ระบบรับฝากข้อความโทรศัพท์อัตโนมัติ

ELECTRONIC VOICE SYSTEM

โดย

นส.เกศรินทร์ ชนะกุล

นาย พรชัย จักรชำรงค์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิศวกรรมคอมพิวเตอร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2535

สารบัญ

บทคัดย่อ

การใช้งานระบบเครือข่ายโทรศัพท์ เพื่อการติดต่อสื่อสาร ได้พัฒนามายาวนาน จนกระทั่งในปัจจุบัน เป็นเครือข่ายการติดต่อสื่อสารที่ใหญ่ที่สุด สามารถติดต่อกันได้เกือบจะทั่วโลก ระบบคอมพิวเตอร์ ก็ได้อาศัยระบบเครือข่ายของโทรศัพท์ ในการส่งข้อมูล ติดต่อสื่อสารระหว่างกัน จะเห็นได้ว่า ระบบคอมพิวเตอร์ และระบบการติดต่อสื่อสาร (Computer and Communication) ได้พัฒนาเพื่อเพิ่มประสิทธิภาพให้แก่กันอย่างมากมาย

โครงงานชิ้นนี้ เป็นโครงสร้างต่อเนื่อง เพื่อพัฒนาระบบ Electronic Voice System ซึ่งเป็นการผสมผสาน ระบบคอมพิวเตอร์ และระบบเครือข่ายโทรศัพท์ ให้สามารถติดต่อ และใช้งานผ่านกันได้อย่างมีประสิทธิภาพสูงสุด ผู้ใช้สามารถใช้งานระบบได้หลายลักษณะ เช่น ใช้เป็นระบบ Voice mail หรือจะใช้เป็นระบบตอบรับอัตโนมัติ และยังสามารถใช้ต่อกับ PABX เพื่อใช้ในการโอนย้ายสายภายในอัตโนมัติได้อีกด้วย

การทำงานของระบบ มีการทำงานเป็นระดับ โดยระดับล่างสุด จะเป็นการใช้งานเกี่ยวกับ ระบบจัดการเสียง (Sound management system) ทำงานอยู่ sound card มีหน้าที่เกี่ยวกับ การอัดเสียง (Record) และการเล่นเสียง (Play) เป็นต้น และมีส่วนที่จัดการเกี่ยวกับระบบโทรศัพท์ (Telephone management system) มีหน้าที่คอยตรวจสอบสถานะของสายโทรศัพท์ที่ติดต่ออยู่ เช่น มีสัญญาณ Ringing เข้ามา หรือ เมื่อต้องการโทรออก แล้วสายปลายทางมีสถานะเป็นอย่างไร เช่น สายว่าง ring back tone หรือสายไม่ว่าง เป็นต้น

ส่วนระบบในระดับที่สูงขึ้นมาคือ ส่วนที่ใช้สร้าง Voice Mail Box ที่ทำหน้าที่ รับฝากข้อความอัตโนมัติ ส่วนที่ 2 คือ ส่วนที่ทำหน้าที่ติดต่อกับ PABX เพื่อให้การโอนสายเป็นงานอัตโนมัติ และส่วนที่ 3 คือ ส่วนที่ใช้สำหรับทำงาน Interactive mode กับผู้ใช้บริการ คือ สามารถอนุโลมการใช้งานผ่านแป้นตัวเลขของโทรศัพท์ได้

ABSTRACT

The telephone network for communication has evolved for many years until it is now the largest network in the world. While the telephone network has been evolving, computer system has also developed simultameously and utilizes this property of telephone network.

This project make use of SOUND CARD to arrange recording and playing voice and CONTROL CARD to regulate the operation of telephone line. Both parts are mixed to form ELECTRONIC VOICE SYSTEM, which can served user for many purposes. The system can perform as a VOCIE MAIL BOX, an INTERACTIVE VOICE INFORMATION, or an AUTOMATIC BRANCHING when connected to a PABX.

This system is separated into many layers. The most bottom is *Sound Management Part*, which arranged for playing and recording sound. Another one is *Telephone Management Part*, which checks for the status of telephone line.

The upper layers are the part for implement Voice Mail, Interactive Information, Calling Back and Automatic Branching, all of these parts make use of both more below parts of the system

บริการรับฝากข้อความทางโทรศัพท์อัตโนมัติ

(Electronic Voice System)

เป็นบริการที่ให้แก่ผู้ใช้โทรศัพท์ สามารถฝากข้อความเมื่อผู้รับไม่อยู่ หรือสายไม่ว่าง นอกจากนั้น ยังบริการให้แก่คนทั่วไป ที่ไม่สมาชิก ในการโอนสายอัตโนมัติ ซึ่งเหมาะสำหรับใช้ในบริษัท หรือองค์กรทั่วไป ที่มีการใช้ PABX ในองค์กร

การทำงานของระบบ ประกอบด้วยส่วนต่าง ๆ ดังนี้

1. ส่วนของ Administration

เป็นส่วนที่กำกับการทำงาน ของระบบทั้งหมด เช่น

- การเพิ่มสมาชิกที่สามารถใช้งานระบบได้
- การลบ User ออก
- การเปลี่ยนแปลงสิทธิของ User เช่น เวลาใน Mail Box
  การกำหนด Service ของ User
- การ Clean UP message ใน mail box ของ User แต่ละคน
- การกำหนด Parameter ต่าง ๆ ของระบบ เช่น
  จำนวนครั้งของกระดิ่งโทรศัพท์ในกรณีไม่มีผู้รับสาย
  directory ที่จะเก็บ message file
- การกำหนด General Information เพื่อให้ User ฟัง

2. ส่วนของ Voice Mail Box

เป็นส่วนที่ทำการ รับฝากข้อความของ User และสามารถส่งข้อมูลเสียงที่เก็บไว้ให้ผู้ฟังฟังได้ User แต่ละคนจะมี Mail Box ตามขนาดเวลาที่กำหนดโดย admin ซึ่งระบบจะคำนวนเวลาที่ใช้ในการเก็บ message และเวลาที่เหลือ หากว่าข้อมูลที่ฝากไว้ ไม่สามารถเก็บไว้ใน mail box ได้ user จะต้องแจ้งให้ admin ทราบ เพื่อให้ admin ลบข้อความใน main box ที่ไม่ต้องการออกเสีย

user แต่ละคนจะมีหมายเลขประจำตัว เพื่อ identify ตนเอง เสียก่อนจึงจะสามารถใช้งานระบบได้ จากนั้น เมื่อต้องการฝากข้อความให้กับ user คนอื่น จะต้องบอกหมายเลขประจำตัวของผู้ที่จะฝากข้อความไว้ให้ และเมื่อได้ฟังข้อความนั้นแล้ว user ระบบจะถาม user ว่า ต้องการลบข้อความนั้นทิ้งหรือไม่ หาก user ต้องการเก็บข้อความนี้ไว้ฟังในภายหลัง user ก็สามารถสั่งให้ระบบเก็บข้อความไว้ก่อนได้

✓ 3. ส่วนของ General Information

เป็นส่วนที่เกี่ยวกับการส่งข้อมูลจากศูนย์บริการให้แก่ผู้ฟัง โดยข้อมูลเหล่านี้อาจได้แก่ข้อมูลทั่วไปที่มีประโยชน์ เช่น ข้อมูลพยากรณ์อากาศ ข้อมูลราคาหุ้น และอื่น ๆ หรืออาจจะเป็นข่าวสารจากศูนย์ ถึงผู้ใช้บริการทุกคน เช่น การเปลี่ยนแปลงการให้บริการ การแจ้งชำระหนี้ เป็นต้น

ข่าวสารหรือข้อมูลนี้ สามารถกำหนดให้เป็นข่าวสารทั่วไป ซึ่ง user ทุกคนเมื่อโทรเข้ามาใช้บริการระบบ จะต้องได้รับฟัง หรือ อาจกำหนดให้เป็นข่าวสารเฉพาะก็ได้ คือ จะกำหนดให้กับ user เป็นรายบุคคลไป ซึ่งกำหนดได้โดย admin

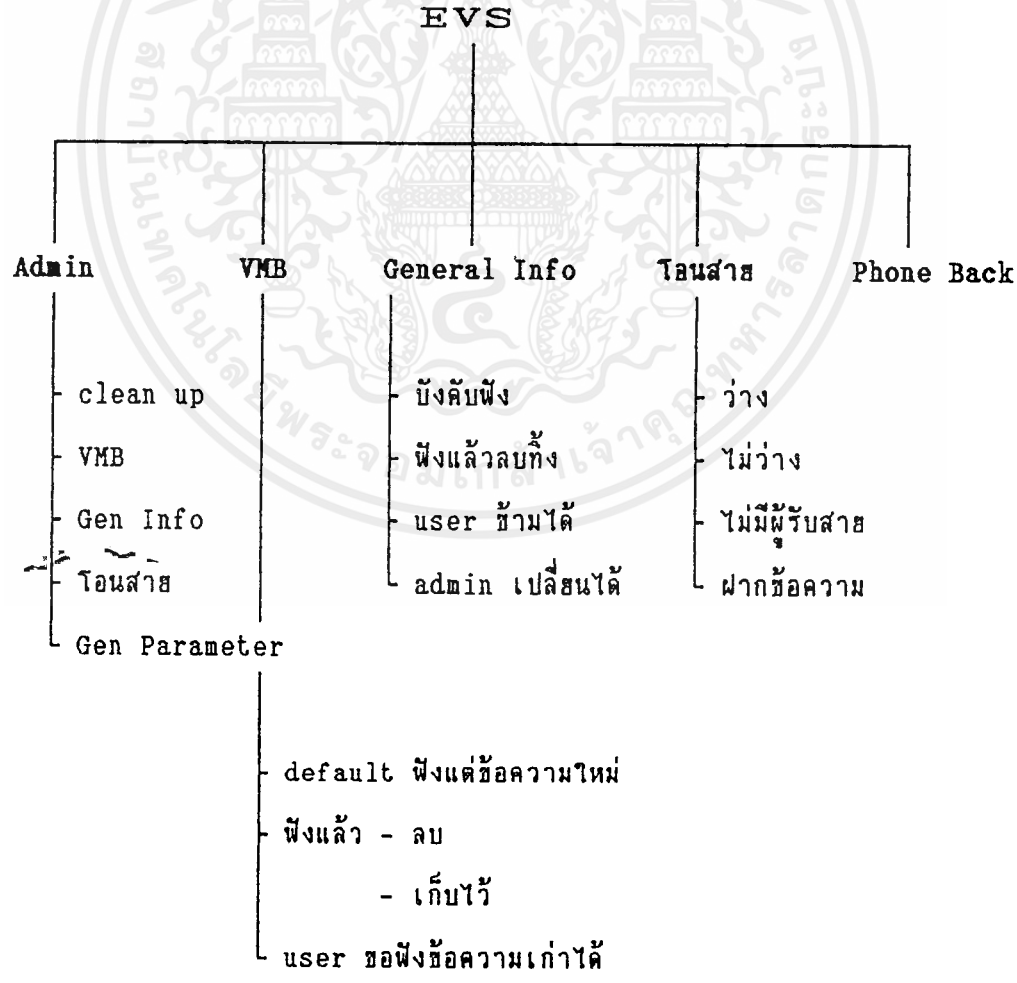
✓ 4. ส่วนของ การโอนสายอัตโนมัติ

เป็นส่วนที่ทำงานเป็นโอเปอเรเตอร์อัตโนมัติ มีการทำงานคือ เมื่อมีผู้โทรเข้ามาและต้องการต่อไปยัง โทรศัพท์ภายในเครื่องอื่น ระบบ EVS จะทำการรับสายโทรศัพท์ พร้อมทั้งตอบรับด้วยข้อความที่เตรียมไว้ หากผู้ที่โทรเข้าไม่ได้ต้องการใช้บริการ voice mail box ก็จะเลือกกดหมายเลขต่อทันที ซึ่งระบบจะทำการต่อสายโทรศัพท์ภายในให้ หากมีผู้รับสาย ระบบก็จะต่อให้สนทนากันได้โดยตรง แต่หากไม่มีผู้รับสาย หรือสายไม่ว่าง ระบบจะถามความประสงค์ของผู้โทรเข้าว่า ต้องการจะยกเลิกการติดต่อ หรือต้องการฝากข้อความไว้ให้กับผู้รับสาย เมื่อผู้รับสายโทรเข้ามายังระบบ ก็จะสามารถรับฟังข้อความที่มีผู้ฝากไว้ให้ได้

5. ส่วนของการโทรกลับ

เป็นบริการที่ให้แก่ผู้ใช้ที่ต้องการให้ระบบ ทำการโทรหาผู้รับสายที่ไม่อยู่ คือ ไม่มีผู้รับสาย ในกรณีที่ไม่มีผู้รับสายเมื่อโทรเข้า ผู้โทรเข้าสามารถสั่งให้ระบบ โทรไปหาผู้รับสายตามหมายเลขโทรศัพท์ที่ผู้รับสายให้ไว้ได้

Diagram ของ โครงงาน EVS



ความรู้เบื้องต้นเกี่ยวกับโทรศัพท์

เครื่องโทรศัพท์ Telephone ประกอบด้วยส่วนต่างๆ ที่สำคัญคือ เครื่องส่ง (Transmitter) , เครื่องรับ (Receiver) ,กระดิ่ง (Ringer) ,Hook Switch และหน้าปัดสำหรับหมุน หรือ กดหมายเลข (Dial) สำหรับเครื่องส่ง และ เครื่องรับรวมเรียกว่า ปากพูดหูฟัง (Handset) ซึ่งเป็นอุปกรณ์ที่ใช้สำหรับเปลี่ยนพลังงานเสียงที่เกิดจากการพูดให้เป็นพลังงานไฟฟ้าที่ได้รับใหกลับเป็นพลังงานเสียงอีกครั้งหนึ่ง โดยเราจะใช้ Transmitter เป็นตัวเปลี่ยนพลังงานเสียงให้เป็นพลังงานไฟฟ้า และใช้ Receiver เป็นตัวเปลี่ยนพลังงาน ไฟฟ้าที่ได้รับให้กลับมาเป็นพลังงานเสียงอีกครั้งหนึ่ง

<u>หลักการทำงานมีดังนี้</u>

เมื่อมีคลื่นเสียงกระทบกับแผ่น Disphragm จะทำให้แผ่น Diaphragm สั่นไปมาพลังงานเสียงก็จะเปลี่ยนเป็นพลังงานกล ในตำหน่งที่แผ่น (Diaphramq) ถูกกดจะทำให้ Electrode แผ่นหน้าเคลื่อนที่เข้าเป็นผลทำให้ผงถ่าน (Carbongranule) ถูกอัดถูกติดกันมากยิ่งขึ้น การอัดตัวของผงถ่านนี้จะทำให้ความต้านทานระหว่าง แผ่น Electrode ทั้งสองมีค่าลดลง ในทางตรงกันข้าม เมื่อแผ่น Diaphragm เคลื่อนที่ออกก็จะเป็นผลทำให้ Electrode แผ่นหน้าเคลื่อนที่ออกด้วย จึงทำให้ความต้านทางของ Transmitter เพิ่มขึ้น

หลักการของ Receiver คือ มีขดลวดพันอยู่ที่ขั้วทั้งสองของแม่เหล็กถาวรที่ต่อกันแบบอนุกรม แต่ขดลวดจะพันกลับทิศทางกัน แม่เหล็กถาวรนี้จะมีอำนาจแม่เหล็กดึงดูดแผ่น Diapham เข้ามาเมื่อมีกระแสไฟสลับ (Speech Current) ไหลผ่านขดลวดก็จะมีผลทำให้เกิดเส้นแรงแม่เหล็กขึ้น ทิศทางของเส้นแรงแม่เหล็กมีทิศทางตรงกันข้ามกับทิศทางกระแสไฟฟ้าที่ไหลในวงจร ซึ่งอาจจะไปเสริมหรือต้านเส้นแรงแม่เหล็กของแม่เหล็กถาวร แผ่น Diaphragm ก็จะเคลื่อนที่เข้าหรือ ออกตามขนาด และความถี่ของ Speech Current ซึ่งจะมีผลทำให้เกิดคลื่นเสียงที่มีขนาด และความถี่เท่ากับ Speech Current ที่ไหลเข้ามาในวงจร คลื่นเสียงที่เกิดขึ้นย่อมมีการสูญเสียไปบ้าง เนื่องจากมีการเปลี่ยนรูปพลังงาน ดังนั้นเอาท์พุตของคลื่นเสียงจะน้อยกว่าอินพุตของพลังงานไฟฟ้าที่ได้รับที่ Receiver

กระดิ่งของเครื่องโทรศัพท์(Ringer) เมื่อมีการเรียกเข้า กระดิ่งที่เครื่องรับโทรศัพท์ของผู้เรียกจะดังขึ้น ซึ่งจะหมายถึง ชุมสายโทรศัพท์ได้ทำการส่งกระแสไฟฟ้าสลับ (Ringing Voltage) มาป้อนที่กระดิ่งของเครื่องโทรศัพท์ โดยทั่วไปแล้วกระแสไฟฟ้าสลับจะมีค่าประมาณ 75-80 โวลท์ ความถี่ 18 - 25 Hz

หน้าปัดของเครื่องโทรศัพท์ มี 2 แบบคือ

- แบบหมุน (Rotary Dial)
- แบบกดปุ่ม (Push Button) ในโครงงานนี้เราจะใช้กับโทรศัพท์ประเภทนี้ได้เท่านั้น โดยหน้าปัดแบบนี้ ใช้กรรมวิธีของ Dual Tone Multifrequency (DTMF) ในการเลือกฟังก์ชันในการติดต่อกับ voice mail box โดยทั่วไปจะมี 12 ปุ่ม แบ่งเป็น 4 แถว และ 3 คอลัมน์ และ เครื่องโทรศัพท์แบบ มี 16 ปุ่ม โดยเพิ่มคอลัมน์ ดังตาราง

High Group Frequency (Hz)

	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	

ตารางแสดงความถี่ผสมที่ใช้ในโทรศัพท์แบบกดปุ่ม

ความถี่ที่ใช้ในแต่ละแถวและคอลัมน์ จะมีความถี่ต่างกัน ความถี่ของทั้ง 4 แถว เรียกว่าเป็นกลุ่มความถี่ต่ำ (Low Group Freauency) และความถที่ทั้ง 4 คอลัมน์เรียกว่าเป็นกลุ่มความถี่สูง(High Group Frenquency)การกดที่หมายเลขใดๆจะทำให้วงจรอิเลคทรอนิคส์ภายในเครื่องโทรศัพท์ผลิตความถี่ออกมา 2 ความถี่ เช่น กดเลข 5 ความถี่ที่ผลิตออกมาคือ 770 Hz และ 1336 Hz เป็นต้น

- มาตรฐานความถี่ ที่ใช้กันและตำแหน่งเลขหมายต่างๆ จะถูดจัดไว้ให้มีลักษณะ ดังรูป สำหรับความผิดพลาดที่ยอมให้เกิดขึ้นได้จะเป็น 1.5 เปอร์เซนต์ สำหรับการผลิตความถี่ และ 2.5 เปอร์เซนต์ สำหรับการรับหมายเลข

ระบบสัญญาณต่างๆของโทรศัพท์

สัญญาณต่างๆที่ใช้ในระบบโทรศัพท์ ประกอบด้วย

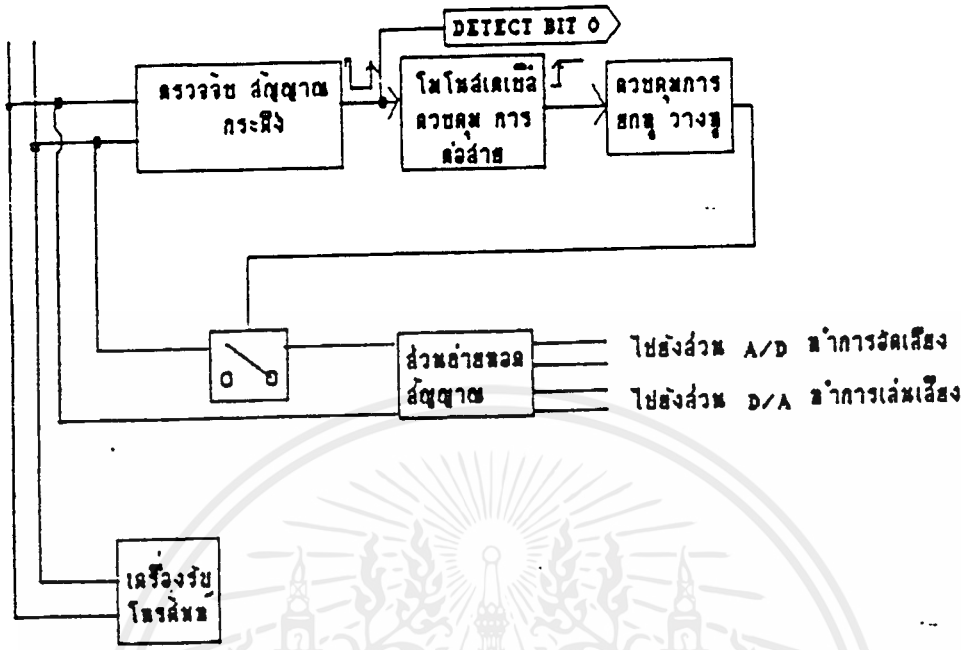
1. สัญญาณเรียก (Ringing Tone) เป็นสัญญาณที่ใช้ในการบอกแก่ผู้ที่อยู่ปลายทางว่าขณะนี้มีผู้ต้องการจะติดต่อด้วยทำการเรียกเข้ามายังโทรศัพท์เครื่องนั้น สัญญาณนี้จะเป็นสัญญาณไฟสลับ (AC) ที่มีขนาดประมาณ 48 โวลท์ และความถี่ 50 เฮิร์ท โดยจะดัง 1 วินาที และดับ 3 วินาทีสลับกันไป จนกว่าจะมีผู้รับสายหรือครบตามเวลาที่กำหนด

2. สัญญาณให้หมุน (Dial Tone) เป็นสัญญาณที่บอกแก่ผู้เรียกเมื่อผู้เรียกยกหูโทรศัพท์เพื่อจะติดต่อไปยังคู่สายปลายทาง ให้หมุนหมายเลขที่ต้องการจะติดต่อนั้นได้ โดยจะเป็นสัญญาณไฟสลับขนาด 5 โวลท์ความถี่ 400 เฮิร์ท

3. สัญญาณไม่ว่าง (Busy Tone) เป็นสัญญาณที่ใช้บอกแก่ผู้เรียกหลังจากที่ผู้เรียกได้ทำการหมุนหมายเลขที่ต้องการจะติดต่อนั้นเรียบร้อยแล้ว ว่าขณะนี้คู่สายปลายทางที่ต้องการติดต่อด้วยกำลังใช้งานอยู่ จึงไม่สามารถตัดต่อคู่สายให้ได้ โดยจะเป็นสัญญาณไฟสลับขนาด 5 โวลท์ ความถี่ 400 เฮิร์ท ดัง 0.5 วินาทีสลับกันไป

4. สัญญาณเรียกกลับ (Ring back Tone) สัญญาณนี้จะเกิดขึ้นในลำดับเดียวกับสัญญาณไม่ว่าง แต่จะเป็นการแจ้งให้ผู้เรียกทราบว่าสามารถติดต่อกับคู่สายปลายทางได้ และรอให้ผู้ที่อยู่ปลายทางทำการตอบรับสัญญาณเรียกนั้นอยู่ โดยจะเป็นสัญญาณไฟสลับขนาด 5 โวลท์ ความถี่ 400 เฮิร์ท ดัง 1 วินาที และดับ 3 วินาทีเช่นเดียวกับสัญญาณเรียก

ระบบส่วนตรวจจับกริ่งโทรศัพท์



บล็อกไดอะแกรมของส่วนตรวจจับกริ่งโทรศัพท์

ระบบรับฝากข้อความทางโทรศัพท์อัตโนมัติ (เฉพาะส่วนที่ติดต่อกับเครื่อง com) ซึ่งจะเป็นส่วนตรวจจับสัญญาณกริ่ง ประกอบด้วยส่วนต่าง ๆ ดังต่อไปนี้คือ

1. ส่วนตรวจจับสัญญาณกระดิ่ง

2. ส่วนโมโนสเตเบิล

1. ส่วนตรวจจับสัญญาณกระดิ่ง

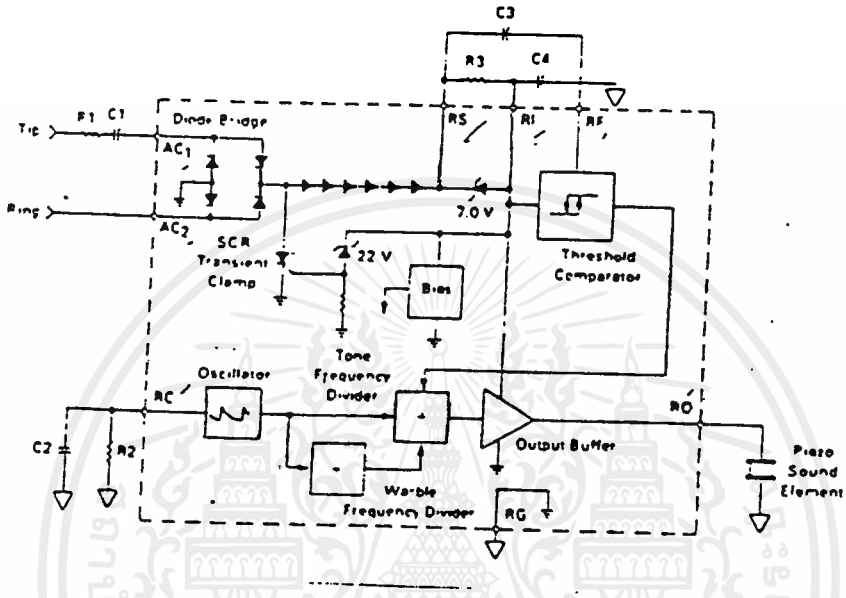
เป็นส่วนที่ใช้สำหรับตรวจจับสัญญาณกระดิ่งที่มาตามสาย ในขณะที่มีผู้โทรเข้ามา อาศัยหลักพื้นฐานของสัญญาณโทรศัพท์ คือ ในขณะที่สายว่าง คู่สายโทรศัพท์จะมีแรงดันประมาณ 48 โวลต์ ซึ่งจ่ายมาจากชุมสายโทรศัพท์ และเมื่อมีผู้เรียกเลขหมายเข้ามา ทางชุมสายจะจ่ายสัญญาณกระดิ่งมาเป็นแรงดันกระแสสลับที่มีแรงดันประมาณ 100 V. p-p ส่วนตรวจจับสัญญาณจะให้พัลส์ออกมาเมื่อมีกระดิ่งดังเข้ามา

1.1 หลักการทำงานของ IC MC34012

สัญญาณเรียกจากคู่สายโทรศัพท์ภายนอก ซึ่งมีลักษณะเป็นสัญญาณไฟฟ้าสลับขนาด 100 โวลต์จะถูกนำมาผ่านวงจรเรียกว่า(Full Wave Diode Bridge) ทางขาAC1และ AC2 ของ IC MC34012 (Telephone Tone Ringing) โดยมี R_1 ทำหน้าที่ควบคุมอินพุทอิมพีแดนซ์และจำกัดกระแสทรานเซียนต์จากสายโทรศัพท์ และ C_1 ทำหน้าที่ควบคุมอินพุตอิมพีแดนซ์ที่มีความถี่ต่ำ IC นี้ ได้ไฟเลี้ยงมาจากการเรคติไฟร์สัญญาณ AC ที่ป้อนเข้ามาทาง AC1 และ AC2 และจะทำให้ไฟนี้ไปทำให้วงจร Tone Generator (อยู่ภายใน IC) ซึ่งประกอบด้วยวงจร Relaxation Oscillator วงจรหารความถี่ (Tone Frequency Devider) ทำงานและวงจร Relaxation Time จะทำให้กำเนิดความถี่ f_o ที่ถูกกำหนดโดย R_2 และ C_2ที่ต่อเข้าที่ขา R_2โดยf_o จะมีค่าตั้งแต่1.0KHz ถึง 10 KHz ขึ้นอยู่กับการเลือกใช้อุปกรณ์ ในที่นี้ f_o มีค่า 4 KHz และทำให้มีสัญญาณเอาท์พุทที่ขา Tone Ringer Output (R_o) มีขนาด $f_o/4$ ถึง $f_o/5$ Hz (ประมาณ 800-1000 Hz) 20 V

สำหรับความต้านทาน R_3 ในวงจรจะเป็นตัวกำหนดตัวแอมปลิจูดของสัญญาณ Ringing ซึ่งเป็นสัญญาณเอาท์พุทออกที่ R_o โดยจะเริ่มจากสัญญาณที่เรคติไฟร์จากบริดจ์ไดโอดที่ขา AC_1 และ AC_2 จะทำให้เกิดค่ากระแสค่าหนึ่งที่ไหลผ่าน R_3ที่เป็นอินพุทของ R_1 โวลเตจที่ตกคร่อม R_3 จะถูกกรองโดย C_3 ณ จุดอินพุทที่เข้าสู่วงจร Threshold เมื่อโวจเตจที่ตกคร่อม C_3 จะมีค่าเกิน 1.7 โวลท์ จะทำให้ (Threshol Comparator) อีนาเบิลสัญญาณ Tone Ringing ออกมา (ค่า Line Transient ที่เกิดจากการหมุนโทรศัพท์ที่ชาร์จ C_3 จะยังมีค่าไม่มากพอที่ทำให้เกิด

สัญญาณ Tone Gen ออกมา) ส่วน C_4 จะทำหน้าที่กรองไฟเลี้ยงให้วงจร Tone Genertor สัญญาณเอาท์พุทที่ขา R_o จะถูกเรียงทิศกระแสโดยไดโอดบริดจ์แล้วส่งสัญญาณให้ออปโตคัปเปลอร์ (Opto Coupler) เปลี่ยนเป็นสถานะของลอจิกส่งให้ไมโครโปรเซสเซอร์ทราบโดยถ้ามีสัญญาณเรียกจะให้ค่าทางสถานะลอจิกเป็น "1" แต่ถ้าไม่มีสัญญาณเรียกจะส่งสถานะลอจิก "0" ออกไป



รูปที่ 2 แสดงบล็อคไดอะแกรมภายในไอซี MC34012

1.2 หลักการทำงานของ OPTOCOUPLE

การนำเอาออปโต้คัปเปลอร์เข้ามาใช้ในวงจร ก็เพื่อจุดประสงค์ในการแยกส่วนของแรงดันไฟสูงที่เกิดจากสายโทรศัพท์ซึ่งมีค่าประมาณ 200 V_{ac} ในขณะที่มีสัญญาณกริ่งดังเข้ามาให้เชื่อมต่อกับวงจรที่มีระดับไฟเลี้ยงขนาด 5 V_{dc} ซึ่งตัวออปโต้คัปเปลอร์นี้จะมีความสำคัญอย่างยิ่งที่จะต้องนำมาใช้ในงานเพื่อให้วงจรทีมีระดับแรงดันต่างกันมาก สามารถทำงานร่วมกันได้ ในส่วนของวงจรนี้ใช้ ออปโต้คัปเปลอร์ เบอร์ 4N25

สำหรับรายละเอียดของ 4N25 เป็นดังนี้

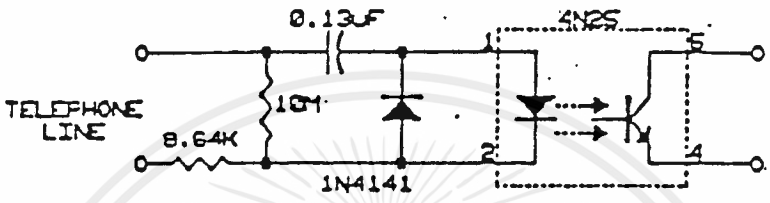
- วงจรมีโครงสร้างคล้ายกับ TTL มาตรฐาน
- ใช้ GaAS ไดโอดซึ่งปล่องแสดงอินฟาเรดไปยัง Silicon NPN

Phototransister

- การแปลง (Transfer) ของไฟกระแสตรงที่มีค่าสูง
- สามารถแยกวงจรที่มีระดับความดันที่ต่างกันได้ถึง + 25 KV
- ความเร็วในการสวิทช์ชิงค์สูง โดยมีค่า

  Rise time (t_r) = 2 μSec

  Fall time (t_f) = 2 μSec

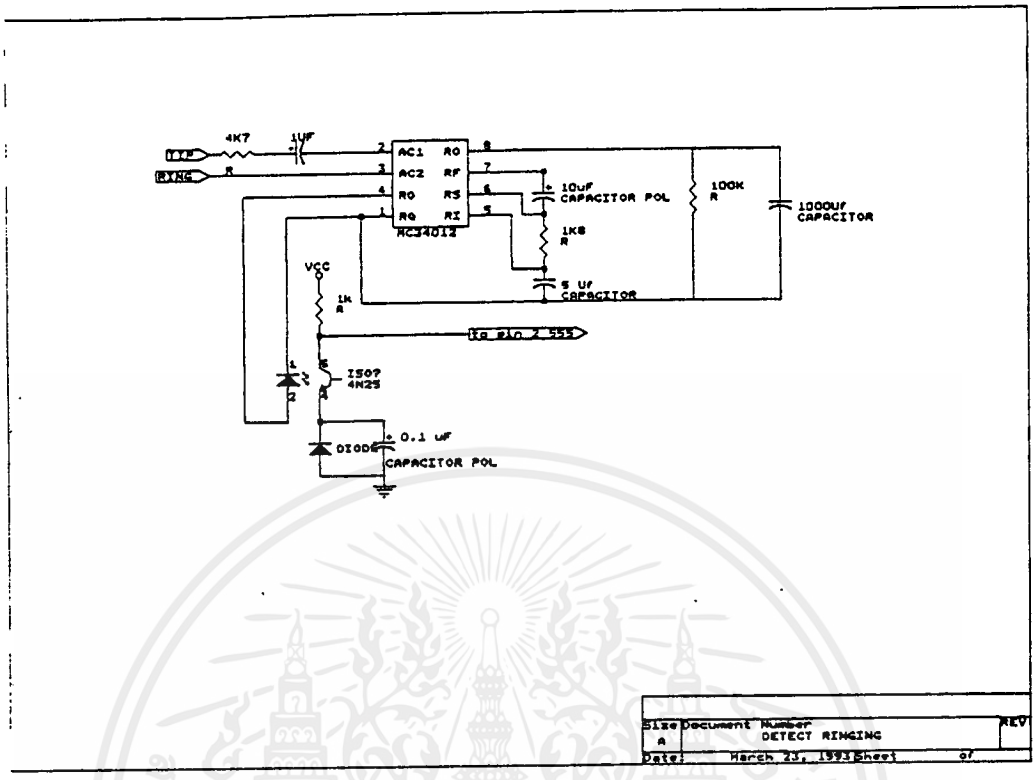


จากรูปที่ 3 เป็นวงจรซึ่งใช้งานได้จริง อุปกรณ์มีหน้าที่ดังนี้

- ค่าความต้านทาน 10M และ 9.64 K ทำหน้าที่เป็นตัวแบ่งแรงดัน (Voltage Divider) ให้ได้ขนาดที่เหมาะสมเพื่อทำการขับ (Drive) อินฟาเรด

- ค่าตัวเก็บประจุ 0.13 μF ทำหน้าที่กั้นไฟกระแสตรง ไม่ใช้ไดโอดอินฟาเรดทำงานเนื่องจากไม่ต้องการให้โฟโต้ทรานซิสเตอร์ (photransistor) ทำงาน Active ในสภาวะปรกติ แต่จะให้ทำงานในกรณีที่มีสัญญาณกริ่ง AC เท่านั้น

1.3 วงจร DETECT RINGING



- สัญญาณโทรศัพท์จะเข้าที่ขา 2 และ 3 เพื่อเป็นการเปลี่ยนสัญญาณที่ได้มาเป็นสัญญาณระดับต่ำ โดย IC MC 34012

- เอาท์พุต จะเป็นสัญญาณระดับต่ำ โดยจะอยู่ที่ขา 1 และ 4 ซึ่งจะออกมาเป็นรูป PLUS โดยมีขนาด 20 โวลท์ 1000 เฮิรทซ์ เรียกว่าสัญญาณ TONE RINGER และนำสัญญาณนี้ไปให้ แก่ OPTO-COUPLE

- OPTO-COUPLE 4N25 จะรับสัญญาณจาก MC 34012 แล้วทำให้เปิดสวิสต์ของ Transister ทำให้กระแสไหล เกิดสัญญาณลอจิกขนาด "1" เมื่อมีสัญญาณมาและเป็น "0" เมื่อไม่มีสัญญาณจาก MC 34012 แล้วส่ง OUTPUT ไปยังส่วน MONO-STABLE

2. โมโนสเตเบิล

เป็นส่วนที่จะให้พัลส์ 1 ลูกต่อสัญญาณกระดิ่ง 1 ครั้ง เป็นสัญญาณให้แก่ส่วนควบคุมการยกหู / วางหู

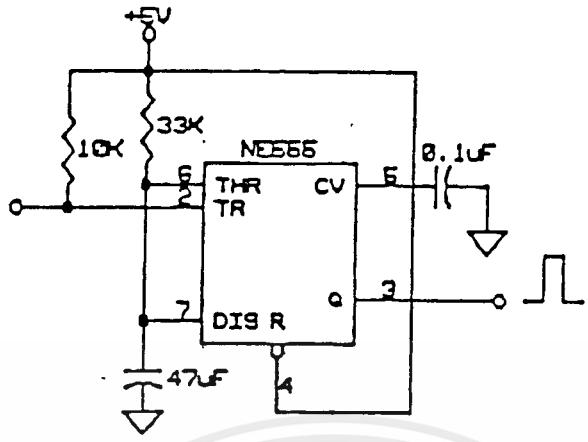
2.1 การทำงานของ IC NE555

เป็นวงจรที่เชื่อมกับโฟโต้ทรานซิสเตอร์ เพื่อทดสอบสัญญาณกริ่งของโทรศัพท์ที่โทรเข้ามาสัญญาณกริ่ง 1 ครั้งจะทำให้วงจรโมโนสเตเบิ้ล สร้างสัญญาณทริก (Trig) 1 ครั้งตามจังหวะของเสียงกริ่ง สำหรับวงจรโมโนสเตเบิ้ลของโครงงานนี้ ใช้ IC NE 555

รายละเอียดของ NE 555

- เป็น IC ที่มีการผลิต time delay และ ออสซิลเลเตอร์ได้ค่อนข้างจะถูกต้อง โดยในส่วนของ time delay จะใช้ร่วมกับอุปกรณ์ 2 ตัวคือ ความต้านทาน และ ตัวเก็บประจุ เพื่อทำการกำหนดค่า time delay เท่านั้น
- ปรับค่า duty cycle ได้
- เอาท์พุตขา 3 เป็นได้ทั้งจ่ายกระแส และดึงกระแสประมาณ 200mA
- เอาท์และไฟเลี้ยงมีระดับแรงดันเท่ากับ TTL มาตรฐาน

2.2 วงจรที่ใช้ในโครงงานนี้



จากรูปวงจร เป็นวงจรโมโนสเตเบิ้ล ซึ่งสามารถอธิบายได้ดังนี้

- ค่าของความต้านทานที่ขา Dis และ ตัวเก็บประจุที่ขา THR เป็นตัวกำหนดความกว้างของพัลส์(Pulse) โดยมีค่า

$$t = 1.1Rc$$

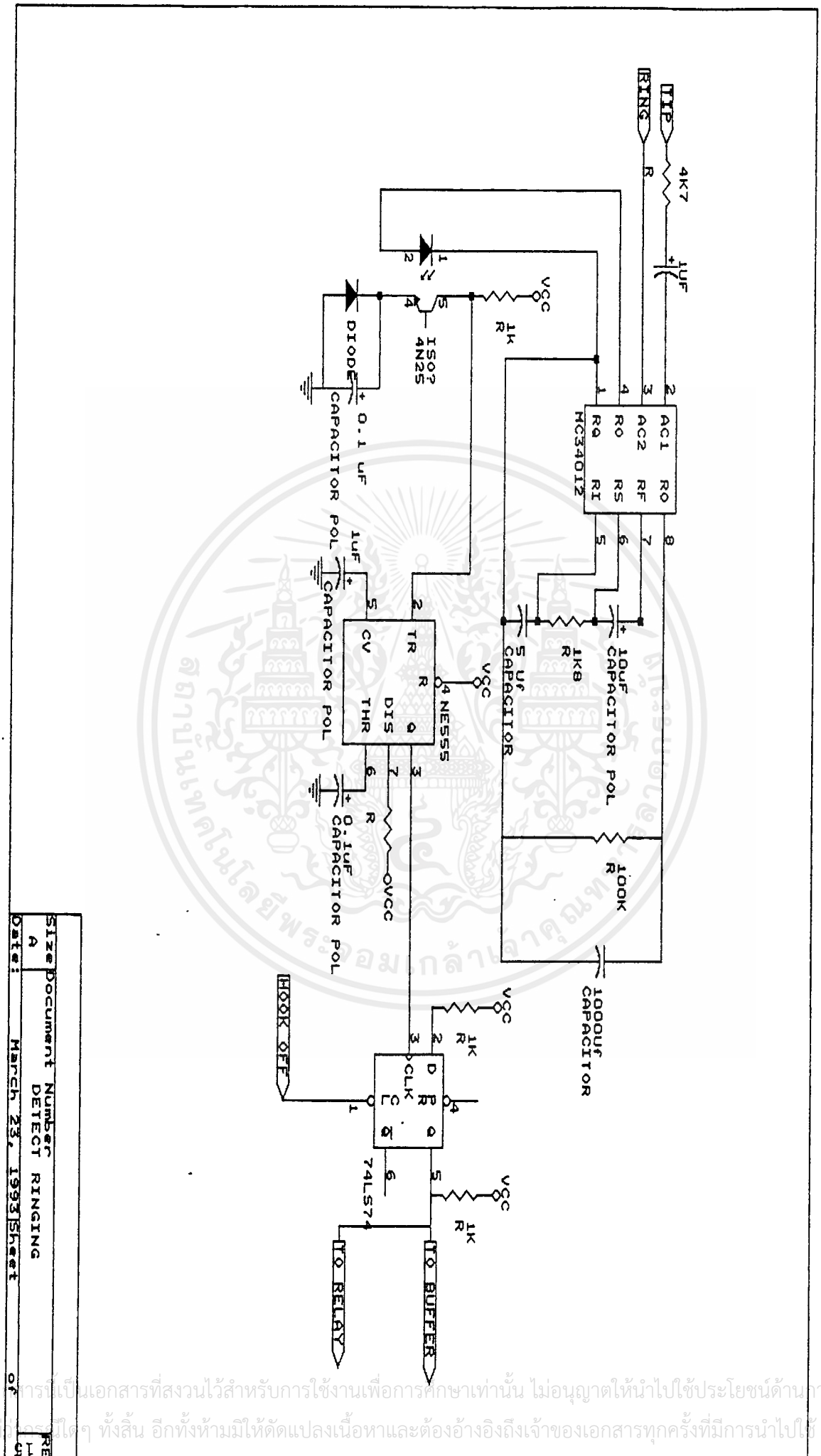
t ในวงจร ~ 1 sec

- ขารีเซ็ต (Reset) ทำหน้าที่รีเซ็ทไอซี NE555 ให้อยู่ในสภาพที่ทำงานในที่นี้ถูกต้องต่อกับระดับแรงดันไฟ 5 โวลท์

- ขาทริกเกอร์(Trigger) ทำหน้าที่รอรับสัญญาณทริกโดยจะทริกที่ขอบสัญญาณขาลง (Tailing edge) เท่านั้น

- OUTPUT เป็นสัญญาณพัลส์ 1 ลูกส่งต่อไปยัง D-FF เพื่อนำไปใช้ในการควบคุมการยกและวางหู และนำไปใช้ในโปรแกรม COMPUTER

3. วงจรรวมที่ใช้งานจริง



ส่วนตรวจจับสัญญาณทางโทรศัพท์

สัญญาณโทรศัพท์ประกอบด้วย

- สัญญาณให้หมุน (Dial Tone)
- สัญญาณไม่ว่าง (Busy Tone)
- สัญญาณเรียกกลับ (Ring Back Tone)

ซึ่งสัญญาณเหล่านี้เป็นสัญญาณที่มีความถี่ 400 Hz เท่ากันหมดแต่จะต่างกันที่ช่วงเวลาในการปรากฏความถี่ 400 Hz นี้เท่านั้น ดังนั้นการตรวจสอบสัญญาณทางโทรศัพท์ต่างๆ จึงใช้ IC LM567 ซึ่งเป็น Tone Decoder

1. ส่วนจับสัญญาณ

1.1 ส่วนประกอบของ IC LM567

เป็นวงจร Phas lock loop คือเป็นไอซีที่ให้สัญญาณ OUTPUT เป็น LOW ออกมาเมื่อ ความถี่ INPUT มีค่าเท่ากับความถี่ศูนย์กลาง (f_c) โดยสามารถตั้งความถี่ศูนย์กลาง และ Bandwidth เราสามารถตรวจจับความถี่ได้ตั้งแต่ 0.01 Hz - 500 kHz โดยมี Diagram ดังรูป

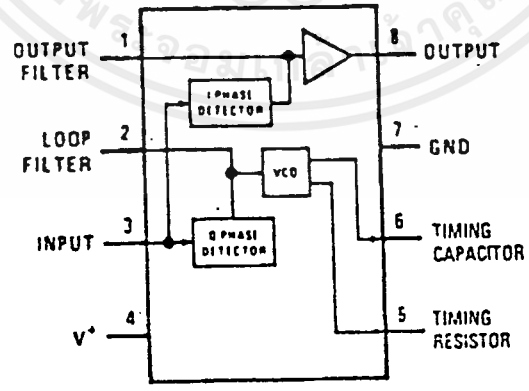


Diagram IC LM567

รายละเอียดและ หน้าที่ของแต่ละขา

- OUTPUT FILTER เป็นขาที่ใช้ใส่ C เพื่อเป็นตัวกรองสัญญาณที่จะส่งออกไปโดยค่า C จะกำหนดได้ดังนี้

$$C_3 = 2C_2$$

โดย C_3 = ค่า C ที่ขา 1

C_2 = ค่า C ที่ขา 2

- LOOP FILTER เป็นขาที่ใช้ใส่ C เพื่อเป็นตัวกรองสัญญาณที่จะรับเข้ามาโดยค่า C จะกำหนดได้ดังนี้

$$C_2 = n/f_o$$

ซึ่ง $1,300 < n < 62,000$

f_o = ค่าความถี่ศูนย์กลาง

- INPUT เป็นขาซึ่งใช้รับสัญญาณความถี่เพื่อนำมาเปรียบเทียบกับความถี่ศูนย์กลาง

- V^+ ไฟเลี้ยงบวก มีค่าได้ดังนี้

$$4.75 < V^+ < 9$$

- TIMING REGISTER เป็นขาที่ใช้กำหนดความถี่ศูนย์กลางร่วมกับขา TIMING - CAPACITOR หรือขา 6

- TIMING CAPASITER เป็นขาที่ใช้กำหนดความถี่ศูนย์กลาง โดยจะกำหนดค่าได้ดังนี้

$$f_o = 1/1.1(R_1C_1)$$

- GND ขา Ground ของ IC

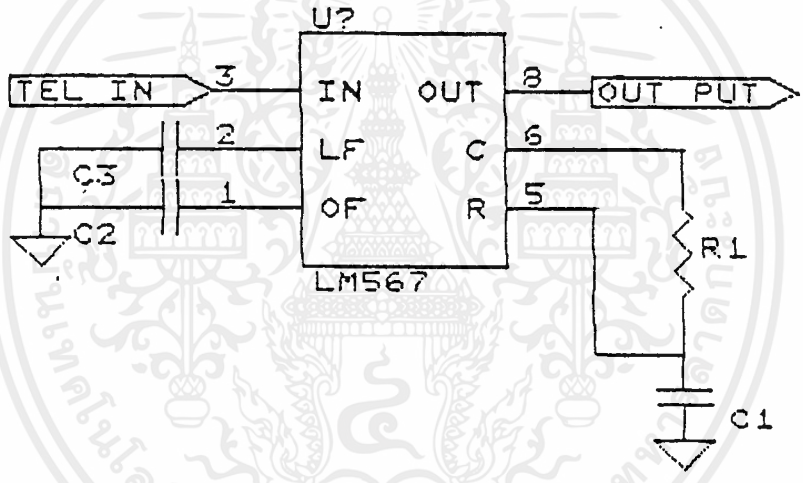
- OUTPUT เป็นขาเอาท์พุต ที่จะส่ง GND ออกมาเมื่อสัญญาณ INPUT มีค่าความถี่เท่ากับ ค่าความถี่ศูนย์กลาง

1.2 วงจร จับสัญญาณที่ใช้ในโครงงานนี้

เนื่องจาก PABX ที่นำมาต่อ ให้กำเนิดสัญญาณความถี่ไม่เท่ากับ ความถี่จากทางองค์การโทรศัพท์ดังนั้นจึงทำวงจรไว้ 2 ชุด โดย

- ใช้ตรวจจับสัญญาณโทรศัพท์ที่มาจาก PABX ซึ่งใช้ในการตรวจสถานะของการโอนสาย

- ใช้ตรวจจับสัญญาณโทรศัพท์ที่มาจาก สายนอก ใช้ในการโทรออกจาก PABX เราสามารถกำหนดความถี่ศูนย์กลางได้จากการกำหนดค่า R ที่ขา 5 และ C ที่ขา 6 ดังรูป



จะได้ค่าต่างๆ ดังนี้

- ความถี่ศูนย์กลาง (f_o) = $1/1.1(R_1C_1)$

  - จับสัญญาณจาก สายนอก

    f_o = 400 Hz

    C_1 = 0.47 μF

    R_1 ~ 5 K

  - จับสัญญาณจาก PABX

f_o = 350 Hz

C_1 = 0.47 μF

R_1 ~ 6 K

- ค่า C ที่ ขา 2 (LOOP FILTER)

  คำนวณจาก C_2 = n/f_o ซึ่ง $1,300 < n < 62,000$

  C_2 = 0.33 μf

- ค่า C ที่ ขา 1 (OUTPUT FILTER)

  คำนวณจาก C_3 = $2C_2$

  C_3 = 1 μF

- Bandwidth (BW) = $1070 \sqrt{[v_i/f_o C_2]}$

  v_i = Input voltage (volt rms) $v_i \leq 200$ mV

  C_2 = ค่า C ที่ขา 2 (μF)

2. ส่วนโมโนสเตเบิล

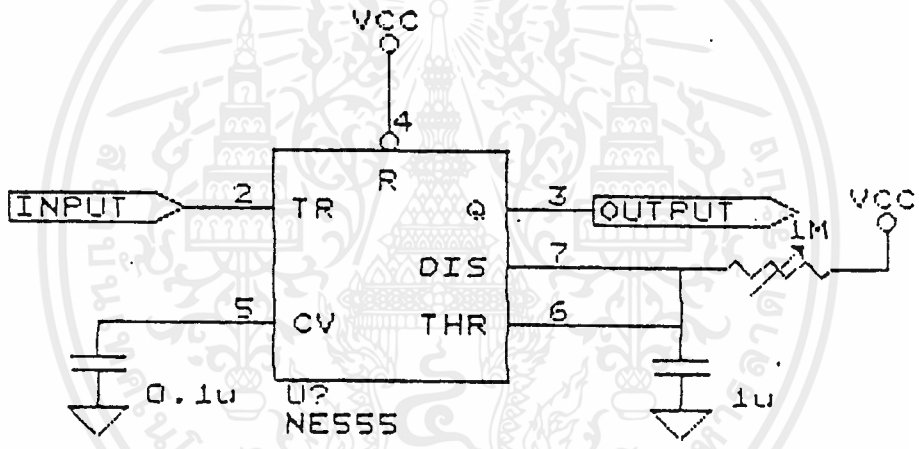
2.1 การทำงานของ IC NE555

เป็นวงจรที่เชื่อมต่อจาก 567 เพื่อเป็นการทำให้สัญญาณเรียบขึ้น ซึ่งรายละเอียดได้กล่าวมาแล้วข้างต้น

ค่าที่นำมาใช้งาน

ค่าของความต้านทานที่ขา Dis และตัวเก็บประจุที่ขา THR เป็นตัวกำหนด

2.2 วงจรที่ใช้ในโครงงานนี้



ความกว้างของพัลส์(Pulse) โดยมีค่า

t = 1.1RC

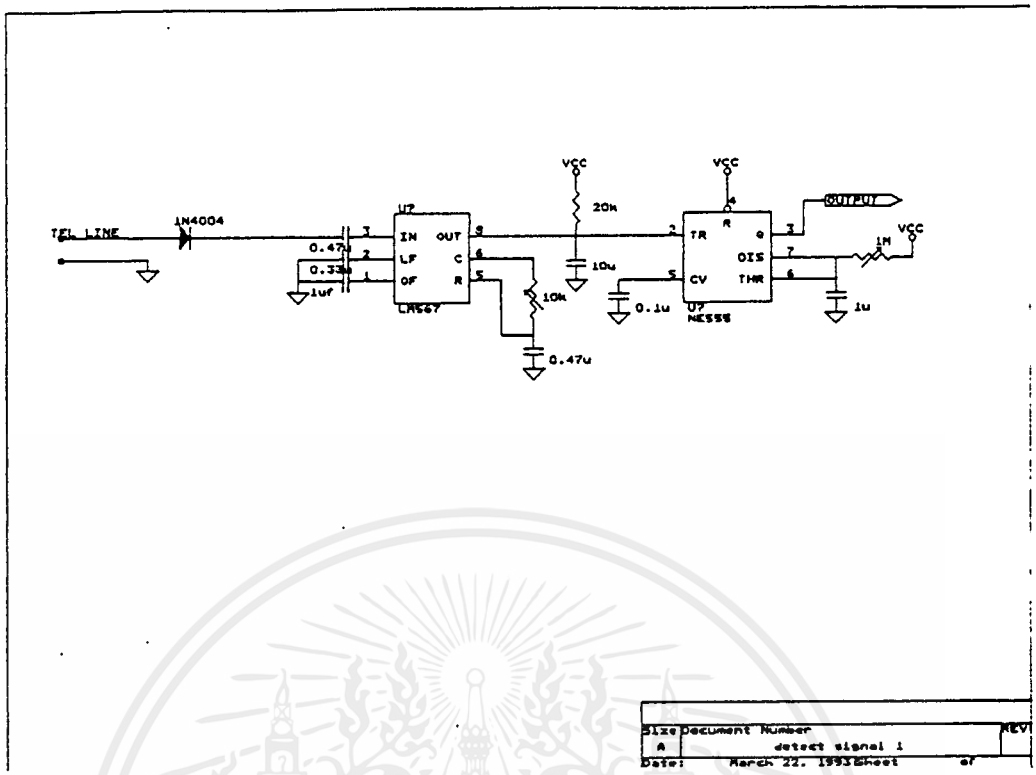
- จับสัญญาณจากสายนอก

t ~ 1 Sec

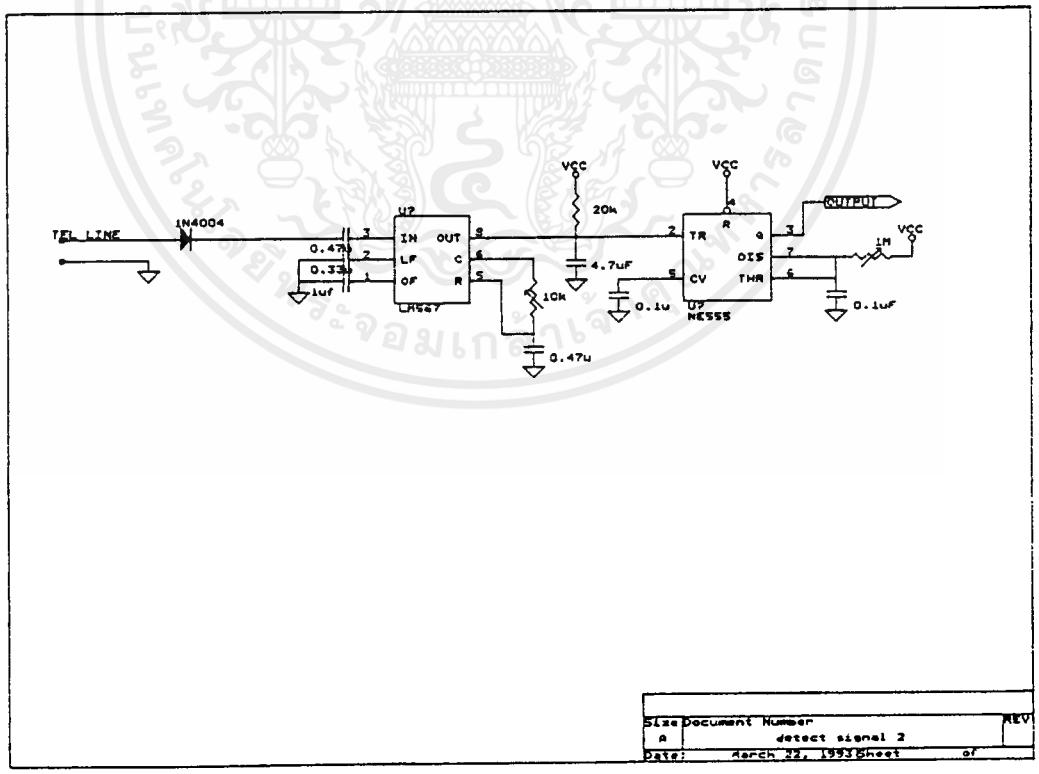
- จับสัญญาณจาก PABX

t ~ 0.1 Sec

3. วงจรรวมที่ใช้งานจริง



จับสัญญาณจากสายนอก

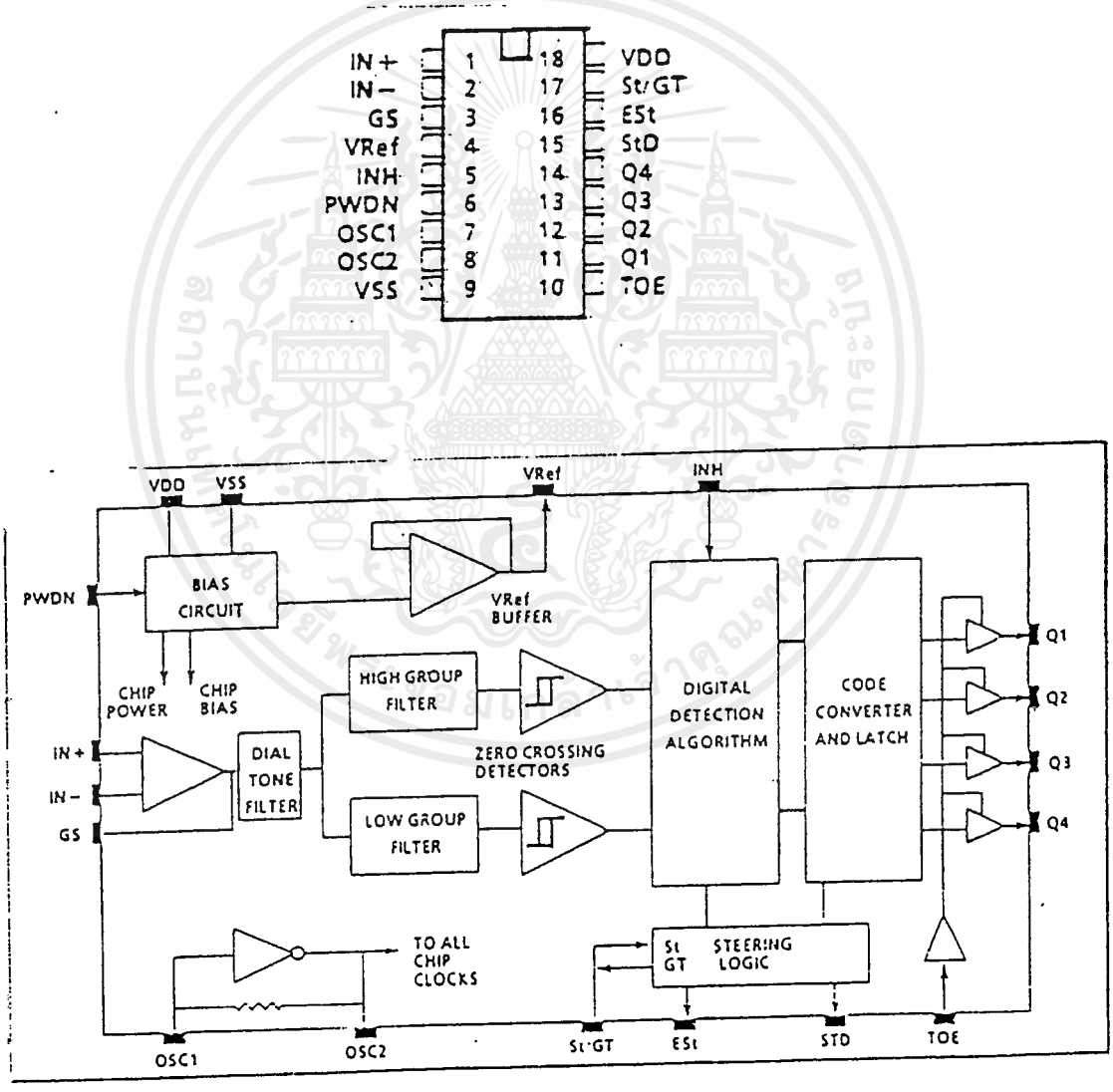


จับสัญญาณจาก PABX

วงจรแปลงความถี่ของระบบโทน (TONE) ของโทรศัพท์เป็นตัวเลขฐานสอง 4 บิท

เนื่องจากโครงการชิ้นนี้ต้องมีการติดต่อระหว่างคอมพิวเตอร์และในการที่ ผู้ใช้จะติดต่อกับ เครื่องคอมพิวเตอร์ จึงต้องทำการส่งผ่านทาง ปุ่มโทรศัพท์ โดยที่เราสามารถใช้ IC 8870 เป็นตัวที่ทำการถอดรหัส ให้กลายเป็น ตัวเลขฐานสอง 4 บิท

1. รายละเอียดของ IC MT8870



ข.แสดงแผนผังภายในของ MT8870

รูปที่ 1 รายละเอียดวงจร ของ MT8870

- รายละเอียด และหน้าที่การทำงานของแต่ละขา

- IN+ นอนอินเวอร์ทติ้งออปแอมป์อินพุต (Non - Inverting)
- IN- อินเวอร์ทติ้งออปแอมป์อินพุต (Inverting)
- GS เป็นขาที่ฟีดแบค (Feedback) จากเอาท์พุต ให้ผู้ใช้ได้เชื่อมตัวความต้านทานเพื่อเลือกค่า ในเกน(Gain) ได้ตามต้องการ
- VREF ค่าความต่างศักย์เปรียบเทียบที่เอาท์พุต ปกติมีค่า $V_{dd}/2$)
- IC* มีการเชื่อมต่อภายในจะถูกต่อเข้ากับ V_{ss})
- OSC1 อินพุตของสัญญาณนาฬิกา
- OSC2 เอาท์พุตของสัญญาณนาฬิกา โดยใช้กับผลึก (Crystal) ที่มีความถี่ 3.5795 MHz ต่อเข้า ระหว่าง OSC1 และ OSC2 เพื่อจ่ายให้กับ ออสซิลเลเตอร์ (Oscillor) ที่อยู่ภายใน
- V_{ss} ไฟเลี้ยงขั้วลบ
- Q1-Q4 เป็นขาเอาท์พุตที่ถูกควบคุมแบบ 3 State ด้วยสัญญาณ TOE จะให้ค่าออกมาเมื่อได้รับค่า ความถี่ที่ถูกต้อง และเอาท์พุตถูกอินาเบิล
- StD Delay Steering output ให้ค่าลอจิก เป็น 1 เมื่อมีข้อมูลระดับแรงดัน
- EST Early Steering output จะให้ค่าโลจิก .1. เมื่อข้อมูลถูกต้อง และถ้าเป็น 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 เป็นความต้านทานที่ถูกต่อแบบการป้อนกลับแบบลบ (Negative feedback)
2. V_{REF} ต่อเข้าขา IN+ เพื่อให้ระดับสัญญาณที่เข้ามา ถูกไบอัสให้อยู่ที่กึ่งกลางของระดับไฟเลี้ยงเท่ากับ $V_{dd}/2$
3. คริสตัล 3.579575 ถูกต่อคร่อมระหว่าง OSC1 และ OSC2 เพื่อให้เกิดการผลิตสัญญาณนาฬิกาภายใน ไอซี

