บริญญานิพนธ์

ประจำปีการศึกษา 2537

เรื่อง อุปกรณ์บันทึกข้อความทางโทรศัพท์อัตโนมัติ

(VOICE MAIL BOX)

ภาควิชา วิศวกรรมโทรคมนาคม

คณะวิศกรรมศาสตร์

สถาบันเทคโนโลยีการศึกษาพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

คณะผู้จัดทำ

นาย ปริญญา สุวรรณมาสี

นาย พรต ชุลศิลป์

นาย นัฐวุฒิ จันทรเขต

_____อาจารย์ที่ปรึกษา

(อ เกรียงไกร วงศ์โรจนาภรณ์)

อุปกรณ์บันทึกข้อความทางโทรศัพท์อัตโนมัติ

โดย นายบริญญา สุวรรณมาลี 32100182
นายพรต จุลศิลป์ 34104227
นายนัฐวุฒิ จันทรเขต 34103176

อาจารย์ที่ปรึกษา อ.เกรียงไกร วงศ์โรจนาภรณ์

บทคัดย่อ

ในปัจจุบันการสื่อสารได้มีบทบาทสำคัญในการพัฒนาเศรษฐกิจ และความเป็นอยู่ของมนุษย์ให้ดีขึ้น อุปกรณ์สื่อสารและโครงข่ายโทรคมนาคมได้มีการพัฒนาให้สามารถใช้บริการได้หลายรูปแบบ การนำเทคโนโลยีคอมพิวเตอร์และไมโครโปรเซสเซอร์มาช่วยในการสื่อสารทำให้เกิดความคล่องตัวมากขึ้น เพราะสามารถใช้โปรแกรมควบคุมให้อุปกรณ์ทำงานได้ตามต้องการและเปลี่ยนแปลงแก้ไขการทำงานได้ง่าย

Voice Mail Box เป็นหน่วยข้อมูลสำหรับช่วยอำนวยความสะดวกในการสื่อสารโดยประกอบด้วยส่วนฮาร์ดแวร์ ซึ่งควบคุมโดย เครื่องไมโครคอมพิวเตอร์ และใช้หน่วยความจำของคอมพิวเตอร์ในการบันทึกข้อมูล Voice Mail Box จะทำหน้าที่ในการรับข้อความในรูปของเสียงจากผู้เรียกโดยแปลงให้อยู่ในรูปข้อมูลดิจิตอล แล้วเก็บไว้ในหน่วยความจำ และจำแสดงข้อความออกมาเป็นสัญญาณเสียงตามเดิม เมื่อผู้รับต้องการข้อความ

ในด้านการพัฒนาของการใช้งาน Voici Mail Box จะเห็นว่า เริ่มตั้งแต่ การสอบถามผลการสอบเข้ามหาวิทยาลัย การให้บริการฝากข้อความทางเสียงของวิทยุติดตามตัวตลอดจนโทรศัพท์เคลื่อนที่ เหล่านี้ เป็นต้น

ABSTRACT

In recent year, the communication has and important facility to develop economy, communication equipment and telecommunication network.

Communication techique has been developed to service user in many purpose by using computer and micro processor tecnology. This technology makes communication equipment used very efficiency because of controling equipment by program and function of equipment changed easily

Voice Mail Box is data unit that make communication equipment used conviniently. By hard wave of voice mail box controled by microcomputer, and memory unit of microcomputer, used for keeping data, voice mail box gets sound "Sender" and change it to be digital signal. The digital signal are kept in memory unit and displayed to receiver is Sound.

สารบัญ

บทที่ 1
บทนำ

ในปัจจุบันเป็นยุคของโลกานุวัตรซึ่งการสื่อสารมีบทบาทสำคัญต่อระบบเศรษฐกิจ การพัฒนาอุปกรณ์ที่ช่วยในการสื่อสารจึงมีความสำคัญตามไปด้วย อุปกรณ์สื่อสารต้องส่งข่าวสารได้รวดเร็วและถูกต้องเพื่อตอบสนองความต้องการในภาวะปัจจุบัน โครงข่ายที่มีประสิทธิภาพก็มีความจำเป็นเช่นกัน โดยโครงข่ายจะสามารถรองรับการบริการข่าวสารได้หลายชนิด เช่น ข่าวสารการพูด , ข้อมูลคอมพิวเตอร์ และ สัญญาณภาพ ฯลฯ

ถึงแม้ว่าโครงข่ายและอุปกรณ์สื่อสารต่าง ๆ จะได้รับการพัฒนาจนมีประสิทธิภาพแล้วก็ตาม แต่การสื่อสารก็ยังพบอุปสรรคบางประการที่เกี่ยวกับสภาพทางธรรมชาติ อย่างเช่น สภาพความแตกต่างของเวลาของสถานที่ต่าง ๆ ในพื้นโลก เช่น สถานที่แห่งหนึ่งกำลังอยู่ในช่วงกลางวัน แต่สถานที่แห่งหนึ่งอยู่ในช่วงกลางคืน ทำให้ไม่สะดวกในการสื่อสารกัน นอกจากนั้นยังอาจเกิดจากเหตุผลบางประการของผู้รับข่าวสารเองที่ไม่สามารถจะรับข่างสารได้ทันทีในเวลานั้น

Voice Mail Box เป็นอุปกรณ์ที่ถูกสร้างขึ้นเพื่อแก้ไขปัญหาที่กล่าวมา โดยทำหน้าที่ในการบันทึกข้อความจากผู้เรียกได้หลาย ๆ คน ให้กับผู้รับได้หลาย ๆ คน โดยใช้รหัสเป็นตัวแยกแยะผู้รับ เมื่อผู้รับมีความพร้อมที่จะรับข้อความก็สามารถโทรมาตรวจสอบข้อความได้ โดยอาจจัดให้มีเครื่องคอมพิวเตอร์ทำการโทรไปเตือนผู้รับข้อความว่ามีข้อความฝากมาให้

นอกจากทำหน้าที่หลักที่กล่าวมาแล้ว Voice Mail Box ยังสามารถจัดบริการพิเศษอื่น ๆได้อีก เช่น การบริการแจ้งผลการสอบ, บริการโทรออกอัตโนมัติ, บริการเตือนเวลา

การทำงานของ Voice Mail Box จะทำงานโดยการตรวจจับการโทรเข้ามาของผู้เรียกแล้วทำการตอบรับโดยใช้ข้อมูลเสียงในรูปดิจิตอลที่เก็บไว้ในหน่วยความจำ ทำการส่งออกไปโดยผ่านส่วนการแปลงสัญญาณดิจิตอลให้เป็นสัญญาณอนาลอก เพื่อแปลงข้อมูลให้อยู่ในรูปของเสียงพูดเพื่อติดต่อกับผู้เรียก จากนั้นผู้เรียกจะทำการถอดรหัสของผู้รับรหัสที่กดจะผ่านมาทางคู่สายในรูปของสัญญาณ DTMF (Dual Tone Multi Frequency) ทางด้านรับจะรับสัญญาณแล้วจะทำการถอดรหัสให้อยู่ในรูปข้อมูลฐาน 2 CPU จะรับข้อมูลแล้วทำการค้นหาไฟล์ของผู้รับ จากนั้นจึงเริ่มทำการบันทึกข้อความของผู้เรียก โดยการแปลงข้อความเสียงให้อยู่ในรูปข้อมูลดิจิตอลและทำการบันทึกเก็บไว้ในหน่วยความจำเมื่อผู้รับต้องการตรวจสอบ

ข้อความ CPU จะทำการโหลดไฟล์ของผู้รับแล้วแปลงข้อมูลกลับเป็นสัญญาณเสียงตามเดิม เพื่อแสดงข้อมูลให้กับผู้รับ

บทที่ 2
ความรู้เบื้องต้นเกี่ยวกับโทรศัพท์

2.1 ส่วนประกอบเกี่ยวกับโทรศัพท์

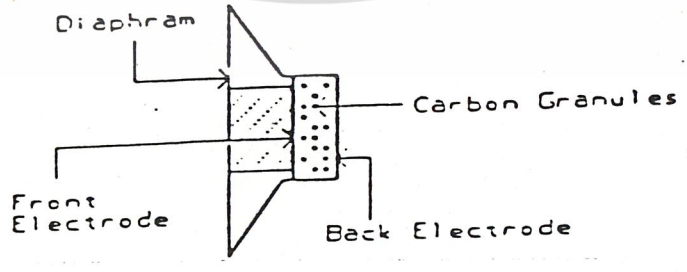
เครื่องโทรศัพท์ (Telephone Set) ประกอบด้วยส่วนที่สำคัญดังนี้คือ เครื่องส่ง (Transmiter) เครื่องรับ(Receiver), กระดิ่ง (Ringer), สวิทช์ฮุค (Hook Switch) และหน้าปัดสำหรับหมุนหรือกดหมายเลข (Dial) สำหรับเครื่องส่งและเครื่องรับรวมเรียกว่า ปากพูดหูฟัง (Handset) ซึ่งเป็นอุปกรณ์ที่ใช้สำหรับเปลี่ยนพลังงานเสียงที่เกิดจากการพูดให้เป็นพลังงานไฟฟ้าและเปลี่ยนพลังงานไฟฟ้าที่ได้รับให้เป็นพลังงานเสียงอีกครั้งโดยใช้เครื่องส่งเป็นตัวเปลี่ยนพลังงานเสียงให้เป็นพลังงานไฟฟ้า และใช้เครื่องรับเป็นตัวเปลี่ยนพลังงานไฟฟ้าให้เป็นพลังงานเสียง สัญลักษณ์ที่ใช้สำหรับ เครื่องส่งและเครื่องรับ แสดงดังรูปที่ 1

สัญลักษณ์ของเครื่องส่ง สัญลักษณ์ของเครื่องรับ

รูปที่ 1 สัญลักษณ์ของเครื่องส่งและเครื่องรับ

เครื่องส่ง

เครื่องโทรศัพท์นั้น จำเป็นที่จะต้องใช้เครื่องที่มีประสิทธิภาพและความไวสูงจึงต้องใช้เครื่องส่งแบบคาร์บอน (Carbon) ซึ่งประกอบด้วยชิ้นส่วนตัวเล็ก ๆ ของคาร์บอนเรียกว่าผงถ่าน (Carbon Granule) แผ่นคาร์บอนอิเล็กโทรด (Carbon electrode) จำนวน 2 แผ่น และไดอะแฟรม (Diaphrame) ดังรูปที่ 2

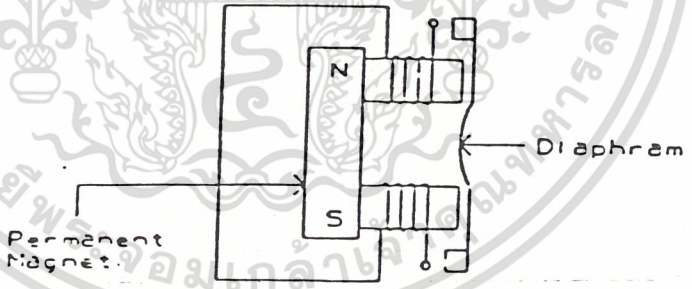


รูปที่ 2 ส่วนประกอบของเครื่องส่ง

เมื่อคลื่นเสียงกระทบกับแผ่นไดอะแฟรม จะทำให้แผ่นไดอะแฟรมสั่นไปมาพลังงานเสียงก็จะเปลี่ยนเป็นพลังงานกล ในตำแหน่งที่แผ่นไดอะแฟรมถูกกดจะทำให้แผ่นอิเล็กโทรดแผ่นหน้าเคลื่อนที่เข้า เป็นผลทำให้ผงถ่านถูกอัดติดกันมากยิ่งขึ้น การอัดตัวของผงถ่านเหล่านี้จะทำให้ความต้านทาน ระหว่างแผ่นอิเล็กโทรดมีค่าลดลง ในทางตรงกันข้ามเมื่อแผ่นไดอะแฟรมเคลื่อนที่ออก ก็จะเป็นผลทำให้แผ่นอิเล็กโทรดแผ่นหน้าเคลื่อนที่ออกด้วยซึ่งจะทำให้ความต้านทานของเครื่องส่งเพิ่มขึ้น

เครื่องรับ

หลักการของเครื่องรับ คือ มีขดลวดพันอยู่ที่ขั้วทั้งสองของแม่เหล็กถาวรนี้จะมีอำนาจแม่เหล็กดึงดูดแผ่นไดอะแฟรมเข้ามา เมื่อมีกระแสไฟสลับ (Speech current) ไหลผ่านขดลวดก็จะผลทำให้เกิดเส้นแรงแม่เหล็กขึ้น ทิศทางของเส้นแรงแม่เหล็กมีทิศทางตรงกันข้ามกับทิศทางกระแสไฟฟ้าที่ไหลในวงจรซึ่งอาจไปเสริมหรือต้านเส้นแรงแม่เหล็กของแม่เหล็กถาวร แผ่นไดอะแฟรมก็จะเคลื่นที่เข้าหรือออกตามขนาดและความถี่ของกระแสไฟฟ้าสลับนั้น ซึ่งจะมีผลทำให้เกิดคลื่นเสียงที่มีขนาดและความถี่เท่ากับกระแสไฟฟ้าสลับที่ไหลเข้ามาในวงจร คลื่นเสียงที่เกิดขึ้นย่อมจะต้องมีการสูญเสียไปบ้างเนื่องจากมีการเปลี่ยนรูปพลังงาน ดังนั้น output ของคลื่นเสียงจะน้อยกว่า input ของพลังงานไฟฟ้าที่ได้รับที่เครื่องรับ



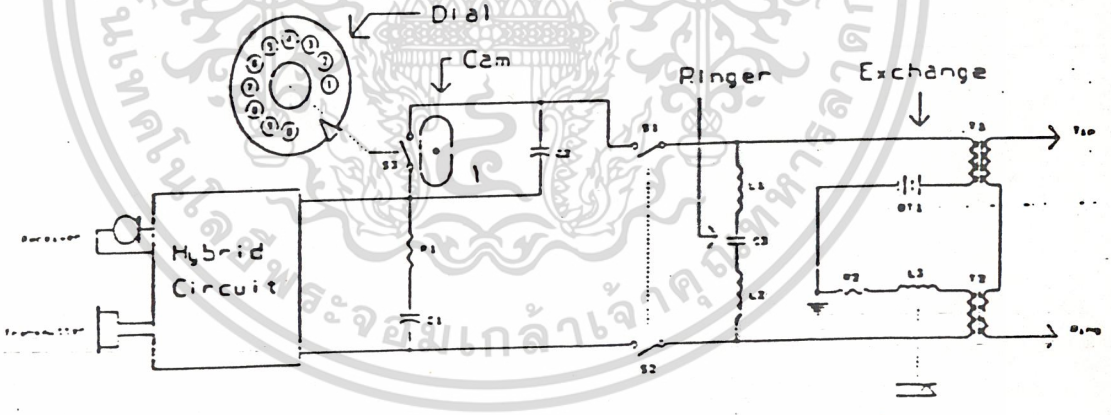
รูปที่ 3 ส่วนประกอบของเครื่องรับ

กระดิ่ง

กระดิ่งของเครื่องโทรศัพท์นั้นเมื่อมีการเรียกเข้า กระดิ่งที่เครื่องโทรศัพท์ของผู้ถูกเรียกจะดังขึ้น ซึ่งหมายถึงชุมสายโทรศัพท์ได้ทำการส่งกระแสไฟฟ้าสลับ(Ringing voltage) มาป้อนที่กระดิ่งของเครื่องรับโทรศัพท์ โดยทั่วไปแล้วกระแสไฟฟ้าสลับจะมีค่าประมาณ 75-80 โวลต์ ความถี่ 18-25 เฮิร์ท

หน้าปัด

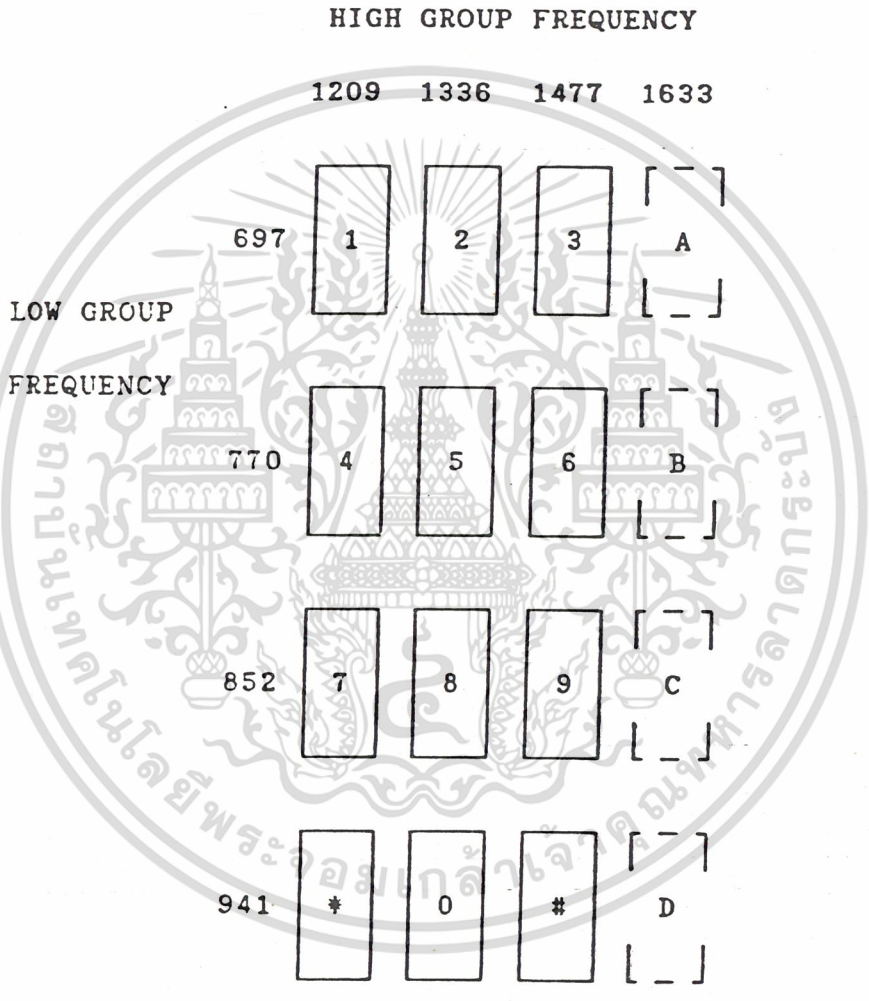
หน้าปัดของเครื่องโทรศัพท์มีอยู่ 2 แบบ คือ แบบหมุน(Rotary dial) ซึ่งการหมุนจะทำให้เกิดพัลส์ขึ้นเป็นจำนวนเท่ากับเลขหมายที่หมุน และแบบกดปุ่ม (Push button) ซึ่งใช้กรรมวิธีของการผสมความถี่ในการส่งเลขหมายโทรศัพท์ หน้าปัดแบบหมุนนั้น เมื่อผู้เรียกยกปากพูดหูฟังขึ้นจากที่รองรับ (Cradle) จะทำให้ Hook switch S_1 และ S_2 ปิดวงจรของสายเส้น Tip และ Ring ซึ่งเป็นผลทำให้ครบวงจรของรีเลย์ (Relay coil)ในชุมสายโทรศัพท์อุปกรณ์สวิทช์ในชุมสายก็จะส่งสัญญาณให้หมุน (Dial tone) มายังเครื่องโทรศัพท์ของผู้เรียก เพื่อเป็นสัญญาณแสดงให้ผู้เรียกทราบว่าเริ่มหมุนหมายเลขได้แล้ว และชุมสายโทรศัพท์ก็พร้อมที่จะรับหมายเลขที่ผู้เรียกหมุนเมื่อผู้เรียกหมุนหมายเลขใดหมายเลขหนึ่งและเมื่อหมุนเสร็จก็ปล่อยมือ หน้าปัดของโทรศัพท์จะหมุนกลับที่เดิม ขณะที่หน้าปัดหมุนกลับที่เดิมจะมีผลทำให้ลูกเบี้ยว (Cam) หมุนตาม การหมุนของลูกเบี้ยวนี้จะทำให้สวิทช์ S_3 เปิดและปิดเป็นจำนวนครั้งเท่ากับเลขหมายที่หมุน การที่สวิทช์ S_3 ปิดวงจรจะทำให้กระแสไฟฟ้าไหลได้ และเมื่อสวิทช์ S_3 เปิดวงจรกระแสก็จะหยุดไหล การที่กระแสไหลและหยุดไหลก็จะมีผลทำให้เกิดพัลส์ขึ้น และจำนวนพัลส์ที่เกิดขึ้นก็จะมีจำนวนเท่ากับเลขหมายที่หมุน เช่น หมุนเลข 1 จะเกิด 1 พัลส์ถ้าเลข 5 ก็จะเกิด 5 พัลส์ ส่วนเลข 0 จะเกิด 10 พัลส์ เป็นต้น



รูปที่ 4 วงจรของโทรศัพท์แบบหมุน

สำหรับหน้าปัดแบบกดปุ่มใช้กรรมวิธีการผสมความถี่ ในการส่งเลขหมายโทรศัพท์โดยทั่วไปมี 12 ปุ่มแบ่งเป็น 4 แถวและ 3 คอลัมน์ แต่มีเครื่องโทรศัพท์บางแบบจะมี 16 ปุ่มโดยเพิ่มคอลัมน์ที่ 4 ความถี่ที่ใช้ในแต่ละแถวและคอลัมน์จะมีความถี่ต่างกัน โดยแบ่งความถี่เป็นสองกลุ่มคือกลุ่มของความถี่ทั้ง 4 แถวเรียกว่ากลุ่มความถี่ต่ำ (Low Group Frequency) และ

กลุ่มความถี่ทั้ง 4 คอลัมน์ เรียกว่ากลุ่มความถี่สูง (High Group Frequency) การกดหมายเลขใดๆจะทำให้วงจรภายในเครื่องโทรศัพท์ผลิตความถี่ออกมา 2 ความถี่ เช่นกดเลข 5 ความถี่ที่ผลิตออกมาคือ 770 Hz และ 1336 Hz เป็นต้น มาตรฐานความถี่ที่ใช้กันและตำแหน่งเลขหมายต่างๆจะถูกจัดให้มีดังรูป โดยกำหนดค่าความผิดพลาดที่ยอมให้เกิดขึ้นเท่ากับ 1.5 % สำหรับการผลิตความถี่และ 2.5 % สำหรับการรับเลขหมาย



