แบบจำลองโครงข่ายคอมพิวเตอร์รูปดาว

THE SIMULATION MODEL OF STAR NETWORK

โดย

นางสาวราณี สิทธิแก้ว

นางสาวรุ่งนภา ไทยประยูร

นาวสาววารีรัตน์ ปลื้มรุ่งโรจน์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา ๒๕๓๖

14

ปริญญานิพนธ์.
ประจำปีการศึกษา ๒๕๓๖
เรื่อง แบบการจำลองโครงข่ายรูปดาว
(The Simulation Model Of Star Network)

ภาควิชา วิศวกรรมโทรคมนาคม
คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้า
เจ้าคุณทหาร ลาดกระบัง

ผู้จัดทำ

1. นางสาวราณี สิทธิแก้ว 33100317
2. นางสาวรุ่งนภา ไทยประยูร 33100319
3. นางสาววารีรัตน์ ปลื้มรุ่งโรจน์ 33100338

.....อาจารย์ที่ปรึกษา
(ดร.สุวิพล สิทธิชีวภาค)

แบบจำลองโครงข่ายคอมพิวเตอร์รูปดาว

THE SIMULATION MODEL OF STAR NETWORK

โดย น.ส.ราณี สิทธิแก้ว
น.ส.รุ่งนภา ไทยประยูร
น.ส.วารีรัตน์ ปลื้มรุ่งโรจน์

อาจารย์ที่ปรึกษา ดร.สุวิพล สิทธิชีวภาค

บทคัดย่อ

โครงงานวิจัยนี้เป็นการจำลองโครงข่ายคอมพิวเตอร์แบบสตาร์ เพื่อศึกษาการทำงานของระบบโดยที่ระบบจะแบ่งออกเป็น 2 ส่วน คือ ส่วนของฮาร์ดแวร์ และส่วนของซอฟแวร์ ซึ่งส่วนของฮาร์ดแวร์นั้นได้สร้างการ์ดเพื่อใช้เป็นอุปกรณ์ เชื่อมต่อเทอร์มินัล 4 เทอร์มินัลทางพอร์ตแบบอนุกรมเข้ากับคอมพิวเตอร์ ศูนย์กลางที่เป็นตัวควบคุมการติดต่อ และส่วนของซอฟแวร์ประกอบด้วย โปรแกรมควบคุม และ โปรแกรมแอปปลิเคชั่น โปรแกรมควบคุมจะประกอบด้วย ส่วนของโปรแกรมที่ใช้ควบคุมเซนเตอร์และเทอร์มินัล โดยจะให้เซนเตอร์ทำงานแบบโพลลิง คอยรับการร้องขอจากเทอร์มินัล นอกจากนี้เทอร์มินัลเราได้เพิ่ม เทคนิคการเรสิเดนท์โปรแกรมเข้าไปเพื่อรอรับการเรียกจากเซนเตอร์ ในขณะที่กำลังทำงานอื่นอยู่ ส่วนโปรแกรมแอปปลิเคชั่นได้ออกแบบไว้สำหรับการเลือกทำงาน 2 แบบ คือ แบบ ชัทโหมดที่ติดต่อได้ 2 ทางกับ แบบส่งข้อมูลเป็นไฟล์ โดยจะขึ้นอยู่กับยูสเซอร์ว่าจะเลือกการติดต่อแบบใด

ABSTRACT

This project has simulated a Star-type Network. In order to study about Star Network system. The system is distinguished into 2 section ; Hardware and Software. In hardware section ,we make an additional card inserted in center' s expantion slot in order to connect for terminal with center via serial port. And software section consists of 2 part;Control programs and Application programs.The control programs consist of center's control program and terminal's controlprograms. Center will sequently poll every terminal. Terminal which want to communicate with another must send a request to center, and then terminal communicate with another terminal via center's switch. Resident program is also added to terminal. then it can do another job before it is interrupted. Application programs consist of 2 subprogram; chatmode(which communicate in full duplex)and transfer file. So user can select application which they want to use.

สารบัญ

เรื่อง	หน้า
บทที่ 1 บทนำ	1
บทที่ 2 ทฤษฎีและหลักการโพลลิงโปรโตคอล	2
2.1 โปรโตคอลที่ใช้กับสตาร์เน็ตเวอร์ค	2
2.2 คาต้าลิงค์โปรโตคอล	4
2.3 โพลลิงค์ดีเลย์	6
บทที่ 3 ทฤษฎีและโครงสร้างพอร์ตอนุกรมสื่อสาร	8
3.1 ส่วนเชื่อมต่ออุปกรณ์	8
3.2 โครงสร้างของพอร์ตอนุกรมสื่อสาร	9
3.3 การใช้งานพอร์ตอนุกรมสื่อสาร	12
บทที่ 4 การอินเทอร์รัพและการจัดการ	15
4.1 ซอฟแวร์อินเทอร์รัพ	16
4.2 ฮาร์ดแวร์อินเท่อร์รัพ	17
4.3 การเขียนโปรแกรมฝังตัวเบื้องต้น	17
4.4 ลักษณะของโปรแกรมฝังตัวที่ใช้ในโครงงาน	19
บทที่ 5 การทำงานและอัลกอริทึม	22
5.1 อัลกอริทึมของเซ็นเตอร์	23
5.2 อัลกอริทึมการสวิทช์ของเซ็นเตอร์	24
5.3 อัลกอริทึมของเทอร์มินัล	25
5.4 อัลกอริทึมของอินเทอร์รัพเซอร์วิทรูทีน	26
5.5 อัลกอริทึมของป๊อปอัพเมนู	27
บทที่ 6 ผลการทดลองและสรุป	28
แนวการทางพัฒนา	29
บทแทรก	30
ก. สตาร์แลนที่ใช้รูปแบบ OSI Model	30
ข. เทคโนโลยีและองค์ประกอบ	45
ค. โปรแกรมการทำงาน	53
ง. วงจรพอร์ตอนุกรมสื่อสาร	84
หนังสืออ้างอิง	85
กิตติกรรมประกาศ	86

บทที่ 1
บทนำ

รูปแบบและโปรโตคอลที่ใช้ในระบบโลคัลแอเรียเนตเวอร์ค (LAN)นั้นแตกต่างกันไป รูปแบบที่มีอยู่ก็ได้แก่ แบบบัส แบบริงและแบบดาว ซึ่ง 2 แบบแรก จะใช้แพร่หลายกว่าแบบดาวและมีโปรโตคอลมากมายที่ใช้กับ 2 รูปแบบนี้ แต่อย่างไรก็ตามแบบดาวก็เป็นรูปแบบที่เริ่มมีใช้กันมานาน เช่น ระบบชุมสายโทรศัพท์ศูนย์กลาง

แบบดาว มีข้อเสียหลายอย่างจึงทำให้ไม่นิยมใช้กันในระบบโลคัลแอเรียเนตเวอร์ค เนื่องจากที่เซนเตอร์จะมีโครงสร้างที่ซับซ้อน ความเชื่อถือได้ต่ำ โหนด(node)ทุกโหนดต้องเชื่อมต่อกับเซนเตอร์ซึ่งค่าใช้จ่ายในส่วนนี้ก็สูงด้วย แต่อย่างไรก็ตามแบบดาวก็มีข้อดีที่แบบอื่นไม่มี เช่น การเชื่อมต่อกับแบบจุดต่อจุดเหมาะแก่การใช้สายออพติค(Optic fiber) ค่าทรูพุท(Throughput)สูง ทำให้ใช้ประโยชน์จากแบนวิดท์ที่มีอยู่ได้เต็มที่ สนับสนุนในการส่งข้อมูลความเร็วสูง ความซับซ้อนจะอยู่ที่ตัวเซนเตอร์แต่โครงสร้างของโหนดแต่ละโหนดจะเป็นแบบง่ายๆธรรมดาๆ ฉะนั้นจึงเสียค่าใช้จ่ายต่ำกว่าแบบอื่นๆ การขยายระบบก็ทำได้ง่ายระบบการทำงานโดยทั่วไปดีกว่า โครงข่ายอื่นๆ

หลายปีที่ผ่านมา โครงข่ายรูปดาวและโปรโตคอลได้ถูกพัฒนา ทั้งของ IBM, HUBNET, BELLCOM ซึ่งเป็นโปรโตคอลที่เกิดการชนได้อย่างอิสระ แต่อย่างไรก็ตามการชนกันมีโอกาสเกิดขึ้นน้อยมาก แต่ละแบบถูกออกแบบมาเพื่อลดความยุ่งยากซับซ้อนที่ตัวเซนเตอร์

บทที่2 ทฤษฎีและหลักการโพลลิงโปรโตคอล

2.1 โปรโตคอลที่ใช้กับสตาร์เนตเวอร์ค

โปรโตคอลที่จะกล่าวถึงมี 3 แบบ คือ

1. การโพลลิง(POLLING)
2. แคเรียเซนซ์ มัลติเปิลแอคเซสวิดท์คอลลิชั่นดีเทคชั่น(CSMA/CD)
3. คอลลิชั่นฟรีโปรโตคอล(COLLISION FREE PROTOCOL)

การโพลลิงที่ใช้ในแบบสตาร์มี 2 ประเภท

-โรล คอลโพลลิง(ROLL CALL POLLING) ตัวเซนเตอร์จะตรวจสอบความต้องการส่งข้อมูลในแต่ละโหนด เซนเตอร์จะเป็นตัวถามโหนดว่าจะส่งข้อมูลหรือไม่ ถ้าโหนดมีข้อมูลที่จะส่ง โหนดก็จะทำการส่งข้อมูลออกมา ในทางกลับกัน ถ้าโหนดไม่มีข้อมูลที่จะส่ง มันจะส่ง เนคกาทีฟ แอคโนเรจ (NEGATIVE ACKNOWLEDGE)

-ฮับ โพลลิง(HUB POLLING)แบบนี้การตอบสนองจะเร็วกว่า การโพล จะส่งผ่านจากโหนดหนึ่งไปยังอีกโหนด โดยไม่มีเซนเตอร์เข้ามาแทรก แต่อย่างไรก็ตามเซนเตอร์จะต้องเป็นคนเริ่มการโพล

กรณีของสตาร์ริง(STAR RING)ซึ่งเป็นการผสมผสานระบบ สตาร์กับริงเข้าด้วยกัน โดยความยาวริงต้องสั้นมากๆและบัฟเฟอร์(BUFFER)ต้องอยู่บนริง การออกแบบระบบเช่นนี้จะช่วยลดการชนกันให้น้อยลงเมื่อใช้โทเคน ริง(TOKEN RING) ระบบโพลลิงจะสิ้นเปลืองเวลา ยิ่งถ้าเพิ่มจำนวน โหนดเข้ามาในระบบมากขึ้น ก็ยิ่งทำให้เกิดความล่าช้าเพราะต้องเช็คในแต่ละโหนด ว่าจะส่งข้อมูลหรือไม่ อย่างไรก็ตาม การ โพลลิงก็เป็นวิธีที่ง่ายที่สุด

CSMA/CD

เนตเวอร์คที่ใช้โปรโตคอลนี้มี ไฟเบอร์เนต(FIBERNET)และไฟเบอร์เนตทู(FIBERNET II)ซึ่งต่างกันที่ลักษณะของเซนเตอร์ในไฟเบอร์เนตการลิงค์(LINK) กับเซนเตอร์ เป็นแบบแพสซีฟ(PASSIVE) คือเซนเตอร์ทำหน้าที่ส่งสัญญาณต่อไปยังผู้รับเท่านั้น แต่ในไฟเบอร์เนตทู เซนเตอร์จะเปลี่ยนจากสัญญาณแสงเป็นสัญญาณทางไฟฟ้าแล้วจึงขยายสัญญาณ จากนั้นจึงแปลงสัญญาณไฟฟ้ากลับเป็นสัญญาณแสงส่งไปให้ด้านรับ หลักการทำงานของโปรโตคอลนี้คือ ต้องดูก่อนว่าช่องสัญญาณว่างหรือไม่ถ้าช่องสัญญาณไม่ว่าง โหนดที่จะส่งข้อมูลต้องรอจนกว่าช่องสัญญาณจะว่างและหากในการส่งข้อมูลนั้นเกิดการชนกันจากโหนดอื่น ก็จะต้องพยายามส่งข้อมูลใหม่อีกครั้ง

