งานอยู่ก่อนแล้ว ซึ่งในกรณีหลังเมื่อมีการร้องขออินเตอร์รัพต์ที่ไปเรียกอินเตอร์รัพต์ฟังก์ชันเดิมก็จะเกิดการกระโดดไปทำงานในอินเตอร์รัพต์ใหม่แทน ดังนั้นในอินเตอร์รัพต์ฟังก์ชันตัวใหม่ของกรณีนี้จะต้องมีการเรียกใช้อินเตอร์รัพต์ฟังก์ชันเดิมด้วย

2. เก็บค่าการชี้อินเตอร์รัพต์ฟังก์ชันเดิมที่ใช้โดยใช้ฟังก์ชัน getvect ( ) ซึ่งจะมีโปรโตไทป์ดังนี้ void interrupt (*getvect(int n))(void) ; คำสงวน interrupt ที่ปรากฏอยู่ในโปรโตไทป์เป็นการกำหนดให้ค่าการชี้ฟังก์ชันที่ฟังก์ชัน getvect ( ) ส่งกลับมานี้เป็นการชี้อินเตอร์รัพต์ฟังก์ชัน ซึ่งหมายถึงค่าตำแหน่งหน่วยความจำเริ่มต้นของอินเตอร์รัพต์ฟังก์ชันนั่นเอง ฟังก์ชัน getvect ( ) จะให้ค่าการชี้อินเตอร์รัพต์ฟังก์ชันจากอินเตอร์รัพต์หมายเลขที่กำหนดเป็นพารามิเตอร์ n ในฟังก์ชัน

3. กำหนดค่าการชี้อินเตอร์รัพต์ฟังก์ชันของตารางอินเตอร์รัพต์เวกเตอร์ในหมายเลขที่ต้องการ ด้วยค่าการชี้อินเตอร์รัพต์ฟังก์ชันที่สร้างขึ้น หลังจากที่อ่านค่าการชี้อินเตอร์รัพต์ในตารางอินเตอร์รัพต์เวกเตอร์ตามหมายเลขที่ต้องการด้วยฟังก์ชัน getvect ( ) แล้วเราก็จะกำหนดค่าการชี้อินเตอร์รัพต์ฟังก์ชันตัวใหม่ลงไปแทนในตัวเก่าที่เก็บค่าไว้ด้วยฟังก์ชัน setvect ( ) เพื่อให้ตารางอินเตอร์รัพต์เวกเตอร์ในหมายเลขที่ต้องการ ชี้ไปยังอินเตอร์รัพต์ฟังก์ชันที่สร้างขึ้นแทนค่าเก่า ฟังก์ชัน setvect ( ) มีรูปดังนี้ void setvect (int n, void interrupt (* isr)( )) ; ฟังก์ชัน setvect ( ) ทำหน้าที่กำหนดค่าการชี้อินเตอร์รัพต์ฟังก์ชันใหม่ในตารางอินเตอร์รัพต์เวกเตอร์ตามหมายเลขอินเตอร์รัพต์ที่กำหนด ค่าที่ฟังก์ชัน setvect ( ) รับไปเป็นพารามิเตอร์คือค่าหมายเลขอินเตอร์รัพต์ที่ต้องการเปลี่ยนค่า และค่าการชี้อินเตอร์รัพต์ฟังก์ชันชนิดไม่มีการส่งผ่านค่า เพราะอินเตอร์รัพต์ฟังก์ชันทั่วไปจะส่งผ่านค่าทางรีจีสเตอร์แทนการส่งผ่านสแต็กดังฟังก์ชันทั่วไป

4. ก่อนที่จะจบการทำงานของโปรแกรม จะต้องทำการคืนค่าการชี้อินเตอร์รัพต์ฟังก์ชันเดิมแก่ตารางอินเตอร์รัพต์ มิฉะนั้นเมื่อโปรแกรมจบการทำงานมีการนำหน่วยความจำที่โปรแกรมนั้นเคยใช้ไปให้โปรแกรมหรือโปรเซสอื่น และมีการเรียกใช้อินเตอร์รัพต์หมายเลขนี้อีก จะทำให้เกิดการกระโดดไปทำงานยังพื้นที่ที่ไม่ใช่อินเตอร์รัพต์ฟังก์ชัน (เพราะได้นำข้อมูลใหม่ไปใส่แทนแล้ว) จะทำให้ระบบเสียหายได้

ข้อควรระวังในการเขียนโปรแกรมอินเตอร์รัพต์

1. ขนาดของอินเตอร์รัพต์ฟังก์ชันควรจะสั้นที่สุดเท่าที่จะทำได้ โดยยึดหลักที่ว่าเวลาที่เสียไปในการจัดการอินเตอร์รัพต์ฟังก์ชันทั้งหมดในการร้องขออินเตอร์รัพต์ครั้งหนึ่ง ๆ จะต้องน้อยกว่าช่วงเวลาในการร้องขอแต่ละครั้ง เพราะถ้าหากเวลารวมมีค่าเกินกว่าช่วงเวลาในการร้องขอ จะเกิดการร้องขออินเตอร์รัพต์เดิมซ้ำในขณะที่ยังไม่จบการทำงานอินเตอร์รัพต์ ซึ่งถ้ายังคงเกิดสภาวะเช่นนี้เรื่อยไปจะทำให้สแต็กเต็มและระบบหยุดทำงานได้ เพราะการเรียกอินเตอร์รัพต์ฟังก์ชันแต่ละครั้งจะมีการเก็บค่าตำแหน่งหน่วยความจำของคำสั่งที่ทำงานค้างไว้ในสแต็กเสมอ

2. อย่าลืมที่จะเรียกใช้อินเตอร์รัพต์ฟังก์ชันเดิมในอินเตอร์รัพต์ฟังก์ชันใหม่ที่สร้างขึ้นมิฉะนั้นอินเตอร์รัพต์เดิมจะไม่ถูกจัดการ

3. อย่าลืมคืนค่าการชี้อินเตอร์รัพต์ฟังก์ชันเดิมให้แก่ตารางอินเตอร์รัพต์เวกเตอร์ ก่อนออกจากโปรแกรม หากไม่คืนค่าการชี้แล้วเมื่อมีการร้องขออินเตอร์รัพต์นี้อีก ก็จะเกิดการกระโดดไปยังจุดที่เคยเป็นตัวโปรแกรมซึ่งหลังจากนั้นอาจจะถูกใช้งานโดยโปรเซสอื่น ทำให้ระบบเสียหายได้

บทที่ 4
แนวคิดและการออกแบบ

แนวคิดและการออกแบบ

ปริญญานิพนธ์นี้ประกอบด้วยส่วนของฮาร์ดแวร์และซอฟต์แวร์ ซึ่งฮาร์ดแวร์นั้นเพื่อตัดปัญหาเราได้ใช้อุปกรณ์ที่มีอยู่แล้ว และที่หาซื้อได้ในตลาดของอุปกรณ์ที่เกี่ยวกับคอมพิวเตอร์ เช่น สายส่งสัญญาณ , การ์ดเสียง

สำหรับซอฟแวร์ เป็นส่วนสำคัญของปริญญานิพนธ์ เพราะเป็นการออกแบบโปรแกรมสื่อสารผ่านพอร์ตอนุกรม โดยใช้สื่อในการสื่อสารแบบต่าง ๆ เช่น ข้อความ ภาพ และเสียง สิ่งสำคัญที่เรานำมาพิจารณาในการออกแบบโปรแกรมนั้นแบ่งเป็น 2 ส่วน ดังนี้

1. สื่อที่นำมาใช้ในการสื่อสาร และการแสดงผล
2. ขั้นตอนการรับส่งข้อมูล

1. สื่อที่นำมาใช้ในการสื่อสารและการแสดงผล

1.1 ข้อความ เป็นการสื่อสารด้วยตัวอักษรภาษาอังกฤษ โดยรับข้อมูลจากคีย์บอร์ด การติดต่อสามารถทำได้ใน 2 ทิศทาง กล่าวคือฝ่ายรับและฝ่ายส่งสามารถส่งข้อความพร้อมกันได้ มีการแสดงผลข้อความที่ทำการโต้ตอบกันทั้ง 2 ฝ่าย ซึ่งจะทำให้การสื่อสารทำได้สะดวกเกิดความเข้าใจตรงกันทั้ง 2 ฝ่าย

1.2 ภาพ เนื่องจากภาพนั้นมีรูปแบบในการจัดเก็บหลายรูปแบบ แต่ละรูปแบบก็ถูกออกแบบโดยบริษัทต่าง ๆ และมีความเหมาะสมกับงานในแต่ละลักษณะ โครงงานนี้เลือกนำการจัดเก็บรูปภาพในแบบของ PCX ไฟล์ ซึ่งเป็นรูปแบบการจัดเก็บภาพที่เก่าแก่วิธีหนึ่ง ที่มีการใช้งานกันอย่างแพร่หลาย เช่น สามารถนำไปใช้งานร่วมกับเวิร์ดโปรเซสเซอร์, การจับหน้าจอ (Screen Capture) เป็นต้น จึงได้ศึกษาถึงโครงสร้างของ PCX ไฟล์ ซึ่งมีการแสดงผลแบบ 256 สี, ใช้ 8 บิต ต่อการเก็บ 1 พิกเซล, ทำการเข้ารหัสแบบรันเลนท์ (Run - Length Coding) เป็นการลดขนาดข้อมูลแบบไม่มีการสูญเสีย อันจะทำให้ไฟล์ข้อมูลมีขนาดเล็กลง นอกจากนี้รูปแบบการเข้ารหัสยังไม่ซับซ้อน ง่ายต่อการทำความเข้าใจ และมีคุณภาพของภาพอยู่ในขั้นดี ซึ่งก็ได้อธิบายไว้แล้วในบทที่ 2

1.3 เสียง การสื่อสารด้วยเสียงในโครงงานนี้เป็นลักษณะของการฝากข่าวสารทางเสียง กล่าวคือ เมื่อผู้รับต้องการส่งข่าวสารทางเสียงก็จะทำการบันทึกเสียงเก็บเป็นไฟล์เสียง แล้วทำการส่งไปยังด้านรับ เมื่อผู้รับได้รับไฟล์เสียงแล้วก็จะทำการเปิดไฟล์เสียง และเล่นกลับได้ ดังนั้นเครื่องคอมพิวเตอร์ทั้งของผู้รับและผู้ส่งจะต้องมีการ์ดเสียงด้วย

- ส่วนของฮาร์ดแวร์เลือกใช้การ์ดซาวน์บลาสเตอร์ (Sound Blaster 16 : SB16)
- ใช้รูปแบบการจัดเก็บแบบ .VOC ซึ่งเรียกใช้ไดร์เวอร์ CT - VOICE.DRV ของบริษัท Creative Labs เพราะมีลักษณะการจัดเก็บที่ง่าย ไม่ยุ่งยาก และใช้สำหรับเสียงที่ไม่ใช่เสียงดนตรีโดยเฉพาะ
- ทำการบันทึกเสียงโดยใช้อัตราการแซมปลิ้ง 5000 เฮิร์ตซึ่งเป็นอัตราที่ต่ำที่สุดที่ SB16 สามารถ