รูปที่ 5 แสดงความถี่ของโทรศัพท์แบบกดปุ่ม

บทที่ 3

สัญญาณต่าง ๆ บนสล็อตของ IBM/PC

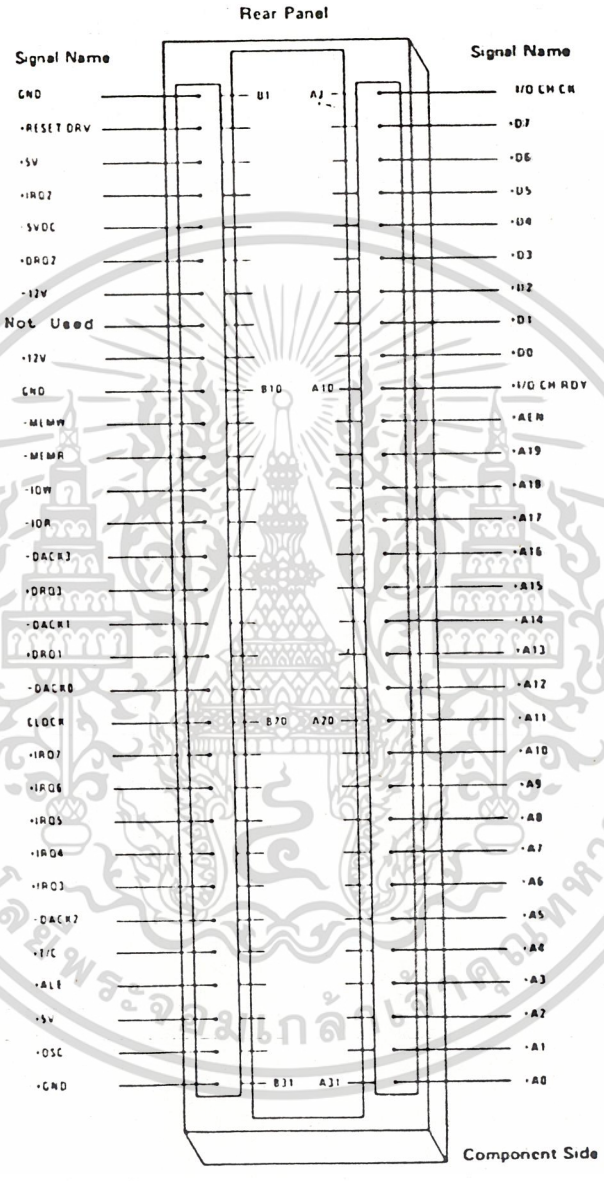
ภายใน IBM/PC ได้มีการออกแบบให้สามารถที่จะเพิ่มเติมวงจรอินเทอร์เฟสเข้าไปในภายหลังได้ โดยผ่านทางสล็อตที่อยู่บนเมนบอร์ด (Main Board) สำหรับสล็อตบนเมนบอร์ดจะมี จำนวน 5 สล็อต (สำหรับใน IBM PC/XT จะมี 8 สล็อต;จะกล่าวถึงในภายหลัง) ซึ่งแต่ละสล็อตจะมีจำนวนขาทั้งสิ้น 62 ขา แบ่งออกเป็น 2 ข้าง ๆ ละ 31 ขาส่วนการเรียกตำแหน่งขาของสล็อตเหล่านี้จะขึ้นอยู่กับว่าขานั้นอยู่ข้างใด (ซ้ายหรือขวา) ของสล็อต โดยที่ขาที่อยู่ข้างซ้ายของสล็อตจะเรียกโดยใช้อักษร "B" นำหน้าเลขตำแหน่งของขา เช่น ขา B16 ก็คือขาทางด้านซ้ายของสล็อตขาที่ 16 (นับจากทางด้านท้ายของเครื่อง) แต่ละขาของสล็อตเหล่านี้จะเชื่อมต่อกับเส้นสัญญาณต่างๆบนเมนบอร์ดทำให้การสร้าง วงจรอินเทอร์เฟสกับ IBM/PC สามารถทำได้โดยสะดวก ซึ่งเส้นสัญญาณที่เชื่อมต่อกับขาของสล็อต เหล่านี้จะประกอบไปด้วยเส้นสัญญาณของบัสแอดเดรส(Address Bus), บัสข้อมูล (Data Bus), บัสควบคุมการเขียน/อ่านข้อมูลจากหน่วยความจำ หรือพอร์ท I/O, เส้นสัญญาณสำหรับการขอ อินเตอร์รัพท์ของวงจรอินเตอร์เฟส,เส้นสัญญาณสำหรับการขอDMA, สัญญาณฐานเวลา (Timing Signal) ต่าง ๆ ที่ใช้ในระบบ,เส้นสัญญาณแสดงการรีเฟรชหน่วยความจำ และสัญญาณสำหรับการ ตรวจสอบความผิดพลาด (I/O CHCK)

นอกจากเส้นสัญญาณเหล่านี้แล้ว สล็อตบนเมนบอร์ดยังเชื่อมต่อกับแหล่งจ่ายไฟต่าง ๆ ที่ใช้ในระบบอีกด้วย คือ +5Vdc, -5Vdc, +12Vdc, -12Vdc

รายละเอียดเกี่ยวกับสัญญาณต่าง ๆ

OSC (Oscilltor;ขา B30) :

ขานี้เป็นเอาท์พุทที่เชื่อมต่อกับสัญญาณคล็อกที่มีค่าความถี่สูงสุดบนเมนบอร์ด คือ 14.31818 MHz ซึ่งมีคาบเวลาประมาณ 70 nanosec. และมี Duty Cycle (ช่วงเวลาใน 1 คาบที่สัญญาณคล็อกมีลอจิกเป็น "1" หารด้วยคาบเวลาทั้งหมด) ประมาณ 50% สัญญาณคล็อกอื่น ๆ ของระบบ เช่น คล็อกที่ป้อนให้กับ 80386หรือ ชิพซัพพอร์ทต่าง ๆ นั้นจะถูกสร้างขึ้นโดยการหารสัญญาณคล็อกนี้ อย่างไรก็ตามสิ่งหนึ่งที่จะต้องคำนึงถึงในการใช้งานสัญญาณ OSC ก็คือสัญญาณนี้จะไม่ Synchronize กับสัญญาณอื่น ๆ บนบัสของระบบ ดังนั้นจึงไม่ควรที่จะนำสัญญาณจากขา OSC นี้ไปใช้เป็นสัญญาณ คล็อกสำหรับวงจรภายนอกอื่น ๆ ที่ทำงานร่วมกับระบบ



CLK (Clock;ขา B20)

ขาสัญญาณนี้เป็นเอาร์พุท ซึ่งต่อกับสัญญาณคล็อกที่ถูกสร้างขึ้นโดยการหารสัญญาณ OSC ด้วย 3 ทำให้ได้ความถี่ประมาณ 4.77 MHz (14.31818 MHz/3) หรือ มีช่วงเวลาใน 1 คาบ (ช่วงเวลา

ของคล็อก 1 ลูก) เท่ากับ 210 nanosec (1/4.77 MHz) สำหรับค่า Duty Cycle ของสัญญาณนี้จะมีค่าประมาณ 1/3 คือ ใน 1 คาบจะมีช่วงเวลาที่เป็นลอจิก "1" เท่ากับ 1/3 ของคาบเวลาทั้งหมด หรือประมาณ 70 nanosec. และช่วงเวลาที่เป็นลอจิก "0" เท่ากับ 2/3 ของคาบเวลาทั้งหมด หรือประมาณ 140 nanosec สัญญาณนี้เป็นสัญญาณที่ถูกใช้เป็นคล็อกของระบบ

RESET DRV (ขา B2) :

ขาสัญญาณนี้เป็นเอาท์พุท ซึ่งจะแอคทีฟ (ลอจิก "1") ในช่วงที่เราเริ่มจ่ายไฟให้กับระบบ และจะยังคงแอคทีฟไปจนกว่าระบบต่างๆภายในIBM/PC จะพร้อมที่จะทำงานได้จากนั้นสัญญาณนี้ก็จะเปลี่ยนกลับเป็นลอจิก "0" นอกจากนี้ในระหว่างการทำงานของ IBM/PC ถ้าระดับแรงดันของแหล่งจ่ายไฟตกลง สัญญาณนี้ก็จะถูกทำให้แอคทีฟเช่นกัน โดยทั่วไปแล้วสัญญาณนี้จะถูกนำไปใช้ในการรีเซ็ทวงจรอินเทอร์เฟสหรืออุปกรณ์เหล่านั้นถูกปรับให้อยู่ในสภาวะที่แน่นอน ก่อนที่จะเริ่มต้นทำงานในระบบ (สภาวะนี้เป็นสภาวะที่เราทราบ และต้องการให้วงจรทำงานในขณะที่ระบบถูกรีเซ็ท)

AO-A19 (Address Bus ; ขา A31-A12)

ขาสัญญาณทั้ง 20 ขานี้เป็นเอาร์พุท ซึ่งใช้สำหรับกำหนดแอดเดรสของหน่วยความจำ หรืออุปกรณ์ I/O ที่ 80386 ต้องการติดต่อด้วย โดยที่สัญญาณ AO จะมีนัยสำคัญต่ำที่สุด (Most Significant Bit) สำหรับค่าแอดเดรสบนบัสแอดเดรส AO-A19 นี้ จะถูกกำหนดโดย 80386ในระหว่างขบวนการอ่าน/เขียนข้อมูลลงในหน่วยความจำหรืออุปกรณ์ I/O แต่ในช่วงของขบวนการDMAนั้น DMA-Controller จะเป็นผู้กำหนดค่าแอดเดรสบนบัสแอดเดรสเอง (ในระหว่างนี้ 80386จะถูกตัดออกจากระบบ)

จะเห็นได้ว่าจำนวนเส้นแอดเดรสนี้จะมีอยู่ 20 เส้น ซึ่งสามารถที่จะอ้างแอดเดรสสำหรับหน่วยความจำได้ถึง 1 Mbyte แต่อย่างไรก็ตามจะมีแอดเดรสบางแอดเดรสที่ถูกใช้งานโดย IBM/PC อยู่ก่อนแล้ว คือแอดเดรสของหน่วยความจำ RAM บนเมนบอร์ดที่ถูกใช้โดยระบบ จำนวน 64 Kbyte

(สำหรับ IBM PC/XT จะเป็นจำนวน 256 Kbyte) และแอดเดรสสำหรับหน่วยความจำ ROM อีก 48 Kbyte ซึ่งถูกจัดในช่วงของแอดเดรสบนสุดใน 1 Mbyte คือ OFCOOH จนถึง OFFFFFH

(สำหรับ IBM PC/XT จะเป็น 64 Kbyte)

สำหรับการอ้างแอดแดรสของพอร์ท I/O นั้น จะใช้เส้นแอดเดรสเพียง 16 เส้น คือ AO-A15

ซึ่งจะทำให้อ้างแอดเดรสของพอร์ทได้ 64K พอร์ท โดยผ่านทางชุดคำสั่ง IN และ OUT ส่วนเส้นแอดเดรสที่เหลือคือ A16-A19 นั้นจะไม่ถูกใช้งาน อย่างไรก็ตามภายใน IBM/PC จะใช้เส้นแอดเดรสของพอร์ทเพียง 10 เส้น คือจาก AO-A9 และค่าแอดเดรสที่ใช้งานจะต้องอยู่ในช่วง O2OOH 2จนถึง O3FFH เท่านั้น (ดูจากรายละเอียดในบทที่ 9 "การจัดแอดเดรสสำหรับหน่วยความจำและพอร์ท I/O")

DO-D7 (Data Bus; ขา A9-A2)

ขาสัญญาณนี้จะเป็นแบบ Bi-Directional ซึ่งต่อกับบัสข้อมูลของระบบ เพื่อทำหน้าที่ในการส่งผ่านข้อมูลระหว่างพอร์ท I/O กับ IBM/PC โดยบิท DO จะมีนัยสำคัญต่ำที่สุดและบิท D7 จะมีนัยสำคัญสูงสุด

สำหรับในบัสไซเคิลของการเขียนข้อมูลสร้างขึ้นโดย 80386 นั้นข้อมูลจะถูกส่งออกมาบนบัสข้อมูล ก่อนที่สัญญาณ $\overline{\text{IOW}}$ (ในการที่ต้องการส่งข้อมูลให้กับพอร์ท) หรือ $\overline{\text{MEMW}}$ (ในกรณีที่ต้องการส่งข้อมูลให้กับหน่วยความจำ) จะเปลี่ยนจากลอจิก "O" เป็นลอจิก "1" (ขอบขาขึ้น) ซึ่งโดยทั่วไปขอบขาขึ้นของสัญญาณ $\overline{\text{IOW}}$ หรือ $\overline{\text{MEMW}}$ นี้ จะถูกใช้เพื่อสั่งให้ ให้พอร์ท I/O หรือหน่วยความจำที่มีแอดเดรสตรงกับค่าแอดเดรสบนบัสแอดเดรสนั้นรับข้อมูลไปเก็บไว้

สำหรับในบัสไซเคิลของการอ่านข้อมูลที่สร้างขึ้นโดย 80386นั้น พอร์ท I/O หรือหน่วยความจำที่ถูกอ้างอิงถึงจะต้องส่งข้อมูลออกมาบนบัสข้อมูล ก่อนที่สัญญาณ $\overline{\text{IOR}}$ (ในกรณีที่ต้องการอ่านข้อมูลจากพอร์ท) หรือ $\overline{\text{MEMR}}$ (ในกรณีที่ต้องการอ่านข้อมูลจากหน่วยความจำ) จะเปลี่ยนจากลอจิก "O" เป็นลอจิก "1" (ขอบขาขึ้น)

ALE (Address Latch Enable; ขา B28)

ขาสัญญาณนี้เป็นขาสัญญาณเอาร์พุทที่ Bus Controller สร้างขึ้นเพื่อใช้สำหรับแสดงการเริ่มต้นของบัสไซเคิล และแสดงให้อุปกรณ์ภายนอกทราบว่าแอดเดรสที่ 80386 ต้องการที่จะติดต่อด้วยนั้นถูกส่งออกมาบนบัสแอดเดรสแล้ว โดยที่สัญญาณ ALE นี้จะเปลี่ยนจากลอจิก "1" เป็น "O" เมื่อมีค่าแอดเดรสที่ถูกต้องถูกส่งออกมาบนบัสข้อมูลเรียบร้อยแล้ว ดังนั้นขอบขาลงของสัญญาณ ALE นี้จะถูกใช้ในการแลทช์ค่าแอดเดรสจากบัสแอดเดรส/ข้อมูล (Address/Data Bus ; ADO-AD7) ของ 80386ทำให้สามารถแยกค่าแอดเดรส (AO-A19) และข้อมูล (AO-A7) ออกจากกันได้ อย่างไรก็ตาม ALE จะแอคทีฟเฉพาะในบัสไซเคิลที่สร้างขึ้นโดย 80386เท่านั้น โดยจะไม่แอคทีฟในระหว่างขบวนการ DMA

$\overline{\text{I/O CHCK}}$ (Address Latch Enable; ขา A1):

ขาสัญญาณนี้เป็นเอาร์พุทที่ใช้ในการแสดงความผิดพลาดเกี่ยวกับพาราตี้ ที่เกิดขึ้นในการทำงานของวงจรอินเตอร์เฟสหรืออุปกรณ์ I/O เมื่อขาสัญญาณนี้ได้รับลอจิก "0" จะทำให้ 80386ถูกอินเตอร์รัพท์แบบ Non-Maskable (NMI) อย่างไรก็ตามเราสามารถที่จะกำหนดให้วงจรภายในของ IBM/PC ทำการขออินเทอร์รัพท์ (เมื่อได้รับสัญญาณ $\overline{\text{I/O CHCK}}$) หรือไม่ก็ได้ โดยทำการกำหนดลอจิกของบิทข้อมูลของพอร์ทที่ควบคุมการขออินเทอร์รัพท์แบบ NMI คือบิท D7 ของพอร์ท OOAOH ในกรณีที่บิท D7 ของพอร์ท OOAOH ถูกเซ็ทเป็น "1" ทำให้วงจรภายนอกขออินเทอร์รัพท์แบบ NMI ได้ (Enable) แต่ถ้าบิท D7 ของพอร์ท OOAOH ถูกเซ็ทเป็น "0"ก็จะเป็นการดิสเอเบิล (Disable)การขออินเทอร์รัพท์แบบ NMI ดังนี้

Enable : ใช้คำสั่ง OUT ส่งข้อมูล 8OH ไปยังพอร์ท OOAOH

Disable : ใช้คำสั่ง OUT ส่งข้อมูล OOH ไปยังพอร์ท OOAOH

และเนื่องจากยังมีอุปกรณ์อื่นที่สามารถขออินเทอร์รัพท์แบบ NMIได้อีก ดังนั้นซอฟท์แวร์ที่ใช้งานจะต้องสามารถตรวจสอบว่าการขออินเทอร์รัพท์นั้นเกิดขึ้นจากแหล่งใดได้ด้วย

I/O CHRDY (I/O Channel Ready ; ขา A10) :

ขาสัญญาณนี้เป็นอินพุทที่ใช้เพิ่มช่วงเวลาในบัสไซเคิลในกรณีที่อุปกรณ์ I/O หรือหน่วยความจำที่เกี่ยวข้องกับขบวนการในบัสไซเคิลที่เกิดขึ้นนั้น ไม่สามารถทำงานทันตามช่วงเวลาปกติของบัสไซเคิลนั้น ๆ ได้ (ช่วงเวลาของบัสไซเคิลที่เกี่ยวกับหน่วยความจำใช้ในช่วงเวลาเท่ากับช่วงเวลาของคล็อก 4 ลูก หรือ 840 nanosec. ในขณะที่บัสไซเคิลที่เกี่ยวกับ I/O จะใช้ช่วงเวลาเท่ากับช่วงเวลาของคล็อก 5 ลูก หรือ 1.50 usec.)

เมื่ออุปกรณ์ I/O หรือหน่วยความจำต้องการที่จะเพิ่มช่วงเวลาในบัสไซเคิลให้นานขึ้นอีกนั้นจะสามารถทำได้โดยการป้อนโลจิก "0" ให้กับขา I/O CHRDY ในช่วงเวลาที่ I/O หรือหน่วยความจำที่ถูกกำหนดขึ้นนั้น ได้รับสัญญาณการดีโค้ดแอดเดรส และสัญญาณ $\overline{\text{MEMR}}$, $\overline{\text{MEMW}}$, $\overline{\text{IOR}}$ หรือ $\overline{\text{IOW}}$ แอคทีฟ

IRQ2-IRQ7 (Interrupt Request 2 Through 7 ; ขา B4 และ B25-B21)

ขาสัญญาณทั้ง 6 นี้เป็นขาอินพุทที่ใช้สำหรับการขออินเทอร์รัพท์จาก 80386โดยสัญญาณเหล่านี้จะต่อเข้ากับ 8259A บนเมนบอร์ดโดยตรง โปรแกรมในส่วน BIOS ของ IBM/PC จะทำโปรแกรม 8259A ให้ IRQ2 มีลำดับความสำคัญสูงสุด (Highest Priority) และ IRQ7 มีลำดับความสำคัญต่ำสุด ในกรณีที่มีการขออินเทอร์รัพท์เกิดขึ้น คือ ระดับลอจิกที่ขา IRQ ขาใดขาหนึ่งถูกเปลี่ยนจากลอจิก "0" เป็นลอจิก "1" (ขอบขาขึ้น) 8259A ก็จะทำการส่งสัญญาณ INT ให้กับ 80386 เพื่อทำการขออินเทอร์รัพท์

สิ่งสำคัญในการขออินเทอร์รัพท์โดยผ่านทาง IRQ2-IRQ7 นี้ ก็คืออุปกรณ์ที่ทำการขออินเทอร์รัพท์โดยผ่านทาง IRQ ขาใดก็จะต้องรักษาสัญญาณที่ขา IRQ นั้น ให้แอคทีฟ (ลอจิก "1") อยู่จนกว่าจะได้รับสัญญาณ $\overline{\text{INTA}}$ (Interrupt Acknowledge) จาก 80386เสียก่อน ถ้าไม่เช่นนั้นการขออินเทอร์รัพท์จะถูกยกเลิก และอินเทอร์รัพท์ Level 7 (IRQ7) ก็จะถูกสร้างขึ้นโดยอัตโนมัติ ไม่ว่าการขออินเทอร์รัพท์ที่ถูกยกเลิกนั้นจะเป็นการขออินเทอร์รัพท์ใน Level หรือ ขาใด

แต่อย่างไรก็ตามสัญญาณ $\overline{\text{INTA}}$ นี้จะไม่ถูกต่อออกมาที่ขาของสล็อตด้วย ดังนั้นโปรแกรมที่ทำการตอบสนองต่อการขออินเทอร์รัพท์(Interrupt Service Routine)จะต้องทำการรีเซ็ทสัญญาณ IRQ เองโดยใช้คำสั่ง OUT ไปยังพอร์ท I/O ที่เกี่ยวข้อง

$\overline{\text{IOR}}$ (I/O Read,ขา B14) :

ขาสัญญาณนี้เป็นเอาท์พุทแอคทีฟที่ลอจิก "0" ที่สร้างขึ้นโดย Bus Controller เพื่อใช้ในการแสดงว่าบัสไซเคิลที่เกิดขึ้นนี้ เป็นบัสไซเคิลของการอ่านข้อมูลจากพอร์ท I/O เพื่อให้พอร์ท I/O ที่มีแอดเดรสตรงกับแอดเดรสบนบัสแอดเดรสนั้นส่งข้อมูลออกมาบนบัสข้อมูล โดยข้อมูลจะต้องถูกส่งออกมาบนบัสข้อมูลก่อนขอบขาขึ้นของสัญญาณ $\overline{\text{IOR}}$ ประมาณ 30 nanosec. เพื่อให้มั่นใจได้ว่า 80386สามารถรับข้อมูลได้ถูกต้อง สำหรับในขบวนการ DMA 8237A-5 DMA Controller จะทำการสร้างสัญญาณ $\overline{\text{IOR}}$ เองโดยที่ค่าแอดเดรสที่อยู่บนบัสแอดเดรสจะเป็นค่าแอดเดรสของหน่วยความจำ (แทนที่จะเป็นแอดเดรสของพอร์ท I/O) ที่พอร์ท I/O ที่ขอ DMA ต้องการจะนำข้อมูลไปเก็บ การที่พอร์ทใดจะส่งข้อมูลออกมาบนบัสข้อมูลนั้น จะต้องอาศัยสัญญาณ $\overline{\text{DACK}}$ จาก DMA Controller เป็นตัวกำหนด เช่นกรณีที่สัญญาณ $\overline{\text{DACK1}}$ แอคทีฟก็จะแสดงว่า พอร์ท I/O ที่ขอ DMA ผ่านทางแชนแนลที่ 1 (DRQ1) เป็นต้น

$\overline{IOW}$ (I/O Write; ขา 13) :