ประโยชน์ของ CSMA / CD ก็คือทำงานได้ดีเมื่อมีโหลดน้อย นอกจากนี้ยังเป็นโปรโตคอลที่ได้มาตราฐาน แม้จะใช้กับโทโปโลจี (TOPOLOGY) ที่ไม่ได้มาตราฐานก็ตาม ทำให้ง่ายต่อการใช้ของผู้ขายและผู้ใช้ ในอีเทอร์เนต(ETHERNET) เพียงแค่เปลี่ยนฮาร์ดแวร์(HARDWARE)ในชั้นฟิสิคัล(PHISICAL) เท่านั้น แต่โปรโตคอลแบบนี้ก็มีข้อเสียเหมือนกัน คือเมื่อโหลดมากขึ้น การแย่งชิงช่องสัญญาณก็มากขึ้น ค่าความล่าช้าในการส่งข้อมูลจะมากขึ้น

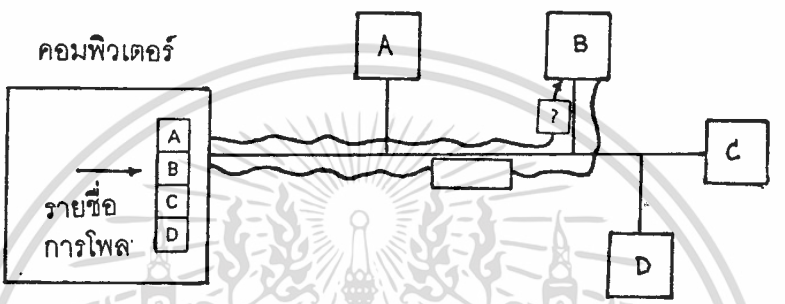
โปรโตคอลที่ 3 ถูกค้นพบพร้อมๆกันโดย IBM ,มหาวิทยาลัยโตรอนโต,ศูนย์ค้นคว้าของ BELL โปรโตคอลนี้อนุญาตให้แต่ละโหนดส่งข้อมูลได้ตามต้องการ เมื่อข้อมูลมาถึงที่ เซนเตอร์ใน ขณะที่เซนเตอร์ว่าง เซนเตอร์ก็จะทำการส่งแพคเกต(packet)ที่ได้รับนั้นทันที แต่ถ้าเซนเตอร์ไม่ว่าง การส่งก็จะล้มเหลว ซึ่งผลของการส่งจะสำเร็จหรือล้มเหลวนั้น โหนดที่เป็นตัวส่งข้อมูลจะรู้ได้โดย การตรวจดูแพคเกต ที่โหนดรับเข้ามา ถ้าไม่เหมือนกับที่ตัวเองส่งไป โหนดก็จะสรุปว่า ตัวเองส่งไม่ สำเร็จ มันจะพยายามส่งใหม่ ส่วนในกรณีที่มีการเรียกเข้าเซนเตอร์พร้อมๆกัน เซนเตอร์จะเป็น ตัวตัดสินใจเองว่าจะเลือกติดต่อกับโหนดไหน

2.2 ดาต้าลิงค์โปรโตคอล

เนื่องจากในการศึกษาโครงงานนี้เราใช้การโพลลิงเป็นโปรโตคอลในส่วนดาต้าลิงค์ ดังนั้นจึงขอยกเอาเนื้อหาเรื่องการโพลลิงมากล่าวโดยละเอียด

โพลลิงเป็นเทคนิคที่สถานีควบคุมใช้ควบคุมการส่งข้อมูลของสถานีลูกข่าย หรือจะกล่าวว่าเป็นขบวนการไต่ถามแต่ละสถานี ว่ามีข้อมูลจะส่งหรือไม่

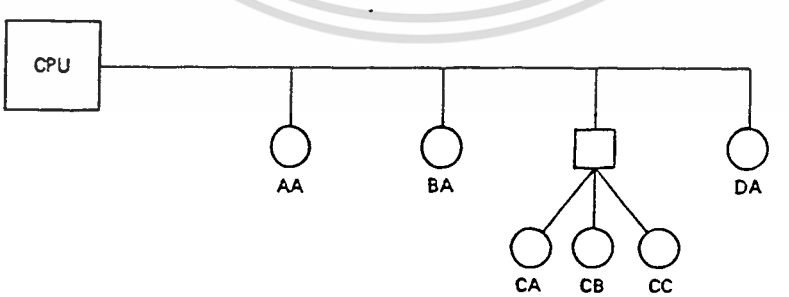
ในกรณีที่ง่ายที่สุด เซนเตอร์จะตรวจสอบไปยังเทอร์มินัลต่างๆ ตามลำดับ เพื่อสอบถามว่ามีข้อมูลส่งหรือไม่ ซึ่งเทอร์มินัลอาจจะมีข้อมูลหรือไม่มีก็ได้ ดังนั้นการสนองตอบการโพลจะเป็นไปได้ 2 กรณีคือ มีข้อมูลส่งหรือไม่มีข้อมูล(ซึ่งจะต้องส่งสัญญาณไปบอกให้เซนเตอร์ได้รู้)



รูปที่ 1.1 การโพลลิงบนสายมัลติดรอพ

รูป1.1 แสดงให้เห็นการโพลที่มี 4 เทอร์มินัล เซนเตอร์จะมีรายชื่อแต่ละเทอร์มินัลที่แสดงลำดับในการถูกโพล จากรูปเทอร์มินัล บี ถูกโพลและมีการส่งข้อมูล

เราสามารถให้ความสำคัญกับเทอร์มินัลใดเทอร์มินัลหนึ่งมากกว่าเทอร์มินัลอื่นได้ โดยการใส่แอดเดรสของเทอร์มินัลนั้นในรายชื่อการโพลบ่อยกว่าเทอร์มินัลอื่น ตัวอย่างเช่น เราอาจจะให้เทอร์มินัลเอ มีความสำคัญมากกว่าเพื่อนโดยอาจใส่แอดเดรส การโพลโดยเรียงเป็น เอ - บี - เอ- ซี- เอ - ดี - เอ - บี เทอร์มินัลอาจจะถูกตัดออกจากรายชื่อการ โพลลิงก็ได้เมื่อต้องการ(เช่น กรณีเทอร์มินัลนั้นเกิดเหตุขัดข้อง)



รูปที่ 1.2 มัลติดรอพ ลาย -เทอร์มินัลแอดเดรสซิง(Multidrop line-terminal addressing)

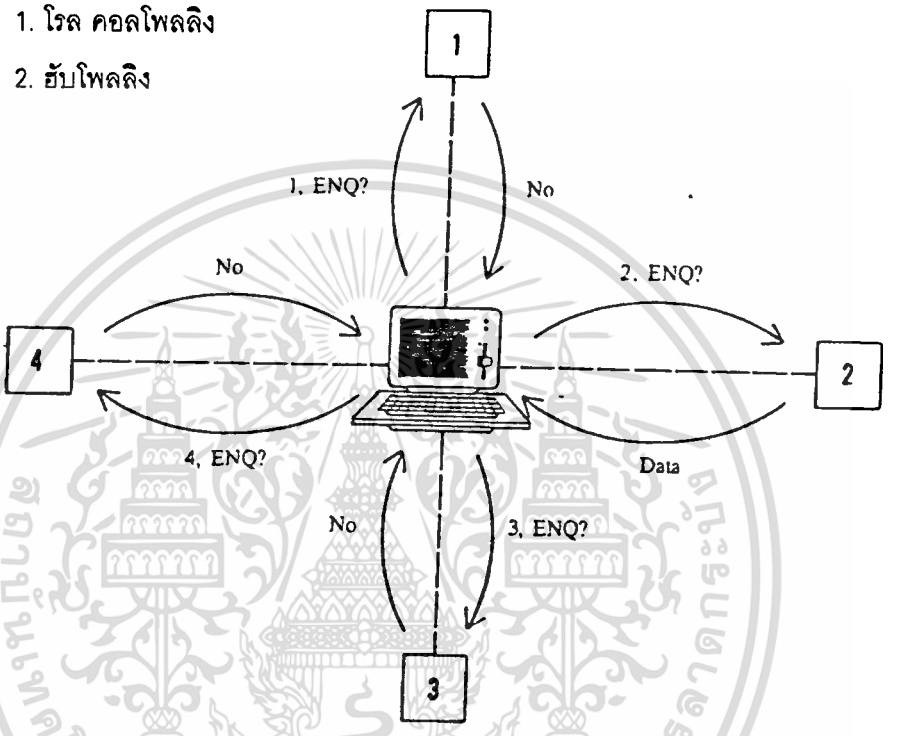
ในระบบการโพลลิงนั้น เราต้องมีเทอร์มินัลที่อ้างอิงแอดเดรสได้ รูปที่1.2 แสดงสายที่เชื่อม 4 เทอร์มินัลและมี 1 เทอร์มินัลที่ต่อกับอีก 3 เทอร์มินัลย่อย การอ้างแอดเดรสจึงแบ่งเป็น 2 ระดับ คือ ตัวแรกชี้ดรอพ(drop) และตัวที่ 2 บอกเทอร์มินัลที่ดรอพนั้น

การโพลแต่ละครั้งนั้นเทอร์มินัลทุกตัวจะได้รับหมด แต่จะมีเพียงเทอร์มินัลที่แอดเดรสตรงกับแอดเดรสในการโพลเท่านั้นที่จะรับรู้ เทอร์มินัลที่เหลือจะไม่สนใจ กฎทั่วไปก็คือ ถ้าเทอร์มินัลพบว่าไม่ใช่แอดเดรสของตนมันก็จะไม่สนใจ เซนเตอร์จะส่งข้อมูลไปถามที่เทอร์มินัลแรกแล้วจะรอการตอบสนองกลับ ถ้าเซนเตอร์ไม่ได้รับการตอบกลับภายในช่วงเวลาที่กำหนดมันจะทำการสอบถามไปยังแอดเดรสถัดไปโดยอัตโนมัติ

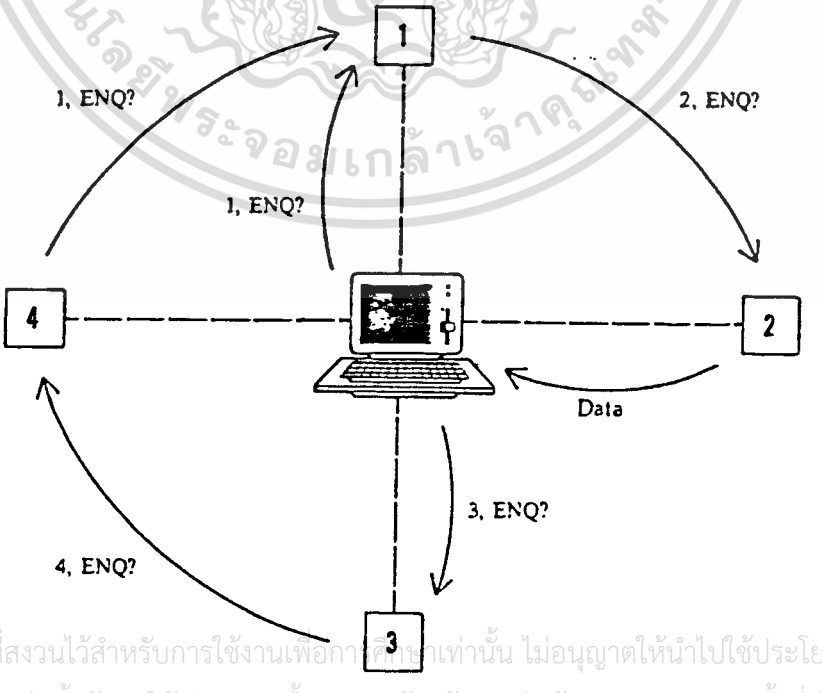
การโพลลิงจะมีอยู่ 2 ประเภท

1. โรล คอลโพลลิง
2. ฮับโพลลิง

a)



b)