ทำได้ในรูปแบบของ .VOC ไฟล์ เพื่อให้มีขนาดเล็กที่สุด

1.4 การแสดงผล

- ทำการแสดงผลข้อความที่ทำการโต้ตอบในหน้าจอเดียวกัน ดังนั้นต้องแบ่งหน้าจอออกเป็น 2 ส่วน คือส่วนหนึ่งแสดงข้อความของผู้ส่ง อีกส่วนหนึ่งแสดงข้อความของคู่สนทนา
- ทำการแสดงผลไฟล์ภาพแบบ PCX ไฟล์ ในหน้าจอเดียวกับที่ทำการสนทนา ดังนั้นจะแบ่งหน้าจอออกเป็นส่วนแสดงภาพ ส่วนแสดงข้อความของผู้ส่ง และส่วนแสดงข้อความของคู่สนทนา นั่นหมายความว่าการแสดงข้อความและภาพต้องทำในโหมดแสดงผลของจอภาพโหมดเดียวกัน และโหมดของจอภาพต้องสามารถแสดงตัวอักษรที่มีขนาดยอมรับได้ ในปริญญานิพนธ์นี้เราได้ออกแบบโดยเรียกใช้โหมดการแสดงภาพแบบ SVGA256 ด้วยว่ามีการแสดงผลแบบ 256 สี และมีโหมดความละเอียดของจอภาพให้เลือกใช้ได้หลายค่า เช่น 320 * 200, 640 * 480 เป็นต้น และสามารถเรียกใช้ฟังก์ชันกราฟฟิกของภาษาซี โดยให้เลือกส่งไฟล์ภาพที่ต้องการส่งประกอบการสนทนาไปยังด้านรับก่อน แล้วจึงมีการแสดงผลภาพ จากนั้นส่วนของการสนทนาก็จะถูกเรียกขึ้นมา ทำให้สามารถส่งรูปภาพประกอบการสนทนาด้วยข้อความได้
- ในส่วนของเสียงจะประกอบด้วย การบันทึกเสียง การเล่นกลับไฟล์เสียง (Playback) ซึ่งต้องมีโปรแกรมมารองรับในส่วนนี้ ในการบันทึกเสียงเราสามารถกำหนดเวลาที่ต้องการใช้ในการบันทึกเสียงได้ โดยเลือกใส่ชื่อไฟล์ที่ต้องการเก็บ ส่วนการเล่นกลับไฟล์เสียง สามารถเล่นกลับไฟล์เสียงที่ต้องการได้ และทำการหยุดได้เมื่อต้องการ ดังนั้นเมื่อเราเลือกใช้รูปแบบการจัดเก็บไฟล์เสียงแบบ .VOC จึงเรียกฟังก์ชันต่าง ๆ ใน CT - VOICE . DRV มาใช้ในการเขียนโปรแกรมสั่งงาน เช่น ฟังก์ชันเซ็คสถานะการทำงานเริ่มต้นของการ์ดเสียง, ฟังก์ชันบันทึก (Record) สัญญาณเสียง, ฟังก์ชันเล่นย้อนกลับ (Playback) ข้อมูลของสัญญาณเสียง เป็นต้น

2. **การรับส่งข้อมูล** เราได้ทำการรับส่งข้อมูลผ่านพอร์ตอนุกรม ตามมาตรฐาน RS - 232C ซึ่งเป็นการสื่อสารแบบอะซิงโครนัส ประกอบไปด้วยบิตเริ่มต้น, บิตสิ้นสุด, บิตพาริตี้ และบิตข้อมูล โดยเรียกใช้ int86 ไลเบอรีของภาษาซีที่เป็นฟังก์ชันช่วยจัดการเกี่ยวกับการอินเตอร์รัพต์ให้

- ตัวอย่างการเรียกใช้งาน int 86

```
int86 ( 0 x 14,&r,&r )
```

: r เป็นตัวแปรแบบ unoin REGS

: 0x14 เป็นอินเทอร์รัพต์ฟังก์ชันให้เรียกใช้งานได้ ซึ่งในที่นี้เป็นอินเทอร์รัพต์ฟังก์ชันที่เกี่ยวกับพอรต์ RS - 232

- ตัวอย่างการเรียกใช้งานในการกำหนดสถานะเริ่มต้นของพอร์ต RS - 232

```
init _ port ( int  comport ,  int  code )
{ union  REGS r ;
  r.x.dx  =  comport ; /* รีจิสเตอร์ dx = comport เลือกใช้งานพอร์ตอนุกรม com1 หรือcom2 */
```

```
    r.h.ah  =  0 ;        /* รีจิสเตอร์ ah = 0 เลือกการกำหนดสถานะเริ่มต้นของพอร์ต */
    r.h.al  =  code ;     /*รีจิสเตอร์ al = code กำหนดอัตราบอด, บิตสิ้นสุด, บิตพาริตี้ */
    int 86 ( 0 x 14,&r,&r ) ;
  }
```

●ตัวอย่างการเรียกใช้งานการส่งข้อมูลออกพอร์ต

```
void  sport ( int  comport )
      { union  REGS r ;
        r.x.dx = comport ;
        r.h.ah = 1 ;          /*รีจิสเตอร์ ah = 1 เลือกการส่งข้อมูลออกพอร์ต */
        r.h.al = ch ;         /*รีจิสเตอร์ al = ch เมื่อ ch คือข้อมูล 1 word ที่ต้องการส่ง */
        int 86 ( 0 x 14,&r,&r ) ;
      }
```

●ตัวอย่างการเรียกใช้งานการรับข้อมูลจากพอร์ต

```
void  rport ( int  comport )
      { union  REGS r;
        r.x.dx = comport ;
        r.h.ah = 2 ;          /*รีจิสเตอร์ ah = 2 เลือกการรับข้อมูลจากพอร์ต */
        int86 ( 0 x 14,&r,&r ) ;
        return  r.h.al ;      /*รีจิสเตอร์ al จะ return ค่าที่อ่านได้จากพอร์ต */
      }
```

นอกจากนี้ยังสามารถอ่านสถานะของพอร์ตเพื่อตรวจดูว่า พอร์ตพร้อมที่จะทำการส่งข้อมูลหรือไม่ มีข้อมูลส่งมาที่พอร์ตหรือไม่ และอื่น ๆ โดยการกำหนดค่าให้รีจิสเตอร์ ah = 3 เป็นต้น

ลักษณะการส่งข้อมูลจะทำการส่งข้อมูลครั้งละ 1 ไบต์ ไปยังด้านรับจนกว่าจะหมดไฟล์ และเพื่อป้องกันข้อผิดพลาดที่อาจจะเกิดจากกรณีเครื่องทางด้านรับทำงานไม่ทันกับทางด้านส่ง จึงเพิ่มขั้นตอนการส่งรหัสในการซิงค์เข้าไป ทางด้านส่งก็จะรอจนกว่าจะได้รับรหัสซิงค์นั้นจึงจะส่งข้อมูลต่อไป ซึ่งทำให้เกิดความล่าช้าในการส่ง .ในส่วนนี้อาจนำการอินเตอร์รัพต์พอร์ตอนุกรมมาใช้โดยการเขียนโปรแกรมควบคุม เมื่อมีข้อมูลส่งมาที่พอร์ตจะเกิดอินเตอร์รัพต์ขึ้น ก็ให้ซีพียูกระโดดมาทำงานยังโปรแกรมที่ได้เลือกไว้, การจองบัฟเฟอร์เพื่อเก็บข้อมูลในการส่งและใช้ภาษาแอสแซมบลี้เขียนโปรแกรมในการติดต่อกับพอร์ตมาใช้ได้จะทำให้สามารถส่งข้อมูลได้เร็วขึ้น แต่ในปริญญานิพนธ์นี้ไม่ได้ทำในส่วนเหล่านี้ด้วย

จากแนวคิด และความต้องการดังที่กล่าวมา ได้นำมาใช้ในการออกแบบการทำงานของโปรแกรมในส่วนต่างๆได้ดังนี้

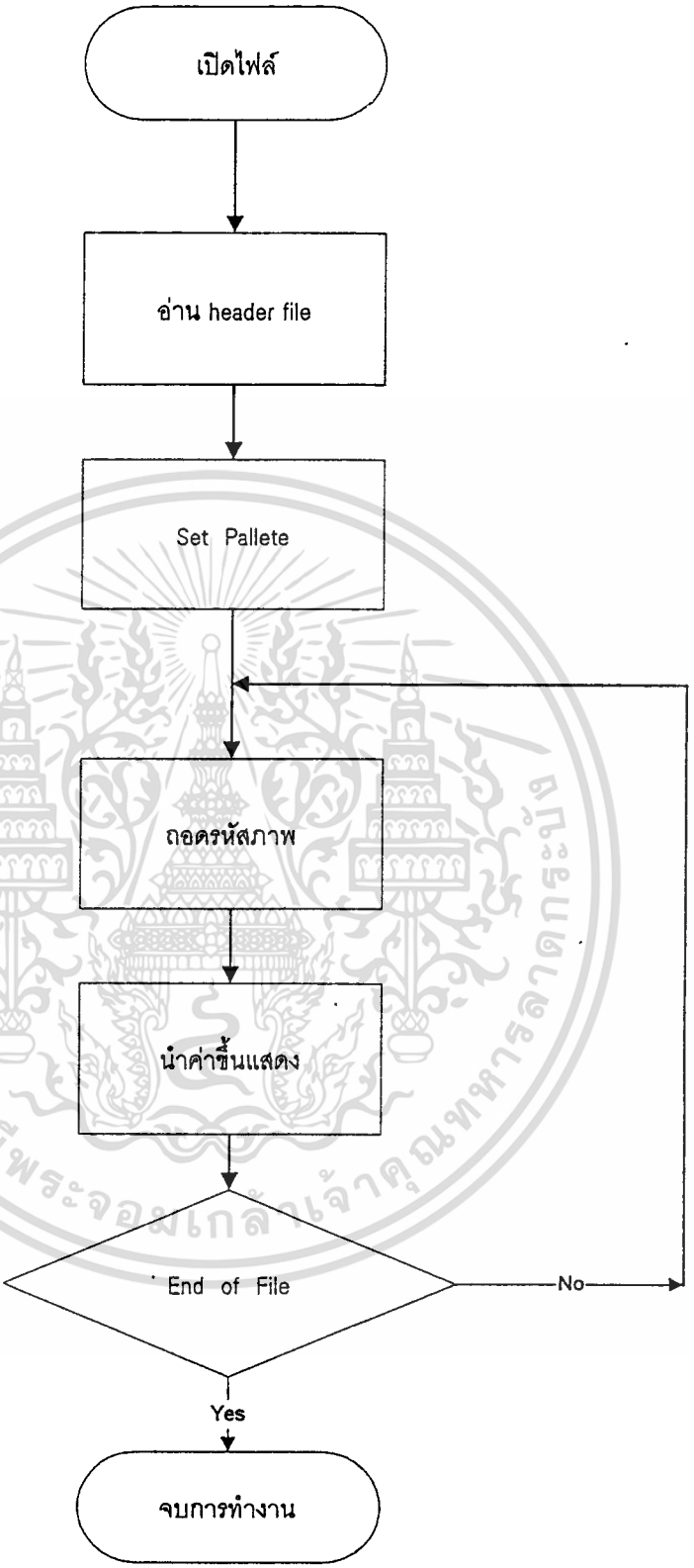
1. ส่วนการสื่อสารด้วยข้อความแบบ 2 ทิศทาง เป็นส่วนที่ใช้ในการสื่อสารด้วยข้อความภาษาอังกฤษ และทำการแสดงผลข้อความที่ทั้งทางด้านผู้ส่ง และด้านผู้รับทำการโต้ตอบซึ่งกันและกัน

2. ส่วนการแสดงภาพ PCX ไฟล์ เป็นส่วนที่รับไฟล์ภาพที่มีการจัดเก็บแบบ PCX มาทำการถอดรหัสข้อมูลแล้วนำมาแสดงบนจอคอมพิวเตอร์