ขาสัญญาณนี้เป็นเอาร์พุทแอคทีฟที่ลอจิก "0" ที่ถูกสร้างขึ้นโดย Bus Controller เพื่อใช้แสดงว่าบัสไซเคิลที่เกิดขึ้นนี้เป็นบัสไซเคิลของการเขียนข้อมูลลงบนพอร์ท I/O เพื่อให้พอร์ท I/O ที่มีแอดเดรสตรงกับแอดเดรสบนบัสแอดเดรสนั้น รับข้อมูลที่อยู่บนบัสข้อมูลไปเก็บไว้ อย่างไรก็ตามเนื่องจากในช่วงเวลาที่สัญญาณ $\overline{IOW}$ นี้แอคทีฟ (ลอจิก "0") ข้อมูลบนบัสข้อมูลอาจจะยังไม่สมบูรณ์ ดังนั้นในการออกแบบจึงควรใช้ขอบขาขึ้นของสัญญาณ $\overline{IOW}$ แทนขอบขาลงในการทำให้พอร์ท I/O ที่เกี่ยวข้องรับข้อมูลไปเก็บไว้ เพื่อให้ข้อมูลบนบัสข้อมูลสมบูรณ์เสียก่อน สำหรับในขบวนการ DMA นั้น DMA-Controller จะทำการสร้างสัญญาณ $\overline{IOW}$ เอง โดยที่ค่าแอดเดรสที่อยู่บนบัสแอดเดรสจะเป็นค่าแอดเดรสของหน่วยความจำที่พอร์ท I/O ที่ขอ DMA ต้องการจะอ่านข้อมูล

$\overline{MEMW}$ (Memory Write; ขา B11):

ขานี้เป็นเอาท์พุทแอคทีฟที่ลอจิก "0" ซึ่ง Bus Controller สร้างขึ้นในระหว่างบัสไซเคิลในการเขียนข้อมูลลงในหน่วยความจำของ 80386สัญญาณ $\overline{MEMW}$ นี้จะถูกส่งออกมาเพื่อให้หน่วยความจำที่แอดเดรสตรงกับค่าแอดเดรสบนบัสแอดเดรสนั้น ทำการรับข้อมูลที่อยู่บนบัสข้อมูลไปเก็บไว้ โดยทั่วไปหน่วยความจำจะรับข้อมูลในช่วงขอบขาขึ้นของสัญญาณ $\overline{MEMW}$

สำหรับในระหว่างขบวนการ DMA นั้น 8237A-5 DMA-Controller จะทำการควบคุมบัสต่าง ๆ ของระบบแทน 80386และสัญญาณ $\overline{MEMW}$ จะถูกใช้ในบัสไซเคิลของการเขียนข้อมูลลงในหน่วยความจำ (ข้อมูลถูกส่งจากอุปกรณ์ I/O ไปให้กับหน่วยความจำ)

$\overline{MEMR}$ (Memory Read; ขา B12)

ขานี้เป็นเอาท์พุท ซึ่งสัญญาณนี้จะแอคทีฟ (ลอจิก "0") ในระหว่างบัสไซเคิลของการอ่านข้อมูลจากหน่วยความจำของ 80386 เพื่อให้หน่วยความจำที่มีแอดเดรสตรงกับค่าแอดเดรสบนบัสแอดเดรสนั้น ทำการส่งข้อมูลออกมาบนบัสข้อมูล โดยหน่วยความจำนั้นจะต้องส่งข้อมูลออกมาในช่วงเวลา 30 nanosec. ก่อนที่สัญญาณ $\overline{MEMW}$ จะกลับเป็นลอจิก "1"ทั้งนี้ก็เพื่อให้ 80386ได้รับข้อมูลที่ถูกต้อง

สำหรับในระหว่างขบวนการ DMA นั้น DMA-Controller จะควบคุมบัสต่าง ๆ ของระบบแทน 80386และสัญญาณ $\overline{MEMR}$จะถูกใช้ในบัสไซเคิลของการอ่านข้อมูลจากหน่วยความจำ (ข้อมูลถูกส่งจากหน่วยความจำให้กับอุปกรณ์ I/O)

DRQ1-DRQ3 (DMA Request 1-3; ขา B18,B6 และ ขา B16) :

ขาสัญญาณทั้งสามนี้เป็นสัญญาณอินพุทแอคทีฟที่ลอจิก "1" ซึ่งอุปกรณ์ภายนอกสามารถใช้ในการขอ DMA จากระบบ โดยการป้อนระดับสัญญาณลอจิก "1" ให้กับขา DRQ ขาใดขาหนึ่ง (ขา DRQ ทั้งสามนี้จะต่อเข้ากับ DRQ1-DRQ3 ของ 8237A-5)

เมื่อ 8237A-5 ได้รับสัญญาณแล้วก็จะตรวจสอบว่ามีการขอ DMA ในแชนแนลที่มีลำดับความสำคัญ (Priority) สูงกว่าหรือไม่ ถ้าไม่มีก็จะทำการขอ DMA จาก 80386และตอบรับการขอ DMA จากอุปกรณ์ภายนอก (สัญญาณ $\overline{\text{DACK}}$ ของแชนแนลที่ขอ DMA จะแอคทีฟ) แต่ถ้ามี 8237A-5 ก็จะทำการขอ DMA ให้กับแชนแนลที่มีความสำคัญสูงกว่าก่อนแล้วจึงทำการขอ DMA ให้กับแชนแนลที่มีความสำคัญต่ำกว่า ภายใน ROM BIOS ของ IBM/PC จะโปรแกรม 8237A-5 ให้ DRQ1 มีลำดับความสำคัญสูงสุดและ DRQ3 มีลำดับความสำคัญต่ำสุด ดังนั้นถ้ามีการขอ DMA ของอุปกรณ์ภายนอกผ่านทางแชนแนลที่ (DRQ1) และแชนแนลที่ 2 (DRQ2) 8237A-5 ก็จะทำการขอ DMA ให้กับแชนแนลที่ 1 ก่อนจากนั้นเมื่อเสร็จจากขบวนการ DMA ของแชนแนลที่ 1 แล้ว จึงจะทำการขอ DMA ให้กับแชนแนลที่ 2 อย่างไรก็ตาม 8237A-5 ยังมีแชนแนลสำหรับการขอ DMA อยู่อีก 1 แชนแนล คือ แชนแนลที่ o (DRQO) ซึ่งในความเป็นจริงแล้วแชนแนลนี้จะมีความสำคัญที่สูงกว่าแชนแนลที่ 1 แต่จะไม่ถูกต่อออกมายังขาของสล็อต เนื่องจาก IBM/PC จะใช้แชลแนลที่ o นี้ในการรีเฟรชหน่วยความจำที่เป็น Dynamic RAM

ในการขอ DMA นั้นสัญญาณ DRQ นี้จะต้องแอคทีฟอยู่ในช่วงระยะเวลาหนึ่งเท่านั้นถ้าสัญญาณ นี้แอคทีฟอยู่นานเกินไป จะทำให้เกิดขบวนการ DMA ขึ้นมากกว่า 1 ขบวนการได้ สำหรับวงจรที่ขอ DMA โดยทั่วไปแล้วจะใช้สัญญาณตอบรับการขอ DMA หรือ สัญญาณ $\overline{\text{DACK}}$ ของแชนแนลที่ขอ DMA นั้น ในการรีเซ็ทสัญญาณ DRQ เช่นอุปกรณ์ภายนอกที่ขอ DMA ผ่านทางแชนแนลที่ 1 (DRQ1) ก็จะคอยตรวจสอบการตอบรับการขอ DMA จากสัญญาณ $\overline{\text{DACK}}$ ของแชนแนลที่ 1($\overline{\text{DACK1}}$) เมื่อได้รับสัญญาณจาก $\overline{\text{DACK1}}$ แล้ว ก็จะรีเซ็ทสัญญาณ DRQ1 (เปลี่ยนจากลอจิก "1" เป็น "O")

$\overline{\text{DACK0}}$-$\overline{\text{DACK3}}$ (DMA Acknowledge O-3; ขา B19, B17, B26 และ B15)

สัญญาณทั้ง4นี้เป็นเอาท์พุทแอคทีฟที่ลอจิก"O"ซึ่ง 8237A-5 สร้างขึ้นเพื่อแสดงให้วงจรภายนอกที่ขอ DMA ทราบว่าการขอ DMA นั้นได้รับการตอบสนองแล้ว และ 8237A-5 จะเข้าสู่ขบวนการ

DMA เพื่อให้การส่งผ่านข้อมูลระหว่างอุปกรณ์ I/O ที่ขอ DMA กับหน่วยความจำที่เกิดขึ้นโดยตรง (คือไม่ต้องผ่าน 80386 โดยสัญญาณ $\overline{\text{DACK}}$ นี้จะแอคทีฟในแชนแนลใดก็ขึ้นอยู่กับขบวนการ DMA ที่จะเกิดขึ้นนั้น เป็นการตอบสนองต่อการขอ DMA ในแชนแนลใด เช่นถ้าขบวนการ DMA ที่เกิดขึ้นนั้นเป็นการตอบสนองต่อการขอ DMA ในแชนแนลที่ 2 (DRQ2) สัญญาณ $\overline{\text{DACK2}}$ ก็จะแอคทีฟ เป็นต้น

ดังที่ได้กล่าวแล้วว่าสัญญาณ DRQ0 นั้น จะไม่ถูกต่อออกมายังขาของสล็อต ดังนั้นวงจรอินเทอร์เฟสจึงไม่สามารถจะขอDMA ผ่านทางแชนแนล 0 ได้ แต่สัญญาณ $\overline{\text{DACK0}}$ จะถูกต่อออกมายังสล็อตด้วย (ขา B19) ทั้งนี้ก็เพื่อที่จะแสดงให้เวงจรอินเทอร์เฟสต่าง ๆ ทราบว่าขบวนการ DMAที่เกิดขึ้นในช่วงเวลาที่ $\overline{\text{DACK0}}$ แอคทีฟนั้น เป็นขบวนการที่ใช้สำหรับการรีเฟรชหน่วยความจำที่เป็น Dynamic RAM ซึ่งวงจรอินเทอร์เฟสที่ใช้หน่วยความจำประเภทนี้สามารถที่จะนำไปใช้ในการรีเฟรช Dynamic RAM ที่อยู่ในวงจรได้

โดยที่การรีเฟรชหน่วยความจำนั้นจะเกิดขึ้นในทุก ๆ 15.12 usec. หรือทุก ๆ 72 คล็อก ดังนั้นสัญญาณ $\overline{\text{DACK0}}$ นี้ก็จะแอคทีฟในทุก ๆ 15.12 usec. ด้วย

AEN (Address Enable; ขา A11)

สัญญาณนี้เป็นเอาท์พุทที่ใช้ในการแสดงว่าบัสไซเคิลที่เกิดขึ้นในช่วงเวลาที่สัญญาณ AEN แอคทีฟ (ลอจิก "1") นั้นเป็นบัสไซเคิลของขบวนการ DMA

สำหรับเมนบอร์ดของ IBM/PC นั้น จะใช้สัญญาณนี้ในการดิสเอเบิล(Disable) Bus Controller และจะใช้ดิสเอเบิลพอร์ท I/O ต่าง ๆ ที่ไม่เกี่ยวข้องกับขบวนการ DMA ที่เกิดขึ้น ที่จำเป็นต้องทำเช่นนี้ก็เพราะในระหว่างขบวนการ DMA นั้น 8237A-5 จะส่งแอดเดรสของหน่วยความจำออกมาบนบัสแอดเดรส และจะทำให้สัญญาณ $\overline{\text{IOR}}$ หรือ $\overline{\text{IOW}}$ แอคทีฟด้วย ดังนั้นถ้าไม่ทำการดิสเอเบิลพอร์ท I/O ที่ไม่เกี่ยวข้องไว้ ก็อาจจะทำให้พอร์ท I/O ที่มีแอดเดรสตรงกับค่าแอดเดรสบนบัสแอดเดรส (ซึ่งเป็นแอดเดรสของหน่วยความจำ) นั้น ทำการอ่านหรือส่งข้อมูลออกมาบนบัสข้อมูลทำให้เกิดความผิดพลาดได้

T/C (Terminal Count; ขา B27) :

สัญญาณนี้ถูกสร้างขึ้นจากการนำเอาสัญญาณเอาท์พุทที่ขา EOP ของ 8237A-5 มากลับลอจิก (โดยใช้เกท Inverter)ทำให้สัญญาณ T/C นี้แอคทีฟที่ลอจิก "1"

สำหรับสัญญาณนี้จะแอคทีฟเมื่อจำนวนไบท์ในการส่งผ่านข้อมูลของขบวนการ DMA ในแชนแนลใดแชนแนลหนึ่ง ครบตามจำนวนที่กำหนดไว้ โดยทั่วไปแล้วสัญญาณที่จะถูกใช้ในการสิ้นสุดขบวนการ DMA ที่ทำการส่งผ่านข้อมูลเป็นบล็อกเนื่องจากสัญญาณนี้จะแอคทีฟโดยไม่แสดงว่าเป็นสัญญาณ ของแชนแนลใดดังนั้นจึงต้องทำการนำสัญญาณT/C นี้ผ่านเกท Inverter แล้วนำไป OR กับสัญญาณ $\overline{\text{DACK}}$ เพื่อให้สามารถทราบได้ว่า สัญญาณ T/C ที่เกิดขึ้นนั้นเป็นสัญญาณของแชนแนลใด สำหรับแชนแนลที่ 0นั้นสัญญาณT/C จะแอคทีฟในช่วงเวลาที่คงที่ คือ ทุก ๆ 990.804 millisec. ซึ่งก็คือช่วงเวลาที่ใช้ในการรีเฟรชหน่วยความจำขนาด 64 Kbyte นั่นเอง

บัสของแหล่งจ่ายไฟของระบบ

+5Vdc (ขา B3 และ B29)

ขาทั้งสองนี้ต่อกับแหล่งจ่ายไฟ DC +5V ของระบบ โดยมีค่าความเที่ยงตรง (Regulated) ±5% คืออยู่ในช่วง +4.57 ถึง +5.25 Vdc

+12Vdc (ขา B9)

ขานี้จะต่อกับแหล่งจ่ายไฟ DC +12V ของระบบ โดยจะมีค่าความเที่ยงตรง (Regulated) ±5% คืออยู่ในช่วง +11.4 ถึง +12.6 Vdc

-5Vdc (ขา B5)

ขานี้จะต่อกับแหล่งจ่ายไฟ DC -5V ของระบบ โดยมีค่าความเที่ยงตรง (Regulated) ±10% คืออยู่ในช่วง -13.2 ถึง -10.8 Vdc

-12Vdc (ขา B7)

ขานี้จะต่อกับแหล่งจ่ายไฟ DC -12V ของระบบ โดยมีค่าความเที่ยงตรง (Regulated) ±10% คืออยู่ในช่วง -13.2 ถึง -10.8 Vdc

GND (ขา B1,B10 และ B13)

ขาทั้งสามนี้จะต่อเข้ากับกราวด์ (Ground) ของระบบ

การจัดสัญญาณบนสล็อตของ IBM PC/XT

สำหรับใน IBM PC/XT นั้นจะมีสล็อตสำหรับเชื่อมต่อกับวงจรภายนอกได้มากขึ้น คือ ใน IBM PC/XT จะทำการเพิ่มจำนวนสล็อตบนเมนบอร์ดขึ้นเป็น 8 สล็อตจะยังคงเหมือนกับใน IBM PC เพียงแต่สัญญาณต่าง ๆ ที่จะถูกส่งออกมายังขาของสล็อตที่ 8 นั้น จะถูกต่อผ่านวงจรขับกระแส (Buffer) ก่อน และในสล็อตที่ 8 นี้ ขา B8 จะถูกใช้งานด้วย โดยจะถูกใช้เป็นขา $\overline{\text{CARD SLCTD}}$ (หรือ Card Selected) ซึ่งขาสัญญาณนี้จะเป็นขาอินพุทจากวงจรภายนอกที่เสียบอยู่บนสล็อตที่ 8 เพื่อให้วงจรบนเมนบอร์ดทราบว่าคาร์ดที่อยู่บนสล็อตนี้ถูกเลือกใช้งานอยู่ ซึ่งจะทำให้ Driver บนเมนบอร์ดทำการอ่านหรือส่งข้อมูลไปยังสล็อตที่ 8

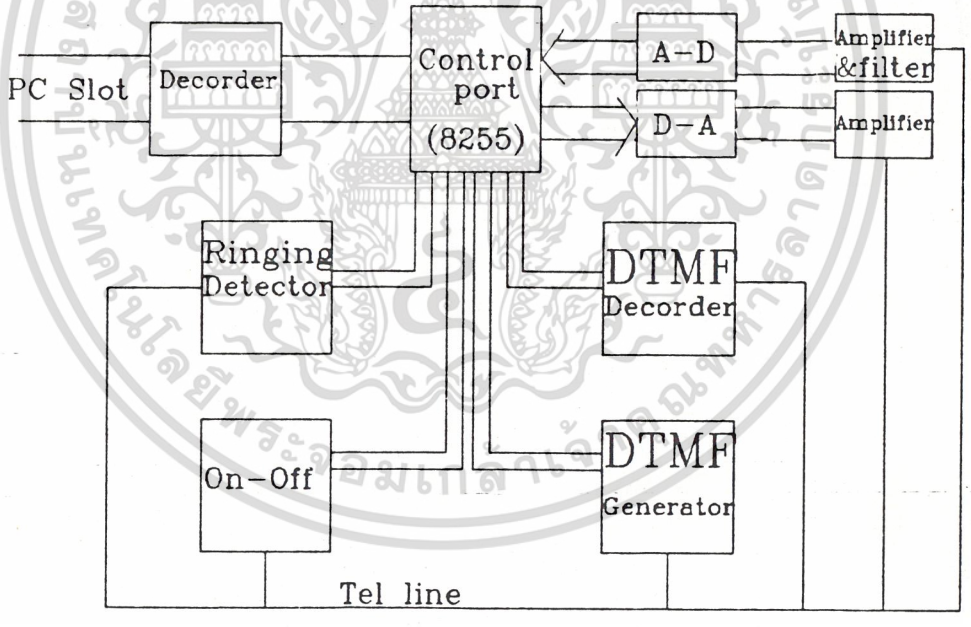
บทที่ 4

การทำงานของ Voice Mail Box

การทำงานของ Voice Mail Box ประกอบด้วยส่วนของ Hardware และส่วนของ Software ประกอบกัน

ส่วนของ Hardware

ส่วนของ Hardware จะประกอบด้วย ส่วนประกอบย่อยต่าง ๆ ดังรูปที่ 1



รูปที่ 1 ส่วนประกอบทาง Hardware

ส่วนประกอบต่าง ๆ ของ Hardware ทั้งหมดที่มีอยู่ 9 ส่วนดังนี้

1) ส่วนของ Decoder

2) ส่วนของ Control port

3) ส่วนของการแปลงสัญญาณ Analog เป็นสัญญาณ Digitol (A -to-D)

4) ส่วนของการการเปลี่ยนสัญญาณ Digital เป็นสัญญาณ Analog (D-to-A)

5) ส่วนของ Amplifier & filter

6) ส่วนตรวจจับสัญญาณ Ringing

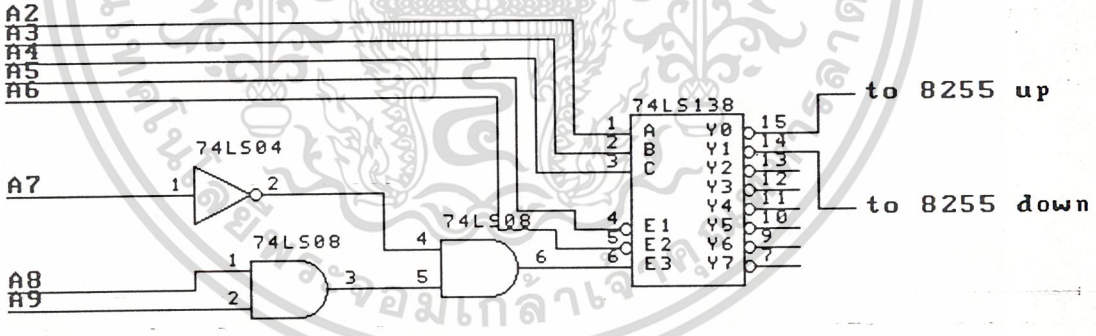
7) ส่วน ยก-วาง หูโทรศัพท์

8) ส่วนถอดรหัสสัญญาณ DTMF

9) ส่วนสร้างสัญญาณ DTMF

ส่วนของ Decoder

เป็นส่วนที่ใช้ในการถอดรหัสของ Address จาก address bus เพื่อใช้ในการแยกแยะการทำงานของ IC ต่าง ๆ เพื่อให้วงจรสามารถทำงานได้อย่างถูกต้อง ส่วนของ Decoder จะแสดงรายละเอียดได้ดังรูป



รูปที่ 2 วงจร Decoder

จากรูปส่วน Decoder จะใช้ address บิทที่ 2 ถึง 9 (A2-A9)มาทำการถอดรหัส ซึ่งในความเป็นจริงแล้ว address bus จะมีทั้งหมด 16 บิท แต่ในที่นี้จะนำมาถอดรหัสทั้งหมด 8 บิท เพราะถือว่าเพียงพอแล้ว

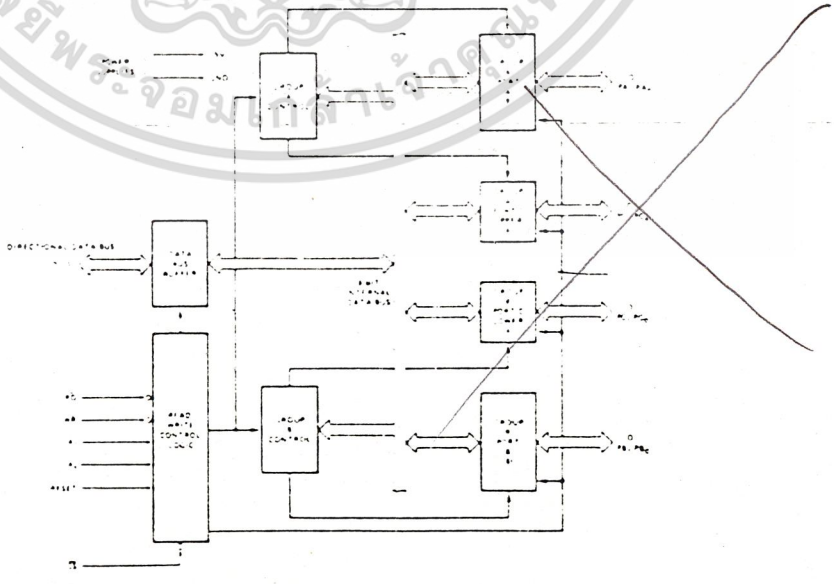
ในวงจรประกอบด้วย IC หลักคือ 8255 จำนวน 2 ตัว ดังนั้นจึงต้องกำหนด address ไว้ 2 address 8255 ตัวแรกจะได้ address เบอร์ 300 H และตัวที่สองจะใช้ 304H ซึ่งจะเขียนเป็นเลขฐานสองได้ดังนี้