รูป ที่ 1.3 a)โรล คอล โพลลิง b)ฮับ โพลลิง

ฮับโพลลิง (รูปที่ 1.3) การโพลเริ่มเมื่อ เซนเตอร์ โพลที่สถานีแรก ถ้าสถานีแรกไม่มีข้อมูลส่งมันจะไม่ตอบกลับไปยังเซนเตอร์โดยตรง แต่มันจะส่งการโพลไปให้สถานีต่อไป สถานีที่ 2 ก็สามารถส่งข้อมูลไปที่เซนเตอร์ได้ แล้วก็จะปล่อยให้เซนเตอร์ส่งการโพลไปยังสถานีที่ 3 หลังจากที่เซนเตอร์ได้รับข้อมูลจากเทอร์มินัลที่ 2 เรียบร้อยแล้ว ถ้าสถานีที่ 2 ไม่มีข้อมูลที่จะส่ง มันจะส่งการโพลไปที่สถานีที่ 3 แต่ละสถานีจะเปลี่ยนแอดเดรสในการโพลให้ตรงกับแอดเดรสของสถานีถัดไป วิธีนี้จะช่วยลดการโพลของเซนเตอร์ จึงเป็นการเพิ่มประสิทธิภาพ อย่างไรก็ตามการเพิ่มขบวนการในการเปลี่ยนแอดเดรส และส่งผ่านการโพลต่อไปยังสถานีถัดไปของสถานีลูกข่ายนั้น ทำให้มันมีราคาแพงขึ้น

แบบฮับโพลลิงจะมีข้อเสียที่แบบโรลคอลไม่มี คือในระบบ โรลคอลนั้นถ้าเซนเตอร์ไม่ได้รับการตอบกลับจากสถานีที่มันโพลภายในระยะเวลาที่กำหนด มันจะทำการโพลไปยังสถานีถัดไปโดยอัตโนมัติ แต่ในแบบฮับโพลลิงนั้น ถ้าเราไม่นำแอดเดรสของสถานีที่ขัดข้องออกไปจากระบบ และเมื่อแอดเดรสนั้นใส่ลงไปในการโพล ระบบก็จะหยุดชะงัก เพราะว่าที่สถานีลูกข่ายนั้นมีเพียงอุปกรณ์ที่เปลี่ยน แอดเดรสใหม่ในการโพลเท่านั้น มันจะไม่รู้เลยว่า แอดเดรสนั้นไม่มีการตอบสนองซึ่งรวมทั้งเซนเตอร์ด้วยที่ไม่รู้ โพลลิงมีข้อดีคือการชนของข้อมูลจะไม่เกิดขึ้น อีกทั้งเป็นการทำงานที่ง่ายไม่ซับซ้อน แต่ก็มีข้อเสียคือ สิ้นเปลืองเวลา ทำให้ล่าช้ากว่าแบบอื่น

2.3 โพลลิงดีเลย์(POLLING DELAY)

การโพลลิงเป็นขบวนการ ที่ไม่ได้เกิดขึ้นในทันทีทันใด ต้องใช้เวลาในการโพลแต่ละเทอร์มินัล ถ้าคำนวณเวลาที่โพลไม่สำเร็จในกรณีที่โครงข่ายเป็นดังรูป 15.4 ถ้าความเร็วในการส่ง 2400 บิท/วินาทีและระยะห่างเท่ากับ 1000 กิโลเมตร จะใช้เวลาประมาณ 150 มิลลิวินาที สำหรับการโพลที่เทอร์มินัลไม่มีข้อมูลที่จะส่ง เวลาที่ใช้ในการโพลทั้งหมดนั้นมีอยู่ 3 ส่วนคือ

1.เวลาที่ใช้ในการโพลจากเซนเตอร์ไปยังเทอร์มินัล

2.เวลาที่ใช้ในการส่งเนคกาทีฟแอคโนเลจ

3.ลูปดีเลย์(Loop delay)

สมมุติว่าเรามีข้อมูลที่ใช้ในการโพลทั้งหมด 15 ตัวอักษร และเราใช้รหัสแอสกี กับการส่งแบบซิงโครนัส เราจะใช้ 8 บิทต่อ 1 ตัวอักษร ดังนั้น เวลาที่ใช้ในการส่งการโพลจะเป็น

$$\text{เวลาที่ใช้} = \frac{\text{15 ตัวอักษร x 8 บิท/ 1 ตัวอักษร}}{\text{2400 บิท/วินาที}}$$

$$= \text{50 มิลลิวินาที}$$

และเวลาที่ใช้ในการส่งเนคกาทีฟแอคโนเลจก็สามารถคำนวณได้คล้ายกัน ถ้าให้ข้อมูลที่ใช้ในการส่งเนคกาทีฟแอคโนเลจประกอบด้วย 5 ตัวอักษร เวลาที่ใช้ในการส่งจะเป็น

$$\text{เวลาที่ใช้} = \frac{\text{5 ตัวอักษร x 8 บิท/ 1 ตัวอักษร}}{\text{2400 บิท/วินาที}}$$

$$= \text{17 มิลลิวินาที}$$

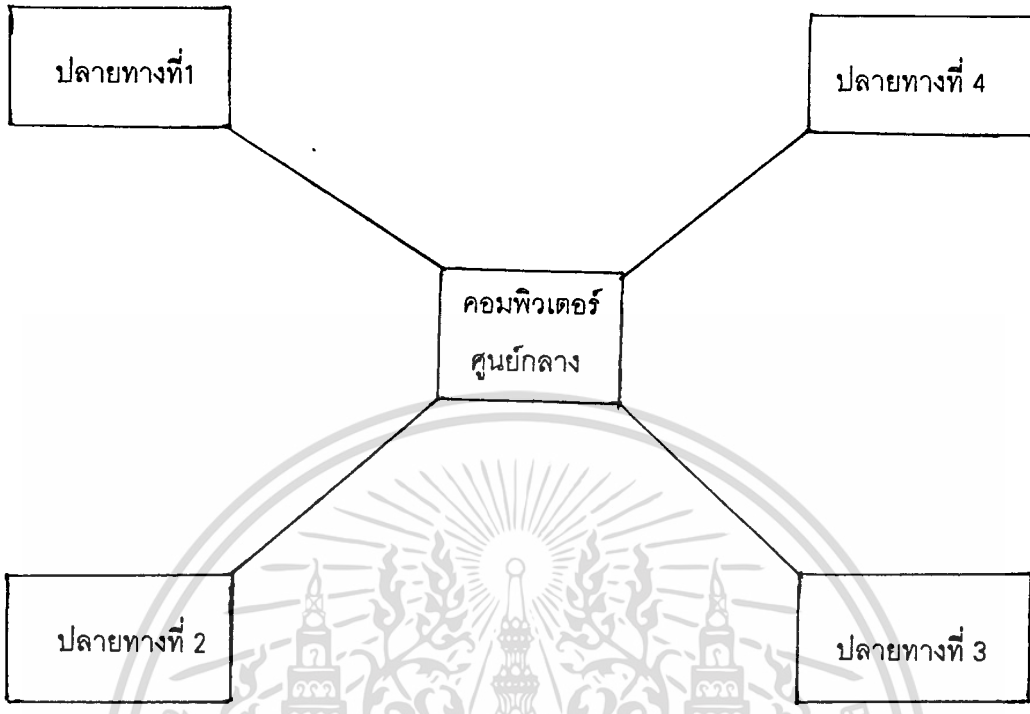
ส่วนดีเลย์ลูป คือผลรวมของเวลาที่ใช้ในการเดินทางของข้อมูลจากเซนเตอร์ไปยังเทอร์มินัล และกลับมาที่เซนเตอร์อีกครั้ง ซึ่งจะเกี่ยวข้องกับค่า โมเดมดีเลย์(Modem delay) ,พรอบพาเกชั่น ดีเลย์ (Propagation delay)

โมเดมดีเลย์ คือ ค่า โมดุเลชั่น/ ดีโมดุเลชั่น ดีเลย์ ที่ใช้ในการผ่านเข้าไปใน โมเดม ซึ่งจะต้องมีค่าดีเลย์นี้ทั้ง 2 ทางคือจาก เซนเตอร์ไปเทอร์มินัลและจากเทอร์มินัลกลับมาที่เซนเตอร์

พรอบพาเกชั่นดีเลย์ คือ ระยะเวลาที่สัญญาณไฟฟ้าใช้ในการเคลื่อนที่จากปลายสายที่ติดต่อกันจากด้านหนึ่งไปยังอีกด้านหนึ่ง

บทที่3 ทฤษฎีและโครงสร้างพอร์ตอนุกรมสื่อสาร

3.1 ส่วนเชื่อมต่ออุปกรณ์(Hardware interface)



การเชื่อมต่อโครงข่ายรูปดาว

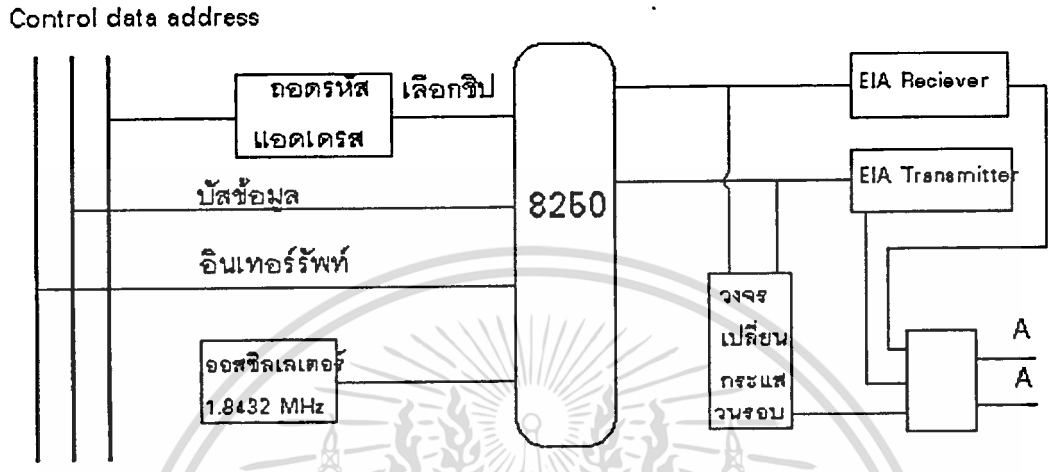
ในการจำลองโครงข่ายรูปดาวที่จัดทำขึ้นในงานวิจัยนี้ ตัวกลางสามารถที่จะเชื่อมต่อ กับปลายทาง 4 ปลายทาง โดยผ่านทางพอร์ตอนุกรมสื่อสาร แต่ในเครื่องมีพอร์ตอนุกรมสื่อสารอยู่ 2 พอร์ต คือ คอม1 (3F8) คอม2(2F8) จึงเพิ่มคอม3(3E8) และคอม4(2E8)

การทำงานของพอร์ตอนุกรมสื่อสารนั้นมี 8250 เป็นชิพหลักในการแปลงข้อมูลที่รับเข้ามาแบบอนุกรมแปลงเป็นขนานส่งให้ซีพียูไปประมวลผล และซีพียูส่งข้อมูลแบบขนานมาให้ 8250 แปลงเป็นอนุกรมส่งออกไปที่คอนเนคเตอร์RS232C แต่ก่อนที่สัญญาณจะถูกส่งออกไป ต้องแปลงให้อยู่ในมาตรฐานการเชื่อมต่อแบบ RS232C ซึ่งในวงจรที่ทำขึ้นมานั้นเราใช้ MC1488 แปลงสัญญาณจาก 0-5 โวลต์ เป็น 12 ~-12 โวลต์ และ MC1489 แปลงจาก 12~-12โวลต์มาเป็น 0~5โวลต์เป็นแรงดันที่ใช้ในคอมพิวเตอร์