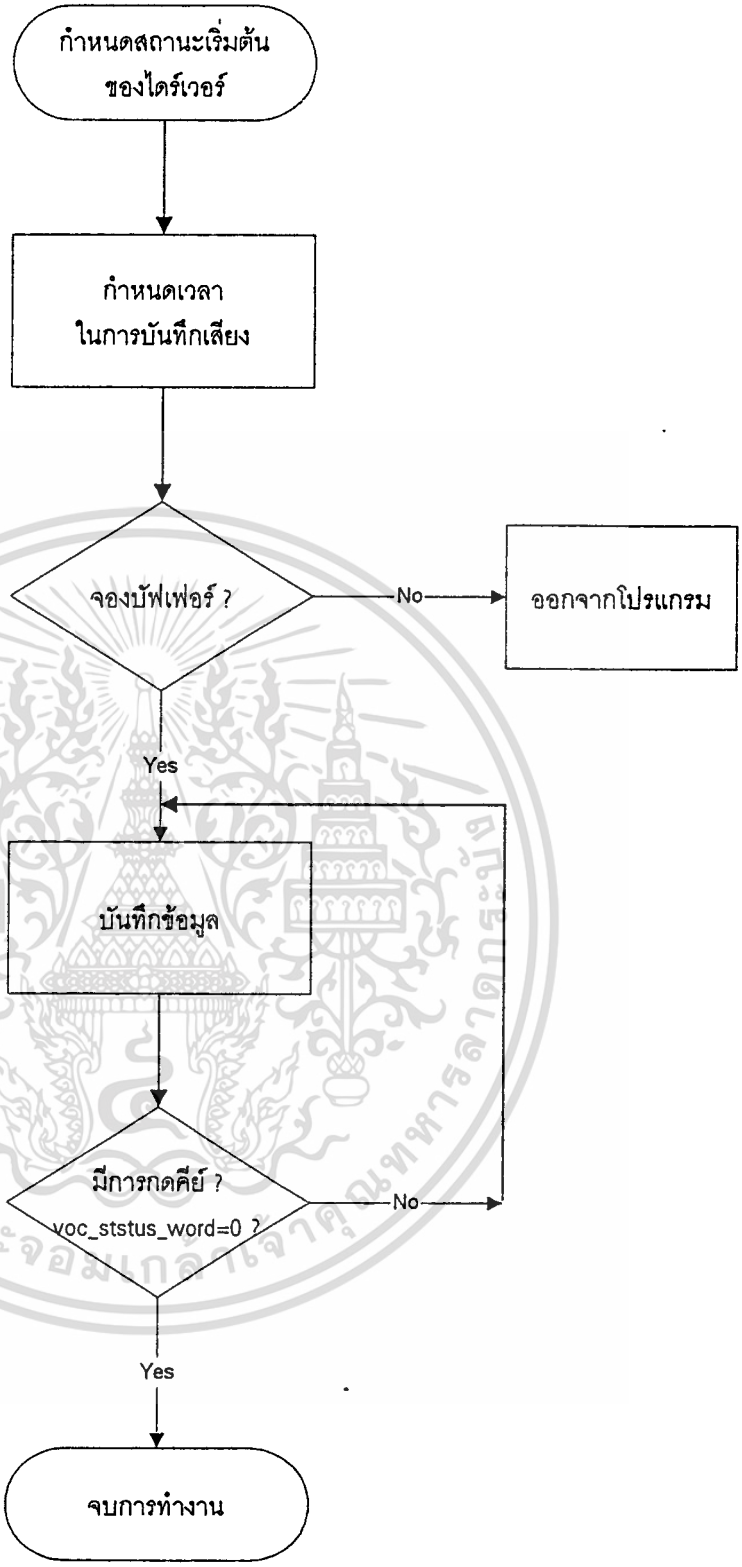
3. ส่วนการบันทึกเสียง เป็นส่วนที่ใช้ในการบันทึกข้อความเสียงที่ต้องการส่งไปยังผู้รับ

4. ส่วนการเล่นกลับไฟล์เสียง เป็นส่วนที่ใช้ในการเล่นกลับไฟล์เสียงที่บันทึกข้อความที่ต้องการฟัง

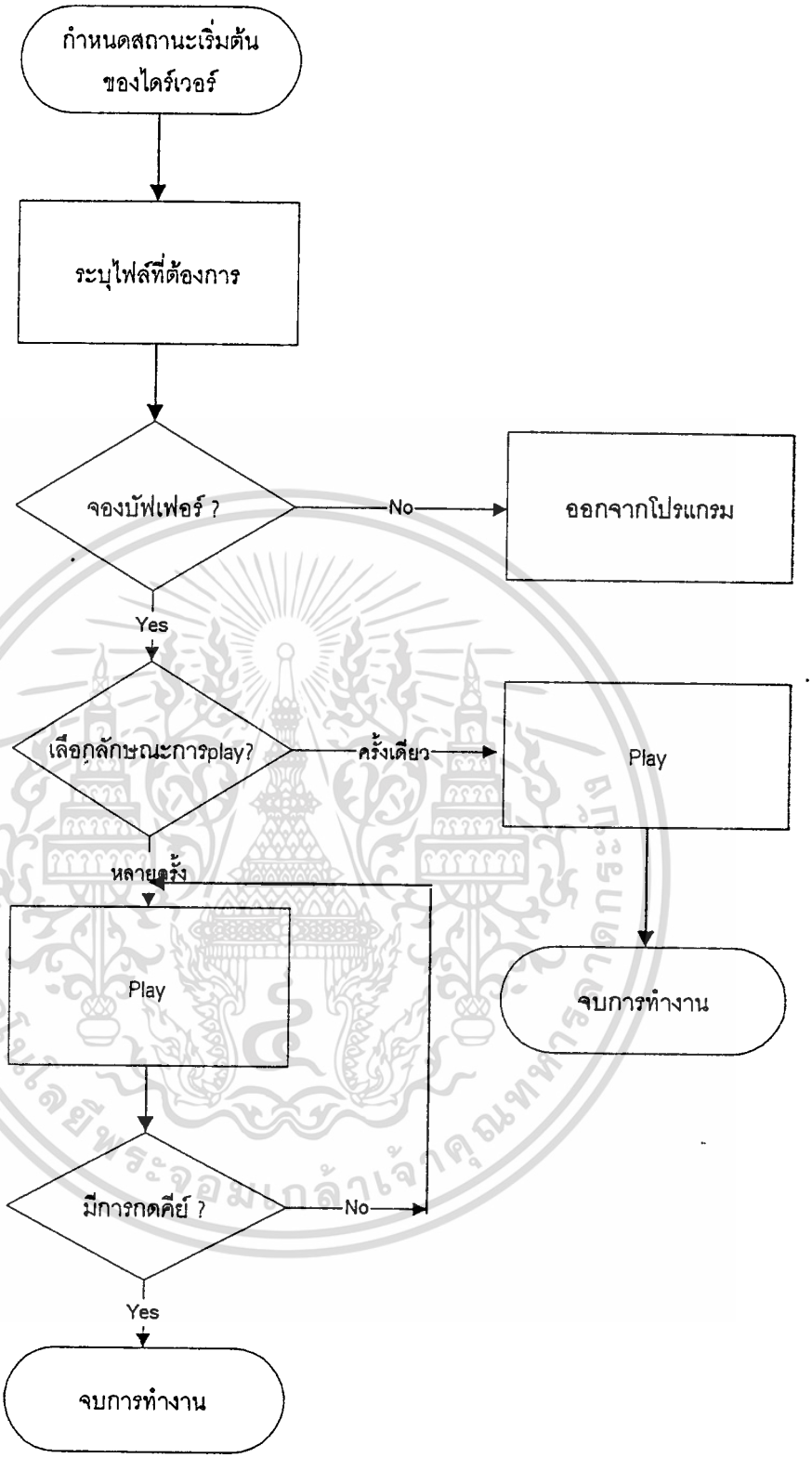
5. ส่วนการรับ และส่งไฟล์ข้อมูล เป็นส่วนที่ใช้ในการรับส่งไฟล์ข้อมูล โดยทำการส่งทีละ 1 ไบต์จนหมดทั้งไฟล์ การส่งแต่ละไบต์จะทำการส่งรหัสซิงค์ทุกครั้ง



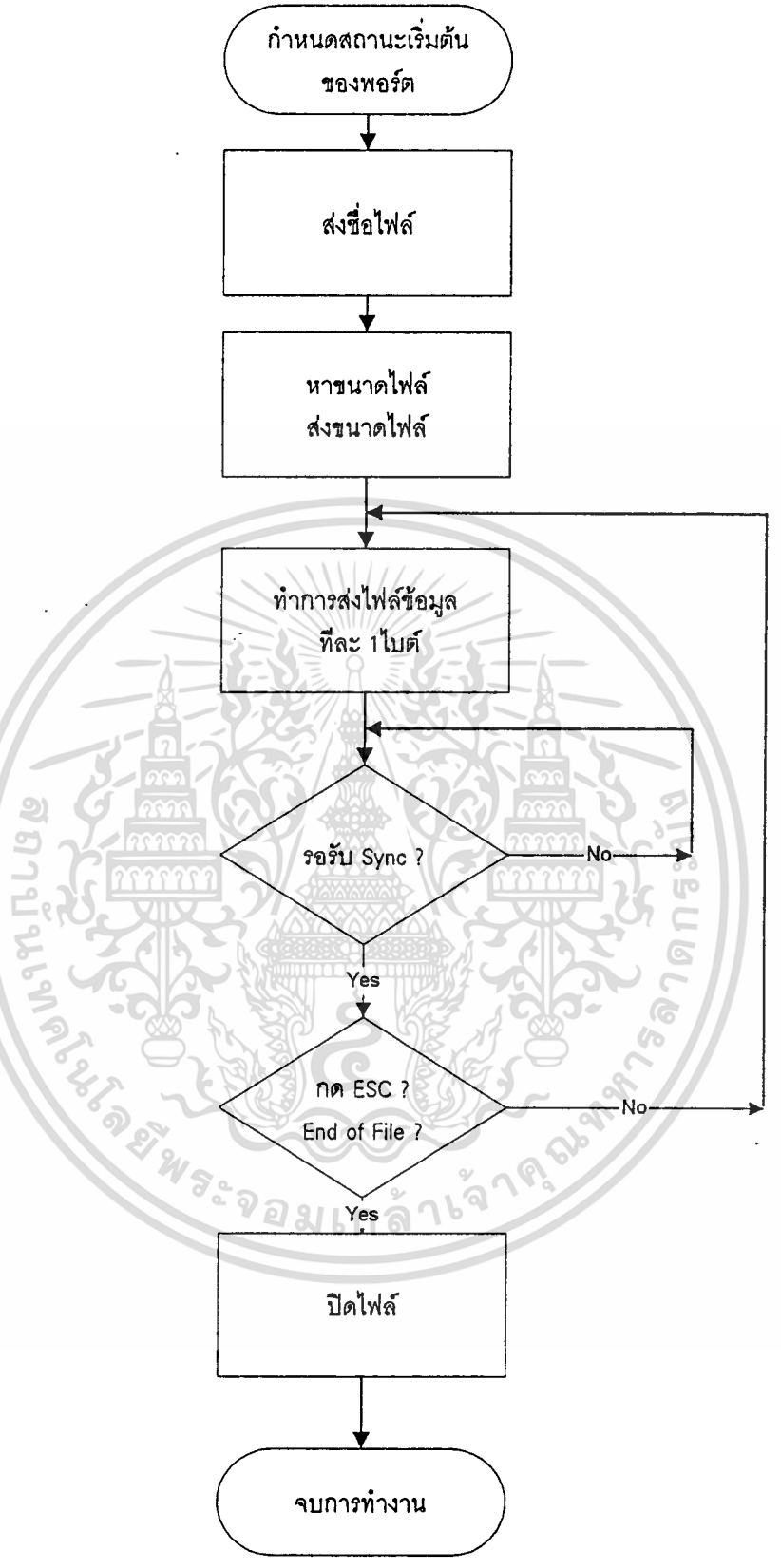
รูปที่4.2 แสดงขั้นตอนการแสดงผลไฟล์ภาพPCX