	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
300H	1	1	0	0	0	0	0	0	0	0
304H	1	1	0	0	0	0	0	1	0	0

เมื่อนำค่าเหล่านี้ผ่านเข้าไปในเกทต่าง ๆ ดังรูป และผ่าน IC เบอร์ 74LS138 ซึ่งทำหน้าที่เป็นตัวถอดรหัสสำหรับค่า 300H จะได้ output มี logic เป็น 0 ที่ขา $-/0$ และสำหรับค่า 304H จะได้ output มี logic เป็น 0 ที่ขา $-/1$ สัญญาณ output จะต่อเข้ากับขา chip select ($\overline{CS}$) ของ 8255 ซึ่งถ้า output เป็น 0 จะทำให้ 8255 ทำงาน

ส่วนของ Control Port

ในส่วนที่จะใช้ IC เบอร์ 8255 ทำหน้าที่เป็นพอร์ตข้อมูลสำหรับการควบคุมการทำงานของวงจรทั้งหมด เนื่องจากการสั่งงานจะทำโดยเครื่องไมโครคอมพิวเตอร์ ซึ่งจะมีการส่งข้อมูลออกมาทาง data bus เพื่อการควบคุมวงจร ข้อมูลที่ส่งออกมาทาง data bus จะปรากฏเพียงช่วงเวลาหนึ่งแล้วจะเปลี่ยนไป เนื่องจากคอมพิวเตอร์จะต้องส่งข้อมูลไปควบคุมส่วนอื่น ๆ ของคอมพิวเตอร์ตลอดเวลา ดังนั้นในวงจรจะต้องมีส่วนที่ทำหน้าที่ latch ข้อมูลที่ส่งมาทาง data bus ซึ่งในที่นี้ก็คือ พอร์ตต่าง ๆ ที่อยู่ใน 8255 แสดงได้ดังรูปที่ 3



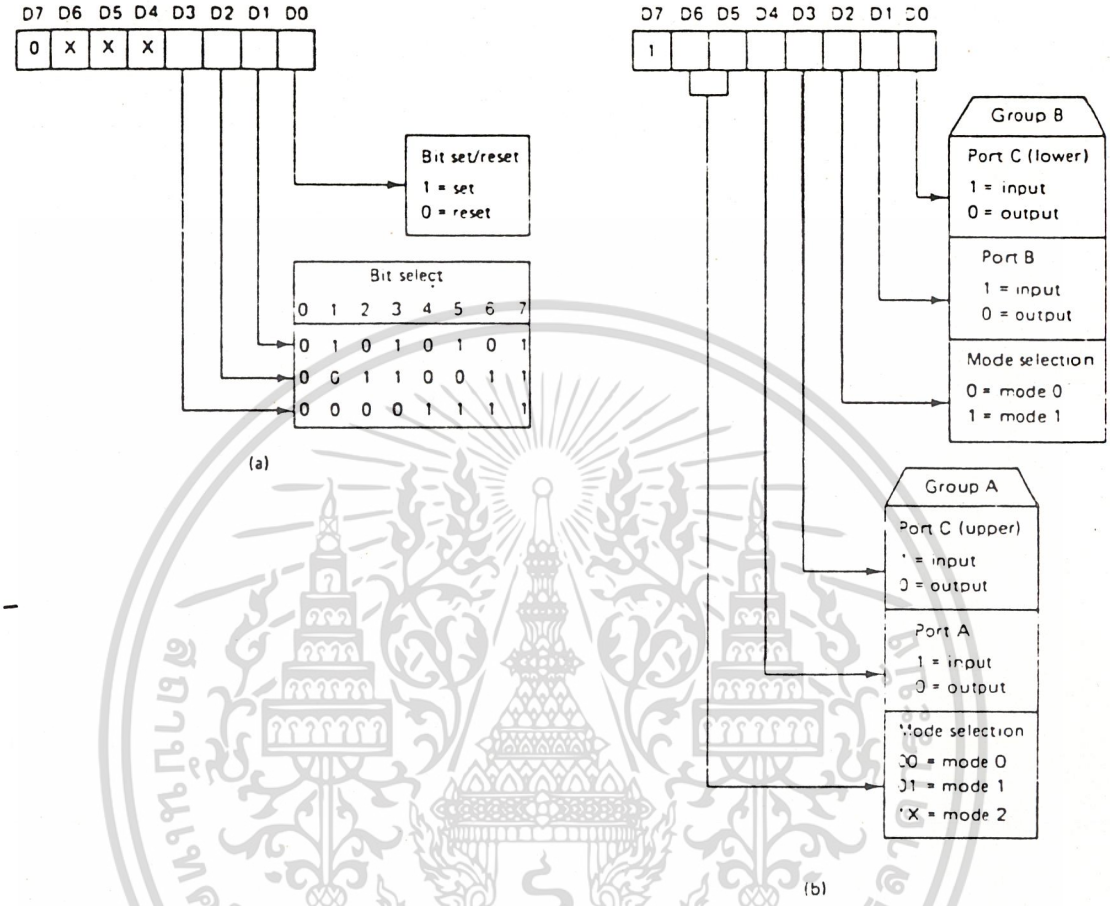
รูปที่ 3 Block diagram ของ8255

จากรูปจะเห็นว่าใน 8255 ประกอบด้วยพอร์ตที่ทำหน้าที่เป็น I/O Port อยู่ 3 พอร์ตคือ พอร์ต A, B และ C ซึ่งพอร์ต C จะมีการแบ่งออกเป็น 2 ส่วน คือ พอร์ต C ด้านบนและด้านล่าง นอกจากนั้นยังมี อีกพอร์ตเป็นพอร์ตสำหรับควบคุมการทำงานของ 8255 (Controll Port) แต่ละพอร์ตที่กล่าวมาจะมี address เป็นของตัวเอง ถ้า CPU ต้องการติดต่อกับพิร์ตใดพอร์ตหนึ่ง จะต้องอ้าง address ผ่านทางบิท A0 และ A1 ดังนี้

A_1	A_0	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	
					Input operation (READ)
0	0	0	1	0	Port A → data bus
0	1	0	1	0	Port B → data bus
1	0	0	1	0	Port C → data bus
					Output operation (WRITE)
0	0	1	0	0	Data bus → port A
0	1	1	0	0	Data bus → port B
1	0	1	0	0	Data bus → port C
1	1	1	0	0	Data bus → control
					Disable function
X	X	X	X	1	Data bus → 3-state
1	1	0	1	0	Illegal condition
X	X	1	1	0	Data bus → 3-state

รูปที่ 4 Truth table ของ 8255

การควบคุมการทำงานของพอร์ตต่าง ๆ ใน 8255 จะทำได้โดยการส่งข้อมูลผ่าน data bus เข้าไปยังพอร์ตควบคุม ข้อมูลในการควบคุมประกอบด้วย 8 บิท แต่ละตำแหน่งของบิท สามารถปรับเปลี่ยนเพื่อควบคุมได้ตามต้องการแต่ละบิทจะแสดงได้ดังรูปที่ 5



รูปที่ 5 พอร์ตควบคุมของ 8255

สำหรับโครงการนี้ 8255 ตัวแรกจะถูกโปรแกรมให้ทำงานได้ดังนี้

- พอร์ต A เป็น input port เพื่อรับข้อมูล 8 บิทจาก ADC 0801
- พอร์ต B เป็น output port เพื่อรับส่งข้อมูลไปยัง DAC 0800
- พอร์ต C ล่าง เป็น output port จะใช้เพียงบิทเดียว ในการ strobe ขา $\overline{WR}$ และ $\overline{INTR}$ ของ ADC 0801

8255 ตัวที่สองจะถูกโปรแกรมให้ทำงานดังนี้

- พอร์ต A เป็น input port เพื่อรับข้อมูล 4 บิท จาก MT8870
- พอร์ต B เป็น output port เพื่อส่งข้อมูล 8 บิท ไปยัง KM5087
- พอร์ต C ล่าง เป็น output port เพื่อส่งสัญญาณไปควบคุมรีเลย์
- พอร์ต C บน เป็น input port ใช้ตรวจสอบสถานะของ JK Flip-Flop

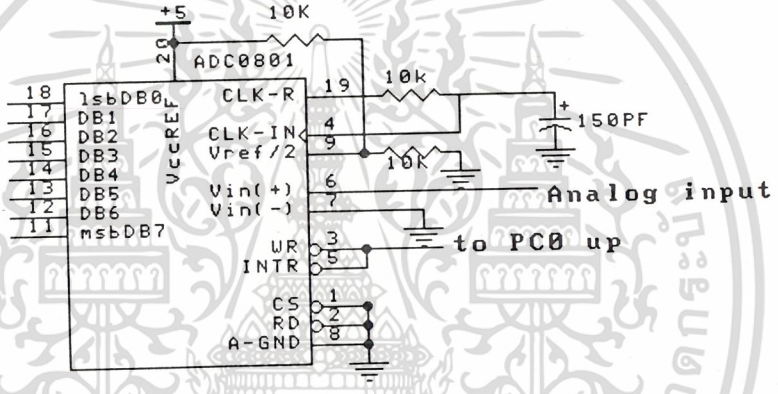
<u>ส่วนของการแปลงสัญญาณ อนาลอก เป็น สัญญาณดิจิตอล (A-to-D)</u>

ส่วนนี้จะใช้ IC เพียงตัวเดียว คือเบอร์ ADL0801 ทำหน้าที่ในการแปลงสัญญาณอนาลอกให้เป็นดิจิตอล

รายละเอียดของ ADL0801 มีดังนี้

- มีช่วงเวลา Access Time 0.135 ns.
- Input/Output สามารถต่อกับ TTL และ CMOS ได้
- ใช้ได้กับ IC สร้างแรงดันอ้างอิง 2.5V
- Input มีย่าน 0-5 V ที่แหล่งจ่ายไฟ 5 V
- ไม่ต้องปรับ Zero Adjust+
- ความผิดพลาด +/- LSB
- Conversion Time 100 μs

วงจรการใช้งาน ADC 0801 แสดงได้ดังรูปที่ 6



รูปที่ 6 วงจรการใช้งาน ADC 0801

จากรูปขา $\overline{WR}$ จะถูกเชื่อมกับ PC0 ของ 8255 เพื่อให้ Software สามารถควบคุมการแปลงสัญญาณได้ โดยการแปลงสัญญาณแต่ละครั้งจะเริ่มเมื่อได้รับ pulse ครบรอบ 1 pulse เมื่อ $\overline{WR}$ เป็น 0 register ภายใน IC จะอยู่ในสถานะ reset เมื่อ $\overline{WR}$ กลับเป็น 1 แล้วการแปลงสัญญาณจากอนาลอกเป็นดิจิตอลจะเริ่มขึ้น

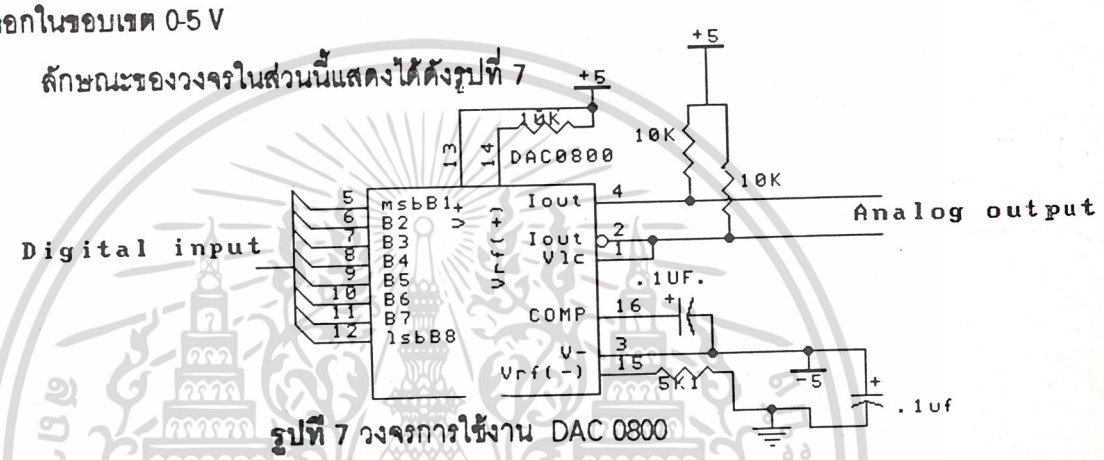
การปรับขอบเขต input ของสัญญาณอนาลอกสามารถทำได้โดยการปรับระดับ Voltage ของขา Vin(-) และขา $V_{REF}/2$ ตัวอย่างเช่น ถ้าต้องการรับสัญญาณอนาลอก ที่มีขอบเขต 0.5 V ถึง 3.5 V ซึ่งมีขนาดของช่วง เท่ากับ 3 V จะต้องทำการ ป้อน Voltage ที่ขา Vin(-) ในระดับ 0.5 V และทำการป้อน Voltage ที่ขา $V_{REF}/2$ ในระดับ 1.5 V

ในที่นี้จะรับขอบเขตของสัญญาณอนาลอกในขอบเขต 0 V ถึง 5 V จึงต้องต่อ Vin(-) เข้ากับกราวด์ และป้อนแรงดัน 5 V ให้กับ $V_{REF}/2$

ส่วนของการเปลี่ยนแปลงสัญญาณดิจิตอลเป็นสัญญาณอนาลอก

ในการแปลงสัญญาณดิจอตอลที่บันทึกไว้กลับเป็นสัญญาณอนาลอกตามเดิมจะใช้ IC เบอร์ DAC 0800 ซึ่ง IC จะรับสัญญาณดิจิตอลจากพอร์ตของ 8255 แล้วแปลงเป็นสัญญาณอนาลอกในขอบเขต 0-5 V

ลักษณะของวงจรในส่วนนี้แสดงได้ดังรูปที่ 7

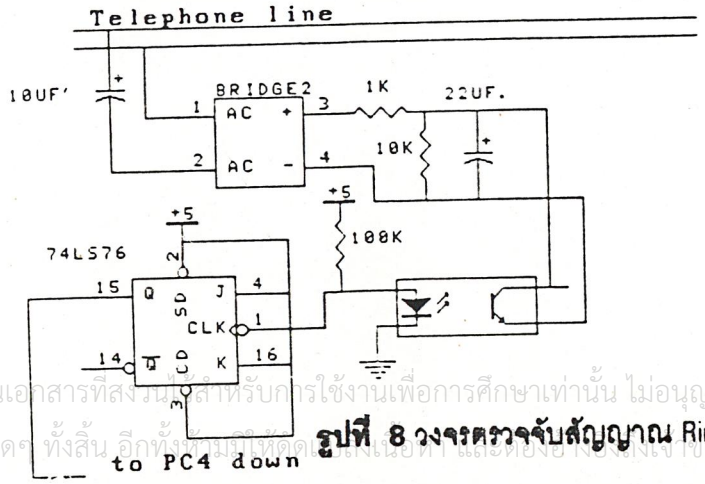


รูปที่ 7 วงจรการใช้งาน DAC 0800

ส่วนตรวจจับสัญญาณ Ringing และส่วนยก-วางหูโทรศัพท์

การยกหูโทรศัพท์อาศัยพื้นฐานว่าในขณะสายว่างคู่สายโทรศัพท์จะมีแรงดันประมาณ 48 โวลท์ ซึ่งจ่ายมาจากชุมสายโทรศัพท์ และเมื่อมีผู้เรียกเข้ามาทางชุมสายโทรศัพท์จะจ่ายสัญญาณกระดิ่งมา มีขนาดแรงดันไฟฟ้าเป็น 120 โวลท์ ความถี่ 20 เฮิร์ต โดยส่ง 1 วินาที หยุด 4 วินาที เป็นจังหวะ แบบนี้ซึ่งแรงดันนี้จะทำให้กระดิ่งในเครื่องโทรศัพท์ทำงาน และทางชุมสายจะรับทราบการยกหูโทรศัพท์จากการที่เรายกหูโทรศัพท์ซึ่งสวิทช์ภายในเครื่องโทรศัพท์จะทำการต่อคู่สายเข้ากับวงจรภายในที่มีความต้านทางกระแสตรงต่ำ ทำให้แรงดันไฟฟ้าลดลงเหลือ 6-10 โวลท์ เมื่อชุมสายรู้แล้วจึงต่อคู่สายของเรากับผู้เรียกเข้าด้วยกัน

รายละเอียดของวงจรแสดงได้ดังรูปที่ 8



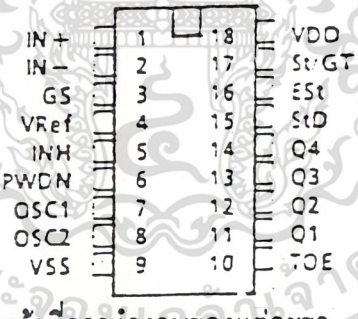
รูปที่ 8 วงจรตรวจจับสัญญาณ Ringing

จากรูป เมื่อสัญญาณ Ringing ถูกส่งมาตามคู่สาย จะผ่านวงจร Bridge Rectifier ซึ่งสัญญาณที่อยู่ในในรูปกระแสสลับจะถูกจัดเรียงใหม่ให้อยู่ในทิศทางเดียว หลังจากนั้นสัญญาณจะผ่าน filter เพื่อกรองให้ได้สัญญาณมีความเรียบขึ้น ซึ่งสัญญาณที่ได้จะมีลักษณะเป็นสัญญาณ pulse สัญญาณ pulse ที่ได้จะถูกนำไปผ่าน opto isolator (4N36) ซึ่ง opto isolator จะทำหน้าที่แยกวงจรของคู่สายออกจากวงจรภายในเมื่อสัญญาณ pulse ผ่าน poto isolator แล้วจะถูก inverse ซึ่งจะไปเซ็ทค่าของ JK Fip-Flop ทำให้ขา Q มี logic เป็น 1ซึ่งจะตรวจพบโดย CPU เมื่อพบแล้ว CPU จะทำการยกหูโทรศัพท์ โดยส่งสัญญาณมาควบคุมให้ relay ทำงานทำให้วงจรภายในสามารถเชื่อมต่อกับวงจรคู่สายได้

ส่วนของการถอดรหัสสัญญาณ DTMF

เนื่องจากโครงการชิ้นนี้ต้องมีการติดต่อระหว่างคอมพิวเตอร์และในการที่ ผู้ใช้จะติดต่อกับเครื่องคอมพิวเตอร์ จึงต้องทำการส่งผ่านทาง ปุ่มโทรศัพท์ โดยที่เราสามารถใช้ IC 8870 เป็นตัวที่ทำการถอดรหัส ให้กลายเป็น ตัวเลขฐานสอง 4 บิท

1. รายละเอียดของ IC MT8870



-รายละเอียดและหน้าที่การทำงานของแต่ละขา

- IN+ นอนอินเวอร์ติ้งออปแอมป์อินพุต (Non-Inverting)
- IN- อินเวอร์ทติ้งออปแอมป์อินพุต (Inverting)
-GS เป็นขาที่ฟีดแบค (Feedback) จากเอาท์พุท ให้ผู้ใช้ได้เชื่อมตัวความต้านทานเพื่อเลือกค่าในเกน(Gain) ได้ตามต้องการ
- VREF ค่าความต่างศักย์เปรียบเทียบที่เอาท์พุท ปกติ มีค่า $V_{dd}/2$
- IC* มีการเชื่อมต่อภายในจะถูกต่อเข้ากับ V_{SS}
- OSC1 อินพุทของสัญญาณนาฬิกา
- OSC2 เอาท์พุทของสัญญาณนาฬิกา โดยใช้กับผลึก (Crystal) ที่มีความถี่ 3.5795 MHz ต่อเข้าระหว่าง OSC1 และ OSC2 เพื่อจ่ายให้กับ ออสซิลเลเตอร์ (Oscillor) ที่อยู่ภายใน

- V_{SS} ไฟเลี้ยงขั้วลบ
- Q1-Q4 เป็นขาเอาท์พุทที่ถูกควบคุมแบบ 3 State ด้วยสัญญาณ TOE จะได้ค่าออกมาเมื่อได้รับค่าความถี่ที่ถูกต้อง และเอาท์พุทถูกอินาเบิล
- StD Delay Steering output จะให้ค่าลอจิก เป็น 1 เมื่อมีข้อมูลระดับแรงดัน
- EST Early Steering output จะให้ค่าลอจิก .1. เมื่อมีข้อมูลถูกต้องและถ้าเป็น 0 จะมี ไม่มีสัญญาณผ่านเข้ามา
- ST/GT Steering input/time output (bi-diection) ทำหน้าที่ส่งสัญญาณควบคุมวงจร RC ภายนอกเพื่อควบคุมไทม์
- V_{DD} ไฟเลี้ยงบวก

-จากรูปสามารถแบ่งได้เป็น

1. เราจะได้ค่าเกนเท่ากับ ลบ เนื่องจาก เราต่อความต้าน 100 K 2 ตัว ค่าที่ขา 2 และ 3 เป็นความ ต้านทานที่ถูกต่อแบบการป้อนกลับแบบลบ (Negative feedback)
2. V_{REF} ต่อเข้าขา IN+ เพื่อให้ระดับสัญญาณที่เข้ามาถูกไบอัสให้อยู่ที่กึ่งกลางของระดับไฟเลี้ยงเท่ากับ $V_{dd}/2$
3. คริสตัล 3.579575 ถูกต่อคร่อมระหว่าง OSC1 และ OSC2 เพื่อให้เกิดการผลิตสัญญาณนาฬิกาภายใน ไอซี

-ค่าความต้านทาน 300 K และค่าความจุตัวเก็บประจุ 0.1 μF ทำหน้าที่ในการกำหนด GUARD TIME ซึ่งเราสามารถที่จะเปลี่ยนแปลงเพิ่มหรือลดค่าของ guard time ได้ดังสมการ

$$T_{TGA} = (RC)\ln[V_{dd}/V_{tt}]$$
$$TGTP = (RC)\ln[V_{dd}/(V_{dd} - V_{Ttt})]$$

-ขา TOE ถูกทำให้ enable บัฟเฟอร์แบบ 3 state ของ q1-q4 ตลอดเวลา

ตารางแสดงความถี่ค่าต่าง ๆ ที่ถูก decode ออกมาได้

F_{LOW}	F_{HIGH}	NO	TOE	Q4	Q3	Q2	Q1
697	1209	1	H	0	0	0	0
697	1336	2	H	0	0	1	0
697	1477	3	H	0	0	1	1
770	1209	4	H	0	1	0	0
770	1336	5	H	0	1	0	1
770	1477	6	H	0	1	1	0
852	1209	7	H	0	1	1	1
852	1336	8	H	1	0	0	0
852	1477	9	H	1	0	0	1
941	1339	0	H	1	0	1	0
941	1209	*	H	1	0	1	1
941	1477	#	H	1	1	0	0
697	1633	A	H	1	1	0	1
770	1633	B	H	1	1	1	0
852	1633	C	H	1	1	1	1
941	1633	D	H	0	0	0	0
-	-	ANY	L	Z	Z	Z	Z