การใช้งานโดยทั่วไปของการ์ด เหมือนกันทุกประการกับพอร์ตอนุกรมสื่อสารที่มีอยู่ในเครื่องคอมพิวเตอร์

3.2 โครงสร้างของพอร์ตสื่อสาร

พอร์ตสื่อสารของเครื่องไมโครคอมพิวเตอร์ 16 บิต ที่จะกล่าวถึงนี้เป็นระบบมาตรฐานตามแบบเครื่องไอบีเอ็มพีซีเอ็กซ์ที โครงสร้างบล็อกไดอะแกรมแสดงดังรูป



รูปที่ 1 โครงสร้างพอร์ตสื่อสารข้อมูลที่ใช้ 8250

พิจารณาได้ว่า 8250 เป็นตัวรับข้อมูลจากบัสของระบบ ซึ่งก็คือสลอตขนาด 31x2(62) ขาของระบบนั่นเองซีพียูติดต่อกับ 8250ในลักษณะที่เป็นพอร์ตอินพุทเอาท์พุท การจัดพอร์ตนี้กำหนดหมายเลขพอร์ตอย่างเจาะจงระบบไมโครคอมพิวเตอร์ ระบบไมโครคอมพิวเตอร์ 16 บิต มีพอร์ตสื่อสาร 2 พอร์ต คือ คอม1และคอม 2มีหมายเลขอินพุทเอาท์พุทพอร์ตดังตารางที่ 1

ตารางที่1 หมายเลขอินพุทเอาท์พุทของคอม1 และคอม 2

อินพุทเอาท์พุทสลอต				เลือกรีจิสเตอร์	สถานะDLAB
คอม1	คอม2	คอม3	คอม4		
3F8	2F8	3E8	2E8	บัฟเฟอร์ IX	DLAB=0(เขียน)
3F8	2F8	3E8	2E8	บัฟเฟอร์ RX	DLAB=0(อ่าน)
3F8	2F8	3E8	2E8	แลตช์ตัวหาร(LSB)	
3F9	2F9	3E9	2E9	แลตช์ตัวหาร(MSB)	
3F9	2F9	3E9	2E9	อีนาเบิลอินเทอร์รัพท์	
3FA	2FA	3EA	2EA	กำหนดอินเทอร์รัพท์	
3FB	2FB	3EB	2EB	ควบคุมสายสื่อสาร	
3FC	2FC	3EC	2EC	ควบคุมโมเดม	
3FD	2FD	3ED	2ED	แสดงสถานะสายสื่อสาร	
3FE	2FE	3EE	2EE	แสดงสถานะโมเด็ม	

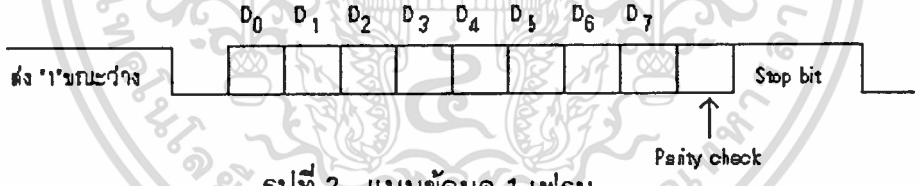
การเลือกหมายเลขอินพุตเอาต์พุตพอร์ตแยกเป็นสองกลุ่ม กลุ่มหนึ่งคือ com1 จะกำหนดหมายเลขพอร์ตจาก 3F8 ถึง 2F8 อีกกลุ่มหนึ่งถ้ากำหนดหมายเลขพอร์ตเป็น 2F8 ถึง 2FE ในการเลือกหมายเลขรีจิสเตอร์ภายใน กำหนดด้วยแอดเดรส 3 บิท A0,A1 และ A2 สำหรับการเลือก com1 และ com2 เราใช้แอดเดรส A8 เป็นตัวเลือกการเลือกนี้ กำหนดเป็นตารางได้ดังตารางที่ 2

การอินเทอร์รัพท์

จากที่เคยกล่าวถึงโครงสร้างการควบคุมอินเทอร์รัพท์ 8259 มาแล้ว 8259 ได้จัดลำดับการอินเทอร์รัพท์ไว้ 8 ระดับ สัญญาณรับอินเทอร์รัพท์ที่เข้าทางชิป 8259 มี 8 เส้น คือ IRQ0 - IRQ7 สำหรับกรณีของระบบสื่อสารอนุกรมได้กำหนดสัญญาณอินเทอร์รัพท์ไว้คือ ให้ IRQ4 เป็นสัญญาณอินเทอร์รัพท์ของระบบ com1 และ IRQ3 เป็นของ com2 ในการที่จะส่งสัญญาณอินเทอร์รัพท์ของรีจิสเตอร์ควบคุมโมเด็มได้รับการเซ็ตค่าเป็น 1 ก่อน จากนั้นข้อมูลอินเทอร์รัพท์ที่อยู่ในรีจิสเตอร์อีนาเบิลอินเทอร์รัพท์จะเป็นตัวสั่งการอินเทอร์รัพท์

รูปแบบข้อมูลที่รับหรือส่ง

การสื่อสารข้อมูลของระบบนี้ เป็นการสื่อสารแบบอะซิงโคนัส รูปแบบของข้อมูลจะมี Start bit , Parity bit และ Stop bit โครงสร้างของข้อมูลแต่ละเฟรมเป็นดังรูปที่ 2



รูปที่ 2 แบบข้อมูล 1 เฟรม

ข้อมูลบิทแรกของการส่งเป็น start bit ระบบจะส่ง start bit ก่อน หลังจากนั้นจะตามด้วยข้อมูล และแทรกด้วยบิทตรวจสอบ parity ตามด้วย stop bit ขนาดของข้อมุลมีค่าได้ตั้งแต่ 5 ถึง 8 บิท แต่ทั่วๆไปใช้ 8 บิท stop bit มีได้ 1 , 1.5 หรือ 2 บิท ค่าเหล่านี้ สามารถกำหนดลงไปในรีจิสเตอร์ควบคุมสายสื่อสาร

การต่อวงจรเข้ากับระบบ

บอร์ดอะแดปเตอร์สื่อสารนี้ โครงสร้างการเชื่อมต่อตามพอร์ต 3E8 -3EF กับ 2E8 - 2EF เป็น com3 และ com4 ตามลำดับ

ส่วนของบัสข้อมูล D0 - D7 จะผ่านบัฟเฟอร์คือ 74LS245 ก่อนเข้าสู่ 8250 ส่วนสัญญาณควบคุม 8250 ทั้ง 10 มีดังนี้

ขา master reset MR จะต่อโดยตรงกับสัญญาณรีเซ็ตของระบบ

ขา $\overline{ADS}$,$\overline{DISTR}$,$\overline{DOSTR}$ ต่อลง ground ทั้งนี้ เพราะแอดเดรสที่ต่อมาที่ชิป 8250 นี้ เป็นสัญญาณแอดเดรส ล้วนแล้วไม่ต้อง strobe อีก

ขา $\overline{DISTR}$ ต่อกับ IOR ของระบบ โดยผ่าน inverter 2 ตัว

ข่า $\overline{DOSTR}$ ต่อกับ IOW ของระบบ

ขา CS1,CS2 ต่อขึ้นเป็นลอจิก 1

ส่วนขา $\overline{CS2}$ มาจากการถอดรหัสให้เป็นพอร์ตของ com1 หรือ com2 ตามต้องการ

ขา XTAL2 ไม่ใช้ ส่วน XTAL1 ต่อมาจากออสซิลเลเตอร์ 1.8432 MHz ระบบวงจรของบอร์ดอะแดรปเตอร์สื่อสารเป็นดังรูปหน้าถัดไป

ส่วนของสัญญาณเอาท์พุตประกอบด้วย สัญญาณควบคุมโมเด็ม $\overline{DSR}$, $\overline{CTS}$ ต่อออกไปยังหัวต่อตามมาตรฐาน EIA RS 232C

ขา SIN , SOUT ต่อออกไปยังขาเอาท์พุตเช่นกัน แต่มีการปรับให้เป็นสัญญาณแรงดันตามมาตรฐาน EIA หรือเลือกส่งเป็นกระแสวนรอบก็ได้

3.3 การใช้งานพอร์ตอนุกรมสื่อสาร

หลักการ อินนิเชียลพอร์ท มีดังนี้

1 กำหนดความเร็วของการส่งข้อมูล

2 กำหนดรูปแบบของข้อมูลที่ใช้ในการส่ง

3 กำหนดให้มีการอินเทอร์รัพต์

การกำหนดความเร็วของการส่งข้อมูล มีขั้นตอนดังนี้

1 เซ็ท DLAB บิท 7 ใน LCR ที่แอดเดรส 3FB ให้เป็น 1

2 เซ็ทตัวหารของ LSB ที่แอดเดรส 3F8 มีค่าเท่ากับอัตราส่งข้อมูลที่แสดงในตาราง

3 เซ็ทตัวหารของ MSB ที่แอดเดรส 3F9 มีค่าเท่ากับอัตราส่งข้อมูลที่แสดงในตาราง

หมายเหตุ 3F8(COM1), 2F8(COM2), 3E8(COM3), 2E8(COM4)

ค่าตัวหารสำหรับการกำหนดอัตราบอด

อัตราบอด	ตัวหาร		ค่าผิดพลาด
	ฐานสิบ	ฐานสิบหก	
50	2304	900	-
75	1536	600	-
110	1047	417	0.026
134.5	857	359	0.058
150	786	300	-
300	384	180	-
600	192	0C0	-
1200	96	060	-
1800	64	040	-
2000	58	03A	0.69

การกำหนดรูปแบบของข้อมูลที่ใช้ในการรับส่ง

ผู้โปรแกรมจะต้องจะกำหนดค่าลงในรีจิสเตอร์ควบคุมสายสื่อสาร ดังนี้

บิท 0 และ 1 เป็นตัวกำหนดความยาวของข้อมูลในการรับส่งโดย

00 ข้อมูลขนาด 5 บิท

01 ข้อมูลขนาด 6 บิท

10 ข้อมูลขนาด 7 บิท

11 ข้อมูลขนาด 8 บิท

บิท 2 เป็นบิทที่ใช้ในการกำหนดจำนวน STOP BIT ถ้าเป็น 0 หมายถึงใช้ STOP BIT 1 BIT

ถ้าเป็น 1 ในกรณีข้อมูลขนาด 5 บิท มีความยาว STOP BIT เป็น 1.5 BIT ถ้าเป็นแบบ 6, 7 หรือ 8 บิท ความยาว STOP BIT เท่ากับ 2 BIT

บิท 3 บิทนี้แสดงการอีนาเบิลให้มีการตรวจสอบพาริตี้ โดยถ้าบิทนี้มีค่าเป็น 1 จะมีการเพิ่มพาริตี้