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

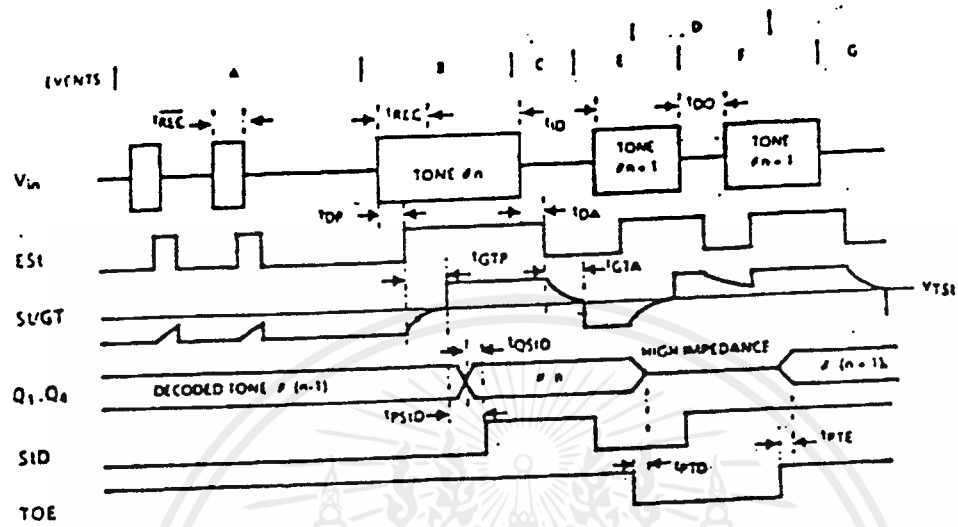
$$T_{TOA} = (RC)\ln[V_{dd}/V_{th}]$$

$$TGTP = (RC)\ln[V_{dd}/(V_{dd}-V_{Tth})]$$

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

ตารางแสดงความถี่ค่าต่างๆที่ถูก 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) มีค่าผิดพลาดเนื่องจากเวลาน้อยเกินไปเอาท์พุตจะไม่มีการเปลี่ยนแปลง

B โทน #n ถูกตรวจสอบพบ และช่วงเวลาถูกต้อง สัญญาณโทนจะถูก decode และ ถูก แลตช์(latch) ไว้ที่เอาท์พุต

C ไอซีตรวจสอบพบสัญญาณโทนสิ้นสุด เอาท์พุตยังคงรักษาข้อมูลเดิมไว้จนกว่ามีโทนใหม่ที่ถูกต้องเข้ามา

D เอาท์พุตถูกทำให้อยู่ในสภาวะไฮอิมพีแดนท์

F ช่วงเวลาของ โทน #n+1 ถูกตรวจว่าสิ้นสุดแล้ว เอาท์พุตยังคงรักษาข้อมูลเดิมไว้จนกว่าจะมีโทนใหม่ที่ถูกต้องเข้ามา

สำหรับสัญลักษณ์ต่างๆ ที่ใช้ มีความหมายดังนี้

t_{rec} คือ ช่วงเวลาอย่างน้อยสุดที่โทนจะถูกตรวจพบ (DETECT) ได้อย่างถูกต้อง

t_{ID} คือ ช่วงเวลาอย่างน้อยสุดระหว่างโทนแต่ละครั้งที่ถูกกดเข้ามา

t_{DO} คือ ช่วงเวลามากสุดที่ถูกยอมรับได้ในกรณีที่สัญญาณโทนถูกทำให้หายไปชั่วขณะ

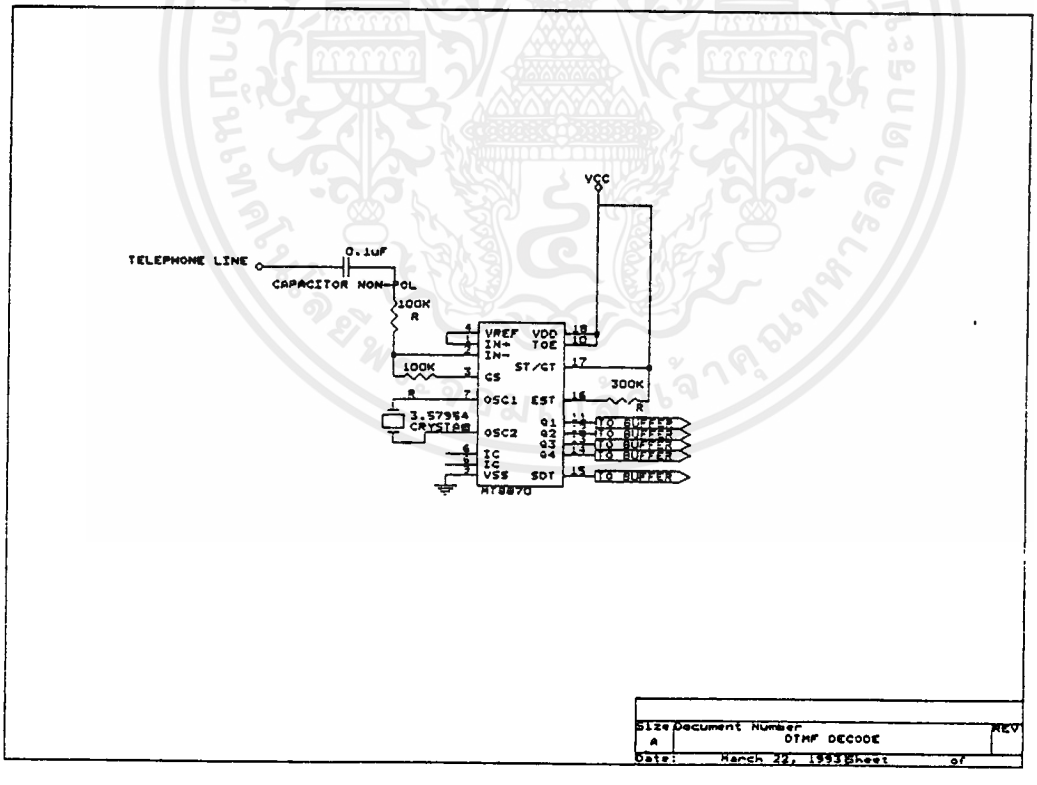
t_{DP} คือ เวลาที่ตรวจสอบพบการมาของสัญญาณที่ถูกต้อง

t_{DA} คือ เวลาที่ตรวจสอบพบการหายไปของสัญญาณที่ถูกต้อง

t_{GTP} คือ สัญญาณ guard time ปรากฏ

t_{GTA} คือ สัญญาณ guard time ไม่ปรากฏ

2. วงจรรวมที่ใช้งานจริง



- จะรับสัญญาณ มาจากสายโทรศัพท์

- จากวงจร จะทราบ GARD TIME คือ

$$T_{TGA} = (RC)\ln[V_{dd}/V_{ss}]$$

$$= 0.048 \text{ sec}$$

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

- การที่จะตรวจสอบว่า ค่านั้นจะกดได้หรือยัง สามารถตรวจสอบจาก ขา SDT ซึ่งถ้ามี รหัสตัวใหม่เข้ามา จะเกิด PULSE 1 ลูก

ส่วนสร้างสัญญาณ DTMF

เนื่องจากโทรศัพท์แบบ TONE (แบบกดปุ่ม) จะมีการทำงานโดยถ้าเรากดปุ่ม เครื่องจะทำการสร้างความถี่ขึ้น 2 ชนิดแล้วทำการรวมสัญญาณ เพื่อส่งไปตามสาย โทรศัพท์เข้าชุมสาย โดยจะมีความถี่ดังนี้

	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 - 8 รายละเอียดดังรูป

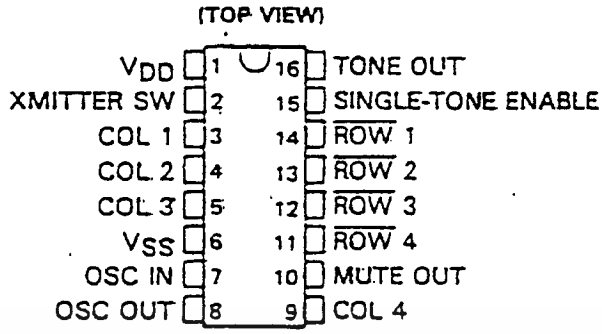
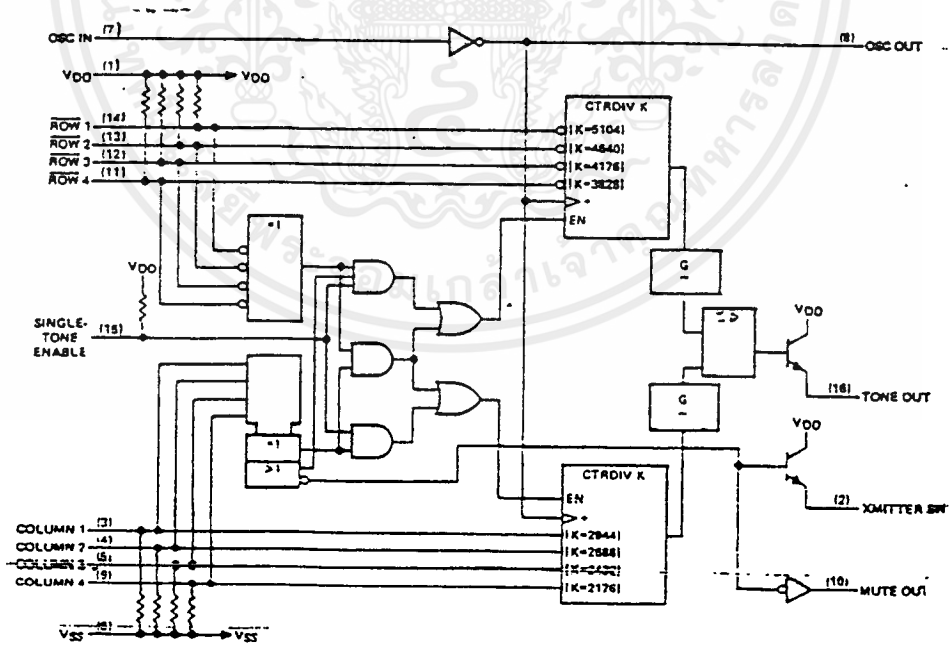


Diagram การทำงานของ IC TCM 5087



รูปแสดงขาต่างๆ IC TCM 5087

รายละเอียดและ หน้าที่ของแต่ละขา

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

- XMITTER SW จะเป็น High IMPEDENCE เมื่อขา column ใดก็ได้ active และ เป็น High เมื่อไม่มีขา column ใดๆ active

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

- V_{SS} เป็น Ground ของไอซี

- TONE OUT เป็นขา OUTPUT โดยจะประกอบด้วยความถี่ 2 ชนิดซึ่งได้จากการกำหนดที่ขา column และ ขา row

- SINGLE-TONE ENABLE เป็นขาที่ใช้กำหนดให้ OUTPUT ออกมาเพียงความถี่เดียว

- ROW1 - ROW2 เป็นขา INPUT เพื่อกำหนดค่าความถี่ active ที่ low

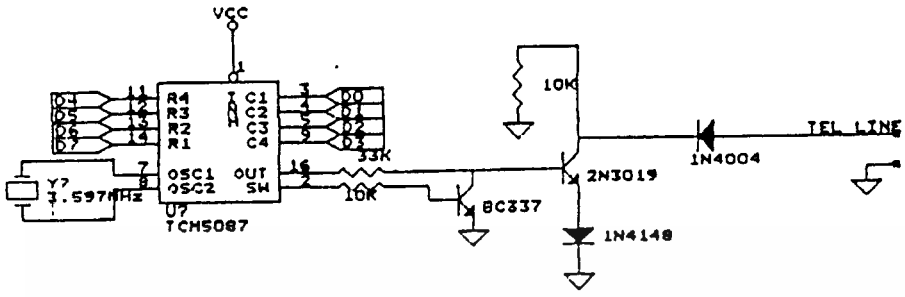
- MUTE OUT จะเป็น High เมื่อ column ใดก็ได้ active และเป็น low เมื่อไม่มีขา column ใดเลยที่ active

การส่งข้อมูลให้ IC สร้างความถี่ให้เท่ากับที่เรากดโทรศัพท์สามารถส่งค่าให้กับขาต่างๆ โดยที่ขา R_1 - R_4 จะทำงาน (active) ที่ Logic 0 ส่วน C_1 - C_4 จะทำงาน (active) ที่ Logic 1 การหมุนเบอร์จะต้องกำหนดค่าให้กับขาต่างๆ 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

1.2 วงจรที่ใช้งาน



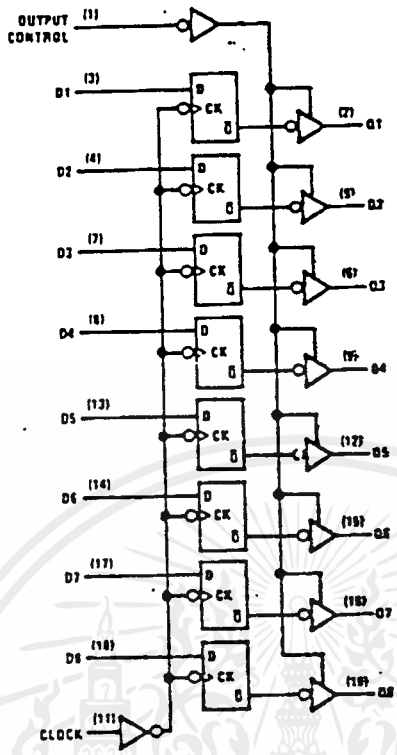
2. ส่วน LATCH ข้อมูล

เนื่องจากการให้ INPUT เข้าไปยัง IC 5087 เพื่อที่จะกำเนิดความถี่จะต้องให้ค่า INPUT ตลอดเวลาดังนั้นเราจึงใช้ IC 74374 ซึ่งเป็น LATCH เป็นตัวช่วย

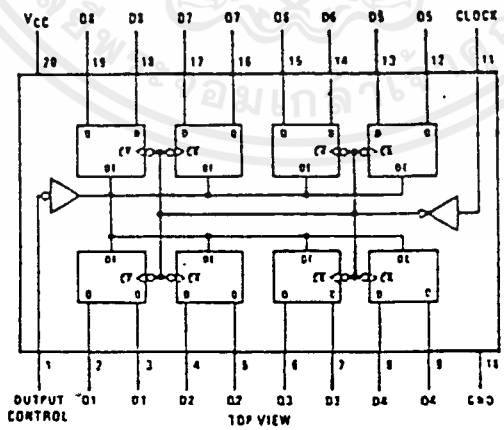
2.1 ส่วนประกอบของ IC 74374

- เป็นที่ประกอบด้วย D flip-flop ทั้งหมด 8 ตัว
- เป็นไอซีประเภท Tri-state ที่ OUTPUT
- การนำข้อมูลเข้าเป็น PARALLEL
- มี BUFFER ที่ขา INPUT

DIAGRAM ของไอซี 74374



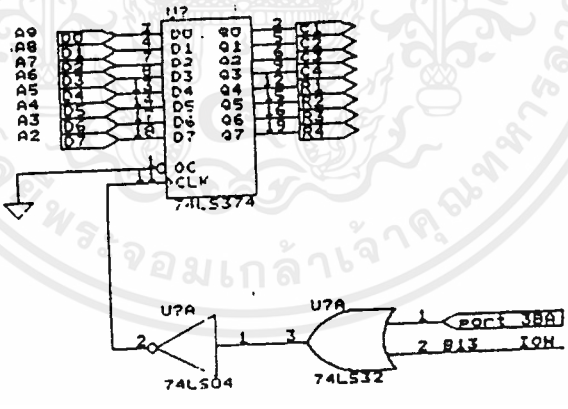
รายละเอียด และหน้าที่ของขา ไอซี 74374



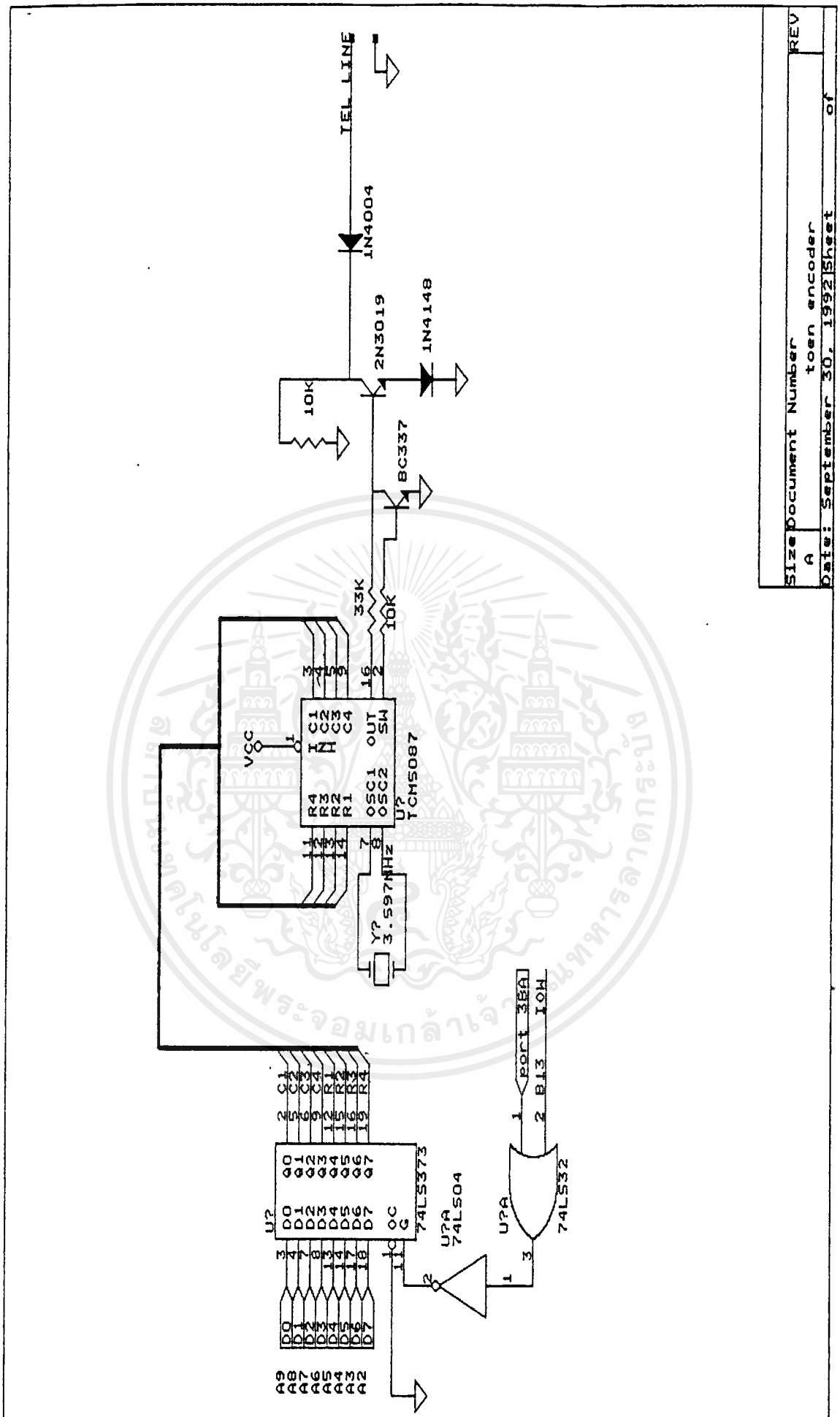
รูปแสดงขาไอซี 74374

- OUTPUT CONTROL เป็นขาใช้ควบคุม OUTPUT ถ้าขานี้เป็น High ขา OUTPUT ทั้งหมดจะเป็น High impedance
- D1 - D8 เป็นขา INPUT ของไอซี
- Q1 - Q8 เป็นขา OUTPUT ของไอซี ซึ่งจะค้างค่าเดิมไว้จนกว่าจะมี INPUT ค่าใหม่ พร้อม ENABLE เข้ามา
- GND ขา Ground ของ ไอซี
- ENABLE เป็นขาที่ใช้เปิด Flip-Flop เพื่อให้ค่าใหม่สามารถเข้ามาได้ ถ้าขา ENABLE เป็น High เมื่อมี INPUT เข้ามา OUTPUT จะเท่ากับ INPUT ทันที
- V_{cc} ไฟเลี้ยงของไอซี โดย $4.5 < V_{cc} < 5.5$

2.2 วงจรที่ใช้งาน



3. วงจรรวมที่ใช้งานจริง (หน้าถัดไป)



ส่วนควบคุมการยกหู / วางหู

เป็นส่วนที่ทำการรับสัญญาณจากส่วนโมโนสเตเบิล ทำหน้าที่ให้ระดับแรงดัน "0" เมื่อมีสัญญาณกระดิ่งเข้ามา ส่วนนี้สร้างจากวงจรฟลิบฟลอปซึ่งสามารถดัดแปลงให้สามารถนับจำนวนครั้งของกระดิ่งให้สามารถนับเป็นจำนวนกี่ครั้งก่อนจะยกหูก็ได้

1. ส่วนของ D-FF

1.1 จุดประสงค์

จะนำ D-FF เป็นตัวทำให้ค่าสัญญาณ latch เพื่อนำไปใช้

- ทำการติดต่อกับ RELAY เพื่อทำการ SIMULATE ยกหู เพื่อทำให้ส่วนที่ใช้ในการถ่ายทอดสัญญาณ สามารถติดต่อกับโทรศัพท์ได้
- ส่งข้อมูลนี้ไปยัง COMPUTER เพื่อทำการ เช็คว่ามีโทรศัพท์เข้ามาหรือไม่

1.2 ตัว D-FF จะได้ INPUT มาจาก 2 ส่วน คือ

- จาก COMPUTER ส่งค่า "0" มาที่ขา CLR เพื่อเป็นการส่งสัญญาณบอก RELAY เพื่อทำการปิด สวิทช์ (SIMULATE วางหู)
- จาก COMPUTER ส่งค่า "0" มาที่ขา PR เพื่อเป็นการส่งสัญญาณบอก RELAY ให้ทำการเปิด สวิทช์ (SIMULATE ยกหู)
- จาก วงจร DETECT RINGING ส่งค่า "1" มาเพื่อส่งไปยัง RELAY เพื่อทำการ SIMULATE ยกหู

2. ตัว RELAY

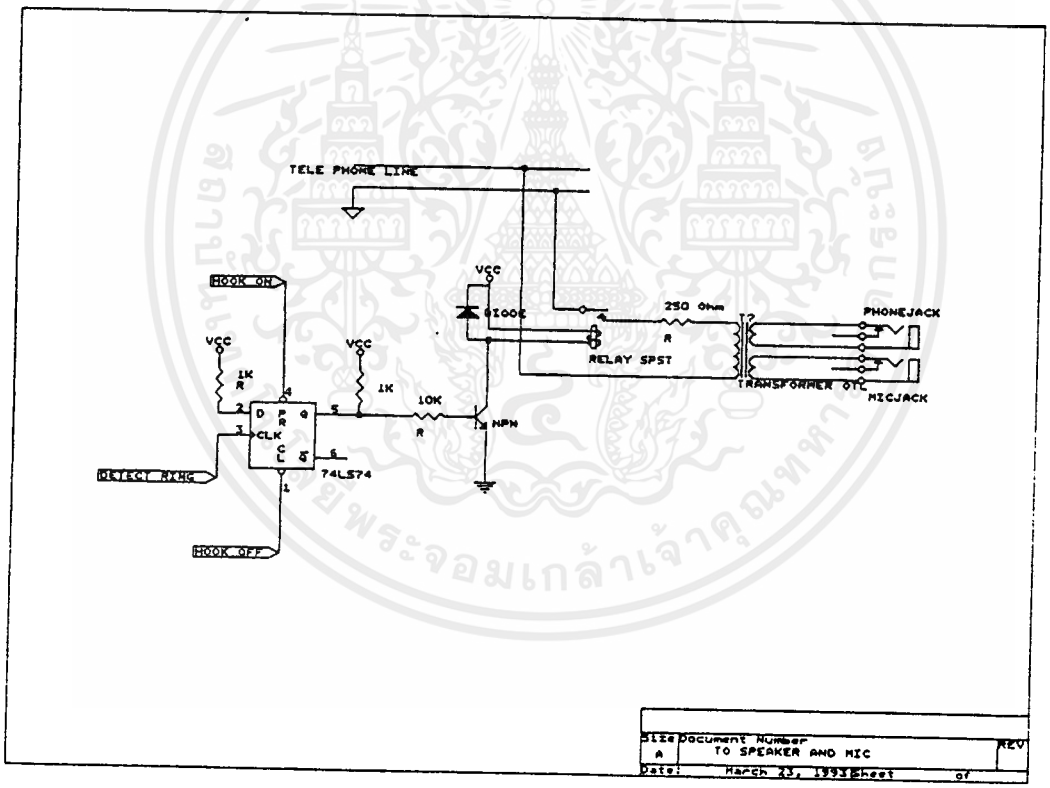
2.1 คุณสมบัติ

เป็นตัวสวิทช์ เปิด-ปิด ซึ่งควบคุมโดยการป้อนไฟ 5 โวลท์ จะทำให้สวิทช์เปิด หรือถ้าเป็น 0 โวลท์ จะทำให้สวิทช์ปิด

2.2 การใช้งาน

เพื่อทำการ SIMULATE ยกหู คือ ทำให้ VOLTAGE ลดระดับลงเหลือประมาณ 5 โวลท์ ซึ่งจะใช้ R ขนาด 500 โอห์ม และ ส่วนนั้นจะต่อกับส่วนถ่ายทอดสัญญาณอีกทีหนึ่ง หรือทำการ SIMULATE วางหู ก็คือ ปิดสวิทช์ตัว RELAY นี้ให้มีขนาดโวลท์เดจ กลับกลายเป็น 50 โวลท์

3. วงจรรวมที่ใช้ในงานจริง



ส่วนถ่ายทอดสัญญาณ

1. ตัว TRANSFORMER

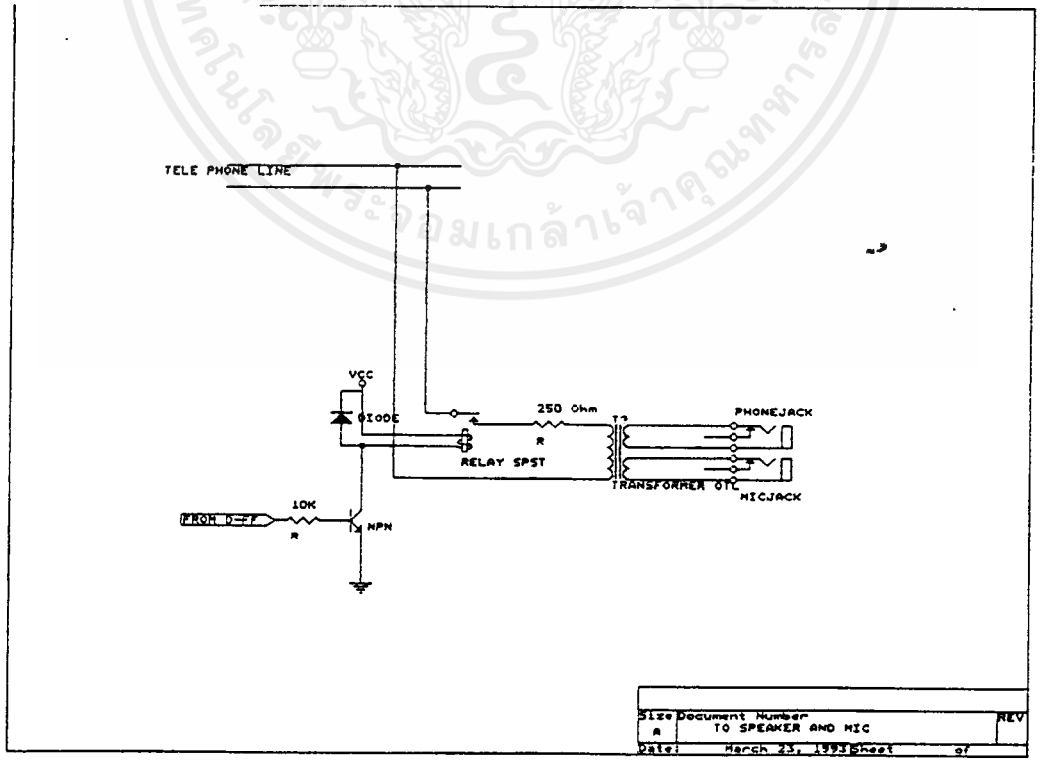
1.1 ส่วนของ TRANSFORMER

สร้างจาก Transformer ซึ่งมีคุณสมบัติเหนี่ยวนำสัญญาณข้ามขดลวด และแยกส่วนของวงจรออกจากกัน สัญญาณที่เหนี่ยวนำข้ามขดลวด จะถูกถ่ายทอดให้แก่วงจรของ SOUND CARD ซึ่งจะเปลี่ยนสัญญาณเสียงให้เป็นข้อมูลทางดิจิตอล เก็บไว้ในฮาร์ดดิสก์ ของเครื่องคอมพิวเตอร์ และเปลี่ยนข้อมูลที่เก็บไว้ให้กลายเป็นเสียงออกมาอีกขดลวด เมื่อทำหน้าที่เล่นเสียงให้ผู้ใช้บริการฟัง โดยส่วนนี้จะต่อจาก RELAY

1.2 ส่วนอีกฝั่งของ TRANSFORMER

จะเป็นตัวต่อไปยัง SOUNDCARD ในส่วนของ SPEAKER และ MICROPHONE เพื่อนำเสียงไปเก็บ หรือเล่นข้อมูลใน HARD DISK

2. วงจรในโครงงานนี้



การติดต่อกับ COMPUTER

1) ในระบบจะประกอบด้วย การ์ด 2 ตัว ได้แก่

- SOUND CARD ซึ่งเป็นการ์ดที่ใช้ทำหน้าที่เกี่ยวกับ เก็บเสียง หรือนำเสียง จาก HARD DISK มาทำการ เล่นให้กับผู้ที่โทรศัพท์เข้ามา

- CONTROL CARD เป็นส่วนที่หน้าที่ดังนี้
  - ตรวจสอบสัญญาณ RINGING
  - ตรวจสอบสัญญาณความถี่ 400 Hz
  - รับข้อมูลจาก DTMF
  - ส่งสัญญาณ ยกหู และวางหู
  - ส่งข้อมูลให้กับ TONE ENCODER

2) PORT ที่ใช้ในแต่ละ การ์ด

- SOUND CARD มี PORT ที่ใช้ 2 PORT คือ 2B8 และ 2B8 โดยมีฟังก์ชันการทำงานดังนี้

INPUT PORT

PORT 2B8H = IRQ_DISABLE

PORT 2B9H = IRQ_ENABLE

PORT 2BAH = RESET_IRQ

OUTPUT PORT

PORT 2B8H = PLAY_DATA

PORT 2B9H = SOUND_OFF

PORT 2BAH = SOUND_ON

PORT 2BBH = RECORD_DATA

- CONTROL CARD เป็น การ์ดที่ใช้ในการควบคุมการทำงานเกี่ยวกับโทรสัพท์ทั้งหมด ซึ่งจะมี 2 PORT เช่นกัน

3B8 จะเก็บข้อความเป็น bit โดยการ inport data เข้าเพื่อทำการตรวจสอบ

D_1	D_2	Q_3	Q_2	Q_1	Q_0	Std	R

บิท 8 บิท 0

R เป็น 1 เมื่อมีสัญญาณ RINGING

STD เป็น 1 เมื่อ DTMF ได้รับข้อมูลใหม่

$Q_0 - Q_3$ แทนข้อมูลที่ถอดรหัส ใน DTMF

D_1 เป็น 1 เมื่อมีสัญญาณ 400 Hz เข้ามา

D_2 เป็น 1 เมื่อมีสัญญาณจาก PABX เข้ามา

3B9 จะเป็น PORT ที่ใช้ในการส่งสัญญาณโดยการ IN PORT นี้ไปเพื่อทำการบอกให้วงจรทำการวางหูโทรศัพท์เพื่อรอรับการโทรเข้ามาในครั้งต่อไป โดยนำสัญญาณ ADDRESS 3B9 ผ่านทาง 74LS138 มา OR กับ IO_READ และนำสัญญาณไปเข้าขา CLR ของ 74LS74 เพื่อทำการ ปิดสวิสต์ RELAY ทำการวางหูโทรศัพท์ (คือทำให้ VOLTAGE มีค่า 50 โวลท์)

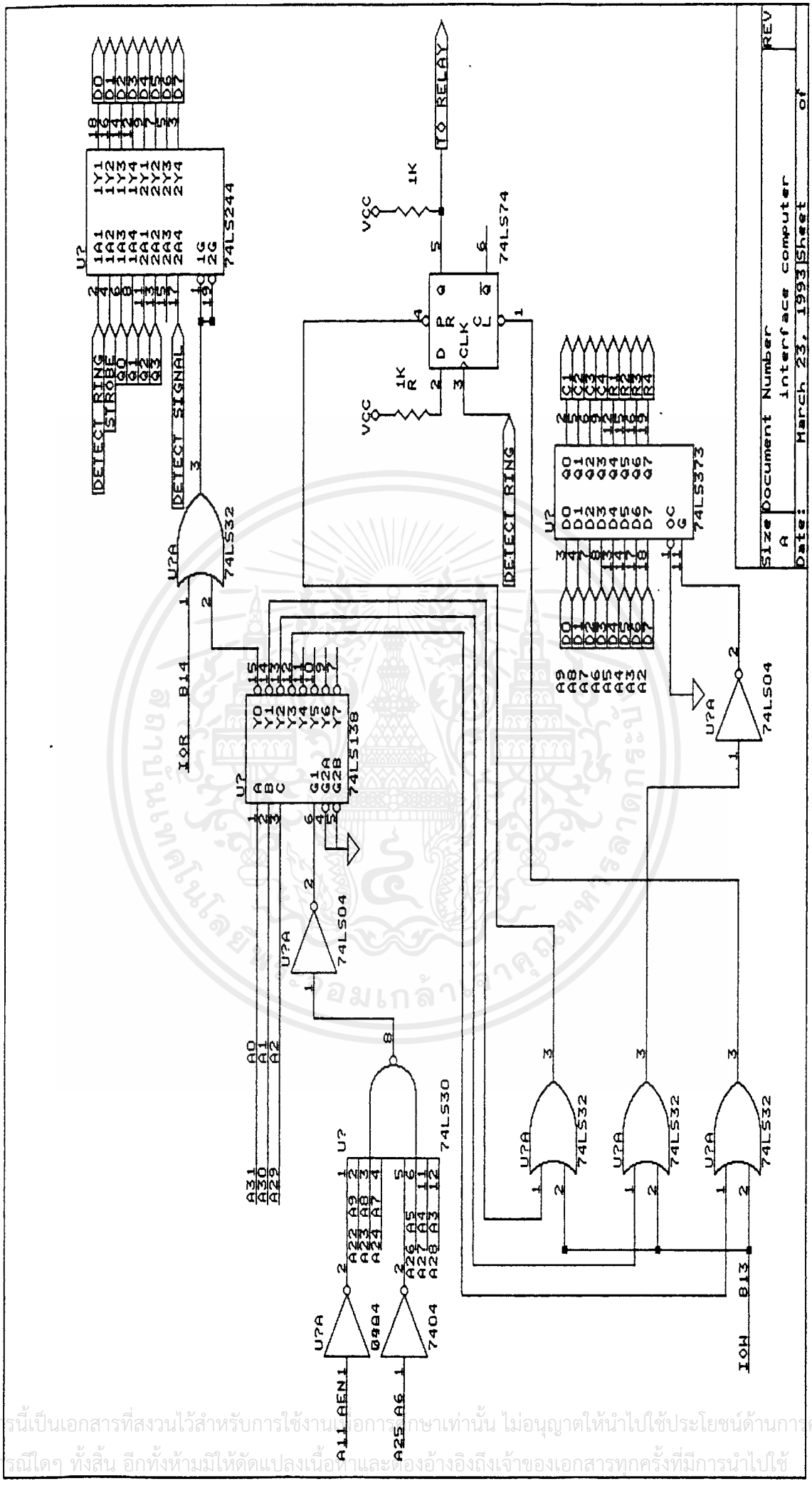
3BA จะเป็น PORT OUTPUT ที่ใช้ในการนำข้อมูลเข้าจาก COMPUTER ส่งให้กับ 5087 TONE ENCODER เพื่อทำการกำเนิดเสียง โดยจะส่งทั้งหมด 8 บิทดังที่กล่าวมาข้างต้น โดยผ่าน 74LS373 ก่อน ได้ดังรูป

R_1	R_2	R_3	R_4	C_1	C_2	C_3	C_4

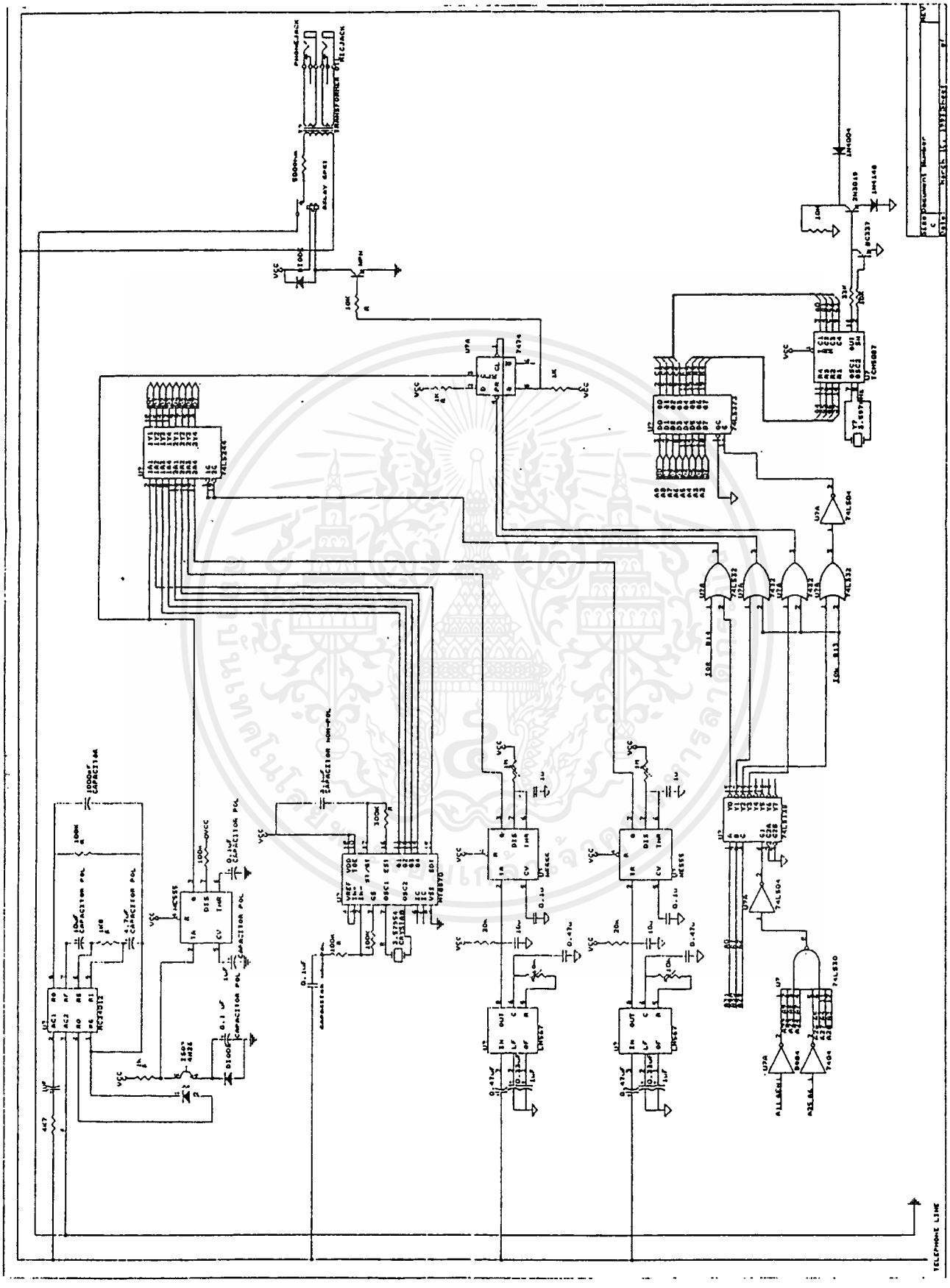
บิท 8 บิท 0

3BB จะเป็น PORT ที่ใช้ในการส่งสัญญาณโดยการ INPORT นี้ไปเพื่อทำการบอกให้วงจรทำการยกหูโทรศัพท์ โดยนำสัญญาณ ADDRESS 3BB ผ่านทาง 74LS138 มา OR กับ IO_READ และนำสัญญาณไปเข้าขา PR เพื่อทำการ SET D-FF ให้ OUTPUT เป็น 1 เป็นการเปิดสวิทช์ RELAY (VOLTAGE มีค่า 5 โวลท์)

3. วงจรที่ใช้ในการติดต่อกับคอมพิวเตอร์



วงจรรวมทั้งหมด



การทำงานของฟังก์ชันต่างๆ ที่สำคัญ

ในไฟล์นี้ เป็นไฟล์ที่ประกอบด้วยฟังก์ชันที่ทำหน้าที่ตรวจสอบสัญญาณต่าง ๆ ในสายโทรศัพท์ ในการทำงานจะต้องใช้ตัวแปรที่ทำงานร่วมกับฮาร์ดแวร์ดังต่อไปนี้

CONTROL_ADDR มีค่า 0X3B8 ใช้สำหรับติดต่อกับพอร์ท หมายเลข 3B8 ซึ่งแต่ละบิทมีความหมายต่างกันดังต่อไปนี้คือ

บิทที่ 0 0 หมายถึง ไม่มีสัญญาณกระดิ่ง
1 หมายถึง มีสัญญาณกระดิ่ง

บิทที่ 1 0 หมายถึง ไม่มีการกดปุ่มบนโทรศัพท์
1 หมายถึง มีการกดปุ่มหมายเลขโทรศัพท์

บิทที่ 2 - 5 ใช้สำหรับเก็บข้อมูลหมายเลขโทรศัพท์ที่ผู้ใช้บริการกดหนึ่งครั้ง

บิทที่ 7 - 8 ใช้สำหรับทดสอบสถานะของสายโทรศัพท์
เช่น สายว่าง สายไม่ว่าง สัญญาณ ring back tone

HOOK_OFF มีค่า 3B9 ใช้สำหรับเป็นพอร์ทอินพุท ที่เมื่อใช้คำสั่ง

```
inp(HOOK_OFF);
```

จะมีผลทำให้ดีโค้ดเดอร์ส่งสัญญาณไปเคลียร์ค่าของฟลิปฟลอปทำให้รีเลย์เปิดวงจร เปรียบเสมือนกับการวางหูของระบบ

DTMF_OUT มีค่า 3BA ใช้สำหรับเป็นพอร์ทเอาท์พุต เพื่อส่งรหัสแทนสัญญาณ DTMF ให้แก่ TCM 5087 เพื่อเปลี่ยนเป็น DTMF ใช้ในการโอนย้ายสาย ต่อไป

HOOK_ON เป็นพอร์ตอินพุต หมายเลข 3BB ใช้สำหรับจำลองการยกหู โดย

```
inp(HOOK_ON);
```

จะเป็นการสั่งให้ยกหูโทรศัพท์ เพื่อโอนย้ายสายต่อไป

DETECT เป็นค่าคงที่มีค่า 0X01 ใช้สำหรับแทนค่าที่อ่านได้จากสัญญาณกระดิ่ง

DTMF เป็นค่าคงที่มีค่า 0x02 ใช้สำหรับแทนค่าที่อ่านได้เมื่อมีการกดหมายเลขโทรศัพท์

PHONE_STATUS_IN เป็นค่าคงที่ 0x80 ในสำหรับตรวจสถานะในสายโทรศัพท์เมื่อทำการโอนสาย

PHONE_STATUS_OUT เป็นค่าคงที่ 0x40 ในสำหรับตรวจสถานะในสายโทรศัพท์เมื่อทำการโทรศัพท์ออกจาก PABX

ฟังก์ชันที่สำคัญต่างๆ

```
/*-------------------------------------------------------------*
|  Function     :  detect                                     |
|  Usage        :  int detect();                              |
|  Description  :  Wait for ringing                           |
|  Return       :  0 for ringing in                           |
|   ascii code  for keyboard hit                              |
|                  -1 for time out and call out to user       |
*-------------------------------------------------------------*/
int detect()
{
   int c,counter;
```

```
    counter = 0;
    while(!kbhit()) {
       if(((inp(CONTROL_ADDR)) & DETECT) == DETECT)
          return 0;
       else {
          delay(20);
          if(counter++ )= Maxcount)
             return  -1;
       }
    }
    if ((c = getch()) == 0 )
       c = getch() | 128;

    return c;
}
```

การทำงานจะทำการตรวจจับว่ามีการกด Keyboard หรือไม่

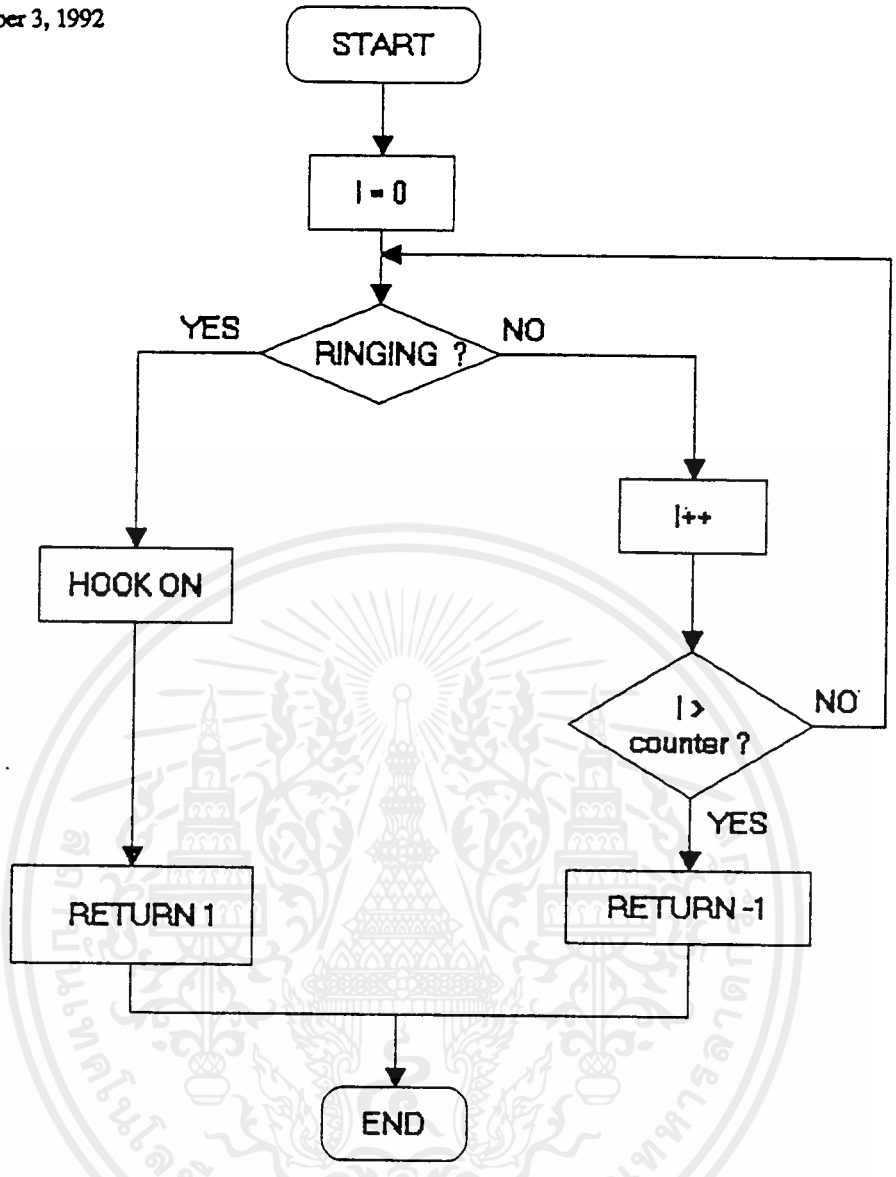
- ถ้ามีการกดจะทำการส่งค่า Key นั้นกลับไป
- ถ้าไม่มีจะทำการเช็คว่ามีสัญญาณกระดิ่งหรือไม่
  - ถ้ามีจะทำการส่งค่า 0 กลับไป เพื่อบอกว่ามีคนโทรเข้ามา
  - ถ้าไม่มีจะนับเวลาไปจนครบค่าที่ตั้งไว้ เมื่อค่านั้นครบจะส่ง -1 กลับไปเพื่อบอกว่าให้ทำการโทรออกได้

ดัง flowchart

TopChart

Tuesday, November 3, 1992

6:20 PM



```
/*-------------------------------------------------------------*
|  Function     : get_id                                      |
|  Usage        : int get_id(unsigned char input[],int length); |
|  Description  : Call getdtmf for one character and concat   |
|                 to construct user id of length "length"     |
|  Return       : -1 for id error ( hook off )                |
|                 1 and id otherwise                          |
*-------------------------------------------------------------*/

int get_id(unsigned char input[],int length)
{
   int i;
   char temp;

   for(i = 0;i<length-1; i++)  {
      if ((temp = getdtmf()) != -1) {
         input[i] = temp;
      }
      else
         return -1;
   }
   input[i] = NULL;
   return 1;
}
```

ฟังก์ชันนี้ทำหน้าที่วนลูปจนครบขนาด ID ที่กำหนดโดยเรียกฟังก์ชันที่รับค่าตัวอักษร 0 - 9 ซึ่งเป็นค่าที่ได้จากฟังก์ชัน getdtmf(); นำมาต่อกันเป็น ตัวอักษรหมายเลขประจำตัวของผู้ใช้บริการ โดยหมายเลขประจำตัวตัวสุดท้าย จะให้มีค่าเป็น '\0' ซึ่งเป็นการสิ้นสุดข้อความในภาษาซี ถ้าการรับค่า DTMF ไม่สำเร็จจะทำการส่งค่า -1 ไป

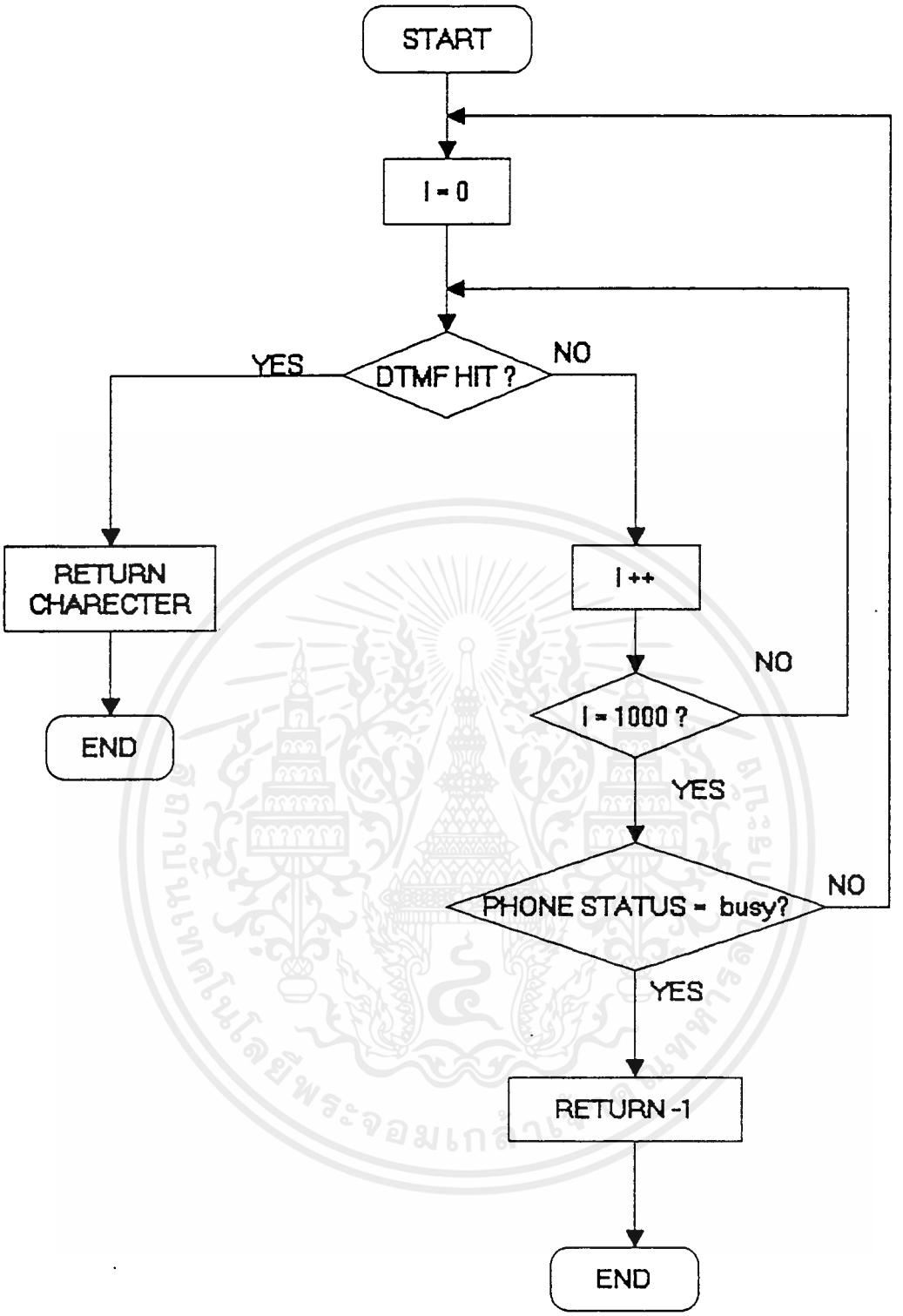
```
/*-------------------------------------------------------------*
|  Function     :  getdtmf                                    |
|  Usage        :  unsigned char getdtmf();                   |
|  Description  :  get one character of DTMF code of one press|
|  Return       :  Character of DTMF code                     |
|                  -1  if error                               |
*-------------------------------------------------------------*/
char getdtmf()
{
   unsigned char inword;
   int i;
   int temp;

   i = 0;
   while( (inp(CONTROL_ADDR) & DTMF ) != DTMF)  {
      i++;
      delay(20);
      if ( i == 1000 ) {              /* if = 1000; check status */
         if ((temp = phone_status_in()) == 4 )
            i = 0;              /* if 0, line has not been hook off */
                                /* wait on for dtmf hit */
         else
            if( temp != 4 )
               return -1;
      }
   }

   inword = inp(CONTROL_ADDR);
   inword = ((inword & 0x3e) >> 2) + '0';
```

```
    switch (inword) {
       case ':' : inword = '0';
                  break;
       case ';' : inword = '*';
                  break;
       case '<' : inword = '#';
                  break;
    }

    while(((inp(CONTROL_ADDR)) & DTMF) == DTMF) {}
    return(inword);
}
```

การทำงานของฟังก์ชันนี้คือ จะวนลูปรับค่าอินพอร์ทจากพอร์ท 3B8H เข้ามา AND กับค่าคงที่ DTMF (0x02) ซึ่งหากมีการกดปุ่มหมายเลขโทรศัพท์บิทนี้จะมีค่าเป็น 1 ไปจนกว่าจะปล่อย ซึ่งจะทำให้บิทที่ 2 นี้มีค่าเป็น 0 ดังนั้น การนำสัญญาณมาเปลี่ยนเป็นหมายเลขประจำตัวของผู้ใช้บริการจะต้อง ตรวจการกดปุ่ม คือทำการวนลูปจนกว่าบิทที่ 2 จะมีค่าเป็น 1 และเมื่อมีค่าเป็น 1 แสดงว่ามีการกดปุ่มหมายเลขโทรศัพท์ ค่าที่ได้จากบิทที่ 2 - 5 จะเป็นค่าแทนปุ่มที่กด และจะต้องมีการตรวจสอบการปล่อยปุ่มที่กด ทำได้โดยการอินพอร์ทค่าจากพอร์ทเดียวกันนี้ และตรวจดูว่าค่าเป็น 0 หรือยัง หากยังก็วนลูปรับค่าเข้ามาเรื่อย ๆ จนกว่าจะมีค่าเป็น 1 จึงสิ้นสุดการทำงาน และส่งค่าหมายเลขกลับ

ถ้าไม่มีการกดคีย์ครบกำหนดเวลา จะทำการตรวจสถานะของสายโทรศัพท์ว่าเป็นอะไร ถ้าเป็น 4 แสดงว่ายังไม่วางหู ก็ทำการรอการกดต่อไป ถ้าไม่ใช่ส่งค่า -1

ดัง flowchart



ตรวจรับค่าจากปุ่มโทรศัพท์

```
/*-------------------------------------------------------------*
|  Function    :   Dtmf_hit                                   |
|  Usage       :   int dtmf_hit();                            |
|  Description :   determine if there is a dtmf-hit           |
|  Return      :   0  :  there is not dtmf-hit                |
|                  1  :  there is dtmf-hit                    |
*-------------------------------------------------------------*/
int dtmf_hit()
{

   if( ( inp(CONTROL_ADDR) & DTMF ) == DTMF) {
      while( ( inp(CONTROL_ADDR) & DTMF ) == DTMF) {}
      return 1;
      }
   else
      return 0;
}
```

เป็นฟังก์ชันที่ใช้ตรวจเช็คเมื่อเวลาทำการอัดเสียงว่าให้หยุดเมื่อใด โดยการตรวจว่ามีการกดแป้นโทรศัพท์หรือไม่ ถ้ามีให้เป็น 1 ถ้าไม่มีให้เป็น 0

```
/*-------------------------------------------------------------*
|   Function:   Get_letter                                     |
|   Usage:   unsigned char get_letter();                       |
|   Description:   get one letter from telphone key            |
|   Return:   letter on telphone key if correct                |
|    0  otherwise                                              |
|    -1 for hook off                                           |
*-------------------------------------------------------------*/

unsigned char get_letter()
{
   unsigned char temp1,temp;
   int i;

   temp1 = 'A';

   for(i=0;i<3;i++) {
      temp = getdtmf();

      switch (temp) {
         case '1' : if(i==0)
                       return 0;
                    else
                       return temp1;
         case '8' :
         case '9' : temp1 = 'A' + (temp-'0'-2)*3 +1 + i;
                    break;
         case '7' : temp1 = 'A' + (temp-'0'-2)*3 + i;
                    if ( i >= 1 )
                       temp1++;
```

```
                    break;
            case ';' :
            case ':' :
            case '>' : return 0;

            default  : temp1 = 'A'+ (temp-'0')*3 -6 + i;
        }
    }
    return temp1;
}
```

ฟังก์ชันนี้เป็นการรับค่า DTMF มาแล้วแปลเป็นตัวอักษรซึ่งฟังก์ชันนี้จะรับทีละตัวอักษรโดยการแบ่งตัวอักษรจะแบ่งโดยกดหมายเลข 1 ซึ่งจะเป็นดังนี้

A	กดแป้นหมายเลข 2 แล้วกดหมายเลข 1
B	กดแป้นหมายเลข 2 กดหมายเลข 2 แล้วกดหมายเลข 1
C	กดแป้นหมายเลข 2 กดหมายเลข 2 กดหมายเลข 2 แล้วกดหมายเลข 1
D	กดแป้นหมายเลข 3 แล้วกดหมายเลข 1
E	กดแป้นหมายเลข 3 กดหมายเลข 3 แล้วกดหมายเลข 1
F	กดแป้นหมายเลข 3 กดหมายเลข 3 กดหมายเลข 3 แล้วกดหมายเลข 1
G	กดแป้นหมายเลข 4 แล้วกดหมายเลข 1
H	กดแป้นหมายเลข 4 กดหมายเลข 4 แล้วกดหมายเลข 1
I	กดแป้นหมายเลข 4 กดหมายเลข 4 กดหมายเลข 2 แล้วกดหมายเลข 1
J	กดแป้นหมายเลข 5 แล้วกดหมายเลข 1
K	กดแป้นหมายเลข 5 กดหมายเลข 5 แล้วกดหมายเลข 1
L	กดแป้นหมายเลข 5 กดหมายเลข 5 กดหมายเลข 5 แล้วกดหมายเลข 1
M	กดแป้นหมายเลข 6 แล้วกดหมายเลข 1
N	กดแป้นหมายเลข 6 กดหมายเลข 6 แล้วกดหมายเลข 1
O	กดแป้นหมายเลข 6 กดหมายเลข 6 กดหมายเลข 6 แล้วกดหมายเลข 1

P กดแป้นหมายเลข 7 แล้วกดหมายเลข 1

R กดแป้นหมายเลข 7 กดหมายเลข 7 แล้วกดหมายเลข 1

S กดแป้นหมายเลข 7 กดหมายเลข 7 กดหมายเลข 7 แล้วกดหมายเลข 1

T กดแป้นหมายเลข 8 แล้วกดหมายเลข 1

U กดแป้นหมายเลข 8 กดหมายเลข 8 แล้วกดหมายเลข 1

V กดแป้นหมายเลข 8 กดหมายเลข 8 กดหมายเลข 8 แล้วกดหมายเลข 1

W กดแป้นหมายเลข 9 แล้วกดหมายเลข 1

X กดแป้นหมายเลข 9 กดหมายเลข 9 แล้วกดหมายเลข 1

Y กดแป้นหมายเลข 9 กดหมายเลข 9 กดหมายเลข 9 แล้วกดหมายเลข 1

```
/*-------------------------------------------------------------*
|  Function     :  hookoff                                    |
|  Usage        :  void hookoff();                            |
|  Description  :  Hook off the telephone line                |
|  Return       :  None                                       |
*-------------------------------------------------------------*/
void hookoff()
{
   inp(OUTPORT);
}
```

เป็นฟังก์ชันที่ทำหน้าที่วางหูโทรศัพท์ โดยทำการอินพอร์ทจากพอร์ท 3B9H ซึ่งสัญญาณจากการอินพอร์ท จะไปสั่งให้รีเลย์เปิดวงจร และเปรียบเสมือนการวางหูโทรศัพท์

```
/*-------------------------------------------------------------*
|   Function    :   Hook_on                                   |
|   Usage       :   void hook_on();                           |
|   Description :   Hook the telphone on                      |
|   Return      :   None                                      |
*-------------------------------------------------------------*/
void hook_on()
{
   inp(HOOK_ON);
}
```

ฟังก์ชัน HOOK_ON() เป็นฟังก์ชันที่ทำหน้าที่ เลียนแบบการยกหูโทรศัพท์ ซึ่งจำเป็นในการใช้งานระบบ สำหรับการโอนสาย เมื่อมีการเรียกใช้บริการโอนสายอัตโนมัติ

```
/*-------------------------------------------------------------*
|   Fundtion    :   Flash                                     |
|   Usage       :   void flash(int time);                     |
|   Description :   Flash delay time msec                     |
|   Return      :   None                                      |
*-------------------------------------------------------------*/
void flash(int time)
{
   hook_off();
   delay(time);
   hook_on();
}
```

ฟังก์ชันที่ทำหน้าที่กดปุ่มที่ใช้วางหู 1 ครั้งโดยใช้ระยะเวลาในการกดลงไปตามค่าที่ฟังก์ชันรับมาในที่นี้คือ time

```
/*-------------------------------------------------------------*
|   Function    :   Phone_status_in                           |
|   Usage       :   int phone_status_in();                    |
|   Description :   check for status of telelphone line       |
|   Return      :   1  for ring back tone                     |
|                   2  for line busy                          |
|                   3  for line dial tone                     |
|                   0  for receiver hook on                   |
*-------------------------------------------------------------*/
int phone_status_in()
{
   unsigned int a;
   int i,no,c;
   int temp;

   for(i=0;i<3;i++) {
      no = 0;
      c = 0;
      a = inp(CONTROL_ADDR);

      while((a & PHONE_STATUS_IN) == PHONE_STATUS_IN) {
         a = inp(CONTROL_ADDR);
         delay(15);
         c++;
         if(c > 60) {        /* dial tone */
            temp = 3;
            break;
         }
      }
      while((a & PHONE_STATUS_IN) != PHONE_STATUS_IN) {
```

```
        if (no > 250) {  /* detect long zero */
           break;
        }
        no++ ;
        a = inp(CONTROL_ADDR);
        delay(15);
    }
    if(no > 250 ) {
       temp = 0;
       break;
    }
    else
       if ( no > 150) {
          temp = 1;
          break;
       }
       else
          if(c > 60) {
            temp = 3;
            break;
          }
          else
            temp = 2;
      }
  return temp;
}
```

ฟังก์ชัน PHONE_STATUS_IN() เป็นฟังก์ชันที่ ตรวจสอบสถานะของการโอนสาย โดยจะตรวจสอบสัญญาณในสายโทรศัพท์ ว่ามีสถานะเป็นอย่างไร สายว่างหรือไม่ หรือสายไม่ว่างเป็นต้น หลักการทำงาน คือ จะใช้ลักษณะของสัญญาณในสายโทรศัพท์ ซึ่ง