จากแผนผังเวลาดังกล่าว สามารถอธิบายช่วงเวลาได้ดังนี้

A ช่วงเวลาของสัญญาณโทน (tone) มีค่าผิดพลาดเนื่องจากเวลาน้อยเกินไปเอาท์พุทจะไม่มีการเปลี่ยนแปลง

B โทน #n ถูกตรวจสอบพบ และช่วงเวลาที่ถูกต้อง สัญญาณโทนจะถูก decode และถูก แลตช์ (latch) ไว้ที่เอาท์พุท

C ไอซีตรวจสอบพบสัญญาณโทนสิ้นสุด เอาท์พุทยังคงรักษาข้อมูลเดิมไว้จนกว่าจะมีโทนใหม่ที่ถูกต้องเข้ามา

D เอาท์พุทถูกทำให้อยู่ในสภาวะไฮอิมพีแดนท์

F ช่วงเวลาของโทน #n+1 ถูกตรวจว่าสิ้นสุดแล้ว เอาท์พุทยังคงรักษาข้อมูลเดิมไว้จนกว่าจะมีข้อมูลใหม่ที่ถูกต้องเข้ามา

สำหรับสัญลักษณ์ต่าง ๆที่ใช้มีความหมายดังนี้

t_{REC} คือ ช่วงเวลาอย่างน้อยที่สุดที่โทนถูกตรวจพบ (DETECT)ได้อย่างถูกต้อง

t_{ID} คือ ช่วงเวลาอย่างน้อยสุดระหว่างโทนแต่ละครั้งที่ถูกกดเข้ามา

t_{DO} คือ ช่วงเวลามากที่สุดที่ยอมรับได้ในกรณีที่สัญญาณโทนถูกทำให้หายไปชั่วขณะ

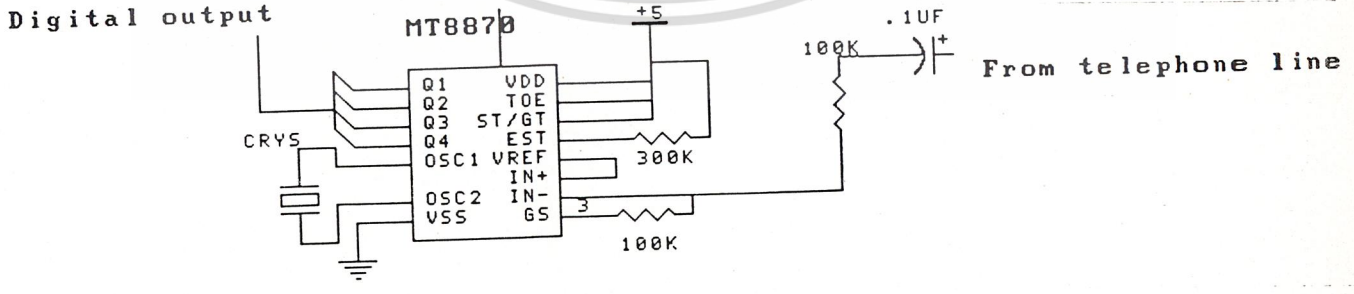
t_{DP} คือ เวลาที่ตรวจสอบพบการมาของสัญญาณที่ถูกต้อง

t_{DA} คือ เวลาที่ตรวจสอบพบการหายไปของสัญญาณที่ถูกต้อง

t_{GTP} คือ สัญญาณ guard time ปรากฏ

t_{GTA} คือ สัญญาณ guard time ไม่ปรากฏ

2. วงจรรวมที่ใช้งานจริง



รูปที่ 9 วงจรรวมที่ใช้งานจริง

- จะรับสัญญาณมาจากสายโทรศัพท์

- จากวงจร จะทราบ GARD TIME คือ

$$T_{TCA} = (RC)\ln[V_{cc}/V_{ss}]$$

$$= 0.048 \text{ sec}$$

- ค่า OUTPUT ที่ได้ จะนำไปเข้า COMPUTER เพื่อนำไปคำนวณ ตัวเลขในการทำ FUNCTION ต่าง ๆ ซึ่งใน IC ตัวนี้สามารถ latch ค่าไว้ที่ขาได้ตลอด จนกว่าจะมีตัวใหม่เข้ามา

- การที่จะตรวจสอบว่า ค่านั้นจะสะกดได้หรือยัง สามารถตรวจสอบได้จาก ขา SDT ซึ่งถ้ามีรหัสตัวใหม่เข้ามา จะเกิด PULSE 1 ลูก

ส่วนสร้างสัญญาณ DTMF

เนื่องจากโทรศัพท์แบบ TONE (แบบกดปุ่ม) จะมีการทำงานโดยถ้าเรากดปุ่ม เครื่องจะทำงานสร้างความถี่ขึ้น 2 ชนิดแล้วทำการรวมสัญญาณ เพื่อส่งไปตามสายโทรศัพท์เข้าชุมสาย โดยมีความถี่ดังนี้

	1209	1336	1477	1633	
697	1	2	3	A	R_1
770	4	5	6	B	R_2
852	7	8	9	C	R_3
941	*	0	#	D	R_4
	C_1	C_2	C_3	C_4	

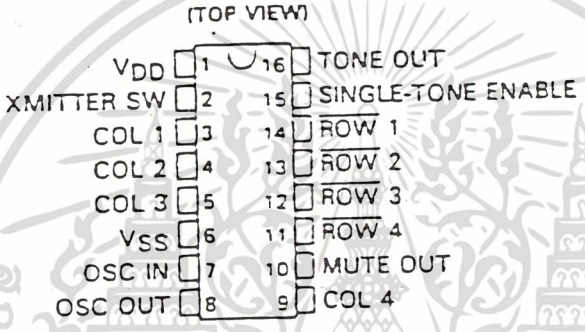
แสดงความถี่ผสมที่ใช้ในโทรศัพท์แบบกดปุ่ม

เราจะใช้ IC TCM 5087 ในการสร้างความถี่ที่กล่าวมา

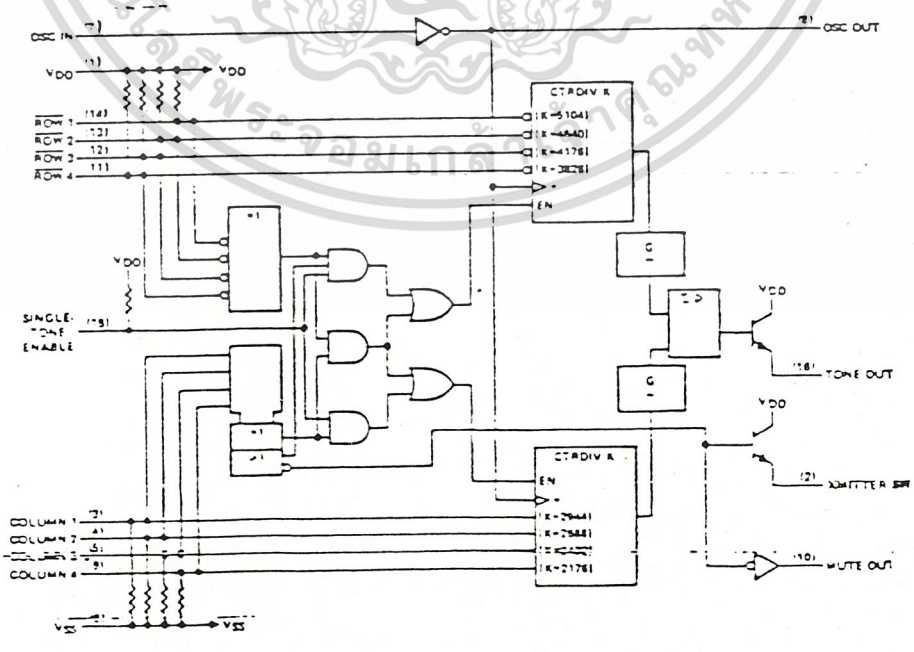
1.ส่วนของ IC TCM 5087

1.1 ส่วนประกอบของ IC TCM 5087

เป็น IC TONE ENCODER คือทำการสร้างสัญญาณความถี่ 2 ชนิดแล้วนำมาผสมกันออกมาเป็น OUTPUT โดยความถี่นั้นจะถูกกำหนดโดย ขา INPUT ทั้ง 8 ขา และต้องการความถี่ 3.579545 ที่ขา 7-8รายละเอียดดังรูปที่ 10



Diagrame การทำงานของ IC TCM 5087



รูปที่ 10 แสดงขาต่าง ๆ IC TCM 5087

รายละเอียดและหน้าที่ของแต่ละขา

- V_{DD} ไฟเลี้ยงบวก มีค่า $3.5 < V_{DD} < 10$

- XMITTERSWจะเป็น High IMPEDENCE เมื่อขา columnใดก็ได้active และเป็น High เมื่อไม่มีขา columnใด ๆ active

- COL1-COL4 เป็นขา INPUT เพื่อกำหนดค่าความถี่ 1 ค่า active ที่ High

- V_{SS} เป็น Ground ของไอซี

- TONE OUT เป็นขา OUTPUT โดยจะประกอบด้วยความถี่ 2 ชนิดซึ่งได้จากการกำหนดที่ขา column และขา row

- SINGLE- TONE ENABLE เป็นขาที่กำหนดให้ OUTPUT ออกมาเพียงความถี่เดียว

-ROW1-ROW2เป็นขา INPUTเพื่อกำหนดค่าความถี่active ที่ low

- MUTE OUT จะเป็น High เมื่อ COLUME ใดก็ได้ active และเป็น low เมื่อไม่มีขา colume ใดเลยที่ active

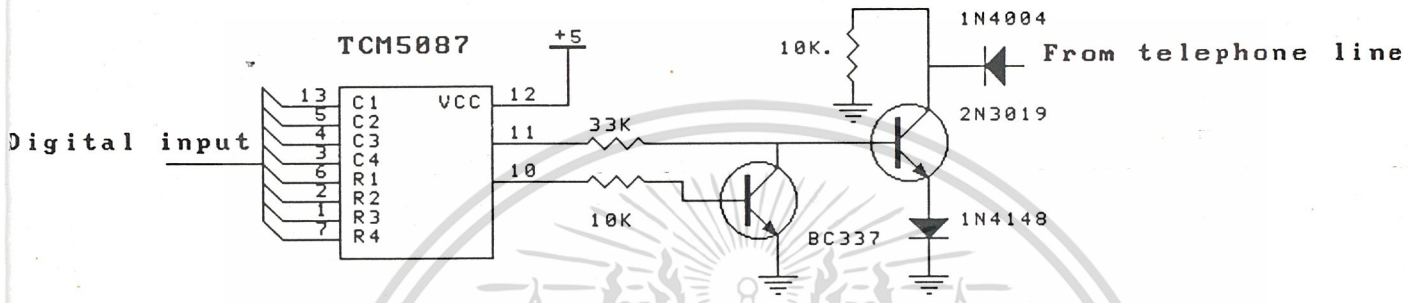
การส่งข้อมูลให้ IC สร้างความถี่ให้เท่ากับที่เรากดโทรศัพท์สามารถส่งค่าให้กับขาต่าง ๆ ที่ขา $R_1 - R_4$ จะทำงาน (active) ที่ Logic 0 ส่วน $C_1 - C_4$ จะทำงาน (active) ที่ Logic 1 การหมุนเบอร์จะต้องกำหนดค่าให้กับขาต่าง ๆ ดังนี้

เบอร์	R_1	R_2	R_3	R_4	C_1	C_2	C_3	C_4	ฐาน 16
1	1	1	1	0	0	0	0	1	E1
2	1	1	1	0	0	0	1	0	E2
3	1	1	1	0	0	1	0	0	E4
4	1	1	0	1	0	0	0	1	D1
5	1	1	0	1	0	0	1	0	D2
6	1	1	0	1	0	1	0	0	D4
7	1	0	1	1	0	0	0	1	B1
8	1	0	1	1	0	0	1	0	B2
9	1	0	1	1	0	1	0	0	B4
0	0	1	1	1	0	0	1	0	72
ปกติ	1	1	1	1	0	0	0	0	F0

ข้อมูลที่จะส่งให้ IC TCM 5087

ข้อมูลที่จะส่งให้ IC TCM 5087

1.2 วงจรที่ใช้ทำงาน



รูปที่ 11 วงจรที่ใช้งาน

ส่วน Software

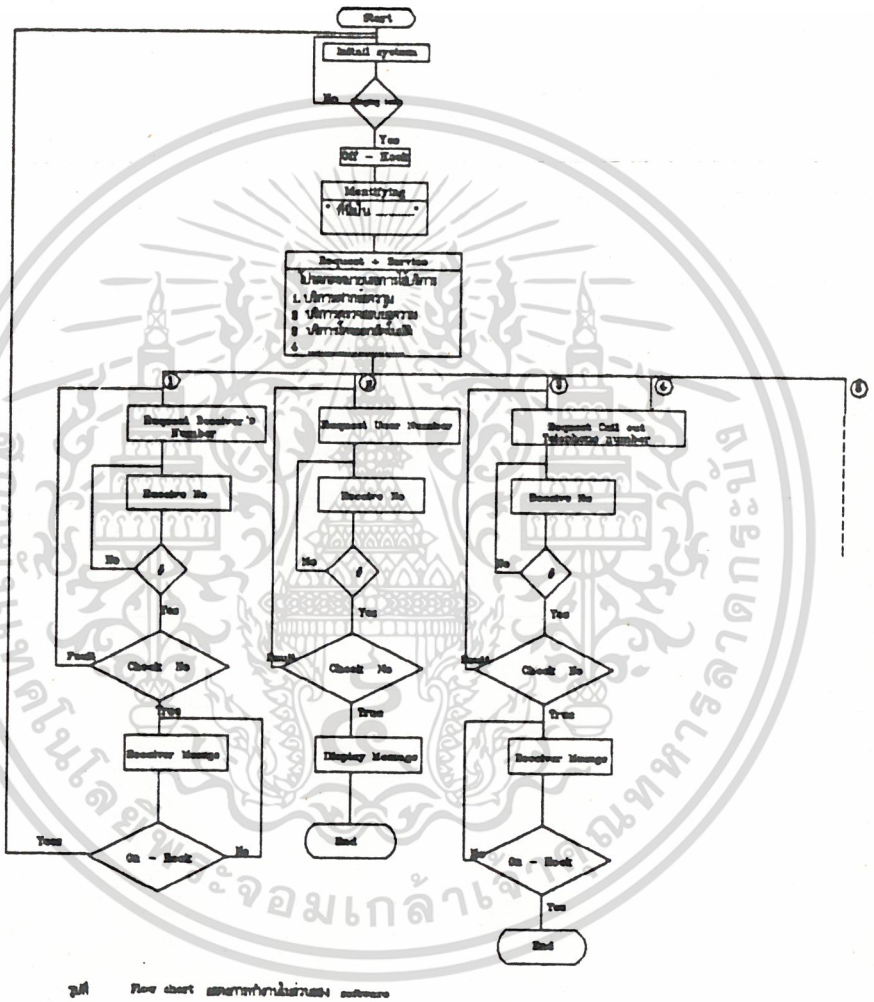
เมื่อได้ทำการออกแบบทาง hardware แล้ว จะต้องมีการออกแบบการควบคุมโดยใช้ software ซึ่งจะสามารถควบคุม hardware ให้ทำงานได้ตามความต้องการในโครงการนี้จะควบคุม hardware ให้ทำงานในลักษณะของ Voice mail box และได้กำหนดฟังก์ชันการทำงานอื่น ๆ ไว้อีกคือ

-ในการให้บริการโทรศัพท์อัตโนมัติ คือในกรณีที่ผู้เรียกต้องการส่งข้อความไปเตือนบุคคลหนึ่งในเวลาหนึ่งซึ่งเป็นเวลาที่ผู้เรียกไม่พร้อมที่จะโทรออก ผู้เรียกสามารถฝากข้อความไว้กับเครื่อง และตั้งเวลาที่จะโทรออกได้

-การให้บริการสอบถามข้อมูล เช่นการสร้างเวลา การแจ้งผลสอบ

การทำงานของ Software

CPU จะถูกโปรแกรมให้สามารถตอบรับโทรศัพท์ได้เมื่อมีผู้โทรเข้ามายังเครื่องคอมพิวเตอร์ โดย CPU จะส่งข้อความในรูปของเสียงพูดเพื่อแสดงว่าตังว่างเบอร์ที่ ผู้เรียกโทรเข้ามาเป็นหน่วยรับฝากข้อความ และทำการสอบถามว่าผู้เรียกต้องตอบโต้ได้โดยการกดหมายเลขทางหน้าปัทม์โทรศัพท์เพื่อแจ้งการบริการที่ผู้เรียกนั้น ๆ เช่น ตัวผู้เรียกกดหมายเลข 1 software จะไปทำงานที่ routineของการรับฝากข้อความ ตัวกดหมายเลข 2 จะทำงานที่ routine ของการตรวจสอบข้อความ ขั้นตอนต่าง ๆ จะถูกแสดงในรูปที่ 12



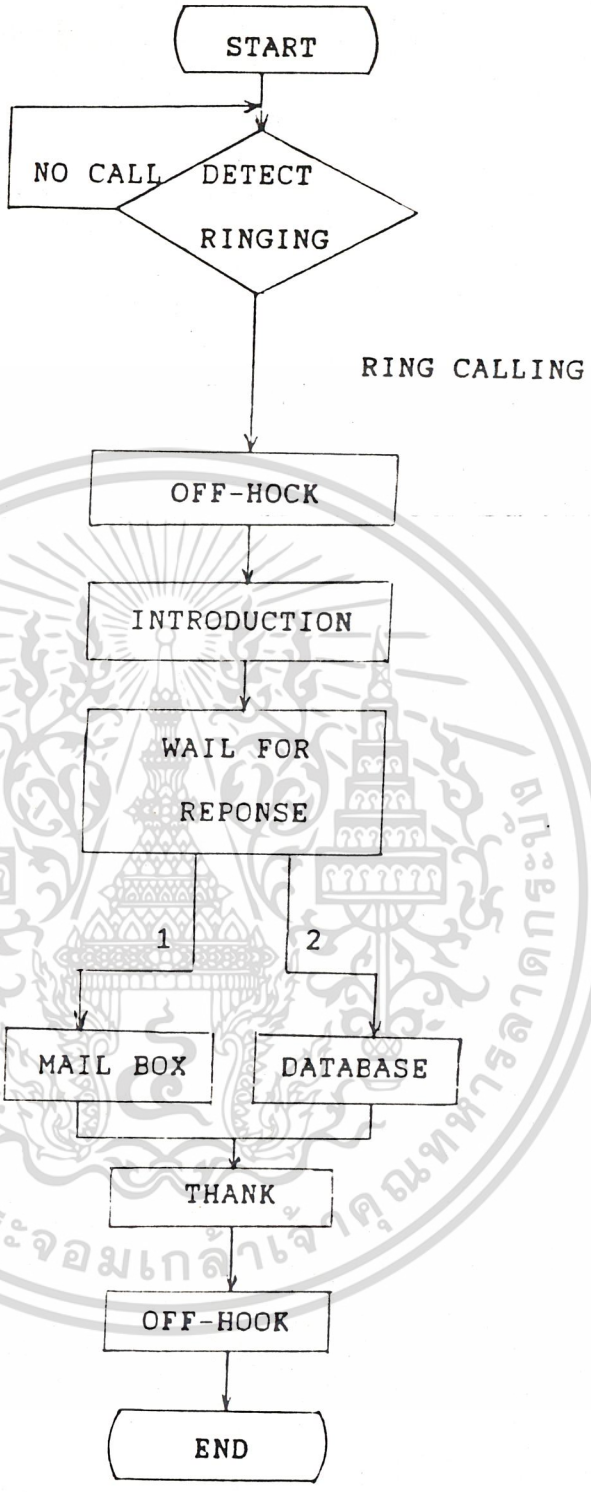
รูปที่ 12 Flow chart แสดงการทำงานในส่วนของ Software

การทำงานของ SOFT WARE จะแบ่งออกเป็น 3 ส่วนย่อย ดังนี้

1. OPERATION
2. MAIL BOX
3. DATA BASE

ส่วนของการปฏิบัติการ (Operation)

โปรแกรมจะทำการตรวจจับสัญญาณ Ringing เมื่อพบสัญญาณจึงทำการยกหูโทรทัศน์ จากนั้นจึงกล่าวคำทักทายกับผู้โทรเข้ามาและแจ้งให้ผู้โทรเลือกใช้การบริการว่าจะใช้บริการฝากข้อความ (MAIL BOX) บริการสอบถามข้อมูล (DATA BASE) หลังจากใช้บริการเสร็จแล้วโปรแกรม จะทำการกล่าวคำอำลาและวางหูโทรศัพท์ จึงเป็นอันสิ้นสุดการปฏิบัติการ



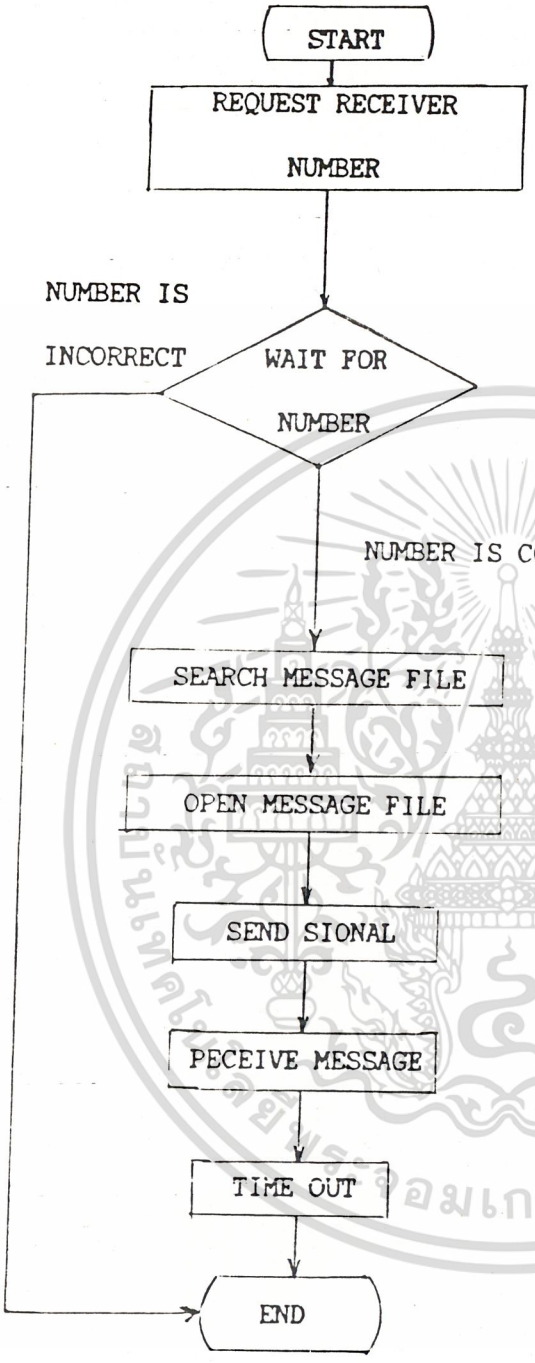
รูปที่ 13 FLOW CHART แสดงการทำงานของ SOFTWARE