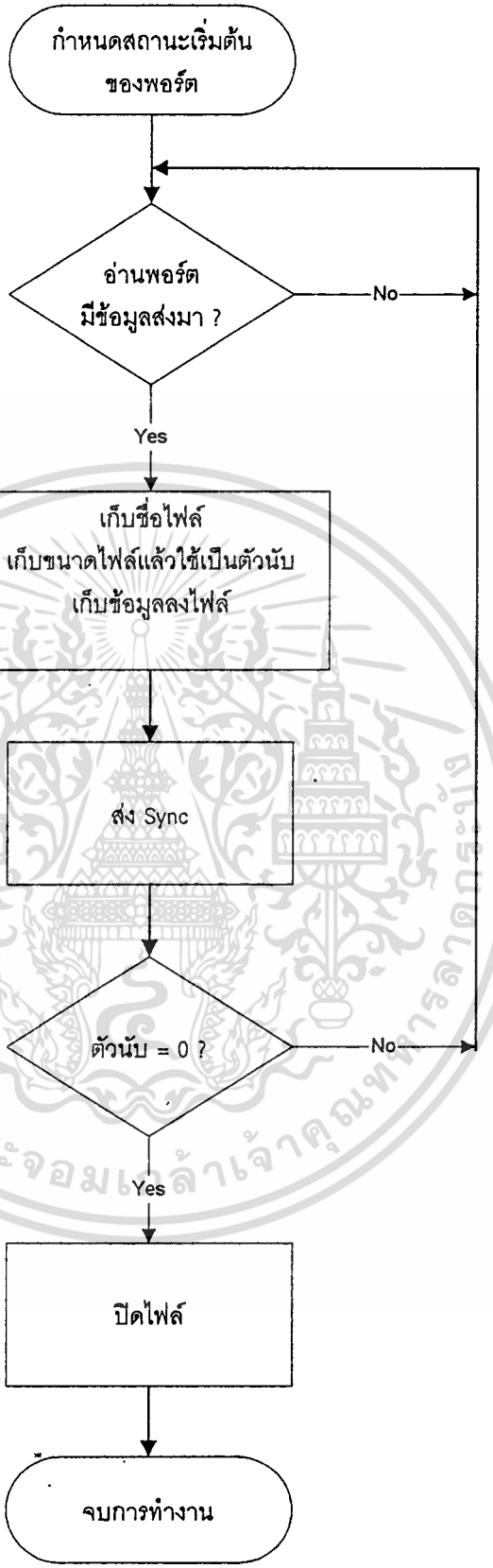
รูปที่4.3 แสดงขั้นตอนการบันทึกเสียง



รูปที่4.4 แสดงขั้นตอนการเล่นไฟล์เสียง



รูปที่4.5 แสดงขั้นตอนการส่งข้อมูล



รูปที่4.6 แสดงขั้นตอนการรับไฟล์ข้อมูล

บทที่ 5
การทดลองและผลการทดลอง

การทดลองและผลการทดลอง

ในปริญญานิพนธ์นี้สิ่งที่ต้องการ คือความสามารถในการสื่อสารข้อมูลผ่านพอร์ตอนุกรมด้วยสื่อหลายอย่าง เราจึงได้ทำการทดลองเขียนโปรแกรมแยกเป็นส่วน ๆ ดังนี้

1. โปรแกรมการสื่อสาร 2 ทิศทาง
2. โปรแกรมการสื่อสารด้วยภาพประกอบข้อความ
3. โปรแกรมการสื่อสารด้วยเสียง

1. โปรแกรมการสื่อสาร 2 ทิศทาง

1.1 การทำงาน จะเริ่มด้วยการกำหนดสถานะเริ่มต้นของพอร์ตอนุกรม เช็คว่ามีการกดคีย์หรือไม่ ถ้ามีจะทำการส่งตัวอักษรที่คีย์ไปยังด้านรับ ถ้าไม่มีก็จะทำการเช็คสถานะของพอร์ตว่ามีข้อมูลส่งมาหรือไม่ หากมีจะทำการอ่านพอร์ต แล้วนำข้อมูลนั้นมาแสดงผล ซึ่งจะแสดงผลทั้งข้อความของผู้ส่งและข้อความที่ได้รับมา

1.2 ผลการทดลอง สามารถรับส่งข้อความที่โต้ตอบกันได้ถูกต้อง

2. โปรแกรมการสื่อสารด้วยภาพประกอบข้อความ

2.1 การทำงาน ทำการส่งไฟล์ภาพ PCX จากด้านส่งไปยังด้านรับ แล้วด้านรับจะทำการเรียกการทำงานในโหมดกราฟฟิกถอดรหัสไฟล์ภาพก่อนจะนำภาพมาแสดงผล แล้วยังไม่ทำการ closegraph ( ) แต่จะทำการเรียกส่วนการสื่อสารแบบ 2 ทิศทางขึ้นมาใช้ทำการติดต่อกันต่อไป ซึ่งทั้งหมดนี้ต่างทำงานในโหมดการแสดงผลจอภาพโหมดเดียวกัน ดังนั้นส่วนการสื่อสารแบบ 2 ทิศทางนี้จะถูกเขียนขึ้นเพื่อใช้งานในโหมดกราฟฟิก

2.2 ผลการทดลอง สามารถส่งภาพที่ต้องการใช้ประกอบการสนทนาไปให้ และแสดงที่ฝ่ายรับได้ จากนั้นก็ทำการโต้ตอบข้อความกันได้จริง แต่ภาพที่ส่งต้องมีขนาดเล็ก หรือความละเอียด 320 * 200 เพื่อประหยัดเนื้อที่ในการแสดงผล แต่การส่งภาพยังทำได้เพียง 1 ภาพต่อการสนทนา 1 ครั้งเท่านั้น คือตอนเริ่มต้น

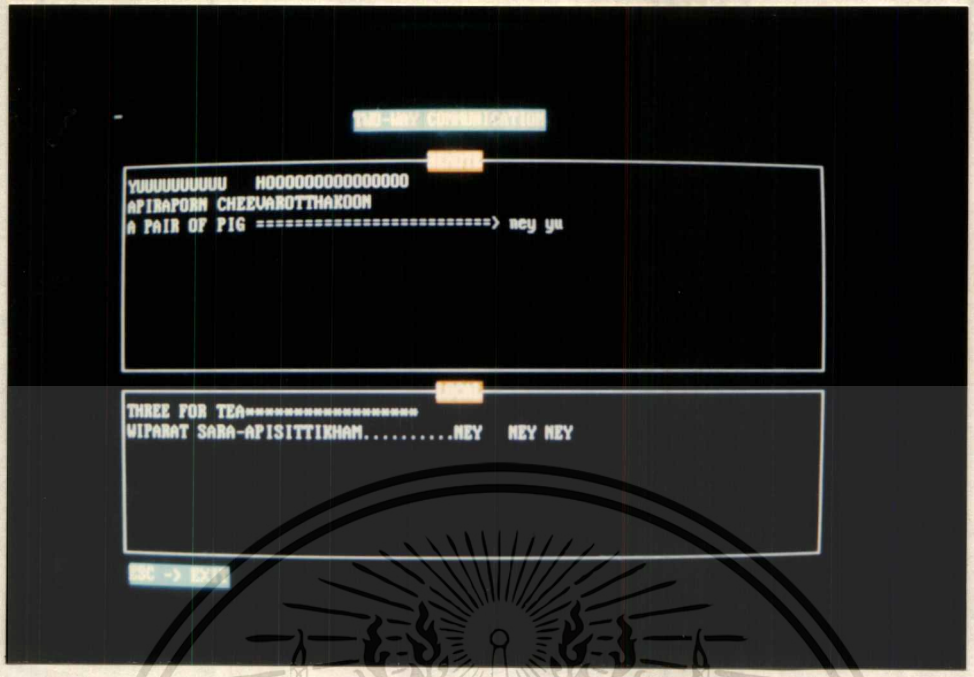
3. โปรแกรมการสื่อสารด้วยเสียง

3.1 การทำงาน แบ่งเป็นส่วนการบันทึกเสียง โดยเริ่มที่การเรียกใช้ฟังก์ชันกำหนดค่าเริ่มต้นโหลดไดร์เวอร์ CT - VOICE แล้วทำการกำหนดเวลาที่จะบันทึกเสียงเพื่อจองบัฟเฟอร์ บันทึกเสียงผ่านไมโครโฟน โดยเรียกใช้ฟังก์ชันบันทึกข้อมูล (BX = 7) จากฟังก์ชันใช้งานของไดร์เวอร์ CT - VOICE แล้วเก็บลงไฟล์ จากนั้นทำการส่งไฟล์เสียงนี้ไปยังฝ่ายรับ

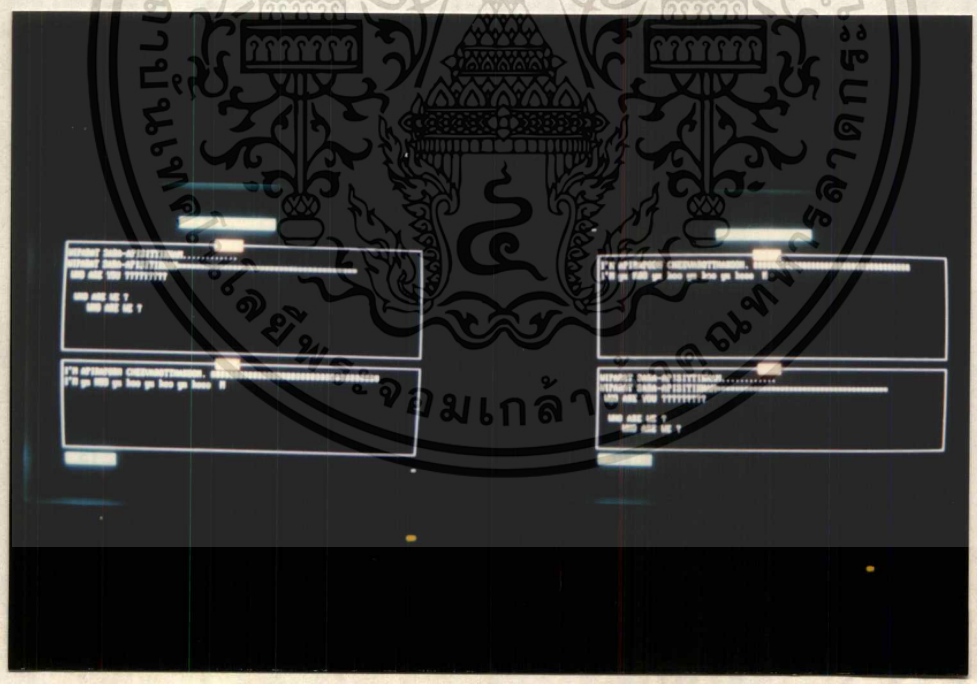
เมื่อฝ่ายรับเรียกใช้ส่วนของการเล่นไฟล์เสียง ซึ่งก็จะทำการกำหนดสถานะเริ่มต้น, กำหนดชื่อไฟล์ที่ต้องการ แล้วทำการโหลดลงบัฟเฟอร์เพื่อทำการเล่นกลับไฟล์เสียง โดยเรียกใช้ฟังก์ชันเล่นกลับ (BX = 6) และสามารถเลือกการเล่นกลับแบบครั้งเดียวหรือหลายครั้งก็ได้

3.2 ผลการทดลอง สามารถบันทึกเสียงได้ที่ความถี่ในการสุ่ม 5000 เฮิร์ต , ขนาดในการสุ่มสัญญาณ 8 บิต และบันทึกเสียงแบบโมโน ในการทดลองสามารถบันทึกเสียงได้สูงสุดที่ 90 วินาที (ขนาดของข้อมูลเท่ากับ 450,000 ไบต์) เพราะมีปัญหาในเรื่องการจองบัฟเฟอร์

ส่วนการเล่นไฟล์เสียง สามารถรับฟังเสียงได้ชัดเจนดี และเลือกได้ว่าจะเล่นครั้งเดียวหรือหลายครั้ง หากเป็นการเล่นกลับครั้งเดียวก็จะเล่นครั้งเดียวแล้วรอให้กดคีย์เพื่อทำการยกเลิก แต่ถ้าเป็นการเล่นแบบหลายครั้งจะเล่นไปเรื่อย ๆ จนกว่าจะกดคีย์เพื่อทำการหยุด แต่ถ้าหากไฟล์มีขนาดใหญ่จะไม่สามารถทำการจองบัฟเฟอร์ได้ ทำให้เล่นกลับไฟล์เสียงไม่ได้