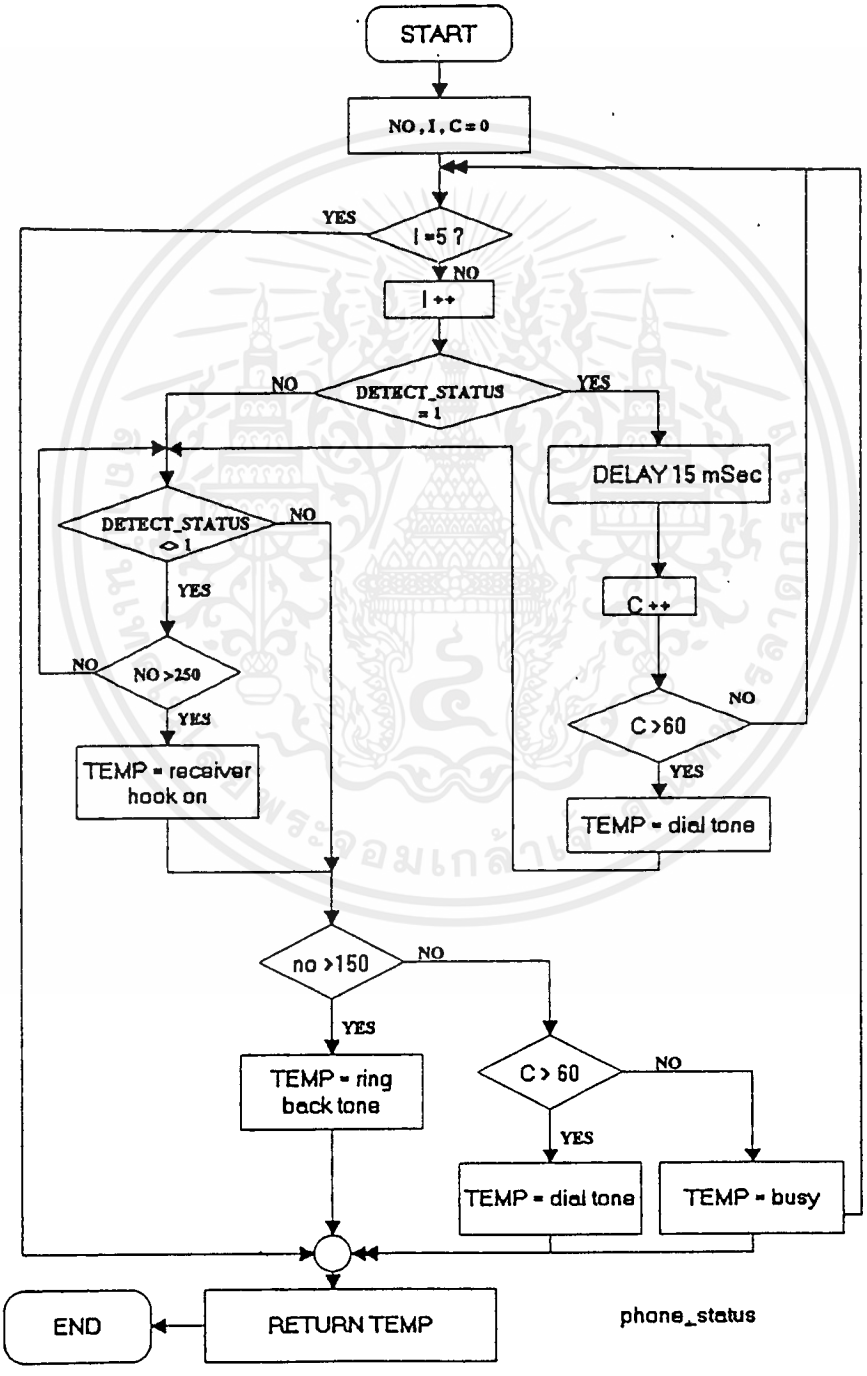
แตกต่างกันไปตามแต่สถานะของสาย เมื่อตรวจสอบได้จะส่งค่ากลับดังนี้คือ

1 หมายถึง สัญญาณ RING BACK TONE

2 หมายถึง สัญญาณ BUSY TONE

3 หมายถึง สัญญาณ READY

ดัง flowchart



```
/*-------------------------------------------------------------*
|   Function    :   Phone_status_out                          |
|   Usage       :   int phone_status_out();                   |
|   Description :   check for status of telelphone line       |
|   Return      :   1  for ring back tone                     |
|                   2  for line busy                          |
|                   3  for line ready                         |
|                   0  for receiver hook on                   |
*-------------------------------------------------------------*/
int phone_status_out()
{
   unsigned int a;
   int i,no,c;
   int temp;

   temp = 4;

   for(i=0;i<3;i++) {
      delay(200);
      no = 0;
      c = 0;

      a = inp(CONTROL_ADDR);

      while((a & PHONE_STATUS_OUT) == PHONE_STATUS_OUT) {
         a = inp(CONTROL_ADDR);
         delay(15);
         c++;
         if(c > 60) {
            temp = 3;
```

```
            break;
        }
    }
    while((a & PHONE_STATUS_OUT) != PHONE_STATUS_OUT) {
        if (no > 300) {
            break;
        }
        no++ ;
        a = inp(CONTROL_ADDR);
        delay(15);
    }
    if(no > 300 ) {
        temp = 0;
        break;
    }
    else
        if ( no > 160 ) {
            temp = 1;
            break;
        }
        else
            if(c > 60) {
                temp = 3;
                break;
            }
            else
                if(no < 10){
                    temp = temp;
                    i--;
                }
```

```
            else
                temp = 2;
    }
    return temp;
}
```

ฟังก์ชัน PHONE_STATUS_OUT() เป็นฟังก์ชันที่ ตรวจสอบสถานะของโทรออก โดยจะตรวจสอบสัญญาณในสายโทรศัพท์ ว่ามีสถานะเป็นอย่างไร สายว่างหรือไม่ หรือสายไม่ว่างเป็นต้น หลักการทำงาน คือ จะใช้ลักษณะของสัญญาณในสายโทรศัพท์ ซึ่งแตกต่างกันไปตามแต่สถานะของสาย เมื่อตรวจสอบได้จะส่งค่ากลับดังนี้คือ

1 หมายถึง สัญญาณ RING BACK TONE

2 หมายถึง สัญญาณ BUSY TONE

3 หมายถึง สัญญาณ READY

```
/*-------------------------------------------------------------*
|   Function    :   Tel_call                                  |
|   Usage       :   int tel_call(char *tel_no_in);            |
|   Description :   simulate telephone-call and  return status|
|                   of line                                   |
|   Return      :                                             |
*-------------------------------------------------------------*/
int tel_call(char *tel_no_in)
{
   char temp;
   int inttemp;

   delay(1000);
   while((temp = *(tel_no_in++)) != NULL) {
       inttemp = conv_to_dtmf(temp);
```

```
        outp(DTMF_OUT,inttemp);
        delay(300);
        outp(DTMF_OUT,NO_DTMF);
        delay(200);
    }
}
```

ฟังก์ชัน TEL_CALL() เป็นฟังก์ชันที่ทำหน้าที่ จำลองการหมุน (กด) โทรศัพท์เพื่อโทรออก โดยมีอินพุต เป็นสตริงของหมายเลขที่จะกด มีการเรียกใช้ฟังก์ชัน CONV_TO_DTMF() เพื่อเปลี่ยนสตริงนั้น ให้เป็นค่าที่ใช้สำหรับส่งให้ TCM 5087 เพื่อใช้สร้างสัญญาณ DTMF ต่อไป

```
/*-------------------------------------------------------------*
|   Function    :   Conv_to_dtmf                              |
|   Usage       :   int conv_to_dtmf(char no_in);             |
|   Description :   Convert character input to dtmf signal    |
|   Return      :   Dtmf signal if success                    |
|                   -1 otherwise                              |
*-------------------------------------------------------------*/

int conv_to_dtmf(char no_in)

{
   switch (no_in) {
      case '1' :  return 0XE1;
      case '2' :  return 0XE2;
```

```
        case '3' :  return 0XE4;
        case '4' :  return 0XD1;
        case '5' :  return 0XD2;
        case '6' :  return 0XD4;
        case '7' :  return 0XB1;
        case '8' :  return 0XB2;
        case '9' :  return 0XB4;
        case '*' :  return 0X71;
        case '0' :  return 0X72;
        case '#' :  return 0X74;
    }
}
```

ฟังก์ชัน CONV_TO_DTMF เป็นฟังก์ชันที่ทำหน้าที่ เปลี่ยนค่าอักษรที่แทนแป้นกดของโทรศัพท์แต่ละแป้น ซึ่งจะรีเทิร์นค่า ตามที่ TCM 5087 ต้องการ ในการสร้างสัญญาณ DTMF

ค่าของแป้นกดแต่ละแป้นมีค่าในเลขฐานสิบหก ดังต่อไปนี้

'1' มีค่า 0XE1
'2' มีค่า 0XE2
'3' มีค่า 0XE4
'4' มีค่า 0XD1
'5' มีค่า 0XD2
'6' มีค่า 0XD4
'7' มีค่า 0XB1
'8' มีค่า 0XB2
'9' มีค่า 0XB4
'*' มีค่า 0X71
'0' มีค่า 0X72
'#' มีค่า 0X74

```
int is_telcard_exist()
{
   int i;

   for (i=0;i<100;i++) {
      if(((inp(CONTROL_ADDR)) & DETECT) == DETECT)
 return 0;

   }
   return 1;
}
```

เป็นฟังก์ชันในการตรวจสอบว่ามีการ์ดอยู่ในเครื่องหรือไม่

```
/*-------------------------------------------------------------*
|   Function :   Enquiry                                       |
|   Usage:   int enquiry(char *tel_no);                        |
|   Description :   direct invert dialing                      |
|   Return:   1 for busy                                       |
|    2 for  receiver hook on                                   |
|    3 for  no receiver hook on                                |
*-------------------------------------------------------------*/
int enquiry(char *tel_no)
{
   int  status,j,i,no;
   j = 0;
   status = 0;
   no = 0;
```

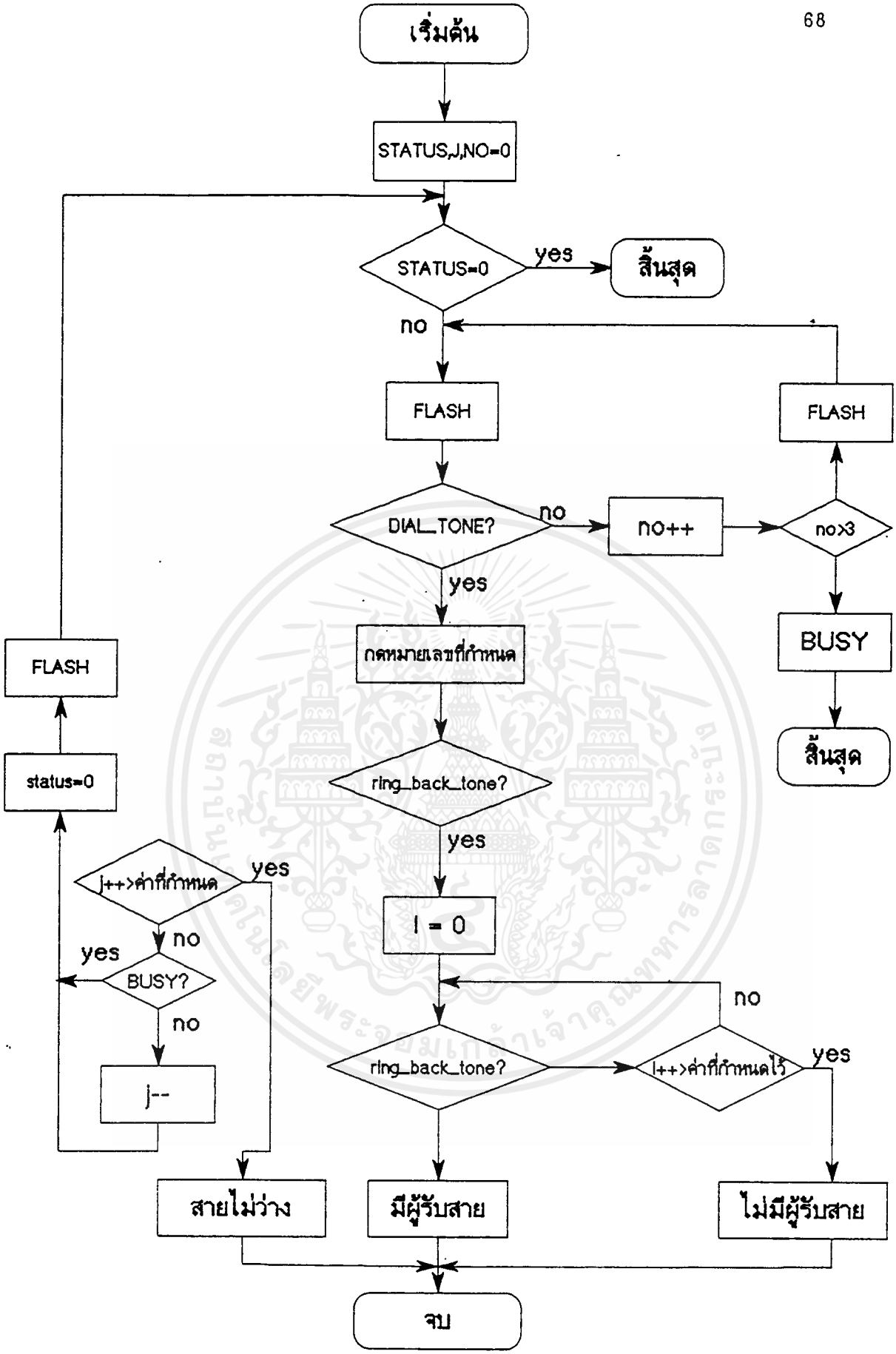
```
while(status == 0){
   flash(500);
   delay(700);

   while(((phone_status_in()) != 3) && (no < 3)) {
      flash(500);
      delay(700);
      flash(500);
      no++;
   }
   if(no >= 3)
      return(BUSY);

   tel_call(tel_no);
   if((phone_status_in()) == 1) {/* ring back tone */
      i = 0;

      while((phone_status_in()) == 1){
         if(i++ > ring_no_receiver){
            return RECEIVER_NOT_HOOK_ON;
         }
      }

      return RECEIVER_HOOK_ON;       /* receiver hook on */
   }
   else if(j++ < retry_enquiry ) {
      if((phone_status_in()) != 2)
         j--;
      status = 0; /*  retry enquiry */
      flash(500);
```

```
            delay(500);
        }
        else {
            status = BUSY;  /* busy and exit */
        }
    }
    return(status);
}
```

เป็นฟังกชันในการโอนสาย โดยรับเบอร์มาแล้วทำการใช้ 5087เป็นตัวสร้างสัญญาณ ให้เหมือนกับว่าคนเป็นผู้กด ค่าที่ส่งกลับได้แก่

- 1 ถ้าสายไม่ว่าง
- 2 ถ้ามีผู้รับสาย
- 3 ถ้าไม่มีผู้รับสาย

ดัง FLOWCHART



โอนสายโทรศัพท์

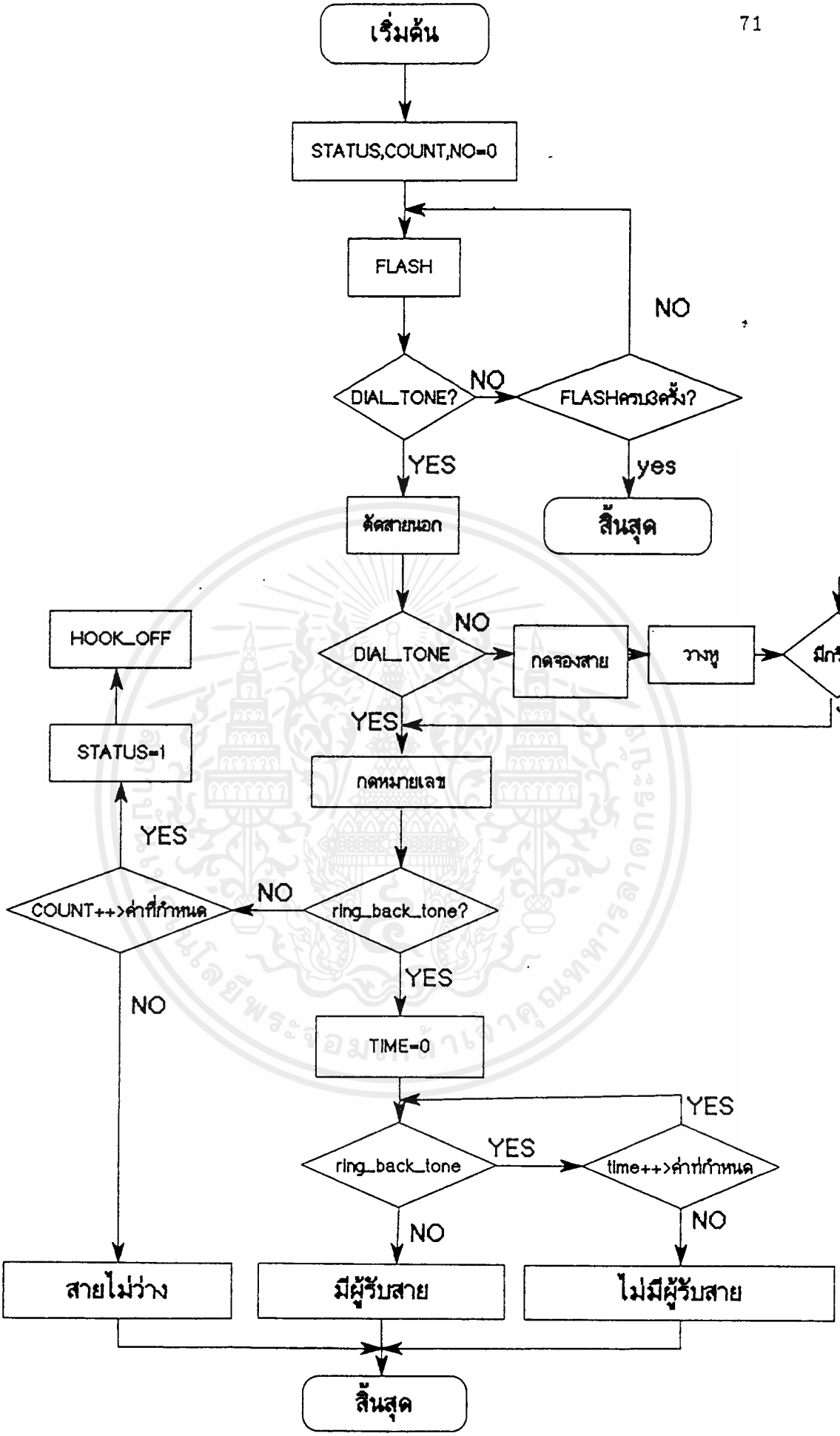
```
/*-------------------------------------------------------------*
|   Function :   telphone out                                  |
|   Usage:   int tel_out(char *tel_no);                        |
|   Description :   telphone out                               |
|   Return:   0 for  no ringing                                |
|             2 for  receiver hook on                          |
|             1 for  busy                                      |
|             3 for  no receiver hook on                       |
*-------------------------------------------------------------*/
int tel_out(char *tel_no)
{
   int status,time,count,no;
   status = 1;
   count = 0;
   no = 0;

   while(status == 1) {
      flash(500);
      delay(100);
      while((phone_status_out()) != 3){
         if(no >= 3 ){
            return 0 ;/* can not flash */
         }
         flash(200);
         delay(10);
         no++;
      }
      tel_call("9");/* extend '9' */
      delay(50);
      no = 0;
```

```
    if((phone_status_out()) == 2){
       tel_call("6");/*  booking line */
       hook_off();
       detect();
    }
    tel_call(tel_no);
    if((phone_status_out()) == 1){/* ring back tone */
       time = 0;
       while((phone_status_out()) == 1){
          if(time++ > ring_no_receiver)
             return 3; /* no receiver hook_on */
       }
       return 2; /* receiver hook on */
    }
    else if(count++ < retry_enquiry) {
       status = 1;
       hook_off();/* repeat entirely */
       delay(2000);
    }
    else {
       return 1;                          /* busy and exit */
    }
  }
}
```

เป็นฟังก์ชันสำหรับใช้โทรออกนอก PABX ซึ่งจะทำการกด 9 เพื่อขอสายนอก และทำการโทรออก ค่าที่ได้จะทำการเช็คสถานะ และส่งค่ากลับดังนี้

- 1 ถ้าสายไม่ว่าง
- 2 ถ้ามีผู้รับสาย
- 3 ถ้าไม่มีผู้รับสาย

ดัง FLOWCHART



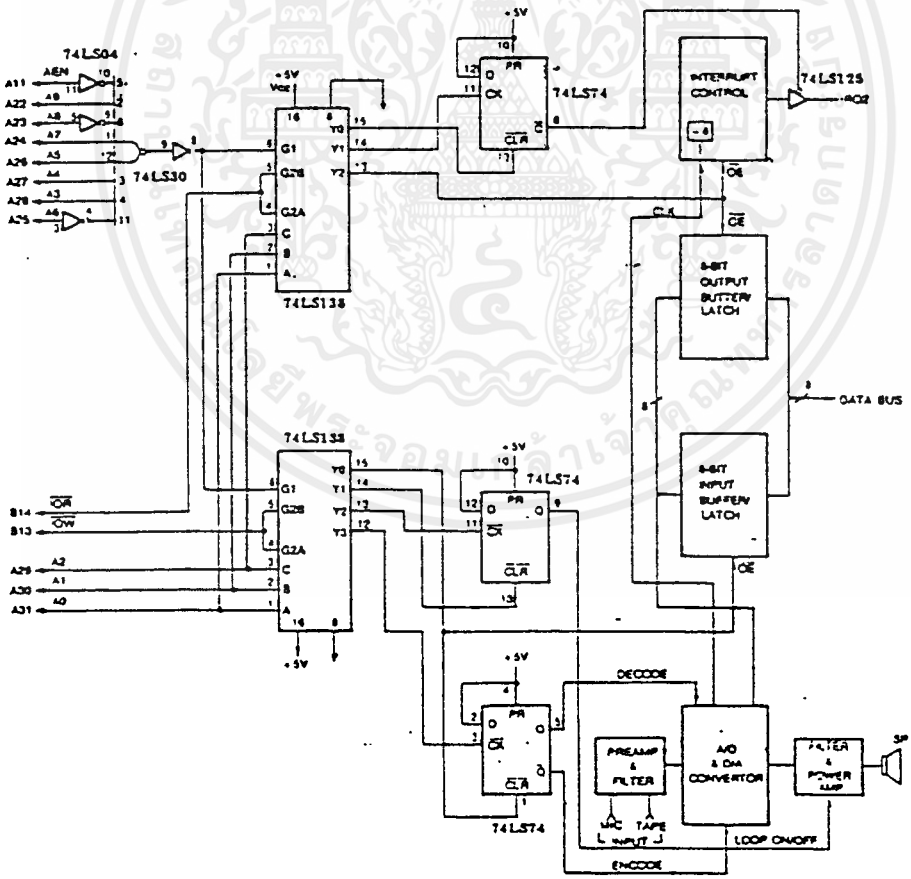
การใช้งาน VP_870 Card

การ์ด VP_870 เป็นการ์ดที่มีโหมดการทำงาน 2 อย่าง คือ

1. Analog to DIgital (A/D) สามารถเปลี่ยนสัญญาณเสียง ซึ่งเป็นสัญญาณ Analog ให้เป็นข้อมูลทาง Digital เพื่อประมวล หรือเพื่อเก็บไว้ในหน่วยเก็บข้อมูล อันได้แก่ Disk หรือ Hard disk

2. Digital to Analog (D/A) เป็นการนำข้อมูล Digital ที่เก็บไว้นั้น นำมาเปลี่ยนเป็นสัญญาณเสียง ซึ่งเป็น Analog เพื่อนำมาประยุกต์ใช้ในงานต่างๆ ต่อไป

ส่วนประกอบของการ์ด VP_870 มีดังรูปต่อไปนี้



การทำงานของวงจร VP_870

สัญญาณ A9 - A3 และ AEN จากสล็อต จะถูกนำมาถอนรหัส เพื่อเลือกช่วงแอดเดรส 2B8H - 2BFH และใช้ IC 74LS138 ซึ่งเป็นไอซีถอดรหัสจาก 3 เป็น 8 นำมาถอดรหัสสัญญาณ A2 - A0 เพื่อเลือกตำแหน่งแอดเดรสแต่ละตำแหน่ง

สัญญาณ IRO และ IOW ใช้เลือกพอร์ตอินพุต และ พอร์ตเอาท์พุต ตามลำดับ โดยที่เมื่อสั่ง INPUT (IRO active) ก็จะเลือก IC 74LS137 ตัวบนให้ทำงาน เมื่อสั่ง Output (IOW active) ก็จะเลือก IC 74LS137 ตัวล่างให้ทำงาน

สัญญาณ A2 - A0 ใช้เลือกตำแหน่งของพอร์ตที่จะใช้งานหาก A2 - A0 มีค่า 000 ก็จะทำให้ Y0 active หากมีค่า 001 ก็จะทำให้ Y1 active เช่นนี้เรื่อยไป

การทำงานของแต่ละพอร์ต มีดังต่อไปนี้

INPUT PORT :

2B8H : เมื่อสั่งให้อินพุตจากพอร์ตนี้ จะได้สัญญาณ จาก Y0 เป็นลอจิก 0 ไป ทำการเคลียร์ฟลิปฟลอปตัวบน (74ls74) ได้ Q มีค่าลอจิกเป็น 1 จึง disable สัญญาณ IRQ2 ที่ออกจากการ์ดตัวนี้

2B9H : เมื่อสั่งอินพุตจากพอร์ตนี้ จะได้สัญญาณ Y2 เป็นลอจิก 0 เป็นคล็อกให้กับฟลิบฟลอปตัวบน ทำให้ Q มีค่าลอจิกเป็น 0 ซึ่งเป็นการ enable สัญญาณ IRQ2 ให้สามารถ ออกจากการ์ดนี้ได้

2BAH : เมื่อสั่งอินพุตจากพอร์ตนี้ จะได้สัญญาณ Y2 เป็นลอจิก 0 ซึ่งทำหน้าที่ 2 อย่างคือ

1.รีเซ็ตสัญญาณ IRQ2 จากลอจิก 1 ให้เป็น ลอจิก 0 คือทำให้อินเตอร์รัพท์ หายไป

2. enable บัฟเฟอร์เอาท์พุต ในโหมดการอัดเสียง

OUTPUT PORT :

2B8H : ทำงานเมื่อสั่งเอาท์พุตที่พอร์ตนี้ มีหน้าที่ 2 อย่างคือ

1. enable บัฟเฟอร์ ในโหมดการเล่น (Play Mode)
2. เซ็ตให้ D/A ทำงานในโหมดการเล่น

2B9H : ทำงานเมื่อสั่งเอาท์พุตที่พอร์ตนี้ Y1 จาก IC 74LS138 ตัวล่างจะทำการเคลียร์ค่า Q ของฟลิปฟลอปตัวกลาง ซึ่งเป็นการเซ็ตให้ Power Amp อยู่ในโหมด SOUND OFF

2BAH : เมื่อสั่งเอาท์พุตจากพอร์ตนี้ Y1 จะเซ็ตค่า Q ของฟลิปฟลอปตัวกลางให้เป็นลอจิก 1 คือเซ็ตให้ Power Amp อยู่ในโหมด SOUND ON

2BBH : ทำงานเมื่อสั่งเอาท์พุตที่พอร์ตนี้ Y2 จะเซ็ตค่า Q ของฟลิปฟลอปตัวล่างมีค่าเป็นลอจิก 1 คือทำให้ A/D ทำงานเป็น RECORD MODE

การโปรแกรมการ์ด VP_870

ก่อนที่จะใช้งานการ์ด VP_870 จะต้องเข้าใจก่อนว่า การ์ดทำงานได้อย่างไร จะต้องเช็คค่าเริ่มต้นอะไรให้แก่การ์ด หรืออุปกรณ์บนเมนบอร์ดบ้าง

การทำงานการ์ด VP_870 จะใช้หลักการของการอินเตอร์รัพท์ โดยที่จะมีโปรแกรมหลักจัดการเกี่ยวกับการแสดงผลหน้าจอ การอ่านไฟล์ที่เก็บรหัสของเสียงเพื่อมาเปลี่ยนเป็นสัญญาณเสียงออกทางลำโพง หรือการเขียนรหัสดิจิตอลของเสียงเก็บลงในไฟล์ ในโหมดการอัด เมื่อมีการอินเตอร์รัพท์จากการ์ด จะเกิดการกระโดดไปตำแหน่งเดิมที่อินเตอร์รัพท์เวกเตอร์ชี้อยู่ ดังนั้นแทนที่จะให้กระโดดไปตำแหน่งเดิม ของ IRQ2 เราก็ต้องทำการย้ายตำแหน่งไปชี้อยู่ที่โปรแกรมที่ นำข้อมูลเสียงมาเปลี่ยนแปลงแสดงออกทางลำโพงคือ การย้ายอินเตอร์รัพท์เวกเตอร์นั่นเอง

ในส่วนโปรแกรมอินเตอร์รัพท์ จะทำงานเป็นตัวนำข้อมูลออกทางลำโพง หรือ จะนำสัญญาณเสียงมาเปลี่ยนเป็นสัญญาณดิจิตอล เพื่อเก็บไว้ ซึ่งแล้วแต่โหมดการทำงานว่าตอนนั้นต้องทำงานอะไรอยู่

สำหรับการ์ด VP_870 นี้มีโหมดการทำงานได้หลายโหมด แล้วแต่โปรแกรมจะเป็นตัวเซ็ตการทำงานดังต่อไปนี้

INIT_VP870_CARD : ทำหน้าที่เตรียมสถานะเริ่มต้นให้แก่การ์ด โดยเริ่มการ เซ็ตตำแหน่งพอร์ท ให้แก่ตัวแปรชื่อต่อไปนี้ คือ

INPUT PORT

PORT 2B8H = IRQ_DISABLE

PORT 2B9H = IRQ_ENABLE

PORT 2BAH = RESET_IRQ

OUTPUT PORT

PORT 2B8H = PLAY_DATA

PORT 2B9H = SOUND_OFF

PORT 2BAH = SOUND_ON

PORT 2BBH = RECORD_DATA

หลังจากนั้น จะเซ็ตให้ตัวแปร IRQ_ENABLE มีค่า 11111011B เพื่อเอาไว้เป็นตัว enable IRQ2 (ตำแหน่ง D2 เป็น 0) ทำการ DISABLE INTERRUPT (คำสั่ง inp(IRQ_DISABLE)) และทำให้เสียงเงียบไป (คำสั่ง outp(SOUND_OFF))

SET_MUTE : คือทำให้เสียงเงียบ (คำสั่ง outp (PLAY_DATA,0x55)คำสั่ง outp(SOUND_OFF,0))

SET_SOUND_ON : คือการทำให้เสียงดังได้ (คำสั่ง outp (SOUND_ON,0))

SET_SOUND_OFF : คือการทำให้เสียงดับได้ (คำสั่ง outp (SOUND_OFF,0))

SET_RECORD_MODE : คือเซ็ตให้ A/D ทำงานให้โหมดการอัด (คำสั่ง outp(RECORD_MODE,0)

SET_CLEAR_IRQ : คือการ reset IRQ2 หลังจากที่ถูกเซ็ต โดย counter หาร 8

PLAY_VOICE : คือการเริ่มให้ IRQ2 ทำงานได้ (คำสั่ง inp(IRQ_ENABLE) เซ็ต mode_flag ให้เป็น 0 (เป็นโหมดการเล่น)

```
STOP_PLAY       ; disable IRQ2 (คำสั่ง inp(IRQ_DISABLE))

RECORD_VOICE    ; คือการทำให้การ์ด  VP_870  ทำหน้าที่โหมดการอัด
                  (คำสั่ง outp(RECORD_MODE,0)) เซ็ต mode_flag
                  ให้เป็น 0xFF (โหมดการอัด) enable IRQ2
```

การจัดการ MEMORY

1.การเตรียม และการคืน memory

เราจะทำการจอง memory ไว้ทั้งหมด 64 Kbyte เพื่อใช้เป็น buffer ในการทำงานทั้ง โหมด record และ play โดยใช้ function farmalloc (OX10000) ของ TURBO_C สำหรับการจอง memory ส่วนนี้

ในการคืน buffer จะมีตัวแปร c_p เก็บ vector memory ที่ทำการจองไว้ในตอนต้น เมื่อเลิกใช้โปรแกรมนี้ จะต้องคืน memory เพื่อที่จะทำให้ตัวอื่นสามารถมาใช้ memory ส่วนนี้ได้ โดยใช้ function farfree(c_p) ของ TURBO_C ในการคืน

ซึ่งเขียนได้ดังนี้

```
/* ทำการจอง memory โดย vo_p เป็น vertor ที่ชี้ไปยังตัวแรก */
   if((vo_p = (unsigned char far *)farmalloc(0x10000)) == 0) {
       printf("Not enough Memory"); /* ทำการจอง memory ไม่ได้*/
       exit(1);
      }

/* ทำการเก็บค่าเริ่มต้น ไว้เพื่อใช้ในการคืน*/
   c_p = vo_p;
       :
       :
       :

/* เมื่อจะออกจาก โปรแกรม จะคืน memory */
   farfree(C_P);
```

2.การ play

จะทำการ play ข้อมูล (จาก interrupt service routine) ไปพร้อมกับทำการอ่าน ข้อมูลจาก file เข้าไปยัง memory โดย

มี pointer ชี้ไปใน memory 2 ตัว คือ

- เพื่อใช้ในการ อ่าน data จาก disk คือ b_p
- เพื่อใช้ใน interrupt service routine คือเป็นตัว vector ที่ใช้ใน sound card เพื่อทำการเปลี่ยน data ที่ vector นั้นชี้ให้ ออกมาเป็นเสียง คือตัวแปร vo_p

ตัวแปรที่เก็บค่าเกี่ยวกับ ขนาด data 2 ตัวคือ

- t_len ใช้เกี่ยบกับการอ่าน data ใน file
- p_len ใช้ใน interrupt service routine คือเป็นตัว เช็คว่าเล่น data หมด file หรือยัง

ขั้นตอนในการ play

1. ทำการเปิด file โดยมี handle เป็นตัวใช้แทน file นั้น
2. ดูพารามิเตอร์ตัวที่ 2 ถ้าหากเป็น -1 แสดงว่า ให้เล่นหมดไฟล์ แต่ถ้าเป็นค่าอื่น ที่ไม่ใช่ -1 แสดงว่าเป็นการเล่นโดยกำหนดตำแหน่งเริ่มต้นและพารามิเตอร์ที่ 3 เป็น ตำแหน่งสิ้นสุดการเล่น คือไม่เล่นทั้งไฟล์
3. หาขนาดของการเล่น เก็บไว้ที่ตัวแปร t_len , p_len
4. กำหนด segment b_p ให้เท่ากับ vo_p
5. ทำการโหลด ค่า data จาก file มา 64 Kbyte ก่อน และทำการ ลด t_len ลงไปอีก 64 Kbyte
6. จะทำการเล่น จนกระทั่งมีการกด DTMF ใดๆ หรือ หมดขนาด t_len
7. เริ่มให้มีการ intterupt ได้ คือเริ่มมีการเล่นได้ โดยในการ interrupt แต่ละครั้ง (ซึ่งในการ interrupt นั้น จะไม่เกี่ยวกับ การอ่านเขียน disk)
   6.1 เช็คค่า p_len ว่าเป็น 0 หรือยัง ถ้าถึงแล้วให้ หยุด การ interrupt เลย
   6.2 ทำการเล่นไป 1 byte และ

- ลดค่า p_len ลง 1
- ทำการ เพิ่มค่า vector vo_p ขึ้น 1

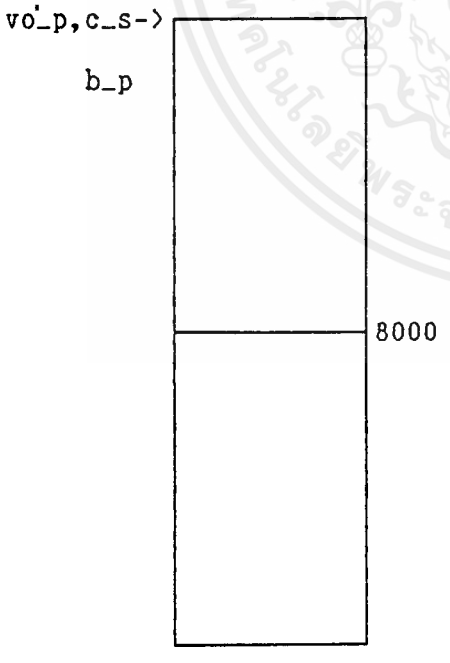
6.3 ค่า vo_p จะทำการวน คือ เมื่อถึง FFFF ก็จะกลับเป็น 0 ใหม่ ใน segment เดียวกัน

7. ถ้า data มีขนาด มากกว่า 64 Kbyte จะทำการโหลดค่าต่อมา เป็น block โดยจะทำการโหลด block ละ 32 Kbyte โดย วิธีดังนี้

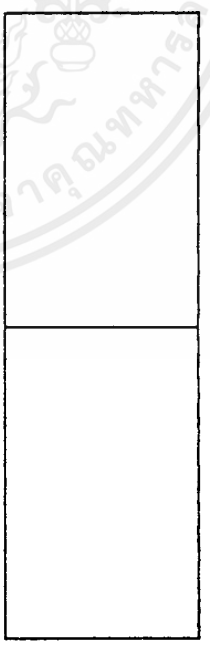
7.1 ทำการ เช็คว่า t_len หมดหรือยัง หรือมีการกด Key หรือไม่

7.2 เมื่อ vo_p มีค่า 32 Kbyte จะทำการโหลดค่า 32 Kbyte ลงไปที่ block ที่ทำการเล่นไปแล้ว (แต่ใน ช่วงการอ่าน file ลง memory จะทำการ interrupt ไปด้วย) โดยใช้ b_p เป็นตัวชี้ว่าจะอ่าน ไว้ที่ใด

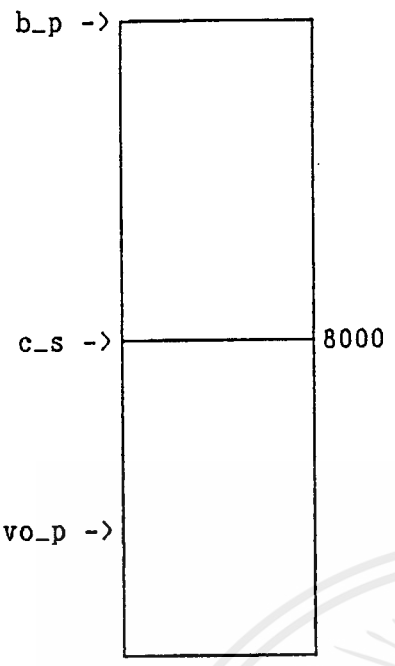
7.3 เมื่อ vo_p มีค่า 64 Kbyte ก็ทำการ โหลดค่าอีก 32 Kbyte ลงไปใน block ที่ทำการเล่น เสร็จไป โดยมี b_p เป็น vector เช่นเดิม



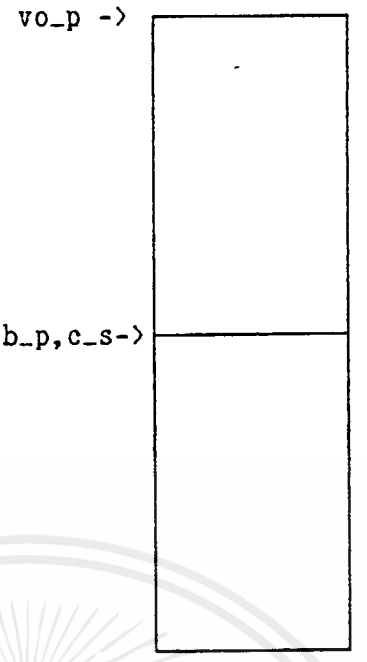
เมื่อเริ่มต้นทำการเล่น



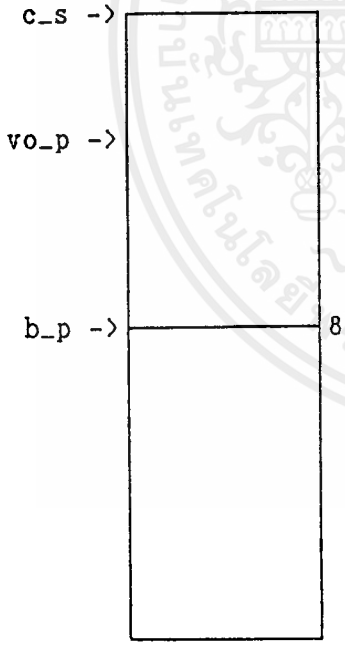
ทำการ read data จาก file ใหม่ โดยเริ่มอ่านไว้ที่ b_p



เมื่ออ่าน data จาก file เสร็จ



เมื่อ play หมด 64 Kbyte



เมื่อทำการ read จาก file เสร็จ

FUNCTION PLAY

```
int play_data(char *fname,long start,long stop)
{
   unsigned int c_s;
   unsigned char far *b_p;
   unsigned long t_len,t_a;
   int handle;
   unsigned long temp;

   if ((handle = _open(fname,O_BINARY|O_RDWR)) == -1)
      return -1;

   if (start == -1) {
      t_len = p_len = filelength(handle);
      start = 0;
      }
   else
      t_len = p_len = stop - start;

   FP_SEG (b_p) = FP_SEG(vo_p);
   FP_OFF(vo_p) = FP_OFF(b_p) = c_s =  0;
   lseek(handle,start,SEEK_SET);
   if (t_len > 0x8000)  {
      read(handle,b_p,0x8000);
      t_len -= 0x8000;
      b_p += 0x8000;
      }
   t_a = (t_len > (unsigned long) 0x8000) ? 0x8000 : t_len;
   read(handle,b_p,t_a);
   t_len -= t_a;
```

```
    outp(PLAY_DATA,0x55);
    outp(SOUND_ON,0);
    mode_flag = 0;
    rate_flag = 0;
    while(p_len  && !stop_play())   {
       if(t_len && (c_s != (FP_OFF(vo_p) & 0x8000))){
          t_a = (t_len > 0x8000) ? 0x8000 : t_len;
          FP_OFF(b_p) = c_s;
          read(handle,b_p,t_a);
          t_len -= t_a;
          c_s ^= 0x8000;
       }
    }
    rate_flag = -1;
    outp(PLAY_DATA,0x55);
    close(handle);
    return 0;
}
```

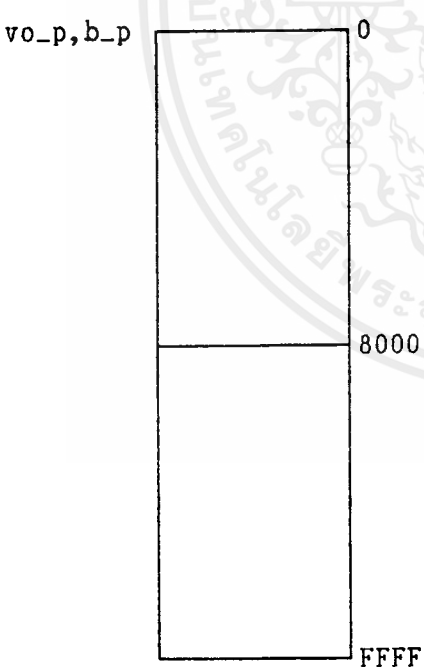
3.record

ลักษณะจะคล้ายกับการ play คือจะทำการเขียน เป็น block ซึ่งจะมีตัวแปรดังนี้

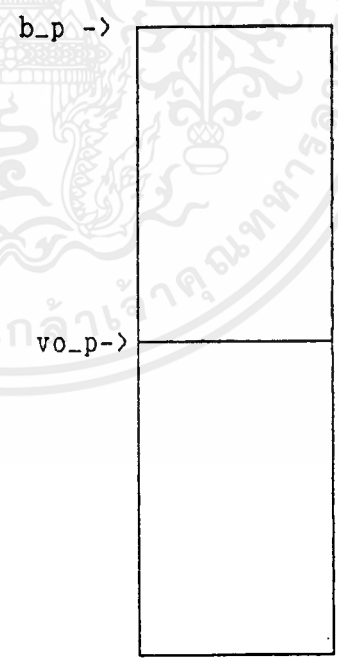
- ใช้ในการรับข้อมูลจาก sound card มาเก็บไปไว้ใน vector vo_p
- ใช้ในการอ่านข้อมูลจาก memory ไปยัง file โดยเริ่มจาก vector b_p และใช้ในการเก็บขนาดของ data
- r_len ใช้ในการนับ data ว่าเกิน 32 Kbyte หรือยัง
- len ใช้ในการเก็บขนาดของ data ที่แท้จริง

ขั้นตอนในการ record

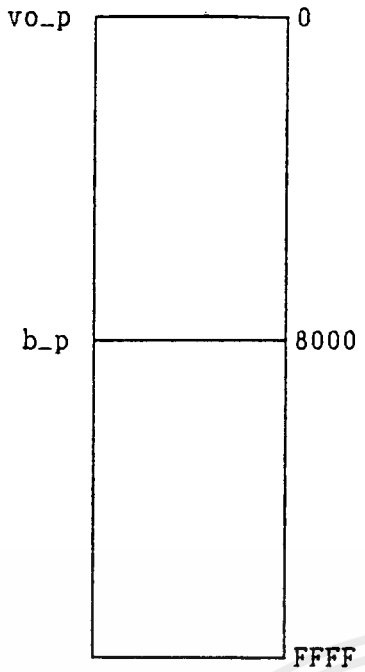
1. ทำการสร้าง file ใหม่
2. จะทำการ record จนกระทั่งมีการกด key หรือ DTMF
   2.1 เมื่อมีการ record จนกระทั่ง มากกว่า 32 Kbyte จะ ทำตามขั้นตอนดังนี้ (แต่ยังมีการอัดจาก sound card ต่อไป)
   2.2 ทำการกำหนดค่า b_p โดยครั้งแรกจะเป็น 0 และครั้งต่อ ไปจะเป็น 0x8000 สลับไปเรื่อยๆ
   2.3 เช็คว่าทำการเขียน data ได้ครบหรือไม่ ถ้าไม่ครบ แสดงว่าเกิด error ออกจากโปรแกรมได้เลย
   2.4 ทำการลด r_len ลง 32 K
   2.5 เพิ่มขนาด len อีก 32 K
3. ถ้ามีการกดคีย์ จะทำการ write ข้อมูลจาก memory ที่เหลือลง ไปยัง file
4. ทำการปิด file



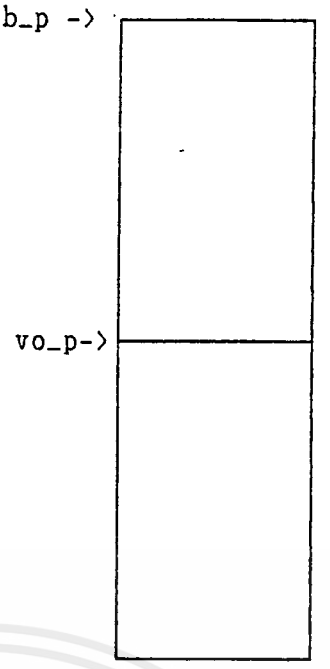
เมื่อเริ่มทำการ record



เมื่ออัด เกิน 32 Kbyte จะเริ่มเขียน ลง file



เมื่ออัดเกิน 64 K
จะเขียนลง file อีกครั้ง
r_len > 64 K



เมื่ออัดเกิน 96 K
เขียนลง file

FUNCTION RECORD

```
int  record_data(char *fname)
{
   unsigned long len;
   unsigned char far*b_p;
   unsigned int bytes;
   int handle;

   if((handle = _creat(fname,FA_ARCH)) ==  -1)
      return -1;

   FP_SEG(b_p) = FP_SEG(vo_p);
   len = 0L;
   FP_OFF(b_p) = FP_OFF(vo_p) = r_len = 0;
```

```
outp (PLAY_DATA,0xFF);
outp(SOUND_ON,0);
outp (RECORD_MODE,0);
mode_flag = -1;
rate_flag = 0;

while ( !stop_record()) {
   if(r_len > 0x8000)  {
      FP_OFF(b_p) = (FP_OFF(vo_p) & 0x8000) ^0x8000;
      bytes = write(handle,b_p,0x8000);
      if(bytes < 0x8000)
         return -2;

      r_len -= 0x8000;
      len   += 0x8000;
   }
}
rate_flag = -1;
outp(PLAY_DATA,0x55);
outp(SOUND_OFF,0);
if(r_len > 0x8000)  {
   FP_OFF(b_p) = (FP_OFF(vo_p) & 0x8000) ^0x8000;
   bytes = write(handle,b_p,0x8000);
   if(bytes < 0x8000)
      return -2;

   r_len -= 0x8000;
}
FP_OFF(b_p) = FP_OFF(vo_p) & 0x8000 ;
bytes = write(handle,b_p,r_len);
```

```
    if(bytes < r_len)
        return -2;

    outp(PLAY_DATA,0x55);
    _close(handle);

    return 1;
}
```

การสร้างโปรแกรม EVS.EXE

โปรแกรม EVS.EXE ประกอบขึ้นด้วย โปรแกรมที่แยกเป็นไฟล์ ต่างหากกัน 13 ไฟล์ ซึ่งแต่ละไฟล์มีฟังก์ชัน หน้าที่การทำงานแตกต่างกัน ไฟล์ทั้ง 13 คือ

MAIN.C

LLMODULE.C

SCR.C

WINDOW.C

TELPHONE.C

ENTRY.C

KRAM.C

RETURN.C

TELAD.C

USER.C

MAIL.C

VCMODULE.C

MESAD.C

1. MAIN.C

เป็นไฟล์ที่ประกอบด้วยฟังก์ชันที่ทำหน้าที่ initial การทำงานของระบบ ได้แก่

- initscr();

  ทำหน้าที่ initial จอภาพตาม card ที่ใช้อยู่

- initial telphone card

  ได้แก่ การตรวจสอบว่า telphone card ว่าต่ออยู่หรือไม่

  การ hook_off โทรศัพท์

- initvoice();

  จองหน่วยความจำบัฟเฟอร์ สำหรับการเล่น และอัดเสียง

- initial list

  ได้แก่ delete file list

  message list

  call out list

- initial database

  ตรวจสอบว่าไฟล์ database มีอยู่หรือไม่ ถ้าไม่มี จะสร้างขึ้นใหม่

- initial call out file

  ตรวจสอบว่าไฟล์ call out มีอยู่หรือไม่ ถ้ามี จะอ่านข้อมูลเพื่อสร้างเป็นลิสต์ ใช้ในการ call out แต่ถ้าไม่มี จะสร้างขึ้นใหม่

2. LLMODULE.C

เป็นไฟล์ที่ทำหน้าที่เกี่ยวกับ Link List ทั้งหมด อันได้แก่ การสร้างลิงค์สำหรับเก็บข้อมูล ซึ่งสามารถเปลี่ยนแปลงโครงสร้างของข้อมูลแต่ละ Node ได้ โดยการเปลี่ยน Structure ในไฟล์ LLDEF.H หรือจะเป็นฟังก์ชันเกี่ยวกับการเพิ่ม List การลบ List การเลื่อน current pointer ที่ใช้สำหรับชี้ตำแหน่งของ Node ปัจจุบัน ให้เปลี่ยนตำแหน่งเป็นตำแหน่งก่อนหน้า หรือ ตำแหน่งถัดไปได้

3. SCR.C

เป็นไฟล์ที่ทำหน้าที่เกี่ยวกับ การจัดการหน้าจอ (Screen) ทั้งหมดในโปรแกรมนี้ ลักษณะของการจัดการหน้าจอทั้งหมด ได้แก่ การตรวจสอบ mode การแสดงผลว่าเป็นโหมดอะไร เพื่อสามารถกำหนดตำแหน่งของหน่วยความจำจอภาพ (Video Ram) และนำไปใช้ในการแสดงผลตัวอักษร รวมทั้งแอททริบิวต์ ซึ่งจะทำให้ประสิทธิภาพการแสดงผล เร็วยิ่งขึ้น กว่าการใช้ฟังก์ชันมาตรฐานของภาษาซี

ฟังก์ชันที่มีในไฟล์ SCR.C ได้แก่ ฟังก์ชันแสดงบล็อก ฟังก์ชันแสดงกรอบสี่เหลี่ยม ฟังก์ชันแสดงเงาของกรอบสี่เหลี่ยม ฟังก์ชันเติมพื้นที่ที่กำหนดด้วยอักษรและแอททริบิวท์ที่กำหนด และอื่น ๆ

4. WINDOW.C

เป็นไฟล์ที่มีฟังก์ชันที่ทำงานเกี่ยวกับ การจัดการวินโดว์ อันได้แก่ การสร้างวินโดว์ การลบวินโดว์ การเลื่อนตำแหน่งของข้อความในวินโดว์ขึ้นหรือลง การแสดงผลตัวอักษรในวินโดว์ และอื่น ๆ

5. TELPHONE.C

เป็นไฟล์ที่มีฟังก์ชันที่ทำงานเกี่ยวกับ ฮาร์ดแวร์ ที่ตรวจจับสัญญาณ โทรศัพท์ รวมทั้งฟังก์ชันที่นำสัญญาณที่ได้ มาเปลี่ยนเป็นข้อมูลตัวอักษรเพื่อใช้ในการตัดสินใจเลือกฟังก์ชันการทำงานจากการกดปุ่มที่เครื่องโทรศัพท์ ฟังก์ชันเหล่านี้ได้แก่ ฟังก์ชันตรวจสอบ

สัญญาณกระดิ่ง (Ringing) ฟังก์ชันรับค่าสัญญาณ DTMF ฟังก์ชันวางหูโทรศัพท์ ฟังก์ชันสร้างหมายเลขประจำตัวของผู้ใช้บริการจากรหัส DTMF ฟังก์ชันการสร้างสัญญาณ DTMF เพื่อใช้ในการโทรออก การตรวจสอบสัญญาณในสายโทรศัพท์ เพื่อตรวจสอบสถานะว่า สายโทรศัพท์ว่างหรือไม่ ถ้าว่าง มีผู้รับหรือไม่ เป็นต้น

6. ENTRY.C

เป็นไฟล์ที่มีฟังก์ชันที่ทำงานเกี่ยวกับ การสร้าง template สำหรับรอรับการใส่ค่า เพื่อนำมาสร้างเป็น record เก็บไว้ใน database ต่อไป ประกอบด้วยการทำงาน 3 ส่วนคือ

1. ส่วนของ character field
2. ส่วนของ radio box
3. ส่วนของ button

7. KRAM.C

เป็นไฟล์ที่มีฟังก์ชันที่ทำงานเกี่ยวกับ การจัดเตรียมสถานะของ database file โดยการทำงานเป็นแบบ hashing function ที่สามารถใช้ key ของ data record ในการคำนวณ เพื่อชี้ไปยังที่อยู่ของข้อมูลได้โดยตรง ดังนั้น จะสามารถเข้าหาข้อมูลได้เร็วมาก ในกรณีที่ใช้บัฟเฟอร์ขนาดไม่ใหญ่มากนัก จะต้องมีการค้นหาในหน่วยความจำก่อน หากค้นหาพบ ก็สามารถดึงข้อมูลมาใช้ได้ทันที หากค้นหาไม่พบ ก็จะต้องมีการอ่านดิสก์ 1 ครั้ง เพื่อนำข้อมูลจากดิสก์เข้ามาเก็บในหน่วยความจำ จากนั้นจึงดึงจากหน่วยความจำไปใช้งานต่อไป จะเห็นว่า จะมีการอ่านดิสก์อย่างมากเพียง 1 ครั้งเท่านั้น จึงดีกว่าการเก็บข้อมูลแบบ sequentail มาก

8. RETURN.C

เป็นฟังก์ชันที่ทำหน้าที่ ส่งค่ากลับไปยัง ฟังก์ชันที่เรียกใช้ฟังก์ชัน data_entry(); ทุกฟังก์ชัน คือเมื่อเสร็จสิ้นการทำงานในระดับหนึ่งแล้ว ต้องการจะกลับไปยังระดับที่ทำงานอยู่ก่อน จะสามารถเลือกการกลับได้ 2 ลักษณะ คือ

ฟังก์ชัน quit(); แทนการกลับแบบ quit หรือ Ok

ฟังก์ชัน cancel(); แทนการกลับแบบ cancel

ฟังก์ชัน quit();

```
int quit()
{
   return OK;
}
```

ฟังก์ชัน cancel();

```
int cancel()
{
   return CANCEL;
}
```

การที่ต้องแยกออกมาเป็นไฟล์ต่างหาก ก็เพราะว่า ในแต่ละไฟล์ที่มีการใช้งานนั้น จะมีการเรียกใช้ฟังก์ชัน quit(); และ cancel(); อยู่เป็นประจำ ในการเรียกฟังก์ชัน data_entry(); ครั้งหนึ่ง ๆ ดังนั้น เพื่อจะได้สามารถเรียกได้โดยตรง ไม่ต้องมีการเขียนฟังก์ชันใหม่ทุกครั้ง ในแต่ละไฟล์

9. TELAD.C

การทำงานของฟังก์ชันในไฟล์ telad.c ได้แก่ ฟังก์ชันที่กำหนดค่าเวลาของการรอคอย ให้กระดิ่งดังขึ้น ซึ่งค่าเวลาดังกล่าว จะเป็นตัวกำหนดการโทรออกของระบบ เช่น กำหนดให้ระบบรอการโทรเข้ามาเป็นเวลา 10 นาที หากในช่วง 10 นาทีนี้มีการโทรเข้า ก็จะไปทำงานบริการผู้โทรเข้าเสียก่อน เมื่อบริการเสร็จสิ้น

แล้ว จะกลับมารอรับโทรศัพท์ต่อไป เป็นเวลา 10 นาที หากในช่วง 10 นาทีนี้ ยังไม่มีผู้โทรเข้ามาขอใช้บริการ ก็จะต่อโทรศัพท์ โทรออกไปหาผู้ใช้บริการ ซึ่งมีผู้ฝากข้อความสั่งให้ระบบ โทรออกไปยัง user คนนั้น โดยที่เบอร์โทรออก เป็นเบอร์ที่ user ได้ให้แก่ admin ไว้

ส่วนการทำงานในส่วนที่ 2 เป็นการกำหนด ระยะเวลาการดังของโทรศัพท์ในกรณีการโอนสายโทรศัพท์ หรือในกรณีการโทรออก เพื่อให้สามารถกำหนดได้เองว่า จะให้โทรศัพท์ดังอยู่เป็นระยะเวลานานเท่าใด จึงจะสรุปได้ว่า ไม่มีผู้รับสาย เช่น อาจจะกำหนดให้ดังอยู่ 10 ครั้ง เป็นต้น

10. USER.C

เป็นไฟล์ที่ประกอบด้วยฟังก์ชันที่ทำงานดังต่อไปนี้คือ

- การเพิ่ม user
- การ delete user
- การ edit user
- การ update user

โดยเรียกใช้ฟังก์ชัน data_entry(); และ parameter เป็น template ของ user ให้แก่ฟังก์ชัน เมื่อเสร็จสิ้นการ edit user แล้ว ก็สามารถเพิ่ม user เข้าเก็บใน database ได้ หรือต้องการนำข้อมูลของ user ออกมาแก้ไข ก็สามารถอ่านข้อมูลออกมาแก้ไข และ update เข้าไปได้ ส่วนการลบ user ก็ทำได้โดยเพียงแต่ใส่ รหัสของ user และเรียกฟังก์ชัน delete_user();

11. MAIL.C

เป็นไฟล์ที่ประกอบด้วยฟังก์ชันที่เกี่ยวกับ การทำงานเป็น ระบบ voice mail box และเป็น ระบบโอนสายอัตโนมัติ มีการทำงานคร่าว ๆ คือ เมื่อ user โทรเข้ามาหาระบบ จะได้รับการกล่าวต้อนรับ และรอให้ user กดเลือกเครื่องหมาย เพื่อเลือกใช้บริการ voice mail box หรือ กดหมายเลขต่อได้โดยตรง ในกรณีที่ไม่ใช่ user ที่สามารถใช้งานบริการอื่นได้

บริการที่ให้เลือกใช้ ก็มีการฝากข้อความไว้ใน mail box การรับฟังข้อความที่มีผู้ฝากไว้ให้ใน mail box หรือการสั่งให้ระบบโทรออกไปหา user อีกคนหนึ่งตาม user id ที่ให้ เป็นต้น

ส่วนการโอนสายอัตโนมัตินั้น ระบบจะตรวจสถานะของสายโทรศัพท์ เพื่อบอกแก่ผู้ใช้บริการว่า ปลายสายมีผู้รับสายหรือไม่ ถ้ามี ก็จะต่อให้คุยกันได้โดยตรง ถ้าไม่มีผู้รับสาย หรือสายไม่ว่าง ก็จะให้ผู้ใช้บริการเลือกว่า จะฝากข้อความไว้ให้ หรือจะยกเลิกการใช้บริการไปเลย เป็นต้น

12.VCMODULE.C

เป็นไฟล์ที่ประกอบด้วยฟังก์ชันพื้นฐาน ที่ทำงานเกี่ยวกับการเล่นเสียง การอัดเสียง และการจองหน่วยความจำ เพื่อสร้างเป็นบัฟเฟอร์สำหรับเก็บข้อมูลเสียงที่อัดได้หรือเสียงที่ต้องการเล่นออกไปให้ user ฟัง

13. MESAD.C

เป็นไฟล์ที่ประกอบด้วยฟังก์ชันที่ทำหน้าที่จัดการเกี่ยวกับ message ของ user แต่ละคน เช่น

- การ clean up message
- การสร้าง general information ให้แก่ user

การ COMPILE

แต่ละไฟล์ที่ได้กล่าวมาข้างต้น สามารถ Compile แยกกันต่างหากได้ แล้วเมื่อต้องการ Executable File ก็นำมา Link รวมกันในขั้นตอนสุดท้ายเพื่อให้ได้ไฟล์ที่สามารถทำงานได้

ใน Project นี้ได้ Compile ด้วย Compiler Turbo C Version 2.0 ของบริษัท Borland ในโหมดของ IDE (Integrated Development Environment) ซึ่งสามารถ Compile ในลักษณะของ Project File คือ สร้าง Text File ที่มีชื่อว่า EVS.PRJ ซึ่งภายในไฟล์นี้จะประกอบด้วยชื่อไฟล์ที่จะประกอบกันเป็นไฟล์ที่ต้องการอันได้แก่ไฟล์ทั้ง 13 ที่ได้กล่าวมาแล้ว ดังนี้คือ

MAIN.C
LLMODULE.C
SCR.C
WINDOW.C
TELPHONE.C
ENTRY.C
KRAM.C
RETURN.C
TELAD.C
USER.C
MAIL.C
VCMODULE.C
MESAD.C

เมื่อเลือกเมนู Project จาก เมนูของ Turbo C ให้ใส่ชื่อไฟล์ VBM.PRJ และเลือก เมนู Run หรือ กด Ctrl-F9 เพื่อสั่ง Compile และ Link ไฟล์ทั้ง 5 รวมกัน เป็นไฟล์ .EXE ชื่อ EVS.EXE เพื่อใช้งานต่อไป

ไฟล์ที่ต้องใช้ประกอบกับโปรแกรม EVS.EXE

ในโปรแกรม EVS.EXE จะต้องใช้ไฟล์เสียงที่เป็นไฟล์เกี่ยวกับการแนะนำการใช้งานระบบ อีกหลายไฟล์ รวมทั้งไฟล์ที่ใช้เก็บประวัติการใช้งานระบบด้วย ไฟล์เหล่านี้ได้แก่

1. INTRO.VOC

เป็นไฟล์ที่ใช้สำหรับกล่าวแนะนำตัวให้แก่ผู้ใช้บริการทราบ ว่าขณะนี้กำลังใช้บริการระบบรับฝากข้อความทางโทรศัพท์อัตโนมัติอยู่ และถามรหัสหรือหมายเลขประจำตัวของผู้ใช้บริการผู้นั้น

ไฟล์สำหรับใช้ในบริการ voice mail box

2. ASKID.VOC

เป็นไฟล์ที่ใช้สำหรับถามรหัสประจำตัวของ user

3. FUNCTION.VOC

เป็นไฟล์ที่ทำหน้าที่กล่าวแนะนำการทำงานของระบบ ว่าผู้ใช้บริการสามารถเลือกใช้บริการระบบ ในลักษณะใดได้บ้าง ข้อความดังกล่าวได้แก่

"ท่านสามารถเลือกใช้บริการได้โดยกดปุ่มหมายเลขดังต่อไปนี้

1. ฝากข้อความ
2. ฟังข้อความที่มีผู้ฝากไว้
3. สั่งให้โทรออก
4. ยกเลิกการใช้บริการ"

4. NOMSG.VOC

เมื่อผู้ใช้บริการเลือกใช้บริการรับฟังข้อความที่มีผู้ฝากไว้ให้ และกดปุ่มหมายเลข 1 ระบบจะตรวจสอบว่ามีผู้ฝากข้อความให้กับผู้ใช้บริการคนนี้หรือไม่ ถ้าหากมี ก็จะนำ

ไฟล์ข้อความออกมาเล่น และส่งเป็นเสียงให้กับผู้ใช้บริการคนนั้น หากไม่มีก็จะบอกกับผู้ใช้บริการว่า ไม่มีข้อความใดฝากไว้

5. STARTMSG.VOC

หากผู้ใช้บริการเลือกใช้บริการฝากข้อความให้กับบุคคลอื่น และกดปุ่มหมายเลข 2 ระบบก็จะบอกให้ผู้ใช้บริการเริ่มฝากข้อความได้ โดยใช้ไฟล์นี้