ส่วนของบริการฝากข้อความ (VOICE MAIL BOX)

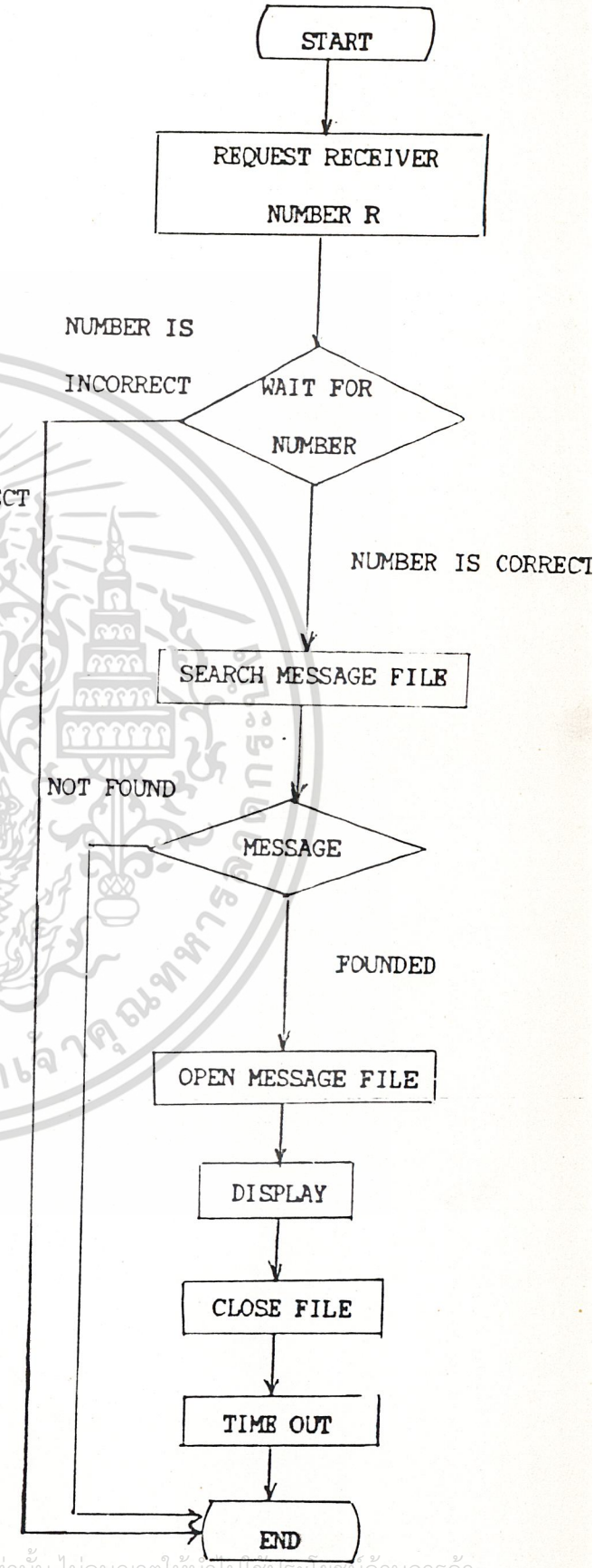
ในการบริการส่วนนี้ทางผู้ควบคุมเครื่องคอมพิวเตอร์ จะทำการกำหนดไฟล์ของผู้ใช้บริการ โดยกำหนดหมายเลขประจำตัวของผู้ใช้บริการ และกำหนดไฟล์สำหรับฝากข้อความให้กับผู้ใช้

เมื่อมีผู้เรียกเข้ามา โปรแกรมจะแจ้งให้ผู้เรียกกดรหัสของผู้รับข้อความเมื่อโปรแกรมได้รับรหัสแล้วจะไปทำการค้นหาไฟล์ของผู้รับมา จากนั้นจึงให้ผู้เรียกฝากข้อความได้ในเวลาที่กำหนด จากนั้นโปรแกรมจึงทำการปิดไฟล์

เมื่อผู้รับโทรมาตรวจสอบข้อความ โปรแกรมจะแจ้งขอรหัสส่วนตัวของผู้รับ เมื่อได้รับรหัสแล้วโปรแกรมจะทำการค้นหาไฟล์ข้อความของผู้รับมาแสดงให้ทราบแล้วการลบข้อมูลออกจึงเป็นการสิ้นสุดการบริการฝากข้อความ



รูปที่ 14 FLOW CHART แสดงการทำงานของ VOICE MAIL BOX ในการรับฝากข้อความ

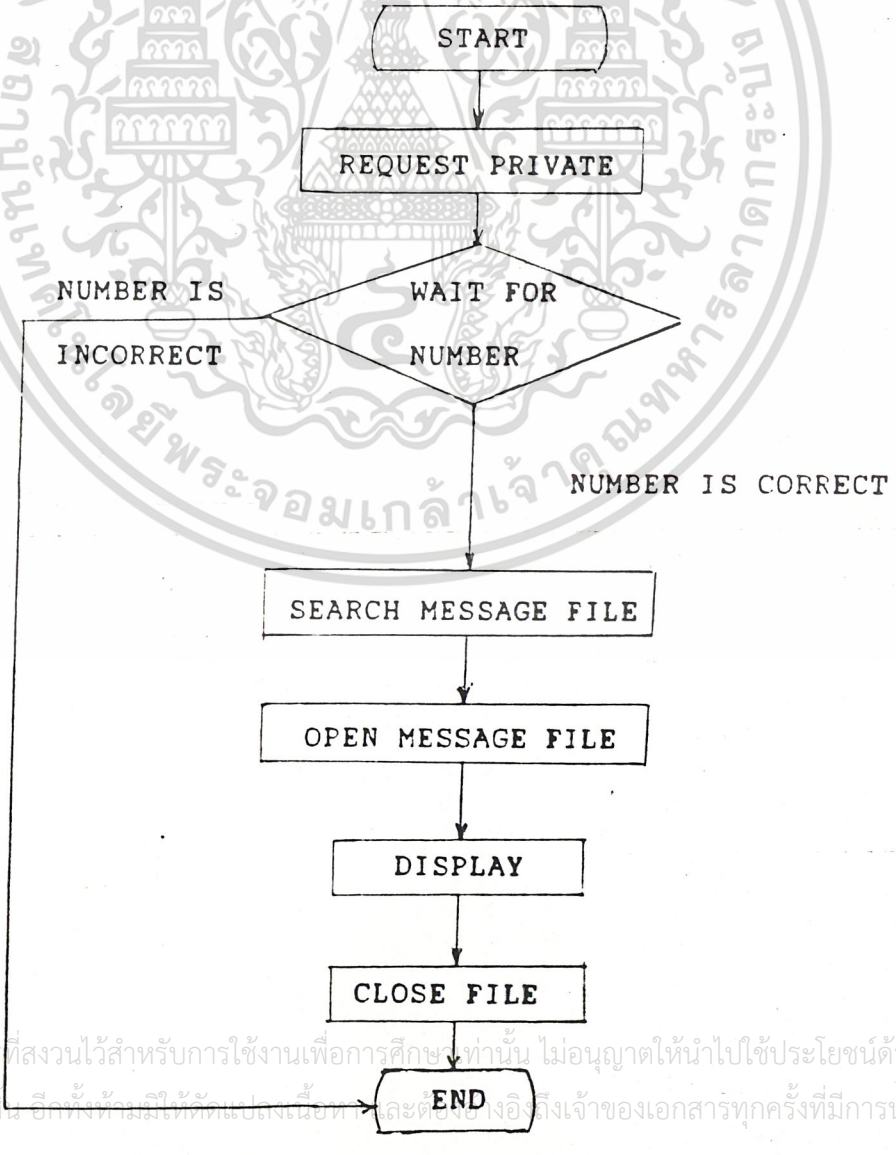


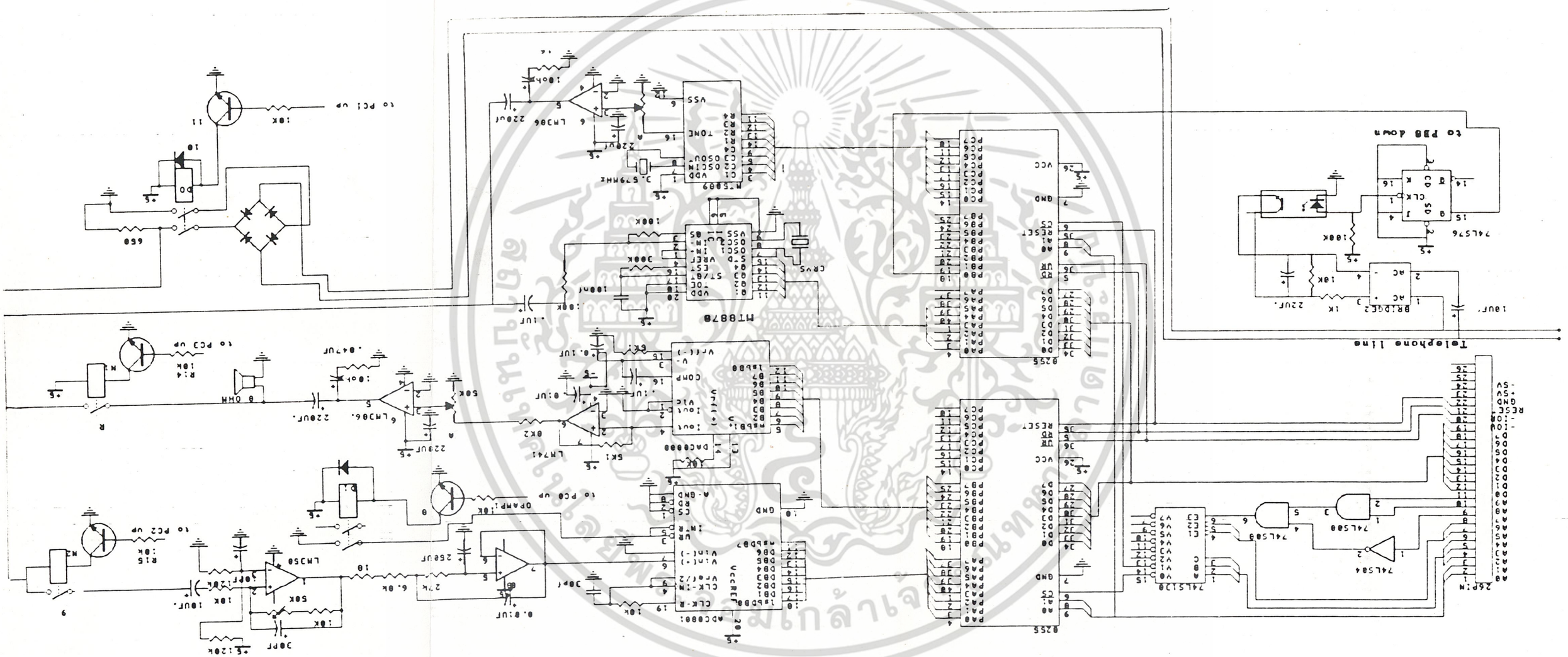
รูปที่ 15 FLOW CHART VOICE MAIL BOX ในการแสดงข้อความ

ส่วนของการบริการสอบถามข้อมูล (DATA BASE)

อาจนำไปให้บริการในการสอบถามข้อมูล เช่น ผลการสอบต่าง ๆ โดยผู้ควบคุมคอมพิวเตอร์ จะทำการเปิดไฟล์ให้กับผู้ใช้บริการโดยการกำหนดหมายเลขผู้ใช้บริการโดยอาจจะเป็นหมายเลขที่ใช้ในการสอบหรือรหัสประจำตัว จากนั้นจึงทำการบันทึกเสียงในการรายงานผลการสอบให้กับผู้ใช้บริการแต่ละคน

เมื่อผู้ใช้บริการต้องการทราบผลการสอบ ก็จะทำการโทรเข้ามาตรวจสอบโปรแกรมจะเรียงของรหัสส่วนตัวของผู้สอบ เมื่อผู้โทรกดรหัสส่วนตัวทางแป้นโทรศัพท์แล้ว ทางโปรแกรมจะทำการรับรหัสมา แล้วจึงไปค้นหาไฟล์ของผู้สอบเพื่อนำมาแสดงผลให้ทราบ จึงเป็นอันสิ้นสุดการบริการ





บทที่ 5

การตรวจสอบการทำงานของวงจร

ในการตรวจสอบว่าวงจรในส่วนต่าง ๆ สามารถจะทำได้โดยการใช้ซอฟท์แวร์ที่เหมาะสมและสอดคล้องกับฮาร์ดแวร์ ในที่นี้จะทำการแยกทดสอบเป็นส่วนต่าง ๆ ได้ดังนี้

การตรวจสอบการทำงานของส่วนการแปลงสัญญาณอนาลอกเป็นดิจิตอล ทำได้โดยการป้อนสัญญาณอนาลอกที่ระดับต่าง ๆ กันเข้าที่ ขา oooo ของ ADC0802 จากนั้นจึงทำการ RUN โปรแกรมเพื่อรับข้อมูลเข้ามาในเครื่องคอมพิวเตอร์เพื่อพิจารณาค่า ซึ่งโปรแกรมที่ใช้ทดสอบจะแสดงได้ดังนี้

```
mian ( )
{
outport b (0 x 303  , 0 x 801 ) ; initial  port
outport b (0 x 302  , 0 x 001 ) ; Set PCo ให้เป็น hing
outport b (0 x 302  , 0 x 000 ) ; Set PCo  เป็น low
a= inport b (0 x 300 )  ; นำข้อมูลเข้า printf ("Data is % b",a ); แสดงข้อมูล
 }
```

เมื่อได้รับข้อมูลแล้วจะนำข้อมูลที่ได้มาเปรียบเทียบกับข้อมูลที่ได้จากการคำนวณซึ่งการคำนวณ จะใช้หลักที่ว่า ขอบเขตของสัญญาณอนาลอกที่ได้รับโดย ADC0802 จะอยู่ในช่วง 0-5 โวลท์ และจะมีการเปลี่ยนแปลงไปเป็นข้อมูลดิจิตอลขนาด 8 บิท ซึ่งมีระดับการควอนไตซ์ 256 ขั้น ดังนั้นแต่ละขั้นจะมีระยะห่างเป็น $\frac{5}{256} \approx 0.02$ โวลท์

ค่าที่ได้จากการทดลอง และค่าที่ได้จากการคำนวณ จะแสดงได้ดังตาราง

input	ค่าที่ได้จากการคำนวณ	ค่าที่ได้จากการทดลอง
0.0	00000000	00000000
0.2	00001010	00001111
0.4	00010100	00011001
0.6	00011110	00100011
0.8	00101001	00101110
1.0	00110011	00111000
1.2	00111101	01000110
1.4	01000111	01001100
1.6	01010010	01010111
1.8	01011100	01100100
2.0	01100110	01101111
2.2	01110000	01111011
2.4	01111010	10000111
2.6	10000101	10000101
2.8	10001111	10010011
3.0	10011001	10100010
3.2	10100011	10101000
3.4	10101101	10101000
3.6	10111000	10111101
3.8	11000010	11000111
4.0	11001100	11010001
4.2	11010110	11011011
4.4	11100000	11100101
4.6	11101011	11110000
4.8	11110101	11111010
5.0	11111111	11111111

จากตารางจะเห็นว่าค่าที่ได้จากการทดลองจะมีค่าความคลาดเคลื่อนจากค่าที่คำนวณเล็กน้อย ความคลาดเคลื่อนมักจะเกิดที่บริเวณ 4 บิท ล่างซึ่งจะมีผลทำให้ค่าความคลาดเคลื่อนเล็กน้อย ซึ่งความผิดพลาดนี้ส่วนหนึ่งจะเกิดจากความคลาดเคลื่อนของแหล่งจ่ายไฟ และส่วนหนึ่งจะเกิดจากตัว IC เอง

การตรวจสอบการทำงานในส่วนของการแปลงสัญญาณดิจิตอลเป็นสัญญาณอนาลอก

จะทำในลักษณะเดียวกันกับส่วนที่แล้ว แต่จะมีทิศทางกลับกัน คือ จะส่งข้อมูลดิจิตอลออกไปทางพอร์ต ของ 8255 เพื่อส่งข้อมูลไปให้DAL0800 เพื่อแปลงกลับเป็นสัญญาณอนาลอก โดยใช้โปรแกรมดังนี้

```
main ( )
{
  outport b (0 x 303 , 0 x 800) ; intial  port
  outport b (0 x 301 , Data   ) ; ส่งค่าออก
}
```

จากการทดลองได้ทำการทดลองส่งค่าต่าง ๆ ออกไปโดยเริ่มตั้งแต่ -6-6 โวลท์ตามคุณสมบัติของ IC ถึงแม้ว่าจะมีการคลาดเคลื่อนของขอบเขต 2 โวลท์เพราะจริง ๆ แล้วควรจะเป็น -5-5 โวลท์ ตามคุณสมบัติของ IC

ถึงแม้ว่าจะมีการคลาดเคลื่อนไปบ้างแต่ก็ยังไม่มีผลกระทบกับสัญญาณเอาร์พุทเท่าไรนัก เพราะว่าระหว่างขั้นยังมีค่าคงที่

ผลการทดลองแสดงได้ดังตาราง

ข้อมูลที่ส่งไป DAC0800	สัญญาณอนาลอก OUTPUT
00000000	-6
00111000	-5
01000111	-4
01010111	-3
01100100	-2
01110011	-1
10000000	0
10001110	+1
10011101	+2
10101100	+3
10111010	+4
11001001	+5
11111111	+6

จากการทดลองผู้ทดลองได้พยายามปรับค่าเลขฐานสองให้ได้เอาท์พุทออกมาเป็นโวลท์เตท ในรูปจำนวนเต็มเพื่อให้ง่ายต่อการวัด

จากการทดลองจะสังเกตุได้ว่าข้อมูลค่าต่ำสุด คือ 00000000 จะให้เอาท์พุทที่ต่ำสุดคือ -6 โวลท์และข้อมูลค่ามากสุด คือ 11111111 จะให้เอาท์พุทสูงที่สุด คือ 6 โวลท์

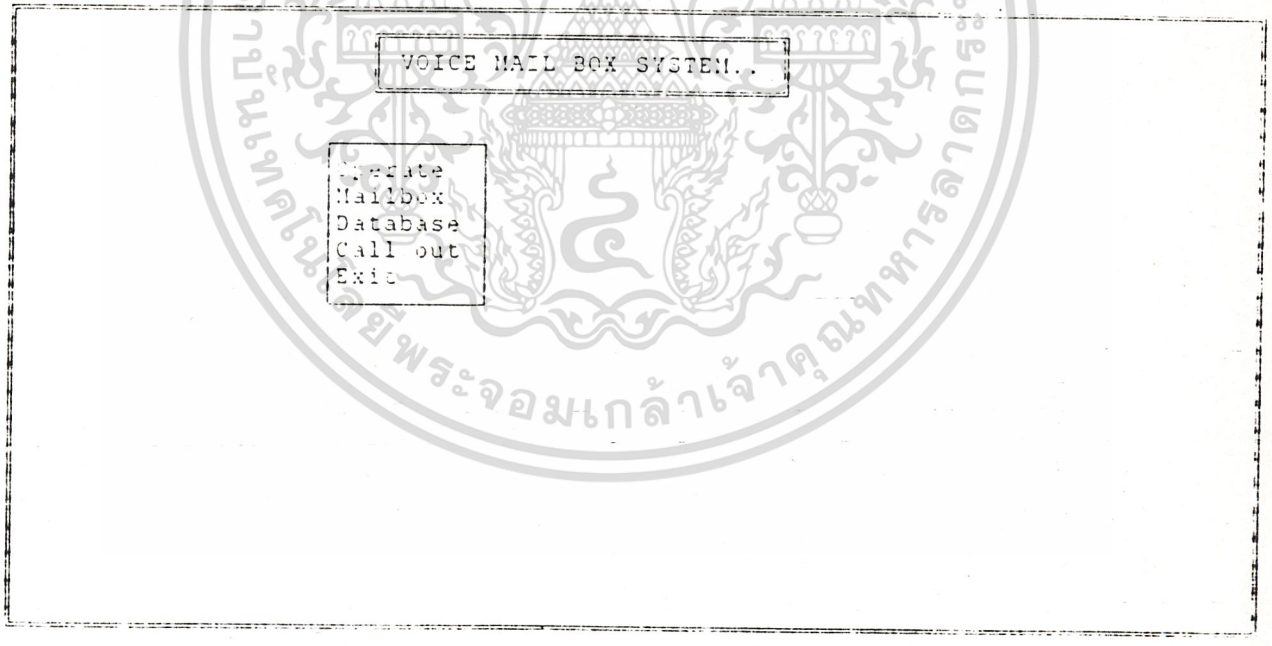
ดังนั้นจึงสามารถสรุปได้ว่าส่วนของการแปลงสัญญาณอนาลอกเป็นสัญญาณดิจิตอล สามารถทำงานได้อย่างเหมาะสมแม้จะมีการคลาดเคลื่อนจากความเป็นจริงไปบ้างก็ตาม