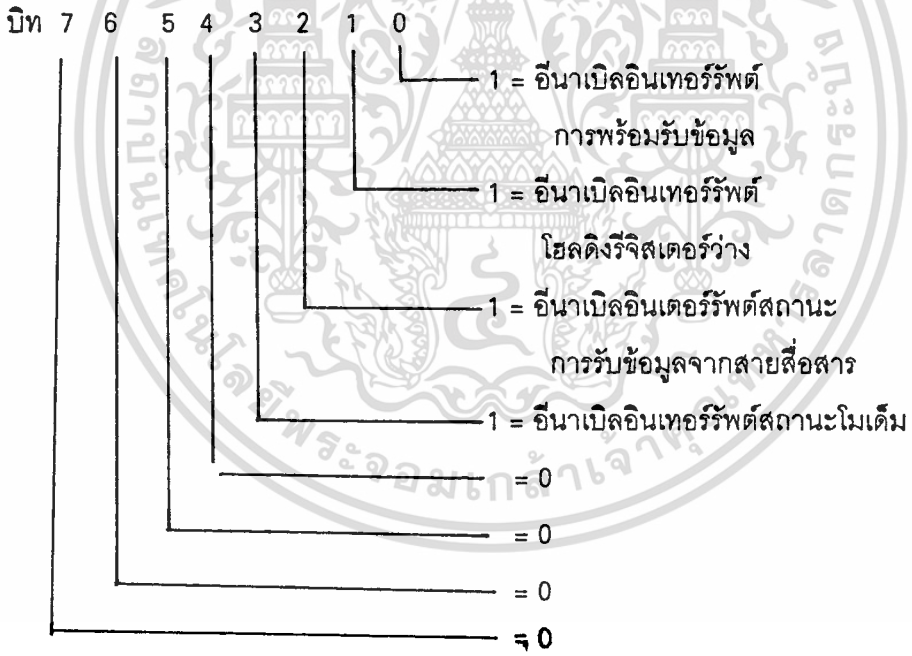
บิท 4 ถ้าเป็น 0 และบิท 3 มีค่าเป็น 1 มีการกำหนดเป็นพาริตี้คี่ ถ้าบิทนี้เป็น 1 จะเป็นพาริตี้คู่

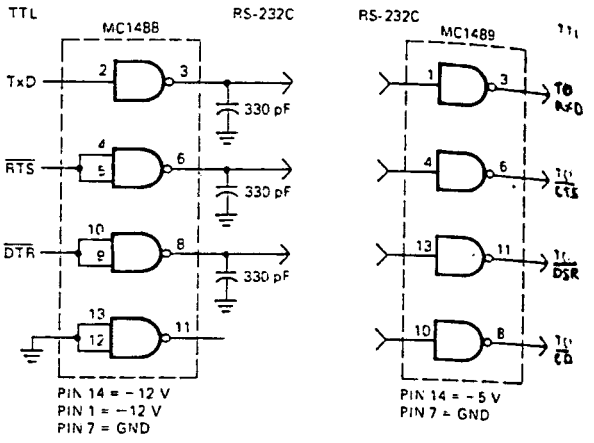
บิท 5 เมื่อบิท 3 มีค่าเป็น 1 บิท 4 มีค่าเป็น 1 บิท 5 มีค่าเป็น 1 จะมีการแทรกหรือตรวจสอบพาริตี้ ด้วยเงื่อนไขกำหนดให้เป็น 0 และถ้าบิท 4 มีค่าเป็น 0 บิท 3 มีค่าเป็น 1 จะมีการกำหนดพาริตี้บิทเป็น 1

บิท 6 เป็นบิทที่ควบคุมการเบรค เมื่อบิท 6 มีค่าเป็น 1 ขาSOUT จะได้รับการกำหนดให้เป็น 0 ตลอด

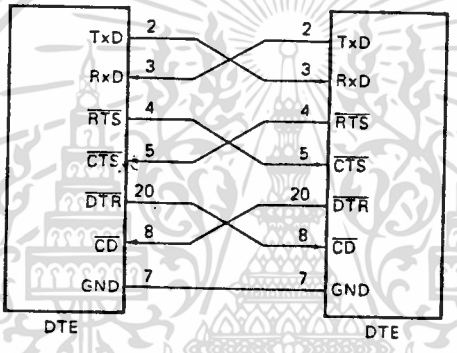
บิท 7 บิทนี้ทำหน้าที่เป็น DLAB ที่มีผลต่อการแลตช์ตัวหาร

การอีนาเบิลการอินเทอร์รัพต์ โดยกำหนดค่าลงในรีจิสเตอร์ ดังนี้





การแปลงสัญญาณจาก TTL เป็น RS 232C และ จาก RS 232C เป็น TTL



การเชื่อมต่อแบบ Null modem

PIN NUMBER	COMMON NAME	RS-232-C NAME	DESCRIPTION	SIGNAL DIRECTION ON DCE
1		AA	PROTECTIVE GROUND	—
2	TXD	BA	TRANSMITTED DATA	IN
3	RXD	BB	RECEIVED DATA	OUT
4	$\overline{RTS}$	CA	REQUEST TO SEND	IN
5	$\overline{CTS}$	CB	CLEAR TO SEND	OUT
6	$\overline{DSR}$	CC	DATA SET READY	OUT
7	GND	AB	SIGNAL GROUND (COMMON RETURN)	—
8	$\overline{CD}$	CF	RECEIVED LINE SIGNAL DETECTOR	OUT
9		—	(RESERVED FOR DATA SET TESTING)	—
10		—	(RESERVED FOR DATA SET TESTING)	—
11			UNASSIGNED	—
12		SCF	SECONDARY REC'D. LINE SIG. DETECTOR	OUT
13		SCB	SECONDARY CLEAR TO SEND	OUT
14		SBA	SECONDARY TRANSMITTED DATA	IN
15		DB	TRANSMISSION SIGNAL ELEMENT TIMING (DCE SOURCE)	OUT
16		SBB	SECONDARY RECEIVED DATA	OUT
17		DD	RECEIVER SIGNAL ELEMENT TIMING (DCE SOURCE)	OUT
18			UNASSIGNED	—
19		SCA	SECONDARY REQUEST TO SEND	IN
20	$\overline{DTR}$	CD	DATA TERMINAL READY	IN
21		CG	SIGNAL QUALITY DETECTOR	OUT
22		CE	RING INDICATOR	OUT
23		CH/CI	DATA SIGNAL RATE SELECTOR (DTE/DCE SOURCE)	IN/OUT
24		DA	TRANSMIT SIGNAL ELEMENT TIMING (DTE SOURCE)	IN
25			UNASSIGNED	—

ขื่อขาต่าง ๆของ DB 25 P ที่ใช้ในการเชื่อมต่อแบบ RS 232C

บทที่4 การอินเทอร์รัพท์และการจัดการ

เมื่อโปรแกรมผู้ใช้ร้องขออินเทอร์รัพท์ ถูกร้องขออินเทอร์รัพท์จากฮาร์ดแวร์ หรือถูกร้องขอจากฟังค์ชั่นอินเทอร์รัพท์อื่นๆ ระบบจะหยุดการทำงานของโปรแกรม อ่านค่าในตารางอินเทอร์รัพท์เวคเตอร์ และกระโดดไปยังฟังค์ชั่นที่ใช้ในการกระทำงานตามอินเทอร์รัพท์ที่ต้องการ เมื่อทำงานตามฟังค์ชั่นเสร็จสิ้นแล้ว ก็จะกลับมาทำงานที่ถูกขัดจังหวะต่อไปฟังค์ชั่นอินเทอร์รัพท์นี้มีชื่อเรียกว่า โปรแกรมบริการอินเทอร์รัพท์(Interrupt Service Routine)

โครงสร้างของตารางอินเทอร์รัพท์ จะประกอบไปด้วยค่าออฟเซ็ตและค่าเซ็กเมนต์ของตำแหน่งฟังค์ชันอินเทอร์รัพท์หมายเลขที่ 0 ถึง 255 อาจเลียนแบบโครงสร้างด้วยภาษาซีได้ดังนี้

```
struct Interrupt VectorTable{
        unsigned  offset;
        unsigned  segment;
} IVT[256];
```

พารามิเตอร์ที่ใช้เข้าออกฟังค์ชั่นเหล่านี้ จะผ่านทางรีจิสเตอร์ และในฟังค์ชั่นอินเทอร์รัพท์แต่ละตัวอาจมีฟังค์ชั่นย่อยลงไปอีก รายละเอียดของฟังค์ชั่นอินเทอร์รัพท์ทั้งหมดสามารถอ่านได้จากคู่มือโปรแกรมระบบคอมพิวเตอร์พีซี

ในภาษาซี มีฟังค์ชั่นเกี่ยวกับการกระทำอินเทอร์รัพท์หลายฟังค์ชั่น ซึ่งที่น่าสนใจมีดังนี้

- ฟังค์ชั่น disable( ) ทำหน้าที่สั่งให้ระบบไม่รับอินเทอร์รัพท์ใดๆที่อาจจะเกิดขึ้น ใช้ในกรณีที่มีการเปลี่ยนแปลงค่าอินเทอร์รัพท์ ไม่ต้องการ ให้เกิดอินเทอร์รัพท์ในขณะที่กำลังเปลี่ยนค่าตารางอินเทอร์รัพท์ หรือไม่ต้องการให้โปรแกรมที่กำลังทำงานเกิดการขัดจังหวะใดๆ มีฮาร์ดแวร์อินเทอร์รัพท์บางตัวจะไม่มีผลต่อฟังค์ชั่นนี้ การยกเลิกฟังค์ชั่น disable ( ) ทำได้โดยใช้ฟังค์ชั่น enable( )

- ฟังค์ชั่น geninterrupt( ) เป็นฟังค์ชั่นสำหรับกระทำ ซอฟแวร์อินเทอร์รัพท์ ตามหมายเลขที่กำหนด

นอกจากการเขียนโปรแกรมเพื่อขออินเทอร์รัพท์จากดอสแล้ว เราอาจเขียนโปรแกรมอินเทอร์รัพท์ด้วยตนเองก็ได้ ฟังค์ชั่นที่เขียนขึ้นจะถูกเรียกใช้เมื่อร้องขออินเทอร์รัพท์จากฟังค์ชั่นอื่นหรือเกิดจากโปรแกรมอื่นก็ได้ กรรมวิธีในการทำงานของโปรแกรมที่มีการใช้ฟังค์ชั่นอินเทอร์รัพท์จะมีขั้นตอนต่อไปนี้อยู่ในการทำงานของโปรแกรม

1. กำหนดหมายเลขอินเทอร์รัพท์ที่ต้องการให้กับฟังค์ชั่นอินเทอร์รัพท์ อาจจะใช้หมายเลขฟังค์ชั่นที่ยังไม่ได้ใช้งาน หรือใช้หมายเลขที่มีผู้ใช้งานแล้ว ในกรณีหลังจะทำให้ฟังค์ชั่นที่เขียนขึ้นจะทำงานไปด้วยกันกับฟังค์ชั่นเดิมที่มีอยู่

2. เก็บค่าหมายเลขอินเทอร์รัพท์เดิมที่ใช้ โดยใช้ฟังค์ชั่น getvect( ) ซึ่งมีรูปแบบดังนี้

```
#include <dos.h>
void  interrupt ( *getvect(int n) )( );
```

คำสงวน interrupt จะกำหนด ให้ฟังค์ชั่นที่เราจะกำหนดต่อไปนั้นเป็นฟังค์ชั่นที่ใช้เป็นโปรแกรมอินเทอร์รัพท์ ฟังค์ชั่นนี้จะให้ค่าในตารางอินเทอร์รัพท์ตามหมายเลข n ที่กำหนด

3.กำหนดค่าแอดเดรสของอินเทอร์รัพท์ฟังค์ชั่นที่สร้างขึ้น โดยใช้ฟังค์ชั่น setvect( ) ซึ่งจะทำหน้าที่กำหนดให้อินเทอร์รัพท์หมายเลข n เปลี่ยนมาชี้ยังฟังค์ชั่นที่เราสร้างขึ้นแทน ISR ตัวเดิมที่เคยชี้อยู่

4. เมื่อจบโปรแกรม จะต้องคืนค่าอินเทอร์รัพท์ฟังค์ชั่นเดิมให้แก่ตารางอินเทอร์รัพท์ มิฉะนั้น เมื่อโปรแกรมจบการทำงานแล้ว ระบบจะเสียหายได้