6. IDAGAIN.VOC

หากผู้ใช้บริการกดปุ่มหมายเลขที่ใช้แทนหมายเลขประจำตัวผิด ระบบก็จะแจ้งให้ผู้ใช้บริการทราบว่า ได้กดปุ่มหมายเลขประจำตัวผิดพลาด และให้กดหมายเลขประจำตัวอีกครั้ง

7. DESTID.VOC

เมื่อผู้ใช้ต้องการฝากข้อความให้กับบุคคลอื่น ระบบจะต้องแจ้งให้ผู้ใช้บริการกดปุ่มหมายเลขประจำตัวของผู้ที่ต้องการฝากข้อความไว้ให้ด้วย และหากกดปุ่มหมายเลขประจำตัวผิด ก็จะบอกให้กดปุ่มหมายเลขประจำตัวใหม่ ด้วยไฟล์ในข้อ 6

8. DELETE.VOC

เมื่อผู้ใช้ฟังข้อความที่มีผู้ฝากไว้แล้ว ระบบจะถามผู้ใช้ว่า จะลบไฟล์ที่ฟังนี้ทิ้งเลยหรือไม่ ซึ่งขึ้นกับ user ว่าจะเก็บไฟล์นี้ไว้ฟังภายหลังหรือไม่ ถ้าหากไม่เก็บ ก็จะสั่งให้ลบทิ้งได้ เพื่อเหลือเนื้อที่ใน mail box ไว้เก็บข้อความอื่นได้ต่อไป

9. OLD.VOC

จากนั้น เมื่อ user ต้องการจะฟังข้อความเก่าทั้งหมด ระบบจะถามว่า จะฟังข้อความเก่าทั้งหมดหรือไม่ ซึ่ง user ก็สามารถสั่งให้ระบบ เล่นข้อความเก่าทั้งหมดให้ฟังได้

ไฟล์สำหรับใช้ในบริการโอนสายอัตโนมัติ

10. NOHOOKON.VOC

เมื่อมีการโอนสาย จะสามารถตรวจสอบสถานะของสายโทรศัพท์ได้ว่า มีผู้รับสายหรือไม่ หรือสายว่างหรือไม่ ในกรณีไม่มีผู้รับสาย จะแจ้งให้ผู้ใช้สายว่า ไม่มีผู้รับสาย

11. BUSY.VOC

ถ้าหากสายไม่ว่าง ก็จะแจ้งให้ผู้ใช้บริการทราบว่า สายไม่ว่าง

ไฟล์สำหรับใช้ในบริการโทรออก

12. TELOUT.VOC

เมื่อผู้ใช้บริการต้องการสั่งให้โทรออก ระบบจะถามรหัสของผู้ที่ user ต้องการจะฝากข้อความให้ระบบ โทรออกไปหา ด้วยไฟล์เสียงนี้

การติดตั้งระบบ

ระบบรับฝากข้อความทางโทรศัพท์อัตโนมัตินี้ ประกอบด้วย ส่วนสำคัญ 3 ส่วนใหญ่ ๆ คือ

1. ส่วน SOUND CARD ที่ใช้สำหรับการเปลี่ยนเสียงให้เป็นข้อมูลดิจิตอลเก็บไว้ในหน่วยความจำสำรองเมื่อผู้ใช้ต้องการฝากข้อความไว้ให้กับบุคคลอื่น และนำข้อมูลที่เก็บไว้ในหน่วยความจำสำรองมาเปลี่ยนเป็นสัญญาณเสียงส่งออกมาเพื่อให้ผู้ใช้บริการฟัง เมื่อต้องการใช้บริการฟังข้อความที่มีผู้ฝากไว้ให้ การติดตั้ง ก็สามารถนำการ์ดนี้ เสียบเข้าไปในสล็อตของเครื่อง IBM PC AT หรือ XT ก็ได้ ที่การ์ดจะมีแจ็คตัวเมีย สำหรับเสียบแจ็ค 3 ช่อง คือ

1. TAPE สำหรับรับสัญญาณจากวิทยุเทป

2. MIC สำหรับรับสัญญาณจากไมโครโฟน

3. SPEAKER สำหรับต่อกับลำโพง

แต่ละช่องจะใช้สำหรับการติดต่อกับสายโทรศัพท์ดังจะได้อธิบายต่อไป

2. ส่วนตรวจสอบสัญญาณในสายโทรศัพท์ และ INTERFACE กับคอมพิวเตอร์

เป็นส่วนที่ทำหน้าที่ตรวจสอบสัญญาณในสายโทรศัพท์ ว่าขณะนี้มีผู้โทรเข้ามาใช้บริการหรือไม่ และ ตรวจสอบสัญญาณ DTMF เพื่อ รับทราบการขอใช้บริการของผู้ใช้บริการ ว่าต้องการใช้บริการฝังข้อความหรือ ฝากข้อความให้กับบุคคลอื่น มีส่วนที่ใช้สำหรับตรวจสอบสถานะของสายโทรศัพท์ว่า สายว่างหรือไม่ หรือมีผู้รับสายหรือไม่ นอกจากนั้นยังมีส่วนที่ทำหน้าที่ สร้างสัญญาณ DTMF เพื่อใช้ในการต่อสาย โทรออกไปภายนอกได้อีกด้วย

นอกจากนี้ ยังมีส่วนที่ใช้ INTERFACE กับคอมพิวเตอร์ โดยทำเป็นการ์ดสำหรับเสียบในสล็อตของเครื่องคอมพิวเตอร์ได้ ดังนั้น ก่อนการใช้งาน จะต้องทำการเสียบการ์ดนี้ลงในสล็อตของเครื่องเสียก่อน และส่วนที่จะติดต่อกับ SOUND CARD คือส่วนที่ออกจาก TRANSFORMER ที่มีแจ็คสำหรับเสียบสัญญาณ 2 ช่อง ซึ่งจะต่อไปยังแจ็ค MIC และ SPEAKER ของ SOUND CARD

ส่วนที่ติดต่อกับสายโทรศัพท์จะเป็นแจ็คอีกหนึ่งตัว สำหรับเสียบสายโทรศัพท์เข้ากับระบบแยกต่างหากออกมา

3. ส่วนซอฟท์แวร์ควบคุมการทำงานของระบบ

เป็นซอฟท์แวร์ที่ตรวจสอบสถานะของระบบ ว่าขณะนี้ระบบมีสถานะเป็นอย่างไร เช่น มีผู้โทรเข้ามาใช้บริการระบบ หรือ กำลังตรวจสอบสัญญาณอยู่ และเมื่อมีผู้โทรเข้ามาใช้บริการระบบแล้ว ผู้ใช้บริการเลือกใช้บริการอะไร หรือกำลังทำการโทรออกอยู่ เหล่านี้เป็นต้น

มีหน้าที่เตรียมสถานะเริ่มต้นของระบบ เช่น จองหน่วยความจำ สำหรับใช้ในการถ่ายเทข้อมูลสัญญาณเสียง หรือ เตรียมไฟล์ที่เก็บข้อมูลสถิติการใช้บริการ ของผู้ใช้

บริการคนหนึ่ง ๆ

การติดตั้งซอฟท์แวร์ ก็ทำได้โดยรันโปรแกรม EVS.EXE รวมทั้งต้องมีไฟล์เสียงที่กล่าวมาแล้วข้างต้นครบถ้วน

การใช้งานระบบ

เริ่มจากการติดตั้งทั้ง การ์ดวงจร การ์ดเสียง และติดตั้งซอฟท์แวร์เสร็จสิ้น จะปรากฏเมนูแบบกดปุ่มให้เลือก ทั้งสิ้น 5 ปุ่มคือ

1. User Admin
2. Telephone Admin
3. Message File
4. Start EVS
5. Quit

โดยมีรายละเอียดการใช้งานในแต่ละหัวข้อดังนี้

1. User Admin

เป็นเมนูที่ใช้สำหรับให้ Admin เพิ่ม user ที่สามารถใช้งานบริการระบบได้ เมื่อเรียกเมนูนี้ จะปรากฏเป็น template ของ user ขึ้นมาให้ admin เติม ชื่อ หมายเลขประจำตัวของ user รหัสผ่าน และเบอร์โทร์ที่จะสามารถติดต่อกลับไปได้

จะมี radio box เพื่อให้ admin เลือกระยะเวลาของ mail box ของ user แต่ละคน ตั้งแต่ 5, 10 และ 15 นาที radio box อีกอันหนึ่งเป็นการเลือกบริการที่สามารถใช้ได้ เช่น สามารถใช้ Voice Mail Box ได้ หรือ สามารถใช้ General Information ได้ หรือสามารถให้โทรกลับได้ (Phone Back) เป็นต้น

ส่วนล่างสุดจะเป็นปุ่มกดเลือกสำหรับการ ทำงานกับรายชื่อของ user ทั้ง การ add user การ delete user การ edit user การ update user เป็นต้น ซึ่งในส่วนการ add user นั้น เมื่อ admin ใส่รายละเอียดของ user จนครบถ้วนแล้ว admin สามารถเลื่อนมากดที่ปุ่มนี้เพื่อ add user ได้ ถ้าหากหมายเลขประจำตัวซ้ำกับหมายเลขประจำตัวในฐานข้อมูล จะมีข้อความเตือนว่า ไม่สามารถ add user ที่มีหมายเลขประจำตัวนี้ได้ ให้เปลี่ยนหมายเลขประจำตัวใหม่

การ edit user นั้น ให้ใส่หมายเลขประจำตัวของผู้ที่ต้องการจะ เปลี่ยนแปลงแก้ไขรายละเอียดของ user คนนั้น จากนั้น ให้กดที่ปุ่ม " Edit User " ถ้าหากมี user ที่มีหมายเลขประจำตัวเช่นนี้อยู่ จะถูกดึงมาแสดงผลใน template และ admin สามารถเปลี่ยนแปลงแก้ไขได้

เมื่อมีการเปลี่ยนแปลงแก้ไขจนกระทั่งถูกต้องเรียบร้อยแล้ว สามารถกดปุ่ม " Update User " เพื่อนำข้อมูลใหม่นี้เก็บไว้ในฐานข้อมูล ณ ตำแหน่งเดิมที่ข้อมูลเคยอยู่

ส่วนการ delete user นั้น ก็เพียงแต่ใส่หมายเลขประจำตัวของ user และกดปุ่ม " Delete User " user ที่มีหมายเลขประจำตัวตรงกับที่ใส่ไว้ จะถูก delete ออกจากระบบ หากไม่มี user ดังกล่าว จะมีข้อความเตือนว่า ไม่มี user คนนี้

2. Telephone Admin

เป็นส่วนที่ทำหน้าที่ในการกำหนดค่าเวลา 2 อย่างคือ

1. ระยะเวลาในการดังของโทรศัพท์ ขณะที่รอให้มีผู้มารับสาย เช่นกำหนดให้ดังอยู่นาน 10 ครั้ง ถ้าหากดังเกินกว่านี้ จะถือว่าไม่มีผู้รับสาย ตัวแปรตัวนี้ก็สามารถกำหนดได้ด้วย ปุ่มฟังก์ชันนี้
2. ระยะเวลาในการรอคอยให้มีการโทรเข้ามาในระบบ ซึ่งช่วงเวลานี้จะไม่มีการทำงานใด ๆ เช่น กำหนดไว้ 10 นาที ถ้าหากไม่มีผู้ใดโทรเข้ามา ระบบจะตรวจดูว่า มี queue ของผู้ที่ต้องโทรออกหรือไม่ หากมี จะนำเบอร์โทรของผู้นั้น มาต่อโทรศัพท์เพื่อโทรออกทันที แต่ถ้าหากในช่วงระยะเวลานี้ มีผู้โทรเข้ามาในระบบ ระบบจะให้บริการแก่ผู้ที่โทรเข้ามาก่อน จากนั้น เมื่อเสร็จสิ้นการทำงานบริการแล้ว จึงค่อยกลับไปรอการโทรเข้าครั้งต่อไป

3. Message File

มีฟังก์ชันการทำงานให้เลือกอยู่ 2 อย่างคือ

1. clean up Message File

โดยที่ใน Mail Box ของ User แต่ละคน อาจจะมี message อยู่จนเต็มแล้ว เป็นหน้าที่ของ admin จะต้องทำการตรวจสอบ Mail Box ของ User และ

ทำการ clean up ถ้า user ร้องขอ

Message File เมื่อฟังแล้ว จะมีอักษรชื่อไฟล์ตัวที่ 8 เปลี่ยนจาก 0 เป็น 1 เพื่อให้สามารถแยกได้ว่า message ใด ฟังแล้วหรือยัง ซึ่งสามารถนำมาเป็นเกณฑ์การลบไฟล์ได้ และตัวแปรอีกตัวหนึ่งคือ ฟังมาแล้วกี่วัน ซึ่งส่วนนี้จะมีฟังก์ชัน ซึ่งใช้สำหรับคำนวณ เวลาที่ไฟล์ถูกสร้างไว้ ว่ามีการสร้างไฟล์ไว้ตั้งแต่เมื่อกี่วันมาแล้ว และนำมาใช้เป็นอีกเกณฑ์หนึ่งในการลบไฟล์ได้

2. Set First Message

เพื่อใช้ในการเป็น General Information สำหรับให้ user ที่โทรเข้าสู่ระบบ ฟังได้ว่า admin ต้องการจะให้ทราบข่าวคราวใด อาจจะเป็นข่าวทั่วไปที่ admin สามารถ set ให้ user ทุกคนได้รับฟังเหมือนกันหมด หรือเป็น message เฉพาะ ที่ admin สามารถกำหนดให้ user คนนั้นคนนี้ฟังโดยเฉพาะได้

4. Start EVS

เป็นปุ่มเลือก ให้ระบบเริ่มทำการรอรับสัญญาณกระดิ่งโทรศัพท์ เพื่อเริ่มให้บริการ เมื่อ user โทรเข้ามาที่ระบบ จะได้ยินเสียงต้อนรับ

" สวัสดีค่ะ นี่คือเสียงจากบริการรับฝากข้อความทางโทรศัพท์อัตโนมัติ "

จากนั้น ระบบจะให้ user กดปุ่มเพื่อเลือกใช้บริการ หรือหากเป็นผู้โทรเข้าทั่ว ๆ ไป ซึ่งต้องการเพียงแค่การโอนสายไปยังปลายสายอื่น ก็สามารถทำได้โดยการกดเบอร์โอนได้โดยตรง หากเป็น user ที่ต้องการใช้บริการพิเศษ ก็ทำได้โดยการกดอักษร #

จะเห็นว่า การให้บริการแยกเป็น 2 ส่วน ให้แก่ผู้โทรเข้าเพื่อโอนสายทั่วไป และส่วนที่ให้บริการแก่ user ที่ register เข้ามาใช้งานระบบ

ส่วนการโอนสายอัตโนมัติ

เมื่อมีการใช้การโอนสายอัตโนมัติ ผู้โทรจะได้ยินเสียงดนตรี ซึ่งเป็นการตั้งที่ตัว PABX เป็นสัญญาณบอกว่า ขณะนี้เป็นการโอนสาย ให้ถือหูรอผู้รับปลายอีกสายหนึ่ง ในช่วงนี้ ระบบสามารถตรวจสอบได้ว่า มีผู้รับสายหรือไม่ โดยการนับสัญญาณกระดิ่งที่ดังติดต่อกันนานกว่าระยะเวลาที่กำหนดไว้ เช่น อาจจะเป็น 10 ครั้ง เป็นต้น หากเกินกว่า 10 ครั้งแล้ว ไม่มีผู้รับสาย ระบบก็จะตอบไปยังผู้ใช้ว่า

" ไม่มีผู้รับสายค่ะ "

และถามประสงค์ของผู้โทรด้วยว่า ต้องการจะฝากข้อความไว้ให้แก่ผู้รับปลายสายหรือไม่ โดยจะถามว่า

" กรุณากดหมายเลขประจำตัวของผู้ที่ท่านจะฝากข้อความไว้ให้
หรือกด 0 เพื่อยกเลิก "

ถ้าหากผู้โทรเข้า รู้หมายเลขประจำตัวของผู้ที่เขาต้องการจะโทรคุยด้วย ก็สามารถฝากข้อความไว้ใน Mail Box ได้ แต่ถ้าไม่รู้ก็อาจจะกด 0 เพื่อยกเลิกการใช้งาน

ในกรณีที่โอนสายแล้ว ระบบตรวจสอบสัญญาณได้ว่า สายไม่ว่าง คือปลายสายมีการใช้สายโทรศัพท์อยู่ ระบบก็จะตอบไปยังผู้โทรเข้าว่า

" สายไม่ว่างค่ะ "

พร้อมทั้งถามความประสงค์ผู้โทรเข้า ว่าจะฝากข้อความไว้ให้แก่ผู้ที่เขาต้องการจะคุยด้วยหรือไม่ เช่นเดียวกับที่ถามในกรณีที่ไม่มีผู้รับสาย หากผู้โทรเข้าไม่ประสงค์ที่จะฝากข้อความไว้ ก็จะกด 0 เพื่อยกเลิกการใช้งาน

ส่วนของการใช้บริการพิเศษ

ได้แก่บริการ Voice Mail Box

บริการ General Information

บริการ Call Back

เมื่อ user กดเครื่องหมาย "#" ระบบจะให้เลือกใช้บริการโดยกล่าวว่า

" ท่านสามารถเลือกใช้บริการ โดยกดปุ่มหมายเลขดังต่อไปนี้

1. ฝากข้อความ
2. รับฟังข้อความที่มีผู้ฝากไว้
3. สั่งให้โทรออก
0. ยกเลิกการใช้บริการ "

เมื่อ กด 1

ระบบจะถามหมายเลขประจำตัวของผู้ที่ user จะฝากข้อความไว้ให้ เมื่อได้หมายเลขประจำตัวถูกต้องแล้ว ก็จะเริ่มให้ฝากข้อความ จากนั้น ก็จะวนกลับไปยังการสอบถามความประสงค์การใช้บริการอีก

เมื่อ กด 2

ระบบจะนำข้อความที่มีผู้ฝากไว้ และยังไม่ได้รับการเปิดฟัง มาเล่นให้ user ฟัง เมื่อฟังเสร็จสิ้นแต่ละข้อความ จะถามความประสงค์ของ user ว่า จะลบข้อความที่เพิ่งฟังนี้ทิ้งหรือไม่ เพื่อที่ user จะได้มีเนื้อ ที่ Mail Box ไว้สำหรับเก็บข้อความอื่นต่อไป

เมื่อ กด 3

ระบบจะถามหมายเลขประจำตัวของผู้ที่ต้องการจะสั่งให้โทรออก และเก็บข้อความที่ต้องการฝากไว้ จากนั้น เมื่อมีการรอสัญญาณกระดิ่งจนถึงช่วงระยะเวลาที่กำหนดไว้ และยังไม่มีผู้ใดโทรเข้ามาใช้บริการ ระบบจะต่อโทรศัพท์ เพื่อโทรออกไปยัง user ที่กำหนดไว้ โดยที่ถ้าหากมีผู้รับ จะเล่นเสียงที่เก็บไว้ให้แก่ผู้รับฟัง และลบข้อความนั้นทิ้ง แต่ถ้าหากไม่มีผู้รับสาย หรือสายไม่ว่าง จะนำข้อความและหมายเลข

โทรศัพท์นั้น มาต่อท้าย queue สำหรับทำการต่อโทรศัพท์ใหม่ต่อไป

การทำงานของระบบสามารถแสดงเป็น FLOWCHART ดังแสดงดังต่อไปนี้

1. USER INTERFACE แสดงขั้นตอนการเริ่มต้นใช้งานระบบ โดยจะตรวจสอบว่ามีการกด key ESC หรือไม่ ถ้ามี แสดงว่าให้หยุดการบริการชั่วคราว เพื่อไปทำงานด้าน Admin ถ้าไม่มีการกดคีย์ จะตรวจสอบว่ามีสัญญาณกระดิ่งดังหรือไม่ ถ้ามี ก็จะทำการต่อโทรศัพท์ เพื่อรับโทรศัพท์และรับ DTMF ตรวจสอบอักษรตัวแรกว่า เป็นตัวเลขหรือไม่ ถ้าเป็นตัวเลข แสดงว่า เป็นการโอนสายอัตโนมัติ แต่ถ้าเป็น "#" แสดงว่า เป็นการใช้บริการพิเศษ

2. DIRECT INVERT DIALING การโอนสายอัตโนมัติ จะมีการตรวจสอบสายว่าว่างหรือไม่ หรือถ้าว่าง มีผู้รับสายหรือไม่ ถ้าไม่ว่างหรือไม่มีผู้รับสาย ก็จะให้เลือกได้ว่าจะฝากข้อความไว้หรือไม่

3. FUNCTION จะมีการตรวจสอบหมายเลขประจำตัวของผู้ที่จะใช้บริการพิเศษ ถ้าหากเป็นผู้ที่มีสิทธิใช้งาน ก็จะถามความประสงค์ว่า ต้องการจะใช้บริการใดบ้าง

บริการที่ให้แก่ผู้ใช้บริการได้แก่

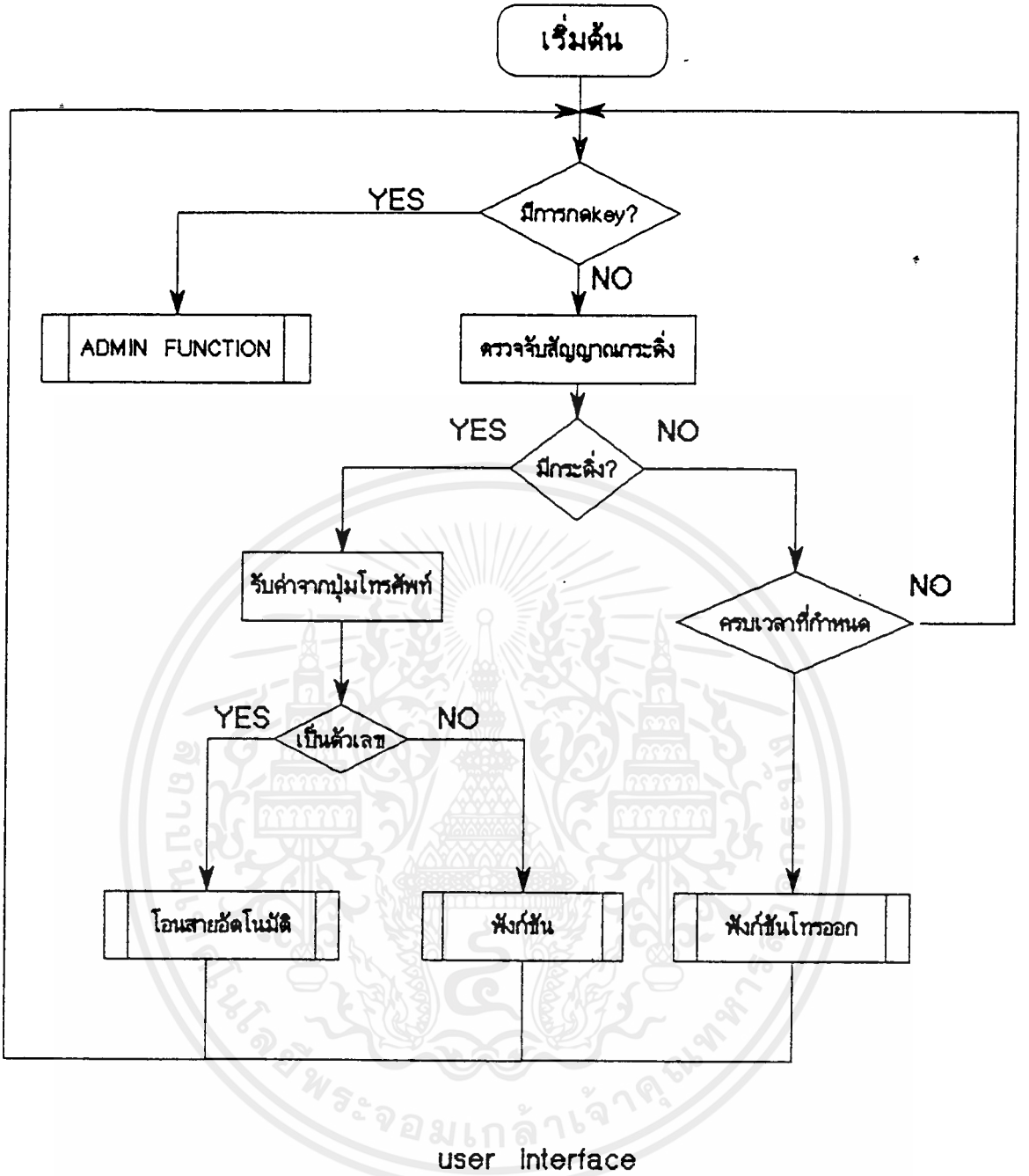
1. ฝากข้อความ (Record)
2. รับฟังข้อความ (Play)
3. สั่งให้โทรออก

4. RECORD การฝากข้อความจะเริ่มจากการถามหมายเลขประจำตัวของผู้ที่ต้องการจะฝากข้อความไว้ให้ และเริ่มฝากข้อความ

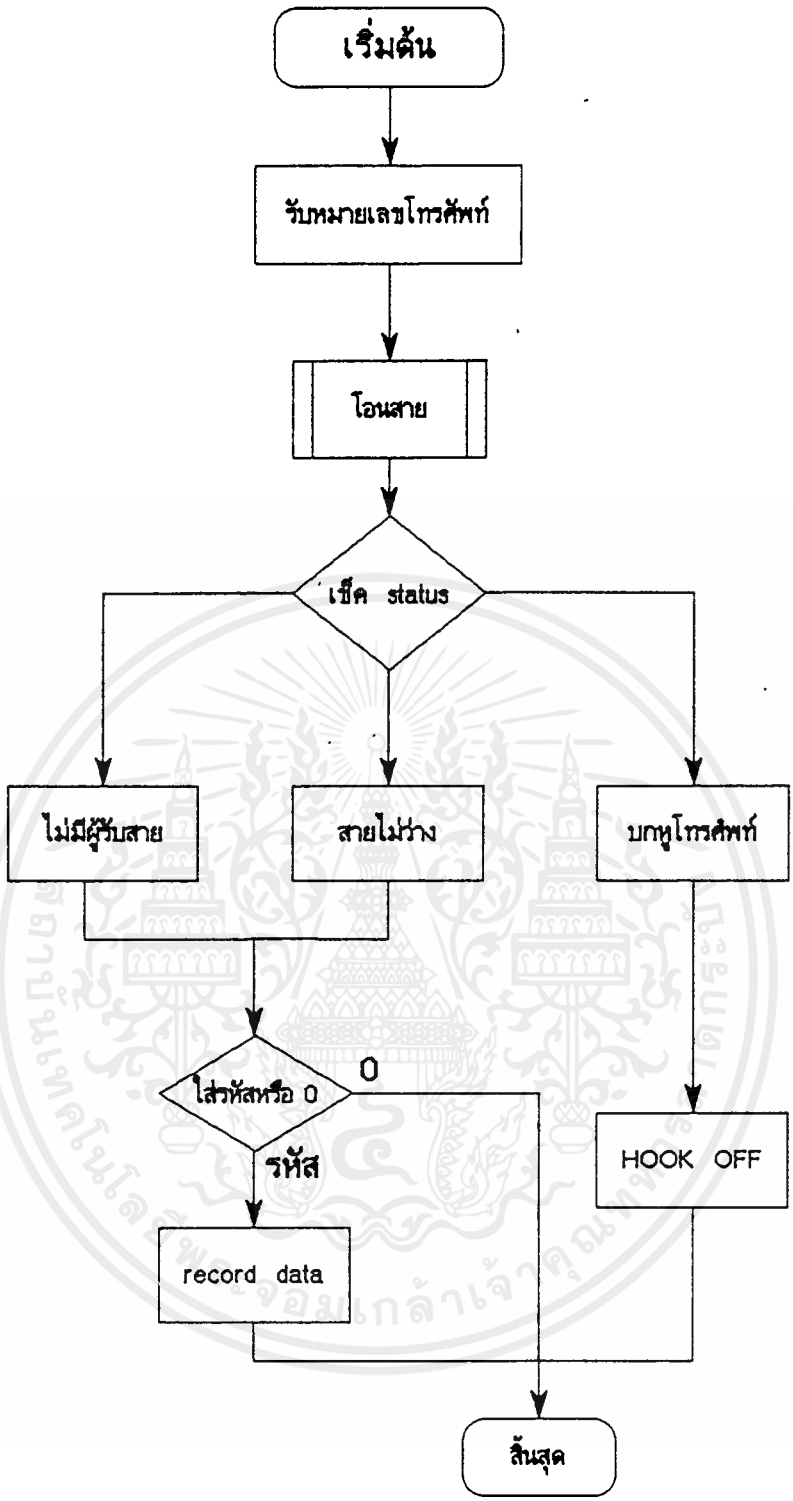
5. PLAY การรับฟังข้อความที่มีผู้ฝากไว้ จะให้ฟังเฉพาะข้อความที่ยังไม่ได้มีการเปิดฟังเท่านั้น และเมื่อฟังเสร็จสิ้นแล้ว จะถามความประสงค์ผู้ใช้ว่าจะลบข้อความนี้ทิ้งหรือไม่ และจะให้ฟังข้อความถัดไปจนหมด

6. สั่งให้โทรออก จะเริ่มด้วยการถามหมายเลขประจำตัวของผู้ที่ต้องการจะสั่งให้โทรออก และรับฝากข้อความไว้

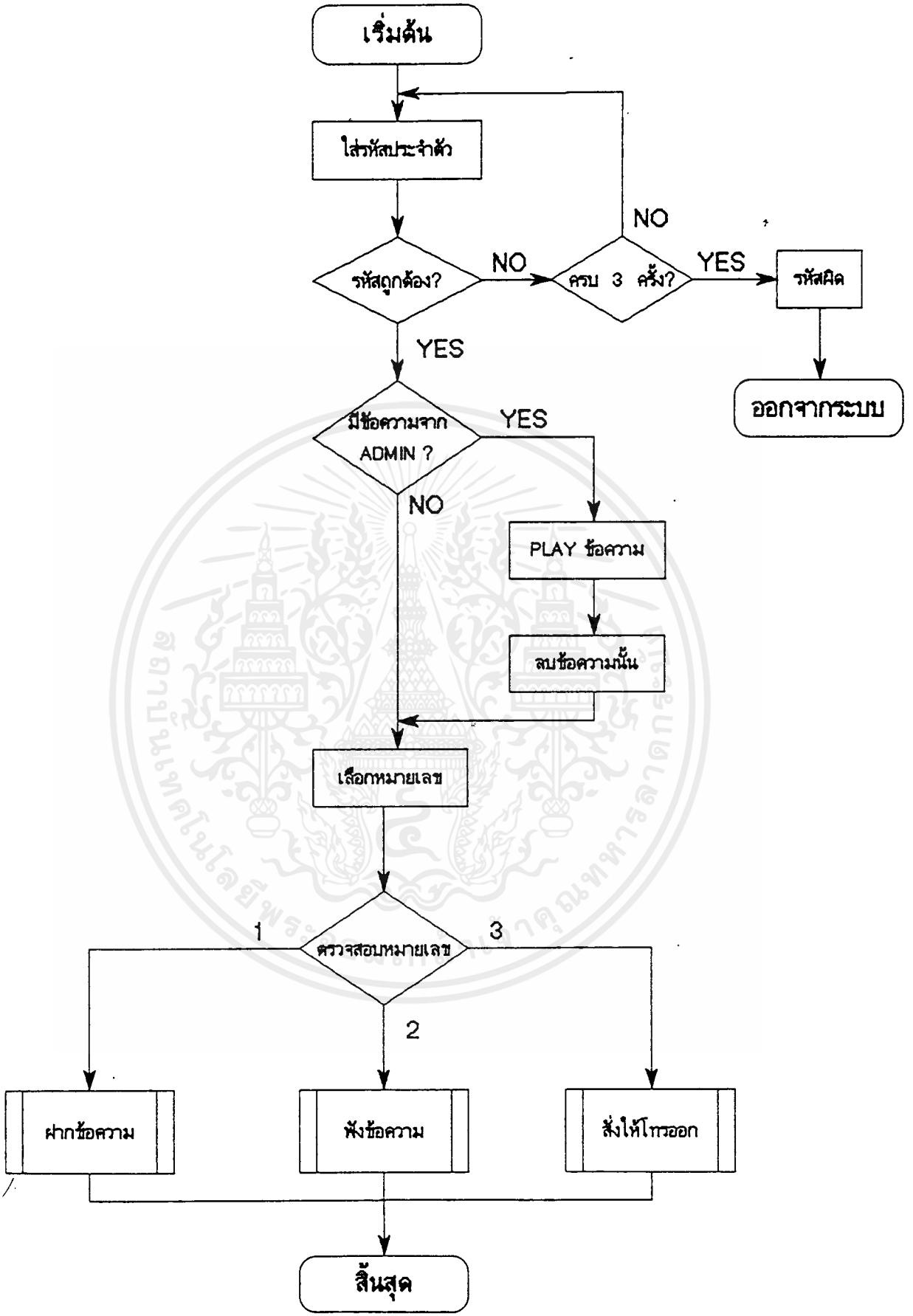
7. CALL OUT เมื่อครบระยะเวลาในการรอสัญญาณกระดิ่งแล้ว และไม่มีผู้ใดโทรเข้ามาเพื่อใช้งาน ระบบจะต่อโทรศัพท์โทรออกไปยังผู้รับ ซึ่งรายชื่อและเบอร์โทรศัพท์จะเก็บไว้ใน queue เมื่อมีผู้รับสาย จะเล่นเสียงที่อัดเก็บไว้ให้ผู้รับฟัง และลบข้อความที่ฝากไว้เสีย แต่ถ้าไม่มีผู้รับสาย หรือสายไม่ว่าง จะนำชื่อและเบอร์โทรศัพท์ไปต่อท้าย queue สำหรับโทรอีกครั้งในภายหลัง



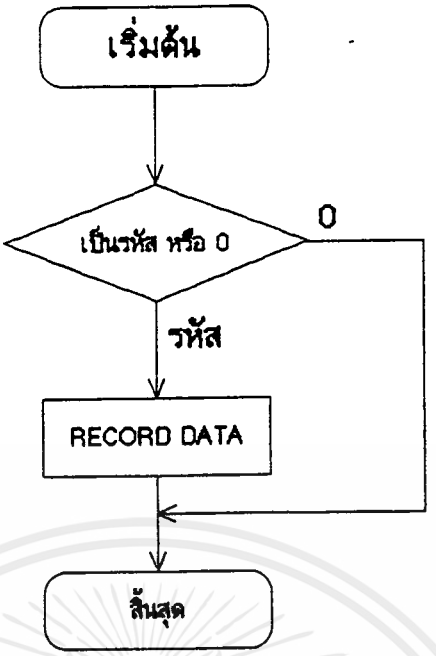
user Interface



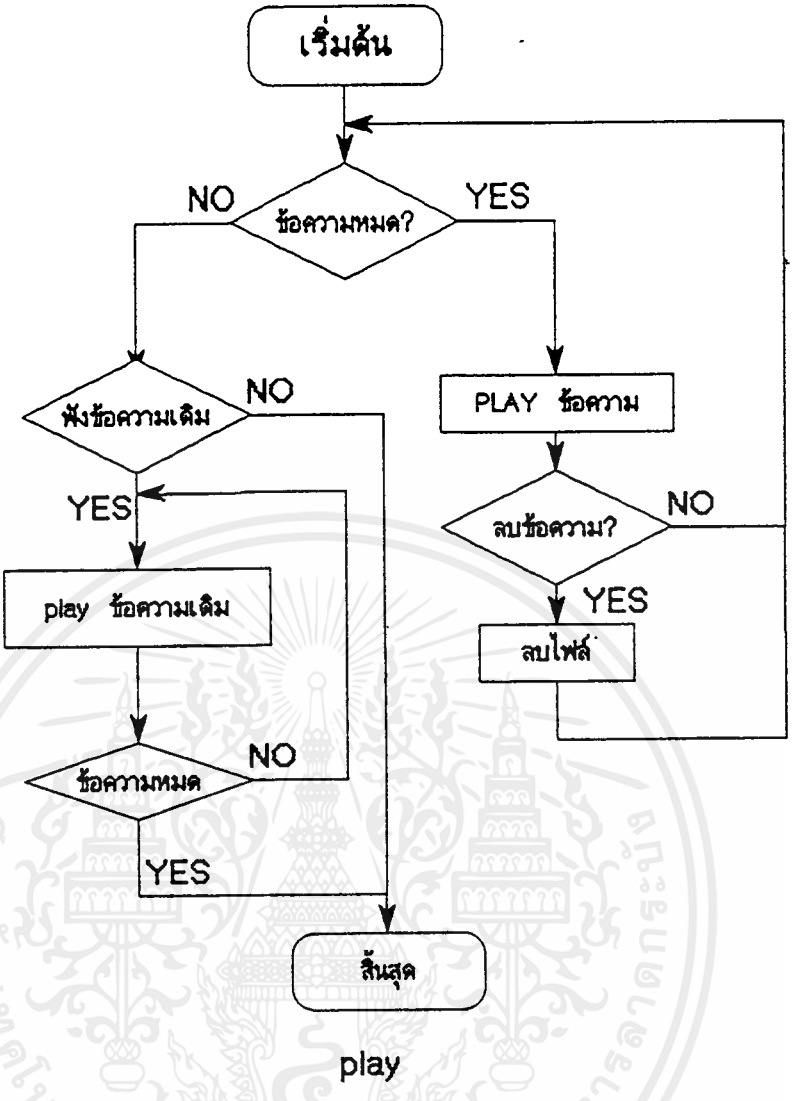
direct invert dialing



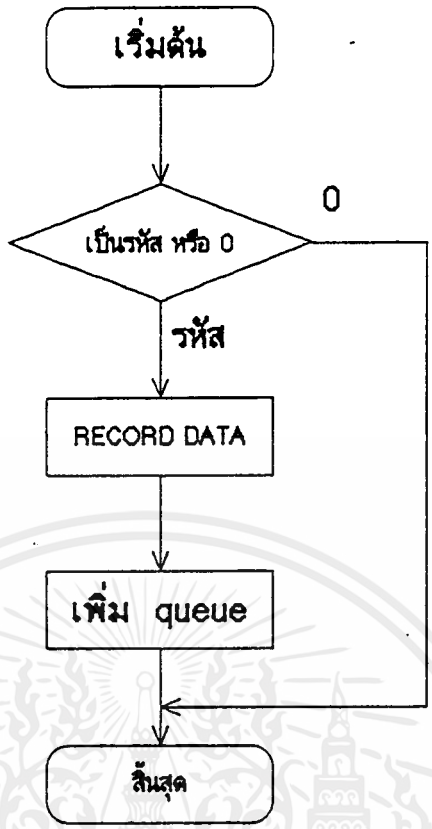
function



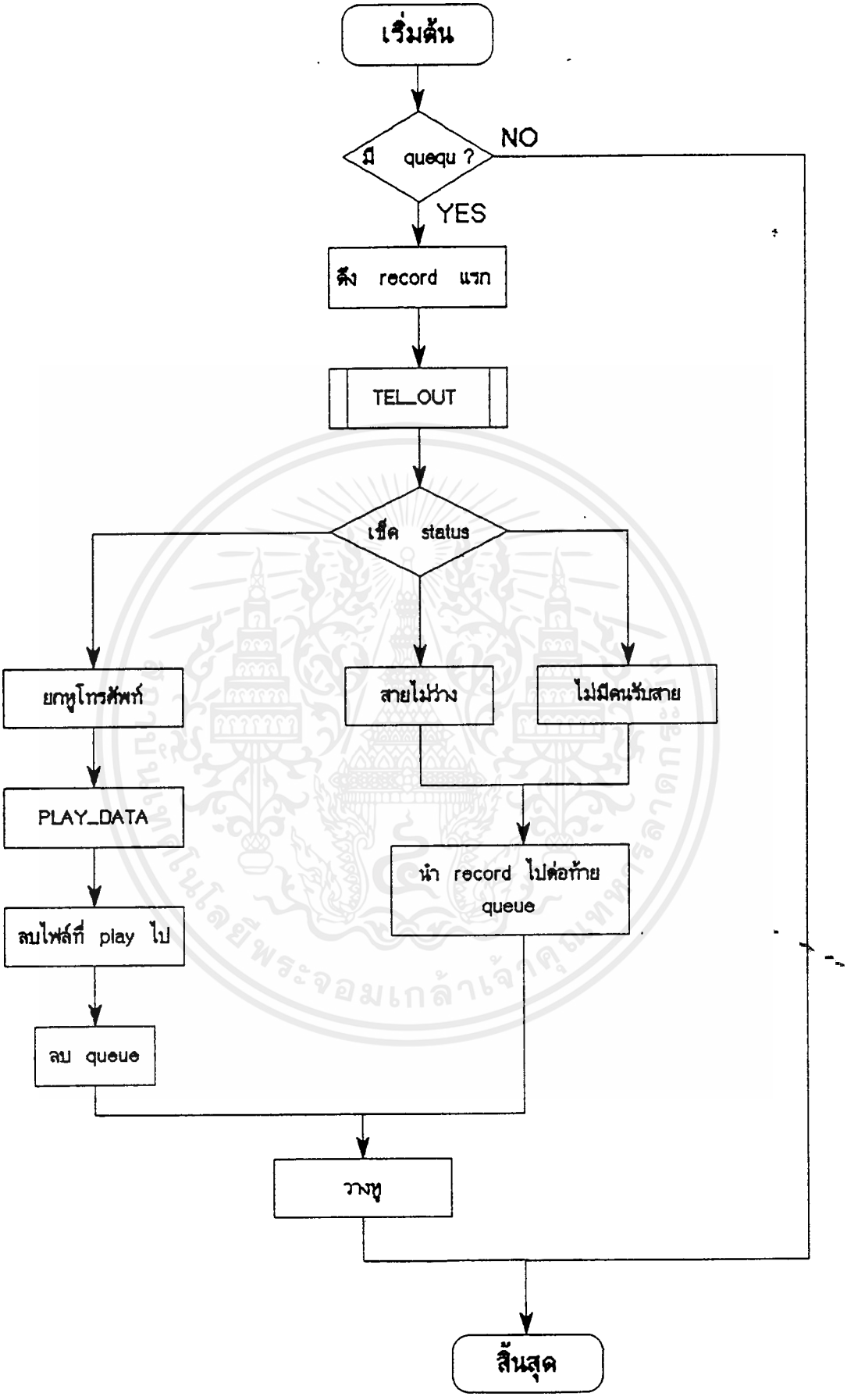
record



play



สั่งโทรออก



CALL OUT

การจัดการฐานข้อมูลผู้ใช้บริการ

ในการให้บริการแก่ผู้ใช้บริการที่เป็นสมาชิกนั้น จะต้องมีการตรวจสอบรหัสประจำตัว และรหัสผ่าน ว่าเป็นผู้ที่สามารถใช้บริการได้หรือไม่ รวมทั้งจะต้องสามารถเก็บรายละเอียดเกี่ยวกับการใช้บริการ ของผู้ใช้บริการแต่ละคน ว่ามีการใช้บริการใดบ้าง เมื่อใด เป็นต้น ซึ่งการจะเก็บข้อมูลเหล่านี้ไว้ จะต้องมีการจัดการฐานข้อมูลของผู้ใช้บริการที่ดี หมายถึง จะต้องรวดเร็ว ทันต่อการตอบสนองการใช้งานของผู้ใช้บริการ การตรวจสอบรหัสประจำตัว และรหัสผ่าน จะต้องรวดเร็ว และถูกต้อง ดังนั้น การเลือกใช้วิธีในการจัดการฐานข้อมูลในบริการ Electronics Voice System นี้จึงได้ใช้วิธีการที่มีชื่อว่า Extensible Hashing ซึ่งเป็นวิธีการเข้าถึงข้อมูลแบบสุ่ม (Keyed Random access Method)

Extensible Hashing

Extensible Hashing ใช้คีย์ของเรคคอร์ดเพื่อคำนวณค่า Hash Code โดยที่จำนวนบิท N บิทแรกของคีย์ (ในที่นี้ใช้ N = 10) จะถูกใช้เป็นดัชนี (Index) เพื่อชี้ไปยังตารางของหมายเลขบล็อก เร็คคอร์ดที่มี Hash Code เหมือนกัน จะถูกเก็บไว้ในบล็อกข้อมูลเดียวกัน

เมื่อบล็อกข้อมูลเต็ม จะต้องมีการแยกบล็อกข้อมูลนั้นออกเป็น 2 บล็อกใหม่ เพื่อให้เกิดพื้นที่ว่าง ให้สามารถใส่ข้อมูลลงไปได้ การกระจายของข้อมูลใน 2 บล็อกใหม่ นั้นขึ้นอยู่กับจำนวนบิทที่มากขึ้นอีก 1 บิท เมื่อเทียบกับก่อนการแยกบล็อก

Index-in-Memory

วิธีการสำหรับการเพิ่มขนาดของ Kram File มีด้วยกัน 2 วิธีคือ

1. เก็บตารางดัชนี ไว้ในหน่วยความจำทั้งหมด และตลอดเวลา เพื่อความง่ายในการเขียนโปรแกรม และความเร็วในการเข้าถึงข้อมูล เมื่อมีการเปลี่ยนแปลงค่าในตารางดัชนี จึงจะมีการเขียนข้อมูลในตารางดัชนีลงสู่ดิสก์ครั้งหนึ่ง เพื่อให้ข้อมูลในหน่วยความจำและบนดิสก์ ถูกต้องตรงกันเสมอ แต่วิธีการนี้จะต้องจำกัดขนาดของไฟล์ที่จะสามารถเก็บข้อมูลได้ เช่น ถ้าหากสามารถจัดเนื้อที่ 140 Kbyte สำหรับเก็บตารางดัชนี

และบัฟเฟอร์ ก็จะสามารถเข้าถึงข้อมูลบนดิสก์ไฟล์ที่มีขนาดประมาณ 128 Mbyte ได้ในการอ่านดิสก์หนึ่งครั้ง

2. เก็บตารางดัชนี ไว้ในหน่วยความจำเพียงบางส่วน และอ่านจากดิสก์เมื่อไม่พบในหน่วยความจำแล้ว ซึ่งจะเห็นว่า จะต้องมีการอ่านดิสก์ 2 ครั้ง แต่ขนาดของหน่วยความจำที่ใช้ในการเข้าถึงข้อมูล ซึ่งประกอบด้วย ตารางดัชนี และบัฟเฟอร์นั้น จะใช้เพียง 16 Kbyte เท่านั้น

โครงสร้างข้อมูล (Data Structure)

โครงสร้างข้อมูลที่ใช้ในโปรแกรมส่วนนี้ มีขนาดที่สามารถปรับเปลี่ยนได้โดยการผ่านพารามิเตอร์ เพื่อให้สามารถปรับขนาดของไฟล์ข้อมูล ให้เหมาะสมกับงานที่จะใช้ได้ ขนาดของไฟล์ที่ใหญ่ที่สุดที่สามารถใช้ได้ในโปรแกรมนี้ สามารถคำนวณได้จาก

DATASIZE * INDEXCOUNT * 0.67

DATASIZE หมายถึง ขนาดของบล็อกข้อมูลแต่ละบล็อกเป็นไบท์ INDEXCOUNT หมายถึง จำนวนของดัชนีในตารางดัชนี และตัวเลข 0.67 เป็น packing factor

ตัวอย่างเช่น ถ้าต้องการเก็บข้อมูลขนาด 100,000 เรคคอร์ด เรคคอร์ดละ 100 ไบท์ ซึ่งจะเป็นไฟล์ขนาดประมาณ 10 Mbyte ค่าของ INDEXCOUNT อาจจะเป็น 8192 และ DATASIZE อาจจะมีค่า 2048 ซึ่งจะได้ขนาดไฟล์ที่สามารถเก็บข้อมูลได้ 16 Mbyte

จำนวนของหน่วยความจำที่ใช้สำหรับเก็บ ตารางดัชนี และบัฟเฟอร์ของข้อมูลสามารถคำนวนได้จาก

(2*INDEXCOUNT) + (3*DATASIZE);

ตัวเลข 3 ที่คูณกับ DATASIZE ก็เพราะจะต้องใช้บัฟเฟอร์อีก 2 บล็อกสำหรับข้อมูลที่แบ่งออก เมื่อเกิดข้อมูลเต็มบล็อก ดังนั้น จากข้อมูลขนาด 100,000 เรคคอร์ด เรคคอร์ดละ 100 ไบท์ จะต้องใช้หน่วยความจำจำนวน 2*8192 + 3*2048 หรือประมาณ 22 Kbyte

การเตรียมสถานะเริ่มต้น (Initialization)

KramInit เป็นฟังก์ชันที่ใช้สำหรับการสร้าง KramFile ไฟล์ใหม่ โดยกำหนดให้บล็อกข้อมูลแรกมีข้อมูลเป็น 0 หมด และกำหนดให้ค่าดัชนี ในตารางดัชนีชี้มายังบล็อกข้อมูลแรกนี้ทั้งหมด และทำการกำหนดค่าเริ่มต้นในส่วนที่เป็นพารามิเตอร์ของไฟล์

KramOpen เป็นฟังก์ชันที่ทำหน้าที่เปิดไฟล์ที่เป็น KramFile การทำงานของฟังก์ชัน เริ่มจากการเปิดไฟล์ และอ่านบล็อกพารามิเตอร์ และตารางของดัชนีเข้าสู่หน่วยความจำ รวมทั้งจองหน่วยความจำสำหรับเป็น บล็อกข้อมูลชั่วคราวสำหรับเก็บข้อมูลที่จะต้องเก็บในบล็อกที่เต็ม

การเพิ่มเรคคอร์ดในไฟล์

KramAdd เป็นฟังก์ชันที่ทำหน้าที่เพิ่มเรคคอร์ดในไฟล์ ถ้าหากสามารถเพิ่มเรคคอร์ดในไฟล์ได้ จะส่งค่า TRUE กลับสู่โปรแกรมหลัก แต่ถ้าหากว่าไม่สามารถเพิ่มเรคคอร์ดได้ จะส่งค่า FALSE กลับมาแทน

KramAdd มีการเรียกใช้ฟังก์ชัน HashCode เพื่อคำนวนค่า HashValue ซึ่งจะใช้สำหรับการหาค่าดัชนีในตารางดัชนี ซึ่งค่าดัชนีนี้จะเป็นตัวบอกว่าข้อมูลที่ต้องการนั้น อยู่ในบล็อกข้อมูลใด ถ้าหากบล็อกข้อมูลนั้นไม่อยู่ในหน่วยความจำ ก็จะเก็บบล็อกข้อมูลปัจจุบันลงสู่ดิสก์ และอ่านบล็อกที่ต้องการมาไว้ในหน่วยความจำแทน จากนั้นก็จะตรวจสอบดูว่า คีย์ที่ให้มานั้นซ้ำกับข้อมูลที่มีอยู่ในบล็อกข้อมูลหรือไม่ ถ้าคีย์ไม่ซ้ำ ก็จะต้องตรวจสอบต่อไปว่า มีที่ว่างสำหรับการเพิ่มข้อมูลเรคคอร์ดนั้นเข้าไปในบล็อกข้อมูลหรือไม่ ถ้าหากคีย์ข้อมูลไม่ซ้ำ และมีที่ว่างสำหรับเก็บข้อมูล ข้อมูลนั้นจะถูกเขียนลงสู่หน่วยความจำในตำแหน่งท้ายบล็อก

ถ้าหากว่า บล็อกนั้นเต็มแล้ว คือไม่มีที่ว่างสำหรับเพิ่มข้อมูลได้อีกแล้ว ก็จะต้องแบ่งข้อมูลในบล็อก ออกเป็น 2 บล็อก ที่ชี้โดย *LowDataPtr* และ *HighDataPtr* โดยมีการเปลี่ยนแปลงข้อมูลในตารางดัชนี เพื่อให้สามารถติดตามตำแหน่งของข้อมูลได้อย่างถูกต้อง ขั้นแรกโดยการตรวจสอบดัชนีในตาราง ซึ่งจะถูกแก้ไข จากนั้นกำหนดค่าของดัชนีที่ได้นั้น ให้มีค่าครึ่งหนึ่งเป็นค่าของบล็อกเดิม และอีกครึ่งหนึ่งมีค่าเป็นหมายเลขของบล็อกใหม่ และตรวจสอบข้อมูลในบล็อกเดิมว่า เมื่อมีการเปลี่ยนค่าของดัชนีไปแล้ว จะต้องเปลี่ยนข้อมูลเรคคอร์ดใดบ้าง ไปไว้ในบล็อกข้อมูลใหม่ และข้อมูล

ใดจะต้องอยู่ในบล็อกหมายเลขเดิม

เมื่อเสร็จสิ้นจากการตรวจสอบแล้ว โปรแกรมจะเขียนข้อมูลเหล่านี้ อันได้แก่ ตารางพารามิเตอร์ ตารางดัชนี และบล็อกข้อมูลทั้งบล็อกเดิม และบล็อกใหม่ ลงสู่ไฟล์ เพื่อความถูกต้องของข้อมูล แล้วจึงวนกลับไปตรวจสอบหาที่ว่างของบล็อกข้อมูลใหม่ ว่าเมื่อแบ่งข้อมูลเป็น 2 บล็อกแล้ว จะมีที่ว่างสำหรับเก็บข้อมูลที่ต้องการหรือไม่

ฟังก์ชันอื่น ๆ

KramRead เป็นฟังก์ชันที่ทำหน้าที่อ่านข้อมูลจาก KramFile โดยมีการส่งผ่านค่าของคีย์ให้แก่ฟังก์ชัน มีการทำงานเช่นเดียวกับ KramAdd ยกเว้นจะไม่มีการแบ่งบล็อก เพราะไม่มีการเพิ่มข้อมูล

KramClose ทำหน้าที่ปิดไฟล์ KramFile ซึ่งก่อนที่จะปิดไฟล์ จะมีการตรวจสอบว่าข้อมูลในหน่วยความจำมีการเปลี่ยนแปลงหรือไม่ ถ้าหากมีการเปลี่ยนแปลง ก็จะเขียนข้อมูลในบล็อกข้อมูลลงสู่ไฟล์ ก่อนที่จะปิดไฟล์ และคืนหน่วยความจำที่เป็นตำแหน่งของตารางดัชนี และบล็อกข้อมูลชั่วคราว

HashCode เป็นฟังก์ชันที่ทำหน้าที่หา HashValue เพื่อใช้ในการชี้ไปยังตารางดัชนี ซึ่งจะได้ค่าของบล็อกข้อมูล ซึ่งมีข้อมูลที่ตรงกับคีย์ที่ให้อยู่

SeekBlock ทำหน้าที่เลื่อนไฟล์พอยน์เตอร์ ไปยังบล็อกข้อมูลที่ต้องการ

สรุป

การใช้วิธีการเก็บข้อมูลแบบ Keyed Random Access Method นี้ เป็นวิธีการแบบใช้คีย์ เพื่อหา Hash Code ชี้ไปยังตำแหน่งของข้อมูลได้โดยตรง จึงมีความเร็วในการเข้าหาข้อมูลสูงมาก เหมาะสำหรับการใช้ในโปรเจค Electronic Voice System นี้ แต่ไม่เหมาะหากต้องการการได้ข้อมูลโดยวิธีการ ดึงข้อมูลทีละเรคคอร์ดโดยเรียงข้อมูลด้วย เพราะจะได้ข้อมูลแต่ละเรคคอร์ดนั้น จะต้องให้คีย์แก่ฟังก์ชันด้วย

ข้อจำกัดอีกข้อหนึ่งสำหรับวิธีการเก็บข้อมูลแบบนี้คือ จะต้องเสียเนื้อที่ดิสก์ไปส่วนหนึ่ง สำหรับเก็บตารางดัชนี และเนื้อที่ที่ว่างของบล็อกข้อมูลแต่ละบล็อก เช่น ข้อมูลขนาดประมาณ 1 Mbyte จะต้องเสียเนื้อที่ในดิสก์ประมาณ 1.5 Mbyte ซึ่งถ้าหากข้อจำกัดนี้ไม่กระทบกระเทือนต่อโปรแกรมโดยรวม วิธีการ Keyed Random Access นี้เหมาะมาก เพราะมีความเร็วในการเข้าถึงข้อมูลสูงมาก

การทำงานของฟังก์ชันในไฟล์ SCR.C

ฟังก์ชันในไฟล์ SCR.C ทั้งหมดเป็นฟังก์ชันที่ทำงานเกี่ยวกับการจัดการพิมพ์ตัวอักษร บนหน้าจอ (screen) เพื่อให้ได้ลักษณะการทำงานที่ต้องการ อันได้แก่

การพิมพ์ตัวอักษรหนึ่งตัว ณ ตำแหน่งที่กำหนด (x,y)

การพิมพ์ข้อความ ณ ตำแหน่งที่กำหนด (x,y)

การเติมพื้นที่ด้วยอักษรที่กำหนด

การวาดกรอบของพื้นที่ที่กำหนด

การแสดงเงาของกรอบที่กำหนด

การตรวจสอบโหมดการแสดงผลของจอภาพที่ใช้

พื้นฐานการทำงานกับ VIDEO RAM

อะแดปเตอร์การแสดงผล

อะแดปเตอร์แสดงผลที่ใช้กันในปัจจุบันมีอยู่ 4 แบบคือ โมโนโครม, CGA, EGA และ VGA (CGA ปัจจุบันเลิกผลิตแล้ว) ซึ่งใน 3 แบบหลัง สามารถแสดงผลได้หลายโหมดหลายสี ดังตารางที่ 1 โปรแกรมที่ใช้ทั้งหมดใช้โหมด 80x25 ซึ่งเมื่อเทียบกับตารางก็คือโหมด 2, 3 หรือ 7

โดยปกติหน่วยความจำ (memory) แบบ RAM ของเครื่องคอมพิวเตอร์ จะถูกสงวนไว้สำหรับเก็บลักษระของตัวอักษรที่แสดงบนจอไว้ส่วนหนึ่ง หน่วยความจำส่วนนี้ในจอโมโนโครมจะเริ่มที่แอดเดรส B0000000H ส่วนจอแบบอื่น จะเริ่มที่แอดเดรส B8000000H

การแสดงผลของตัวอักษรในโหมด 80x25 นั้น อักษรแต่ละตัวต้องใช้หน่วยความจำ จำนวน 2 ไบท์ โดยไบต์แรกเก็บรหัสแอสกี้ (ASCII) ของตัวอักษร ไบท์ที่ 2 เก็บแอททริบิวต์ของการแสดงผลแบบสี แสดงดังตารางที่ 2 และสำหรับแบบโมโนโครม แอททริบิวท์จะเหมือนกับแบบอื่น ๆ เฉพาะบิทที่แสดงตัวอักษรกระพริบและบิทที่แสดงตัวอักษรเข้มเท่านั้น นอกจากนั้นแล้วแบบโมโนโครมยังมีแอททริบิวท์ที่มีค่าเท่ากับ 7 เป็นการแสดงตัวอักษรแบบปกติ แอททริบิวท์เท่ากับ 70 เป็นการแสดงตัวอักษรแบบ reverse และแอททริบิวท์เท่ากับ 1 เป็นตัวอักษรแบบขีดเส้นใต้

ตารางที่ 1.1

โหมด	แบบ	ขนาด	อะแดปเตอร์
0	text, b/w	40×25	CGA, EGA
1	text, 16 colors	40×25	CGA, EGA
2	text, b/w	80×25	CGA, EGA
3	text, 16 colors	80×25	CGA, EGA
4	graphics, 4 colors	320×200	CGA, EGA
5	graphics, 4 grey tones	320×200	CGA, EGA
6	graphics, b/w	640×200	CGA, EGA
7	text, b/w	80×25	monochrome
8	graphics, 16 colors	160×200	PCjr
9	graphics, 16 colors	320×200	PCjr
10	graphics, 4 colors PCjr, 16 colors EGA	640×200	PCjr, EGA
13	graphics, 16 colors	320×200	EGA
14	graphics, 16 colors	640×200	EGA
15	graphics, 4 colors	640×350	EGA

ตารางที่ 1.2

บิต	ค่าไบนารี	ความหมาย
0	1	blue foreground
1	2	green foreground
2	4	red foreground
3	8	low intensity
4	16	blue background
5	32	green background
6	64	red background
7	128	blinking character

แอททริบิวท์ของตัวอักษร สามารถหาได้จากการบวกค่าของแอททริบิวท์แต่ละแบบเข้าด้วยกัน เช่น จากตารางที่ 2 ต้องการตัวอักษรกระพริบสีเหลือง พื้นเขียว แอททริบิวท์จะมีค่าเท่ากับ

= ค่าสีแดง + ค่าสีเขียว + พื้นสีเหลือง + ค่ากระพริบ

= 2 + 4 + 32 + 128

= 166

ส่วนโมโนโครม ก็ใช้หลักการเดียวกัน

หน่วยความจำที่สงวนไว้สำหรับการแสดงผล โดยปกติจะมากกว่าที่ใช้ในการแสดงผลแบบ 80x25 เหตุผลข้อแรกคือ เพื่อไว้ใช้ในการแสดงผลโหมดกราฟิก ข้อสองคือ เพื่อสามารถเก็บการแสดงผลหลายหน้าจอไว้ในหน่วยความจำ และสามารถเลือกว่าจะนำหน้าจอไหนมาแสดงได้ เรียกหน่วยความจำของแต่ละหน้าจอว่า video page ซึ่งโดยทั่วไปดอสใช้ page 0 ในการแสดงผล และโปรแกรมทั้งหมด ทำงานกับ page 0 เช่นกัน

วิธีการติดต่อกับส่วนแสดงผลทำได้ 3 วิธีคือ วิธีแรก ติดต่อผ่าน function call ของดอส วิธีนี้ถ้านำมาเขียนโปรแกรมจะทำงานช้ามาก วิธีที่สองคือ ผ่าน function call ของ BIOS ซึ่งทำงานได้เร็วพอควร และวิธีที่สามคือ เขียนอ่านข้อมูลลงบนหน่วยความจำสำหรับการแสดงผลโดยตรง วิธีนี้การทำงานของโปรแกรมจะเร็วที่สุด แต่การเขียนโปรแกรมจะยุ่งยากขึ้น

การตรวจสอบโหมดการแสดงผล

การตรวจสอบโหมดการแสดงผลสามารถทำได้โดยการใช้ interrupt ของ BIOS ดังแสดงในฟังก์ชัน getvmode ดังนี้

```
/*-------------------------------------------------------------*
|  Function     :  getvmode                                   |
|  Usage        :  static int getvmode();                     |
|  Description  :  get current video mode                     |
|  Return       :  current video mode                         |
*-------------------------------------------------------------*/
static int getvmode()
{
   union REGS reg;

   reg.h.ah = 15;/* int 10h, service 15 */
   return int86(0x10,&reg,&reg) & 255;
}
```

ค่าที่ได้จะถูกส่งให้กับฟังก์ชัน initscr(); เพื่อใช้เป็นตัวตัดสินใจว่า จะกำหนดตำแหน่งของ VIDEO RAM ให้เริ่มที่ตำแหน่งใด โดยตรวจสอบจากค่าที่ได้จาก getvmode();

การกำหนดตำแหน่งของ VIDEO RAM

เมื่อได้โหมดการแสดงผลของการแสดงผลมาแล้ว จะนำมาตัดสินเลือกตำแหน่งที่จะเริ่มเป็น VIDEO RAM โดยถ้าหากโหมดการแสดงผลเป็น 2 หรือ 3 VIDEO RAM จะถูกกำหนดที่ตำแหน่ง B8000000h ถ้าหากโหมดการแสดงผลเป็น 7 หมายถึง VIDEO RAM จะถูกกำหนดที่ตำแหน่ง B0000000h

ตัวแปรที่ใช้สำหรับเก็บตำแหน่งเริ่มต้นของ VIDEO RAM คือ ตัวแปรชื่อ vidram ซึ่งมีการ declare ตัวแปรดังนี้คือ

```
static char far *vidram;
```

```
/*-------------------------------------------------------------*
|  Function      :   initscr                                  |
|  Usage         :   void initscr(void);                      |
|  Description   :   initial videoram for direct video access |
|  Return value  :   None                                     |
*-------------------------------------------------------------*/
void initscr()
{
   int vmode;

   vmode = getvmode();
   if((vmode!= 2) && (vmode!=3) && (vmode!=7))
      exit(1);
   vidram = (vmode == 7) ? (char far *) 0xb0000000 :
                           (char far *) 0xb8000000 ;
}
```

การแสดงกรอบ window

การแสดงกรอบวินโดว์ รวมทั้งการแสดงเงาของกรอบนั้น เป็นการเรียกใช้ฟังก์ชันย่อย ๆ หลายฟังก์ชันอันได้แก่