ผลการทดลอง

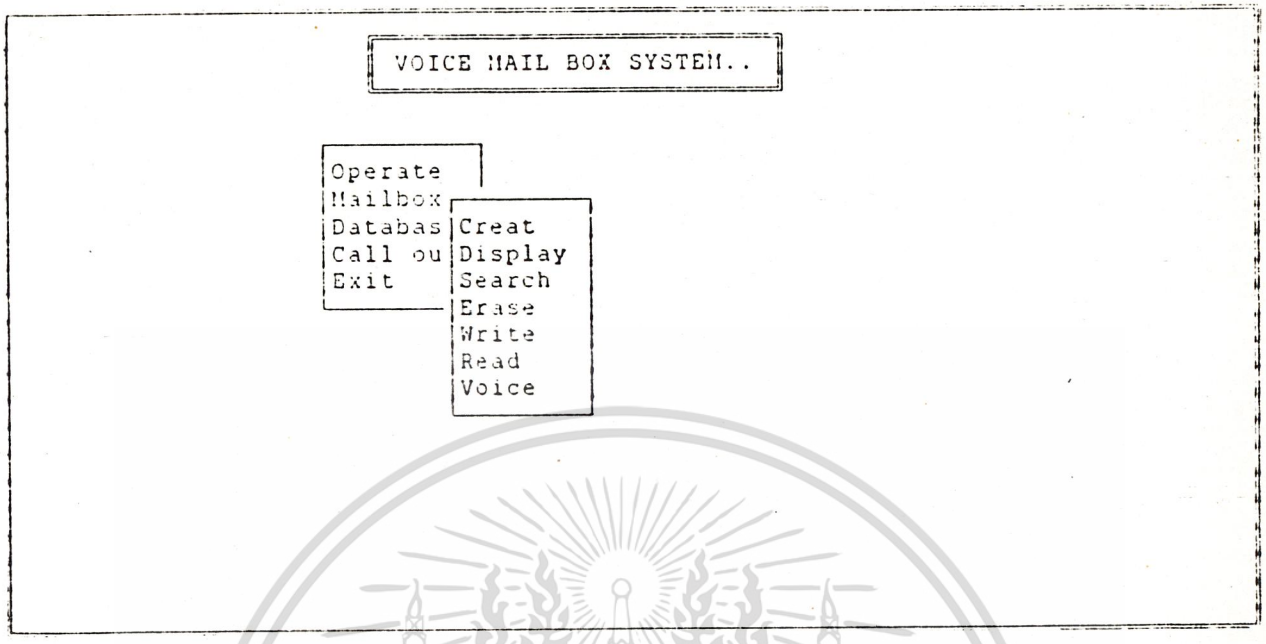
ลำดับการทดลอง

1. นำการ์ดไปต่อเข้ากับสายโทรทัศน์ และนำไปเสียบที่ SLOT ของเครื่องคอมพิวเตอร์

2. ทำการเปิดเครื่องและ RUN โปรแกรม MYPROJECT. EXE จะปรากฏที่หน้าจอของคอมพิวเตอร์ ดังรูป

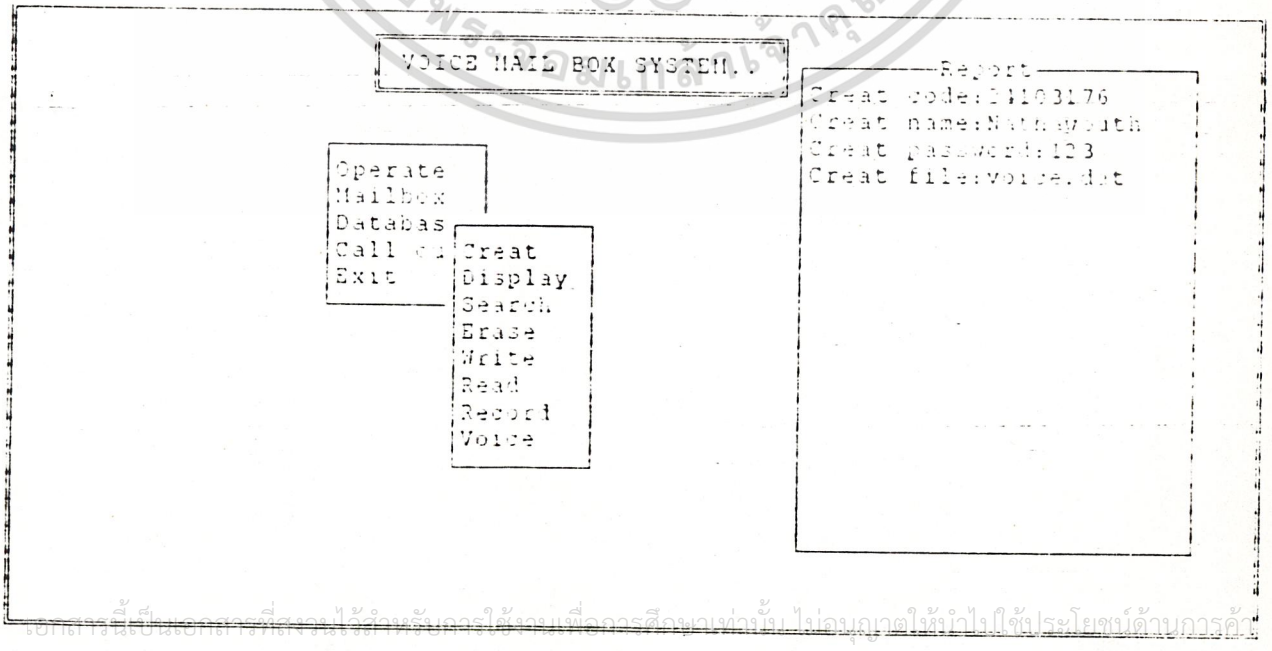


3. เมื่อต้องการให้บริการรับฝากข้อความให้เลือก MENU MAIL BOX จะปรากฏ MENU ย่อยอีกดังรูป



จากนั้นจึงเลือก MENU CREAT เพื่อสร้าง FILE ให้กับผู้ใช้บริการตามที่ต้องการ

4. เลือก MENU OPERATE เพื่อให้ SOFTWARE เริ่มทำงาน ในขณะที่โปรแกรมทำงานจึงมีการแสดงผลออกมาทางหน้าจอ ดังรูป



5. เมื่อต้องการให้บริการสอบถามข้อมูลก็ให้เลือก MENU BASTE แล้วทำการสร้าง FILE ในลักษณะเดียวกันกับบริการ VOICE MAIL BOX แต่ในกรณีนี้จะต้องมีการบันทึกเสียงลงไปใน FILE ด้วย โดยการเลือก MENU RECORD

6. เมื่อทำการสร้าง FILE เสร็จแล้วจึงทำการตรวจสอบการทำงานโดยการโทรศัพท์เข้ามายังเครื่อง PC และทำการติดต่อกับเครื่อง PC โดยกดปุ่มทางหน้าปัทม์โทรศัพท์ ส่วนเครื่อง PC จะติดต่อกับผู้โทรโดยใช้เสียงเราจึงสามารถใช้บริการ VOICE MAIL BOX หรือ VOICE DATA BASE ได้ตามความคาดหมาย

สรุปผลการทดลอง

อุปกรณ์ Voice Mail Box จะสามารถให้บริการทางด้านการรับฝากข้อความ หรือให้บริการด้านข้อมูลเสียง นอกจากนั้นยังสามารถนำมาใช้บริการอื่น ๆ อีกเช่น การเตือนเวลา การแจ้งเวลา ซึ่งสามารถเพิ่มการบริการได้ด้วยการปรับปรุง software โดยไม่ต้องแก้ไข Hardware แต่อย่างใด

ในโครงการนี้ผู้ทดลองได้สร้างโปรแกรมฐานข้อมูลขนาดเล็ก และเก็บข้อความเสียงลงในไฟล์และเก็บไฟล์ในลักษระ Double link list ซึ่งทำให้สามารถสร้างไฟล์เก็บไว้ได้เป็นจำนวนมาก และไม่สิ้นเปลือง Marddisk

แนวทางการพัฒนา

ในการทดลองได้ประสบปัญหา จากสัญญาณรบกวนที่มาจากเครื่อง Computer ซึ่งเป็นปัญหาที่หลีกเลี่ยงไม่ได้ เพื่อให้เสียงคมและชัดขึ้น ควรจะจัดหา filter ที่มีคุณภาพสูงมารองสัญญาณรบกวนออกไป Amplifier ที่ดีก็มีส่วนในการเพิ่มคุณภาพเสียงด้วย

นอกจากนี้ควรมีการพัฒนา Voice Mail Box ให้ใช้ได้ในระหว่างเครื่อง PC ด้วยกัน หรือนำมาใช้ในระบบ LAN โดยใช้ในลักษระเดียวกันกับ ELECTRONIN MAIL

```
#include<link.h>
#include<bg.h>
#include<ring.h>
int ser[1];
char vice[1];
void pd_driver()
 {
      int choice1,choice2,choice3,selection;
      struct member{
      char name[10];
      char code[10];
      char filename[5]; }person[5];
      struct channel{
      char user[10];
      char no[10];
      char pass[10];
      char file[5]; }chan[5];

     makewin();

     while((choice1=pulldown(0)) != -1) {
     switch(choice1) {
     case 0:          /*want to operate*/
      goto_xy(0,1);
      window(2);
      initial();
      ring_detect();
      off_hook();
      strcpy(UserFile,"Hello.dat");
      play();
      delay(500);
      strcpy(UserFile,"Sellect.dat");
      play();
      ser_select();
     if(strcmp(vice,"1")==0)
{      signal();
      ReadFile();
      delay(1000);
      window(4);
      SearchFile();
      delay(1000);
      window_puts(4,"I will open ");
      window_puts(4,UserFile);
      record();
      delay(2000);
      deactivate(4);       }
  else if(strcmp(vice,"2")==0)
{      signal();
     ReadFile();
     delay(1000);
     window(4);
     SearchFile();
     window_puts(4,"I will open ");
```

```
window_puts(4,UserFile);
play();
delay(2000);
deactivate(4);       }
deactivate(2);
break;
```

```
case 1:              /*mail box function*/
   while((choice2=pulldown(1)) != -1) {    /*show mail channel*/
    switch(choice2){
    case 0:                                  /*creat data*/

        CreatData();
        break;
    case 1:                                  /*display data*/

        DisplayData();
        break;
    case 2:                                  /*search record*/

         SearchData();
         break;
    case 3:                                  /*erase a record*/
         EraseData();
         break;
    case 4:                                  /*write the list*/
         WriteFile();
         break;
    case 5:                  /*read the list*/
         ReadFile();
         break;
    case 6:          /*voice*/
          play();
          break;
```

```
                    }   /*close sub switch*/
        }               /*close while loop*/
restore_video(1);
break;
case 2:                 /*data base function*/
while((choice2=pulldown(2))!=-1) {
  switch(choice2) {
   case 0:      /*creat data*/
            CreatData();
     break;
  case 1:
            DisplayData();     /*display record*/
  break;
  case 2:
            SearchData();                    /*search record*/
  break;
  case 3:
            EraseData();          /*erase a record*/
   break;
  case 4:           /*write a list*/
           WriteFile();
   break;
  case 5:
           ReadFile();  /*read the list*/
    break;
  case 6:     /*record*/
      record();
      break;
   case 7:
      play();
      break;          }

                                }     /*close while loop */


restore_video(2);
break;
case 4:
```

```
        exit(1);


                        }  /*close main switch*/


          }                /*close while loop*/
        restore_video(0);
        }                   /*close function*/

 void sel_service()
  { char ch;
    int a,b,c;
    int i=0;
 outportb(0x307,0x090);
 window(4);
 window_puts(4,"Wait for signal.");
 window_puts(4,"\n");
 do{
 a=inportb(0x304);
 b=a&31;
 if(b>=16)
 { c=b-16;

  ser[i]=c;
  window_puts(4,"*");

  i++;
  delay(200); }
    }while(i<1);
  window_puts(4,"\n");
    ch=*ltoa(ser[i],&ch,10);
    vice[i]=ch;
window_puts(4,"Ok!");
delay(2000);
 deactivate(4);
 }
```

```
   #include<stdio.h>
struct _person {
int nRecordType;
struct _person *NextPerson;
struct _person *PrevPerson;
char name[31];
char code[61];
char pass[26];
char mess[6];
char phone[31];
int nRecordNumber;
}PersonName;

struct _person *FirstPerson = NULL;
struct _person *LastPerson = NULL;
struct _person *Person = NULL;
struct _person *TempPerson = NULL;
char FileName[80]="ch";
char buffer[80];
int num[6];
char str[6];
int nRecord = 0;
```

```
void makewin()
{ make_window(0,"",30,17,50,22,1);
  make_window(1,"",52,17,72,22,1);
  make_window(2,"Operate",20,5,50,15,1);
  make_window(3,"Report",50,2,75,20,1);
  make_window(4,"",40,5,60,10,1); }
```

```
#include"dos.h"
void off_hook()
 { outportb(0x307,0x080);
  outportb(0x306,0xff);
  window_puts(2,"Off-hook already.\n");
  }
```

```
   #include<stdio.h>
struct _person {
int nRecordType;
struct _person *NextPerson;
struct _person *PrevPerson;
char name[31];
char code[61];
char pass[26];
char mess[6];
char phone[31];
int nRecordNumber;
}PersonName;

struct _person *FirstPerson = NULL;
struct _person *LastPerson = NULL;
struct _person *Person = NULL;
struct _person *TempPerson = NULL;
char FileName[80]="ch";
char buffer[80];
int num[6];
char str[6];
int nRecord = 0;
```

```
#include "stdio.h"
#include "dos.h"
#include "bios.h"
#include "stdlib.h"
#include "string.h"

#define BORDER 1
#define ESC 27
#define MAX_FRAME 16
#define REV_VID 0x70
#define NORM_VID 7

void save_vide(int num), restore_vide(int num);
extern goto_xy(int x, int y), cls(void);
void write_strin(int x, int y, char *p, int attrib);
void write_cha(int x, int y, char ch, int attrib);
void draw_borde(int num);
void display_header(int num);
void window_gets(int num, char *s), size(int num);
void move(int num), window_cls(int num);
void window_cleol(int num), window(int num);
void deactivate(int num);
void window_bksp(int num);
int window_upline(int num), window_downline(int num);
int make_window(int num, char *header, int startx, int starty,
                int endx, int endy, int border);
int window_getche(int num);
int window_putchar(int num, char ch);
int window_xy(int num, int x, int y);
extern video_mode(void);
int get_special(void);
int push(int i), pop(void);
extern readkey(void);

char far *vid_me;

struct window_frame {
  int startx, endx, starty, endy; /* window position */
  int curx, cury; /* current cursor position in window */
  unsigned char *p; /* pointer to buffer */
  char *header; /* header message */
  int border; /* border on/off */
  int active; /* on screen yes/no */
} fram[MAX_FRAME];

/*************************************************************/
/* Window functions                                          */
/*************************************************************/

/* Display a pull-down window. */
void window(int num) /* window number */
{
  int  vmode;
  int x, y;

  vmode = video_mode();
  if((vmode!=2) && (vmode!=3) && (vmode!=7)) {
```

```
    printf("video must be in 80 column text mode");
    exit(1);
  }
  /* set proper address of video RAM */
  if(vmode==7) vid_me = (char far *) 0xB0000000;
  else vid_me = (char far *) 0xB8000000;

  /* get active window */
  if(!fram[num].active) { /* not currently in use */
    save_vide(num);       /* save the current screen */
    fram[num].active = 1; /* set active flag */
  }

  if(fram[num].border) draw_borde(num);
  display_header(num); /* display the window */

  x = fram[num].startx + fram[num].curx + 1;
  y = fram[num].starty + fram[num].cury + 1;
  goto_xy(x, y);
}

/* Construct a pull-down window frame.
   1 is returned if window frame can be constructed;
   otherwise 0 is returned.
*/
make_window(
  int num, /* window number */
  char *header, /* header text */
  int startx, int starty, /* X,Y coordinates of upper left corner */
  int endx, int endy, /* X,Y coordinates of lower right corner */
  int border) /* no border if 0 */
{
  unsigned char *p;

  if(num>MAX_FRAME) {
    printf("Too many windows\n");
    return 0;
  }

  if((starty>24) || (starty<0) || (startx>78) || (startx<0)) {
    printf("range error");
    return 0;
  }

  if((endy>24) || (endx>79)) {
    printf("window won't fit");
    return 0;
  }

  /* allocate enough memory to hold it */
  p = (unsigned char *) malloc(2*(endy-starty+1)*(endx-startx+1));
  if(!p) exit(1); /* put your own error handler here */
```

```
  /* construct the frame */
  fram[num].startx = startx; fram[num].endx = endx;
  fram[num].starty = starty; fram[num].endy = endy;
  fram[num].p = p;
  fram[num].header = header;
  fram[num].border = border;
  fram[num].active = 0;
  fram[num].curx = 0; fram[num].cury = 0;
  return 1;
}

/* Deactivate a window and remove it from the screen. */
void deactivate(int num)

{
  /* reset the cursor postion to upper left corner */
  fram[num].curx = 0;
  fram[num].cury = 0;
  fram[num].active = 0;
  restore_vide(num);
}

/* Interactively change the size of a window.
*/
void size(int num)
{
  char ch;
  int x, y, startx, starty;

  /* activate if necessary */
  if(!fram[num].active) window(num);

  startx = x = fram[num].startx;
  starty = y = fram[num].starty;
  window_xy(num, 0, 0);
```

```
do {
  ch = get_special();
  switch(ch) {
    case 75: /* left */
      startx--;
      break;
    case 77: /* right */
      startx++;
      break;
    case 72: /* up */
      starty--;
      break;
    case 80: /* down */
      starty++;
      break;
    case 71: /* up left */
      starty--; startx--;
      break;
    case 73: /* up right */
      starty--; startx++;
      break;
    case 79: /* down left*/
      starty++; startx--;
      break;
    case 81: /* down right */
      starty++; startx++;
      break;
    case 60: /* F2: cancel and use original size */
      startx = x;
      starty = y;
      ch = 59;
  }
  /* see if out-of-range */
  if(startx<0) startx++;
  if(startx>=fram[num].endx) startx--;
  if(starty<0) starty++;
  if(starty>=fram[num].endy) starty--;
  deactivate(num);
  fram[num].startx = startx;
  fram[num].starty = starty;
  window(num);
} while(ch!=59);  /* F1 to stop */
deactivate(num);
}

/* Interactively move a window. */
void move(int num)
{
  char ch;
  int x, y, ex, ey, startx, starty, endx, endy;

  /* activate if necessary */
  if(!fram[num].active) window(num);

  startx = x = fram[num].startx;
  starty = y = fram[num].starty;
  endx = ex = fram[num].endx;
  endy = ey = fram[num].endy;
  window_xy(num, 0, 0);
```

```
do {
  ch = get_special();
  switch(ch) {
    case 75: /* left */
      startx--;
      endx--;
      break;
    case 77: /* right */
      startx++;
      endx++;
      break;
    case 72: /* up */
      starty--;
      endy--;
      break;
    case 80: /* down */
      starty++;
      endy++;
      break;
    case 71: /* up left */
      starty--; startx--;
      endy--; endx--;
      break;
    case 73: /* up right */
      starty--; startx++;
      endy--; endx++;
      break;
    case 79: /* down left*/
      starty++; startx--;
      endy++; endx--;
      break;
    case 81: /* down right */
      starty++; startx++;
      endy++; endx++;
      break;
    case 60: /* F2: cancel and use original size */
      startx = x;
      starty = y;
      endx = ex;
      endy = ey;
      ch = 59;
  }

  /* see if out-of-range */
  if(startx<0) {
    startx++;
    endx++;
  }
  if(endx>=79) {
    startx--;
```

```
      endx--;
    }
    if(starty<0) {
      starty++;
      endy++;
    }
    if(endy>=25) {
      starty--;
      endy--;
    }
    deactivate(num);
    fram[num].startx = startx;
    fram[num].starty = starty;
    fram[num].endx = endx;
    fram[num].endy = endy;
    window(num);
  } while(ch!=59); /* F1 to stop */
  deactivate(num);
}

/* Display the header message in its proper location. */
void display_header(int num)
{
  register int x, len;

  x = fram[num].startx;

  /* Calculate the correct starting position to center
     the header message - if negative, message won't
     fit.
  */
  len = strlen(fram[num].header);
  len = (fram[num].endx - x - len) / 2;
  if(len<0) return; /* don't display it */
  x = x + len;

  write_strin(x, fram[num].starty,
              fram[num].header, NORM_VID);
}

/* Draw a window's border. */
void draw_borde(int num)
{
  register int i;
  char far *v, far *t;
```

```
  v = vid_me;
  t = v;
  for(i=fram[num].starty+1; i<fram[num].endy; i++) {
     v += (i*160) + fram[num].startx*2;
     *v++ = 179; /* vertical bar */
     *v = NORM_VID;
     v = t;
     v += (i*160) + fram[num].endx*2;
     *v++ = 179;
     *v = NORM_VID;
     v = t;
  }
  for(i=fram[num].startx+1; i<fram[num].endx; i++) {
     v += (fram[num].starty*160) + i*2;
     *v++ = 196; /* horizontal bar */
     *v = NORM_VID;
     v = t;
     v += (fram[num].endy*160) + i*2;
     *v++ = 196;
     *v = NORM_VID;



     v = t;
  }

  /* draw the corners of the border */
  write_cha(fram[num].startx, fram[num].starty,
            (char) 218, NORM_VID);
  write_cha(fram[num].startx, fram[num].endy,
            (char) 192, NORM_VID);
  write_cha(fram[num].endx, fram[num].starty,
            (char) 191, NORM_VID);
  write_cha(fram[num].endx, fram[num].endy,
            (char) 217, NORM_VID);
}

/*****************************************************************/
/* Window I/O functions                                          */
/*****************************************************************/

/* Write a string at the current cursor position
   in the specified window.
   Returns 0 if window not active;
   1 otherwise.
*/
window_puts(int num, char *str)
{
   /* make sure window is active */
  if(!fram[num].active) return 0;

  for( ; *str;  str++)
    window_putchar(num, *str);
  return 1;
}
```

```
/* Write a character at the current cursor position
   in the specified window.
   Returns 0 if window not active;
   1 otherwise.
*/
window_putchar(int num, char ch)
{
  register int x, y;
  char far *v;

  /* make sure window is active */
  if(!fram[num].active) return 0;

  x = fram[num].curx + fram[num].startx + 1;
  y = fram[num].cury + fram[num].starty + 1;

  v = vid_me;
  v += (y*160) + x*2; /* compute the address */
  if(y>=fram[num].endy) {
    return 1;
  }
  if(x>=fram[num].endx) {
    return 1;
  }

  if(ch=='\n') { /* newline char */
    y++;
    x = fram[num].startx+1;
    v = vid_me;
    v += (y*160) + x*2; /* compute the address */
    fram[num].curx = 0;  /* reset X */
    fram[num].cury++; /* increment Y */
  }
  else {
    fram[num].curx++;
    *v++ = ch;  /* write the character */
    *v++ = NORM_VID;    /* normal video attribute */
  }
  window_xy(num, fram[num].curx, fram[num].cury);
  return 1;
}

/* Position cursor in a window at specified location.
   Returns 0 if out of range;
   non-zero otherwise.
*/
window_xy(int num, int x, int y)
{
  if(x<0 || x+fram[num].startx>=fram[num].endx-1)
    return 0;
  if(y<0 || y+fram[num].starty>=fram[num].endy-1)
    return 0;
  fram[num].curx = x;
  fram[num].cury = y;
  goto_xy(fram[num].startx+x+1, fram[num].starty+y+1);
  return 1;
}
```

```
/* Read a string from a window. */
void window_gets(int num, char *s)
{
  char ch, *temp;



  temp = s;
  for(;;) {
    ch = window_getche(num);
    switch(ch) {
      case '\r':  /* the ENTER key is pressed */
        *s='\0';
        return;
      case '\b': /* backspace */
        if(s>temp) {
          s--;
          fram[num].curx--;
          if(fram[num].curx<0) fram[num].curx = 0;
            window_xy(num, fram[num].curx, fram[num].cury);
              write_cha(fram[num].startx+fram[num].curx+1,
          fram[num].starty+fram[num].cury+1, ' ', NORM_VID);
        }
        break;
      default: *s = ch;
        s++;
    }
  }
}


/* Input keystrokes inside a window.
   Returns full 16 bit scan code.
*/
window_getche(int num)
{
  union inkey {
    char ch[2];
    int i;
  } c;

  if(!fram[num].active) return 0; /* window not active */

  window_xy(num, fram[num].curx, fram[num].cury);

  c.i = readkey();       /* read the key */
```

```
    if(c.ch[0]) {
      switch(c.ch[0]) {
        case '\r': /* the ENTER key is pressed */
          break;
        case '\b': /* backspace */
          break;
        default:
          if(fram[num].cury+fram[num].starty < fram[num].endy-1) {
            write_cha(fram[num].startx+ fram[num].curx+1,
              fram[num].starty+fram[num].cury+1, c.ch[0], NORM_VID);
            fram[num].curx++;
          }
      }
      if(fram[num].curx < 0) fram[num].curx = 0;
      if(fram[num].curx+fram[num].startx > fram[num].endx-2)
        fram[num].curx--;
      window_xy(num, fram[num].curx, fram[num].cury);
    }
    return c.i;
  }

  /* Clear a window. */
  void window_cls(int num)
  {
    register int i,j;
    char far *v, far *t;

    v = vid_me;
    t = v;
    for(i=fram[num].starty+1; i<fram[num].endy; i++)
      for(j=fram[num].startx+1; j<fram[num].endx; j++) {
        v = t;
        v += (i*160) + j*2;
        *v++ = ' ';  /* write a space */
        *v = NORM_VID;    /* normal */
    }
    fram[num].curx = 0;
    fram[num].cury = 0;
  }

  /* Clear to end of line. */
  void window_cleol(int num)
  {
    register int i, x, y;

    x = fram[num].curx;
    y = fram[num].cury;
    window_xy(num, fram[num].curx, fram[num].cury);

    for(i=fram[num].curx; i<fram[num].endx-1; i++)
      window_putchar(num,' ');
    window_xy(num, x, y);
  }
```

```
/* Move cursor up one line.
   Returns non-zero if successful;
   0 otherwise.
*/



window_upline(int num)
{
  if(fram[num].cury > 0) {
    fram[num].cury--;
    window_xy(num, fram[num].curx, fram[num].cury);
    return 1;
  }
  return 0;
}

/* Move cursor down one line.
   Returns non-zero if successful;
   0 otherwise.
*/
window_downline(int num)
{
  if(fram[num].cury < fram[num].endy-fram[num].starty-1) {
    fram[num].cury++;
    window_xy(num, fram[num].curx, fram[num].cury);
    return 1;
  }
  return 1;
}

/* Back up one character. */
void window_bksp(int num)
{
  if(fram[num].curx>0) {
    fram[num].curx--;
    window_xy(num, fram[num].curx, fram[num].cury);
    window_putchar(num, ' ');
    fram[num].curx--;
    window_xy(num, fram[num].curx, fram[num].cury);
 }
}
```

```
/*****************************************************/
/* Misc. functions                                   */
/*****************************************************/

/* Display a string with specifed attribute. */
void write_strin(int x, int y, char *p, int attrib)
{
  char far *v;

  v = vid_me;
  v += (y*160) + x*2; /* compute the address */
  while(*p) {
    *v++ = *p++;  /* write the character */
    *v++ = attrib;    /* write the attribute */
   }
}

/* Write character with specified attribute. */
void write_cha(int x, int y, char ch, int attrib)
{
  char far *v;

  v = vid_me;
  v += (y*160) + x*2;
  *v++ = ch;  /* write the character */
  *v = attrib;     /* write the attribute */
}

/* Save a portion of the screen. */
void save_vide(int num)



{
  register int i,j;
  char *buf_ptr;
  char far *v, far *t;

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

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

  buf_ptr = fram[num].p;
  v = vid_me;
  t = v;
  for(i=fram[num].startx; i<fram[num].endx+1; i++)
    for(j=fram[num].starty; j<fram[num].endy+1; j++) {
      v = t;
      v += (j*160) + i*2;
      *v++ = *buf_ptr++;  /* write the character */
      *v = *buf_ptr++;    /* write the attribute */
  }
  fram[num].active = 0; /* restore_video */
}

/* Return the position code of arrow and function keys. */
get_special(void)
{
  union inkey {
    char ch[2];
    int i;
  } c;

  c.i = readkey(); /* read the key */

  return c.ch[1];
}
```

```
/* Pull-down menu routines for text mode operation and
   short sample program.
*/

#include "stdio.h"
#include "dos.h"
#include "stdlib.h"
#include "bios.h"
#include "ctype.h"
#include "string.h"

#define BORDER 1
#define ESC 27
#define MAX_FRAME 15
#define REV_VID 0x70
#define NORM_VID 7

void save_video(int num);
void restore_video(int num);
void goto_xy(int x, int y), cls(void);
void write_video(int x, int y, char *p, int attrib);
void display_menu(int num);
void write_string(int x, int y, char *p, int attrib);
void write_char(int x, int y, char ch, int attrib);
void pd_driver(void);
int make_menu(int num, char *menu[], char *keys,
              int count, int x, int y, int border);
int get_resp(int num), pulldown(int num);
int video_mode(void);
void display_menu(int num), draw_border(int num);
char far *vid_mem;
int readkey(void);

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

char *func[]= {
  "Operate",
  "Mailbox",
  "Database",
  "Call out",
  "Exit"
};
```

```
char *mail[]= {
  "Creat",
  "Display",
  "Search",
  "Erase",
  "Write",
  "Read",
  "Voice"
};



char *datb[]= {
  "Creat",
  "Display",
  "Search",
  "Erase",
  "Write",
  "Read",
  "Record",
  "Voice"
};


main(void)
{
  cls();
  background();


  /* first, create the menu frames */
  make_menu(0,func, "omdce", 5, 20, 5, BORDER);
  make_menu(1,mail, "cdsewrv" , 7, 28, 7, BORDER);
  make_menu(2,datb, "cdsewrrv", 8, 28, 8, BORDER);
  pd_driver(); /* activate the menu system */

  return 0;
}

/* Demonstrate the pull-down menu functions. */

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

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