ในการขออินเทอร์รัพท์ของโปรแกรมที่เขียนด้วยภาษาซีด้วยฟังค์ชั่น geninterrupt( ) ค่าในรีจิสเตอร์จะถูกเก็บลงในสแต็ก การเขียนฟังค์ชั่นอินเทอร์รัพท์ที่มีการผ่านค่าทางรีจิสเตอร์จึงอาจใช้สแต็กเป็นทางผ่านข้อมูลเข้าออกได้โดยการกำหนดพารามิเตอร์หลังชื่อฟังค์ชั่นอินเทอร์รัพท์ดังรูปทั่วไป และเมื่อจบการทำงานของฟังค์ชั่นอินเทอร์รัพท์ ค่าในตัวแปรที่กำหนดจะถูกเก็บลงสแต็ก โปรแกรมที่ถูกขัดจังหวะ(ซึ่งถูกเขียนด้วยภาษาซี) จะคืนค่าในสแต็กเพื่อทำงานต่อไป ดังนั้น เราสามารถส่งค่าคืนได้โดยการใส่ค่ากลับลงในตัวแปรที่กำหนดไว้ในรูปพารามิเตอร์เหล่านี้

หากเราเขียนฟังค์ชั่นอินเทอร์รัพท์สำหรับใช้กับโปรแกรมที่เขียนด้วยภาษาอื่นหรือไม่ต้องการมีการส่งค่ากลับ (อาจใช้รีจิสเตอร์สำหรับการส่งค่าไปได้) เราก็อาจไม่กำหนดพารามิเตอร์ก็ได้ ฟังค์ชั่นอินเทอร์รัพท์สำหรับกรณีทั่วไปจึงมีลักษณะดังรูป

```
void interrupt ชื่อฟังค์ชั่น ( void )
{           ภายในมีลักษณะเช่นเดียวกับฟังค์ชั่นทั่วๆไป
        (*ฟังค์ชั่นอินเทอร์รัพท์เดิม)( );  /*ในกรณีที่ใช้หมายเลขร่วมกับฟังค์ชั่นอินเทอร์รัพท์ที่มีอยู่
          แล้ว*/
}
```

4.1 ซอฟแวร์อินเทอร์รัพท์(Software Interrupts)

ลักษณะของ ซอฟแวร์อินเทอร์รัพท์(Software Interrupts) มีดังนี้

1. จะเกิดขึ้นเมื่อโปรแกรมมีการเรียกใช้คำสั่ง อินเทอร์รัพท์นัมเบอร์(Interrupt number)ต่างๆ

2. มีการผ่านค่าต่างๆลงใน รีจิสเตอร์(register)ก่อนมีการเรียกใช้ อินเทอร์รัพท์(Interrupt)เช่น มีการผ่านค่าฟังก์ชั่นของ อิเทอร์รัพท์ 21 ลงใน รีจิสเตอร์เอเฮส(register AH)

3. หลังจากทำคำสั่ง อินเทอร์รัพท์นั้นแล้ว ค่าต่างๆใน รีจิสเตอร์ สามารถเปลี่ยนแปลงได้

ลักษณะของซอฟแวร์อินเทอร์รัพท์ ที่เห็นได้ชัดๆได้แก่ อิเทอร์รัพท์ของ DOS ซึ่งเป็นการติดต่อระหว่าง ซอฟแวร์อินเทอร์รัพท์ กับ ระบบโอเปอร์เรทติง(operating system) โดยการเรียกใช้ ฟังค์ชั่นค

อล(funcftion call)ต่างๆ ในการเรียกใช้ซอฟแวร์อินเทอร์รัพท์ ต่างๆนั้น เราจะต้องทราบค่าแอดเดรส (address) ของ

อินเทอร์รัพท์เบอร์ที่เราต้องการจะเรียกก่อน เพื่อจะได้ทราบตำแหน่งของมัน โดยค่า แอดเดรส เหล่านี้ จะดูได้จากตารางอินเทอร์รัพท์เวคเตอร์เทเบิล(interrupt vector table)ซึ่งอยู่ที่ตำแหน่ง แอดเดรส ที่ 0000:0000 ถึง 0000:0400 โดยตารางนี้ จะระบุค่าฟาร์ พอยท์เตอร์(far pointer)ของโปรแกรมบริการอินเทอร์รัพท์ (interrupt service routine)

4.2 ฮาร์ดแวร์อินเทอร์รัพท์(Hardware Interrupts)

ฮาร์ดแวร์อินเทอร์รัพท์ จะเกิดจากอุปกรณ์พวกไอโอดีไวซ์(I/O device)ต่างๆ เช่น คีย์บอร์ด(keyboard)จะกำเนิดสัญญาณ อิเทอร์รัพท์ ทุกครั้งที่มีการกด ลักษณะของฮาร์ดแวร์อินเทอร์รัพท์ก็จะคล้ายคลึงกับซอฟ แวร์อินเทอร์รัพท์ คือ จะมีการเรียกอินเทอร์รัพท์ เซอร์วิส รูทีนของตนทุกครั้งที่มีการกำเนิดสัญญาณ อินเทอร์รัพท์ และทุกครั้งที่มีการเรียก อินเทอร์รัพท์เซอร์วิสรูทีน ก็จะต้องเรียกผ่านตาราง อินเทอร์รัพท์ เวคเตอร์เทเบิล เพื่อหา พอยท์เตอร์(pointer)ไปชี้ที่ อินเทอร์รัพท์เซอร์วิสรูทีน สำหรับ แอดเดรส ของตาราง ก็อยู่ที่ตำแหน่ง 0000:0000 ถึง 0000:0400 เช่นเดียวกับตารางของ ซอฟแวร์อินเทอร์รัพท์

ส่วนใหญ่ขนาดของโปรแกรมของ ฮาร์ดแวร์อินเทอร์รัพท์ มักจะสั้น เนื่องจากอุปกรณ์ต่างๆที่ร้องขอการ อิเทอร์รัพท์ มักจะรอไม่ได้ เช่น กรณีของ คอมพอร์ท (COM port) ทันทีที่มีข้อมูลเข้ามาก็จะมีการส่ง สัญญาณอินเทอร์รัพท์ ไปบอกกับซีพียู(CPU)ให้มาอ่านข้อมูลไป ไม่เช่นนั้น ข้อมูลตัวถัดไปจะมาทับข้อมูล เดิม ซึ่งจะทำให้เกิดการผิดพลาดขึ้น นอกจากนั้นแล้วอินเทอร์รัพท์รูทีนยังห้ามใช้ คำสั่งของ DOS อีกด้วย เพราะ DOS นั้นถูกออกแบบมาใช้กับโปรแกรมเท่านั้น แต่ถ้ามีการเรียกใช้ก็อาจทำให้ระบบปฏิบัติการ เกิด ความผิดพลาดขึ้นได้

4.3 การเขียนโปรแกรมเรซิเดนท์เบื้องต้น

โดยปกติ เมื่อโปรแกรมทำงานเสร็จสิ้นแล้ว ก็จะโอนการทำงานให้กับระบบปฏิบัติการ และ โปรแกรมก็จะถูกถอดออกจากหน่วยความจำ แต่ยังมีโปรแกรมอีกประเภทหนึ่ง เมื่อโปรแกรมโอนการทำงาน ให้กับระบบปฏิบัติการแล้ว ตัวโปรแกรมจะยังคงอยู่ในหน่วยความจำ และจะทำงานเมื่อได้รับการร้องขอ อินเทอร์รัพท์ โปรแกรมประเภทนี้เรียกว่า โปรแกรมเรซิเดนท์ (Resident program)

การวางตัวของโปรแกรมเรซิเดนท์จะมีลักษณะในทำนองเดียวกันกับไบออสและเคอร์เนิลของระบบ นั่นคือจะใช้การร้องขออินเทอร์รัพท์ของโปรแกรมเรซิเดนท์นั้นๆ ในการเรียกใช้ฟังค์ชั่นภายในโปรแกรม โครง สร้างของโปรแกรมเรซิเดนท์จะประกอบไปด้วย 3 ส่วนคือ

1.ส่วนติดตั้งอินเทอร์รัพท์ และส่วนติดตั้งโปรแกรม

2. ส่วนอินเทอร์รัพท์ฟังค์ชั่น สำหรับใช้ในการทำงาน

3. ส่วนการถอดอินเทอร์รัพท์ และส่วนถอดโปรแกรมออกจากหน่วยความจำ

การเรียกฟังค์ชั่นในโปรแกรมเรซิเดนท์จากโปรแกรมอื่น อาจมีวิธีเรียกได้ 2 วิธีคือ

1. ร้องขออินเทอร์รัพท์ที่โปรแกรมเรซิเดนท์นั้นใช้อยู่ ในวิธีนี้เป็นวิธีมาตรฐานที่ใช้กันโดยทั่วไป

2.ใช้การร้องขออินเทอร์รัพท์ของโปรแกรมเรซิเดนท์ เพื่ออ่านค่าตำแหน่งฟังค์ชั่นที่ต้องการ และนำมากำหนดให้กับไดนามิกฟังค์ชั่น เมื่อต้องการเรียกใช้ฟังค์ชั่น ก็เพียงแต่เรียกใช้ไดนามิกฟังค์ชั่น ในวิธีนี้จะทำให้การเรียกใช้ฟังค์ชั่นเป็นไปได้ด้วยความสะดวกและรวดเร็วมากขึ้น

หมายเหตุ

1. ก่อนรันโปรแกรมจะต้องออกจากคอมไพเลอร์เสียก่อน

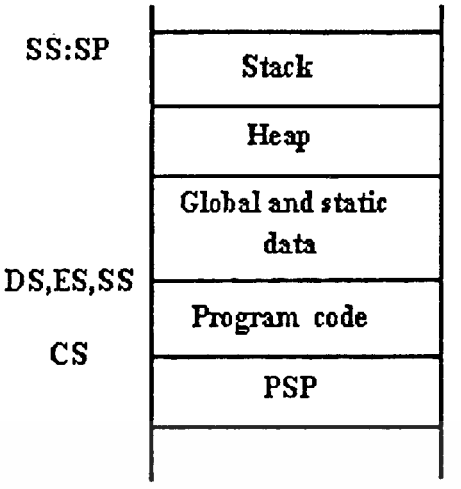
2. โดยทั่วไปมักจะเขียนโปรแกรมเรซิเดนท์ด้วยภาษาแอสเซมบลี เนื่องจากโปรแกรมที่ได้จะมีขนาดเล็ก และสามารถควบคุมผลข้างเคียงที่เกิดขึ้นได้มากกว่า(การเขียนโปรแกรมด้วยภาษาซี บางครั้งจะเกิดการเซ็ตค่าในรีจิสเตอร์และการทำงานบางอย่างที่ไม่จำเป็น ซึ่งเป็นข้อเสียของการเขียนโปรแกรมด้วยภาษาระดับสูงโดยทั่วไป)

3. หากโปรแกรมเรซิเดนท์ที่เกิดขึ้น ต้องถูกร้องขออินเทอร์รัพท์ทุกๆช่วงเวลาหนึ่ง จะต้องระวังไม่ให้เวลาในการทำงานในฟังค์ชั่นอินเทอร์รัพท์มีค่ามากกว่าช่วงเวลาในการร้องขออินเทอร์รัพท์ เพราะจะเกิดอินเทอร์รัพท์เดิมซ้ำ ในขณะที่ยังทำงานในส่วนฟังค์ชั่นอินเทอร์รัพท์ไม่เสร็จสิ้น ลักษณะเช่นนี้เรียกว่า รีเอนทรานซ์(re-entrance) จะทำให้เกิดการเรียกในลักษณะนี้ซ้อนกันไป จนกระทั่งระบบหยุดทำงานในที่สุด