ฟังก์ชันการวาดกรอบ

ฟังก์ชันการวาดเงา

ฟังก์ชันการเติมตัวอักษรในวินโดว์ด้วยสีที่กำหนด

ฟังก์ชันการวาดวินโดว์นี้ ยังมี 2 ลักษณะ คือ กรอบเป็นเส้นเดี่ยว และกรอบเป็นเส้นคู่ โดยเรียกฟังก์ชัน Block(); สำหรับวินโดว์กรอบเส้นเดี่ยว และเรียกฟังก์ชัน Dblock(); สำหรับวินโดว์กรอบเส้นคู่

```
/*-------------------------------------------------------------*
|  Function     :  Block                                      |
|  Usage        :  void block(int startx,int starty,int endx, |
|                             int endy,int attrib);           |
|  Description  :  make shaded window of specific style       |
|  Return       :  None                                       |
*-------------------------------------------------------------*/
void block(int startx,int starty,int endx,int endy,int attrib)
{
   int xscale = 3;
   int yscale = 1;
   register int i,j;
   int x_len,y_len;
   int x_mid,y_mid;

   x_len = endx-startx;   y_len = endy-starty;
   x_mid = startx+(x_len/2);  y_mid = starty+(y_len/2);

   for(i=xscale+3,j=yscale;(i*2)<x_len &&
       (j*2)<y_len;i+=(xscale*2),j+=(yscale*2)) {
       fill(x_mid-i,y_mid-j,x_mid+i,y_mid+j,' ',attrib);
       frame(x_mid-i,y_mid-j,x_mid+i,y_mid+j,attrib);
       delay(50);
   }
   fill(startx,starty,endx,endy,' ',attrib);
   frame(startx,starty,endx,endy,attrib);
   shade(startx,starty,endx,endy);
}
```

```
/*-------------------------------------------------------------*
|  Function     :  Dblock                                     |
|  Usage        :  void dblock(int startx,int starty,int endx,|
|                              int endy,int attrib);          |
|  Description  :  make shaded window of specific style,      |
|                  with double line frame                     |
|  Return       :  None                                       |
*-------------------------------------------------------------*/
void dblock(int startx,int starty,int endx,int endy,int attrib)
{
   int xscale = 3;
   int yscale = 1;
   register int i,j;
   int x_len,y_len;
   int x_mid,y_mid;

   x_len = endx-startx;   y_len = endy-starty;
   x_mid = startx+(x_len/2);  y_mid = starty+(y_len/2);

   for(i=xscale+3,j=yscale;(i*2)<x_len &&
       (j*2)<y_len;i+=(xscale*2),j+=(yscale*2)) {
       fill(x_mid-i,y_mid-j,x_mid+i,y_mid+j,' ',attrib);
       dframe(x_mid-i,y_mid-j,x_mid+i,y_mid+j,attrib);
       delay(50);
   }
   fill(startx,starty,endx,endy,' ',attrib);
   dframe(startx,starty,endx,endy,attrib);
   shade(startx,starty,endx,endy);
}
```

ฟังก์ชันสร้างเงาของกรอบวินโดว์

เป็นฟังก์ชันที่รับอินพุตคือ ตำแหน่งของกรอบวินโดว์ จากนั้นนำตำแหน่งที่ได้มาคำนวณตำแหน่งของเงาที่จะแสดงเยื้องมาทางขวา 2 ตัวอักษร และเยื้องลงข้างล่าง 1 ตัวอักษร ที่ตำแหน่งของเงา จะไม่เขียนตัวอักษรอะไรลงไป เพียงแต่เปลี่ยนแอททริบิวท์ให้เป็น Low Intensity เพื่อให้มองเห็นเป็นเงา

```
/*-------------------------------------------------------------*
|  Function     :  shade                                      |
|  Usage        :  void shade(int x,int y,int endx,int endy);  |
|  Description  :  make shade of the specified block          |
|  Return       :  None                                       |
*-------------------------------------------------------------*/
static void shade(int x,int y,int endx,int endy)
{
   register int i,j;
   unsigned char far *vptr;
   char far *temp;

   temp = vidram;
   for(j=endy;j<endy+1;j++)
      for(i=x+1;i<=endx;i++) {
         vptr = temp + (j*160) + i*2;
         *(++vptr) = LOW_INTENSE;
      }
   for(j=y;j<endy+1;j++)
      for(i=endx;i<=endx+1;i++) {
         vptr = temp + (160*j) + (i*2);
         *(++vptr) = LOW_INTENSE;
      }
}
```

ฟังก์ชันวาดกรอบของวินโดว์

เป็นฟังก์ชันที่ทำหน้าวาดกรอบของวินโดว์ โดยมีอินพุตเป็น ตำแหน่งของวินโดว์คือ มุมบนซ้าย (startx,starty) และมุมล่างขวา (endx,endy) นำมาคำนวนความกว้างและความยาวของพื้นที่วินโดว์ จากนั้นเข้าลูปเพื่อวาดกรอบซึ่งเป็นการเรียกฟังก์ชัน write_char(); วาดตัวอักษรที่เป็นกรอบทั้งแนวนอนและแนวตั้ว ให้ครบทั้งสี่ด้าน จากนั้นจึงเป็นการวาดมุมทั้ง 4

และจากที่ได้กล่าวไปแล้วในหัวข้อฟังก์ชันการวาดวินโดว์ Block(); ซึ่งกล่าวไว้ว่าสามารถเลือกกรอบว่าจะมีรูปแบบเป็นเส้นคู่หรือเส้นเดี่ยวก็ได้ การเลือกเส้นคู่หรือเดียวทำได้โดยการเลือก ฟังก์ชัน frame(); หรือ dframe(); ซึ่งแทนการวาดเส้นเดี่ยว และ เส้นคู่ ตามลำดับ

```
/*-------------------------------------------------------------*
|  Function     :   frame                                     |
|  Usage        :   void frame(int startx,int starty,         |
|                           int endx,int endy);               |
|  Description  :   make frame of specified block             |
|                   The top left corner is (1,1)              |
|  Return value :   None                                      |
*-------------------------------------------------------------*/
void frame(int startx,int starty,int endx,int endy,int attrib)
{
   register int i;
   unsigned char far *vptr, far *temp;


   temp = vidram;
   for(i=startx;i<(endx-1);i++) {
      vptr = temp + ((starty-1)*160) + i*2;
```

```
        *vptr++ = 196;
        *vptr = attrib;
        vptr = temp + ((endy-1)*160) + i*2;
        *vptr++ = 196;
        *vptr = attrib;
    }
    for(i=starty;i<(endy-1);i++) {
        vptr = temp + (i*160) + (startx-1)*2;
        *vptr++ = 179;
        *vptr = attrib;
        vptr = temp + (i*160) + (endx-1)*2;
        *vptr++ = 179;
        *vptr = attrib;
    }
    write_char(startx,starty,218,attrib);
    write_char(startx,endy,192,attrib);
    write_char(endx,starty,191,attrib);
    write_char(endx,endy,217,attrib);
}

/*-------------------------------------------------------------*
|  Function     :    dframe                                    |
|  Usage        :    void dframe(int startx,int starty,        |
|                                int endx,int endy);          |
|  Description  :    make frame of specified block             |
|                    The top left corner is (1,1)              |
|  Return       :    None                                      |
*-------------------------------------------------------------*/
void dframe(int startx,int starty,int endx,int endy,int attrib)
{
```

```
    register int i;
    unsigned char far *vptr, far *temp;

    temp = vidram;
    for(i=startx;i<(endx-1);i++) {
       vptr = temp + ((starty-1)*160) + i*2;
       *vptr++ = 205;
       *vptr = attrib;
       vptr = temp + ((endy-1)*160) + i*2;
       *vptr++ = 205;
       *vptr = attrib;
    }
    for(i=starty;i<(endy-1);i++) {
       vptr = temp + (i*160) + (startx-1)*2;
       *vptr++ = 186;
       *vptr = attrib;
       vptr = temp + (i*160) + (endx-1)*2;
       *vptr++ = 186;
       *vptr = attrib;
    }
    write_char(startx,starty,201,attrib);
    write_char(startx,endy,200,attrib);
    write_char(endx,starty,187,attrib);
    write_char(endx,endy,188,attrib);
}
```

ฟังก์ชันการเขียนตัวอักษร ณ ตำแหน่งที่กำหนด

การเขียนตัวอักษร ณ ตำแหน่งที่กำหนด มีอินพุตคือ ตำแหน่ง (x,y) ซึ่งจะถูกนำไปคำนวณตำแหน่งที่จะเขียนตัวอักษรลงไป จากพื้นฐานการเก็บข้อมูลใน VIDEO RAM ซึ่งจะเริ่มที่ตำแหน่ง B8000000 หรือ B0000000 ขึ้นกับโหมดการแสดงผล ตัวอักษรแต่ละตัวจะใช้หน่วยความจำ 2 ไบท์ ดังได้กล่าวไว้แล้ว ดังนั้น การคำนวนตำแหน่งที่ต้องการจะต้องคูณด้วย 2 เป็นตำแหน่งที่คำนวนสัมพัทธ์กับ ตำแหน่ง B0000000 หรือ B8000000

```
/*-------------------------------------------------------------*
|  Funtion      :  write_char                                 |
|  Usage        :  write_char(int x,int y,char ch,int attrib); |
|  Description  :  write character to by direct  video access |
|  Return       :  None                                       |
*-------------------------------------------------------------*/
yoid write_char(int x,int y,char ch,int attrib)
{
   char far *vptr;

   vptr = vidram;
   vptr += ((y-1)*160) + (x-1)*2;
   *vptr++ = ch;
   *vptr = attrib;
}
```

การเติมพื้นที่ที่กำหนดด้วยตัวอักษร

การเติมตัวอักษรที่กล่าวนี้ ส่วนใหญ่จะเป็นตัวอักษร ว่าง (Blank) และกำหนดสีต่าง ๆ ซึ่งใช้ในการเคลียร์หน้าจอ หรือ การปูพื้น สำหรับใช้เป็นพื้นหลังของโปรแกรม การคำนวนตำแหน่งที่จะเขียนตัวอักษรลงไป เป็นไปตามพื้นฐานการทำงานกับ VIDEO RAM ที่ได้กล่าวไปแล้ว

```
/*-------------------------------------------------------------*
|  Function     :  fill                                       |
|  usage        :  void  fill(int startx,int starty,int endx, |
|                          int endy,int style,int attrib);    |
|  Description  :  Fill specific block with specific style,   |
|                  top left of screen is  (1,1)               |
|  Return       :  None                                       |
*-------------------------------------------------------------*/
void fill(int startx,int starty,int endx,int endy,int style,
          int attrib)
{
   register int i,j;
   unsigned char far *vptr, far *temp;

   temp = vidram;
   for(j=starty-1;j<endy;j++)
      for(i=startx-1;i<endx;i++) {
         vptr = temp + (j*160) + i*2;
         *vptr++ = style;
         *vptr = attrib;
      }
}
```

ฟังก์ชันการเขียนข้อความที่ตำแหน่งที่กำหนด

มีอินพุตคือ ตำแหน่ง (x,y) และ ข้อความที่ต้องการเขียน รวมทั้งแอทริบิวท์ที่ใช้กำหนดสีและลักษณะของการแสดงผลด้วย ใช้ฟังก์ชันพื้นฐาน write_char(); เขียนอักษรที่ตำแหน่งที่กำหนด ทีละตัวและเลื่อนตำแหน่งไปเรื่อย ๆ จนกว่าจะสิ้นสุดข้อความ

```
/*-------------------------------------------------------------*
|  Function     :  write_string                               |
|  Usage        :  write_string(int x,int y,char *s,          |
|                                int attrib)                  |
|  Description  :  write string to videoram                   |
|                  x,y      : coordinate to write             |
|                  char *s  : pointer to string               |
|                  attrib   : attribute of string             |
|  Return       :     None                                    |
*-------------------------------------------------------------*/
void write_string(int x,int y,char *z,int attrib)
{
   register int i;
   char far *vptr;

   vptr = vidram;
   vptr += ((y-1)*160) + (x-1)*2;
   for(i=x;*z;i++) {
      *vptr++ = *z++;
      *vptr++ = attrib;
   }
}
```

ฟังก์ชัน reverse();

เป็นฟังก์ชันที่ทำหน้าที่กำหนดแอททริบิวท์ของจอใหม่ เพื่อให้โปรแกรมที่เรียกสามารถเปลี่ยนแอททริบิวท์ ณ ตำแหน่งที่กำหนดได้

```
/*-------------------------------------------------------------*
|   Function    :  reverse();                                 |
|   Usage       :  int reverse(int x,int y,int length,int attr) |
|   Description :  reverse attribute at x,y for length long   |
|   Return      :  None                                       |
*-------------------------------------------------------------*/
void reverse(int x,int y,int length,int attr)
{
   register int i;
   char far *vptr;

   vptr = vidram;
   vptr += ((y-1)*160) + (x-1)*2;
   for(i=0;i<length;i++) {
     *(++vptr) = attr;
     vptr++;
   }
}
```

ฟังก์ชัน button();

เป็นฟังก์ชันที่ทำหน้าที่ แสดงผลปุ่มที่กด โดยมีชื่อของปุ่มหรือฟังก์ชันการทำงานแสดงผลอยู่บน button เพื่อบอกว่าปุ่มนี้มีการทำงานอะไร และจะแสดงผลอักษรที่กำหนดให้เป็น hotkey ด้วยแอททริบิวท์อื่นที่กำหนดไว้ ให้มีสีแตกต่างจากพื้นและอักษรตัวอื่น ๆ

```
/*-------------------------------------------------------------*
|   Function    : button();                                   |
|   Usage       : void button(int x,int y,char *str,int fg,   |
|                   int bg,int hk);                           |
|   Description : place button at position x,y                |
|   Return      : None                                        |
*-------------------------------------------------------------*/
void button(int x,int y,char *str,int fg,int bg,char hk)
{
   int len;
   int hk_pos;

   len = strlen(str);
   write_string(x,y,str,fg|(bg<<4));
   hk_pos = strchr(str,hk) - str;
   write_char(x+hk_pos,y,hk,HOTKEYFG|(HOTKEYBG<<4));
   buttonshade(x,y,len);
}
```

ฟังก์ชัน buttonshade();

เป็นฟังก์ชันที่ทำหน้าที่ แสดงเงาของ button

```
/*-------------------------------------------------------------*
|   Function    : buttonshade();                              |
|   Usage       : void buttonshade(int x,int y,int len);      |
|   Description : display shade of button at x,y              |
|   Return      : None                                        |
*-------------------------------------------------------------*/
```

```
void buttonshade(int x, int y, int len)
{
   register int i;

   for(i=1;i<=len;i++)
      write_char(x+i,y+1,223,BLACK|(wkw.bg<<4));

   write_char(x+i-1,y,220,BLACK|(wkw.bg<<4));
}
```

การทำงานของฟังก์ชันในไฟล์ WINDOW.C

ฟังก์ชันในไฟล์ window.c เป็นฟังก์ชันที่ทำหน้าที่เกี่ยวกับ การจัดการ วินโดว์ เพื่อใช้สำหรับ จัดการการใช้งานหน้าจอให้มีประสิทธิภาพมากที่สุด โดย ลักษณะการทำงานของวินโดว์ คือ เป็นการใช้งานบางส่วนของหน้าจอ สำหรับทำงานอื่นที่ไม่เกี่ยวกับงานปัจจุบัน โดยปรากฏขึ้นมาทับหน้าจอปัจจุบัน และเมื่อทำงานเสร็จแล้ว วินโดว์นั้นก็จะหายไป พร้อมทั้ง หน้าจอเดิมที่ถูกทับไว้ก็จะกลับมาทำงานเดิมต่อไป

การทำงานดังที่ได้อธิบายมานี้ จริง ๆ แล้ว วินโดว์ที่ปรากฏขึ้นมาไม่ได้ปรากฏขึ้นมาทับหน้าจอเดิม แต่เป็นการเขียนวินโดว์ทับลงไปในหน่วยความจำเลย โดยมีการเก็บข้อมูลบนหน้าจอเดิมไว้ สำหรับนำกลับมาแสดงผลเมื่อวินโดว์ที่ปรากฏขึ้นมาใหม่นั้น ทำงานเสร็จและหายไปจากหน้าจอ

ฟังก์ชันที่ทำงานเกี่ยวกับวินโดว์ ที่มีในไฟล์ window.c นี้ได้แก่

ฟังก์ชันการสร้างวินโดว์	establish_window();
ฟังก์ชันการลบวินโดว์	delete_window();
ฟังก์ชันการเลื่อนวินโดว์	scroll_window();
ฟังก์ชันการเขียนชื่อของวินโดว์	window_title();
ฟังก์ชันการเลื่อนวินโดว์ขึ้น 1 บรรทัด	upline();
ฟังก์ชันการเลื่อนวินโดว์ลง 1 บรรทัด	downline();
ฟังก์ชันการซ่อนเคอร์เซอร์	hide_cursor();
ฟังก์ชันการเลื่อนตำแหน่งไปบรรทัดแรกของวินโดว์	firstline();
ฟังก์ชันการเลื่อนตำแหน่งไปบรรทัดสุดท้ายของวินโดว์	lastline();
ฟังก์ชันการลบข้อมูลภายในวินโดว์	clear_window();
ฟังก์ชันการเปลี่ยนรูปแบบของเคอร์เซอร์	set_cursor_type();
ฟังก์ชันการแสดงผลอักษรตำแหน่งเทียบกับวินโดว์	win_dstr();

ฟังก์ชันทำการสร้างวินโดว์

ทำหน้าที่สร้างวินโดว์ โดยมีการเก็บค่าตัวแปรที่กำหนดตำแหน่งขอบของวินโดว์ ทั้งมุมบนซ้าย และมุมล่างขวา ขนาดความสูงและกว้างของวินโดว์ และแอททริบิวท์ของหน้าจอและพื้นหลัง (Foreground and Background) เมื่อได้ข้อมูลทั้งหมดแล้ว ก็เรียกฟังก์ชันทำหน้าที่เก็บข้อมูลพื้นที่ที่ต้องถูกเขียนทับไว้ และสุดท้ายก็เรียกฟังก์ชันทำการเขียนกรอบของวินโดว์จากฟังก์ชันในไฟล์ SCR.C

```
/*-------------------------------------------------------------*
|  Function     :  establish_window                           |
|  Usage        :  void establish_window(left,top,right,      |
|                  bottom,foreg,backg,save);                  |
|  Description  :  building window                            |
|                    left,top     :  top left corner          |
|                    right,bottom :  bottom right corner      |
|                    foreg,backg  :  foreground,background    |
|                                    attribute                |
|                    save         :  flag to save background  |
|                                    0  =  not save           |
|                                    1  =  save               |
|  Return       :  None                                       |
*-------------------------------------------------------------*/

void establish_window(left,top,right,bottom,foreg,backg,save)
{
   if(curr_wnd < MAX_WINDOWS)  {
      if(curr_wnd)
         wdo[curr_wnd-1] = wkw;
      setmem(&wkw,sizeof(wkw),0);
      wkw.lf = left;
      wkw.tp = top;
      wkw.rt = right;
      wkw.bt = bottom;
```

```
      wkw.fg = foreg;
      wkw.bg = backg;
      wkw.wd = right+1-left;
      wkw.ht = bottom-top+1;
      if (save)   {
         if ((wkw.wsave=malloc((wkw.ht+1)*(wkw.wd+2)*2)) == NULL)
            return;
         gettext(left,top,right+2,bottom+1,wkw.wsave);
      }
      wdo[curr_wnd++] = wkw;
      block(left,top,right,bottom,foreg|(backg<<4));
      current_window();
   }
}
```

จากฟังก์ชัน establish_window() จะเห็นว่ามีการเรียกฟังก์ชัน current_window() ซึ่งทำหน้าที่เรียกฟังก์ชัน window() ของภาษาซี ทำให้การทำงานกับวินโดว์ปัจจุบันเป็นการสั่งงานเทียบกับวินโดว์ปัจจุบัน

```
/*-------------------------------------------------------------*
|  Function    :  current_window                              |
|  Usage       :  void current_window();                      |
|  Description :  set current window to active                |
|  Return      :  None                                        |
*-------------------------------------------------------------*/
void current_window()
{
   window(wkw.lf,wkw.tp,wkw.rt,wkw.bt);
   hidecursor();
}
```

ฟังก์ชันทำหน้าที่เขียนชื่อวินโดว์

เป็นฟังก์ชันที่ทำหน้าที่หาตำแหน่ง ของกึ่งกลางของกรอบบนของวินโดว์ เพื่อใช้ในการเขียนชื่อของวินโดว์ ด้วยฟังก์ชัน win_dstr();

```
/*-------------------------------------------------------------*
|  Function     :  window_title                               |
|  Usage        :  void window_title(char *ttl,int attrib)    |
|  Description  :  write title string *ttl to current         |
|                  window, attribute specify to attrib        |
|  Return       :  None                                       |
*-------------------------------------------------------------*/
void window_title(char *ttl,int attrib)
{
   win_dstr((wkw.wd-strlen(ttl)) /2,1,ttl,attrib);
}
```

ฟังก์ชันทำหน้าที่ลบวินโดว์

ฟังก์ชันนี้ทำหน้าที่นำข้อมูลจอภาพที่เก็บไว้มาแสดงกลับคืนบนหน้าจอ เป็นการลบวินโดว์เดิม และปรับค่าที่ติดตามลำดับของวินโดว์ให้ลดลง เพื่อเป็นลำดับของวินโดว์ที่อยู่ข้างล่างที่เป็นวินโดว์ที่ทำงานใหม่

```
/*-------------------------------------------------------------*
|  Function     :  delete_window                              |
|  Usage        :  void delete_window();                      |
|  Description  :  remove current window                      |
|  Return       :  None                                       |
*-------------------------------------------------------------*/
void delete_window()
{
   if (curr_wnd)   {
      if (wkw.wsave)   {
         puttext(wkw.lf,wkw.tp,wkw.rt+2,wkw.bt+1,wkw.wsave);
```

```
        free(wkw.wsave);
      }
      setmem(wdo+curr_wnd-1, sizeof (struct wn),0);
      --curr_wnd;
      if (curr_wnd) {
         wkw = wdo[curr_wnd-1];
         current_window();
      }
      else  window(1,1,80,25);
   }
}
```

ฟังก์ชันทำหน้าที่เลื่อนวินโดว์ขึ้นลง

ทำหน้าที่เลื่อนข้อความในวินโดว์ขึ้นหรือลง ขึ้นกับอินพุตที่ให้แก่ฟังก์ชัน คือค่า d ซึ่งมีค่าและความหมายดังนี้

d = 0 : หมายถึง เลื่อนลง
d = 1 : หมายถึง เลื่อนขึ้น

```
/*-------------------------------------------------------------*
|  Function     :  scroll_window                              |
|  Usage        :  void scroll_window(int d)                  |
|  Description  :  scroll the window direction up or down     |
|                        d = 1  :  up                         |
|                        d = 0  : down                        |
|  Return       :  None                                       |
*-------------------------------------------------------------*/
void scroll_window(d)
{
      movetext(wkw.lf+1, wkw.tp+1+d, wkw.rt-1, wkw.bt-2+d,
               wkw.lf+1, wkw.tp+2-d);
}
```

ฟังก์ชันที่ทำหน้าที่เลื่อนขึ้นลงดังกล่าวมาแล้วนั้น นำมาใช้ในฟังก์ชัน upline(); และdownline(); ซึ่งเป็นฟังก์ชันที่เลื่อนข้อความขึ้นลงเช่นกัน แต่มีการตรวจสอบตำแหน่งปัจจุบันด้วย ว่าปัจจุบันอยู่ที่ตำแหน่งใด หากอยู่ที่บรรทัดบนสุด แล้วสั่งเลื่อนขึ้น ก็จะไม่ทำอะไร หรือหากอยู่ที่บรรทัดล่างสุด และสั่งเลื่อนลง ก็จะไม่ทำอะไร เป็นต้น

```
/*-------------------------------------------------------------*
|  Function     :  upline                                     |
|  Usage        :  static void upline();                      |
|  Description  :  move up one line                           |
|  Return       :  None                                       |
*-------------------------------------------------------------*/
static void upline()
{
   if (lineno > 1) {
      if (wkw.wy == 1)   {
         if (wkw.wtop > 1)   {
            --wkw.wtop;
            scroll_window(0);
         }
      }
      else
         --wkw.wy;
   }
   else if (wkw.wlines <= wkw.ht)
      lastline();
}

/*-------------------------------------------------------------*
|  Function     :  downline                                   |
|  Usage        :  static void downline();                    |
|  Description  :  move down one line                         |
|  Return       :  None                                       |
*-------------------------------------------------------------*/
```

```
static void downline()
{
   if (lineno < wkw.wlines)   {
      if (wkw.wy == wkw.ht)   {
         scroll_window(1);
         wkw.wtop++;
      }
      else
         wkw.wy++;
   }
   else  if (wkw.wlines <= wkw.ht)
      firstline();
}
```

ฟังก์ชันเลื่อนเคอร์เซอร์ไปที่บรรทัดบนสุดและล่างสุด

ฟังก์ชัน firstline(); ทำหน้าที่เช่นเดียวกับการเลื่อนวินโดว์ขึ้นลง เพียงแต่เป็นการเลื่อนไปที่ตำแหน่งบรรทัดบนสุดเลยเท่านั้น ส่วนฟังก์ชัน lastline(); ก็เช่นเดียวกัน เป็นการเลื่อนวินโดว์ไปที่ตำแหน่งบรรทัดสุดท้ายเลย

```
/*-------------------------------------------------------------*
|  Function     :  firstline                                  |
|  Usage        :  static void firstline();                   |
|  Description  :  move to the first  line                    |
|  Return       :  None                                       |
*-------------------------------------------------------------*/
static void firstline()
{
   wkw.wtop = wkw.wy = 1;
}
```

```
/*-------------------------------------------------------------*
|  Function     :  lastline                                   |
|  Usage        :  static void lastline();                    |
|  Description  :  move to the last line                      |
|  Return       :  None                                       |
*-------------------------------------------------------------*/
static void lastline()
{
   wkw.wtop = wkw.wlines - (wkw.ht-1);
   if (wkw.wtop <1)
      wkw.wtop = 1;
   wkw.wy = wkw.ht;
   if (wkw.wy > wkw.wlines)
      wkw.wy = wkw.wlines;
}
```

ฟังก์ชันทำหน้าที่ซ่อนเคอร์เซอร์

ฟังก์ชัน hidecursor(); ทำหน้าที่ซ่อนเคอร์เซอร์ ไม่ให้ปรากฏแก่ผู้ใช้ เมื่อต้องการติดต่อกับผู้ใช้ผ่านทางเมนู เป็นต้น การซ่อนเคอร์เซอร์ ทำได้โดยการใช้ฟังก์ชันของ BIOS กำหนดตำแหน่งของเคอร์เซอร์ ให้เกินตำแหน่งที่เป็นไปได้

```
/*-------------------------------------------------------------*
|  Function     :  hidecursor                                 |
|  Usage        :  void hidecursor();                         |
|  Description  :  use BIOS to hide the cursor                |
|  Return       :  None                                       |
*-------------------------------------------------------------*/
void hidecursor()
{
   rg.h.ah = 2;
   rg.x.dx = 0x1900;
   rg.h.bh = 0;
   int86(0x10,&rg,&rg);
}
```

ฟังก์ชันทำหน้าที่เปลี่ยนรูปแบบของเคอร์เซอร์

ฟังก์ชัน set_cursor_type(); ทำหน้าที่เปลี่ยนขนาดของเคอร์เซอร์ ซึ่งทำให้ผู้ใช้สามารถรู้โหมดการทำงานของโปรแกรมได้ เช่น หากเป็นเคอร์เซอร์ขนาดใหญ่เป็นการใช้งานในโหมดการพิมพ์แทรก หรือเคอร์เซอร์ขนาดปกติ เป็นการพิมพ์ในโหมดการพิมพ์ทับ การทำงานของฟังก์ชัน ก็เพียงแต่ผ่านค่าขนาดของเคอร์เซอร์ที่ต้องการ ให้แก่ฟังก์ชัน ฟังก์ชันก็จะเรียกอินเตอร์รัพท์ของ BIOS เพื่อกำหนดขนาดของเคอร์เซอร์ใหม่

```
/*-------------------------------------------------------------*
|  Function     :  set_cursor_type                            |
|  Usage        :  void set_cursor_type(unsigned t)           |
|  Description  :  use BIOS to set the cursor type            |
|  Return       :  None                                       |
*-------------------------------------------------------------*/
void set_cursor_type(unsigned t)
{
   rg.x.ax = 0x0100;
   rg.x.bx = 0;
   rg.x.cx = t;
   int86(0x10,&rg,&rg);
}
```

ฟังก์ชันทำหน้าที่รับอินพุตจากคีย์บอร์ด

ทำหน้าที่เรียกฟังก์ชันของ BIOS เพื่อทำหน้าที่รับอินพุตจากคีย์บอร์ด นำมาปรับเปลี่ยน เพื่อให้ได้รูปแบบของค่าที่ได้จากการกดคีย์บอร์ดที่เหมาะสมกับการใช้งาน

```
/*-------------------------------------------------------------*
|  Function     :  getkey                                     |
|  Usage        :  int getkey();                              |
|  Description  :  read the keyboard                          |
|  Return       :  keyboard read                              |
*-------------------------------------------------------------*/
```

```
int getkey()
{
   int c;

   if ((c = getch()) == 0 )
      c = getch() | 128;
   if (c == helpkey && helpfunc) {
      (*helpfunc)();
      c = getkey();
   }
   return c;
}
```

ฟังก์ชันเขียนอักษรตำแหน่งเทียบกับวินโดว์

ฟังก์ชัน win_dstr(); เป็นฟังก์ชันทำหน้าที่เขียนตัวอักษร ที่ตำแหน่งที่กำหนด โดยตำแหน่งที่กำหนดเป็นตำแหน่งที่คิดเทียบกับตำแหน่งกรอบของวินโดว์ ไม่ใช่กรอบของจอภาพทั้งหมด ดังนั้น จึงมีประโยชน์ต่อการใช้งานวินโดว์ ที่ไม่ต้องคอยห่วงเกี่ยวกับตำแหน่งที่จะต้องคิดเทียบกับจอภาพ

```
/*-------------------------------------------------------------*
|  Function     :  win_dstr                                   |
|  Usage        :  void win_dstr(int x,int y,char *str,       |
|                                int attr);                   |
|  Description  :  write string  position according to        |
|                  active window                              |
|                  top left corner is  1,1                    |
|  Return value :  None                                       |
*-------------------------------------------------------------*/
void win_dstr(x,y,str,attr)
int x,y;
char *str;
int attr;
```

```
{
   write_string(wkw.lf+x-1,wkw.tp+y-1,str,attr);
}
```

ฟังก์ชันทำหน้าที่ลบข้อมูลในวินโดว์

ฟังก์ชัน clear_window(); เป็นฟังก์ชันที่ทำหน้าที่ คำนวนบริเวณที่อยู่ภายในวินโดว์ และ ใช้ฟังก์ชัน fill(); เพื่อเติมตัวอักษรว่าง ๆ ในบริเวณดังกล่าว เป็นการลบข้อมูลในวินโดว์

```
/*-------------------------------------------------------------*
|  Function     :  clear_window                               |
|  Usage        :  void clear_window                          |
|  Description  :  clear the current window                   |
|  Return       :  None                                       |
*-------------------------------------------------------------*/
void clear_window()
{
   fill(wkw.lf+1,wkw.tp+1,wkw.rt-1,wkw.bt-1,' ',wkw.fg|(wkw.bg<<4));
}
```

การทำงานของฟังก์ชันภายในไฟล์ ENTRY.C

ในไฟล์ ENTRY.C เป็นฟังก์ชันที่จัดการเกี่ยวกับ การรับค่าคีย์เพื่อเก็บข้อมูลเป็น field ภายใน record ใด ๆ และสามารถมี field ที่เป็น button เพื่อรับคีย์ ENTER หรือ hotkey เพื่อทำงานตามที่กำหนดให้ และส่วนที่เป็นการแสดง radio box ซึ่งยังแบ่งเป็น radio box ที่มีค่าเดียว และ radio box ที่มีค่าได้หลายค่า

การทำงานแบ่งเป็น 3 ส่วนดังนี้

1. ส่วนที่จัดการกับการรับค่าตัวอักษรเป็น field ของ record
2. ส่วนที่จัดการกับ radio box
3. ส่วนที่จัดการกับ button

field แต่ละชนิดจะมี field type แตกต่างกัน คือ

character field มี field type เป็น 1
radio box แบบค่าเดียว มี field type เป็น 2
radio box แบบหลายค่า มี field type เป็น 3
button มี field type เป็น 4

1. ส่วนที่จัดการกับ character field

เป็นส่วนที่จัดการรับค่าตัวอักษร และตรวจสอบคีย์ว่าหมด field หรือยัง เพื่อจะได้ขึ้นไปยัง field ถัดไป และจัดการตัวอักษรในลักษณะ word processing ได้ แก่ฟังก์ชันดังต่อไปนี้

```
disp_field(FIELD *, char *, char *);
read_field(int);
home_cursor(void);
backspace(void);
end_cursor(void);
forward(void);
fore_word(void);
back_word(void);
```

```
delete_char(void);
delete_word(void);
```

2. ส่วนจัดการกับ radio box

เป็นส่วนที่จัดการแสดงผลของ radio box เพื่อใช้ในลักษณะของการเลือกกำหนดค่าให้แต่ละ field มีค่าเป็น 1 หรือ 0 ลักษณะของ radio box มี 2 ลักษณะ คือ

1. เป็น radio box ที่เลือกค่าได้ค่าเดียว
2. เป็น radio box ที่เลือกค่าได้หลายค่า

ตัวอย่างของ radio box คือ

```
service_type
[x] telphone_back
[ ] voice mail box
[x] general information
```

เครื่องหมาย x ในแต่ละช่องของ radio box หมายถึง เลือกให้มีค่าเป็น 1 ถ้าไม่มีเครื่องหมายนี้ แสดงว่า ไม่เลือก field นี้

ฟังก์ชันที่เกี่ยวกับการจัดการ radio box ได้แก่

```
init_rad(RADIO *rad,int init);
next_rad(void);
prev_rad(void);
first_rad(void);
last_rad(void);
mark_rad(void);
disp_rad(void);
read_rad(int c);
rad_desc(RADIO *radin);
```

3. ส่วนจัดการกับ button

เป็นส่วนที่รอรับค่าคีย์ เพื่อรอรับ คีย์ ENTER หรือ hotkey เพื่อเรียกฟังก์ชันที่กำหนดให้แต่ละ button มีฟังก์ชันที่ใช้แสดงตัว button ฟังก์ชันแสดงลักษณะการกด button และฟังก์ชันที่เก็บค่าของ button เดิมเมื่อมีการเรียกฟังก์ชัน data_entry ในฟังก์ชันใหม่ซ้ำอีกครั้ง

ฟังก์ชันที่เกี่ยวกับ button ได้แก่

```
read_bt(int c);
bt_press(void);
active_bt(void);
button_call(int  *c);
is_hotkey(char c);
```

ฟังก์ชันทั่วไปที่ทำงานใน ENTRY.C

ฟังก์ชัน data_entry

เป็นฟังก์ชันหลักที่ทำหน้าที่เรียก และจัดการฟังก์ชันอื่น ๆ ที่เกี่ยวข้องทั้งหมด อันได้แก่ ฟังก์ชันเกี่ยวกับ character field, radio box และ button มีการเตรียมสถานะเริ่มต้นของแต่ละ field มีการตรวจสอบคีย์ที่ return กลับมาจากแต่ละ field ว่าจะต้องเปลี่ยนไปยัง field ถัดไปหรือยัง หรือเป็นคีย์ ESC หรือไม่ เหล่านี้เป็นต้น

ฟังก์ชัน data_entry จะเก็บหน้าจอส่วนบรรทัดสุดท้ายไว้ เพื่อใช้สำหรับแสดงข้อความ help ของแต่ละ field และเมื่อสิ้นสุดการทำงานของ data_entry แล้ว จะคืนหน้าจอส่วนนี้กลับเป็นตัวอักษรเดิม

parameter ตัวแรกคือ pointer ชี้ไปยัง template ที่เป็นโครงสร้างของ field แต่ละ field ซึ่งแสดงไว้ในไฟล์ entry.h แล้ว ส่วน parameter ตัวที่สอง เป็น flag เพื่อบอกว่า field แต่ละ field จะต้องมีการ initial ให้มีค่าเป็นช่องว่าง สำหรับ character field และ ช่องว่างสำหรับ radio box หรือไม่ และ parameter ตัวที่ 3 เป็นตัวแปรที่กำหนดให้ฟังก์ชันเริ่มทำการรอรับค่าคีย์ที่ field ใด

```
int data_entry(FIELD *fldin, int init, int firstfld)
{
   int exitcode = 0, done = FALSE;
   char *help_save;

   textcolor(FIELDFG);
   textbackground(FIELDBG);

   /* save help area at screen bottom */
   help_save = (char *) malloc(160);
   gettext(1,25,80,25,help_save);

   write_string(1,25," Help | ",0x4f);
   insert_line();
   fhead = ftail = fld = fldin;
   while (ftail->frow)
      ftail++;
   while (firstfld-- && fld != ftail)
      fld++;
   --fld;
   if (init)
      init_template();
   rad_template();
   field_tally();

   /* ----- collect data from keyboard into screen --------- */
   while (done == FALSE) {
      data_value(fld);
      exitcode = inp_field(0);
      field_tally();
      switch (exitcode) {
         case DN        :
         case '\r'      :
```

```
        case '\t'       :
        case CTRL_FWD   :
        case FWD        : done = (ftail == fhead+1);
                          fld++;
                          if (fld == ftail)
                             fld = fhead;
                          break;
        case CTRL_BS    :
        case UP         : if (fld == fhead)
                             fld = ftail;
                          --fld;
                          break;
        case CTRL_HOME  :
                          fld = fhead;
                          break;
        case CTRL_END   :
                          fld = ftail-1;
                          break;
        default         :
             if((exitcode == ESC) || (exitcode == CANCEL)) {
                puttext(1,25,80,25,help_save);
                return CANCEL;
             }
             else
                if (exitcode == OK)
                   done = TRUE;

             break;
      }
   }
   puttext(1,25,80,25,help_save);  /* restore help area */
   return OK;
}
```

ฟังก์ชัน data_value();

เป็นฟังก์ชันที่ทำหน้าที่ แสดงผลของ field แต่ละ field โดยตรวจสอบ field type ว่าเป็น type ชนิดใด และแยกไปทำการแสดงผล field ตาม type ที่ได้ โดยถ้าหาก type เป็น 1 แสดงว่าเป็น character field จะเรียกฟังก์ชัน disp_field(); field type เป็น 2 หรือ 3 แสดงว่า เป็น radio box จะเรียกฟังก์ชัน disp_rad(); และถ้าเป็น 4 แสดงว่าเป็น button จะเรียกฟังก์ชัน button เพื่อแสดงผล

```
/*-------------------------------------------------------------*
|   Function    :  data_value();                              |
|   Usage       :  void data_value(FIELD *fldv);              |
|   Description :  display the data value in a field          |
|   Return      :  None                                       |
*-------------------------------------------------------------*/
static void data_value(FIELD *fldv)
{
   int helplen;
   char blank[80];

   switch ( fldv->type ) {
      case 1 :
         fldv->fx = 1;
         disp_field(fldv, fldv->fvalue.fbuff, fldv->mask.fbuff);
         break;
      case 2 :
      case 3 : init_rad(fldv->fvalue.value,0);
               disp_rad();
               break;
      case 4 :
         button(fldv->fcol+wkw.lf,fldv->frow+wkw.tp,
                fldv->fvalue.fbuff,BUTTONFG,BUTTONBG,fldv->fx);
         break;
```

```
    }

    memset(blank,32,80);
    helplen = strlen(fld->fhelp);
    write_string(9,25,fld->fhelp,0x4f);
    write_string(9+helplen,25,blank+8+helplen,0x4f);
}
```

ฟังก์ชัน field_tally();

เป็นฟังก์ชันที่ทำหน้าที่แสดงผลทุก field เรียงจาก field แรก ไปยัง field สุดท้าย โดยเรียกฟังก์ชัน data_value(); อีกทีหนึ่ง

```
/*-------------------------------------------------------------*
|   Function    :  field_tally();                             |
|   Usage       :  void field_tally();                        |
|   Description :  display all the fields in a window         |
|   Return      :  None                                       |
*-------------------------------------------------------------*/
void field_tally()
{
   FIELD *fldt;

   fldt = fhead;
   while (fldt != ftail)  {
      data_value(fldt);
      fldt++;
   }
}
```

ฟังก์ชันที่จัดการเกี่ยวกับ button

ฟังก์ชัน read_bt(c);

เป็นฟังก์ชันที่รอค่าคีย์ที่กด ขณะที่ active field เป็น button ซึ่งได้แก่ คีย์ ENTER หรือ hotkey ของ button แต่ละอัน ฟังก์ชันเริ่มด้วยการวนลูปรอรับค่าคีย์ และแสดง current active button โดยเรียกฟังก์ชัน active_bt(); คีย์ที่กดเข้ามาจะถูกตรวจสอบว่าเป็นคีย์ TAB หรือ ENTER หรือเป็น hotkey หรือไม่ ถ้าใช่ จะมีการเรียกฟังก์ชันของ button นั้น ด้วยฟังก์ชัน button_call(&c) โดยก่อนเรียกจะเก็บค่าของ current field ทั้ง head field, tail field และ current field ไว้ เพื่อที่จะได้นำกลับมาทำงานต่อไปได้หลังจากเสร็จสิ้นการทำงานตามฟังก์ชันของ button แล้ว

ถ้าหากคีย์ที่กดไม่ใช่ทั้ง TAB หรือ ENTER หรือ hotkey ก็จะวนกลับไปรับค่าต่อไป

```
/*-------------------------------------------------------------*
|   Function    : read_bt(c)                                  |
|   Usage       : int read_bt(c);                             |
|   Description : wait for tab to next button or hot key      |
|   Return      : key press                                   |
*-------------------------------------------------------------*/
static int read_bt(c)
{
   int done = FALSE;
   int but_no;

   while ( TRUE ) {
      hidecursor();
      active_bt();        /* display current active button */
      c = getkey();
      switch(c) {
         case '\r' :
         case '\n' : done = button_call(&c);
                     /* Enter pressed; call function */
```

```
                    break;

        default   : if ((but_no = is_hotkey(c)) != -1) {
                        /* -----Hot key pressed ------*/
                        /* set active field */
                       fld = fhead+but_no;
                       field_tally();
                       active_bt();  /* display active button */
/* call function */    done = button_call(&c);
                       break;
                    }
                    if ((c=='\t') || (c== ESC))
                       done = TRUE;
                    break;
      }
      if (done)
         break;
   }
   return c;
}
```

ฟังก์ชัน is_hotkey();

เป็นฟังก์ชันที่ทำหน้าที่ตรวจสอบคีย์ที่กด ว่าเป็น hotkey หรือไม่ ถ้าเป็น hotkey จะ return หมายเลขของ field กลับมา แต่ถ้าไม่ใช่ hotkey จะ return ค่า -1

```
/*-------------------------------------------------------------*
|   Function    :  is_hotkey();                               |
|   Usage       :  int is_hotkey(char c);                     |
|   Description :  Check whether key pressed is hot key       |
|   Return      :  -1         : not hot key                   |
|                  field no.  : in case of hot key            |
*-------------------------------------------------------------*/
```

```
static int is_hotkey(char c)
{
   FIELD *fd;
   int i;

   fd = fhead; i = 0;
   while(fd != ftail) {
      if(fd->type == 4) {          /* button field */
         if(c == tolower(fd->fx))
            return i;              /* return field no. */
      }
      fd++; i++;
   }
   return -1;                      /* not hot key */
}
```

ฟังก์ชัน button_call();

เป็นฟังก์ชันที่ทำหน้าที่เก็บสถานะเดิมของ field ต่าง ๆ เพื่อทำการเรียกฟังก์ชัน ซึ่งฟังก์ชันอาจจะมีการเรียกใช้ฟังก์ชัน data_entry อีกครั้ง ทำให้มีการทำงานกับโครงสร้าง FIELD ใหม่ เมื่อเสร็จสิ้นการทำงานในฟังก์ชันนั้นแล้ว และกลับไปยัง data_entry เดิม จะต้องนำ field เดิมมาทำงานต่อไปได้ เช่นเดียวกับการจัดการ stack ของการเรียกฟังก์ชันในภาษา assembly ที่ใช้การ push และ pop

```
/*-------------------------------------------------------------*
|   Function    :  button_call();                             |
|   Usage       :  int button_call(int *c);                   |
|   Description :  call function of current button            |
|   Return      :  return code of function called             |
*-------------------------------------------------------------*/
static int button_call(int *c)
{
   FIELD *oldfhead;
```

```
FIELD *oldftail;
FIELD *oldfld;

RADIO *oldrhead;
RADIO *oldrtail;
RADIO *oldwrad;
int temp;/* store button function returned value */
int done;

bt_press();/* display button being pressed */

oldfhead = fhead;/* store old f and r */
oldftail = ftail;
oldfld = fld;
oldrhead = rhead;
oldrtail = rtail;
oldwrad = wrad;

/* call function of current button */
temp = (*fld->mask.func)();
if (temp == OK) {
   done = TRUE;
   *c = OK;
}
if (temp == CANCEL) {
   done = TRUE;
   *c = ESC;
}
if (temp == '\t') {
   done = TRUE;
}

fhead = oldfhead;                    /* retrieve old f and r */
```

```
    ftail = oldftail;
    fld =  oldfld;
    rhead = oldrhead;
    rtail = oldrtail;
    wrad = oldwrad;
    return done;

}
```

ฟังก์ชัน bt_press();

เป็นฟังก์ชันที่ทำหน้าที่ แสดง button ในลักษณะของการถูกกด คือ จะไม่มีเงาแสดงให้เห็น เป็นเวลาช่วงสั้น ๆ

```
/*-------------------------------------------------------------*
|   Function    : bt_press();                                 |
|   Usage       : void bt_press(void);                        |
|   Description : display current button being pressed        |
|   Return      : None                                        |
*-------------------------------------------------------------*/
static void bt_press(void)
{
    int x,y;
    int len;
    long i;

    len = strlen(fld->fvalue.fbuff);

    x = fld->fcol+wkw.lf;
    y = fld->frow+wkw.tp;

    fill(x,y,x+len+1,y+1,' ',wkw.fg|(wkw.bg<<4));
    write_string(x+1,y,fld->fvalue.fbuff,ACTBUTFG|(ACTBUTBG<<4));
    for(i=0;i<20000;i++) {    /* delay for button pressed */
```

```
    }
    active_bt();
}
```

ฟังก์ชัน active_bt();

เป็นฟังก์ชันที่แสดง current active button ซึ่งจะแสดงได้ด้วย สีของ button ที่แตกต่างจาก button อื่น พร้อมด้วยมี เครื่องหมายหัวลูกศรทางซ้ายขวาของ button

```
/*-------------------------------------------------------------*
|   Function    :  active_bt();                               |
|   Usage       :  void active_bt(void);                      |
|   Description :  display current button in active mode      |
|   Return      :  None                                       |
*-------------------------------------------------------------*/
static void active_bt(void)
{
    int len;
    int x,y;

    len = strlen(fld->fvalue.fbuff);
    x = fld->fcol+wkw.lf;
    y = fld->frow+wkw.tp;
    write_string(x,y,fld->fvalue.fbuff,ACTBUTFG|(ACTBUTBG<<4));
    buttonshade(x,y,len);
    write_char(x,y,16,BLACK | (ACTBUTBG<<4));  /* left  arrow */
    /* right arrow */
    write_char(x+len-1,y,17,BLACK | (ACTBUTBG<<4));
}
```

กลุ่มฟังก์ชันที่จัดการเกี่ยวกับ radio box

ฟังก์ชัน rad_to_code();

เป็นฟังก์ชันที่ทำหน้าที่เปลี่ยน ข้อมูลที่มีโครงสร้างเป็น RADIO เพื่อให้เป็น code ขนาด 2 ไบท์ โดยมีรูปแบบดังต่อไปนี้

ชื่อของ radiobox
[ ] field แรก บิทที่ 1
[ ] field ที่สอง บิทที่ 2
[ ] field ที่สาม บิทที่ 3
. .
. .
. .
. .

หาก field ใดถูก mark ด้วยเครื่องหมาย " x " บิทที่ลำดับเดียวกัน จะมีค่าเป็น 1 และ หาก field ใดไม่ได้ถูก mark บิทที่ลำดับเดียวกัน จะมีค่าเป็น 0

```
/*-------------------------------------------------------------*
|   Function    :  rad_to_code();                             |
|   Usage       :  int rad_to_code(RADIO *rad);               |
|   Description :  convert data in radio form to pack code    |
|   Return      :  packed code                                |
*-------------------------------------------------------------*/
int rad_to_code(RADIO *rad)
{
   int temp,i;

   temp = 0;
   for(i=0;rad->row;i++,rad++)
      temp |= (rad->value)<<i;
   return temp;
}
```

ฟังก์ชัน code_to_rad();

เป็นฟังก์ชันที่ทำหน้าที่ ตรงกันข้ามกับฟังก์ชัน rad_to_code(); คือจะรับค่า code ซึ่งเป็นข้อมูลแบบ integer เข้าไป เพื่อเปลี่ยนให้ได้ข้อมูลแบบ radio box

```
/*-------------------------------------------------------------*
|   Function    :  code_to_rad();                             |
|   Usage       :  void code_to_rad(int int_in,RADIO *rad);   |
|   Description :  convert packed code to data in radio form  |
|   Return      :  None                                       |
*-------------------------------------------------------------*/
void code_to_rad(int int_in,RADIO *rad)
{
   int i;

   for(i=0;rad->row;i++,rad++)
      rad->value = (int_in>>i) & 1;
}
```

ฟังก์ชัน init_rad();

เป็นฟังก์ชันทำหน้าที่ initial ค่าใน radio box ให้มีค่าเป็น 0 เมื่อกำหนดให้ parameter init มีค่าเป็น 1 หรือให้มีค่าเดิม หากค่าของ init มีค่า 0

```
/*-------------------------------------------------------------*
|   Function    : init_rad();                                 |
|   Usage       : void init_rad(RADIO *rad,int init);         |
|   Description : set rhead, rtail and wrad to group of radio |
|                   box of rad                                |
|                   if init = 1, radio value is set to 0      |
|   Return      : None                                        |
*-------------------------------------------------------------*/
```

```
static void init_rad(RADIO *rad,int init)
{
   rhead = rtail = wrad = rad;
   while ( rtail->row )
      rtail++;          /* set rtail */

   if(init) {
      while ( rad->row ) {
         rad->value = 0;
         rad++;
      }
   }
}
```

ฟังก์ชัน rad_desc();

เป็นฟังก์ชันที่ใช้แสดงข้อความที่อธิบายความหมายของ แต่ละ field ใน radio box ตัวอย่างและตำแหน่งของข้อความเหล่านี้คือ

```
service_type
[x] telphone_back
[ ] voice mail box
[x] general information
```

service_type เป็นชื่อของ radio box อันนี้ ส่วน telphone_back, voice mail box และ general information เป็นข้อความที่อธิบาย field ของ radio box แต่ละ field

```
/*-------------------------------------------------------------*
!   Function    :  rad_desc();                                  !
!   Usage       :  void rad_desc(RADIO *radin);                 !
!   Description :  display radio box description                !
!   Return      :  None                                         !
*-------------------------------------------------------------*/
```

```
static void rad_desc(RADIO *radin)
{
   RADIO *temp;

   temp = radin;
   while(temp->row != NULL) {
      win_dstr(temp->col+5,temp->row+1,temp->rad_des,
               RADIOFG|(RADIOBG<<4));
      temp++;
   }
}
```

ฟังก์ชันที่จัดการกับตำแหน่งของ radio box field

ได้แก่ฟังก์ชัน

next_rad();
prev_rad();
first_rad();
last_rad();

ทำหน้าที่กำหนดตำแหน่งปัจจุบันของ field ของ radio box ว่าทำงานกับ field ใด

next_rad(); เลื่อน current field ไปยัง field ต่อไป
prev_rad(); เลื่อน current field ไปยัง field ก่อน
first_rad(); เลื่อน current field ไปยังต้น field
last_rad(); เลื่อน current field ไปยังท้าย field

```
/*-------------------------------------------------------------*
|   Function    :  next_rad();                                |
|   Usage       :  void next_rad(void);                       |
|   Description :  set working radio to next box              |
|                  if next radio is tail; set to head         |
|   Return      :  None                                       |
*-------------------------------------------------------------*/
```

```
static void next_rad(void)
{
   wrad++;
   if (wrad == rtail)
      wrad = rhead;
}


/*-------------------------------------------------------------*
|   Function    : prev_rad();                                 |
|   Usage       : void prev_rad(void);                        |
|   Description : set working radio to previous box           |
|                 if previos is head; set to tail             |
|   Return      : None                                        |
*-------------------------------------------------------------*/
static void prev_rad(void)
{
   if (wrad == rhead)
      wrad = rtail-1;
   else
      wrad--;
}

/*-------------------------------------------------------------*
|   Function    :  first_rad();                               |
|   Usage       :  void first_rad(void);                      |
|   Description :  set working radio to first box             |
|   Return      :  None                                       |
*-------------------------------------------------------------*/
static void first_rad(void)
{
   wrad = rhead;
}
```

```
/*-------------------------------------------------------------*
|   Function    : last_rad();                                 |
|   Usage       : void last_rad(void);                        |
|   Description : set working radio to last box               |
|   Return      : None                                        |
*-------------------------------------------------------------*/
static void last_rad(void)
{
   wrad = rtail-1;
}
```

ฟังก์ชัน mark_rad();

เป็นฟังก์ชันที่กำหนดค่าให้กับ field ของ radio box โดยพิจารณาตาม field type ของ radio box ด้วย คือ

หาก radio box นี้มี type เป็น 2 แสดงว่า แต่ละ field ของ radio box จะมีค่า 1 ได้เพียง field เดียวเท่านั้น ดังนั้น ในการกำหนดค่า จะมีการเคลียร์ค่าของแต่ละ field ให้มีค่าเป็น 0 ก่อน แล้วจึงกำหนดให้ current field มีค่าเป็น 1

หาก radio box มี type เป็น 3 แสดงว่าเป็น radio box แบบหลายค่า ดังนั้นการ mark จะเป็นการ mark หรือ unmark กรณีใดกรณีหนึ่ง ดังนั้นการทำงานจะเป็นการทำ XOR กับข้อมูลเดิมด้วย 1 คือ ถ้าเดิมมีค่าเป็น 1 เมื่อ unmark จะมีค่าเป็น 0 ถ้าเดิมมีค่าเป็น 0 เมื่อ mark แล้วจะมีค่าเป็น 1

```
/*-------------------------------------------------------------*
|   Function    : mark_rad();                                 |
|   Usage       : void mark_rad(void);                        |
|   Description : Set value in radio according to radio type  |
|   Return      : None                                        |
*-------------------------------------------------------------*/
static void mark_rad(void)
{
```

```
   RADIO *temp;

   temp = wrad;
   switch(fld->type) {
      case 2 :                          /* only one value radio */
         temp = rhead;
         while (temp != rtail) {
            temp->value = 0;
            temp++;
         }

         wrad->value = 1;               /* set working radio */
         break;
      case 3 :                          /* multi value radio */
         wrad->value ^= 1;              /* toggle value */
         break;
   }
}
```

ฟังก์ชัน disp_rad();

เป็นฟังก์ชันที่ทำหน้าที่ แสดงเครื่องหมาย " x " เพื่อแสดงว่า field นี้ถูกเลือกให้มีค่าเป็น 1 หรือหาก field มีค่าเป็น 0 ก็จะไม่แสดงเครื่องหมาย ใดๆ

```
/*-------------------------------------------------------------*
|   Function    :  disp_rad();                                |
|   Usage       :  void disp_rad();                           |
|   Description :  display value of each radio box            |
|   Return      :  None                                       |
*-------------------------------------------------------------*/
static void disp_rad()
{
   RADIO *temp;
```

```
    temp = rhead;
    while ( temp != rtail ) {
       if ( temp->value == 1 ) {
          write_char(temp->col+wkw.lf+1,temp->row+wkw.tp,'X',
          RADIOFG|(RADIOBG<<4));
          temp++;
       }
       else {
          write_char(temp->col+wkw.lf+1,temp->row+wkw.tp,' ',
          RADIOFG|(RADIOBG<<4));
          temp++;
       }
    }
}
```

ฟังก์ชัน read_rad();

เป็นฟังก์ชันที่รอรับค่าคีย์ ในขณะที่ทำงานใน field ที่เป็น radio box โดยคีย์ที่กด จะเป็นคีย์ SPACE เพื่อ mark field หรือ คีย์ TAB เพื่อเปลี่ยนไปยัง field ถัดไป และคีย์ลูกศรขึ้นลง เพื่อเปลี่ยนไปยัง field ภายในของ radio box

```
/*-------------------------------------------------------------*
|   Function    :  read_rad();                                |
|   Usage       :  int read_rad(int c);                       |
|   Description :  wait for marking radio                     |
|   Return      :  ESC or '\t' for next field                 |
*-------------------------------------------------------------*/
static int read_rad(int c)
{
   int done = FALSE;

   first_rad();
   while( TRUE ) {
```

```
        gotoxy(wrad->col+2,wrad->row+1);
        c = getkey();
        switch(c) {
           case DN   :
           case FWD  : next_rad();
                       break;
           case UP   :
           case BS   : prev_rad();
                       break;
           case HOME : first_rad();
                       break;
           case END  : last_rad();
                       break;
           case ' '  : mark_rad();
                       break;
           default   : if ((c == '\t') || (c == ESC) ) {
                          done = TRUE;
                       }
                       break;
        }
        if ( done )
           break;
        disp_rad();
     }
     return c;
}
```

ฟังก์ชัน rad_template();

เป็นฟังก์ชันที่ทำหน้าที่เตรียมการแสดงผล radio box บนจอภาพ อันได้แก่ ชื่อของ radio box เครื่องหมาย [ ] ข้อความที่อธิบายแต่ละ field ของ radio box และแสดงสีพื้นที่แตกต่างจากพื้นเดิม เพื่อให้เห็นความแตกต่างระหว่าง radio box กับ field อื่น ๆ

```
/*-------------------------------------------------------------*
|   Function    :  rad_template();                            |
|   Usage       :  void rad_template();                       |
|   Description :  initial template for radio box             |
|   Return      :  None                                       |
*-------------------------------------------------------------*/
void rad_template()
{
   FIELD *fldt;
   RADIO *temp;
   int radlen;
   int radnum;
   int templen;

   fldt = fhead;
   while(fldt != ftail) {
      if((fldt->type == 2) || (fldt->type == 3)) {
         temp = fldt->fvalue.value;
         radlen = 0;
         radnum = 0;
         while(temp->row != 0) {
            if ((templen = strlen(temp->rad_des)) > radlen)
               radlen = templen;
            radnum++;
            temp++;
         }
         fill(fldt->fcol+wkw.lf-1,fldt->frow+wkw.tp+1,
              fldt->fcol+wkw.lf+radlen+5,fldt->frow+
              wkw.tp+radnum,' ',RADIOFG|(RADIOBG<<4));
         win_dstr(fldt->fcol,fldt->frow+1,fldt->mask.fbuff,
                  ENTRYFG|(ENTRYBG<<4));
         rad_desc(fldt->fvalue.value);
         temp = fldt->fvalue.value;
```

```
            while(temp->row != 0) {
                write_char(temp->col+wkw.lf,temp->row+wkw.tp,
                    '[',RADIOFG|(RADIOBG<<4));
                write_char(temp->col+2+wkw.lf,temp->row+wkw.tp,
                    ']',RADIOFG|(RADIOBG<<4));
                temp++;
            }
        }
        fldt++;
    }
}
```

กลุ่มฟังก์ชันที่ทำหน้าที่จัดการ character field

ฟังก์ชัน disp_field();

เป็นฟังก์ชันแสดงตัวอักษรของ field ปัจจุบัน ที่เป็น character field

```
/*-------------------------------------------------------------*
|   Function    :  disp_field();                              |
|   Usage       :  void disp_field(FIELD *fldv,char *bf,      |
|                          char *msk);                        |
|   Description :  display a data field                       |
|   Return      :  None                                       |
*-------------------------------------------------------------*/
static void disp_field(FIELD *fldv, char *bf, char *msk)
{
    char cl[80], *cp = cl;

    while (*msk) {
        *cp++ = (*msk != FIELDCHAR ? *msk : *bf++);
        msk++;
    }
```

```
   *cp = '\0';
   win_dstr(fldv->fx + fldv->fcol -1,fldv->frow, cl,
            FIELDFG|(FIELDBG<<4));
}
```

ฟังก์ชัน init_template();

เป็นฟังก์ชันที่ทำหน้าที่ initial แต่ละ field ให้มีสถานะเริ่มต้น ตามค่า field type ของแต่ละ field คือถ้าเป็น 1 (character field) ก็จะเคลียร์ค่าตัวอักษร ให้เป็นตัวอักษร ' ' (SPACE) ทั้งหมด ถ้ามีค่าเป็น 2 หรือ 3 (radio box) จะเคลียร์ค่าให้เป็น 0 หรือถ้าเป็น 4 (button) จะไม่ทำงานอะไร

```
/*-------------------------------------------------------------*
|   Function    :  init_template();                           |
|   Usage       :  void init_template();                      |
|   Description :  clear a template to all blanks             |
|   Return      :  None                                       |
*-------------------------------------------------------------*/
void init_template()
{
   FIELD *fldc;
   char *bf, *msk;

   fldc = fhead;
   while (fldc != ftail) {
      switch ( fldc->type ) {
         case 1 :                    /* character field */
                  bf = fldc->fvalue.fbuff;
                  msk = fldc->mask.fbuff;
                  while (*msk)  {
                     if (*msk == FIELDCHAR)
                        *bf++ = ' ';
```

```
                msk++;
            }
            fldc++;
            break;
        case 2 :                    /* radio box */
        case 3 :
            init_rad(fldc->fvalue.value,0);
            fldc++;
            break;
        case 4 :                    /* case of button */
            fldc++;                 /* do nothing */
            break;
    }
  }
}
```

ฟังก์ชัน read_field();

เป็นฟังก์ชันที่ทำหน้าที่รอรับคีย์ ในช่วงที่เป็น character field คือคีย์ที่เป็นตัวอักษรทั้งหมด รวมทั้ง function key ทุกอย่างที่กำหนดไว้ เช่น ^D, คีย์เครื่องหมายลูกศร และ ฟังก์ชันคีย์อื่น ๆ ซึ่งใช้ในการ editing ตัวอักษรที่กดเข้ามา หากเป็น endstroke key จะส่งค่าคีย์นั้นกลับไปยัง function ที่เรียก แต่ถ้าไม่ใช่ จะตรวจสอบสถานะว่า อยู่ในโหมด insert หรือไม่ ถ้า insert คีย์ที่กดเข้ามาจะถูกแทรกเข้าไปในบัฟเฟอร์ แต่ถ้าเป็นไม่ใช่ mode insert คีย์ตัวอักษรที่ถูกกดเข้ามาจะเขียนทับข้อมูลในบัฟเฟอร์

```
/*-------------------------------------------------------------*
|   Function    : read_field();                                |
|   Usage       : int read_field(c);                          |
|   Description : read a field from the keyboard              |
|   Return      : key press                                   |
*-------------------------------------------------------------*/
static int read_field(c)
```

```
{
   int done = FALSE, first = TRUE;

   buff = fld->fvalue.fbuff;
   home_cursor();
   while(TRUE)      {
      gotoxy(fld->fx+fld->fcol-1, fld->frow);
      if (!c || !first)
         c = getkey();
      first = FALSE;
      switch (c)  {
         case CTRL_D  :  delete_word();
                         break;
         case CTRL_FWD : done = !fore_word();
                         break;
         case CTRL_BS  : done = !back_word();
                         break;
         case HOME     : fld->fx = 1;
                         home_cursor();
                         break;
         case END      : end_cursor();
                         break;
         case FWD      : forward();
                         break;
         case BS       : backspace();
                         break;
         case '\b'     : if (!backspace())
                             break;
                         gotoxy(fld->fx+fld->fcol-1, fld->frow);
         case DEL      : delete_char();
                         disp_field(fld, buff, mask);
                         break;
         case INS      : inserting ^= TRUE;
```

```
                        insert_line();
                        break;
        default         : if (endstroke(c)) {
                             done = TRUE;
                             break;
                          }
                          if (inserting)  {
                             memmove(buff+1,buff,strlen(buff)-1);
                             disp_field(fld,buff,mask);
                             gotoxy(fld->fcol+fld->fx-1,fld->frow);
                          }
                          *buff = (char) c;
                          write_char(wkw.lf+fld->fcol+fld->fx-2,
                          wkw.tp+fld->frow-1,(char) c,
                          FIELDFG|(FIELDBG<<4));
                          forward();
                          if (!*mask) {
                             fld->fx--;
                             mask--;
                             buff--;
                          }
                          break;
      }
      if (done || !*mask)
         break;
   }
   wkw.wx = fld->fx+1;
   return c;
}
```

ฟังก์ชันเกี่ยวกับ editing character

เป็นฟังก์ชันที่ทำหน้าที่คล้าย word processor เพื่อจัดการกับตัวอักษรที่กดเข้ามา เช่น การลบตัวอักษร การพิมพ์แทรกตัวอักษร การลบคำ การลบทั้งบรรทัด การเลื่อนไปยังคำถัดไป การเลื่อนไปยังคำก่อนหน้า การเลื่อนไปยังท้ายบรรทัด การเลื่อนไปยังท้ายบรรทัด การเปลี่ยนโหมด insertเป็นต้น