```
  /* get active window */
  if(!frame[num].active) { /* not currently in use */
    save_video(num);       /* save the current screen */
    frame[num].active = 1; /* set active flag */
  }

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

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


/* Construct a pull down menu fra...
   It returns 1 if the menu frame can be constructed;
   otherwise 0 is returned.
*/
make_menu(
  int num,       /* menu number */
  char *menu[], /* menu text */
  char *keys,    /* hot keys */
  int count,     /* number of menu items */




  int x, int y, /* X,Y coordinates of left hand corner */
  int border;    /* no border if 0 */
)
{
  register int i, len;
  int endx, endy;
  unsigned char *p;

  if(num>MAX_FRAME) {
    printf("Too many menus\n");
    return 0;
  }

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

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

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

  /* construct the frame */
  frame[num].startx = x; frame[num].endx = endx;
  frame[num].starty = y; frame[num].endy = endy;
  frame[num].p = p;
  frame[num].menu = (char **) menu;
  frame[num].border = border;
  frame[num].keys = keys;
  frame[num].count = count;
  frame[num].active = 0;
  return 1;
}

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

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

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

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

  v = vid_mem;


  t = v;

  /* draw vertical lines */
  for(i=frame[num].starty+1; i<frame[num].endy; i++) {
     v += (i*160) + frame[num].startx*2;
     *v++ = 179;
     *v = NORM_VID;
     v = t;
     v += (i*160) + frame[num].endx*2;
     *v++ = 179;
     *v = NORM_VID;
     v = t;
  }
```

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

    /* draw corners */
    write_char(frame[num].startx, frame[num].starty,
               (char) 218, NORM_VID);
    write_char(frame[num].startx, frame[num].endy,
               (char) 192, NORM_VID);
    write_char(frame[num].endx, frame[num].starty,
               (char) 191, NORM_VID);
    write_char(frame[num].endx, frame[num].endy,
               (char) 217, NORM_VID);
}

/* Input user's selection */
get_resp(int num)
{
    union inkey {
      char ch[2];
      int i;
    } c;
    int arrow_choice=0;
    char *key_choice;
    int x, y;

    x = frame[num].startx+1;
    y = frame[num].starty+1;

    /* highlight the first selection */
    goto_xy(x, y);
    write_string(x, y, frame[num].menu[0], REV_VID);

    for(;;) {
      c.i = readkey();     /* read the key */

      /* reset the selection to normal video */
      goto_xy(x, y+arrow_choice);
      write_string(x, y+arrow_choice,
        frame[num].menu[arrow_choice], NORM_VID); /* redisplay */

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

```
      if(key_choice) return key_choice-frame[num].keys;

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

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

/* Display a string with specified attribute. */
void write_string(int x, int y, char *p, int attrib)
{
  char far *v;

  v = vid_mem;
  v += (y*160) + x*2; /* compute the address */
  while(*p) {
    *v++ = *p++;  /* write the character */
    *v++ = attrib;    /* write the attribute */
  }
}

/* Write a character with specified attribute. */
void write_char(int x, int y, char ch, int attrib)
{
  char far *v;

  v = vid_mem;
  v += (y*160) + x*2; /* compute the address */
  *v++ = ch;  /* write the character */
  *v = attrib;    /* write the attribute */
}
```

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

  buf_ptr = frame[num].p;
  v = vid_mem;
  for(i=frame[num].startx; i<frame[num].endx+1; i++)
    for(j=frame[num].starty; j<frame[num].endy+1; j++) {
      t = (v + (j*160) + i*2);
      *buf_ptr++ = *t++;
      *buf_ptr++ = *t;



     *(t-1) = ' ';  /* clear the window */
    }
}

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

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

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

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

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

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

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

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

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



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

```
#include<string.h>
#include<ctype.h>
/*#include<conio.h> */
#include<stdio.h>
#include<process.h>
#include<stdlib.h>
#include<troub.h>


#define TRUE 1
#define FALSE (!TRUE)

#define Increment_Amount 1
#define Person_Record 1
char UserFile[10];
void Menu(void);
void CreatData(void);
void DisplayData(void);
void EraseData(void);
void SearchData(void);
void WriteFile(void);
void ReadFile(void);
      void CreatData(void)
      {
       char numstr[80];
       struct _person *TempPerson;

Person = (struct _person *)calloc(sizeof(PersonName);
window(3);
window_puts(3,"Creat code:");
++nRecord;
window_gets(3,Person->code);
if(strlen(Person->code) > 0)
{
if(FirstPerson==NULL)
{
window_puts(3,"\n");

window_puts(3,"It is first record \n");
Person->NextPerson = NULL;
Person->PrevPerson = NULL;
FirstPerson = Person;
LastPerson = Person;
TempPerson = NULL;
}
else
{
TempPerson = FirstPerson;
}
while(TempPerson)
{
```

```
if(strcmp(Person->code,TempPerson->code)<0||
TempPerson ==LastPerson)
{
if(strcmp(Person->code,TempPerson->code)<0&&
TempPerson==FirstPerson)
{
Person->NextPerson = FirstPerson;
FirstPerson = Person;


Person->PrevPerson=NULL;
TempPerson = Person->NextPerson;
TempPerson->PrevPerson = Person;
}
else
{
if(strcmp(Person->code,TempPerson->code)>0&&
TempPerson==LastPerson)
{

Person->PrevPerson = LastPerson;
LastPerson = Person;
Person->NextPerson= NULL;
TempPerson= Person->PrevPerson;
TempPerson->NextPerson = Person;
}
else
{
Person->PrevPerson =
TempPerson->PrevPerson;
Person->NextPerson = TempPerson;
TempPerson->PrevPerson= Person;
TempPerson = Person->PrevPerson;
TempPerson->NextPerson = Person;
}
}
TempPerson = NULL;
}
else
{
TempPerson = TempPerson->NextPerson;
}
}
Person->nRecordNumber = nRecord;
if(TRUE)
{
window_puts(3,"\n");
window_puts(3,"Creat name:");
window_gets(3,Person->n...);
```

```
window_puts(3,"\n");
window_puts(3,"Creat password:");
window_gets(3,Person->pass);
window_puts(3,"\n");
window_puts(3,"Creat file:");
window_gets(3,Person->mess);

}
}
else
{
window_puts(3,"Sorry,name must not be blank.");
}
getch();
deactivate(3);
}
void DisplayData(void)
{
struct _person *TempPerson;
if(TempPerson ==NULL)
{printf("\nEmpty list.\n");return;}
TempPerson = FirstPerson;
window(3);
do
{
window_puts(3,"Code:");


window_puts(3,TempPerson->code);
window_puts(3,"\n");
window_puts(3,"Name:");
window_puts(3,TempPerson->name);
window_puts(3,"\n");
window_puts(3,"Password:");
window_puts(3,TempPerson->pass);
window_puts(3,"\n");
window_puts(3,"Messege file:");
window_puts(3,TempPerson->mess);
TempPerson = TempPerson->NextPerson;
}
while(TempPerson!=NULL);
getch();
deactivate(3);
}
```

```
void EraseData(void)
{
 struct _person *TempPerson;
 struct _person *LastPerson;
 char DelName[81];
 window(4);
 if(TempPerson ==NULL)
 { window_puts(4,"Empty list.\n");return; }
 window_puts(4,"Enter code:");
 window_gets(4,DelName);
 TempPerson = FirstPerson;
 do
 {
 if(strcmp(TempPerson->code,strlwr(DelName))==0
 ||strcmp(TempPerson->code,strupr(DelName))==0)
 {
 if(TempPerson==FirstPerson)
 FirstPerson = TempPerson->NextPerson;
 else
  LastPerson->NextPerson = TempPerson->NextPerson;

  free(TempPerson);
  return;
  }
  LastPerson = TempPerson;
  TempPerson = TempPerson->NextPerson;

  }
  while(TempPerson !=NULL);
  window_puts(4,"No such name on list\n");
  getch();
  deactivate(4);
  }

  void SearchData(void)
  {
  struct _person *TempPerson;
  struct _person *LastPerson;
  char DelName[81];
  window(4);
  if(TempPerson ==NULL)
  { window_puts(4,"Empty list.\n");return;}
  window_puts(4,"Enter code:");
  window_gets(4,DelName);
  getch();
  deactivate(4);
  window(3);
  TempPerson = FirstPerson;
  do
```

```
{
if(strcmp(TempPerson->code,strlwr(DelName))==0
||strcmp(TempPerson->code,strupr(DelName))==0)
{
window_puts(3,"Code:");
window_puts(3,TempPerson->code);
window_puts(3,"\n");
window_puts(3,"Name:");
window_puts(3,TempPerson->name);
window_puts(3,"\n");
window_puts(3,"Password:");
window_puts(3,TempPerson->pass);
window_puts(3,"\n");
window_puts(3,"Filename:");
window_puts(3,TempPerson->mess);
}
LastPerson = TempPerson;
TempPerson = TempPerson->NextPerson;
}
while(TempPerson!=NULL);
getch();
deactivate(3);
}

void WriteFile(void)
{
struct _person *TempPerson;
FILE *ptr_file;
window(4);
if(TempPerson==NULL)
{window_puts(4,"Empty list.");return;}
if((ptr_file=fopen(FileName,"wb"))==NULL)
{
window_puts(4,"\nCan't open file");
window_puts(4,FileName);
return;
}
TempPerson = FirstPerson;
do
{
fwrite(TempPerson,sizeof(struct _person),1,ptr_file);
TempPerson = TempPerson->NextPerson;
}
while(TempPerson != NULL);
fclose(ptr_file);
window_puts(4,"\nFile written.\n");
getch();
deactivate(4);
}
```

```
void ReadFile(void)
{
struct _person *TempPerson;
FILE *ptr_file;
window(4);
if((ptr_file=fopen(FileName,"rb"))==NULL)
{
window_puts(4,"Can't open file ");
window_puts(4,FileName):
}
FirstPerson = NULL;
while(TRUE)
{
TempPerson = malloc(sizeof(struct _person));
if(TempPerson == NULL)
{window_puts(4,"Allocation erro");return;}


if(fread(TempPerson,sizeof(struct _person),1,ptr_file)!=1)
{
free(TempPerson);
fclose(ptr_file);
window_puts(4,"File read\n");
delay(2000);
deactivate(4);
return;
}
 TempPerson->NextPerson = FirstPerson;
 FirstPerson = TempPerson;
 }
 }


 void AutoSearch(void)
 {
 struct _person *TempPerson;
 struct _person *LastPerson;
 char DelName[81];
 window(4);
 if(TempPerson ==NULL)
 { window_puts(4,"Empty list.\n");return;}
 window_puts(4,"Find ");
 strcpy(DelName,str);
 window_puts(4,DelName);
 delay(1000);
 deactivate(4);
 window(3);
 TempPerson = FirstPerson;
 do
```

```
if(strcmp(TempPerson->code,strlwr(DelName))==0
||strcmp(TempPerson->code,strupr(DelName))==0)
{
window_puts(3,"Code:");
window_puts(3,TempPerson->code);
window_puts(3,"\n");
window_puts(3,"Name:");
window_puts(3,TempPerson->name);
window_puts(3,"\n");
window_puts(3,"Password:");
window_puts(3,TempPerson->pass);
window_puts(3,"\n");
window_puts(3,"Filename:");
window_puts(3,TempPerson->mess);
strcpy(UserFile,TempPerson->mess);
}
LastPerson = TempPerson;
TempPerson = TempPerson->NextPerson;
}
while(TempPerson!=NULL);
delay(2000);
deactivate(3);
}

void dtmf()
{ int a,b,c,d,k;
  int i=0;


outportb(0x307,0x090);
window(4);
window_puts(4,"Wait for signal.");
window_puts(4,"\n");
do{


a=inportb(0x304);
b=a&31;
if(b>=16)
{ c=b-16;

 num[i]=c;
 window_puts(4,"*");

 i++;
 delay(200); }
   }while(i<7);
 window_puts(4,"\n");
 window_puts(4,"Ok!");
delay(2000);
 deactivate(4);
 }
void signal(void)
 {
    int i;
    char ch;
    int a;
```

```
   dtmf();
 for(i=0;i<7;i++)
{ ch=*ltoa(num[i],&ch,10);
  str[i]=ch;

  }
window(4);
window_puts(4,"Number is:");
window_puts(4,str);
delay(2000);
deactivate(4);
  }
 void record()
  { unsigned i;
    int j;
   FILE *file_pointer;
   char ch;
   if((file_pointer=fopen(UserFile,"w"))!=NULL) {

   for(i=0;i<60000;i++)
   { ch=inportb(0x300);
    fputc(ch,file_pointer);
    if(i%77==0)
   { fputc('\n',file_pointer);}
    for(j=0;j<70;j++)
    ;
                }
    }

    fclose(file_pointer);
    }
   void play()
   {
   FILE *file_pointer;
   char ch;
   int i;
   clrscr();
   if((file_pointer=fopen(UserFile,"r"))!=NULL)
   {
   ch=fgetc(file_pointer);
   while(ch!=EOF)
   {
    outportb(0x301,ch);
    ch=fgetc(file_pointer);



    for(i=0;i<70;i++)
    ;
    }
    }
    fclose(file_pointer);
 }
```

สรุปผลการทดลอง

อุปกรณ์ Voice Mail Box จะสามารถให้บริการทางด้านการรับฝากข้อความ หรือให้บริการด้านข้อมูลเสียง นอกจากนั้นยังสามารถนำมาใช้บริการอื่น ๆ อีกเช่น การเตือนเวลา การแจ้งเวลา ซึ่งสามารถเพิ่มการบริการได้ด้วยการปรับปรุง software โดยไม่ต้องแก้ไข Hardware แต่อย่างใด

ในโครงการนี้ผู้ทดลองได้สร้างโปรแกรมฐานข้อมูลขนาดเล็ก และเก็บข้อความเสียงลงในไฟล์และเก็บไฟล์ในลักษระ Double link list ซึ่งทำให้สามารถสร้างไฟล์เก็บไว้ได้เป็นจำนวนมาก และไม่สิ้นเปลือง Marddisk

แนวทางการพัฒนา

ในการทอลองได้ประสบปัญหา จากสัญญาณรบกวนที่มาจากเครื่อง Computer ซึ่งเป็นปัญหาที่หลีกเลี่ยงไม่ได้ เพื่อให้เสียงคมและชัดขึ้น ควรจะจัดหา filter ที่มีคุณภาพสูงมารองสัญญาณรบกวนออกไป Amplifier ที่ดีก็มีส่วนในการเพิ่มคุณภาพเสียงด้วย

นอกจากนี้ควรมีการพัฒนา Voice Mail Box ให้ใช้ได้ในระหว่างเครื่อง PC ด้วยกัน หรือนำมาใช้ในระบบ LAN โดยใช้ในลักษระเดียวกันกับ ElECTRONIN MAIL

กิตติกรรมประกาศ

โครงงานนี้คงจะไม่สามารถสำเร็จเรียบร้อยลงด้วยดี ถ้าปราศจากบุคคลเหล่านี้ คือ อ. เกรียงไกร วงศ์โรจนาภรณ์ อาจารย์ที่ปรึกษา ซึ่งคอยให้คำแนะนำและคอยช่วยเหลือเป็นอย่างดียิ่ง รวมทั้งเพื่อน ๆ ทุกคน ที่คอยให้กำลังใจและช่วยเหลือเสมอมา อีกทั้งน้องๆ ทุกคนที่มีส่วนร่วมในการช่วยงานด้านการทำปริญญานิพนธ์

ขอสิ่งศักดิ์สิทธิ์ทั้งหลายในสากลโลกโปรดช่วยอำนวยพรแก่บุคคลเหล่านี้ ให้ประสบแต่ความสุขความเจริญตลอดไปด้วยเทอญ

หนังสืออ้างอิง

จิติ หนูแก้ว,"เทคนิคการเชื่อมต่อ IBM PC กับอุปกรณ์ภายนอกเพื่อประยุกต์ใช้งานต่างๆ ", บริษัทซีเอ็ดยูเคชัน จำกัด, กรุงเทพฯ, 2535.

JOHN UNFENBECK,"MICROCOMPUTER AND MICROPROCESSORS", SECOND EDITION, Prentice-Hall International,Inc. Singaprore.

National Semiconductor Coporation. "Data Conversion/ Acquisition Databook", Sonta Clara, California; USA,1984.

"คู่มือไอซีไมโครโปรเซสเซอร์ และไอซีที่เกี่ยวข้อง", บริษัทซีเอ็ดยูเคชัน จำกัด กรุงเทพ,2536.