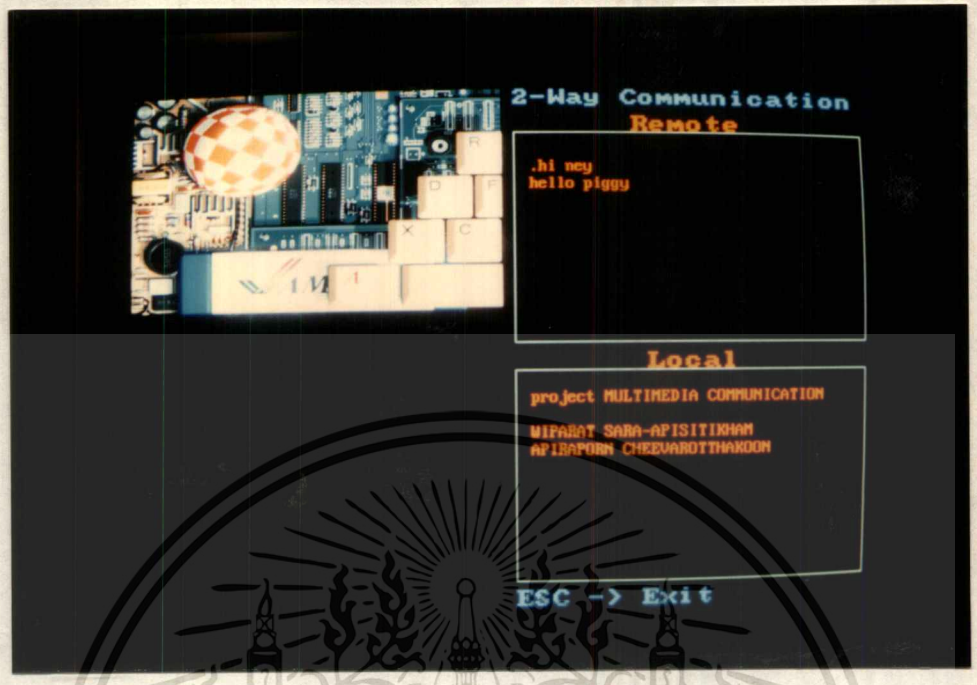


ก) แสดงการแสดงผลของโปรแกรม

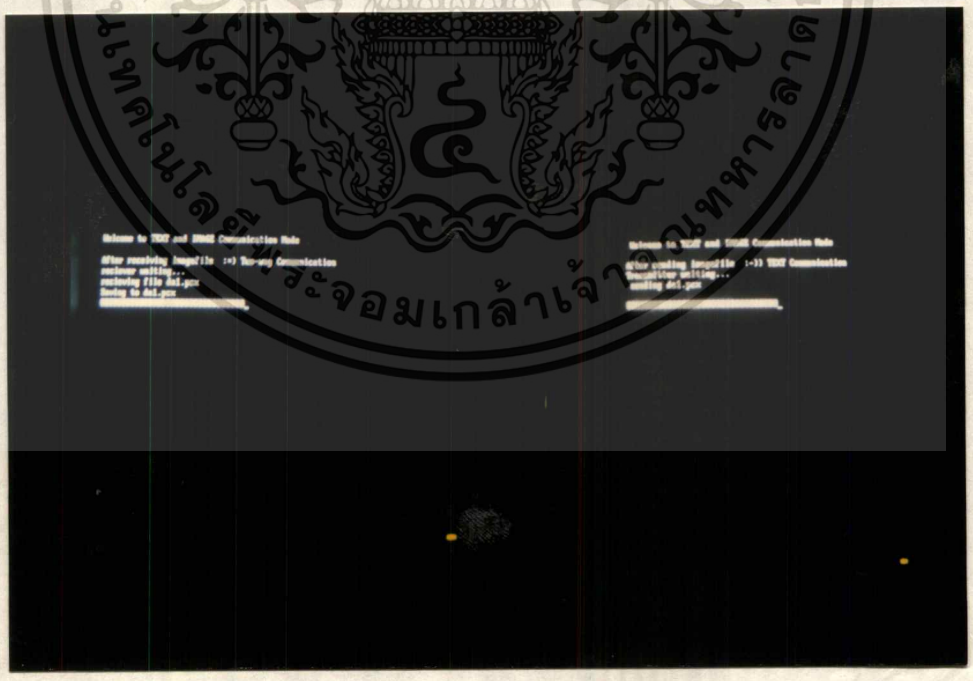


ข) แสดงภาพขณะทำการติดต่อสื่อสาร

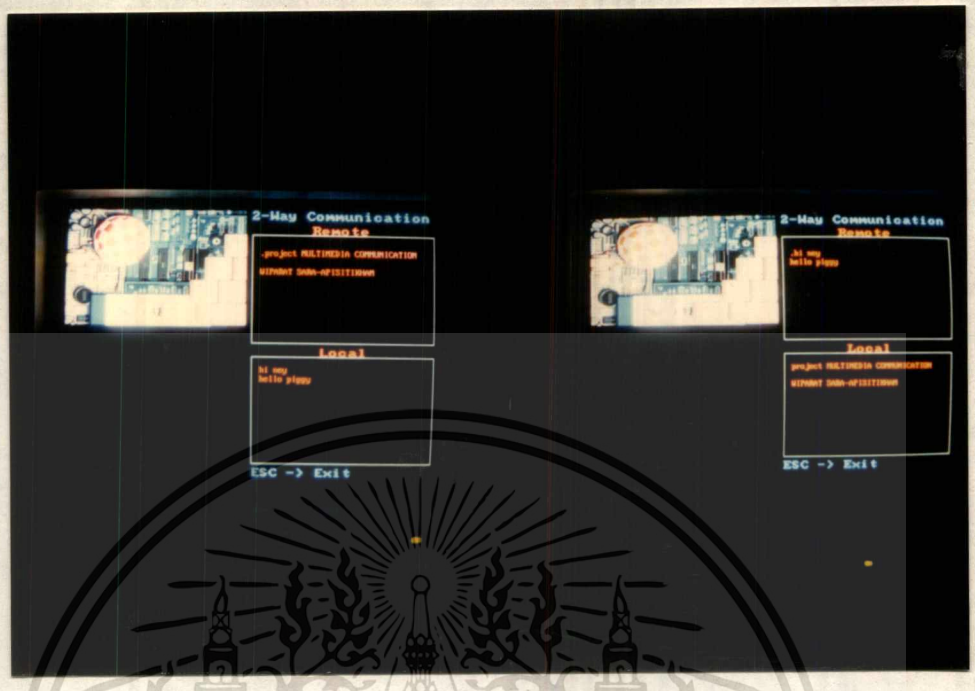
รูปที่ 5.1 แสดงผลการการทำงานของโปรแกรมการสื่อสารแบบ 2 ทิศทาง



ก). แสดงลักษณะการแสดงผลของโปรแกรม



ข). แสดงภาพขณะทำการส่งไฟล์ภาพ da.pcx



ค). แสดงภาพขณะทำการสื่อสาร

รูปที่ 5.2 แสดงผลการทำงานของโปรแกมการสื่อสารด้วยภาพประกอบข้อความ

บทที่ 6
สรุปผลการทดลอง , ปัญหา และแนวทางพัฒนา

6.1 สรุปผลการทดลอง และ ปัญหา

โปรแกรมที่ใช้ให้ผลการทดลองเป็นที่น่าพอใจพอสมควรตรงตามจุดประสงค์ คือ หากต้องการติดต่อสื่อสารเฉพาะข้อความก็เรียกใช้โปรแกรมการสื่อสาร 2 ทิศทาง ถ้าต้องการสื่อสารด้วยข้อความแต่มีภาพประกอบเพื่อให้การสื่อสารน่าสนใจ และสื่อความหมายได้มากขึ้น ก็เลือกใช้โปรแกรมการสื่อสารด้วยภาพประกอบข้อความ แต่ถ้าต้องการฝากข่าวสารเป็นเสียงไปยังผู้รับก็เลือกใช้โปรแกรมการสื่อสารด้วยเสียง แต่โปรแกรมเหล่านี้ก็ยังมีข้อจำกัดและปัญหาอยู่มาก เช่น

1. สามารถสื่อสารด้วยไฟล์ภาพ PCX เท่านั้น ควรศึกษา และนำไฟล์ภาพแบบอื่นมาใช้ร่วมด้วย
2. สามารถส่งภาพได้เพียง 1 ภาพ ต่อการติดต่อด้วยข้อความ 1 ครั้ง
3. การบันทึกเสียงมีข้อจำกัดด้านเวลา เนื่องจากจองบัฟเฟอร์ไม่ได้มากเพียงพอ
4. การส่งไฟล์ยังมีความล่าช้าทั้งไฟล์ภาพ และไฟล์เสียง เนื่องจากไฟล์ทั้งสองแบบมีขนาดใหญ่ จึงควรหาวิธีลดขนาดของข้อมูลให้น้อยกว่านี้
5. ข้อจำกัดของพอร์ต RS - 232 เรื่องของความเร็วในการส่ง, บัฟเฟอร์ที่ใช้ในการรับ - ส่งข้อมูล ซึ่งแก้ไขให้ดีขึ้นได้ด้วยวิธีการที่ได้กล่าวไว้ในบทที่ 4 แล้ว

6.2 แนวทางพัฒนา

การพัฒนาโปรแกรมเหล่านี้ให้สามารถทำการติดต่อสื่อสารได้ดียิ่งขึ้น จะทำได้โดยการลดข้อจำกัดต่าง ๆ เช่น

- การเพิ่มความเร็ว ทำได้โดยนำกรรมวิธีการลดขนาดข้อมูลมาใช้ทำให้ข้อมูลมีขนาดเล็กลงสามารถทำการส่งได้เร็วขึ้นและง่ายต่อการเก็บหรือหาวิธีลดความล่าช้าที่เกิดจากข้อจำกัดของพอร์ตRS-232
- เพิ่มแอพพลิเคชันในการใช้งาน โดยทำให้โปรแกรมสามารถแสดงภาพได้หลายภาพ และหลายรูปแบบการจัดเก็บในการติดต่อแบบข้อความแต่ละครั้ง หรือพัฒนาให้สามารถใช้กับภาพเคลื่อนไหวได้
- นอกจากนี้อาจนำโปรแกรมฝังตัวมาใช้ เพื่อจะได้ไม่เสียเวลาในการวนเช็คพอร์ตว่ามีข้อมูลส่งมาหรือไม่ สามารถใช้งานโปรแกรมต่าง ๆ ได้จนกว่าจะมีอินเตอร์รัพต์เข้ามา