ฟังก์ชันเหล่านี้ได้แก่

```
insert_line();
home_cursor();
end_cursor();
forward();
backspace();
fore_word();
back_word();
delete_char();
delete_word();
```

```
/*-------------------------------------------------------------*
|   Function    : insert_line();                              |
|   Usage       : void insert_line();                         |
|   Description : set insert/exchange cursor shape            |
|   Return      : None                                        |
*-------------------------------------------------------------*/
void insert_line()
{
   set_cursor_type(inserting ? 0x0407 : 0x0607);
}
```

```
/*-------------------------------------------------------------*
|   Function    : home_cursor();                              |
|   Usage       : void home_cursor();                         |
|   Description : home the cursor in the field                |
```

```
|   Return      : None                                                    |
*-------------------------------------------------------------------*/
static void home_cursor()
{
   buff = fld->fvalue.fbuff+fld->fx-1;
   mask = fld->mask.fbuff+fld->fx-1;
   while (*mask != FIELDCHAR) {
      fld->fx++;
      mask++;
   }
}

/*-------------------------------------------------------------------*
|   Function    : end_cursor();                                       |
|   Usage       : void end_cursor();                                  |
|   Description : move the cursor to the end of the field             |
|   Return      : None                                                |
*-------------------------------------------------------------------*/
static void end_cursor()
{
   fld->fx += strlen(mask)-1;
   buff += strlen(buff)-1;
   mask += strlen(mask)-1;
   while(*buff== ' ')
      if (!backspace())
         break;
   forward();
}

/*-------------------------------------------------------------------*
|   Function    : forward();                                          |
|   Usage       : void forward();                                     |
|   Description : move the cursor forward one character               |
```

```
|   Return      : None                                                    |
*-------------------------------------------------------------------*/
static void forward()
{
   do {
      fld->fx++;
      mask++;
   } while (*mask && *mask != FIELDCHAR);
   buff++;
}

/*-------------------------------------------------------------------*
|   Function    : backspace();                                        |
|   Usage       : int backspace();                                    |
|   Description : move back one character                             |
|   Return      : TRUE if can move back                               |
|                 FALSE otherwise                                     |
*-------------------------------------------------------------------*/
static int backspace()
{
   if (buff != fld->fvalue.fbuff) {
      --buff;
      do {
         --mask;
         --(fld->fx);
      } while (*mask != FIELDCHAR);
      return TRUE;
   }
   return FALSE;
}
```

```
/*-------------------------------------------------------------*
|   Function    :  fore_word();                               |
|   Usage       :  int fore_word();                           |
|   Description :  move forward one word                      |
|   Return      :  TRUE if can move forward                   |
|                  FALSE otherwise                            |
*-------------------------------------------------------------*/
static int fore_word()
{
   int ct = 2, test = *buff == ' ';

   while (ct--) {
      while ((*buff == ' ') == test && *mask)
         forward();
      if (!*mask)
         return FALSE;
      if (test)
         break;
      test = !test;
   }
   return TRUE;
}


/*-------------------------------------------------------------*
|   Function    : back_word();                                |
|   Usage       : int back_word();                            |
|   Description : move backward one word                      |
|   Return      : TRUE if can move back                       |
|                 FALSE otherwise                             |
*-------------------------------------------------------------*/
static int back_word()
{
```

```
    int test;

    if (buff == fld->fvalue.fbuff)
       return FALSE;
    if (*(buff-1) == ' ')
       backspace();
    test = *buff == ' ';
    while ((*buff == ' ') == test && buff != fld->fvalue.fbuff)
       backspace();
    if (test)
       while (*buff != ' ' && buff != fld->fvalue.fbuff)
          backspace();
    if (*buff == ' ')
       forward();
    return TRUE;
}

/*-------------------------------------------------------------*
|   Function    : delete_char();                               |
|   Usage       : void delete_char();                         |
|   Description : delete a character                          |
|   Return      : None                                        |
*-------------------------------------------------------------*/
static void delete_char()
{
   memmove(buff, buff+1, strlen(buff));
   *(buff+strlen(buff)) = ' ';
}


/*-------------------------------------------------------------*
|   Function    : delete_word();                              |
|   Usage       : void delete_word();                         |
```

```
|   Description : delete a word                                    |
|   Return      : None                                             |
*-----------------------------------------------------------------*/
static void delete_word()
{
   int test = *buff == ' ';
   int ln = strlen(buff);

   while((*buff == ' ') == test && ln--)
      delete_char();
   if (!test)
      delete_char();
   disp_field(fld,buff,mask);
}
```

ฟังก์ชัน endstroke();

เป็นฟังก์ชันที่ตรวจสอบ คีย์ที่กดเข้ามาว่า เป็นคีย์ที่กำหนดให้สิ้นสุดการ edit ในแต่ละ field หรือไม่ ถ้าใช่ ฟังก์ชันจะ return ค่า 1 แต่ถ้าหากไม่ใช่ จะ return ค่า 0

```
/*-----------------------------------------------------------------*
|   Function    : endstroke();                                     |
|   Usage       : int endstroke(int c);                            |
|   Description : test c for an ending keystroke                   |
|   Return      : True for c being endstroke                       |
|                 False otherwise                                  |
*-----------------------------------------------------------------*/
static int endstroke(int c)
{
   switch (c) {
      case '\r' :
      case '\n' :
```

```
        case '\t' :
        case ESC  :
        case F1   :
        case F2   :
        case F3   :
        case F4   :
        case F5   :
        case F6   :
        case F7   :
        case F8   :
        case F9   :
        case F10  :
        case PGUP :
        case PGDN :
        case HOME :
        case END  :
        case CTRL_FWD  :
        case CTRL_BS   :
        case CTRL_HOME :
        case CTRL_END  :
        case UP   :
        case DN   :
           return TRUE;
        default   :
           return FALSE;
    }
}
```

การทำงานของฟังก์ชันที่อยู่ในไฟล์ LLMODULE.C

ฟังก์ชันที่อยู่ในไฟล์ LLMODULE.C เป็นฟังก์ชันที่ทำงานเกี่ยวกับการจัดการลิงค์ลิสต์ โดยมีการทำงานดังต่อไปนี้

การสร้างลิงค์

การเพิ่มลิงค์

การลบลิงค์

การเลื่อนตำแหน่ง pointer เพื่อชี้ที่ node ก่อนหน้า

การเลื่อนตำแหน่ง pointer เพื่อชี้ที่ node ถัดไป

การทำการตรวจสอบว่า node ใด มีข้อมูลใน node ตรงกับข้อมูลที่ให้เข้าไป

การเลื่อนตำแหน่ง pointer เพื่อชี้ไปยังต้นลิสต์

การเลื่อนตำแหน่ง pointer เพื่อชี้ไปยังท้ายลิสต์

การหาขนาดของลิสต์

การเอาค่าของ node ปัจจุบันกลับออกมา

การทำงานของฟังก์ชันในไฟล์นี้ทั้งหมด จะต้องมีการ initial ลิสต์เสียก่อน เพื่อให้ระบบสามารถรู้ได้ว่า แต่ละ node นั้นมีขนาดเท่าใด ซึ่งการกำหนดรูปแบบของแต่ละ node จะสามารถกำหนดได้ในไฟล์ LLDEF.H

การกำหนดตัวแปรที่ใช้ในฟังก์ชันต่าง ๆ สามารถทำได้ดังต่อไปนี้

```
static struct LINKLIST *list;
```

ฟังก์ชันทำหน้าที่กำหนดฟังก์ชันเปรียบเทียบให้แก่ลิงค์ลิสต์

ฟังก์ชันที่ทำการเปรียบเทียบค่าในลิงค์ลิสต์นั้น ถ้าหากเปลี่ยนรูปแบบ หรือโครงสร้างของลิงค์ลิสต์แต่ละ node ฟังก์ชันที่ทำการเปรียบเทียบค่าใน node ก็ไม่สามารถใช้ได้แล้ว ดังนั้น การทำให้ลิงค์ลิสต์ สามารถทำงานกับ โครงสร้างทุกรูปแบบ จึงจะต้องทำให้ลิงค์ลิสต์มี field หนึ่งที่ทำหน้าที่กำหนดฟังก์ชันที่ใช้ได้ใหม่ สำหรับโครงสร้างที่เปลี่ยนไป ฟังก์ชัน llsetmatch(); มีหน้าที่ดังกล่าว

```
/*-------------------------------------------------------------*
|  Function     :  Llsetmatch                                 |
|  Usage        :  void llsetmatch(int (*numatch)();          |
|  Description  :  Set matching function                      |
|  Return       :  None                                       |
*-------------------------------------------------------------*/
void llsetmatch(int (*numatch)())
{
   list->match = numatch;
}
```

ฟังก์ชันทำการตรวจสอบค่าของ node

ฟังก์ชันนี้ทำหน้าที่ตรวจสอบค่าภายในลิงค์ลิสต์ โดยเปรียบเทียบค่าในลิสต์กับค่าที่กำหนดให้ ทีละลิงค์ ถ้าหากพบจะส่งค่า 1 กลับ แต่ถ้าหากไม่พบ จะส่งค่า 0 กลับไป

ฟังก์ชันที่จะทำการเปรียบเทียบนั้น เพื่อจะให้สามารถเปรียบเทียบกับข้อมูลได้หลายลักษณะ จึงมีการกำหนดฟังก์ชันเปรียบเทียบได้ใหม่ทุกครั้งสำหรับลิสต์แต่ละลิสต์

```
/*-------------------------------------------------------------*
|  Funtion      :  llcheck                                    |
|  Usage        :  int llcheck(char *lookfor);                |
|  Description  :  Set clp to desired link                    |
|  Return       :  return True if found, False otherwise      |
*-------------------------------------------------------------*/
int llcheck(char *lookfor)
{
   int temp;
```

```
   for(;;)  {
      temp = (*list->match) (lookfor,list->clp->item);
      if (temp == 0)
         return (1);
      else
         if (temp < 0)
            return(0);
         else
            if (!llnext())
               return(2);
   }
}
```

ฟังก์ชัน llsetlist();

ทำหน้าที่เซ็ตให้ฟังก์ชันที่ทำงานทุกฟังก์ชันในไฟล์นี้ ให้ทำงานกับลิสต์ที่ผ่านให้กับฟังก์ชันนี้ เป็นการเซ็ตลิสต์ที่ทำงานปัจจุบัน

```
/*-------------------------------------------------------------*
|  Function     :  llsetlist                                  |
|  Usage        :  void llsetlist(struct LINKLIST *new_list);  |
|  Description  :  Set this module to work with a new list.   |
|  Return       :  None                                       |
*-------------------------------------------------------------*/
void llsetlist(struct LINKLIST *new_list)
{
   list = new_list;
}
```

ฟังก์ชัน llsetsize();

เป็นฟังก์ชัน ที่ทำหน้าที่กำหนดขนาดที่ใช้ในการจองหน่วยความจำให้กับแต่ละ node ของลิสต์ ทำให้โมดูลสามารถทำงานได้กับโครงสร้างของลิสต์ที่มีรูปแบบและขนาดต่างกัน

```
/*-------------------------------------------------------------*
|  Function     :  llsetsize                                  |
|  Usage        :  void llsetsize(int size);                  |
|  Description  :  Set the storage requirements for the list  |
|  Return       :  None                                       |
*-------------------------------------------------------------*/
void llsetsize(int size)
{
   list->itemlength = size;
}
```

ฟังก์ชัน llcrlink();

เป็นฟังก์ชันที่ทำหน้าที่จองหน่วยความจำให้กับแต่ละ node โดยมีขนาดเท่ากับค่าของ size ที่อยู่ใน field หนึ่งของโครงสร้างลิสต์

```
/*-------------------------------------------------------------*
|  Function     :  llcrlink                                   |
|  Usage        :  static struct LINKTYPE *llcrlink();        |
|  Description  :  Allocate storage for a link.               |
|  Return       :  Pointer to storage for a link              |
*-------------------------------------------------------------*/
static struct LINKTYPE *llcrlink()
{
```

```
   struct LINKTYPE *link;

   link = (struct LINKTYPE *) malloc (sizeof(struct LINKTYPE));
   link->item = malloc(list->itemlength);
   return(link);
}
```

ฟังก์ชัน llinit();

เป็นฟังก์ชันที่ทำหน้าที่ initial ลิงค์ลิสต์ โดยทำการจองเนื้อที่ให้ลิสต์แรก ทำการเซ็ต head และ tail และ clp ให้ชี้ไปยัง node ใหม่ที่ได้ รวมทั้งเซ็ตขนาดของลิสต์ให้มีค่าเป็น 1 ดังนั้น next และ previous จึงถูกเซ็ตให้มีค่าเป็น NULL

```
/*-------------------------------------------------------------*
|  Function     :  llinit                                     |
|  Usage        :  void llinit(char *newitem);                |
|  Description  :  Initialize the structure                   |
|  Return       :  None                                       |
*-------------------------------------------------------------*/
void llinit(char *newitem)
{
   struct LINKTYPE *llcrlink();

   list->head = list->tail = list->clp = llcrlink();
   list->clp->next = list->clp->previous = NULL;
   moveitem(newitem,list->clp->item);
   list->listlength = 1;
}
```

ฟังก์ชัน llhead();

ทำหน้าที่เซ็ต clp ให้ชี้ไปที่หัวของลิสต์

```
/*-------------------------------------------------------------*
|  Function     :  llhead                                     |
|  Usage        :  void llhead();                             |
|  Description  :  Set the CLP to the head of the list.       |
|  Return       :  None                                       |
*-------------------------------------------------------------*/
void llhead()
{
   list->clp = list->head;
}
```

ฟังก์ชัน llzero();

ทำหน้าที่กำหนดให้ head , tail และ clp ให้มีค่าเป็น NULL และ เซ็ตขนาดของลิสต์ให้มีค่าเป็น 0 ซึ่งหมายถึง ลิสต์นี้ไม่มี node

```
/*-------------------------------------------------------------*
|  Function     :  llzero                                     |
|  Usage        :  void llzero()                              |
|  Description  :  Initial length of link list to zero        |
|  Return       :  None                                       |
*-------------------------------------------------------------*/
void llzero()
{
   list->head = list->tail = list->clp = NULL;
   list->listlength = 0;
}
```

ฟังก์ชัน lltail();

ทำหน้าที่ กำหนดค่าของ clp ให้ชี้ไปที่ tail ของลิสต์

```
/*-------------------------------------------------------------*
|  Function     :  lltail                                     |
|  Usage        :  void lltail();                             |
|  Description  :  Set the Clp to the tail of the list.       |
|  Return       :  None                                       |
*-------------------------------------------------------------*/
void lltail()
{
   list->clp = list->tail;
}
```

ฟังก์ชัน llnext();

เป็นฟังก์ชันที่ทำหน้าที่ เลื่อน clp ให้ชี้ไปยัง node ถัดไปจากตำแหน่งปัจจุบัน ถ้าหากอยู่ที่ตำแหน่งท้ายลิสต์อยู่แล้ว และไม่สามารถเลื่อนต่อไปได้ จะส่งค่า 0 กลับมา ถ้าหากสามารถเลื่อนได้ ก็จะเลื่อน และส่งค่า 1 กลับมา

```
/*-------------------------------------------------------------*
|  Function     :  llnext                                     |
|  Usage        :  int llnext();                              |
|  Description  :  Set the CLP to the next link               |
|  Return       :  return FALSE if at end of list,            |
|                  TRUE otherwise.                            |
*-------------------------------------------------------------*/
int llnext()
```

```
{
   if (list->clp->next == NULL )
      return (0);
   else {
      list->clp = list->clp->next;
      return (1);
   }
}
```

ฟังก์ชัน llprevious();

ทำหน้าที่เลื่อน clp ให้ชี้ไปยังตำแหน่ง node ที่อยู่ก่อนหน้า node ปัจจุบัน และ เช่นเดียวกับ llnext(); ที่เมื่ออยู่ที่หัวแล้ว และไม่สามารถเลื่อนไปได้ จะส่งค่า 0 กลับมา และหากเลื่อนได้ จะส่งค่า 1 กลับมา

```
/*-------------------------------------------------------------*
|  Function     :  llprevious                                 |
|  Usage        :  int llprevious();                          |
|  Description  :  Set the CLP to the previous link           |
|  Return       :  return FALSE if at head of list,           |
|                  TRUE otherwise.                            |
*-------------------------------------------------------------*/
int llprevious()
{
   if (list->clp->previous == NULL)
      return 0;
   else {
      list->clp = list->clp->previous;
      return (1);
   }
}
```

ฟังก์ชัน llretrieve();

ทำหน้าที่ เอาค่าที่อยู่ใน node ปัจจุบัน ส่งให้แก่ตัวแปรที่ส่งเข้าไปเพื่อรับค่าที่ส่งนี้

```
/*-------------------------------------------------------------*
|  Function     :  llretrieve                                 |
|  Usage        :  void llretrieve(char *newitem)             |
|  Description  :  Retrieve the item from the CLP link.       |
|  Return       :  None                                       |
*-------------------------------------------------------------*/
void llretrieve(char *newitem)
{
   moveitem(list->clp->item, newitem);
}
```

ฟังก์ชัน lladd();

ทำหน้าที่ใส่ node เข้าไปในลิสต์ที่ตำแหน่งที่ clp ชี้อยู่ โดยการเปลี่ยนการชี้ของ pointer ที่ชี้ต่ออยู่ระหว่าง previous และ next ของ clp โดยเริ่มต้นจะมีการตรวจสอบเสียก่อนว่า ลิสต์นี้เป็นลิสต์ว่างหรือไม่ ถ้าหากเป็นลิสต์ว่าง ก็ initial ลิสต์นั้น และสิ้นสุดการทำงาน

```
/*-------------------------------------------------------------*
|  Function     :  lladd                                      |
|  Usage        :  void lladd(char *newitem)                  |
|  Description  :  Add a new link containing this item to     |
|                  the link following the CLP, and reset      |
|                  CLP to new link.                           |
|  Return       :  None                                       |
*-------------------------------------------------------------*/
```

```
void lladd(char *newitem)
{
   struct LINKTYPE *newlink;
   struct LINKTYPE *llcrlink();

/* If empty,  initialize list.
   ---------------------------- */
   if (ll_length() == 0) {
      llinit(newitem);
      return;
   }

/* Create new link.
   ----------------- */
   newlink = llcrlink();
   moveitem(newitem, newlink->item);
   list->listlength++;

/* Reset pointers.
   ----------------- */
   newlink->next = list->clp->next;
   newlink->previous = list->clp;
   if (list->tail == list->clp)
      list->tail = newlink;
   else
      list->clp->next->previous = newlink;
   list->clp->next = newlink;
   list->clp = newlink;
}
```

ฟังก์ชัน lladdhead();

ทำหน้าที่เช่นเดียวกับ lladd(); เพียงแต่เป็นการ add ที่ตำแหน่งหัวลิสต์เท่านั้น

```
/*-------------------------------------------------------------*
|  Function     :  lladdhead                                  |
|  Usage        :  void lladdhead(char *newitem)              |
|  Description  :  Add a new head, reset CLP.                 |
|  Return       :  None                                       |
*-------------------------------------------------------------*/
void lladdhead(char *newitem)
{
   struct LINKTYPE *newlink;
   struct LINKTYPE *llcrlink();

/* If empty,  initialize list.
   --------------------------- */
   if (ll_length() == 0) {
      llinit(newitem);
      return;
   }
/* Create new link.
  ------------------- */
   newlink = llcrlink();
   moveitem(newitem,newlink->item);
   list->listlength++;

/* Reset pointers.
   ------------------- */
   newlink->previous = NULL;
```

```
   newlink->next = list->head;
   list->head->previous = newlink;
   list->clp = list->head = newlink;
}
```

ฟังก์ชัน lladdtail();

เป็นฟังก์ชันที่ทำหน้าที่เพิ่มลิงค์เข้าที่ท้ายลิสต์

```
/*-------------------------------------------------------------*
|   Function    :  lladdtail();                               |
|   Usage       :  int lladdtail(char *item);                 |
|   Description :  add item to tail of list                   |
|   Return      :  None                                       |
*-------------------------------------------------------------*/
void lladdtail(char *newitem)
{
   struct LINKTYPE *newlink;
   struct LINKTYPE *llcrlink();

/* If empty,  initialize list.
   --------------------------- */
   if (ll_length() == 0) {
      llinit(newitem);
      return;
   }
/* Create new link.
  ------------------- */
   newlink = llcrlink();
   moveitem(newitem,newlink->item);
```

```
   list->listlength++;

/* Reset pointers.
   ------------------- */
   newlink->previous = list->tail;
   newlink->next = NULL;
   list->tail->next = newlink;
   list->clp = list->tail = newlink;
}
```

ฟังก์ชัน lldelete();

เป็นฟังก์ชันที่ทำหน้าที่ ลบ node ที่ clp ชี้อยู่ และทำการรีเซ็ตค่าของ pointer ให้ชี้อย่างถูกต้อง

```
/*-------------------------------------------------------------*
|  Function     :  lldelete                                   |
|  Usage        :  void lldelete();                           |
|  Description  :  Delete and free the CLP, reset CLP to haed.|
|  Return       :  None                                       |
*-------------------------------------------------------------*/
void lldelete()
{
   struct LINKTYPE *before, *after;

/* Is this the only link?
   ---------------------- */
   if (list->head == list->clp && list->tail == list->clp) {
      list->head = list->tail = NULL;
   }
```

```
   /* Is this the head?
      -------------------- */
      else if (list->head == list->clp) {
         list->head = list->head->next;
         list->head->previous = NULL;
      }

   /* Is this the tail?
      ---------------------- */
      else if (list->tail == list->clp) {
         list->tail = list->tail->previous;
         list->tail->next = NULL;
      }

   /* Otherwise, it must be inside the list.
      ---------------------------------------- */
      else {
         before = list->clp->previous;
         after = list->clp->next;
         before->next = after;
         after->previous = before;
      }

   /* Delete CLP.
      ------------- */
      free(list->clp);
      list->clp = list->head;
      list->listlength--;
   }
```

ฟังก์ชัน lllength();

เป็นฟังก์ชันที่ทำหน้าที่ส่งค่าความยาวของลิสต์กลับให้แก่ฟังก์ชันที่เรียกฟังก์ชันนี้

```
/*-------------------------------------------------------------*
|  Function     :  ll_length                                  |
|  Usage        :  int ll_length();                           |
|  Description  :  Return the length of the list              |
|  Return       :  Length of the list                         |
*-------------------------------------------------------------*/
int ll_length()
{
   return (list->listlength);
}
```

ฟังก์ชัน llindex();

เป็นฟังก์ชันที่ทำหน้าที่เลื่อน clp ให้ชี้ไปยังลิงค์ที่ต้องการ relative กับ head

```
/*-------------------------------------------------------------*
|  Function    :  llindex                                     |
|  Usage       :  int llindex(int index);                     |
|  Description :  Move clp to specified node                  |
|  Return      :  on success  return  number of node          |
|                 on failure  return  last node number        |
*-------------------------------------------------------------*/
int llindex(int index)
{
   int temp = 1;

   llhead();
```

```
   if (list->clp == NULL)
      return 0;

   while( --index != 0 ) {
      if (llnext() != 0) {
         temp++;
      }
      else  break;
   }
   return(temp);
}
```

ฟังก์ชัน get_wk_list();

เป็นฟังก์ชันที่ทำหน้าที่ return ค่าของ current list pointer กลับมายังโปรแกรมที่เรียกใช้ เพื่อเก็บไว้ และสามารถเปลี่ยนไปใช้งานลิสต์อื่น จากนั้น จึงนำเอาลิสต์เดิมที่เก็บไว้ กลับมาทำงานต่อไปได้

```
/*-------------------------------------------------------------*
|   Function    : get_wk_list();                              |
|   Usage       : struct LINKLIST *get_wk_list();             |
|   Description : get current working list                    |
|   Return      : current working list                        |
*-------------------------------------------------------------*/
struct LINKLIST *get_wk_list()
{
   return list;
}
```

ฟังก์ชัน llupdate();

เป็นฟังก์ชันที่ทำหน้าที่ เปลี่ยนแปลงค่าของลิงค์ปัจจุบัน

```
/*-------------------------------------------------------------*
|   Function    : llupdate();                                 |
|   Usage       : void llupdate(char *item);                  |
|   Description : update link list value                      |
|   Return      : None                                        |
*-------------------------------------------------------------*/
int llupdate(char *item)
{
   if (ll_length() == 0)
      return 0;
   else {
      lladd(item);
      llprevious();
      lldelete();
   }
}
```

ฟังก์ชัน select_list();

เป็นฟังก์ชันที่ทำหน้าที่รับลิสต์เข้าไป แสดงผลในลักษณะของบรรทัดในวินโดว์ สามารถแสดงผลลิสต์ที่มีขนาดยาวเท่าใดก็ได้ โดยวินโดว์ที่จะแสดงนั้น จะมีขนาดคงที่อยู่ขนาดหนึ่ง (ในที่นี้มีขนาด 10 บรรทัด) ถ้าลิสต์มีขนาดเล็กกว่านี้จะแสดงเพียงเท่าขนาดของลิสต์ แต่ถ้าลิสต์มีขนาดใหญ่กว่า จะแสดงผลเพียง 10 บรรทัด ส่วนที่เกินไปสามารถแสดงผลได้โดยการกดปุ่มลูกศรลง เพื่อดูลิงค์ที่อยู่ที่ลำดับมากกว่า 10 ได้จนกว่าจะสิ้นสุดลิสต์

```
/*-------------------------------------------------------------*
|   Function    :  select_list();                             |
|   Usage       :  int select_list(int lf,int tp,             |
|                          struct LINKLIST *list,int high);   |
|   Description :  display list of list                       |
|   Return      :  selected list bar                          |
*-------------------------------------------------------------*/
int select_list(int lf,int tp,struct LINKLIST *list,int high)
{
   int height,wd;
   int c;

   lineno = 1;
   height = (high >= 10) ? 10 : high;

   establish_window(lf,tp,lf+14,tp+height+1,WHITE,BLUE,1);
   wkw.wtop = 1;
   wkw.wx = wkw.wy = 1;

   while( TRUE ) {
      dlist(list);
      highlight(lineno);
      hidecursor();
      c = getkey();
      if ( c == '\r' || c == ESC)
         break;
      switch (c) {
         case UP :
                   lupline();
                   break;
```

```
        case DN :
                ldownline(list);
                break;
        default : putch(BELL);
      }
   }
   delete_window();
   return c == ESC ? 0 : lineno;
}
```

ฟังก์ชัน dlist();

เป็นฟังก์ชันที่ทำหน้าที่แสดงผลลิสต์ในวินโดว์ โดยจะแสดงผลเท่ากับขนาดของวินโดว์เท่านั้น ส่วนที่มากกว่าจะไม่ถูกแสดง

```
/*-------------------------------------------------------------*
|   Function    :  dlist();                                   |
|   Usage       :  void dlist(struct LINKLIST *list);         |
|   Description :  display list in window                     |
|   Return      :  none                                       |
*-------------------------------------------------------------*/
static void dlist(struct LINKLIST *list)
{
   int height;
   int i = 1;

   height = wkw.ht;

   llindex(wkw.wtop);
   while((list->clp != NULL) && (--height != 1)) {
```

```
        win_dstr(2,++i,list->clp->item,7);
        win_dstr(2+strlen(list->clp->item),i,spaces+79-wkw.wd+
                strlen(list->clp->item)+2,7);
        llnext();
    }
}
```

ฟังก์ชัน highlight();

เป็นฟังก์ชันที่ทำหน้าที่ เปลี่ยนแอททริบิวท์ของลิงค์ปัจจุบัน เพื่อให้เป็นที่สังเกต และเลือกโดยการกด ENTER

```
/*-------------------------------------------------------------*
|   Function    : highlight();                                |
|   Usage       : void highlight(int row);                    |
|   Description : display line of text of reverse attribute   |
|   Return      : None                                        |
*-------------------------------------------------------------*/
static void highlight(int row)
{
    if (llindex(row) == row) {
        wkw.wy = (row-wkw.wtop+1 > wkw.ht-2) ? 10 : row-wkw.wtop+1;
        win_reverse(2,wkw.wy+1,wkw.wd-2,0x70);
    }
}
```

ฟังก์ชัน lupline();

เป็นฟังก์ชันที่ทำหน้าที่ เลื่อนลิงค์ขึ้นไปทางต้นลิงค์ 1 ลิงค์

```
/*-------------------------------------------------------------*
|   Function    : lupline();                                  |
|   Usage       : void lupline();                             |
|   Description : move up one link                            |
|   Return      : None                                        |
*-------------------------------------------------------------*/
static void lupline()
{
   if (lineno>1) {
      lineno--;
      if (wkw.wtop == lineno+1)
         wkw.wtop--;
   }
}
```

ฟังก์ชัน ldownline();

เป็นฟังก์ชัน ทำหน้าที่เลื่อนลิงค์ลงด้านล่าง 1 ลิงค์

```
/*-------------------------------------------------------------*
|   Function    : ldownline();                                |
|   Usage       : void ldownline(struct LINKLIST *list);      |
|   Description : move down one link                          |
|   Return      : None                                        |
*-------------------------------------------------------------*/
static void ldownline(struct LINKLIST *list)
{
```

```
    if (list->clp->next != NULL) {
        if ((lineno < wkw.ht-2) || (lineno-wkw.wtop+1<wkw.ht-2))
            lineno++;
        else {
            wkw.wtop++;
            lineno++;
        }
    }
}
```

การทำงานของฟังก์ชันในไฟล์ MESAD.C

ฟังก์ชันในไฟล์ MESAD.C เป็นฟังก์ชันที่ทำหน้าที่เกี่ยวกับการจัดการ message file เช่น การอัดเสียงเพื่อเก็บไว้ให้ user ฟัง การ clean up message file และกลุ่มฟังก์ชันย่อย ๆ อีกหลายฟังก์ชัน ดังนี้คือ

1. กลุ่มฟังก์ชันเกี่ยวกับการจัดการ GENERAL INFORMATION
   - ฟังก์ชันการอัดเสียง
   - ฟังก์ชันการเล่นเสียง
   - ฟังก์ชันการกำหนด message ให้แก่ user แต่ละคน

2. กลุ่มฟังก์ชันเกี่ยวกับการ clean up message

   - del_message();
   - ch_dir();
   - del();
   - chd();
   - dtrue(unsigned int,char *);
   - daynum(int,int,int);
   - check_heard(char *);

ฟังก์ชัน del_message();

เป็นฟังก์ชันหลักของ ส่วน clean up ที่เตรียมสถานะของการเรียกฟังก์ชันที่เกี่ยวข้อง ได้แก่ del();, ch_dir();, chd();, dtrue(); และ check_heard();

```
static int del_message()
{
```

```
   establish_window(7,9,70,22,ENTRYFG,ENTRYBG,1);
   win_dstr(8,4,"File  ",ENTRYFG|(ENTRYBG<<4));
   win_dstr(8,6,"Date No.",ENTRYFG|(ENTRYBG<<4));
   data_entry(del_field,1,1);
   delete_window();
}
```

ฟังก์ชัน first_mes();

เป็นฟังก์ชันที่เตรียมสถานะเริ่มต้นของการเก็บ General Information เพื่อบริการให้ user ฟัง ก่อนที่จะได้ไปใช้บริการอย่างอื่นต่อไป การเตรียม General Information นี้ สามารถกำหนดให้ฟังโดย user ทุกคน หรือเฉพาะบางคนก็ได้

ฟังก์ชันเริ่มการทำงานด้วย การเรียกฟังก์ชัน data_entry(); พร้อมกับส่ง parameter เป็น f_m_field ซึ่งเป็น template ของการเก็บ General Information ภายในเป็นการกำหนด flag ว่า message ที่อัดไว้นี้ จะเป็นข่าวสารถึง user ทุกคนหรือเพียงบางคน จากนั้น จึงเรียกฟังก์ชัน record(); เพื่ออัดเสียงเก็บไว้ เสียงที่เก็บไว้สามารถฟังได้ ด้วยการเรียกฟังก์ชัน play(); เมื่อต้องการฟังว่าเสียงที่อัดไว้นั้น ถูกต้องหรือไม่

เมื่อสิ้นสุดการทำงานของ data_entry(); ฟังก์ชันจะตรวจสอบ flag ถ้าหากเป็นการกำหนดให้ user ทุกคนฟัง ฟังก์ชันจะเซ็ต to_all_flag ให้เป็น 1 เพื่อกำหนดให้ เล่นไฟล์เสียงนี้ให้แก่ทุกคนที่โทรเข้ามาในระบบ หรือถ้ากำหนดให้ฟังเพียงบางคน ก็จะเรียก data_entry(); ใหม่ ด้วย parameter เป็น template ของการรับค่าของ user id เพื่อสามารถกำหนดให้ message ที่เก็บไว้ถูกนำมาเล่นสำหรับ user คนใดคนหนึ่งโดยเฉพาะได้

```
static int first_mes()
{
   char fname[MAXPATH];
   char userfname[MAXPATH];

   establish_window(7,8,68,18,WHITE,ENTRYBG,1);
```

```
    win_dstr(8,9,"Status : ",WHITE|(ENTRYBG<<4));
    data_entry(f_m_field,1,1);
    if(f_m_field[0].fvalue.value->value == 1)
       to_all_flag = 1;
    else {
       establish_window(7,18,68,23,WHITE,MAGENTA,1);
       win_dstr(7,2,"User Id  :",BLACK|(CYAN<<4));
       while(data_entry(s_m_field,1,1)) {
          strcpy(edit_st.id,uid);
          if (KramRead(KramFile,edit_st.id,edit_st.name) == NULL)
             error_message(" User Not Found ");
          else {
             edit_st.first_flag = 1;
             KramUpdate(KramFile,edit_st.id,edit_st.name);
             strcpy(userfname,voc_dir); strcat(userfname,"\\");
             strcat(userfname,uid);
             strcat(userfname,"firs.voc");
             strcpy(fname,voc_dir); strcat(fname,"\\");
             strcat(fname,"MESSAGEF.TMP");
             newcopy(fname,userfname);
          }
       }
       delete_window();
       unlink(fname);
    }
    delete_window();
}
```

ฟังก์ชัน ch_dir();

เป็นฟังก์ชันที่ทำหน้าที่ เปลี่ยน directory เพื่อให้สามารถลบไฟล์ message ที่ต้องการได้ หากเลือก path ที่จะเปลี่ยนแล้ว และกด "Ok" ก็จะเปลี่ยนเป็น directory ใหม่ แต่ถ้ากด " Cancel " ก็จะเปลี่ยนเป็น directory เดิม

```
static int ch_dir()
{
   int temp;
   FIELD dir_template[] = {
      {1,3,16,1,dir,dirmask,"New Directory Name"},
      {4,4,5,'C'," Chdir ",NULL,"Change Directory"},
      {4,4,20,'O'," Ok ",NULL,"Return to Upper Menu"},
      {4,4,35,'C'," Cancel ",NULL,"Return to Upper Menu
       without Chdir"},
      NULL
   };

   dir_template[1].mask.func = chd;
   dir_template[2].mask.func = quit;
   dir_template[3].mask.func = cancel;

   strcpy(cur_dir,"C:\\");
   getcurdir(0,cur_dir+strlen("C:\\"));

   setmem(dir,30,0);
   establish_window(12,10,60,16,CHDIRFG,CHDIRBG,1);
   win_dstr(5,3,"New Dir",WHITE|(CHDIRBG<<4));
   window_title(" Change Directory ",CHDIRFG|(CHDIRBG<<4));
   temp = data_entry(dir_template,1,1);
```

```
   if (temp == CANCEL)
      chdir(cur_dir);
   delete_window();

}
```

ฟังก์ชัน del();

เป็นฟังก์ชันที่ถูกเรียกโดย del_message(); อีกทีหนึ่ง เป็นส่วนที่ทำการตรวจสอบ message file ว่าถูกต้องตามลักษณะที่กำหนดหรือไม่ เช่น เป็นไฟล์ที่ฟังแล้วหรือไม่ และ ฟังมาแล้วกี่วัน ซึ่งลักษณะเหล่านี้ สามารถกำหนดได้โดย admin เพื่อความยืดหยุ่นของโปรแกรม

ฟังก์ชันเรียก dtrue(); เพื่อตรวจสอบเวลาที่ติดมากับไฟล์ว่า ถูกเก็บไว้กี่วันแล้ว ตรงกับจำนวนวันที่กำหนดหรือไม่ ส่วนลักษณะที่ message ได้ถูกเปิดฟังแล้วหรือไม่ กำหนดได้ด้วยชื่อไฟล์เอง คือ หากเป็นไฟล์ที่ยังไม่ถูกเปิดฟัง จะมีชื่อไฟล์อักษรตัวที่ 8 เป็นอักษร "0" แต่ถ้าหากถูกเปิดฟังแล้ว ชื่อไฟล์อักษรตัวที่ 8 จะเป็นอักษร "1" แทน

```
static int del()
{
   int temp;
   int done;
   struct ffblk ffblk;

   llsetlist(&dellist);

   done = findfirst(df.del_file,&ffblk,0);

   while(!done) {
      if (dtrue(ffblk.ff_fdate,df.day_num) &&
          check_heard(ffblk.ff_name))
```

```
        lladd(ffblk.ff_name);
      done = findnext(&ffblk);
   }
   llhead();
   if (ll_length() != 0)
      select_list(5,10,&dellist,ll_length());
   else
      error_message(" There is no message of specified day ");

   while((&dellist)->clp != NULL) {
      unlink((&dellist)->clp->item);
      lldelete();
    }
    init_template();
    return '\t';
}
```

ฟังก์ชัน dtrue();

เป็นฟังก์ชันที่รับค่า วันที่ของไฟล์ และจำนวนวันที่ต้องการตรวจสอบ มาทำการแปลความหมาย และเปรียบเทียบ ว่าตรงกับที่ต้องการหรือไม่ ถ้าหากตรง ก็จะส่งค่า 1 กลับไป ถ้าหากไม่ตรง ก็จะส่งค่า 0 กลับไป

```
static int dtrue(unsigned int dayin,char *daywant)
{
    int i_daywant;
    int i_dayin;
    int mon,day,year;
    struct date d;

    day = dayin  &   0x001f;          /* bit 0-4 : Day */
```

```
    mon = (dayin  &  0x01e0)>>5;     /* bit 5-8 : Month */
    year = (dayin & 0xfe00)>>9;      /* bit 9-15 : Year */
    i_dayin = daynum(day,mon,year);

    getdate(&d);                     /* get current date */
    day = d.da_day;
    mon = d.da_mon;
    year = d.da_year-1980;
    i_daywant = daynum(day,mon,year);
    return ( i_daywant-i_dayin >= atoi(daywant));
}
```

ฟังก์ชัน daynum();

เป็นฟังก์ชันที่ทำหน้าที่แปลง รหัสวันที่ของไฟล์ ให้เป็นจำนวนวันที่ในลักษณะของ integer

```
static int daynum(int day,int mon,int year)
{
   int i_dayin;
   i_dayin = day;
   while(mon) {
      switch (mon) {
         case 1  :
         case 3  :
         case 5  :
         case 7  :
         case 8  :
         case 10 :
         case 12 : i_dayin += 31;
```

```
                break;
        case 4  :
        case 6  :
        case 9  :
        case 11 : i_dayin += 30;
                  break;
        case 2  : if ( year % 4 )
                     i_dayin += 29;
                  else
                     i_dayin += 28;
                  break;
      }
      mon--;
    }
    i_dayin += year*365;
    if (year%4)
       i_dayin += year/4;

    return i_dayin;
}
```

ฟังก์ชัน record();

ทำการอัดเสียง และเก็บไว้ใน directory ชื่อ voc\sysvoc เพื่อเก็บไว้สำหรับให้ user ฟัง มี 2 ลักษณะคือ หากเก็บไว้สำหรับ user ทุกคน จะชื่อว่า "FIRSTMES.VOC" แต่ถ้าหากเก็บไว้สำหรับ user เพียงบางคน จะเก็บไว้ในชื่อ "MESSAGEF.TMP" ซึ่งจะถูก copy ไปเป็น message file ของ user แต่ละคนแยกต่างหากกัน และถูกลบทิ้งเมื่อ copy เสร็จสิ้นแล้ว

```
static int record()
{
   char fname[13] = "MESSAGEF.TMP";

   if(f_m_field[0].fvalue.value->value == 1)
      strcpy(fname,"firstmes.voc");

   win_dstr(17,9,"  Recording... Press any key to stop  ",
            BLACK|(WHITE<<4));
   record_vdata(fname);
   win_dstr(17,9,"                                      ",
            ENTRYFG|(ENTRYBG<<4));
}
```

ฟังก์ชัน play();

เป็นฟังก์ชันที่ทำหน้าที่ เล่นเสียง message ที่เก็บไว้ผ่านทางลำโพง ให้ admin สามารถตรวจสอบความถูกต้องของ message ที่เก็บไว้ เสียก่อน ก่อนที่จะทำงานใด ๆ ต่อไป

```
static int play()
{
   char fname[13] = "MESSAGEF.TMP";

   if(f_m_field[0].fvalue.value->value == 1)
      strcpy(fname,"firstmes.voc");

   win_dstr(17,9,"  Playing ... Press any key to stop  ",
            BLACK|(WHITE<<4));
   play_vdata(fname);
```

```
    win_dstr(17,9,"                                    ",
            ENTRYFG|(ENTRYBG<<4));
}
```

ฟังก์ชัน check_heard();

เป็นฟังก์ชันที่ทำหน้าที่ตรวจสอบชื่อ message file ที่เก็บไว้ ว่าเป็นไฟล์ที่ถูกเปิดฟังแล้วหรือยัง เพราะตามลักษณะที่ได้กล่าวไว้แล้วคือ ถ้าหากไฟล์ที่เก็บไว้ยังไม่ถูกเปิดฟัง อักษรชื่อไฟล์ตัวที่ 8 จะเป็นอักษร "0" แต่ถ้าหากถูกเปิดฟังแล้ว ชื่อไฟล์อักษรตัวที่ 8 จะเปลี่ยนเป็น 1 ฟังก์ชันจึงสามารถตรวจสอบการถูกเปิดฟังของ message ได้

```
static int check_heard(char *fname)
{
   int temp;

   temp = rad_to_code(heardornot);
   switch (temp) {
      case 1 : if (*(fname+7) == '0')
                  return 1;
               else
                  return 0;
      case 2 : if (*(fname+7) == '1')
                  return 1;
               else
                  return 0;
      case 4 : return 1;
   }
}
```

ฟังก์ชัน newcopy();

เป็นฟังก์ชันที่ทำหน้าที่ copy ไฟล์ โดยรับ parameter เป็นชื่อไฟล์ต้นฉบับ และไฟล์สำเนา ถ้าหาก copy ได้จะ return 0 ถ้าหาก copy ไม่ได้จะ return -1 สำหรับการเปิดไฟล์ไม่ได้ และ -2 สำหรับ "disk full"

```
static int newcopy(char *src,char *des)
{
   int shandle;          /* handle of source */
   int dhandle;          /* handle of destination */
   int byte_read;        /* bytes read */
   int byte_write;       /* bytes written */
   long f_length;        /* length of source file */
   unsigned char buff[BUFFLEN];          /* buffer */

   shandle = open(src,O_RDWR);
   if (shandle == -1) {
      error_message(sys_errlist[errno]);
      return -1;
   }
   f_length = filelength(shandle); /* get source file length */

   dhandle = _creat(des,FA_ARCH);  /* create destination file */
   if (dhandle == -1) {
      error_message(sys_errlist[errno]);
      return -1;
   }

   while ( f_length ) {
      byte_read = _read(shandle,buff,BUFFLEN);
      byte_write = write(dhandle,buff,byte_read);
```

```
      if (byte_write != byte_read) {
         error_message(" Disk Full ");
         return -2;
      }
      if (byte_write == -1) {
         error_message(" Copy Error ");
         return -3;
      }
      f_length -= byte_read;
   }
   close(shandle);
   close(dhandle);
   return 0;
}
```

ฟังก์ชัน play_vdata();

เป็นฟังก์ชันที่ทำหน้าที่ เล่นเสียงไฟล์ที่เก็บไว้ที่ voc_dir directory เพื่อสามารถเปลี่ยนแปลง directory ที่เก็บ message ได้

```
int play_vdata(char *fname)
{
   char fullname[MAXPATH];

   strcpy(fullname,voc_dir);
   strcat(fullname,"\\");
   strcat(fullname,fname);
   play_data(fullname,-1,0);
}
```

ฟังก์ชัน record_vdata();

เป็นฟังก์ชันสำหรับอัดเสียง เพื่อเก็บไว้ที่ voc_dir directory เพื่อให้สามารถเปลี่ยนแปลง directory ที่ใช้เก็บเสียงได้โดยง่าย

```
int record_vdata(char *fname)
{
   char fullname[MAXPATH];

   strcpy(fullname,voc_dir);
   strcat(fullname,"\\");
   strcat(fullname,fname);
   record_data(fullname);
}
```

ฟังก์ชัน delete();

รับ message file เข้าไป และลบไฟล์นั้น โดยจะไปลบไฟล์ที่ voc_dir directory เท่านั้น

```
int delete(char *fname)
{
   char fullname[MAXPATH];

   strcpy(fullname,voc_dir);
   strcat(fullname,"\\");
   strcat(fullname,fname);
   remove(fullname);
}
```

การทำงานของ ฟังก์ชัน ในไฟล์ USER.C

ในไฟล์ USER.C นี้ ประกอบด้วยฟังก์ชันสำคัญ ๆ หลายฟังก์ชัน ได้แก่

```
add_user();
delete_user();
update_user();
edit_user();
```

ซึ่งแต่ละฟังก์ชันนี้ ทำการเปลี่ยนแปลงแก้ไข ข้อมูลใน user database ตามแต่ละฟังก์ชันที่กำหนดดังต่อไปนี้

ฟังก์ชัน add_user();

เป็นฟังก์ชันที่ทำหน้าที่ เพิ่ม user ใน user database โดยเรียกฟังก์ชัน data_entry และส่ง parameter เป็น template ของ user เข้าไป จากนั้น เมื่อ กด button " Add User " ก็จะนำ template ของ user นั้น มาเปลี่ยนให้อยู่ในรูปของ record ของ user คือ ส่วนที่เป็น radio box จะถูกเปลี่ยนให้เป็น code และส่วนที่เป็น time of mail box จะถูกเปลี่ยนเป็น string บอกเวลาที่เหลืออยู่ใน mail box เก็บไว้เป็น field หนึ่งใน user record สุดท้าย จะนำ user id เป็น key เพื่อ add เข้าไปใน database ซึ่งเรียกฟังก์ชัน KramAdd(); ซึ่งจะ return ค่ากลับมาเป็น TRUE ถ้าสามารถ add ได้ หรือ FALSE ถ้า key ซ้ำ หรือ add ไม่ได้

```
int add_user()
{
   int temp;
   char tempc[TIMELEFT];

   temp = rad_to_time(time);
```

```
    itoa(temp,tempc,10);
    strcpy(edit_st.time_left,tempc);
    edit_st.time = rad_to_code(time);
    edit_st.service = rad_to_code(service_type);
    if (KramAdd(KramFile,edit_st.id,edit_st.name) == FALSE ) {
       error_message("Add User Error");
       return'\r';
    }
    else {
       init_template();
       return'\t';
    }
}
```

ฟังก์ชัน delete_user();

เป็นฟังก์ชันที่ทำหน้าที่ delete user ออกจาก database โดยจะรับเพียง user id และนำไปเป็น parameter เรียกฟังก์ชัน KramDelete(); ซึ่งจะ return ค่ากลับมาเป็น TRUE ในกรณีที่สามารถลบ user ได้ และ FALSE ในกรณีที่ไม่พบ user ที่ต้องการลบ

```
int delete_user()
{
    if ( KramDelete(KramFile,edit_st.id) == FALSE) {
       error_message("Delete User Error");
       return '\r';
    }
    else {
       init_template();
       return '\t';
```

```
    }
}
```

ฟังก์ชัน update_user();

เป็นฟังก์ชันที่รับค่าของ user template ที่ได้จากฟังก์ชัน data_entry (); เพื่อนำไป update ข้อมูลใน database โดยเมื่อทำการ edit ค่าของ user เสร็จสิ้นแล้ว ฟังก์ชันจะนำ user id ไปค้นหาใน database และเขียนทับเข้าไปด้วยฟังก์ชัน KramUpdate(); ซึ่งจะ return ค่า เป็น TRUE ถ้าสามารถค้นหา user พบ และ update ได้ หรือ FALSE ถ้าหากค้นหาไม่พบ

```
static int update_user()
{
    int t_left;             /* time left in minutes */
    int c_t_left;           /* time left in pack code */
    int time_sel;           /* service time selected */

    t_left = atoi(edit_st.time_left);
    time_sel = rad_to_time(time);
    if ( time_sel > old_time)
       itoa(time_sel-old_time+t_left,edit_st.time_left,10);
    else  {
       if ( t_left>=time_sel)
          itoa(time_sel-old_time+t_left,edit_st.time_left,10);
       else
          strcpy(edit_st.time_left,"0");
    }
    edit_st.time = rad_to_code(time);
    edit_st.service = rad_to_code(service_type);
    if(KramUpdate(KramFile,edit_st.id,edit_st.name) == FALSE) {
```

```
        error_message(" Update User error ");
        return '\r';
    }
    else {
        init_template();
        return '\t';
    }
}
```

ฟังก์ชัน edit_user();