4. สังเกตการรับส่งค่าในกรณีที่ฟังค์ชั่นอินเทอร์รัพท์มีพารามิเตอร์ต่อท้าย เนื่องจากในการร้องขออินเทอร์รัพท์ของภาษาซี รีจิสเตอร์ทั้งหมดจะถูกเก็บลงสแต็กก่อนเรียกใช้ฟังค์ชั่นอินเทอร์รัพท์ (เมื่อการร้องขออินเทอร์รัพท์เกิดจากตัวโปรแกรมเองโดยใช้ฟังค์ชั่น geninterrupt( )) ดังนั้น เราจึงใช้วิธีกำหนดค่าพารามิเตอร์หลังฟังค์ชั่นอินเทอร์รัพท์โดยการเรียกตัวแปรที่สอดคล้อง ก็จะทำให้เราสามารถอ่านค่าจากรีจิสเตอร์(ที่ถูกเก็บค่าลงในสแต็ก) หรืออาจจะเรียกจากรีจิสเตอร์โดยตรง และสามารถ ส่งค่ากลับทางพารามิเตอร์ เนื่องจากตำแหน่งหน่วยความจำของพารามิเตอร์จะอยู่ในสแต็ก หลังจากการร้องขออินเทอร์รัพท์ของฟังค์ชั่น geninterrupt( ) ภาษาซีจะเพิ่มคำสั่งคืนค่ารีจิสเตอร์จากสแต็กทำให้ค่าที่เราส่งผ่านทางพารามิเตอร์กลับสู่รีจิสเตอร์ได้

5. ขนาดของโปรแกรมอินเทอร์รัพท์จะต้องไม่เกิน 64 กิโลไบท์

6. ในการคืนหน่วยความจำ จะมีการคืนส่วนที่เป็นสภาวะแวดล้อมของโปรแกรม(environment) และตามด้วยตัวโปรแกรมและ PSP รายละเอียดของ PSP สามารถ หาอ่านได้จากหนังสือการโปรแกรมระบบ ฟังค์ชั่น keep( ) เป็นฟังค์ชั่นสำหรับออกจากโปรแกรมและกระทำการเรซิเดนท์ จะส่งค่า.0 ออกไปยังดอส และกำหนดขนาดของโปรแกรมที่ต้องการทำเรซิเดนท์ได้จากรีจิสเตอร์ SS ซึ่งจะชี้ที่ตำแหน่งหน่วยความจำสูงสุดของพื้นที่โปรแกรม และ _psp ซึ่งจะชี้อยู่ที่ PSP อันเป็นหน่วยความจำต่ำสุดของพื้นที่โปรแกรมดังรูป



ใช้เฉพาะเทอร์โบซีและบอร์แลนด์ซี

4.4 ลักษณะโปรแกรมฝังตัว(Resident Program)ที่ใช้ในโครงงาน

ในโครงงานเราได้ใช้โปรแกรมเรสซิเดนซ์รันไว้ที่เทอร์มินัลปลายทางทุกตัว เพื่อทำการแจ้งให้เทอร์มินัลทราบเวลาที่ต้นทางต้องการที่จะติดต่อด้วย ขณะนั้นปลายทางอาจกำลังทำงานอื่นอยู่ก็ได้ โดยเทอร์มินัลต้นทางจะทำการร้องขอการติดต่อไปที่เซนเตอร์ แล้วเซนเตอร์ก็จะทำการส่งสัญญาณไปบอกเทอร์มินัลปลายทางอีกทีหนึ่ง สัญญาณที่ส่งไปบอกนี้จะเป็นตัวไปปลุกโปรแกรมที่ฝังตัวนี้ให้ทำงาน โปรแกรมฝังตัวนี้จะแจ้งการติดต่อให้ผู้ใช้(User)ทราบ โดยแสดงขึ้นทางหน้าจอของผู้ใช้ สำหรับการทำงานโดยละเอียดของโปรแกรมก็มีดังนี้

1. ต้นทางส่งสัญญาณร้องของการติดต่อไปที่เซนเตอร์

2. เมื่อเซนเตอร์รับทราบว่าต้นทางต้องการที่จะติดต่อกับปลายทางเครื่องใด ก็จะทำการส่งสัญญาณไปบอกที่ปลายทางเครื่องนั้น

3. สัญญาณที่เซนเตอร์ส่งเข้ามาทางพอร์ตอนุกรม(serial port) ก็จะไปทำการปลุกโปรแกรมที่ทำการฝังตัวไว้ที่เทอร์มินัลเครื่องปลายทาง

4. โปรแกรมฝังตัวก็จะทำการแสดงผลขึ้นมาทางหน้าจอให้ผู้ใช้ทราบว่า มีคนต้องการที่จะติดต่อด้วย และต้องที่ติดต่อแบบ ส่งชุดข้อมูล(Transfer File) หรือแบบติดต่อกันทั้งสองทาง (Chat mode)พร้อมทั้งถามผู้ใช้อีกว่าต้องการที่จะติดต่อกลับหรือไม่ ถ้าต้องการก็ให้กดอักษร'y' แล้วออกจากโปรแกรมเดิมที่ทำงานอยู่ ไปรัน(run)โปรแกรมรับข้อมูล โดยขณะนั้นก็จะทำการส่งสัญญาณตอบรับไปบอกที่เซนเตอร์ด้วย แต่ถ้าปลายทางไม่ต้องการที่จะติดต่อด้วยก็กดอักษร 'n' โปรแกรมก็จะคืนการทำงานเดิมให้ และส่งสัญญาณตอบปฏิเสธไปบอกที่เซนเตอร์เช่นกัน

5. หลังจากที่ตอบคำถามเรียบร้อยแล้ว โปรแกรมก็จะทำการคืนหน้าจอให้ พร้อมจบโปรแกรม แต่ก็ยังคงฝังตัวต่อไป และก็จะทำงานอีกเมื่อมีการส่งข้อมูลเข้ามาทางพอร์ตอนุกรม(serial port)

วิธีการเขียนโปรแกรมฝังตัว(Resident Program) ที่ใช้ในโครงงาน

สำหรับวิธีการเขียนโปรแกรมฝังตัวที่ใช้ในโครงงานนั้น ก็มีอยู่ 2 วิธี วิธีแรกนั้นจะใช้ INT 1C ซึ่งเป็น Interrupt Timer ของระบบ โดย Interrupt Timer นี้จะส่งสัญญาณไปอินเทอร์รัพท์ซีพียูทุก 18.2 ครั้งใน 1 วินาที โดยเราจะทำการใส่โปรแกรม Interrupt Service Routine ไว้ใน INC 1C แล้วให้ซีพียูไปทำการอ่านพอร์ตทุกครั้งที่มีการอินเทอร์รัพท์เกิดขึ้น แล้วดูว่าข้อมูลที่เข้ามาใช่ข้อมูลที่ต้องการหรือไม่ ถ้าใช่ก็จัดการแสดงรายการ(pop up) ขึ้นมาทางหน้าจอ แต่ตัวอินเทอร์รัพท์ 1C นี้ก็จะมีปัญหาคือ โปรแกรมที่เราทำการใส่ลงไปใน INT 1C นั้น จะต้องสั้นมาก มิฉะนั้นจะเกิดปัญหาการอินเทอร์รัพท์ซ้อนขึ้นมาได้ เช่น ในขณะที่ซีพียูกำลังทำงานในส่วนของ pop up ยังไม่เสร็จ INT 1C ก็ไปอินเทอร์รัพท์ซีพียูอีกรอบหนึ่ง และซีพียูก็จะต้อง pop up อีก เนื่องจากข้อมูลที่เข้ามานั้นยังคงค้างอยู่ ทำให้เกิดการ pop up ซ้อนขึ้นได้ ส่วนวิธีที่ 2 นั้น เป็นวิธีที่เราใช้ในโครงงาน วิธีนี้เราจะใช้ INT 0B ที่ต่ออยู่กับ com2 ก็จะไปปลุก INT 0B ให้ทำงาน แสดงรายการ pop up ขึ้นมา สำหรับขั้นตอนการเขียนก็มีดังต่อไปนี้

1. ศึกษาหาข้อมูลการส่งสัญญาณอินเทอร์รัพท์ของพอร์ตอนุกรม(serial port) ว่า ส่งสัญญาณไปอินเทอร์รัพท์ 8259 ได้อย่างไร และ com2 ที่เราใช้รับข้อมูลจากเซนเตอร์ต่อกับ IRQ เบอร์อะไรของ 8259 ซึ่งจากการค้นคว้าหาข้อมูลก็ได้คำตอบว่า การเซ็ตให้ 8250 ส่งสัญญาณไปอินเทอร์รัพท์ได้นั้น เราจะต้องไป enable การส่งสัญญาณไปอินเทอร์รัพท์ของ 8250 ด้วย การเซ็ตสัญญาณ OUT 2 ใน Modem Control Register (2fc) ให้มีค่าเป็น 1 จากนั้นก็ไป enable การอินเทอร์รัพท์ที่ Interrupt Identification register (2f9)

2. เมื่อทำการเซ็ต 8250 ให้สามารถส่งสัญญาณไปอินเทอร์รัพท์ได้แล้ว ก็ต้องทำการเซ็ต 8259 ให้สามารถรับส่งสัญญาณอินเทอร์รัพท์จาก 8250 ด้วย จากการศึกษาพบว่า พอร์ต com2 นั้นต่ออยู่กับ IRQ 3 ของ 8259 เราจะต้องไปทำการ enable IRQ 3 ของ 8259 ให้ทำงาน โดยการ mark บิทที่ 3 ของ 8259 ให้เป็น 0 แล้วส่งออกไปที่พอร์ต 0x21

3.หลังจากทำการ เซตการทำงานให้ 8250 และ 8259 แล้ว ก็ทำการเขียนโปรแกรมอินเทอร์รัพท์ต่อไป ซึ่งโปรแกรมอินเทอร์รัพท์ที่เราทำการเขียนขึ้นมานี้ จะนำไปไว้ที่ อินเทอร์รัพท์ 0B จากการค้นคว้าหาข้อมูลเราทราบว่า IRQ 3 นั้นต่ออยู่กับอินเทอร์รัพท์ 0B ดังนั้นเราจึงนำโปรแกรมที่เราทำการเขียนขึ้นมาใหม่นี้ไปแทนที่อินเทอร์รัพท์ 0B โดยการย้ายโปรแกรมบริการอินเทอร์รัพท์เดิมออก แล้วเอาที่เราเขียนขึ้นใหม่ใส่เข้าไปแทน โดยใช้คำสั่ง getvect และ setvectในภาษาซี

4.สำหรับการเขียนโปรแกรมบริการอินเทอร์รัพท์ในการฝังตัวนั้น เราจะต้องหลีกเลี่ยงการให้คำสั่งในดอสเช่น printf เป็นต้น เนื่องจากดอสถูกออกแบบให้ทำงานได้งานเดียวเท่านั้น(single task)การที่

เราจะไปใช้คำสั่งของดอสในโปรแกรมของเรานั้น จะทำให้เกิดปัญหาเรื่อง reentrance ซึ่งมีผลทำให้การทำงานของระบบผิดพลาดได้ ดังที่เราได้กล่าวมาแล้วในทฤษฎีข้างต้น ดังนั้นคำสั่งการทำงานที่ใช้ในอินเทอร์รัพท์เซอร์วิสรูทีน เราจะต้องทำการเขียนขึ้นมาใหม่ โดยใช้ฟังค์ชั่นของ BIOS และในการเขียนหน้าจอนั้น เราก็ทำการเขียนลงในหน่วยความจำ ของ video ram เลย ในส่วนตรงนี้นับว่าเป็นการยุ่งยากพอสมควรเทียบกับการเขียนโปรแกรมทั่วไป