ภาคผนวก

```
/********** TWO-WAY PROGRAME**********/

#include <dos.h>
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#define UL 218
#define UR 191
#define DL 192
#define DR 217
#define H  196
#define V  179
#define COM1 0
#define BAUD 231
/*8 Data Bit 2 Stop Bit No Parity*/
void init_port();
void clear();

main()
{
      int stat,y=1,x=1,x1=1,y1=1;
      init_port(COM1,BAUD);
      clear(COM1);
      clrscr();
      /* CREATE REMOTE AREA AND LOCAL AREA*/
      gotoxy(27,2);
      textcolor(15);
      textbackground(3);
      cprintf("TWO-WAY COMMUNICATION");
      normvideo();
      box(2,4,78,14);
      gotoxy(35,4);
      textcolor(15);
      textbackground(6);
      cprintf("REMOTE");
      normvideo();
      box(2,15,78,23);
      gotoxy(36,15);
      textcolor(15);
      textbackground(6);
      cprintf("LOCAL");
      normvideo();
      gotoxy(3,24);
      textcolor(15);
      textbackground(3);
      cprintf("ESC -> EXIT");
      normvideo();
      for(;;)     {
      if(kbhit())
            {
            x1=send(x1, y1);
            switch(x1) {
               case 75:send(x1,y1);
                     x1=1;
                     y1=y1++;
                     if (y1==8)
                     y1=7;
                     break;
               case 0: x1=1;
                     y1=y1++;
```

```
                        break;
                        }
                }
        else
            {
            if (check_stat(COM1) & 0x100)
                {
                x=rec(x,y);
                switch(x) {
                      case 75:rec(x,y);
                            x=1;
                            y=y++;
                            if (y==10)
                            y=9;
                            break;
                      case 0: x=1;
                            y=y++;
                            break;
                        }
                }
              }
        }
}

rec(int x,int y)
{
        int ch;
        ch=read_port(COM1);
        window(3,5,77,13);
        gotoxy(x,y);
        cprintf("%c",ch);
        x=wherex();
        if((ch==0xA)&&(y==9))
            {
            movetext(3,6,77,13,3,6);
            gotoxy(1,wherey());
            clreol();
            x=1;
            }
        else if((ch==0xA)&&(y<9))
              {
              x=0;
              y=y++;
              }
        window(1,1,80,25);
        return x;
}

send(int x,int y)
{
        int ch;
        window(3,16,77,22);
        gotoxy(x,y);
        ch=getche();
        x=wherex();
        if((ch==13)&&(y==7))
            {
            movetext(3,17,77,22,3,17);
            gotoxy(1,wherey());
            clreol();
```

```
            x=1;
            }
        else if((ch==13)&&(y<7))
                {
                x=0;
                y=y++;
                }
        window(1,1,80,25);
        switch(ch) {
            case 27:exit(0);
            case 13:send_port(COM1,0xA);
            /*Line feed*/
            break;
            default:send_port(COM1,ch);
                    }
        return x;
}

void init_port(int com,int code)
{
        union REGS r;
        r.x.dx=com;
        r.h.ah=0;
        r.h.al=code;
        int86(0x14,&r,&r);
}

read_port(int com)
{
        union REGS r;
        r.x.dx=com;
        r.h.ah=2;
        /*while(!(check_stat(COM2) & 0x100))
        {
                if(kbhit())
                if(getch()==27)
                exit(0);
        }*/
        int86(0x14,&r,&r);
        if(r.h.ah & 0x80)
            {
            gotoxy(64,24);
            textcolor(1);
            textbackground(14);
            cprintf("ERROR DETECTED");
            normvideo();
            }
        else
        return r.h.al;
}

check_stat(int com)
{
        union REGS r;
        r.x.dx=com;
        r.h.ah=3;
        int86(0x14,&r,&r);
        return r.x.ax;
}
```

```
void clear(int com)
{
      union REGS r;
      r.x.dx=com;
      r.h.ah=2;
      int86(0x14,&r,&r);
}

box(int x1,int y1,int x2,int y2)
{
      int k;
      gotoxy(x1,y1);
      putch(UL);
      for(k=1;k<(x2-x1);k++)  {
      gotoxy(x1+k,y1);
      putch(H);
      }
      gotoxy(x2,y1);
      putch(UR);
      for(k=1;k<(y2-y1);k++)  {
      gotoxy(x1,y1+k);
      putch(V);
      gotoxy(x2,y1+k);
      putch(V);
      }
      gotoxy(x1,y2);
      putch(DL);
      for(k=1;k<(x2-x1);k++)  {
      gotoxy(x1+k,y2);
      putch(H);
      }
      gotoxy(x2,y2);
      putch(DR);
      }

send_port(int port,int ch)
{
      union REGS r;
      r.x.dx=port;
      r.h.ah=1;
      r.h.al=ch;
      int86(0x14,&r,&r);
      if(r.h.ah & 0x80) {
      gotoxy(64,24);
      textcolor(1);
      textbackground(14);
      cprintf("ERROR DETECTED");
      normvideo();
      }
      else        {
      gotoxy(64,24);
      cprintf("               ");
      }
      }
```

```
/* TWO- WAY COMMUNICATION WITH PCX IMAGE PROGRAM*/

#include <graphics.h>
#include <dos.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <process.h>
#include <graphics.h>
#include <bios.h>
#include <conio.h>
#include "svga256.h"
#include "twk256.h"
#define COM1 0
#define BAUD 231   /* Initial port :baud rat,word,stop bit,parity bit*/
#define LOCLEFT   44   /* Initial area of LOCAL & REMOTE */
#define LOCRIGHT  78
#define LOCUP     18
#define LOCDOWN   25
#define REMLEFT   44
#define REMRIGHT  78
#define REMUP      5
#define REMDOWN   16

void setvgapalette256(DacPalette256 *PalBuf);
int  huge DetectVGA256(void);
void Setmode256();
void setMode(int mode);
void plot256(int x,int y,int color);
void setEGApalette(int palette,int color);
void setVGApalette(unsigned char *buffer);
void init_port();
void clear();
void send_filename(char *filename);
void send_file(char *filename);
void send_file();
void rec_file();
void wait();
void twoway();
unsigned long filesize(FILE  *fp);

 double XMax,XMin,YMax,YMin,PVal,QVal;
 char file_name[16];
 unsigned char PALETTE[16];
 union REGS reg;
 struct SREGS inreg;

typedef       struct {       /* Structure of PCX header */
      char manufacturer;
      char version;
      char encoding;
      char bits_per_pixel;
      int  xmin,ymin;
      int  xmax,ymax;
      int  hres;
      int  vres;
      char __palette[48];
      char reserved;
      char color_planes;
           int  bytes_per_line;
```

```
            int  palette_type;
            char filler[58];
      } PCXHEAD;
PCXHEAD   header;

 char  ch;
 unsigned char palette[768];
 unsigned char picture[3000];
 int    i,j,depth,width,xoff,yoff,x;
 int    picx,picy,c,n,bytes;
 int    mode = 2;
 FILE   *imagefile;

char  fname[40]="da1.pcx";
unsigned char PALETTE[16];
struct SREGS inreg;

main(argc,argv)
int argc;
char *argv[];
{
      clrscr();

      if(argc<2) {
            printf("Usage : tran s filename or trans r\n");
            exit(1);
      }
      printf("File transfer program in operation. To abort, \n");
      printf("press any key. \n\n");

      init_port(COM1,BAUD);
      clear(COM1);

      if(tolower(*argv[1])=='s') send_file(argv[2]);  /* Send file */
      else rec_file();                    /* Else for receive file */

      Setmode256();                 /* Set graphic mode (svga256) */
      viewpcx();                    /* Decode PCX file coding...display */
      twoway();
}

/*-----------------------------------------------*/
/*     Two-way communication in graphic mode     */
/*-----------------------------------------------*/

void twoway()
{
      int x1,y1,x2,y2;
      /*Create LOCAL & REMOTE area */
      setfillpattern(EMPTY_FILL,1);
      settextstyle(DEFAULT_FONT,HORIZ_DIR,2);
      setcolor(14);
      rectangle(330,40,639,230);
      rectangle(330,255,639,445);
      setcolor(10);
      outtextxy(330,0,"2-Way Communication");
      outtextxy(330,454,"ESC -> Exit");
      setcolor(7);
      outtextxy(435,25,"Remote");
      outtextxy(445,240,"Local");
```

```
      x1=LOCLEFT;y1=LOCUP;
      x2=REMLEFT;y2=REMUP;

      /* Read key.....SEND */
      /* Read port....RECEIVE */
      for(;;)     {
                if(kbhit())     {
            x1=send(x1, y1);
            switch(x1)  {
            case LOCRIGHT:  send(x1,y1);
                        x1=LOCLEFT;
                        y1=y1++;
                        if (y1==LOCDOWN)
                        y1=LOCDOWN-1;
                        break;
            case LOCLEFT-1: x1=LOCLEFT;
                        y1=y1++;
                        break;
                        }
                  }
         else{
            if (check_stat(COM1) & 0x100) /* Check port status to receive
*/
               {
               x2=rec(x2,y2);
               switch(x2)          {
                  case REMRIGHT: rec(x2,y2);
                                 x2=REMLEFT;
                                 y2=y2++;
                                   if (y2==REMDOWN)
                                 y2=REMDOWN-1;
                                 break;
                  case REMLEFT-1:x2=REMLEFT;
                                  y2=y2++;
                                  break;
                                  }
             }

            }
      }
}
/* Receive data and display */
rec(int x3,int y3)
{
      int ch;
      ch=rport(COM1);
      gotoxy(x3,y3);
      setcolor(15);
      printf("%c",ch);
      x3=wherex();
      if((ch==0xA)&&(y3==REMDOWN))
         {
         x3=REMLEFT;
         y3=REMUP;
         gotoxy(x3,y3);
         textcolor(0);
         clreol();
         }
      else  if((ch==0xA)&&(y3<REMDOWN)) {
                  x3=REMLEFT-1;
```

```
                y3=y3++;
            }
        return x3;
}

/* Send data */
send(int x,int y)
{
        int ch;
        gotoxy(x,y);
        ch=getch();
        cprintf("%c",ch);
        x=wherex();
        if((ch==13)&&(y==LOCDOWN))
        {
           x=LOCLEFT;
           y=LOCUP;
           gotoxy(x,y);
           textcolor(0);
           clreol();
        }
        else   if((ch==13)&&(y<LOCDOWN))
                  {
                  x=LOCLEFT-1;
                  y=y++;
                  }
        switch(ch)  {
           case 27:exit(0);
           case 13:sport(COM1,0xA);
           /*Line feed*/
           break;
           default:sport(COM1,ch);
        }
        return x;
}

/* Initial port */
void init_port(int com,int code)
{
        union REGS r;
        r.x.dx=com;
        r.h.ah=0;
        r.h.al=code;
        int86(0x14,&r,&r);
}

/* Read data from port */
rport(int com)
{
        union REGS r;
        r.x.dx=com;
        r.h.ah=2;
        /*while(!(check_stat(COM2) & 0x100))  {
        if(kbhit())
        if(getch()==27)
        exit(0);
        }*/
        int86(0x14,&r,&r);
        if(r.h.ah & 0x80) {
           printf("READ ERROR DETECTED IN SERIAL PORT\n");
```

```
        }
        else
        return r.h.al;
}

/* Send data out port */
sport(int port,int ch)
{
        union REGS r;
        r.x.dx=port;
        r.h.ah=1;
        r.h.al=ch;
        int86(0x14,&r,&r);
        if(r.h.ah & 0x80){
          printf("SEND ERROR DETECTED IN SERIAL PORT\n");
        }
}
/* Check port status */
check_stat(int com)
{
        union REGS r;
        r.x.dx=com;
        r.h.ah=3;
        int86(0x14,&r,&r);
        return r.x.ax;
}

/* Clear register buffer */
void clear(int com)
{
        union REGS r;
        r.x.dx=com;
        r.h.ah=2;
        int86(0x14,&r,&r);
}


/* ---------------------------------------*/
/*   Function of PCX decoding and viewimage  */
/*---------------------------------------*/

void cls(int color)
{
 union REGS r;
 r.x.ax=0x0600;
 r.x.cx=0;
 r.x.dx=0x184F;
 r.h.bh=color;
 int86(0x10,&r,&r);
}


void setEGApalette(int palette,int color)
{
  union REGS r;
  PALETTE[palette]=color;
  r.h.ah=0x01;
  r.h.al=0;
  r.h.bh=color;
  r.h.bl=palette;
```

```
 int86(0x10,&r,&r);
}

void setVGApalette(unsigned char *buffer)
{
union REGS r;
r.x.ax=0x1012;
segread(&inreg);
inreg.es=inreg.ds;
r.x.bx=0;
r.x.cx=256;
r.x.dx=(int)&buffer[0];
int86x(0x10,&r,&r,&inreg);
}

int huge DetectVGA256(void)
{
  return(mode);
}

void Setmode256()
{
  unsigned char palbuf[256][3];
  int Gd = DETECT, Gm, i,j,k;
  installuserdriver("Svga256",DetectVGA256);
  initgraph(&Gd,&Gm,"");

}

void setvgapalette256(DacPalette256 *PalBuf)
{ struct REGPACK reg;
  reg.r_ax = 0x1012;
  reg.r_bx = 0;
  reg.r_cx = 256;
  reg.r_es = FP_SEG(PalBuf);
  reg.r_dx = FP_OFF(PalBuf);
  intr(0x10,&reg);
}

/*-------------------------------------------*/
/*       file Sending &  file Receiving      */
/*-------------------------------------------*/

/* Function : Sendfile */
void send_file(char *fname)
{
       FILE *fp;
       char ch;
       int n;
       union {
                  char c[4];
                  unsigned long count;
                }cnt;

      if (!(fp =fopen(fname,"rb")))     /* Open selected file */
              {
              printf("CANNOT OPEN INPUT FILE\n");
              exit(1);
              }
       printf("\n\n Welcome to TEXT and IMAGE Communication Mode\n\n");
```

```
        printf("After sending imagefile  :-)) TEXT Communication\n");


        send_filename(fname);              /*Send file name */
        wait(COM1);
        cnt.count = filesize(fp);          /* Find filesize */
        sport(COM1,cnt.c[0]);              /* Send filesize */
        wait(COM1);
        sport(COM1,cnt.c[1]);
        wait(COM1);
        sport(COM1,cnt.c[2]);
        wait(COM1);
        sport(COM1,cnt.c[3]);

        n=0;
        do                                 /* Send file : each byte */
                {
                ch = getc(fp);
                if(ferror(fp))
                  {
                  printf("ERROR READING INPUT FILE\n");
                  break;
                  }
                if(!feof(fp))
                  {
                  wait(COM1);              /* Wait for synccode */
                  sport(COM1,ch);
                  }
                  n++;
                  if(n==100){
                  printf("\002");          /*Print show sending step */
                  n=0;}
                } while (!feof(fp));
                sport(COM1,'.');
                printf("\nSuccess of sending: ");
                fclose(fp);
        }                                  /* End of sending */


/* Function : Send file */
void rec_file()
{
        FILE *fp;
        int a;
        char ch;
        union {
                char c[4];
                unsigned long count;
        }cnt;

        get_file_name(fname);              /* Receive filename */
        printf("recieving file %s\n",fname);
        remove(fname);
        if(!(fp=fopen(fname,"wb"))) {
                printf("CANNOT OPEN OUTPUT FILE\n");
                exit(1);
        }
        printf("Saving to %s\n",fname);
        sport(COM1,'.');
        cnt.c[0]=rport(COM1);              /* Receive filesize */
```

```
        sport(COM1,'.');
        cnt.c[1]=rport(COM1);
        sport(COM1,'.');
        cnt.c[2]=rport(COM1);
        sport(COM1,'.');
        cnt.c[3]=rport(COM1);
        sport(COM1,'.');

        a=0;

        for(;cnt.count>0;cnt.count--) {  /* Filesize is a count */
              ch=rport(COM1);           /* Receive data */
              putc(ch,fp);
              a++;
              if(ferror(fp)) {
                    printf("ERROR WRITING FILE\n");
                    exit(1);

              }

              sport(COM1,'.');          /* Send  synccode */
              if( bioskey(1) )
               {
                 printf("Quit...");
                 exit(1);
               }

              if(a==500){               /* Print show receiving step */
              printf("\002");
              a=0;}
        }
        printf("\nReceiving successful.");
        fclose(fp);
}                                       /* End of receiving */

/* Start file sending with filename sending */
void send_filename(f)
char *f;
{   int m;

        m=0;
        printf("Transmitter waiting...\n");
        clear(COM1);
        do {
                                  /* Loop send synccode */
              sport(COM1,'?');          /* & wait for respond */
              m++;
              if(m==5000){
              printf("\nNO RESPOND FROM RECEIVER.!!!");
              printf("\n      Y for CANCEL Transmitting ");
              printf("\n      N for WAIT\n");
              switch(getch()){
                 case 'y':exit(1);
                 case 'Y':exit(1);
                 case 'n':m=0;
                 case 'N':m=0;}
              }

        } while(!(kbhit()) && !(check_stat(COM1) & 256));
        if(kbhit())
```

```
        {

            printf("Quit..");
            exit(1);
            }
      wait(COM1);
      printf(" sending %s\n\n",f);
      while(*f) {                          /* Send filename */
            sport(COM1,*f++);
            wait(COM1);
      }
      sport(COM1,'\0');
}

/* Start file receiving with receive filename */
get_file_name(f)
char *f;
{
      printf("reciever waiting...\n");
      while(rport(COM1)!='?');
      sport(COM1,'.');
      while((*f=rport(COM1))) {
            if(*f!='?') {
                  f++;
                  sport(COM1,'.');
            }
      }
}

/* Find filesize */
unsigned long filesize(fp)
FILE *fp;
{
      unsigned long int length;
      length = filelength(fileno(fp));
      rewind(fp);
      return(length);
}

/* Wait for synccode */
void wait(port)
int port;
{
      if(rport(port)!='.') {
            printf("   COMMUNICATION ERROR\n");
            exit(1);