เป็นฟังก์ชันที่รับ user id เข้าไป เพื่อเรียกฟังก์ชัน KramRead(); เพื่อ อ่าน user record ออกมา สำหรับทำการ editing เปลี่ยนแปลง field ต่าง ๆ ของ user record จากนั้น จึงเรียกฟังก์ชัน update_user(); เพื่อเก็บค่าใหม่เข้าไปใน database อีกครั้งหนึ่ง

```
int edit_user()
{
    if(KramRead(KramFile,edit_st.id,edit_st.name) == NULL) {
        error_message("Reading User Error");
        init_template();
        return '\t';
    }
    else {
        code_to_rad(edit_st.time,time);
        code_to_rad(edit_st.service,service_type);
        old_time = rad_to_time(time);
        return '\t';
    }
}
```

```
/*----------------------  MAIN.C  ---------------------*/

#include<stdio.h>
#include<conio.h>
#include<io.h>
#include<fcntl.h>
#include<dir.h>
#include<dos.h>
#include<string.h>
#include<sys\stat.h>
#include<fcntl.h>

#include"..\header\scr.h"
#include"..\header\window.h"
#include"..\header\lldef.h"
#include"..\header\vc.h"
#include"..\header\telphone.h"
#include"..\header\kram.h"
#include"..\header\entry.h"
#include"..\header\mail.h"
#include"..\header\user.h"
#include"..\header\mesad.h"
#include"..\header\return.h"
#include"..\header\telad.h"

extern struct edit edit_st;
extern struct LINKLIST msglist;
extern struct LINKLIST dellist;
extern struct LINKLIST tout_list;
extern FILE *KramFile;
extern int tout_handle;
extern struct tout_struct tout_template;

static int initial();
static int t_end();
static int init_toutlist();
static int save_toutlist();


FIELD main_template[] = {
   {4,3,4,'U', "    User Admin     ",NULL,"Add,Delete,and Change User's Details"},
   {4,5,4,'T', "  Telephone Admin  ",NULL,"Set telephone configuration"},
   {4,7,4,'M', "   Message File    ",NULL,"Manipulate message file"},
   {4,9,4,'S', "     Start EVS     ",NULL,"Start Electronic Voice System"},
   {4,11,4,'Q', "       Quit        ",NULL,"Quit"},
   {0}
};



main()
{

   if (initial() == 0) {
      exit(1);
```

```
    }
    establish_window(5,6,73,20,WHITE,BLUE,1);
    win_dstr(28,4,"Administration for User Data",WHITE|(ENTRYBG<<4));
    win_dstr(28,6,"Set Telephone and PABX configuration",WHITE|(ENTRYBG<<4));
    win_dstr(28,8,"Message File Manipulation",WHITE|(ENTRYBG<<4));
    win_dstr(28,10,"Start Detect Ringing ",
             WHITE|(ENTRYBG<<4));
    win_dstr(28,12,"Exit",WHITE|(ENTRYBG<<4));
    data_entry(main_template,1,1);
    delete_window();
    t_end();
}

int initial()
{
    int telcard_exist;
    int KeyLength,DataLength;
    char FileName[13];

/*-------------------- Initial screen ------------------------*/
    clrscr();
    initscr();
    fill(1,1,80,25,176,0x71);
    fill(3,2,78,4,' ',0x1e);
    frame(3,2,78,4,0x1e);
    write_string(26,3,"ELECTRONIC VOICE SYSTEM",0x1e);

/*--------------- Initial telephone --------------------------*/
    hook_off();
    outp(DTMF_OUT,NO_DTMF);
    delay(100);
    telcard_exist = is_telcard_exist();
    if (telcard_exist != 1) {
       error_message("Telephone Card Not Exist");
       textcolor(LIGHTGRAY);
       textbackground(BLACK);
       clrscr();
       return 0;
    }

/*----------------- init button function -------------------*/
    main_template[0].mask.func = edit;
    main_template[1].mask.func = tel_ad;
    main_template[2].mask.func = message;
    main_template[3].mask.func = mail;
    main_template[4].mask.func = quit;


/*-------------- Initial   Sound Card ------------------------*/
    initvoice();

/*-------------- Initial  Message  List ----------------------*/
    llsetlist(&msglist);
    llzero();
    llsetsize(13);              /* filename in byte */
```

```
/*-------------- Initial  Call out List ---------------------*/
   llsetlist(&tout_list);
   llzero();
   llsetsize(sizeof(struct tout_struct));

/*--------------- Initial Deleted File List -----------------*/
   llsetlist(&dellist);
   llzero();
   llsetsize(sizeof(struct ffblk));

/* ------------- initial database -----------------*/

   strcpy(FileName,"person.dbs");
   if (access(FileName,0) == -1) { /* file does not exist */
      KeyLength =  sizeof(edit_st.id);
      DataLength = sizeof(edit_st) - sizeof(edit_st.id);

      KramInit(FileName,KeyLength,DataLength);
   }
   KramFile = KramOpen(FileName);

/*-------------- Initial call out file ---------------*/
   init_toutlist();

/* -------------- disable ^break ----------------- */
   setcbrk(0);
   return(1);

}


int t_end()
{
   textbackground(DARKGRAY);
   textcolor(LIGHTGRAY);
   clrscr();
   end_voice();
   KramClose(KramFile);
   setcbrk(1);                    /* enable ^break */
   save_toutlist();
   return 1;
}

int init_toutlist()
{
   long filelen;
   int i;
   char FileName[13];

   strcpy(FileName,"callout.dta");
   tout_handle = open(FileName,O_CREAT|O_RDWR,S_IREAD|S_IWRITE);
   llsetlist(&tout_list);
   filelen = filelength(tout_handle);
   for(i=0;i<filelen/(sizeof(struct tout_struct));i++) {
```

```
      _read(tout_handle,&tout_template,sizeof(struct tout_struct));
      lladd((char *) &tout_template);
   }
   close(tout_handle);
}

int save_toutlist()
{
   int i,list_len;
   char FileName[13];

   strcpy(FileName,"callout.dta");
   tout_handle = open(FileName,O_TRUNC|O_RDWR);
   llsetlist(&tout_list);
   llhead();
   list_len = ll_length();
   for(i=0;i<list_len;i++) {
      _write(tout_handle,(&tout_list)->clp->item,sizeof(struct tout_struct));
      lldelete();
   }
   close(tout_handle);
}
```

```
/*---------------------------- SCR.C -------------------------*/

#include<dos.h>
#include<string.h>
#include"..\header\scr.h"
#include"..\header\window.h"

static int get_vmode();
static void shade();

static char far *vidram;

extern struct wn wkw;

/*-------------------------------------------------------------*
|  Function      :   initscr                                  |
|  Usage         :   void initscr(void);                      |
|  Description   :   initial videoram for direct video access |
|  Return value  :   None                                     |
*-------------------------------------------------------------*/
void initscr()
{
   int vmode;

   vmode = getvmode();
   if((vmode!= 2) && (vmode!=3) && (vmode!=7))
      exit(1);
   vidram = (vmode == 7) ? (char far *) 0xb0000000 :
                           (char far *) 0xb8000000 ;
}

/*-------------------------------------------------------------*
|  Function     :  getvmode                                   |
|  Usage        :  static int getvmode();                     |
|  Description  :  get current video mode                     |
|  Return       :  current video mode                         |
*-------------------------------------------------------------*/
static int getvmode()
{
   union REGS reg;

   reg.h.ah = 15;                                /* int 10h, service 15 */
   return int86(0x10,&reg,&reg) & 255;
}


/*-------------------------------------------------------------*
|  Function     :  Block                                      |
|  Usage        :  void block(int startx,int starty,int endx, |
|                              int endy,int attrib);          |
|  Description  :  make shaded window of specific style       |
|  Return       :  None                                       |
*-------------------------------------------------------------*/
void block(int startx,int starty,int endx,int endy,int attrib)
{
```

```
    int xscale = 3;
    int yscale = 1;
    register int i,j;
    int x_len,y_len;
    int x_mid,y_mid;

    x_len = endx-startx;   y_len = endy-starty;
    x_mid = startx+(x_len/2);      y_mid = starty+(y_len/2);

    for(i=xscale+3,j=yscale;(i*2)<x_len && (j*2)<y_len;i+=(xscale*2),j+=(yscale*2)) {
        fill(x_mid-i,y_mid-j,x_mid+i,y_mid+j,' ',attrib);
        frame(x_mid-i,y_mid-j,x_mid+i,y_mid+j,attrib);
        delay(50);
    }
    fill(startx,starty,endx,endy,' ',attrib);
    frame(startx,starty,endx,endy,attrib);
    shade(startx,starty,endx,endy);
}

/*-------------------------------------------------------------*
|  Function     :  Dblock                                     |
|  Usage        :  void dblock(int startx,int starty,int endx,|
|                              int endy,int attrib);          |
|  Description  :  make shaded window of specific style,      |
|                  with double line frame                     |
|  Return       :  None                                       |
*-------------------------------------------------------------*/
void dblock(int startx,int starty,int endx,int endy,int attrib)
{
    int xscale = 3;
    int yscale = 1;
    register int i,j;
    int x_len,y_len;
    int x_mid,y_mid;

    x_len = endx-startx;   y_len = endy-starty;
    x_mid = startx+(x_len/2);      y_mid = starty+(y_len/2);

    for(i=xscale+3,j=yscale;(i*2)<x_len &&
        (j*2)<y_len;i+=(xscale*2),j+=(yscale*2)) {
        fill(x_mid-i,y_mid-j,x_mid+i,y_mid+j,' ',attrib);
        dframe(x_mid-i,y_mid-j,x_mid+i,y_mid+j,attrib);
        delay(50);
    }
    fill(startx,starty,endx,endy,' ',attrib);
    dframe(startx,starty,endx,endy,attrib);
    shade(startx,starty,endx,endy);
}

/*-------------------------------------------------------------*
|  Function     :  shade                                      |
|  Usage        :  void shade(int x,int y,int endx,int endy);  |
|  Description  :  make shade of the specified block          |
|  Return       :  None                                       |
*-------------------------------------------------------------*/
```

```
static void shade(int x,int y,int endx,int endy)
{
   register int i,j;
   unsigned char far *vptr;
   char far *temp;

   temp = vidram;
   for(j=endy;j<endy+1;j++)
      for(i=x+1;i<=endx;i++) {
         vptr = temp + (j*160) + i*2;
         *(++vptr) = LOW_INTENSE;
      }
   for(j=y;j<endy+1;j++)
      for(i=endx;i<=endx+1;i++) {
         vptr = temp + (160*j) + (i*2);
         *(++vptr) = LOW_INTENSE;
      }
}


/*-------------------------------------------------------------*
|  Function     :    frame                                     |
|  Usage        :    void frame(int startx,int starty,         |
|                              int endx,int endy);            |
|  Description  :    make frame of specified block             |
|                    The top left corner is (1,1)              |
|  Return value :    None                                      |
*-------------------------------------------------------------*/
void frame(int startx,int starty,int endx,int endy,int attrib)
{
   register int i;
   unsigned char far *vptr, far *temp;


   temp = vidram;
   for(i=startx;i<(endx-1);i++) {
      vptr = temp + ((starty-1)*160) + i*2;
      *vptr++ = 196;
      *vptr = attrib;
      vptr = temp + ((endy-1)*160) + i*2;
      *vptr++ = 196;
      *vptr = attrib;
   }
   for(i=starty;i<(endy-1);i++) {
      vptr = temp + (i*160) + (startx-1)*2;
      *vptr++ = 179;
      *vptr = attrib;
      vptr = temp + (i*160) + (endx-1)*2;
      *vptr++ = 179;
      *vptr = attrib;
   }
   write_char(startx,starty,218,attrib);
   write_char(startx,endy,192,attrib);
   write_char(endx,starty,191,attrib);
   write_char(endx,endy,217,attrib);
```

```
}

/*-------------------------------------------------------------*
|  Function     :    dframe                                    |
|  Usage        :    void dframe(int startx,int starty,        |
|                                int endx,int endy);           |
|  Description  :    make frame of specified block             |
|                    The top left corner is (1,1)              |
|  Return       :    None                                      |
*-------------------------------------------------------------*/
void dframe(int startx,int starty,int endx,int endy,int attrib)
{
   register int i;
   unsigned char far *vptr, far *temp;

   temp = vidram;
   for(i=startx;i<(endx-1);i++) {
      vptr = temp + ((starty-1)*160) + i*2;
      *vptr++ = 205;
      *vptr = attrib;
      vptr = temp + ((endy-1)*160) + i*2;
      *vptr++ = 205;
      *vptr = attrib;
   }
   for(i=starty;i<(endy-1);i++) {
      vptr = temp + (i*160) + (startx-1)*2;
      *vptr++ = 186;
      *vptr = attrib;
      vptr = temp + (i*160) + (endx-1)*2;
      *vptr++ = 186;
      *vptr = attrib;
   }
   write_char(startx,starty,201,attrib);
   write_char(startx,endy,200,attrib);
   write_char(endx,starty,187,attrib);
   write_char(endx,endy,188,attrib);
}


/*-------------------------------------------------------------*
|  Funtion      :  write_char                                  |
|  Usage        :  write_char(int x,int y,char ch,int attrib); |
|  Description  :  write character to by direct  video access  |
|  Return       :  None                                        |
*-------------------------------------------------------------*/
void write_char(int x,int y,char ch,int attrib)
{
   char far *vptr;

   vptr = vidram;
   vptr += ((y-1)*160) + (x-1)*2;
   *vptr++ = ch;
   *vptr = attrib;
}
```

```
/*-------------------------------------------------------------*
¦  Function     :  fill                                        ¦
¦  usage        :  void  fill(int startx,int starty,int endx,  ¦
¦                            int endy,int style,int attrib);  ¦
¦  Description  :  Fill specific block with specific style,    ¦
¦                  top left of screen is  (1,1)               ¦
¦  Return       :  None                                        ¦
*-------------------------------------------------------------*/
void fill(int startx,int starty,int endx,int endy,int style,int attrib)
{
   register int i,j;
   unsigned char far *vptr, far *temp;

   temp = vidram;
   for(j=starty-1;j<endy;j++)
      for(i=startx-1;i<endx;i++) {
         vptr = temp + (j*160) + i*2;
         *vptr++ = style;
         *vptr = attrib;
      }
}


/*-------------------------------------------------------------*
¦  Function     :  write_string                                ¦
¦  Usage        :  write_string(int x,int y,char *s,           ¦
¦                               int attrib)                   ¦
¦  Description  :  write string to videoram                    ¦
¦                       x,y      : coordinate to write       ¦
¦                       char ¦s  : pointer to string         ¦
¦                       attrib   : attribute of string       ¦
¦  Return       :     None                                     ¦
*-------------------------------------------------------------*/
void write_string(int x,int y,char *z,int attrib)
{
   register int i;
   char far *vptr;

   vptr = vidram;
   vptr += ((y-1)*160) + (x-1)*2;
   for(i=x;*z;i++) {
      *vptr++ = *z++;
      *vptr++ = attrib;
   }
}


/*-------------------------------------------------------------*
¦   Function    :  reverse();                                  ¦
¦   Usage       :  int reverse(int x,int y,int length,int attr ¦
¦   Description :  reverse attribute at x,y for length long    ¦
¦   Return      :  None                                        ¦
*-------------------------------------------------------------*/
```

```
void reverse(int x,int y,int length,int attr)
{
   register int i;
   char far *vptr;

   vptr = vidram;
   vptr += ((y-1)*160) + (x-1)*2;
   for(i=0;i<length;i++) {
     *(++vptr) = attr;
     vptr++;
   }
}
```

```
/*-------------------------------------------------------------*
¦   Function    : button();                                     ¦
¦   Usage       : void button(int x,int y,char *str,int fg,     ¦
¦                             int bg,int hk);                   ¦
¦   Description : place button at position x,y                  ¦
¦   Return      : None                                          ¦
*-------------------------------------------------------------*/
void button(int x,int y,char *str,int fg,int bg,char hk)
{
   int len;
   int hk_pos;

   len = strlen(str);
   write_string(x,y,str,fg¦(bg<<4));
   hk_pos = strchr(str,hk) - str;
   write_char(x+hk_pos,y,hk,HOTKEYFG¦(HOTKEYBG<<4));
   buttonshade(x,y,len);
}
```

```
/*-------------------------------------------------------------*
¦   Function    : buttoneshade();                               ¦
¦   Usage       : void buttonshade(int x,int y,int len);        ¦
¦   Description : display shade of button at x,y                ¦
¦   Return      : None                                          ¦
*-------------------------------------------------------------*/
void buttonshade(int x, int y, int len)
{
   register int i;

   for(i=1;i<=len;i++)
      write_char(x+i,y+1,223,BLACK¦(wkw.bg<<4));

   write_char(x+i-1,y,220,BLACK¦(wkw.bg<<4));
}
```

```
/* ----------------- window.c -------------------- */

#include <stdio.h>
#include <alloc.h>
#include <string.h>
#include <conio.h>
#include <mem.h>
#include <dos.h>
#include <stdlib.h>
#include "..\header\window.h"
#include "..\header\scr.h"

int editing;
static union REGS rg;

/* ------------ window definition structure -------------- */
struct  wn  wdo[MAX_WINDOWS];
int  curr_wnd;          /* current window */
struct wn wkw;          /* a working window structure */

static void upline(void);
static void downline(void);
static void firstline(void);
static void lastline(void);
static void dline(int,int,int);

/*-------------------------------------------------------------*
¦  Function     :  establish_window                            ¦
¦  Usage        :  void establish_window(left,top,right,       ¦
¦                  bottom,foreg,backg,save);                  ¦
¦  Description  :  building window                             ¦
¦                    left,top     :  top left corner         ¦
¦                    right,bottom :  bottom right corner     ¦
¦                    foreg,backg  :  foreground,background   ¦
¦                                    attribute              ¦
¦                    save         :  flag to save background ¦
¦                                    0  =  not save         ¦
¦                                    1  =  save             ¦
¦  Return       :  None                                       ¦
*-------------------------------------------------------------*/

void establish_window(left,top,right,bottom,foreg,backg,save)
{
   if(curr_wnd < MAX_WINDOWS)  {
      if(curr_wnd)
         wdo[curr_wnd-1] = wkw;
      setmem(&wkw,sizeof(wkw),0);
      wkw.lf = left;
      wkw.tp = top;
      wkw.rt = right;
      wkw.bt = bottom;
      wkw.fg = foreg;
      wkw.bg = backg;
      wkw.wd = right+1-left;
      wkw.ht = bottom-top+1;
```

```
      if (save)   {
         if ((wkw.wsave=malloc((wkw.ht+1)*(wkw.wd+2)*2)) == NULL)
            return;
         gettext(left,top,right+2,bottom+1,wkw.wsave);
      }
      wdo[curr_wnd++] = wkw;
      block(left,top,right,bottom,foreg|(backg<<4));
      current_window();
   }
}

/*-------------------------------------------------------------*
|  Function     :  current_window                             |
|  Usage        :  void current_window();                     |
|  Description  :  set current window to active               |
|  Return       :  None                                       |
*-------------------------------------------------------------*/
void current_window()
{
   window(wkw.lf,wkw.tp,wkw.rt,wkw.bt);
   hidecursor();
   if (wkw.fg || wkw.bg) {
          textcolor(wkw.fg);
          textbackground(wkw.bg);
   }
}


/*-------------------------------------------------------------*
|  Function     :  window_title                               |
|  Usage        :  void window_title(char *ttl,int attrib)    |
|  Description  :  write title string *ttl to current         |
|                  window, attribute specify to attrib        |
|  Return       :  None                                       |
*-------------------------------------------------------------*/
void window_title(char *ttl,int attrib)
{
   win_dstr((wkw.wd-strlen(ttl)) /2,1,ttl,attrib);
}


/*-------------------------------------------------------------*
|  Function     :  delete_window                              |
|  Usage        :  void delete_window();                      |
|  Description  :  remove current window                      |
|  Return       :  None                                       |
*-------------------------------------------------------------*/
void delete_window()
{
   if (curr_wnd)   {
      if (wkw.wsave)   {
         puttext(wkw.lf,wkw.tp,wkw.rt+2,wkw.bt+1,wkw.wsave);
         free(wkw.wsave);
      }
      setmem(wdo+curr_wnd-1, sizeof (struct wn),0);
```

```
      --curr_wnd;
      if (curr_wnd) {
         wkw = wdo[curr_wnd-1];
         current_window();
      }
      else  window(1,1,80,25);
   }
}

/*-------------------------------------------------------------*
|  Function     :  scroll_window                              |
|  Usage        :  void scroll_window(int d)                  |
|  Description  :  scroll the window direction up or down     |
|                        d = 1  :  up                         |
|                        d = 0  : down                        |
|  Return       :  None                                       |
*-------------------------------------------------------------*/
void scroll_window(d)
{
      movetext(wkw.lf+1, wkw.tp+1+d, wkw.rt-1, wkw.bt-2+d,
               wkw.lf+1, wkw.tp+2-d);
}


/* ---------- display text in a window ---------------- */
void text_window(char *txt[], int ln)
{
   int height = wkw.ht;

   wkw.wtext = txt;
   wkw.wtop = ln;
   wkw.wy = 1;
   while (height-- && txt[ln-1])
      dline(ln++,wkw.fg, wkw.bg);
   wkw.wlines = 0;
   while (*txt++)
      wkw.wlines++;
}

static int lineno;

/* ------------ page and scroll through a window of text ---------- */
int select_window (int ln,int foreg,int backg,int (*func)(int,int))
{
   int c = 0;
   int frtn;
   int height, dln, ptop;

   if(ln> wkw.wtop + wkw.ht-1 ;; ln < wkw.wtop)
      text_window(wkw.wtext, ln);
   else
      wkw.wy = ln - wkw.wtop + 1;
   while (TRUE) {
      lineno = wkw.wtop + wkw.wy - 1;
      ptop = wkw.wtop;
```

```
dline(lineno,foreg, backg);
if(wkw.wx == 0)
   hidecursor();
else
   gotoxy(wkw.wx, wkw.wy+1);
c = getkey();
if (c == '\r' || c == ESC)
   break;
switch (c)   {
   case  CTRL_HOME :
      firstline();
      break;
   case  CTRL_END  :
      lastline();
      break;
   case  PGUP :
      wkw.wtop -= wkw.ht;
      if (wkw.wtop > wkw.wlines - (wkw.ht-1))
         wkw.wtop = wkw.wlines - (wkw.ht-1);
      if (wkw.wtop < 1)
         wkw.wtop = 1;
      break;
   case  UP :
      upline();
      break;
   case  DN :
      downline();
      break;
   default :
      if (!editing && wkw.wlines <= wkw.ht)  {
         if (c == HOME) {
            firstline();
            break;
         }
         if (c == END) {
            lastline();
            break;
         }
      }
      if (func)  {
         frtn = (*func)(c,lineno);
         if (frtn == ERROR)
            putch(BELL);
         else if (frtn) {
            wkw.wy = frtn;
            return  frtn;
         }
         c = 0;
      }
      break;
   }
switch (c)  {
   case  HOME      :
   case  CTRL_HOME :
   case  END       :
```

```
        case  CTRL_END  :
        case  PGUP      :
        case  PGDN      :  if (wkw.wtop != ptop)   {
                              height = wkw.ht;
                              dln = wkw.wtop;
                              while (height-- && wkw.wtext[dln-1])
                                 dline(dln++, wkw.fg, wkw.bg);
                                 break;
                           }
        default         :   dline(lineno, wkw.fg, wkw.bg);
                               break;

      }
   }
   return c == ESC ? 0 : lineno;
}

/*-------------------------------------------------------------*
¦  Function     :  upline                                      ¦
¦  Usage        :  static void upline();                       ¦
¦  Description  :  move up one line                            ¦
¦  Return       :  None                                        ¦
*-------------------------------------------------------------*/
static void upline()
{
   if (lineno > 1) {
      if (wkw.wy == 1)   {
         if (wkw.wtop > 1)   {
            --wkw.wtop;
            scroll_window(0);
         }
      }
      else
         --wkw.wy;
   }
   else if (wkw.wlines <= wkw.ht)
      lastline();
}

/*-------------------------------------------------------------*
¦  Function     :  downline                                    ¦
¦  Usage        :  static void downline();                     ¦
¦  Description  :  move down one line                          ¦
¦  Return       :  None                                        ¦
*-------------------------------------------------------------*/
static void downline()
{
   if (lineno < wkw.wlines)   {
      if (wkw.wy == wkw.ht)   {
         scroll_window(1);
         wkw.wtop++;
      }
      else
         wkw.wy++;
   }
```

```
   else  if (wkw.wlines <= wkw.ht)
      firstline();
}

/*-------------------------------------------------------------*
|  Function     :  firstline                                  |
|  Usage        :  static void firstline();                   |
|  Description  :  move to the first  line                    |
|  Return       :  None                                       |
*-------------------------------------------------------------*/
static void firstline()
{
   wkw.wtop = wkw.wy = 1;
}

/*-------------------------------------------------------------*
|  Function     :  lastline                                   |
|  Usage        :  static void lastline();                    |
|  Description  :  move to the last line                      |
|  Return       :  None                                       |
*-------------------------------------------------------------*/
static void lastline()
{
   wkw.wtop = wkw.wlines - (wkw.ht-1);
   if (wkw.wtop <1)
      wkw.wtop = 1;
   wkw.wy = wkw.ht;
   if (wkw.wy > wkw.wlines)
      wkw.wy = wkw.wlines;
}

char spaces[80] =
"                                                                              ";
/* -------- display a line of text, highlight or normal ---------*/
static void dline(ln, foreg, backg)
{
   if (foreg || backg)  {
      textcolor(foreg);
      textbackground(backg);
      --ln;
      win_dstr(2, ln-wkw.wtop+3, *(wkw.wtext + ln),foreg|(backg<<4));
      if (strlen(*(wkw.wtext + ln)) < wkw.wd-2)
         win_dstr(2+strlen(*(wkw.wtext + ln)),
           ln-wkw.wtop+3,
           (spaces + 79 - wkw.wd +
           strlen(*(wkw.wtext + ln)) + 2),foreg|(backg<<4));
   }
}

/*-------------------------------------------------------------*
|  Function     :  hidecursor                                 |
|  Usage        :  void hidecursor();                         |
|  Description  :  use BIOS to hide the cursor                |
|  Return       :  None                                       |
*-------------------------------------------------------------*/
```

```
void hidecursor()
{
   rg.h.ah = 2;
   rg.x.dx = 0x1900;
   rg.h.bh = 0;
   int86(0x10,&rg,&rg);
}


/*-------------------------------------------------------------*
¦  Function     :  set_cursor_type                             ¦
¦  Usage        :  void set_cursor_type(unsigned t)            ¦
¦  Description  :  use BIOS to set the cursor type            ¦
¦  Return       :  None                                        ¦
*-------------------------------------------------------------*/
void set_cursor_type(unsigned t)
{
   rg.x.ax = 0x0100;
   rg.x.bx = 0;
   rg.x.cx = t;
   int86(0x10,&rg,&rg);
}


void (*helpfunc)(void);
int helpkey;

/*-------------------------------------------------------------*
¦  Function     :  getkey                                      ¦
¦  Usage        :  int getkey();                               ¦
¦  Description  :  read the keyboard                           ¦
¦  Return       :  keyboard read                               ¦
*-------------------------------------------------------------*/
int getkey()
{
   int c;

   if ((c = getch()) == 0 )
      c = getch() ¦ 128;
   if (c == helpkey && helpfunc) {
      (*helpfunc)();
      c = getkey();
   }
   return c;
}

/*-------------------------------------------------------------*
¦   Function    : error_message();                            ¦
¦   Usage       : void error_message(char *ermsg)             ¦
¦   Description : write an error message                      ¦
¦   Return      : None                                        ¦
*-------------------------------------------------------------*/
void error_message(char *ermsg)
{
   int esc_pos;
```

```
   int lf = (80-strlen(ermsg)+2) / 2;
   int rt = lf+max(strlen(ermsg)+1,15);
   esc_pos = (rt-lf-strlen("(Press [Esc])")+2)/2;
   establish_window(lf,11,rt,14,WHITE,RED,TRUE);
   window_title(" Error ",WHITE|(RED<<4));
   gotoxy(2,2);
   cputs(ermsg);
   gotoxy(esc_pos,3);
   cputs("(Press [Esc])");
   hidecursor();
   do
      putch(BELL);
   while (getkey() != ESC);
   delete_window();
}


/*-------------------------------------------------------------*
|  Function    : win_dstr                                     |
|  Usage       : void win_dstr(int x,int y,char *str,int attr); |
|  Description : write string pos according to active window  |
|                top left corner is  1,1                      |
|  Return      : None                                         |
*-------------------------------------------------------------*/
void win_dstr(int x,int y,char *str,int attr)
{
   write_string(wkw.lf+x-1,wkw.tp+y-1,str,attr);
}

/*-------------------------------------------------------------*
|   Function    : win_reverse();                              |
|   Usage       : void win_reverse(int x,int y,int length,    |
|                                  int attr);                 |
|   Description : specify attribute at x,y for length "length" |
|   Return      : None                                        |
*-------------------------------------------------------------*/
void win_reverse(int x,int y,int length,int attr)
{
   reverse(wkw.lf+x-1,wkw.tp+y-1,length,attr);
}

/*-------------------------------------------------------------*
|  Function     :  clear_window                               |
|  Usage        :  void clear_window                          |
|  Description  :  clear the current window                   |
|  Return       :  None                                       |
*-------------------------------------------------------------*/
void clear_window()
{
   fill(wkw.lf+1,wkw.tp+1,wkw.rt-1,wkw.bt-1,' ',wkw.fg|(wkw.bg<<4));
}
```

```
/* -------------- entry.c ----------------- */

#include<stdio.h>
#include<ctype.h>
#include<stdlib.h>
#include<string.h>
#include<conio.h>
#include<dos.h>
#include "..\header\window.h"
#include "..\header\entry.h"
#include "..\header\scr.h"

#define   FIELDCHAR '_'
int inserting = INSERTING;          /* insert mode , TRUE/FALSE */

extern struct wn wkw;

/* ------------------- local prototypes ------------- */
static void addfield(FIELD *);
static void data_value(FIELD *);
static int  endstroke(int);
static int inp_field(int c);

/* ----------- function for character field ------------ */
static void disp_field(FIELD *, char *, char *);
static int  read_field(int);
static void home_cursor(void);
static int  backspace(void);
static void end_cursor(void);
static void forward(void);
static int  fore_word(void);
static int  back_word(void);
static void delete_char(void);
static void delete_word(void);

/* ------- function for radio box ---------------- */
static void init_rad(RADIO *rad,int init);
static void next_rad(void);
static void prev_rad(void);
static void first_rad(void);
static void last_rad(void);
static void mark_rad(void);
static void disp_rad(void);
static int read_rad(int c);
static void rad_desc(RADIO *radin);

/*------------ function for button ---------------*/
static int read_bt(int c);
static void bt_press(void);
static void active_bt(void);
static int button_call(int  *c);
static int is_hotkey(char c);


static FIELD *fhead;
```

```
static FIELD *ftail;
FIELD *fld;

static RADIO *rhead;
static RADIO *rtail;
RADIO *wrad;


/* %%%%%%%%%%%%%%%%%%%% group of function for button %%%%%%%%%%%%%%%%%%% */

/*-------------------------------------------------------------*
|   Function    : read_bt(c)                                  |
|   Usage       : int read_bt(c);                             |
|   Description : wait for tab to next button or hot key      |
|   Return      : key press                                   |
*-------------------------------------------------------------*/
static int read_bt(c)
{
   int done = FALSE;
   int but_no;

   while ( TRUE ) {
      hidecursor();
      active_bt();              /* display current active button */
      c = getkey();
      switch(c) {
         case '\r' :
         case '\n' : done = button_call(&c);    /* Enter pressed; call function */
                     break;

         default   : if ((but_no = is_hotkey(c)) != -1) {
                        /* ---------- Hot key pressed ------------ */
                        fld = fhead+but_no;     /* set active field */
                        field_tally();
                        active_bt();            /* display active button */
                        done = button_call(&c); /* call function */
                        break;
                     }
                     if ((c=='\t') || (c== ESC))
                        done = TRUE;
                     break;
      }
      if (done)
         break;
   }
   return c;
}

/*-------------------------------------------------------------*
|   Function    :  is_hotkey();                               |
|   Usage       :  int is_hotkey(char c);                     |
|   Description :  Check whether key pressed is hot key       |
|   Return      :  -1 : not hot key                           |
|                  field no. : in case of hot key             |
*-------------------------------------------------------------*/
```

```
static int is_hotkey(char c)
{
   FIELD *fd;
   int i;

   fd = fhead; i = 0;
   while(fd != ftail) {
      if(fd->type == 4) {              /* button field */
         if(c == tolower(fd->fx))
            return i;                  /* return field no. */
      }
      fd++; i++;
   }
   return -1;                          /* not hot key */
}


/*-------------------------------------------------------------*
|   Function    :  button_call();                             |
|   Usage       :  int button_call(int *c);                   |
|   Description :  call function of current button            |
|   Return      :  return code of function called             |
*-------------------------------------------------------------*/
static int button_call(int *c)
{
   FIELD *oldfhead;
   FIELD *oldftail;
   FIELD *oldfld;

   RADIO *oldrhead;
   RADIO *oldrtail;
   RADIO *oldwrad;
   int temp;            /* store button function returned value */
   int done;

   bt_press();          /* display button being pressed */

   oldfhead = fhead;            /* store old f and r */
   oldftail = ftail;
   oldfld = fld;
   oldrhead = rhead;
   oldrtail = rtail;
   oldwrad = wrad;

   temp = (*fld->mask.func)();  /* call function of current button */
   if (temp == OK) {
      done = TRUE;
      *c = OK;
   }
   if (temp == CANCEL) {
      done = TRUE;
      *c = ESC;
   }
   if (temp == '\t') {
      done = TRUE;
```

```
    }

    fhead = oldfhead;            /* retrieve old f and r */
    ftail = oldftail;
    fld =  oldfld;
    rhead = oldrhead;
    rtail = oldrtail;
    wrad = oldwrad;
    return done;

}

/*-------------------------------------------------------------*
|   Function    :  bt_press();                                |
|   Usage       :  void bt_press(void);                       |
|   Description :  display current button being pressed       |
|   Return      :  None                                       |
*-------------------------------------------------------------*/
static void bt_press(void)
{
   int x,y;
   int len;
   long i;

   len = strlen(fld->fvalue.fbuff);

   x = fld->fcol+wkw.lf;
   y = fld->frow+wkw.tp;

   fill(x,y,x+len+1,y+1,' ',wkw.fg|(wkw.bg<<4));
   write_string(x+1,y,fld->fvalue.fbuff,ACTBUTFG | (ACTBUTBG<<4));
   for(i=0;i<20000;i++) {    /* delay for button pressed */
   }
   active_bt();
}


/*-------------------------------------------------------------*
|   Function    :  active_bt();                               |
|   Usage       :  void active_bt(void);                      |
|   Description :  display current button in active mode      |
|   Return      :  None                                       |
*-------------------------------------------------------------*/
static void active_bt(void)
{
   int len;
   int x,y;

   len = strlen(fld->fvalue.fbuff);
   x = fld->fcol+wkw.lf;
   y = fld->frow+wkw.tp;
   write_string(x,y,fld->fvalue.fbuff,ACTBUTFG|(ACTBUTBG<<4));
   buttonshade(x,y,len);
   write_char(x,y,16,BLACK | (ACTBUTBG<<4));        /* left  arrow */
   write_char(x+len-1,y,17,BLACK | (ACTBUTBG<<4));    /* right arrow */
```

```
}

/* ********* group of functions deal with radio box ********** */

/*-------------------------------------------------------------*
|   Function    :  rad_to_code();                             |
|   Usage       :  int rad_to_code(RADIO *rad);               |
|   Description :  convert data in radio form to pack code    |
|   Return      :  packed code                                |
*-------------------------------------------------------------*/
int rad_to_code(RADIO *rad)
{
   int temp,i;

   temp = 0;
   for(i=0;rad->row;i++,rad++)
      temp |= (rad->value)<<i;
   return temp;
}



/*-------------------------------------------------------------*
|   Function    :  code_to_rad();                             |
|   Usage       :  void code_to_rad(int int_in,RADIO *rad);   |
|   Description :  convert packed code to data in radio form  |
|   Return      :  None                                       |
*-------------------------------------------------------------*/
void code_to_rad(int int_in,RADIO *rad)
{
   int i;

   for(i=0;rad->row;i++,rad++)
      rad->value = (int_in>>i) & 1;
}


/*-------------------------------------------------------------*
|   Function    : init_rad();                                 |
|   Usage       : void init_rad(RADIO *rad,int init);         |
|   Description : set rhead, rtail and wrad to group of radio |
|                 box of rad                                  |
|                 if init = 1, radio value is set to 0        |
|   Return      : None                                        |
*-------------------------------------------------------------*/
static void init_rad(RADIO *rad,int init)
{
   rhead = rtail = wrad = rad;
   while ( rtail->row )
      rtail++;                                  /* set rtail */

   if(init) {
      while ( rad->row ) {
         rad->value = 0;
```

```
            rad++;
         }
      }
   }

/*-------------------------------------------------------------*
|   Function    :  rad_desc();                                |
|   Usage       :  void rad_desc(RADIO *radin);               |
|   Description :  display radio box description              |
|   Return      :  None                                       |
*-------------------------------------------------------------*/
static void rad_desc(RADIO *radin)
{
   RADIO *temp;

   temp = radin;
   while(temp->row != NULL) {
      win_dstr(temp->col+5,temp->row+1,temp->rad_des,RADIOFG|(RADIOBG<<4));
      temp++;
   }
}

/*-------------------------------------------------------------*
|   Function    :  next_rad();                                |
|   Usage       :  void next_rad(void);                       |
|   Description :  set working radio to next box              |
|                  if next radio is tail; set to head         |
|   Return      :  None                                       |
*-------------------------------------------------------------*/
static void next_rad(void)
{
   wrad++;
   if (wrad == rtail)
      wrad = rhead;
}

/*-------------------------------------------------------------*
|   Function    : prev_rad();                                 |
|   Usage       : void prev_rad(void);                        |
|   Description : set working radio to previous box           |
|                   if previos is head; set to tail           |
|   Return      : None                                        |
*-------------------------------------------------------------*/
static void prev_rad(void)
{
   if (wrad == rhead)
      wrad = rtail-1;
   else
      wrad--;
}

/*-------------------------------------------------------------*
|   Function    :  first_rad();                               |
```

```
¦   Usage       :  void first_rad(void);                          ¦
¦   Description :  set working radio to first box                 ¦
¦   Return      :  None                                           ¦
*-----------------------------------------------------------------*/
static void first_rad(void)
{
   wrad = rhead;
}


/*-----------------------------------------------------------------*
¦   Function    : last_rad();                                     ¦
¦   Usage       : void last_rad(void);                            ¦
¦   Description : set working radio to last box                   ¦
¦   Return      : None                                            ¦
*-----------------------------------------------------------------*/
static void last_rad(void)
{
   wrad = rtail-1;
}


/*-----------------------------------------------------------------*
¦   Function    : mark_rad();                                     ¦
¦   Usage       : void mark_rad(void);                            ¦
¦   Description : Set value in radio according to radio type      ¦
¦   Return      : None                                            ¦
*-----------------------------------------------------------------*/
static void mark_rad(void)
{
   RADIO *temp;

   temp = wrad;
   switch(fld->type) {
      case 2 :                          /* only one value radio */
               temp = rhead;
               while (temp != rtail) {
                  temp->value = 0;
                  temp++;
               }

               wrad->value = 1;         /* set working radio */
               break;
      case 3 :                          /* multi value radio */
               wrad->value ^= 1;        /* toggle value */
               break;
   }
}

/*-----------------------------------------------------------------*
¦   Function    :  disp_rad();                                    ¦
¦   Usage       :  void disp_rad();                               ¦
¦   Description :  display value of each radio box                ¦
¦   Return      :  None                                           ¦
*-----------------------------------------------------------------*/
```

```
static void disp_rad()
{
   RADIO *temp;

   temp = rhead;
   while ( temp != rtail ) {
      if ( temp->value == 1 ) {
         write_char(temp->col+wkw.lf+1,temp->row+wkw.tp,'X',
                    RADIOFG|(RADIOBG<<4));
         temp++;
      }
      else {
         write_char(temp->col+wkw.lf+1,temp->row+wkw.tp,' ',
                    RADIOFG|(RADIOBG<<4));
         temp++;
      }
   }
}


/*-------------------------------------------------------------*
|   Function    :  read_rad();                                |
|   Usage       :  int read_rad(int c);                       |
|   Description :  wait for marking radio                     |
|   Return      :  ESC or '\t' for next field                 |
*-------------------------------------------------------------*/
static int read_rad(int c)
{
   int done = FALSE;

   first_rad();
   while( TRUE ) {
      gotoxy(wrad->col+2,wrad->row+1);
      c = getkey();
      switch(c) {
         case DN   :
         case FWD  : next_rad();
                     break;
         case UP   :
         case BS   : prev_rad();
                     break;
         case HOME : first_rad();
                     break;
         case END  : last_rad();
                     break;
         case ' '  : mark_rad();
                     break;
         default   : if ((c == '\t') || (c == ESC) ) {
                        done = TRUE;
                     }
                     break;
      }
      if ( done )
         break;
      disp_rad();
```

```
    }
    return c;
}


/*-------------------------------------------------------------*
|   Function    :  rad_template();                            |
|   Usage       :  void rad_template();                       |
|   Description :  initial template for radio box             |
|   Return      :  None                                       |
*-------------------------------------------------------------*/
void rad_template()
{
    FIELD *fldt;
    RADIO *temp;
    int radlen;
    int radnum;
    int templen;

    fldt = fhead;
    while(fldt != ftail) {
        if((fldt->type == 2) || (fldt->type == 3)) {
            temp = fldt->fvalue.value;
            radlen = 0;
            radnum = 0;
            while(temp->row != 0) {
                if ((templen = strlen(temp->rad_des)) > radlen)
                    radlen = templen;
                radnum++;
                temp++;
            }
            fill(fldt->fcol+wkw.lf-1,fldt->frow+wkw.tp+1,
                 fldt->fcol+wkw.lf+radlen+5,fldt->frow+wkw.tp+radnum,
                 ' ',RADIOFG|(RADIOBG<<4));
            win_dstr(fldt->fcol,fldt->frow+1,fldt->mask.fbuff,ENTRYFG|(ENTRYBG<<4));
            rad_desc(fldt->fvalue.value);
            temp = fldt->fvalue.value;
            while(temp->row != 0) {
                write_char(temp->col+wkw.lf,temp->row+wkw.tp,
                           '[',RADIOFG|(RADIOBG<<4));
                write_char(temp->col+2+wkw.lf,temp->row+wkw.tp,
                           ']',RADIOFG|(RADIOBG<<4));
                temp++;
            }
        }
        fldt++;
    }
}


/* ~~~~~~~~~~~~  group of functions for character fields ~~~~~~~~~~~~ */

/*-------------------------------------------------------------*
|   Function    :  disp_field();                              |
|   Usage       :  void disp_field(FIELD *fldv,char *bf,      |
```

```
¦                  char *msk);                                          ¦
¦   Description :  display a data field                                 ¦
¦   Return      :  None                                                 ¦
*-------------------------------------------------------------------*/
static void disp_field(FIELD *fldv, char *bf, char *msk)
{
   char cl[80], *cp = cl;

   while (*msk) {
      *cp++ = (*msk != FIELDCHAR ? *msk : *bf++);
      msk++;
   }
   *cp = '\0';
   win_dstr(fldv->fx + fldv->fcol -1,fldv->frow, cl,FIELDFG¦(FIELDBG<<4));
}


/*-------------------------------------------------------------------*
¦   Function    :  data_value();                                      ¦
¦   Usage       :  void data_value(FIELD *fldv);                      ¦
¦   Description :  display the data value in a field                  ¦
¦   Return      :  None                                               ¦
*-------------------------------------------------------------------*/
static void data_value(FIELD *fldv)
{
   int helplen;
   char blank[80];

   switch ( fldv->type ) {
      case 1 :
               fldv->fx = 1;
               disp_field(fldv, fldv->fvalue.fbuff, fldv->mask.fbuff);
               break;
      case 2 :
      case 3 : init_rad(fldv->fvalue.value,0);
               disp_rad();
               break;
      case 4 :
               button(fldv->fcol+wkw.lf,fldv->frow+wkw.tp,fldv->fvalue.fbuff,
               BUTTONFG,BUTTONBG,fldv->fx);
               break;
   }

   memset(blank,32,80);
   helplen = strlen(fld->fhelp);
   write_string(9,25,fld->fhelp,0x4f);
   write_string(9+helplen,25,blank+8+helplen,0x4f);
}


/*-------------------------------------------------------------------*
¦   Function    :  field_tally();                                     ¦
¦   Usage       :  void field_tally();                                ¦
¦   Description :  display all the fields in a window                 ¦
¦   Return      :  None                                               ¦
```

```
¦                  char *msk);                                          ¦
¦   Description :  display a data field                                 ¦
¦   Return      :  None                                                 ¦
*-------------------------------------------------------------------*/
static void disp_field(FIELD *fldv, char *bf, char *msk)
{
   char cl[80], *cp = cl;

   while (*msk) {
      *cp++ = (*msk != FIELDCHAR ? *msk : *bf++);
      msk++;
   }
   *cp = '\0';
   win_dstr(fldv->fx + fldv->fcol -1,fldv->frow, cl,FIELDFG¦(FIELDBG<<4));
}



/*-------------------------------------------------------------------*
¦   Function    :  data_value();                                      ¦
¦   Usage       :  void data_value(FIELD *fldv);                      ¦
¦   Description :  display the data value in a field                  ¦
¦   Return      :  None                                               ¦
*-------------------------------------------------------------------*/
static void data_value(FIELD *fldv)
{
   int helplen;
   char blank[80];

   switch ( fldv->type ) {
      case 1 :
               fldv->fx = 1;
               disp_field(fldv, fldv->fvalue.fbuff, fldv->mask.fbuff);
               break;
      case 2 :
      case 3 : init_rad(fldv->fvalue.value,0);
               disp_rad();
               break;
      case 4 :
                  button(fldv->fcol+wkw.lf,fldv->frow+wkw.tp,fldv->fvalue.fbuff,BUTTONFG,BUTTO
               break;
   }

   memset(blank,32,80);
   helplen = strlen(fld->fhelp);
   write_string(9,25,fld->fhelp,0x4f);
   write_string(9+helplen,25,blank+8+helplen,0x4f);
}



/*-------------------------------------------------------------------*
¦   Function    :  field_tally();                                     ¦
¦   Usage       :  void field_tally();                                ¦
¦   Description :  display all the fields in a window                 ¦
¦   Return      :  None                                               ¦
```

```
*-------------------------------------------------------------*/
void field_tally()
{
   FIELD *fldt;

   fldt = fhead;
   while (fldt != ftail)  {
      data_value(fldt);
      fldt++;
   }
}


/*-------------------------------------------------------------*
¦   Function    :  init_template();                           ¦
¦   Usage       :  void init_template();                      ¦
¦   Description :  clear a template to all blanks             ¦
¦   Return      :  None                                       ¦
*-------------------------------------------------------------*/
void init_template()
{
   FIELD *fldc;
   char *bf, *msk;

   fldc = fhead;
   while (fldc != ftail) {
      switch ( fldc->type ) {
         case 1 :                              /* character field */
                  bf = fldc->fvalue.fbuff;
                  msk = fldc->mask.fbuff;
                  while (*msk)  {
                     if (*msk == FIELDCHAR)
                        *bf++ = ' ';
                     msk++;
                  }
                  fldc++;
                  break;
         case 2 :                                 /* radio box */
         case 3 :
                  init_rad(fldc->fvalue.value,0);
                  fldc++;
                  break;
         case 4 :                        /* case of button */
                  fldc++;                /* do nothing */
                  break;
      }
   }
}

char *mask, *buff;

/*-------------------------------------------------------------*
¦   Function    : read_field();                               ¦
¦   Usage       : int read_field(c);                          ¦
¦   Description : read a field from the keyboard              ¦
```

```
|   Return      : key press                                          |
*-----------------------------------------------------------------*/
static int read_field(c)
{
   int done = FALSE, first = TRUE;

   buff = fld->fvalue.fbuff;
   home_cursor();
   while(TRUE)      {
      gotoxy(fld->fx+fld->fcol-1, fld->frow);
      if (!c || !first)
         c = getkey();
      first = FALSE;
      switch (c)  {
         case CTRL_D  :  delete_word();
                         break;
         case CTRL_FWD : done = !fore_word();
                         break;
         case CTRL_BS  : done = !back_word();
                         break;
         case HOME     : fld->fx = 1;
                         home_cursor();
                         break;
         case END      : end_cursor();
                         break;
         case FWD      : forward();
                         break;
         case BS       : backspace();
                         break;
         case '\b'     : if (!backspace())
                           break;
                         gotoxy(fld->fx+fld->fcol-1, fld->frow);
         case DEL      : delete_char();
                         disp_field(fld, buff, mask);
                         break;
         case INS      : inserting ^= TRUE;
                         insert_line();
                         break;
         default       : if (endstroke(c)) {
                            done = TRUE;
                            break;
                         }
                         if (inserting)  {
                            memmove(buff+1,buff,strlen(buff)-1);
                            disp_field(fld,buff,mask);
                            gotoxy(fld->fcol+fld->fx-1,fld->frow);
                         }
                         *buff = (char) c;
                         write_char(wkw.lf+fld->fcol+fld->fx-2,
                                    wkw.tp+fld->frow-1,(char) c,
                                    FIELDFG|(FIELDBG<<4));
                         forward();
                         if (!*mask) {
                            fld->fx--;
                            mask--;
```

```
                        buff--;
                     }
                     break;
      }
      if (done || !*mask)
         break;
   }
   wkw.wx = fld->fx+1;
   return c;
}

/*-------------------------------------------------------------*
|   Function    : inp_field();                                 |
|   Usage       : int inp_field(int c);                        |
|   Description : wait for key press in accordance with field  |
|                 type                                         |
|   Return      : key press                                    |
*-------------------------------------------------------------*/
static int inp_field(int c)
{
   switch( fld->type ) {
      case 1 : return(read_field(c));
      case 2 :
      case 3 : init_rad(fld->fvalue.value,0);
               return(read_rad(c));
      case 4 : return(read_bt(c));
   }
}

/*-------------------------------------------------------------*
|   Function    : insert_line();                               |
|   Usage       : void insert_line();                          |
|   Description : set insert/exchange cursor shape             |
|   Return      : None                                         |
*-------------------------------------------------------------*/
void insert_line()
{
   set_cursor_type(inserting ? 0x0407 : 0x0607);
}


/*-------------------------------------------------------------*
|   Function    : home_cursor();                               |
|   Usage       : void home_cursor();                          |
|   Description : home the cursor in the field                 |
|   Return      : None                                         |
*-------------------------------------------------------------*/
static void home_cursor()
{
   buff = fld->fvalue.fbuff+fld->fx-1;
   mask = fld->mask.fbuff+fld->fx-1;
   while (*mask != FIELDCHAR) {
      fld->fx++;
      mask++;
   }
```

```
}

/*-------------------------------------------------------------*
|   Function    : end_cursor();                               |
|   Usage       : void end_cursor();                          |
|   Description : move the cursor to the end of the field     |
|   Return      : None                                        |
*-------------------------------------------------------------*/
static void end_cursor()
{
   fld->fx += strlen(mask)-1;
   buff += strlen(buff)-1;
   mask += strlen(mask)-1;
   while(*buff== ' ')
      if (!backspace())
         break;
   forward();
}

/*-------------------------------------------------------------*
|   Function    : forward();                                  |
|   Usage       : void forward();                             |
|   Description : move the cursor forward one character       |
|   Return      : None                                        |
*-------------------------------------------------------------*/
static void forward()
{
   do {
      fld->fx++;
      mask++;
   } while (*mask && *mask != FIELDCHAR);
   buff++;
}

/*-------------------------------------------------------------*
|   Function    : backspace();                                |
|   Usage       : int backspace();                            |
|   Description : move back one character                     |
|   Return      : TRUE if can move back                       |
|                 FALSE otherwise                             |
*-------------------------------------------------------------*/
static int backspace()
{
   if (buff != fld->fvalue.fbuff) {
      --buff;
      do {
         --mask;
         --(fld->fx);
      } while (*mask != FIELDCHAR);
      return TRUE;
   }
   return FALSE;
}
```

```
/*-------------------------------------------------------------*
|   Function    :  fore_word();                                |
|   Usage       :  int fore_word();                            |
|   Description :  move forward one word                       |
|   Return      :  TRUE if can move forward                    |
|                  FALSE otherwise                             |
*-------------------------------------------------------------*/
static int fore_word()
{
   int ct = 2, test = *buff == ' ';

   while (ct--) {
      while ((*buff == ' ') == test && *mask)
         forward();
      if (!*mask)
         return FALSE;
      if (test)
         break;
      test = !test;
   }
   return TRUE;
}


/*-------------------------------------------------------------*
|   Function    : back_word();                                 |
|   Usage       : int back_word();                             |
|   Description : move backward one word                       |
|   Return      : TRUE if can move back                        |
|                 FALSE otherwise                              |
*-------------------------------------------------------------*/
static int back_word()
{
   int test;

   if (buff == fld->fvalue.fbuff)
      return FALSE;
   if (*(buff-1) == ' ')
      backspace();
   test = *buff == ' ';
   while ((*buff == ' ') == test && buff != fld->fvalue.fbuff)
      backspace();
   if (test)
      while (*buff != ' ' && buff != fld->fvalue.fbuff)
         backspace();
   if (*buff == ' ')
      forward();
   return TRUE;
}

/*-------------------------------------------------------------*
|   Function    : delete_char();                               |
|   Usage       : void delete_char();                          |
|   Description : delete a character                           |
|   Return      : None                                         |
```

```
*-------------------------------------------------------------*/
static void delete_char()
{
   memmove(buff, buff+1, strlen(buff));
   *(buff+strlen(buff)) = ' ';
}


/*-------------------------------------------------------------*
|   Function    : delete_word();                              |
|   Usage       : void delete_word();                         |
|   Description : delete a word                               |
|   Return      : None                                        |
*-------------------------------------------------------------*/
static void delete_word()
{
   int test = *buff == ' ';
   int ln = strlen(buff);

   while((*buff == ' ') == test && ln--)
      delete_char();
   if (!test)
      delete_char();
   disp_field(fld,buff,mask);
}

/*-------------------------------------------------------------*
|   Function    : endstroke();                                |
|   Usage       : int endstroke(int c);                       |
|   Description : test c for an ending keystroke              |
|   Return      : True for c being endstroke                  |
|                 False otherwise                             |
*-------------------------------------------------------------*/
static int endstroke(int c)
{
   switch (c) {
      case '\r' :
      case '\n' :
      case '\t' :
      case ESC  :
      case F1   :
      case F2   :
      case F3   :
      case F4   :
      case F5   :
      case F6   :
      case F7   :
      case F8   :
      case F9   :
      case F10  :
      case PGUP :
      case PGDN :
      case HOME :
      case END  :
      case CTRL_FWD  :
```

```
      case CTRL_BS   :
      case CTRL_HOME :
      case CTRL_END  :
      case UP    :
      case DN    :
         return TRUE;
      default    :
         return FALSE;
   }
}

/* ------------ Process data entry for a screen template. -------- */

int data_entry(FIELD *fldin, int init, int firstfld)
{
   int exitcode = 0, done = FALSE;
   char *help_save;

  textcolor(FIELDFG);
  textbackground(FIELDBG);

   help_save = (char *) malloc(160);    /* save help area at screen bottom */
   gettext(1,25,80,25,help_save);

   write_string(1,25," Help ",0x4f);
   insert_line();
   fhead = ftail = fld = fldin;
   while (ftail->frow)
      ftail++;
   while (firstfld-- && fld != ftail)
      fld++;
   --fld;
   if (init)
      init_template();
   rad_template();
   field_tally();

   /* -------------- collect data from keyboard into screen --------- */
   while (done == FALSE) {
      data_value(fld);
      exitcode = inp_field(0);
      field_tally();
      switch (exitcode) {
         case DN  :
         case '\r':
         case '\t':
         case CTRL_FWD :
         case FWD       : done = (ftail == fhead+1);
                          fld++;
                          if (fld == ftail)
                             fld = fhead;
                          break;
         case CTRL_BS   :
         case UP        : if (fld == fhead)
```

```
                        fld = ftail;
                     --fld;
                     break;
        case CTRL_HOME :
                     fld = fhead;
                     break;
        case CTRL_END  :
                     fld = ftail-1;
                     break;
        default      : if((exitcode == ESC) || (exitcode == CANCEL)) {
                        puttext(1,25,80,25,help_save);
                        return CANCEL;
                     }
                     else
                        if (exitcode == OK)
                           done = TRUE;

                     break;
    }
  }
  puttext(1,25,80,25,help_save);       /* restore help area */
  return OK;
}
```

```
/*----------------------- LLMODULE.C -------------------------*/

#include"..\header\lldef.h"
#include"..\header\window.h"

#include<stdio.h>
#include<mem.h>
#include<stdlib.h>

static struct LINKLIST *list;            /* working link list */

extern struct wn wkw;
static void highlight(int row);
static void ldownline(struct LINKLIST *list);
static void lupline();
static void dlist(struct LINKLIST *list);

/*-------------------------------------------------------------*
¦  Macro        :  moveitem                                    ¦
¦  Usage        :  moveitem(A,B);                              ¦
¦  Description  :  Use this macro as moveitem(from,to)        ¦
¦  Return       :  None                                        ¦
*-------------------------------------------------------------*/

#define moveitem(A,B)   movmem(A,B,list->itemlength)


/*-------------------------------------------------------------*
¦  Function     :  Llsetmatch                                  ¦
¦  Usage        :  void llsetmatch(int (*numatch)();           ¦
¦  Description  :  Set matching function                       ¦
¦  Return       :  None                                        ¦
*-------------------------------------------------------------*/
void llsetmatch(int (*numatch)())
{
   list->match = numatch;
}


/*-------------------------------------------------------------*
¦  Funtion      :  llcheck                                     ¦
¦  Usage        :  int llcheck(char *lookfor);                 ¦
¦  Description  :  Set clp to desired link                     ¦
¦  Return       :  return True if found, False otherwise       ¦
*-------------------------------------------------------------*/
int llcheck(char *lookfor)
{
   int temp;

   for(;;)  {
      temp = (*list->match) (lookfor,list->clp->item);
      if (temp == 0)
         return (1);
      else
         if (temp < 0)
```

```
            return(0);
         else
            if (!llnext())
               return(2);
      }
   }


/*-------------------------------------------------------------*
|  Function     :  llsetlist                                  |
|  Usage        :  void llsetlist(struct LINKLIST *new_list);  |
|  Description  :  Set this module to work with a new list.   |
|  Return       :  None                                       |
*-------------------------------------------------------------*/
void llsetlist(struct LINKLIST *new_list)
{
   list = new_list;
}


/*-------------------------------------------------------------*
|  Function     :  llsetsize                                  |
|  Usage        :  void llsetsize(int size);                  |
|  Description  :  Set the storage requirements for the list  |
|  Return       :  None                                       |
*-------------------------------------------------------------*/
void llsetsize(int size)
{
   list->itemlength = size;
}


/*-------------------------------------------------------------*
|  Function     :  llcrlink                                   |
|  Usage        :  static struct LINKTYPE *llcrlink();        |
|  Description  :  Allocate storage for a link.               |
|  Return       :  Pointer to storage for a link              |
*-------------------------------------------------------------*/
static struct LINKTYPE *llcrlink()
{
   struct LINKTYPE *link;

   link = (struct LINKTYPE *) malloc (sizeof(struct LINKTYPE));
   link->item = malloc(list->itemlength);
   return(link);
}

/*-------------------------------------------------------------*
|  Function     :  llinit                                     |
|  Usage        :  void llinit(char *newitem);                |
|  Description  :  Initialize the structure                   |
|  Return       :  None                                       |
*-------------------------------------------------------------*/
void llinit(char *newitem)
{
```

```
   struct LINKTYPE *llcrlink();

   list->head = list->tail = list->clp = llcrlink();
   list->clp->next = list->clp->previous = NULL;
   moveitem(newitem,list->clp->item);
   list->listlength = 1;
}

/*-------------------------------------------------------------*
|  Function     :  llhead                                     |
|  Usage        :  void llhead();                             |
|  Description  :  Set the CLP to the head of the list.       |
|  Return       :  None                                       |
*-------------------------------------------------------------*/
void llhead()
{
   list->clp = list->head;
}


/*-------------------------------------------------------------*
|  Function     :  llzero                                     |
|  Usage        :  void llzero()                              |
|  Description  :  Initial length of link list to zero        |
|  Return       :  None                                       |
*-------------------------------------------------------------*/
void llzero()
{
   list->head = list->tail = list->clp = NULL;
   list->listlength = 0;
}

/*-------------------------------------------------------------*
|  Function     :  lltail                                     |
|  Usage        :  void lltail();                             |
|  Description  :  Set the Clp to the tail of the list.       |
|  Return       :  None                                       |
*-------------------------------------------------------------*/
void lltail()
{
   list->clp = list->tail;
}


/*-------------------------------------------------------------*
|  Function     :  llnext                                     |
|  Usage        :  int llnext();                              |
|  Description  :  Set the CLP to the next link               |
|  Return       :  return FALSE if at end of list,            |
|                  TRUE otherwise.                            |
*-------------------------------------------------------------*/
int llnext()
{
   if (list->clp->next == NULL )
      return (0);
```

```
   else {
      list->clp = list->clp->next;
      return (1);
   }
}


/*-------------------------------------------------------------*
|  Function     :  llprevious                                  |
|  Usage        :  int llprevious();                           |
|  Description  :  Set the CLP to the previous link            |
|  Return       :  return FALSE if at head of list,            |
|                  TRUE otherwise.                             |
*-------------------------------------------------------------*/
int llprevious()
{
   if (list->clp->previous == NULL)
      return 0;
   else {
      list->clp = list->clp->previous;
      return (1);
   }
}


/*-------------------------------------------------------------*
|  Function     :  llretrieve                                  |
|  Usage        :  void llretrieve(char *newitem)              |
|  Description  :  Retrieve the item from the CLP link.        |
|  Return       :  None                                        |
*-------------------------------------------------------------*/
void llretrieve(char *newitem)
{
   moveitem(list->clp->item, newitem);
}


/*-------------------------------------------------------------*
|  Function     :  lladd                                       |
|  Usage        :  void lladd(char *newitem)                   |
|  Description  :  Add a new link containing this item to      |
|                  the link following the CLP, and reset       |
|                  CLP to new link.                            |
|  Return       :  None                                        |
*-------------------------------------------------------------*/
void lladd(char *newitem)
{
   struct LINKTYPE *newlink;
   struct LINKTYPE *llcrlink();

/* If empty,  initialize list.
   --------------------------- */
   if (ll_length() == 0) {
      llinit(newitem);
      return;
```

```
   }

/* Create new link.
   ----------------- */
   newlink = llcrlink();
   moveitem(newitem, newlink->item);
   list->listlength++;

/* Reset pointers.
   ----------------- */
   newlink->next = list->clp->next;
   newlink->previous = list->clp;
   if (list->tail == list->clp)
      list->tail = newlink;
   else
      list->clp->next->previous = newlink;
   list->clp->next = newlink;
   list->clp = newlink;
}

/*-------------------------------------------------------------*
¦  Function     :  lladdhead                                    ¦
¦  Usage        :  void lladdhead(char *newitem)                ¦
¦  Description  :  Add a new head, reset CLP.                   ¦
¦  Return       :  None                                         ¦
*-------------------------------------------------------------*/
void lladdhead(char *newitem)
{
   struct LINKTYPE *newlink;
   struct LINKTYPE *llcrlink();

/* If empty,  initialize list.
   --------------------------- */
   if (ll_length() == 0) {
      llinit(newitem);
      return;
   }
/* Create new link.
  ------------------- */
   newlink = llcrlink();
   moveitem(newitem,newlink->item);
   list->listlength++;

/* Reset pointers.
   ------------------- */
   newlink->previous = NULL;
   newlink->next = list->head;
   list->head->previous = newlink;
   list->clp = list->head = newlink;
}


/*-------------------------------------------------------------*
¦   Function     :  lladdtail();                                ¦
¦   Usage        :  int lladdtail(char *item);                  ¦
```

```
¦   Description :  add item to tail of list                    ¦
¦   Return      :  None                                        ¦
*-------------------------------------------------------------*/
void lladdtail(char *newitem)
{
   struct LINKTYPE *newlink;
   struct LINKTYPE *llcrlink();

/* If empty,  initialize list.
   --------------------------- */
   if (ll_length() == 0) {
      llinit(newitem);
      return;
   }
/* Create new link.
  ------------------- */
   newlink = llcrlink();
   moveitem(newitem,newlink->item);
   list->listlength++;

/* Reset pointers.
   ------------------- */
   newlink->previous = list->tail;
   newlink->next = NULL;
   list->tail->next = newlink;
   list->clp = list->tail = newlink;
}


/*-------------------------------------------------------------*
¦  Function    :  lldelete                                     ¦
¦  Usage       :  void lldelete();                             ¦
¦  Description :  Delete and free the CLP, reset CLP to haed.  ¦
¦  Return      :  None                                         ¦
*-------------------------------------------------------------*/
void lldelete()
{
   struct LINKTYPE *before, *after;

/* Is this the only link?
   ----------------------- */
   if (list->head == list->clp && list->tail == list->clp) {
      list->head = list->tail = NULL;
   }

/* Is this the head?
   ------------------- */
   else if (list->head == list->clp) {
      list->head = list->head->next;
      list->head->previous = NULL;
   }

/* Is this the tail?
   --------------------- */
   else if (list->tail == list->clp) {
```

```
      list->tail = list->tail->previous;
      list->tail->next = NULL;
   }

/* Otherwise, it must be inside the list.
   ---------------------------------------- */
   else {
      before = list->clp->previous;
      after = list->clp->next;
      before->next = after;
      after->previous = before;
   }

/* Delete CLP.
   ------------- */
   free(list->clp);
   list->clp = list->head;
   list->listlength--;
}

/*-------------------------------------------------------------*
|  Function     :  ll_length                                  |
|  Usage        :  int ll_length();                           |
|  Description  :  Return the length of the list              |
|  Return       :  Length of the list                         |
*-------------------------------------------------------------*/
int ll_length()
{
   return (list->listlength);
}

/*-------------------------------------------------------------*
|  Function     :  llindex                                    |
|  Usage        :  int llindex(int index);                    |
|  Description  :  Move clp to specified node                 |
|  Return       :  on success  return  number of node         |
|                  on failure  return  last node number       |
*-------------------------------------------------------------*/
int llindex(int index)
{
   int temp = 1;

   llhead();

   if (list->clp == NULL)
      return 0;

   while( --index != 0 ) {
      if (llnext() != 0) {
         temp++;
      }
      else  break;
   }
   return(temp);
}
```

```
/*-------------------------------------------------------------*
|   Function    : get_wk_list();                              |
|   Usage       : struct LINKLIST *get_wk_list();             |
|   Description : get current working list                    |
|   Return      : current working list                        |
*-------------------------------------------------------------*/
struct LINKLIST *get_wk_list()
{
   return list;
}


/*-------------------------------------------------------------*
|   Function    : llupdate();                                 |
|   Usage       : void llupdate(char *item);                  |
|   Description : update link list value                      |
|   Return      : None                                        |
*-------------------------------------------------------------*/
int llupdate(char *item)
{
   if (ll_length() == 0)
      return 0;
   else {
      lladd(item);
      llprevious();
      lldelete();
   }
}


static int lineno = 1;
static char spaces[] =
"                                                                ";

/*-------------------------------------------------------------*
|   Function    :  select_list();                             |
|   Usage       :  int select_list(int lf,int tp,             |
|                       struct LINKLIST *list,int high);      |
|   Description :  display list of list                       |
|   Return      :  selected list bar                          |
*-------------------------------------------------------------*/
int select_list(int lf,int tp,struct LINKLIST *list,int high)
{
   int height,wd;
   int c;

   lineno = 1;
   height = (high >= 10) ? 10 : high;

   establish_window(lf,tp,lf+14,tp+height+1,WHITE,BLUE,1);
   wkw.wtop = 1;
   wkw.wx = wkw.wy = 1;
```

```
  while( TRUE ) {
     dlist(list);
     highlight(lineno);
     hidecursor();
     c = getkey();
     if ( c == '\r' || c == ESC)
        break;
     switch (c) {
        case UP :
                  lupline();
                  break;
        case DN :
                  ldownline(list);
                  break;
        default : putch(BELL);
     }
  }
  delete_window();
  return c == ESC ? 0 : lineno;
}