      }
}
/*-----------------------------------------*/
/*       Decode & Display PCX file         */
/*-----------------------------------------*/

viewpcx()
 {


 if( (imagefile = fopen(fname,"rb")) == NULL)
  {
   printf("Error reading %s",fname);
   exit(1);
```

```
}

fread((char *)&header,1,sizeof(PCXHEAD),imagefile);

width = header.xmax - header.xmin + 1;
depth = header.ymax - header.ymin + 1;
bytes = header.bytes_per_line;

picx = width;
picy = depth;

fseek(imagefile,-769L,SEEK_END);
c = fgetc(imagefile);
fread(palette,1,768,imagefile);

for(i=0;i<768;i++)                          /* Pallate */
 {
  palette[i] = palette[i] >> 2;
 }

fseek(imagefile,128L,SEEK_SET);

Setmode256();
setVGApalette(palette);

xoff = 0; yoff = 0;

for (i=0;i<depth;i++)
    {                                       /* Decode PCX coding */
        n = 0;
        do{
         c = fgetc(imagefile) &0xff;
         if((c & 0xc0) == 0xc0)
           {
             j = c & 0x3f;
             c = fgetc(imagefile);
             while(j--)
               {
               picture[n] = (unsigned char)c;
               n++;
               }
           }
         else
           {
             picture[n] = (unsigned char)c;
             n++;
           }
         }while(n<bytes);

      for(x = 0;x<(picx-1);x++)       /* Display */
       {
           putpixel(x+xoff,i+yoff,picture[x]);
       }
}

fclose(imagefile);

}                                           /* End of display PCX file */.
```

```
/**************** RECORD PROGRAM ************/

#include "voctool.h"
#include <stdio.h>
#include <dos.h>
#include <conio.h>
#include <stdlib.h>
#include <alloc.h>

void record (unsigned char huge*buf,unsigned long int count);

void main()
{
      unsigned char huge*data;
      unsigned char *tail;
      unsigned long int count,number;
      unsigned int n;
      char string[8];
      char filename[40];
      FILE *stream;

      clrscr();

      printf("Program for record sound(.VOC).\n");
      printf("Enter filename : ");
      gets(filename);
      printf("Enter the recording time (second) : ");
      gets(string);

      number = atoi(string);
      count = (number*5000);
      if((data= farmalloc (count)) == NULL)
         {
         printf("CAN'T ALLOCATE MEMORY (data).");
         }
         /* record data*/
      record(data,count);
      if((stream = fopen(filename,"wb")) == NULL)
         {
         printf("\nCAN'T OPEN THIS FILE: %s",filename);
         exit(0);
         }
      fprintf(stream, "Creative Voice File \x1A%c\x0A\x01\x29\x11", 0);
      n=fwrite(data,5000,(count/5000),stream);
      printf("\nnumber of item : %d",n);
      fclose(stream);
      printf("Save file success !!");
      getch();
}

void record(unsigned char huge*buf,unsigned long int count)
{
      unsigned char huge*voc_buffer=0;
      char far*old_voc_ptr;
      char ch;

      if(_voc_init_driver() == FALSE) _textnumerror();
      printf("\nPress r to record sample :  ");
      ch = getche();
      if(ch == 'r')
```

```
        {
        printf("\nPress any key to stop recording.");
        _voc_set_speaker(FALSE);
        _voc_set_filter_r();
        voc_buffer = buf;
        if(voc_buffer == NULL) _textnumerror();
        _voc_input(voc_buffer,count);
        do{}
        while((voc_status_word != 0)&& (kbhit()==0));
        _voc_stop();
        }
/*farfree(voc_buffer);*/
      _voc_deinstall_driver();
}
```

```
/********** PLAYBACK PROGRAME *********/

#include <conio.h>
#include "voctool.h"

void main()
{
  VOCPTR voc_buffer = 0;
  char   ch;
  char vocfile[40];

  clrscr();
  if (_voc_init_driver() == FALSE) _textnumerror();

/* Main for playback */
  printf("          CT-Voice Driver Version : ");
  printf("          %d.%02d\n",vocdriver_version >> 8, vocdriver_version
% 256);
  printf("          PLAY PLAY PLAY sampling sound..-> \n\n");
  printf("          Enter VOCfile    : ");
  gets(vocfile);

 /* Loads VOC file into memory */
  voc_buffer = _voc_get_buffer(vocfile);
/
* VOC file could not be loaded */
  if (voc_buffer == NULL)_textnumerror();

  printf("          (S)ingle play or (M)ultiple play?\n");
  printf("           Press a key... only 's' or 'm': ");
  ch = getche();
  printf("\n\n");
  switch (ch) {
       case 's': printf("                Stop...->Press a key ");
                 _voc_output(voc_buffer);
                 do {} while ((voc_status_word != 0) && (kbhit() == 0));
                 if (kbhit() != 0) _voc_stop();
                 break;
       case 'm': printf("                Cancel...->Press <ESC>");
                 ch = ' ';
                 do {
                 _voc_output_loop(voc_buffer);
                 do {} while ((voc_status_word != 0) && (kbhit() == 0));
                 if (kbhit() != 0) ch = getch();
                } while (ch != (char)27);
                 _voc_stop();
                 break;
       }

/* Free VOC file memory */
  _dos_freemem(FP_SEG(voc_buffer));
  _voc_deinstall_driver();

} /* End of playback */
```

```
/*********** VOCTOOL.H **********/

#include <io.h>
#include <stdio.h>
#include <stdlib.h>
#include <dos.h>
#include <bios.h>
#include <fcntl.h>
#define  FALSE  0
#define  TRUE   1
#define  DWORD  unsigned long
#define  WORD   unsigned int
#define  BYTE   unsigned char
#define  VOCPTR char far*
const char voctool_version[] = "v1.5";
//const char voctool_version[] = "v4.01";
const BYTE voc_breakend      = 0;
const BYTE voc_breaknow      = 1;
   WORD    voc_status_word      = 0;
   WORD    voc_err_stat         = 0;
   char    vocfile_header[]     =
            "Creative Voice File\0x1A\0x1A\0\0x0A\1\0x29\0x11\1";
   BYTE    voc_paused           = 0;
   BYTE    vocdriver_installed  = 0;
   WORD    vocdriver_version    = 0;
   BYTE    vocfile_headerlength = 0;
   VOCPTR  voc_ptr_to_driver    = 0;
/** Prototypes **/
   void     _textnumerror(void);
   void     _print_voc_errmessage(void);
   VOCPTR   _voc_get_buffer(char *voicefile);
   WORD     _voc_getversion(void);
   void     _voc_setport(WORD portnumber);
   void     _voc_setirq(WORD irqnumber);
   WORD     _voc_init_driver(void);
   void     _voc_deinstall_driver(void);
   void       _voc_input(unsigned char huge*bufferaddress,unsigned long
int count);
   void     _voc_set_filter(void);
   void     _voc_set_speaker(BYTE OnOff);
   void     _voc_output(char far *bufferaddress);
   void     _voc_output_loop(char far *bufferaddress);
   void     _voc_stop(void);
   void     _voc_pause(void);
   void     _voc_continue(void);
   void     _voc_breakloop(WORD breakmode);

void _textnumerror(void)
{
   printf("Error %d: ",voc_err_stat);
   _print_voc_errmessage();
   exit(voc_err_stat);
}

void _print_voc_errmessage(void)
/* INPUT: None; OUTPUT: None; PURPOSE: Displays SB error as text. */
{
   switch (voc_err_stat) {
       case 100 : printf(" CT-VOICE.DRV driver file not loaded");break;
       case 110 : printf(" No memory free for driver file ");break;
```

```
      case 120 : printf(" Wrong driver file ");break;
      case 200 : printf(" VOC file not found ");break;
      case 210 : printf(" No memory free for VOC file ");break;
      case 220 : printf(" File is not in VOC format ");break;
      case 300 : printf(" Memory allocation error occurred ");break;
      case 400 : printf(" No Sound Blaster card found ");break;
      case 410 : printf(" Wrong port address used ");break;
      case 420 : printf(" Wrong interrupt used ");break;
      case 500 : printf(" No loop in preparation ");break;
        case 510 : printf(" No sample in the output ");break;
      case 520 : printf(" No paused sample available ");break;
       }
}
VOCPTR _voc_get_buffer(char *voicefile)
/* INPUT: Filename as string; OUTPUT: Pointer to buffer with VOC data;
 * PURPOSE: Loads a file into memory and returns the value pointer to
           the data upon being successfully loaded. */
{
       int        filehandle;
       long       filesize;
         WORD       blocksize;
       WORD       byte_read ;
       char huge *filepointer;
       VOCPTR     bufferaddress;
       BYTE       check;
       WORD       segment;
/* VOC file not found */
   if (_dos_open(voicefile,O_RDONLY,&filehandle) != 0)
      {
       voc_err_stat = 200;
       return(FALSE);
        }
   filesize = filelength(filehandle);
   blocksize = (filesize + 15L) /16;
/* Insufficient memory for VOC file */
   check = _dos_allocmem(blocksize,&segment);
   if (check != 0)
      {
       voc_err_stat = 210;
       return(FALSE);
      }
   FP_SEG(bufferaddress) = segment;
   FP_OFF(bufferaddress) = 0;
   filepointer = (char huge*)bufferaddress;
/* Load sample file */
   do
      {
       _dos_read(filehandle,filepointer,0x8000,&byte_read);
       filepointer += byte_read ;
      } while (byte_read == 0x8000);
   _dos_close(filehandle);
/* File not in VOC format */
   bufferaddress[0] = 'C';
   bufferaddress[1] = 'r';

   if ((bufferaddress[0] != 'C') || (bufferaddress[1] != 'r'))
        {
       voc_err_stat = 220;
       return(FALSE);
      }
```

```
/* Load successful */
   voc_err_stat = 0;
   vocfile_headerlength = (BYTE)bufferaddress[20];
   return(bufferaddress);
}
WORD _voc_getversion(void)
/* INPUT: None; OUTPUT: DRV ver. no.; PURPOSE: Determines DRV ver. no. */
{
   WORD     vdummy;
   asm {
      mov       bx,0
      call      voc_ptr_to_driver
      mov       vdummy, ax
       }
   return(vdummy);
}
void _voc_setport(WORD portnumber)
/* INPUT: Port address number; OUTPUT: None;
 * PURPOSE : Sets port address before initialization. */
{
   asm {
      mov     bx,1
      mov     ax,portnumber
      call    voc_ptr_to_driver
       }
}
void _voc_setirq(WORD irqnumber)
/* INPUT: Interrupt number; OUTPUT: None;
 * PURPOSE: Sets the interrupt number before initialization. */
{
   asm {
      mov     bx,2
      mov     ax,irqnumber
      call    voc_ptr_to_driver
       }
}
WORD _voc_init_driver(void)
/* INPUT: None; OUTPUT: Error msg. no., depending on results
 * PURPOSE: Initializes driver software. */
{
       int              filehandle;
       unsigned long    filesize;
       WORD             blocksize;
       char huge       *filepointer;
       WORD             byte_read;
       BYTE             check;
       WORD             segment;
       WORD             vseg;
       WORD             vofs;
       WORD             vout;
   vocdriver_installed = FALSE;
/* Driver file not found */
   if (_dos_open("CT-VOICE.DRV",O_RDONLY,&filehandle) != 0)
      {
       voc_err_stat = 100;
       return(FALSE);
      }
   filesize = filelength(filehandle);
   blocksize = (filesize + 15L) /16;
/* Insufficient memory */
```

```
    check = _dos_allocmem(blocksize,&segment);
    if (check != 0)
       {
        voc_err_stat = 110;
        return(FALSE);
       }
    FP_SEG(voc_ptr_to_driver) = segment;
    FP_OFF(voc_ptr_to_driver) = 0;
    filepointer = (char huge*)voc_ptr_to_driver;
/* Load driver file */
    do
      {
       _dos_read(filehandle,filepointer,0x8000,&byte_read);
       filepointer += byte_read ;
      } while (byte_read == 0x8000) ;
    _dos_close(filehandle) ;
/* Driver file doesn't start with "CT"? Not a CT-Voice driver */
//    if ((voc_ptr_to_driver[3] != 'C') || (voc_ptr_to_driver[4] != 'T'))
    if ((voc_ptr_to_driver[3] != 'K') || (voc_ptr_to_driver[4] != 'T'))

       {
        voc_err_stat = 120;
        return(FALSE);
       }
    voc_ptr_to_driver[0] = 0xE9;
    voc_ptr_to_driver[1] = 0x0B;
    voc_ptr_to_driver[2] = 0x03;

/* Determine driver version */
    vocdriver_version = _voc_getversion();
/* Start driver */
    vseg = FP_SEG(&voc_status_word);
    vofs = FP_OFF(&voc_status_word);
    asm {
       mov       bx,3
       call      voc_ptr_to_driver
       mov       vout,ax
       mov       bx,5
       mov       es,vseg
       mov       di,vofs
       call      voc_ptr_to_driver
        }
/* No Sound Blaster card found */
    if (vout == 1)
       {
        voc_err_stat = 400;
        return(FALSE);
       }
/* Wrong port address used */
    if (vout == 2)
       {
        voc_err_stat = 410;
        return(FALSE);
       }
/* Wrong interrupt number used */
    if (vout == 3)
       {
        voc_err_stat = 420;
        return(FALSE);
       }
```

```
   vocdriver_installed = TRUE;
   return(TRUE);
}
void _voc_deinstall_driver(void)
/* INPUT: None; OUTPUT: None; PURPOSE: Switches off driver, releases
memory. */
{
   if (vocdriver_installed)
      {
       asm {
        mov   bx,9
        call  voc_ptr_to_driver
         }
       _dos_freemem(FP_SEG(voc_ptr_to_driver));
      }
}
void _voc_input(unsigned char huge*bufferaddress,unsigned long int count)
{

   unsigned int low,high;
   WORD vseg;
   WORD vofs;
   vseg =FP_SEG(bufferaddress);
   vofs =FP_OFF(bufferaddress);
   low  =(count & 0xffff);
   high =(count >> 16);

   asm{
      mov   bx,7
      mov   es,vseg
      mov   di,vofs
      mov   ax,5000
      mov   dx,high
      mov   cx,low
      call  voc_ptr_to_driver
      }
}

void _voc_set_filter_r(void)
{
   asm{
      mov   bx,22
      mov   ax,0
      mov   cx,0
      call  voc_ptr_to_driver
      }
}

void _voc_set_speaker(BYTE on_off)
/* INPUT: TRUE for speaker on, FALSE for speaker off; OUTPUT: None;
 * PURPOSE: Toggles SB card output. */
{
   asm {
      mov       bx,4
      mov       al,on_off
      call      voc_ptr_to_driver
       }
}
void _voc_output(VOCPTR bufferaddress)
```

```
/* INPUT: Pointer to sample data; OUTPUT: None; PURPOSE: Plays back sample.
*/
{
   WORD  vseg;
   WORD  vofs;
   _voc_set_speaker(TRUE);
   vseg = FP_SEG(bufferaddress);
   vofs = FP_OFF(bufferaddress) + vocfile_headerlength;
   asm {
      mov       bx,6
      mov       es,vseg
      mov       di,vofs
      call      voc_ptr_to_driver
       }
}
void _voc_output_loop(VOCPTR bufferaddress)
/*    Difference from _voc_output: Speaker not switched on before each
sample -
 *    this can lead to crackling noise on some SB cards during loop
playback. */
{
   WORD  vseg;
   WORD  vofs;
   vseg = FP_SEG(bufferaddress);
   vofs = FP_OFF(bufferaddress) + vocfile_headerlength;
   asm {
      mov       bx,6
      mov       es,vseg
      mov       di,vofs
      call      voc_ptr_to_driver
       }
}
void _voc_stop(void)
/* INPUT: None; OUTPUT: None; PURPOSE: Stops a sample. */
{
   asm {
      mov       bx,8
      call      voc_ptr_to_driver
       }
}
void _voc_pause(void)
/* INPUT: None; OUTPUT: None; PURPOSE: Pauses a sample. */
{
   WORD  pcheck;
   voc_paused = TRUE;
   asm {
      mov       bx,10
      call      voc_ptr_to_driver
      mov       pcheck,ax
       }
   if (pcheck == 1)
      {
       voc_paused = FALSE;
       voc_err_stat = 510;
      }
}
void _voc_continue(void)
/* INPUT: None; OUTPUT: None; PURPOSE: Continues a paused sample. */
{
   WORD  pcheck;
```

```
    asm {
       mov        bx,11
       call       voc_ptr_to_driver
       mov        pcheck,ax
        }
    if (pcheck == 1)
       {
        voc_paused = FALSE;
        voc_err_stat = 520;
       }
}
void _voc_breakloop(WORD breakmode)
/* INPUT: Break mode; OUTPUT: None; PURPOSE: Interrupts a sample loop. */
{
    asm {
       mov        bx,12
       mov        ax,breakmode
       call       voc_ptr_to_driver
       mov        breakmode,ax
        }
    if (breakmode == 1) voc_err_stat = 500;
}
```

```
/*********************************************/
/*                                           */
/*       SuperVGA 256 BGI driver defines     */
/*          Copyright (c) 1991               */
/*        Jordan Hargraphix Software         */
/*                                           */
/*********************************************/

#ifndef _DAC256_
#define _DAC256_
typedef unsigned char DacPalette256[256][3];
#endif

extern int far _Cdecl Svga256_fdriver[];

/* These are the currently supported modes */
#ifndef SVGA320x200x256
#define     SVGA320x200x256         0     /* 320x200x256 Standard VGA */
#define     SVGA640x400x256         1     /* 640x400x256 Svga/VESA */
#define     SVGA640x480x256         2     /* 640x480x256 Svga/VESA */
#define     SVGA800x600x256         3     /* 800x600x256 Svga/VESA */
#define SVGA1024x768x256      4     /* 1024x768x256 Svga/VESA */
#define SVGA640x350x256       5     /* 640x350x256 Svga */
#define SVGA1280x1024x256     6     /* 1280x1024x256 VESA */
#endif

#ifndef XNOR_PUT
#define     XNOR_PUT    5
#define NOR_PUT         6
#define NAND_PUT  7
#define TRANS_COPY_PUT  8     /* Doesn't work with 16-color drivers */
#endif
```

```
/*************************************************/
/*                                               */
/*        Tweaked 256 BGI driver defines         */
/*          Copyright (c) 1991                   */
/*        Jordan Hargraphix Software             */
/*                                               */
/*************************************************/

#include <dos.h>

#ifndef _DAC256_
#define _DAC256_
typedef unsigned char DacPalette256[256][3];
#endif

extern int far _Cdecl Twk256_fdriver[];

/* These are the currently supported modes */
#ifndef TWK320x400x256
#define     TWK320x400x256          0
#define     TWK320x480x256          1
#define     TWK360x480x256          2
#define     TWK376x564x256          3
#define     TWK400x564x256          4
#define     TWK400x600x256          5
#endif

#ifndef XNOR_PUT
#define     XNOR_PUT    5
#define NOR_PUT         6
#define NAND_PUT  7
#define TRANS_COPY_PUT  8     /* Doesn't work with 16-color drivers */
#endif
```

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลงได้ด้วยความอนุเคราะห์และช่วยเหลือจากบุคคลต่าง ๆ มากมาย ขอขอบคุณ

อาจารย์สุวิพล สิทธิชีวภาค ที่ใช้คำปรึกษาและดูแลพวกเรามาตลอด

อาจารย์กฤษณ์ วงศ์รุจิระ ที่ช่วยสอนภาษาซี

อาจารย์นภัทร สระเอี่ยม ที่ให้คำแนะนำ

พี่ธงชัย มณีชูเกตุ ให้ความอนุเคราะห์ภาพถ่าย

พี่เปี๊ยก ให้คำปรึกษาในการทำงาน

นายอดิรุตน์ เยี่ยมเจริญ, นายชูเกียรติ เจียเจษฎากุล, นายอุดม จันทร์จรัสสุข ให้คำแนะนำในการเขียนโปรแกรมภาษาซี

เพื่อนที่ห้องโปรเจ็คอาจารย์ไกรสิน ส่งวัฒนา ให้ความช่วยเหลือเรื่องเครื่องพิมพ์ และเพื่อน ๆ ร่วมห้องโปรเจ็คทุกคนที่เป็นกำลังใจเสมอมา

ผู้จัดทำ

บรรณานุกรม

[1.] ชูชัย ธนสารตั้งเจริญ, ทินกร ดุ๊ก. การสื่อสารข้อมูล. Physic Center. กรุงเทพ 2536.

[2.] ไพศาล สงวนหมู่, ยืน ภู่วรวรรณ. การสื่อสารข้อมูลและไมโครคอมพิวเตอร์-เน็ตเวอร์ค. บริษัทซีเอ็ดยูเคชั่น จำกัด. กรุงเทพ 2536.

[3.] ธันวา ศรีประโมง. การเขียนโปรแกรมภาษาซีสำหรับวิศวกรรม. มหาวิทยาลัยเทคโนโลยี-มหานคร, กรุงเทพ 2537.

[4] ศิวัฒน์ ศิวะบวร และคณะ. การประยุกต์ใช้งานภาษาซี. บริษัทซีเอ็ดยูเคชั่น จำกัด. กรุงเทพ 2535.

[5.] Axel Stolz, "The Sound Blaster Book", Abacus. USA 1992.

[6] Joe Campbell, "C Progamming Guide to SERIAL COMMUNICATION", Howard W.Sams & Company. USA 1989.

[7] Martin L. Moore, "The Ultimate Sound Blaster Book". Que Corporation.