/*-------------------------------------------------------------*
|   Function    :  dlist();                                   |
|   Usage       :  void dlist(struct LINKLIST *list);         |
|   Description :  display list in window                     |
|   Return      :  none                                       |
*-------------------------------------------------------------*/
static void dlist(struct LINKLIST *list)
{
   int height;
   int i = 1;

   height = wkw.ht;

   llindex(wkw.wtop);
   while((list->clp != NULL) && (--height != 1)) {
      win_dstr(2,++i,list->clp->item,7);
      win_dstr(2+strlen(list->clp->item),i,spaces+79-wkw.wd+
               strlen(list->clp->item)+2,7);
      llnext();
   }
}

/*-------------------------------------------------------------*
|   Function    : highlight();                                |
|   Usage       : void highlight(int row);                    |
|   Description : display line of text of reverse attribute   |
|   Return      : None                                        |
*-------------------------------------------------------------*/
static void highlight(int row)
{
   if (llindex(row) == row) {
      wkw.wy = (row-wkw.wtop+1 > wkw.ht-2) ? 10 : row-wkw.wtop+1;
      win_reverse(2,wkw.wy+1,wkw.wd-2,0x70);
   }
```

```
}

/*-------------------------------------------------------------*
|   Function    : lupline();                                  |
|   Usage       : void lupline();                             |
|   Description : move up one link                            |
|   Return      : None                                        |
*-------------------------------------------------------------*/
static void lupline()
{
   if (lineno>1) {
      lineno--;
      if (wkw.wtop == lineno+1)
         wkw.wtop--;
   }
}

/*-------------------------------------------------------------*
|   Function    : ldownline();                                |
|   Usage       : void ldownline(struct LINKLIST *list);      |
|   Description : move down one link                          |
|   Return      : None                                        |
*-------------------------------------------------------------*/
static void ldownline(struct LINKLIST *list)
{
   if (list->clp->next != NULL) {
      if ((lineno < wkw.ht-2) || (lineno-wkw.wtop+1<wkw.ht-2))
         lineno++;
      else {
         wkw.wtop++;
         lineno++;
      }
   }
}
```

```
/*------------------- telphone.c ------------------------*/

#include<stdio.h>
#include<stddef.h>
#include<dos.h>
#include"..\header\telphone.h"


int ring_no_receiver = 10;              /* times of ring back for no receiver */
int retry_enquiry    = 1;               /* time of retry enquiry */
long Maxcount = 1000;                   /* count for tel out */

/*-------------------------------------------------------------*
|  Function     :  detect                                      |
|  Usage        :  int detect();                               |
|  Description  :  Wait for ringing                            |
|  Return       :  0 for ringing in                            |
|                  ascii code  for keyboard hit                |
|                  -1 for time out and call out to user        |
*-------------------------------------------------------------*/
int detect()
{
   int c,counter;

   counter = 0;
   while(!kbhit()) {
      if(((inp(CONTROL_ADDR)) & DETECT) == DETECT)
         return 0;
      else {
         delay(20);
         if(counter++ >= Maxcount)
            return  -1;
      }
   }
   if ((c = getch()) == 0 )
      c = getch() | 128;

   return c;
}


/*-------------------------------------------------------------*
|  Function     : get_id                                       |
|  Usage        : int get_id(unsigned char input[],int length); |
|  Description  : Call getdtmf for one character and concat    |
|                 to construct user id of length "length"      |
|  Return       : -1 for id error ( hook off )                 |
|                 1 and id otherwise                           |
*-------------------------------------------------------------*/

int get_id(unsigned char input[],int length)
{
   int i;
   char temp;
```

```
    for(i = 0;i<length-1; i++)  {
       if ((temp = getdtmf()) != -1) {
          input[i] = temp;
       }
       else
          return -1;
    }
    input[i] = NULL;
    return 1;
}


/*-------------------------------------------------------------*
|  Function     :  getdtmf                                    |
|  Usage        :  unsigned char getdtmf();                   |
|  Description  :  get one character of DTMF code of one press|
|  Return       :  Character of DTMF code                     |
|                  -1  if error                               |
*-------------------------------------------------------------*/
char getdtmf()
{
    unsigned char inword;
    int i;
    int temp;

    i = 0;
    while( (inp(CONTROL_ADDR) & DTMF ) != DTMF)  {
       i++;
       delay(20);
       if ( i == 1000 ) {                  /* if = 1000; check status */
          if ((temp = phone_status_in()) == 4 )
             i = 0;                        /* if 0, line has not been hook off */
                                           /* wait on for dtmf hit */
          else
             if( temp != 4 )
                return -1;
          }
    }

    inword = inp(CONTROL_ADDR);
    inword = ((inword & 0x3e) >> 2) + '0';

    switch (inword) {
       case ':' : inword = '0';
                  break;
       case ';' : inword = '*';
                  break;
       case '<' : inword = '#';
                  break;
    }

    while(((inp(CONTROL_ADDR)) & DTMF) == DTMF) {}
    return(inword);
}
```

```
/*-------------------------------------------------------------*
|  Function     :   Dtmf_hit                                  |
|  Usage        :   int dtmf_hit();                           |
|  Description  :   determine if there is a dtmf-hit          |
|  Return       :   0  :  there is not dtmf-hit               |
|                   1  :  there is dtmf-hit                   |
*-------------------------------------------------------------*/
int dtmf_hit()
{
   if( ( inp(CONTROL_ADDR) & DTMF ) == DTMF) {
      while( ( inp(CONTROL_ADDR) & DTMF ) == DTMF) {}
      return 1;
   }
   else
      return 0;
}



/*-------------------------------------------------------------*
|   Function    :   Get_letter                                |
|   Usage       :   unsigned char get_letter();               |
|   Description :   get one letter from telphone key          |
|   Return      :   letter on telphone key if correct         |
|                   0  otherwise                              |
|                   -1 for hook off                           |
*-------------------------------------------------------------*/
char get_letter()
{
   char templ,temp;
   int i;

   templ = 'A';
   for(i=0;i<4;i++) {
      temp = getdtmf();

      switch (temp) {
         case '1' : if(i==0)
                       return 0;
                    else
                       return templ;
         case '8' :
         case '9' : templ = 'A' + (temp-'0'-2)*3 +1 + i;
                    break;
         case '7' : templ = 'A' + (temp-'0'-2)*3 + i;
                    if ( i >= 1 )
                       templ++;
                    break;
         case '*' :
         case '0' :
         case '#' : return 0;
```

```
            case -1  : return -1;

            default  : temp1 = 'A'+ (temp-'0')*3 -6 + i;
         }
      }
      return temp1;
}


/*-------------------------------------------------------------*
|   Function    :   Hook_on                                    |
|   Usage       :   void hook_on();                            |
|   Description :   Hook the telphone on                       |
|   Return      :   None                                       |
*-------------------------------------------------------------*/
void hook_on()
{
   inp(HOOK_ON);
}


/*-------------------------------------------------------------*
|  Function     :  hookoff                                     |
|  Usage        :  void hookoff();                             |
|  Description  :  Hook off the telephone line                 |
|  Return       :  None                                        |
*-------------------------------------------------------------*/
void hook_off()
{
   inp(HOOK_OFF);
}


/*-------------------------------------------------------------*
|   Fundtion    :   Flash                                      |
|   Usage       :   void flash(int time);                      |
|   Description :   Flash delay time msec                      |
|   Return      :   None                                       |
*-------------------------------------------------------------*/
void flash(int time)
{
   hook_off();
   delay(time);
   hook_on();
}

/*-------------------------------------------------------------*
|   Function    :   Phone_status_in                            |
|   Usage       :   int phone_status_in();                     |
|   Description :   check for status of telelphone line        |
|   Return      :   1  for ring back tone                      |
|                   2  for line busy                           |
|                   3  for line dial tone                      |
|                   0  for receiver hook on                    |
*-------------------------------------------------------------*/
```

```
int phone_status_in()
{
   unsigned int a;
   int i,no,c;
   int temp;

/*   temp = 4;*/
   for(i=0;i<3;i++) {
      no = 0;
      c = 0;
      a = inp(CONTROL_ADDR);

      while((a & PHONE_STATUS_IN) == PHONE_STATUS_IN) {
         a = inp(CONTROL_ADDR);
         delay(15);
         c++;
         if(c > 60) {            /* dial tone */
            temp = 3;
            break;
         }
      }
      while((a & PHONE_STATUS_IN) != PHONE_STATUS_IN) {
         if (no > 250) {         /* detect long zero ; receiver hook on */
            break;
         }
         no++ ;
         a = inp(CONTROL_ADDR);
         delay(15);
      }
      if(no > 250 ) {
         temp = 0;
         break;
      }
      else
         if ( no > 150) {
            temp = 1;
            break;
         }
         else
            if(c > 60) {
               temp = 3;
               break;
            }
/*          else
               if(no < 10) {
                 temp = temp;
                 i--;
               }*/
               else
                 temp = 2;
   }
   return temp;
}

/*-------------------------------------------------------------*
```

```
¦   Function    :   Phone_status_out                                    ¦
¦   Usage       :   int phone_status_out();                             ¦
¦   Description :   check for status of telelphone line                 ¦
¦   Return      :   1  for ring back tone                               ¦
¦                   2  for line busy                                    ¦
¦                   3  for line ready                                   ¦
¦                   0  for receiver hook on                             ¦
*-------------------------------------------------------------------*/
int phone_status_out()
{
   unsigned int a;
   int i,no,c;
   int temp;

   temp = 4;

   for(i=0;i<3;i++) {
      delay(200);
      no = 0;
      c = 0;

      a = inp(CONTROL_ADDR);

      while((a & PHONE_STATUS_OUT) == PHONE_STATUS_OUT) {
         a = inp(CONTROL_ADDR);
         delay(15);
         c++;
         if(c > 60) {
            temp = 3;
            break;
         }
      }
      while((a & PHONE_STATUS_OUT) != PHONE_STATUS_OUT) {
         if (no > 300) {
            break;
         }
         no++ ;
         a = inp(CONTROL_ADDR);
         delay(15);
      }
      if(no > 300 ) {
         temp = 0;
         break;
      }
      else
         if ( no > 160 ) {
            temp = 1;
            break;
         }
         else
            if(c > 60) {
               temp = 3;
               break;
            }
            else
```

```
            if(no < 10){
               temp = temp;
               i--;
            }
            else
            temp = 2;
      }
/*    printf("%2d",temp);*/
      return temp;
}


/*-------------------------------------------------------------*
|   Function    :   Tel_call                                  |
|   Usage       :   void tel_call(char *tel_no_in);           |
|   Description :   simulate telephone-call                   |
|   Return      :   0  if success                             |
*-------------------------------------------------------------*/
void tel_call(char *tel_no_in)
{
   char temp;
   int inttemp;

   while((temp = *(tel_no_in++)) != NULL) {
       inttemp = conv_to_dtmf(temp);
       outp(DTMF_OUT,inttemp);
       delay(300);
       outp(DTMF_OUT,NO_DTMF);
       delay(200);
   }
}


/*-------------------------------------------------------------*
|   Function    :   Conv_to_dtmf                              |
|   Usage       :   int conv_to_dtmf(char no_in);             |
|   Description :   Convert character input to dtmf signal    |
|   Return      :   Dtmf signal if success                    |
|                   -1 otherwise                              |
*-------------------------------------------------------------*/
int conv_to_dtmf(char no_in)
{
   switch (no_in) {
      case '1' :  return 0XE1;
      case '2' :  return 0XE2;
      case '3' :  return 0XE4;
      case '4' :  return 0XD1;
      case '5' :  return 0XD2;
      case '6' :  return 0XD4;
      case '7' :  return 0XB1;
      case '8' :  return 0XB2;
      case '9' :  return 0XB4;
      case '*' :  return 0X71;
      case '0' :  return 0X72;
      case '#' :  return 0X74;
```

```
    }
    return 1;
}

int is_telcard_exist()
{
    int i;

    for (i=0;i<100;i++) {
        if(((inp(CONTROL_ADDR)) & DETECT) == DETECT)
            return 0;
    }
    return 1;
}

/*-------------------------------------------------------------*
|   Function    :   Enquiry                                    |
|   Usage       :   int enquiry(char *tel_no);                 |
|   Description :   direct invert dialing                      |
|   Return      :   1 for busy                                 |
|                   2 for  receiver hook on                    |
|                   3 for  no receiver hook on                 |
*-------------------------------------------------------------*/

int enquiry(char *tel_no)
{
    int  status,j,i,no;

    j = 0;
    status = 0;
    no = 0;

    while(status == 0){
        flash(500);
        delay(700);
        while(((phone_status_in()) != 3) && (no < 3)) {
            flash(500);
            delay(700);
            flash(500);
            no++;
        }
        if(no >= 3)
            return(BUSY);

        tel_call(tel_no);
        if((phone_status_in()) == 1) {     /* ring back tone */
            i = 0;
            while((phone_status_in()) == 1){
                if(i++ > ring_no_receiver){
                    return RECEIVER_NOT_HOOK_ON;            /* no receiver hook on */
                }
            }
            return RECEIVER_HOOK_ON;              /* receiver hook on */
        }
        else if(j++ < retry_enquiry ) {
```

```
        if((phone_status_in()) != 2)
           j--;
        status = 0;                          /*  retry enquiry */
        flash(500);
        delay(500);
      }
      else {
        status = BUSY;                                /* busy and exit */
      }
   }
   return(status);
}


/*-------------------------------------------------------------*
|   Function    :   telphone out                               |
|   Usage       :   int tel_out(char *tel_no);                 |
|   Description :   telphone out                               |
|   Return      :   0 for  no ringing                          |
|                   2 for  receiver hook on                    |
|                   1 for  busy                                |
|                   3 for  no receiver hook on                 |
*-------------------------------------------------------------*/
int tel_out(char *tel_no)
{
   int status,time,count,no;

   status = 1;
   count = 0;
   no = 0;

   while(status == 1) {
      flash(500);
      delay(100);
      while((phone_status_out()) != 3){
         if(no >= 3 ){
            return 0 ;                       /* can not flash */
         }
         flash(200);
         delay(10);
         no++;
      }
      tel_call("9");                         /* extend '9' */
      delay(50);
      no = 0;
      if((phone_status_out()) == 2){
         tel_call("6");                      /*  booking line */
         hook_off();
         detect();
      }
      tel_call(tel_no);
      if((phone_status_out()) == 1){                 /* ring back tone */
            time = 0;
         while((phone_status_out()) == 1){
            if(time++ > ring_no_receiver)
```

```
                return 3;                    /* no receiver hook_on */
            }
            return 2;                        /* receiver hook on */
        }
        else if(count++ < retry_enquiry) {
            status = 1;
            hook_off();     /* repeat entirely */
            delay(2000);
        }
        else {
            return 1;                        /* busy and exit */
        }
    }
}
```

```
/*---------------------- vcmodule.c ----------------------*/

#include<dos.h>
#include<fcntl.h>
#include<alloc.h>
#include<io.h>
#include<conio.h>
#include"..\header\vc.h"
#include"..\header\telphone.h"

#define   BASE_ADDR     0x2b8
#define   DTMF          0x02
#define   CONTROL_ADDR  0X3B8
#define   MAX_REC_TIME  15              /* max time for one record */

/*out port*/
static   PLAY_DATA     =  BASE_ADDR + 0;
static   SOUND_OFF     =  BASE_ADDR + 1;
static   SOUND_ON      =  BASE_ADDR + 2;
static   RECORD_MODE   =  BASE_ADDR + 3;

/*in port*/
static   IRQ_DISABLE   =  BASE_ADDR + 0;
static   IRQ_ENABLE    =  BASE_ADDR + 1;
static   RESET_IRQ     =  BASE_ADDR + 2;
static   RECORD_DATA   =  BASE_ADDR + 2;

#undef   BASE_ADDR

#define IRQ     10
#define IMR     ~0x04
#define OCW1    0x0021
#define OCW2    0x0020
#define EOI     0x0020


#undef   FP_SEG
#undef   FP_OFF
#define  FP_SEG(fp) (*((unsigned *)&(fp)+1))
#define  FP_OFF(fp) (*((unsigned *)&(fp)))

static   void interrupt far vo_int (void);
static   void (interrupt far *old_vect)();
static int stop_play();
static int stop_record();

unsigned char far *vo_p,*c_p;
unsigned int rate_flag = -1;
unsigned long p_len,r_len;
unsigned int mode_flag = -1;


/*-------------------------------------------------------------*
|  Function     :   Initvoice                                  |
|  Usage        :   int initvoice()                            |
```

```
¦  Description  :   initial voice card for play and record      ¦
¦  Return       :   None                                        ¦
*-------------------------------------------------------------*/
int initvoice()
{

   if((vo_p = (unsigned char far *)farmalloc(0x10000)) == 0)
      return -1;

   c_p = vo_p;
   inp(IRQ_DISABLE);
   outp(PLAY_DATA,0x55);
   outp(SOUND_OFF,0);
   old_vect= getvect(IRQ);
   disable();
   setvect(IRQ,vo_int);
   enable();
   inp(IRQ_ENABLE);
   inp(RESET_IRQ);
   outp(OCW1,(inp(OCW1) & IMR));
   return 0;
}


/*-------------------------------------------------------------*
¦  Function     :   Play_data                                   ¦
¦  Usage        :   int play_data(char *fname,long start,       ¦
¦                                 long stop)                   ¦
¦  Description  :   play file specified by fname from start     ¦
¦                   to stop                                     ¦
¦                   if start = -1 then start from begin of      ¦
¦                     file to end of file                       ¦
¦  Return       :   -1  :  Open file error                      ¦
¦                    0  :  Play complete                        ¦
*-------------------------------------------------------------*/
int play_data(char *fname,long start,long stop)
{
   unsigned int c_s;
   unsigned char far *b_p;
   unsigned long t_len,t_a;
   int handle;
   unsigned long temp;

   if ((handle = _open(fname,O_BINARY¦O_RDWR)) == -1)
      return -1;

   if (start == -1) {
      t_len = p_len = filelength(handle);
      start = 0;
      }
   else
      t_len = p_len = stop - start;

   FP_SEG (b_p) = FP_SEG(vo_p);
   FP_OFF(vo_p) = FP_OFF(b_p) = c_s =  0;
```

```
    lseek(handle,start,SEEK_SET);
    if (t_len > 0x8000)  {
       read(handle,b_p,0x8000);
       t_len -= 0x8000;
       b_p += 0x8000;
       }
    t_a = (t_len > (unsigned long) 0x8000) ? 0x8000 : t_len;
    read(handle,b_p,t_a);
    t_len -= t_a;

    outp(PLAY_DATA,0x55);
    outp(SOUND_ON,0);
    mode_flag = 0;
    rate_flag = 0;
    while(p_len  && !stop_play())   {
       if(t_len && (c_s != (FP_OFF(vo_p) & 0x8000)))     {
          t_a = (t_len > 0x8000) ? 0x8000 : t_len;
          FP_OFF(b_p) = c_s;
          read(handle,b_p,t_a);
          t_len -= t_a;
          c_s ^= 0x8000;
          }
       }
    rate_flag = -1;
    outp(PLAY_DATA,0x55);
    close(handle);
    return 0;
}


/*-------------------------------------------------------------*
|  Function     :   Record_data                               |
|  Usage        :   int record_data(char *fname)              |
|  Description  :   record voice input to file specified by   |
|                   fname                                     |
|  Return       :   0  :  record complete                     |
|                  -1  :  create file error                   |
|                  -2  :  disk full                           |
*-------------------------------------------------------------*/
int  record_data(char *fname)
{
   unsigned long len;
   unsigned char far*b_p;
   unsigned int bytes;
   int handle;

   if((handle = _creat(fname,FA_ARCH)) ==  -1)
      return -1;

   FP_SEG(b_p) = FP_SEG(vo_p);
   len = 0L;
   FP_OFF(b_p) = FP_OFF(vo_p) = r_len = 0;
   outp (PLAY_DATA,0xFF);
   outp(SOUND_ON,0);
   outp (RECORD_MODE,0);
```

```
   mode_flag = -1;
   rate_flag = 0;

   while ( !stop_record()) {
      if(r_len > 0x8000)  {
         FP_OFF(b_p) = (FP_OFF(vo_p) & 0x8000) ^0x8000;
         bytes = write(handle,b_p,0x8000);
         if(bytes < 0x8000)
            return -2;

         r_len -= 0x8000;
         len   += 0x8000;
      }
   }
   rate_flag = -1;
   outp(PLAY_DATA,0x55);
   outp(SOUND_OFF,0);
   if(r_len > 0x8000)  {
      FP_OFF(b_p) = (FP_OFF(vo_p) & 0x8000) ^0x8000;
      bytes = write(handle,b_p,0x8000);
      if(bytes < 0x8000)
         return -2;

      r_len -= 0x8000;
   }
   FP_OFF(b_p) = FP_OFF(vo_p) & 0x8000 ;
   bytes = write(handle,b_p,r_len);
   if(bytes < r_len)
      return -2;

   outp(PLAY_DATA,0x55);
   _close(handle);

   return 1;
}

/*-------------------------------------------------------------*
 |   Function    : vo_int();                                   |
 |   Usage       : static void interrupt far vo_int(void);     |
 |   Description : Interrupt service routine for playing and   |
 |                 recording voice                             |
 |   Return      : None                                        |
 *-------------------------------------------------------------*/
static void interrupt far vo_int(void)
{
   outp(OCW2,EOI);
   outp(SOUND_ON,0);
   if (! rate_flag)   {
      if (! mode_flag)   {
         if(p_len)  {
            p_len--;
            outp(PLAY_DATA,*vo_p++);
            inp(RESET_IRQ);
            }
```

```
         else   {
            FP_OFF(vo_p) = 0;
            outp(PLAY_DATA,0x55);
            inp(RESET_IRQ);
            rate_flag = -1;
            }
         }
      else  {
         outp(SOUND_OFF,0x00);
         *vo_p ++ = inp(RECORD_DATA);
         r_len++;
         }
      }
   else
      inp(RESET_IRQ);
}


/*-------------------------------------------------------------*
|   Function    : end_voice();                                |
|   Usage       : void end_voice();                           |
|   Description : set vector to old_vect and free buffer      |
|   Return      : None                                        |
*-------------------------------------------------------------*/
void end_voice()
{
   disable();
   setvect(IRQ,old_vect);
   enable();
   farfree(c_p);
}


/*-------------------------------------------------------------*
|   Function    : stop_play();                                |
|   Usage       : int stop_play();                            |
|   Description : determine whether dtmf is hit               |
|   Return      : 1 for dtmf hit                              |
|                 0 otherwise                                 |
*-------------------------------------------------------------*/
static int stop_play()
{
   if(dtmf_hit())
      return 1;
   return 0;
}


/*-------------------------------------------------------------*
|   Function    : stop_record();                              |
|   Usage       : int stop_record();                          |
|   Description : Determine whether keyboard or dtmf is hit   |
|   Return      : 1 for stop_recording                        |
|                 0 otherwise                                 |
*-------------------------------------------------------------*/
```

```
static int stop_record()
{
   if(dtmf_hit())
      return 1;
   if(kbhit()) {
      getch();
      return 1;
   }
   return 0;
}
```

```
/*----------------------- kram.c ------------------------*/

#include<stdio.h>
#include<mem.h>
#include<stdlib.h>
#include<io.h>
#include"..\header\kram.h"

static   KramParamPtr  ParamPtr;
char *TempKeyValue;

static int find_in_slot(void *KeyValue,int *DataOffset,int RecordLength);


/*-----------------------------------------------------------------*
¦   Function    :   HashCode(char *Key);                          ¦
¦   Usage       :   long HashCode(char *Key);                     ¦
¦   Return      :   32 bit hashcode                               ¦
¦   Description :   Find hashcode of key inputed                  ¦
*-----------------------------------------------------------------*/
static long  HashCode(char *Key)
{
   int  i;
   long result;
   long temp1;
   long temp2;
   unsigned char bytetemp;

   result = 0;

   for(i=0;i<ParamPtr->para_field.KeyLength;i++) {
      temp1 = result << 7;
      temp2 = result >> 25;
      result = temp1 ¦ temp2;
      bytetemp = Key[i];
      result = result ^ bytetemp;
   }
   return  result;
}

/*-----------------------------------------------------------------*
¦  Use this procedure to position the file pointer to a            ¦
¦  particular data block. In order to do this, we must skip        ¦
¦  the parameter block and the index block. Also note that the     ¦
¦  first data block is #1.                                         ¦
*-----------------------------------------------------------------*/
static void SeekBlock(FILE *KramFile,int BlockNum)
{
   long BlockPosition;

   rewind(KramFile);
   BlockPosition = PARAMSIZE + INDEXSIZE + (DATASIZE * (BlockNum-1));
   fseek(KramFile,BlockPosition,SEEK_SET);
}
```

```
/*-------------------------------------------------------------*
|  Use this procedure to initialize a new KRAM file. It sets   |
|  up the key length and the data length according to the input |
|  argements. The high data block number is set to 1 and data  |
|  block #1 is initialized to zeroes (an empty block). The     |
|  index block is set to all 1's, so that all accesses will    |
|  go to the empty data block.                                 |
*-------------------------------------------------------------*/
int KramInit(char *FileName, int KeyLength, int DataLength)
{
   FILE *KramFile;
   KramIndexType Index;
   KramDataType  Data;
   KramParamType Params;
   int i;

   KramFile = fopen(FileName,"wb");

   setbuf(KramFile,NULL);

   memset(Params.Dummy,0,sizeof(Params.Dummy));

   Params.para_field.KeyLength = KeyLength;
   Params.para_field.DataLength = DataLength;

/*  the highest data block number in use is #1 */
   Params.para_field.HighBlock = 1;
   fwrite(&Params,sizeof(Params),1,KramFile);


/* Initialize the index block to all 1's, as only data block #1 exists */
   for(i=0;i<INDEXCOUNT;i++)
      Index[i] = 1;

   fwrite(Index,sizeof(Index),1,KramFile);

/* Initialize the first data block to all zeroes */
   for (i=0;i<DATASIZE;i++)
      Data[i] = 0;

   fwrite(Data,sizeof(Data),1,KramFile);
   fclose(KramFile);

   return (1);
}


/*-------------------------------------------------------------*
|   Function        KramOpen();                                |
|   Usage       :   FILE *KramOpen(char *KramFilename)         |
|   Description :   Open Kram file of name KramFilename        |
|   Return      :   Pointer to Kram file on success            |
|                   Null otherwise                             |
*-------------------------------------------------------------*/
```

```
FILE *KramOpen(char *KramFileName)
{
   int RecsRead;
   FILE *KramFile;

/* initial parameter block */

   ParamPtr = (KramParamPtr ) malloc(sizeof(KramParamType));
   setmem(ParamPtr,sizeof(KramParamType),0);

   KramFile = fopen(KramFileName,"r+b");

   RecsRead = fread(ParamPtr,sizeof(KramParamType),1,KramFile);

   if (RecsRead != 1) {          /* Not a Kram file */
      free(ParamPtr);
      return 0;
   }

   ParamPtr->para_field.IndexPtr = (KramIndexPtr ) malloc(sizeof(KramIndexType));

   if (ParamPtr->para_field.IndexPtr == NULL) {
      free(ParamPtr);
      return 0;                  /* Not enough memory */
   }

   ParamPtr->para_field.DataPtr = (KramDataPtr ) malloc(sizeof(KramDataType));

   if (ParamPtr->para_field.DataPtr == NULL) {
      free(ParamPtr);            /* Not enough memory */
      free(ParamPtr->para_field.DataPtr);
      return 0;
   }

   ParamPtr->para_field.CurrentBlock = 0;

   RecsRead = fread(ParamPtr->para_field.IndexPtr,sizeof(KramIndexType),1,KramFile);

   if ( RecsRead != 1 ) {
      free(ParamPtr);
      free(ParamPtr->para_field.IndexPtr);
      free(ParamPtr->para_field.DataPtr);
      return 0;
   }

/* Initialize TempKeyValue */
   TempKeyValue = (char *) malloc(ParamPtr->para_field.KeyLength);
   if (TempKeyValue == NULL ) {          /* not enough memory */
      free(TempKeyValue);
      return 0;
   }
   return( KramFile );
}
```

```
/*-------------------------------------------------------------*
¦  Use this procedure to close the file after use. It also     ¦
¦  deallocates the dynamics storage used for the parameter,    ¦
¦  index, and data blocks.                                     ¦
¦                                                            ¦
¦  IMPORTANT NOTE: If you have modified the file, by adding    ¦
¦  records, for example, you MUST close the file to ensure     ¦
¦  that all the records have been written to the file.         ¦
*-------------------------------------------------------------*/
void KramClose(FILE *KramFile)
{

   if (ParamPtr->para_field.BlockModified) {
      SeekBlock(KramFile,ParamPtr->para_field.CurrentBlock);
      fwrite(*ParamPtr->para_field.DataPtr,DATASIZE,1,KramFile);
   }

   free(ParamPtr->para_field.DataPtr);
   free(ParamPtr->para_field.IndexPtr);
   free(ParamPtr);
   free(TempKeyValue);
   fclose(KramFile);
}


/*-------------------------------------------------------------*
¦  Use this procedure to add records to a KRAM file that      ¦
¦  has been opened by KramOpen. You must supply the file      ¦
¦  pointer that was returned by KramOpen in "KramFile",       ¦
¦  the key in "KeyValue", and the data in "DataValue". The    ¦
¦  algorithm is known as "externsible hashing".               ¦
*-------------------------------------------------------------*/
int KramAdd(FILE *KramFile,void *KeyValue,void *DataValue)
{

   KramDataPtr  LowDataPtr;
   KramDataPtr  HighDataPtr;

   long HashVal;

   int BlockNumber;
   int TempBlockNumber;
   int RecordLength;
   int i,j,k;
   int SlotFound;
   int FoundFirst;
   int FoundLast;
   int FirstBlock;
   int LastBlock;
   int MidBlock;
   int OldOffset;
   int LowOffset;
   int HighOffset;
   int Duplicate;
   int KeepLooking;
```

```
RecordLength = ParamPtr->para_field.KeyLength + ParamPtr->para_field.DataLength;

do {
   HashVal = HashCode(KeyValue) & (INDEXCOUNT - 1);

   BlockNumber = *(*ParamPtr->para_field.IndexPtr+HashVal+1);

   if ( ParamPtr->para_field.CurrentBlock != BlockNumber ) {
      if ((ParamPtr->para_field.BlockModified)
      && (ParamPtr->para_field.CurrentBlock != 0)) {
         ParamPtr->para_field.BlockModified = FALSE;
         SeekBlock(KramFile,ParamPtr->para_field.CurrentBlock);
         fwrite(ParamPtr->para_field.DataPtr,DATASIZE,1,KramFile);
      }
      SeekBlock(KramFile,BlockNumber);
      fread(ParamPtr->para_field.DataPtr,DATASIZE,1,KramFile);
      ParamPtr->para_field.CurrentBlock = BlockNumber;
   }

   Duplicate = FALSE;
   KeepLooking = TRUE;
   SlotFound = FALSE;
   i = 1;
   OldOffset = 0;

   /* initialize length of temporary string used for comparison */

   while (KeepLooking && (i <= (DATASIZE/RecordLength))) {
      if ( *(*ParamPtr->para_field.DataPtr+OldOffset) == 0 ) {
         KeepLooking = FALSE;
         SlotFound = TRUE;
      }
      else {
         memmove(TempKeyValue,*ParamPtr->para_field.DataPtr+OldOffset,
                  ParamPtr->para_field.KeyLength);
         if (!strcmp(TempKeyValue,KeyValue)) {
            Duplicate = TRUE;
            KeepLooking = FALSE;
         }
         else {
            OldOffset += RecordLength;
            i += 1;
         }
      }
   }

   if ( SlotFound ) {
      memmove(*ParamPtr->para_field.DataPtr+OldOffset,KeyValue,
              ParamPtr->para_field.KeyLength);

      memmove(*ParamPtr->para_field.DataPtr+OldOffset+ParamPtr->
            para_field.KeyLength,DataValue,ParamPtr->para_field.DataLength);
      ParamPtr->para_field.BlockModified = TRUE;
   }
   else if ( Duplicate == FALSE ) {
```

```
/* First we must determine how many index entries are affected */
/* by the split */

            find_id_block(&FirstBlock,&LastBlock,BlockNumber);

            if (FirstBlock >= LastBlock) {
/*  we are out of room */
               return(-2);              /* Kram File is full */
            }
/*  Now we have to allocate another block for the split. */
            ParamPtr->para_field.HighBlock += 1;
            MidBlock = (FirstBlock + LastBlock) / 2;

/* we will assign the items that have the higher hash code */
/* to the new block */
            for (i=MidBlock + 1;i<=LastBlock;i++)
               *(*ParamPtr->para_field.IndexPtr+i) = ParamPtr->para_field.HighBlock;

/* Now we have to go through all the items in the block to be split */
/* and assign them to the old block or the new one according to their */
/* new hash codes, 1 bit longer than the previous ones. An extra */
/* temporary block makes this easier. */
            LowDataPtr = (KramDataPtr ) malloc (sizeof(KramDataType));
            HighDataPtr = (KramDataPtr ) malloc (sizeof(KramDataType));

            memset(LowDataPtr,0,sizeof(KramDataType));
            memset(HighDataPtr,0,sizeof(KramDataType));

            OldOffset = 0;
            LowOffset = 0;
            HighOffset = 0;
            for ( i = 0;i< DATASIZE / RecordLength;i++) {
               memmove(TempKeyValue,*ParamPtr->para_field.DataPtr+OldOffset,
                  ParamPtr->para_field.KeyLength);

               HashVal = HashCode(TempKeyValue) & (INDEXCOUNT - 1);
               TempBlockNumber = *(*ParamPtr->para_field.IndexPtr+HashVal+1);
               if (TempBlockNumber == BlockNumber) {
 /* send to the lower one */
                  memmove((*LowDataPtr)+LowOffset,*ParamPtr->para_field.
                        DataPtr+OldOffset,RecordLength);

                  LowOffset += RecordLength;
               }
               else {
                  memmove((*HighDataPtr)+HighOffset,*ParamPtr->para_field.DataPtr+
                        OldOffset,RecordLength);

                  HighOffset += RecordLength;
               }
               OldOffset += RecordLength;
            }
            SeekBlock(KramFile,BlockNumber);
            fwrite(*LowDataPtr,DATASIZE,1,KramFile);
```

```
            SeekBlock(KramFile,ParamPtr->para_field.HighBlock);
            fwrite(*HighDataPtr,DATASIZE,1,KramFile);

            free(LowDataPtr);
            free(HighDataPtr);

/* Make sure the same block isn't used the next time we have to */
/* expand the file. Also note that the data block is out of date. */
            ParamPtr->para_field.CurrentBlock = 0;
            rewind(KramFile);
            fwrite(ParamPtr,sizeof(KramParamType),1,KramFile);
            fwrite(*ParamPtr->para_field.IndexPtr,
            sizeof(KramIndexType),1,KramFile);
         }
   } while (!( SlotFound || Duplicate ));

   if (Duplicate)
      return FALSE;
   else
      return TRUE;
}

/*-------------------------------------------------------------*
|   Function    : KramRead();                                 |
|   Usage       : void *KramRead(FILE *KramFile,void *KeyValue, |
|                                void *DataValue);            |
|   Description : retrieve record specified by KeyValue       |
|   Return      : Pointer to data record                      |
|                 Null otherwise                              |
*-------------------------------------------------------------*/
void *KramRead(FILE *KramFile,void *KeyValue,void *DataValue)
{
   int DataOffset;

   if ( find_slot(KramFile,KeyValue,&DataOffset) == TRUE ) {
      memmove(DataValue,*ParamPtr->para_field.DataPtr+DataOffset+
      ParamPtr->para_field.KeyLength,ParamPtr->para_field.DataLength);
      return(DataValue);
   }
   else
      return NULL;
}


/*-------------------------------------------------------------*
|  Use this procedure to get the key and data lengths from     |
|  a KRAM file that has been opened by KramOpen. You must      |
|  supply the file pointer that was returned by KramOpen       |
|  in "KramFile".                                             |
*-------------------------------------------------------------*/
void KramInfo(int *KeyLength,int *DataLength)
{
   *KeyLength = ParamPtr->para_field.KeyLength;
   *DataLength = ParamPtr->para_field.DataLength;
}
```

```
/*-------------------------------------------------------------*
|   Function    : KramUpdate();                                |
|   Usage       : int KramUpdate(FILE *KramFile,void *KeyValue, |
|                                void *DataValue);             |
|   Description : write new data to same data                  |
|   Return      : TRUE if can be modified                      |
|                 FALSE otherwise                              |
*--------------------------------------------------------------*/
int KramUpdate(FILE *KramFile,void *KeyValue,void *DataValue)
{
   int DataOffset;

   if ( find_slot(KramFile,KeyValue,&DataOffset) == TRUE ) {
      memmove(*ParamPtr->para_field.DataPtr+DataOffset+ParamPtr->para_field.KeyLength,
              DataValue,ParamPtr->para_field.DataLength);
      ParamPtr->para_field.BlockModified = TRUE;
      return 1;
   }
   else
      return 0;
}

int KramDelete(FILE *KramFile,void *KeyValue)
{
   int DataOffset;              /* offset of deleted data */
   int LastOffset;              /* offset of last data record */
   int RecordLength;
   int FoundFirst,FoundLast;
   int FirstBlock,LastBlock;
   int j;
   int TempBlockNumber;
   int del_begin,del_end;
   int handle;
   long filesize;

   RecordLength = ParamPtr->para_field.KeyLength + ParamPtr->para_field.DataLength;

   /*---------- find data slot ------------ */
   if (find_slot(KramFile,KeyValue,&DataOffset) == TRUE) {
      /*------- find last data slot -------- */
      if (find_in_slot("aaaa",&LastOffset,RecordLength) == 0) {
         /*-- set LastOffset to last data record by finding incorrect key --*/
         LastOffset -= RecordLength;
         if ((LastOffset == DataOffset) && (LastOffset == 0) &&
         (ParamPtr->para_field.HighBlock != 1)) {
            /* only one data record in the block */

            /* find current block */
            find_id_block(&del_begin,&del_end,ParamPtr->para_field.CurrentBlock);

            if(ParamPtr->para_field.HighBlock != ParamPtr->para_field.CurrentBlock) {
               /* find HighBlock index */
               find_id_block(&FirstBlock,&LastBlock,ParamPtr->para_field.HighBlock);
```

```
            /* set to current block */
            for(j=FirstBlock;j<=LastBlock;j++)
               *(*ParamPtr->para_field.IndexPtr+j) =
                     ParamPtr->para_field.CurrentBlock;

            /* move data block from highblock to current block */
            SeekBlock(KramFile,ParamPtr->para_field.HighBlock);
            fread(ParamPtr->para_field.DataPtr,DATASIZE,1,KramFile);
            SeekBlock(KramFile,ParamPtr->para_field.CurrentBlock);
            fwrite(ParamPtr->para_field.DataPtr,DATASIZE,1,KramFile);
         }

         /* set current block to upper block or lower block */
         if (del_begin == 0)      /* first block */
            TempBlockNumber = *(*ParamPtr->para_field.IndexPtr+del_end+1);
         else
            TempBlockNumber = *(*ParamPtr->para_field.IndexPtr+del_begin-1);

         for(j=del_begin;j<=del_end;j++)
            *(*ParamPtr->para_field.IndexPtr+j) = TempBlockNumber;

         ParamPtr->para_field.HighBlock--;
         ParamPtr->para_field.CurrentBlock = 0;
         ParamPtr->para_field.BlockModified = FALSE;
         handle = fileno(KramFile);
         filesize = filelength(handle);
         filesize -= DATASIZE;
         chsize(handle,filesize);
         rewind(KramFile);
         fwrite(ParamPtr,sizeof(KramParamType),1,KramFile);
         fwrite(*ParamPtr->para_field.IndexPtr,
               sizeof(KramIndexType),1,KramFile);
         return 1;
      }
      if (LastOffset != DataOffset) {
         memmove(*ParamPtr->para_field.DataPtr+DataOffset,
                *ParamPtr->para_field.DataPtr+LastOffset,
                RecordLength);
      }
   }
   memset(*ParamPtr->para_field.DataPtr+LastOffset,0,RecordLength);
   ParamPtr->para_field.BlockModified = TRUE;
   return 1;
  }
  else
   return 0;
}


/*-------------------------------------------------------------*
¦  Function     :  find_slot()                                  ¦
¦  Usage        :  int find_slot(FILE *KramFile,void *KeyValue, ¦
```

```
¦                  int *DataOffset);                                ¦
¦  Description   :  find slot of data of key "KeyValue"             ¦
¦  Return        :  0 for data slot not found                       ¦
¦                   if data found  1  is returned                   ¦
¦                   and DataOffset is set to offset of data         ¦
*-----------------------------------------------------------------*/
int find_slot(FILE *KramFile,void *KeyValue,int *DataOffset)
{
   long HashVal;
   int BlockNumber;
   int RecordLength;
   int i,j,k;
   int KeepLooking;
   int SlotFound;

   RecordLength = ParamPtr->para_field.KeyLength + ParamPtr->para_field.DataLength;

   HashVal = HashCode(KeyValue) & (INDEXCOUNT - 1);

   BlockNumber = *(*ParamPtr->para_field.IndexPtr+HashVal+1);

   if ( ParamPtr->para_field.CurrentBlock != BlockNumber ) {
      if ((ParamPtr->para_field.BlockModified)
      && (ParamPtr->para_field.CurrentBlock != 0)) {
         ParamPtr->para_field.BlockModified = FALSE;
         SeekBlock(KramFile,ParamPtr->para_field.CurrentBlock);
         fwrite(ParamPtr->para_field.DataPtr,DATASIZE,1,KramFile);
      }
      SeekBlock(KramFile,BlockNumber);
      fread(ParamPtr->para_field.DataPtr,DATASIZE,1,KramFile);
      ParamPtr->para_field.CurrentBlock = BlockNumber;
   }
   if(find_in_slot(KeyValue,DataOffset,RecordLength) == 1)
      return 1;
   else
      return 0;
}

/*-----------------------------------------------------------------*
¦   Function    :  find_id_block();                                 ¦
¦   Usage       :  int find_id_block(int *FirstBlock,               ¦
¦                        int *LastBlock,int BlockNumber);           ¦
¦   Description :  find block of index of BlockNumber               ¦
¦   Return      :  None                                             ¦
*-----------------------------------------------------------------*/
void find_id_block(int *FirstBlock,int *LastBlock,int BlockNumber)
{
   int FoundFirst,FoundLast;
   int j;

   FoundFirst = FALSE;
   FoundLast = FALSE;
   *LastBlock = INDEXCOUNT -1;
   for (j=0;j<INDEXCOUNT;j++) {
      if ((*(*ParamPtr->para_field.IndexPtr+j) == BlockNumber) &&
```

```
      (!FoundFirst)) {
         *FirstBlock = j;
         FoundFirst = TRUE;
      }
      if ((*(*ParamPtr->para_field.IndexPtr+j) != BlockNumber) &&
      FoundFirst && (!FoundLast)) {
         *LastBlock = j-1;
         FoundLast = TRUE;
         break;
      }
   }
}

/*-------------------------------------------------------------*
|   Function    : find_in_slot();                             |
|   Usage       : int find_in_slot(int *DataOffset);          |
|   Description : find data record in current data block      |
|   Return      : 0 for data not found                        |
|                 1 for data record found and DataOffset set to |
|                 offset of data                              |
*-------------------------------------------------------------*/
static int find_in_slot(void *KeyValue,int *DataOffset,int RecordLength)
{
   int KeepLooking;
   int i;
   int SlotFound;

   KeepLooking = TRUE;
   SlotFound = FALSE;
   i = 1;
   *DataOffset = 0;

   while (KeepLooking && (i <= (DATASIZE/RecordLength))) {
      if ( *(*ParamPtr->para_field.DataPtr+(*DataOffset)) == 0 )
         KeepLooking = FALSE;
      else {
         memmove(TempKeyValue,*ParamPtr->para_field.DataPtr+(*DataOffset),
                        ParamPtr->para_field.KeyLength);
         if (!strncmp(TempKeyValue,KeyValue,ParamPtr->para_field.KeyLength)) {
            SlotFound = TRUE;
            KeepLooking = FALSE;
         }
         else {
            *DataOffset += RecordLength;
               i += 1;
         }
      }
   }
   if(SlotFound)
      return 1;
   else
      return 0;
}
```

```
/*------------------- return.c ----------------------*/

#include "..\header\return.h"
#include "..\header\window.h"

int quit()
{
   return OK;
}

int cancel()
{
   return CANCEL;
}
```

```
/*------------------ mail.c -------------------------*/

#include<stdio.h>
#include<dir.h>
#include<string.h>
#include<time.h>
#include<dos.h>

#include"..\header\lldef.h"
#include"..\header\entry.h"
#include"..\header\window.h"
#include"..\header\vc.h"
#include"..\header\telphone.h"
#include"..\header\scr.h"
#include"..\header\kram.h"
#include"..\header\mail.h"
#include"..\header\user.h"
#include"..\header\mesad.h"

static char intro(void);
static int endmail();
static int ex_func();
static int did(char mode);
static int getmsg(unsigned char id[]);
static void storemsg(char firstnum);
static int check_id(unsigned char input[]);
static int setupwin();
static int status(int statusin,int order);
static int service(int mode);
static int check_call(char *);
static int new_mes(char *f_name);
static int call_out();
static int GetCallOut();
static void update_time(int mes_time,int p_or_r);

extern struct edit edit_st;
extern char voc_dir[];

FILE *KramFile;
struct LINKLIST msglist;
struct LINKLIST tout_list;
struct tout_struct tout_template;
char wid[USERIDNUM];
int tout_handle;

int mail()
{
   char mode;
   int temp;
   int done;

   setupwin();
   while(1) {
      if ((temp = detect()) == 0) {
         status(PIN,-1);
```

```
            done = FALSE;
            while(!done) {
               play_vdata("firstmes.voc");
               play_vdata("intro.voc");
               play_vdata("choice.voc");
               if ((mode = getdtmf()) != -1) {
                  if (mode == '#')
                     done = ex_func();
                  if (strchr("123456789",mode) != NULL)
                     done = did(mode);
                  if (mode == '0')
                     done = TRUE;
               }
               else done = TRUE;
            }
            hook_off();
            delay(500);
            status(DTR,-1);
         }
         else  if (temp == ESC) {
            endmail();
            return '\r';
         }
         else if (temp == -1) {
            status(TOUT,-1);
            call_out();
            hook_off();
            status(DTR,-1);
         }
      }
   }


int ex_func()
{
   char mode,temp;
   int ans;
   char filename[13];

   play_vdata("askid.voc");
   ans = check_id(wid);
   switch (ans) {
      case 1 :  status(PIN,1);
                if(edit_st.first_flag) {                  /* if first flag set */
                   strcpy(filename,edit_st.id);
                   strcat(filename,"firs.voc");
                   play_vdata(filename);                  /* play first message */
                   delete(filename);                      /* and delete out */
                   edit_st.first_flag = 0;                /* update first flag */
                   KramUpdate(KramFile,edit_st.id,edit_st.name);
                }
                while (((mode = intro()) != '0') && (mode != -1)) {
                   switch (mode) {
                      case '1' :  storemsg('0');
                                  break;
```

```
                    case '2' :  getmsg(wid);
                                break;
                    case '3' :  play_vdata("telout.voc");
                                  /* ask for tel out or not */
                                temp = getdtmf();
                                if (temp == '0')
                                   break;
                                if (strchr("123456789",temp) != NULL)
                                   GetCallOut(temp);
                                break;
                  }
                }
                return FALSE;
      default : return TRUE;
   }
}

static int did(char mode)
{
   char ex_call[EX_NUM];
   int status;
   char temp;

   ex_call[0] = mode;
   if(check_call(ex_call) != 1)  {              /* check enquiry number */
      play_vdata("no_call.voc");
      return TRUE;                          /* return and exit */
   }
   else {
      status = enquiry(ex_call);
      switch ( status ) {
         case BUSY :
                    flash(500);                 /* to return to user */
                    play_vdata("busy.voc");
                    play_vdata("destid0.voc");
                    break;
         case RECEIVER_HOOK_ON :
                    return TRUE;                /* hook off to enquiry */
         case RECEIVER_NOT_HOOK_ON :
                    flash(500);
                    play_vdata("nohookon.voc");
                    play_vdata("destid0.voc");
                    break;
      }
      temp = getdtmf();
         switch(temp) {
            case '1' :
            case '2' :
            case '3' :
            case '4' :
            case '5' :
            case '6' :
            case '7' :
            case '8' :
            case '9' : storemsg(temp);
```

```
                        break;
                default  : break;
            }
    }
    return FALSE;
}


static char intro()
{
    play_vdata("function.voc");
    return(getdtmf());
}


int getmsg(unsigned char id[])
{
    struct ffblk ffblk;
    int done;
    char vocfile[MAXPATH];
    char temp;
    struct time t_start,t_stop;

    llsetlist(&msglist);

    memset(vocfile,0,sizeof(vocfile));

    service(LISTEN);
    strcpy(vocfile,voc_dir);
    strcat(vocfile,"\\");
    strcat(vocfile,id);
    strcat(vocfile,"*.voc");

    if ((done = findfirst(vocfile,&ffblk,0)) == 0) {

        while(!done && new_mes(ffblk.ff_name)) {
            lladd(ffblk.ff_name);
            done = findnext(&ffblk);
        }
        llhead();

        while((&msglist)->clp != NULL) {
            gettime(&t_start);
            play_vdata((&msglist)->clp->item);
            gettime(&t_stop);
            play_vdata("delete.voc");
            if (getdtmf() == '1')
                update_time(t_stop.ti_min-t_start.ti_min,PLAY);
                delete((&msglist)->clp->item);
            lldelete();
        }
    }
    else
        play_vdata("nomsg.voc");
```

```
    service(NONE);
}

/*-------------------------------------------------------------*
|   Function    : storemsg();                                 |
|   Usage       : void storemsg(char firstnum);               |
|   Description : store message to user                       |
|                 if firstnum = 0 then storemsg of vmb        |
|                 else firstnum is the first number of user id|
|   Return      : None                                        |
*-------------------------------------------------------------*/
static void storemsg(char firstnum)
{
   int idint;
   char filename[13];
   int temp;
   unsigned char uid[USERIDNUM] ;       /* first USERIDNUM chars of file name */
   char forder[4];                      /* next 3 chars of filename */
   struct time t_start,t_stop;          /* time of vmb record */

   service(STORE);
   memset(uid,0,sizeof(uid));
   if (firstnum == '0')
      play_vdata("destid.voc");
   else
      *uid = firstnum;
   if(check_id(uid) == 1) {
      edit_st.filenum = ++edit_st.filenum % 8;
      temp = KramUpdate(KramFile,edit_st.id,edit_st.name);

      strcpy(filename,uid);
      itoa(edit_st.filenum,forder,2);
      if((temp = strlen(forder)) != 3) {
         memmove(forder+3-temp,forder,temp+1);
         memset(forder,'0',3-temp);
      }
      strcat(filename,forder);
      strcat(filename,"0.voc"); /* have not been heard */
      play_vdata("startmsg.voc");
      gettime(&t_start);
      record_vdata(filename);
      gettime(&t_stop);
      update_time(t_stop.ti_min-t_start.ti_min,RECORD);
   }
   service(NONE);
}

/*-------------------------------------------------------------*
|   Function    :  check_id()                                 |
|   Usage       :  int check_id(unsigned char *input,char mode);|
|   Description :  get user id and get user record            |
|   Return      :  1 and user record in edit_st               |
|                  0 for get_id error                         |
|                  -1 for not user id                         |
```

```
*-------------------------------------------------------------*/
int check_id(unsigned char input[])
{
   int counter;
   int pscounter;

   counter = 0;

   if (*input != 0) {           /* first number for other service except vmb */
      if (get_id(input+1,USERIDNUM-1) != -1) {
         memset(&edit_st,0,sizeof(edit_st));
         strcpy(edit_st.id,input);
         if(KramRead(KramFile,edit_st.id,edit_st.name) != NULL)
            return 1;
         else {
            counter++;
            play_vdata("idagain.voc");
         }
      }
      else return 0;
   }
                        /* check_id for vmb */
   do  {
      if (get_id(input,USERIDNUM) != -1) {
         memset(&edit_st,0,sizeof(edit_st));
         strcpy(edit_st.id,input);
         if(KramRead(KramFile,edit_st.id,edit_st.name) != NULL)
            return 1;
         else
            counter++;

         if ( counter < 3)
            play_vdata("idagain.voc");
      }
      else
         return 0;
   }  while (counter < 3);

   return -1;
}


int setupwin()
{

    establish_window(10,7,69,13,WHITE,BLUE,1);
    window_title(" System Status ",0x1f);

    establish_window(10,16,69,22,WHITE,BLUE,1);
    window_title(" SERVICE ",0x1f);

    write_string(14,9, "Status     :  ",0x1f);
    write_string(14,10,"User name  :  ",0x1f);
    write_string(14,11,"User Id    :  ",0x1f);
    write_string(26,18,"Listening Stored Message",0x1f);
```

```
    write_string(26,20,"    Storing Message     ",0x1f);
    status(DTR,-1);
    return 1;
}

int status(int statusin,int order)
{

   switch (statusin) {
      case DTR  :
         write_string(28,9,"Detect Ringing",0x1f);
         break;
      case PIN  :
         write_string(28,9,"Phone In      ",0x1f);
         break;
      case TOUT :
         write_string(28,9,"Call out      ",0x1f);
         break;
   }

   if(order != -1) {
      write_string(28,10,edit_st.name,0x1f);
      write_string(28,11,edit_st.id,0x1f);
   }
   else {
      write_string(28,10,"                    ",0x1f);
      write_string(28,11,"                    ",0x1f);
   }
}


int service(int mode)
{
   switch(mode) {

   case NONE :
          write_string(26,18,"Listening Stored Message",0x1f);
          write_string(26,20,"    Storing Message     ",0x1f);
          break;
   case LISTEN :
          write_string(26,18,"Listening Stored Message",0x71);
          break;
   case STORE  :
          write_string(26,20,"    Storing Message     ",0x71);
          break;
   }
}

static int endmail()
{
   delete_window();
   delete_window();
}
```

```
/*-------------------------------------------------------------*
|   Function    :  check_call();                              |
|   Usage       :  int check_call(char *call);                |
|   Description :  Check whether there is enquiry number      |
|   Return      :  1 for enquiry number correct               |
|                  0 otherwise                                |
*-------------------------------------------------------------*/
static int check_call(char *call)
{
/*  In this case enquiry number is composed of two digit.        */
/*  To be consistent with function description, check input call */
/*  with enquiry call of your PABX                               */
/*  return 1 for enquiry number correct, 0 otherwise             */

   call++;              /* skip first number */
   *call++ = getdtmf();
   *call = '\0';
   return 1;
}

static int new_mes(char *fname)
{
   if (*(fname+7) == '0')
      return 1;
   else
      return 0;
}

static int call_out()
{
   int status;

   llsetlist(&tout_list);
   if(ll_length() != 0) {
      llhead();
      llretrieve((char *) &tout_template);
      status = tel_out(tout_template.tel_no);
      switch ( status ) {
         case RECEIVER_HOOK_ON    :
                     play_vdata("intro.voc");
                     play_vdata(tout_template.filename);
                     delete(tout_template.filename);
                     lldelete();
                     break;
         case BUSY :                    /* append to tail */
         case RECEIVER_NOT_HOOK_ON :    /*  append to tail */
                     lldelete();
                     lladdtail((char *) &tout_template);
                     break;
      }
   }
}

static int GetCallOut(char firstnum)
{
```

```
    unsigned char uid[USERIDNUM] ;       /* first USERIDNUM chars of file name */
    char filename[13];
    char ext[] = "tout.voc";
    int tel_len;

    *uid = firstnum;

    llsetlist(&tout_list);

    if (get_id(uid+1,USERIDNUM-1) != -1) {
       memset(&edit_st,0,sizeof(edit_st));
       strcpy(edit_st.id,uid);
       if(KramRead(KramFile,edit_st.id,edit_st.name) != NULL) {
          strcpy(filename,uid);          /* get call out file name */
          strcat(filename,ext);

          strcpy(tout_template.filename,filename);       /* save in list */
          tel_len = strchr(edit_st.tel_no,' ') - edit_st.tel_no;
          strncpy(tout_template.tel_no,edit_st.tel_no,tel_len);
          *(tout_template.tel_no+tel_len+1) = '\0';
          strcpy(tout_template.uid,edit_st.id);
          lladdtail((char *) &tout_template);

          play_vdata("startmsg.voc");
          record_vdata(filename);
       }
       else
          play_vdata("fasleid.voc");
    }
}

static void update_time(int mes_time,int p_or_r)
{
    int t_left;

    t_left = atoi(edit_st.time_left);

    if (p_or_r == PLAY)                    /* get message, more time left */
       t_left += mes_time;
    else                                   /* store message, less time left */
       t_left -= mes_time;
    itoa(edit_st.time_left,t_left,10);
    KramUpdate(KramFile,edit_st.id,edit_st.name);
}
```

```
/*--------------------- mesad.c ---------------------------*/

#include<string.h>
#include<stdlib.h>
#include<dir.h>
#include<dos.h>
#include<fcntl.h>
#include<io.h>
#include<stdio.h>
#include<sys\stat.h>

#include"..\header\mesad.h"
#include"..\header\window.h"
#include"..\header\entry.h"
#include"..\header\scr.h"
#include"..\header\lldef.h"
#include"..\header\return.h"
#include"..\header\vc.h"
#include"..\header\user.h"

#define    CHDIRFG      WHITE
#define    CHDIRBG      5

extern struct edit edit_st;
extern FILE *KramFile;

struct LINKLIST dellist;
char voc_dir[MAXPATH] = "C:\\VOC\\SYSVOC";
int to_all_flag;

static int del_message();
static int first_mes();
static int ch_dir();
static int del();
static int chd();
static int dtrue(unsigned int,char *);
static int daynum(int,int,int);
static int record();
static int play();
static int check_heard(char *);


static char cur_dir[80];
static char uid[USERIDNUM];

static char mask[] = "_____________";

FIELD mes_menu[] = {
   {4,3,5,'D'," Delete Message File  ",NULL,"User Admin Delete Message File"},
   {4,5,5,'S',"  Set First Message   ",NULL,"Set First Message for User Greeting"},
   {4,7,5,'Q',"        Quit          ",NULL,"Return to Main Menu"},
   {0}
};

struct df_template {
```

```
        char del_file[10];
        char day_num[5];
} df;


RADIO  heardornot[] = {
   {8,8,1,"Have not been heard"},
   {8,9,0,"Have been heard"},
   {8,10,0,"All messages"},
   NULL
};

FIELD del_field[] = {
   {1,4,19,1,df.del_file,mask+5,"Deleted File's Name"},
   {1,6,19,1,df.day_num,mask+10,"Days of messages that have not been heard"},
   {2,7,8,1,(char *) heardornot,"Message Type","Type of message to be deleted"},
   {4,4,43,'D', "    Delete    ",NULL,"Delete File"},
   {4,6,43,'C',"  Change Dir  ",NULL,"Change Directory of Message File"},
   {4,8,43,'O',"      Ok      ",NULL,"Return to Message File Menu"},
   NULL
};

RADIO to_whom[] = {
   {8,3,1,"To all user"},
   {8,4,0,"to specific user"},
   NULL
};

FIELD f_m_field[] = {
   {2,2,7,1,(char *) to_whom,"Set Message to","Set First Message to user"},
   {4,2,37,'R',"  Record Message  ",NULL,"Record First Message"},
   {4,4,37,'P',"   Play Message   ",NULL,"Play First Message"},
   {4,6,37,'O',"        Ok        ",NULL,"Return to Previous Menu"},
   NULL
};

FIELD s_m_field[] = {
   {1,2,20,1,uid,mask+sizeof(mask)-USERIDNUM,"User id to whom first message is set"},
   {4,3,7,'O',"   Ok    ",NULL,"Save Message to User"},
   {4,3,25,'C',"    Cancel    ",NULL,"Return to Menu"},
   NULL
};


int message()
{
   mes_menu[0].mask.func = del_message;
   mes_menu[1].mask.func = first_mes;
   mes_menu[2].mask.func = quit;

   del_field[3].mask.func = del;
   del_field[4].mask.func = ch_dir;
   del_field[5].mask.func = quit;

   f_m_field[1].mask.func = record;
```

```
    f_m_field[2].mask.func = play;
    f_m_field[3].mask.func = quit;

    s_m_field[1].mask.func = quit;
    s_m_field[2].mask.func = cancel;

    establish_window(10,7,70,16,ENTRYFG,ENTRYBG,1);
    win_dstr(33,4,"Delete Message File",LIGHTCYAN|(ENTRYBG<<4));
    win_dstr(33,6,"Record Message to User",LIGHTCYAN|(ENTRYBG<<4));
    win_dstr(33,8,"Return to Main Menu",LIGHTCYAN|(ENTRYBG<<4));

    data_entry(mes_menu,1,1);
    delete_window();

}

static int del_message()
{
    establish_window(7,9,70,22,ENTRYFG,ENTRYBG,1);
    win_dstr(8,4,"File  ",ENTRYFG|(ENTRYBG<<4));
    win_dstr(8,6,"Date No.",ENTRYFG|(ENTRYBG<<4));
    data_entry(del_field,1,1);
    delete_window();
}

static int first_mes()
{
    char fname[MAXPATH];
    char userfname[MAXPATH];

    establish_window(7,8,68,18,WHITE,ENTRYBG,1);
    win_dstr(8,9,"Status : ",WHITE|(ENTRYBG<<4));
    data_entry(f_m_field,1,1);
    if(f_m_field[0].fvalue.value->value == 1)
       to_all_flag = 1;
    else {
       establish_window(7,18,68,23,WHITE,MAGENTA,1);
       win_dstr(7,2,"User Id  :",BLACK|(CYAN<<4));
       while(data_entry(s_m_field,1,1)) {
          strcpy(edit_st.id,uid);
          if (KramRead(KramFile,edit_st.id,edit_st.name) == NULL)
             error_message(" User Not Found ");
          else {
             edit_st.first_flag = 1;
             KramUpdate(KramFile,edit_st.id,edit_st.name);
             strcpy(userfname,voc_dir); strcat(userfname,"\\");
             strcat(userfname,uid);
             strcat(userfname,"firs.voc");
             strcpy(fname,voc_dir); strcat(fname,"\\");
             strcat(fname,"MESSAGEF.TMP");
             newcopy(fname,userfname);
          }
       }
       delete_window();
       unlink(fname);
```

```
    }
    delete_window();
}

static char dir[30];
static char dirmask[] = "______________________________";

static int ch_dir()
{
    int temp;
    FIELD dir_template[] = {
        {1,3,16,1,dir,dirmask,"New Directory Name"},
        {4,4,5,'C'," Chdir ",NULL,"Change Directory"},
        {4,4,20,'O',"  Ok   ",NULL,"Return to Upper Menu"},
        {4,4,35,'C'," Cancel ",NULL,"Return to Upper Menu without Chdir"},
        NULL
    };

    dir_template[1].mask.func = chd;
    dir_template[2].mask.func = quit;
    dir_template[3].mask.func = cancel;

    strcpy(cur_dir,"C:\\");
    getcurdir(0,cur_dir+strlen("C:\\"));


    setmem(dir,30,0);
    establish_window(12,10,60,16,CHDIRFG,CHDIRBG,1);
    win_dstr(5,3,"New Dir",WHITE|(CHDIRBG<<4));
    window_title(" Change Directory ",CHDIRFG|(CHDIRBG<<4));
    temp = data_entry(dir_template,1,1);
    if (temp == CANCEL)
        chdir(cur_dir);
    delete_window();

}

static int del()
{
    int temp;
    int done;
    struct ffblk ffblk;

    llsetlist(&dellist);

    done = findfirst(df.del_file,&ffblk,0);

    while(!done) {
        if (dtrue(ffblk.ff_fdate,df.day_num) && check_heard(ffblk.ff_name))
            lladd(ffblk.ff_name);
        done = findnext(&ffblk);
    }
    llhead();
    if (ll_length() != 0)
        select_list(5,10,&dellist,ll_length());
```

```
    else
       error_message(" There is no message of specified day ");

    while((&dellist)->clp != NULL) {
       unlink((&dellist)->clp->item);
       lldelete();
     }
     init_template();
     return '\t';
 }


 static int chd()
 {
    int temp;
    char *tp;
    char tdir[MAXDIR];

    strcpy(tdir,dir);
    if((tp = memchr(tdir,' ',sizeof(tdir))) != NULL)
       memset(tp,0,sizeof(tp));

    if (chdir(dir) != 0) {
       error_message(sys_errlist[errno]);
       return '\r';
    }
    else
       return '\t';                  /* for going to next field */
 }

 static int dtrue(unsigned int dayin,char *daywant)
 {
    int i_daywant;
    int i_dayin;
    int mon,day,year;
    struct date d;

    day = dayin  &   0x001f;      /* bit 0-4 : Day */
    mon = (dayin  &  0x01e0)>>5; /* bit 5-8 : Month */
    year = (dayin & 0xfe00)>>9;  /* bit 9-15 : Year */
    i_dayin = daynum(day,mon,year);

    getdate(&d);                  /* get current date */
    day = d.da_day;
    mon = d.da_mon;
    year = d.da_year-1980;
    i_daywant = daynum(day,mon,year);
    return ( i_daywant-i_dayin >= atoi(daywant));
 }


 static int daynum(int day,int mon,int year)
 {
    int i_dayin;
```

```
   i_dayin = day;
   while(mon) {
      switch (mon) {
         case 1  :
         case 3  :
         case 5  :
         case 7  :
         case 8  :
         case 10 :
         case 12 : i_dayin += 31;
                   break;
         case 4  :
         case 6  :
         case 9  :
         case 11 : i_dayin += 30;
                   break;
         case 2  : if ( year % 4 )
                      i_dayin += 29;
                   else
                      i_dayin += 28;
                   break;
      }
      mon--;
   }
   i_dayin += year*365;
   if (year%4)
      i_dayin += year/4;

   return i_dayin;
}

static int record()
{
   char fname[13] = "MESSAGEF.TMP";

   if(f_m_field[0].fvalue.value->value == 1)
      strcpy(fname,"firstmes.voc");

   win_dstr(17,9,"  Recording... Press any key to stop  ",BLACK|(WHITE<<4));
   record_vdata(fname);
   win_dstr(17,9,"                                      ",ENTRYFG|(ENTRYBG<<4));
}

static int play()
{
   char fname[13] = "MESSAGEF.TMP";

   if(f_m_field[0].fvalue.value->value == 1)
      strcpy(fname,"firstmes.voc");

   win_dstr(17,9,"  Playing ... Press any key to stop  ",BLACK|(WHITE<<4));
   play_vdata(fname);
   win_dstr(17,9,"                                      ",ENTRYFG|(ENTRYBG<<4));
}
```

```
static int check_heard(char *fname)
{
   int temp;

   temp = rad_to_code(heardornot);
   switch (temp) {
      case 1 : if (*(fname+7) == '0')
                  return 1;
               else
                  return 0;
      case 2 : if (*(fname+7) == '1')
                  return 1;
               else
                  return 0;
      case 4 : return 1;
   }
}

static int newcopy(char *src,char *des)
{
   int shandle;                  /* handle of source */
   int dhandle;                  /* handle of destination */
   int byte_read;                /* bytes read */
   int byte_write;               /* bytes written */
   long f_length;                /* length of source file */
   unsigned char buff[BUFFLEN];        /* buffer */

   shandle = open(src,O_RDWR);
   if (shandle == -1) {
      error_message(sys_errlist[errno]);
      return -1;
   }
   f_length = filelength(shandle);      /* get source file length */

   dhandle = _creat(des,FA_ARCH);       /* create destination file */
   if (dhandle == -1) {
      error_message(sys_errlist[errno]);
      return -1;
   }

   while ( f_length ) {
      byte_read = _read(shandle,buff,BUFFLEN);
      byte_write = write(dhandle,buff,byte_read);
      if (byte_write != byte_read) {
         error_message(" Disk Full ");
         return -2;
      }
      if (byte_write == -1) {
         error_message(" Copy Error ");
         return -3;
      }
      f_length -= byte_read;
   }
   close(shandle);
   close(dhandle);
```

```
    return 0;
}

int play_vdata(char *fname)
{
    char fullname[MAXPATH];

    strcpy(fullname,voc_dir);
    strcat(fullname,"\\");
    strcat(fullname,fname);
    play_data(fullname,-1,0);
}

int record_vdata(char *fname)
{
    char fullname[MAXPATH];

    strcpy(fullname,voc_dir);
    strcat(fullname,"\\");
    strcat(fullname,fname);
    record_data(fullname);
}

int delete(char *fname)
{
    char fullname[MAXPATH];

    strcpy(fullname,voc_dir);
    strcat(fullname,"\\");
    strcat(fullname,fname);
    remove(fullname);
}
```

```
/*---------------------- telad.c ----------------------*/

#include<stdlib.h>

#include"..\header\telad.h"
#include"..\header\window.h"
#include"..\header\scr.h"
#include"..\header\entry.h"
#include"..\header\telphone.h"
#include"..\header\menu.h"
#include"..\header\return.h"

static int ring_time();
static int cout_time();
static int set_cout();
static int set_up();


extern int ring_no_receiver;            /* times of ring back for no receiver */
extern int retry_enquiry;               /* time of retry enquiry */
extern int Maxcount;                    /* count for tel out */

char r_template[6];
char ms[] = "_____";

FIELD tel_menu[] = {
   {4,3,5,'R',"    Ringing Time    ",NULL,"Specify Ring Back Tone Time"},
   {4,5,5,'T',"  Time of Call Out  ",NULL,"Specify waiting time before call out"},
   {4,7,5,'Q',"       Quit         ",NULL,"Return to previous menu"},
   {0}
};


int tel_ad()
{
   tel_menu[0].mask.func = ring_time;
   tel_menu[1].mask.func = cout_time;
   tel_menu[2].mask.func = quit;

   establish_window(6,8,74,17,TEXTFG,TEXTBG,1);
   win_dstr(30,4,"Specify Ring Back Tone Time",WHITE|(TEXTBG<<4));
   win_dstr(30,6,"Specify Waiting Time before Call Out",WHITE|(TEXTBG<<4));
   win_dstr(30,8,"Return to Previous menu",WHITE|(TEXTBG<<4));
   data_entry(tel_menu,1,1);
   delete_window();
}


int ring_time()
{
   FIELD r_t[] = {
      {1,2,20,1,r_template,ms,"Ringing time of No Receiver Hook on"},
      {4,3,5,'S', "   Set up   ",NULL,"Return to menu"},
      {4,3,30,'O',"     Ok     ",NULL,"Cancel and return to menu"},
      NULL
```

```
    };

    r_t[1].mask.func = set_up;
    r_t[2].mask.func = quit;

    establish_window(15,14,65,19,WHITE,ENTRYBG,1);
    win_dstr(5,2,"Ringing Time",WHITE|(ENTRYBG<<4));
    data_entry(r_t,1,1);
    delete_window();
}

int cout_time()
{
    FIELD c_t[] = {
       {1,2,20,1,r_template,ms,"Time before each call out"},
       {4,3,5,'S', "   Set up   ",NULL,"Set up time before each call out"},
       {4,3,30,'O',"     Ok     ",NULL,"Return to previous menu"},
       NULL
    };

    c_t[1].mask.func = set_cout;
    c_t[2].mask.func = quit;

    establish_window(15,14,65,19,WHITE,ENTRYBG,1);
    win_dstr(5,2,"Call out Time",WHITE|(ENTRYBG<<4));
    data_entry(c_t,1,1);
    delete_window();

}

static int set_up()
{
    int  temp;

    temp = atoi(r_template);
    if(temp == 0) {
       error_message("  Input Value Error  ");
       init_template();
       return '\t';
    }
    else {
        ring_no_receiver = temp;
    }
}

static int set_cout()
{
    double temp;

    temp = atof(r_template);
    if (temp == 0) {
       error_message("  Input Value Error  ");
       init_template();
       return '\t';
    }
```

```
    else {
       Maxcount = (60*60*temp)/0.03;
    }
}
```

```
/*------------------ user.c ------------------------*/

#include"..\header\user.h"
#include"..\header\entry.h"
#include"..\header\window.h"
#include"..\header\kram.h"
#include"..\header\return.h"

#include<stdlib.h>
#include<string.h>

static char mask[] = "_________________________"; /* 25 chars */

struct edit  edit_st;

RADIO time[] = {
   {7,9,1," 5 Minutes"},
   {7,10,0,"10 Minutes"},
   {7,11,0,"15 Minutes"},
   NULL
};

RADIO service_type[] = {
   {38,9,1,"Voice Mail"},
   {38,10,0,"General Information"},
   {38,11,0,"Phone Back"},
   NULL
};

FIELD edit_template[] = {
   {1,3,17,1,edit_st.name,mask,"User's Name"},
   {1,3,57,1,edit_st.id,mask+20,"User's Identity"},
   {1,5,17,1,edit_st.password,mask+15,"User's Password"},
   {1,5,57,1,edit_st.tel_no,mask+15,"User's Telephone Number"},
   {1,7,29,1,edit_st.time_left,mask+19,"Time left in Mail Box"},
   {2,8,7,1,(char *) time,"Time of mailbox","Set time of User's MailBox"},
   {3,8,38,1,(char *) service_type,"User's Service","Set User's Service"},
   {4,13,7,'A'," Add User  ",NULL,"Add User's Details"},
   {4,13,26,'D'," Delete User  ",NULL,"Delete User's Details"},
   {4,13,45,'E'," Edit User  ",NULL,"Change User's Details"},
   {4,15,18,'U'," Update  ",NULL,"Cancel User Administration"},
   {4,15,38,'O', "  Ok  ",NULL,"Return to main menu"},
   {0}
};

extern FILE *KramFile;

static int add_user();
static int delete_user();
static int update_user();
static int rad_to_time();
static int edit_user();


int edit()
```

```
{
   edit_template[7].mask.func = add_user;
   edit_template[8].mask.func = delete_user;
   edit_template[9].mask.func = edit_user;
   edit_template[10].mask.func = update_user;
   edit_template[11].mask.func = quit;

   establish_window(5,5,73,22,YELLOW,BLUE,1);
   win_dstr(5,3,"Name : ",0x1f);
   win_dstr(45,3,"User id : ",0x1f);
   win_dstr(5,5,"Password : ",0x1f);
   win_dstr(45,5,"Tel : ",0x1f);
   win_dstr(5,7,"Time Left in MailBox :          Minutes",0x1f);
   data_entry(edit_template,1,1);
   delete_window();
}

int add_user()
{
   int temp;
   char tempc[TIMELEFT];

   temp = rad_to_time(time);
   itoa(temp,tempc,10);
   strcpy(edit_st.time_left,tempc);
   edit_st.time = rad_to_code(time);
   edit_st.service = rad_to_code(service_type);
   if (KramAdd(KramFile,edit_st.id,edit_st.name) == FALSE ) {
      error_message("Add User Error");
      return'\r';
   }
   else {
      init_template();
      return'\t';
   }
}

int delete_user()
{
    if ( KramDelete(KramFile,edit_st.id) == FALSE) {
       error_message("Delete User Error");
       return '\r';
    }
    else {
       init_template();
       return '\t';
    }
}

static int old_time;    /* old service time */
static int update_user()
{
   int t_left;          /* time left in minutes */
   int c_t_left;        /* time left in pack code */
   int time_sel;        /* service time selected */
```

```
    t_left = atoi(edit_st.time_left);
    time_sel = rad_to_time(time);
    if ( time_sel > old_time)
       itoa(time_sel-old_time+t_left,edit_st.time_left,10);
    else  {
       if ( t_left>=time_sel)
          itoa(time_sel-old_time+t_left,edit_st.time_left,10);
       else
          strcpy(edit_st.time_left,"0");
    }
    edit_st.time = rad_to_code(time);
    edit_st.service = rad_to_code(service_type);
    if(KramUpdate(KramFile,edit_st.id,edit_st.name) == FALSE) {
       error_message(" Update User error ");
       return '\r';
    }
    else {
       init_template();
       return '\t';
    }
}

int edit_user()
{
    if(KramRead(KramFile,edit_st.id,edit_st.name) == NULL) {
       error_message("Reading User Error");
       init_template();
       return '\t';
    }
    else {
       code_to_rad(edit_st.time,time);
       code_to_rad(edit_st.service,service_type);
       old_time = rad_to_time(time);
       return '\t';
    }
}



static int rad_to_time(RADIO *rad)
{
    int temp,i;

    temp = 0;
    for(i=1;rad->row;i++,rad++)
       if (rad->value)
          temp = i;

    switch (temp) {
       case 1 : return 5;
       case 2 : return 10;
       case 3 : return 15;
    }
}
```

บรรณานุกรม

VP-870 PC VOICE CARD USER MANUAL;

ROGER SESSIONS, "REUSABLE DATA STRUCTURES FOR C", PRIME COMPUTER INCORPORATED FRAMINGHAM, MA.,PRENTICE HALL, Englewood Cliffs, New Jersey 07632;

วารสารเซมิคอนดัคเตอร์ อิเลคทรอนิคส์ เล่มที่ 88 ประจำเดือน กย.-ตค. 2531 ,สำนักพิมพ์ ซีเอ็ดยูเคชัน ;

วารสารเซมิคอนดัคเตอร์ อิเลคทรอนิคส์ เล่มที่ 104 ประจำเดือน มค.-กพ. 2534 ,สำนักพิมพ์ ซีเอ็ดยูเคชัน ;

วารสารเซมิคอนดัคเตอร์ อิเลคทรอนิคส์ เล่มที่ 105 ประจำเดือน มีค.-เมย. 2534 ,สำนักพิมพ์ ซีเอ็ดยูเคชัน ;

หนังสือ "รวมโครงงาน เล่ม 6", สำนักพิมพ์ ซีเอ็ดยูเคชัน

HERBERT SCHILDT. C POWER USER GUIDE : MCGRAW HILL. 1988