5.สำหรับการทำงานในโปรแกรมบริการอินเทอร์รัพท์ นั้น เมื่อมีการอินเทอร์รัพท์การรับข้อมูลเข้ามาก็จะทำการเปรียบเทียบดูว่าข้อมูลที่เข้ามานั้น เป็นสัญญาณที่ส่งมาจากเซนเตอร์หรือไม่ ถ้าใช่ก็จะทำการแสดงรายการpop up ขึ้นมาโดยจะต้องทำการเก็บหน้าจอการทำงานเดิมไว้แล้วจึงค่อยทำการแสดงผลทางหน้าจอ สิ่งที่แสดงหน้าจอนั้นจะแจ้งให้ผู้ใช้ทราบว่ามีคนที่ต้องการติดต่อด้วยและติดต่อด้วยแบบใด ส่วนผู้ใช้ต้องการที่จะติดต่อกลับไปหรือไม่นั้น ก็จะทำการตอบกลับมาที่โปรแกรม โปรแกรมจะทำการส่งข้อมูลไปบอกทางเซนเตอร์ให้ทราบอีกทีหนึ่ง จากนั้นก็ทำการคืนหน้าจอเดิมให้กับผู้ใช้ปลายทาง ก็จะเป็นการเสร็จสิ้นการทำงานของโปรแกรมบริการอินเทอร์รัพท์

6. ให้เซ็ต model ของคอมไพเลอร์ เป็นแบบ Tiny เนื่องจาก model แบบนี้ จะไปทำการเซ็ตให้ Data Segment และ Code Segment อยู่ที่ตำแหน่งเดียวกัน

7. ควรจะไล่ดู soure code ของโปรแกรมประกอบไปด้วย จะช่วยให้เข้าใจโปรแกรมได้กระจ่างยิ่งขึ้น

บทที่ 5
การทำงานและอัลกอริทึมของโปรแกรม

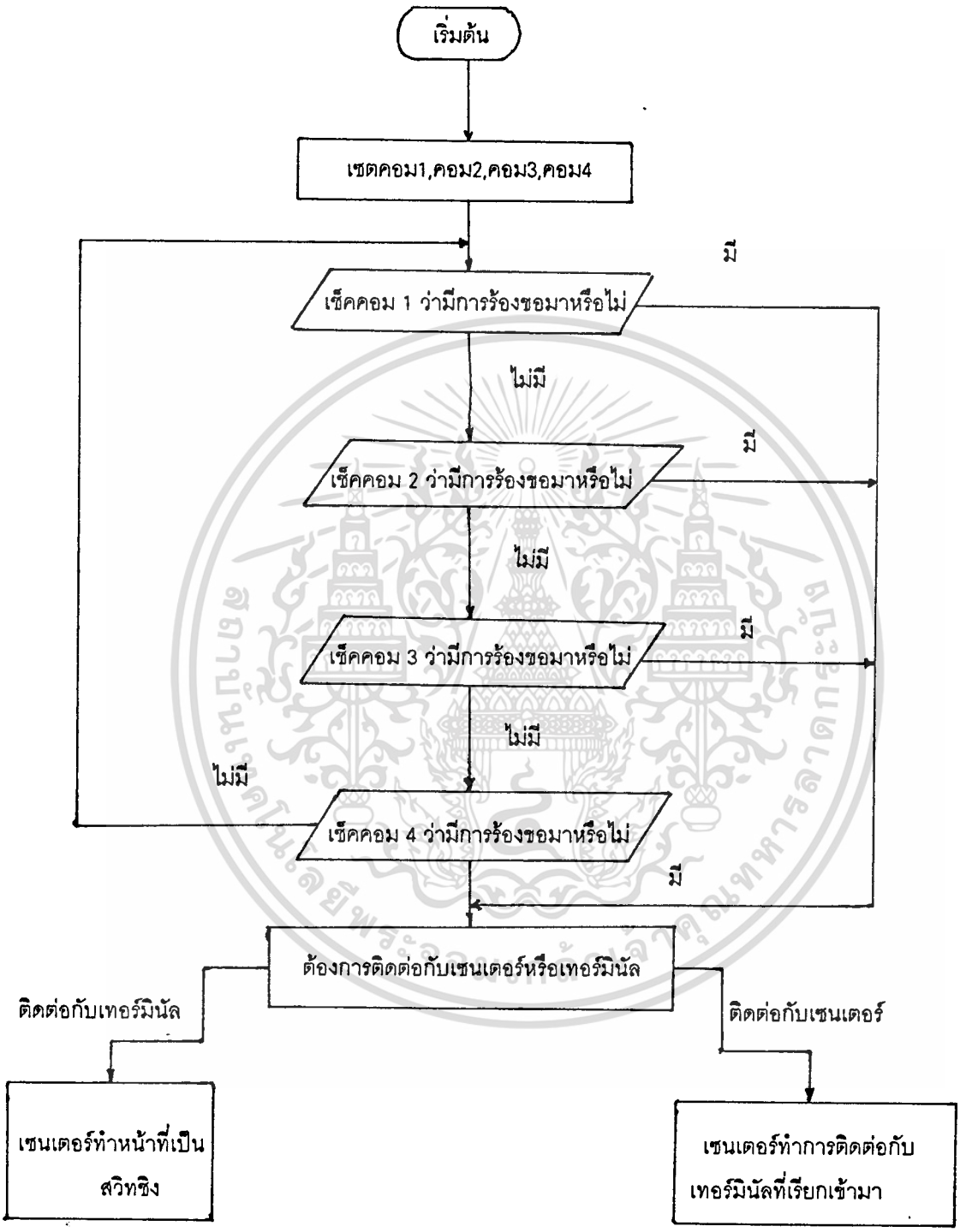
เทอมที่ผ่านมา เราได้เขียนโปรแกรมควบคุมการส่งข้อมูลแบบฟูลดูเพลกซ์(Ful duplex) โดยมีเซนเตอร์ต่อกับเทอร์มินัล 2 เครื่อง และ ดาต้าลิงค์โปรโตคอลที่ใช้ในการทำงานของเซนเตอร์เป็นแบบโพลลิง นอกจากนั้นก็ได้ศึกษาลู่ทางในการเพิ่มเทอร์มินัลที่เชื่อมต่อกับเซนเตอร์ให้มากขึ้น โดยได้ทำคาร์ด (card) ที่สามารถใช้เป็น คอม 3 และคอม 4 ต่อเนื่องมาจนเสร็จสมบูรณ์สามารถใช้งานได้แล้วในเทอมนี้

ดังนั้นในส่วน ซอฟแวร์ ในเทอมนี้เราต้องเพิ่มส่วนควบคุมการทำงานของคอม 3 ,คอม 4 ด้วย ในส่วน ดาต้าลิงค์โปรโตคอล เรายังคงใช้แบบโพลลิง เหมือนเดิม โดยเพิ่มความสามารถพิเศษให้เซนเตอร์ ด้วยการใช้เทคนิคการเรสสิเดนท์(Resident) เข้าไปอีกด้วย ซึ่งรายละเอียดของการเรสสิเดนท์จะกล่าวถึงในภายหลัง ส่วนแอพพลิเคชั่น(Application) ในเทอมนี้ก็ได้เพิ่มการส่งข้อมูลที่เป็นไฟล์เข้าไป ซึ่งระบบการทำงานในโครงข่ายอธิบายได้ดังนี้

การทำงานของเซนเตอร์ เซนเตอร์จะทำการวนลูปเช็คว่ามีการติดต่อมาจากเทอร์มินัลทั้ง 4 หรือไม่ โดยเช็คไปทีละเทอร์มินัล จากเทอร์มินัลที่ 1 ไปจนถึงเทอร์มินัลสุดท้าย หลังจากนั้น ก็ต้องตรวจดูว่าเทอร์มินัลที่ร้องขอมานั้นต้องการจะติดต่อกับเซนเตอร์เองหรือติดต่อกับเทอร์มินัลอื่น แล้วเซนเตอร์จึงปฏิบัติงานตามที่ร้องขอ ในกรณีที่ต้องการติดต่อกับเทอร์มินัลอื่นซึ่งไม่ใช่เซนเตอร์ เซนเตอร์จะทำการไปเรียกเทอร์มินัลปลายทางให้รับรู้

ในส่วนเทอร์มินัลนั้นเมื่อต้องการจะติดต่อกับเทอร์มินัลใดจะต้องร้องขอไปที่ เซนเตอร์ก่อน แล้วบอกให้เซนเตอร์ทราบว่า ต้องการติดต่อกับเทอร์มินัลใด จากนั้นก็เลือกการติดต่อว่าจะเป็นแบบใด ซึ่งยูสเซอร์สามารถเลือกได้ 2 แบบ คือ แบบ ติดต่อกันทันทีพร้อมกัน2 ทาง และแบบส่งข้อมูลเป็นไฟล์

5.1 การทำงานของเซนเตอร์

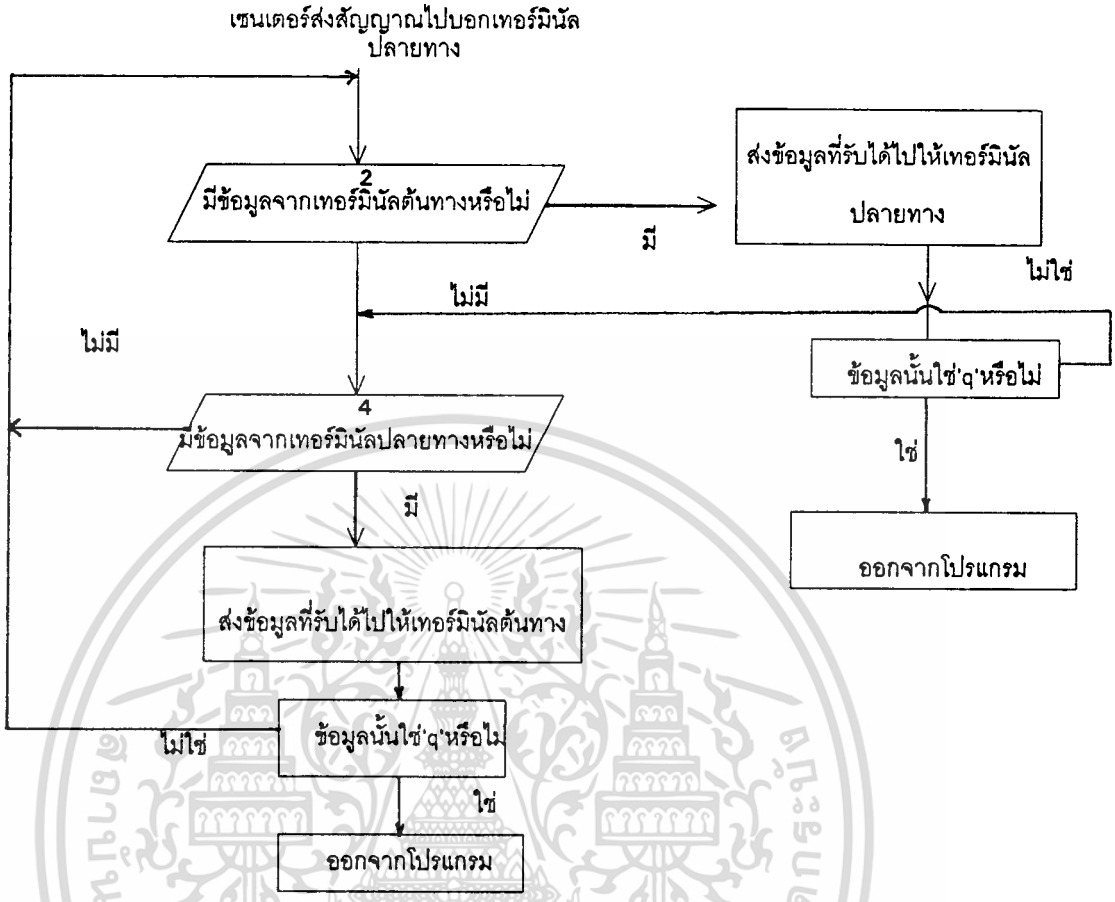


TopChart

Sunday, March 20, 1994

5:30 PM

5.2 การสวิทของเซนเตอร์

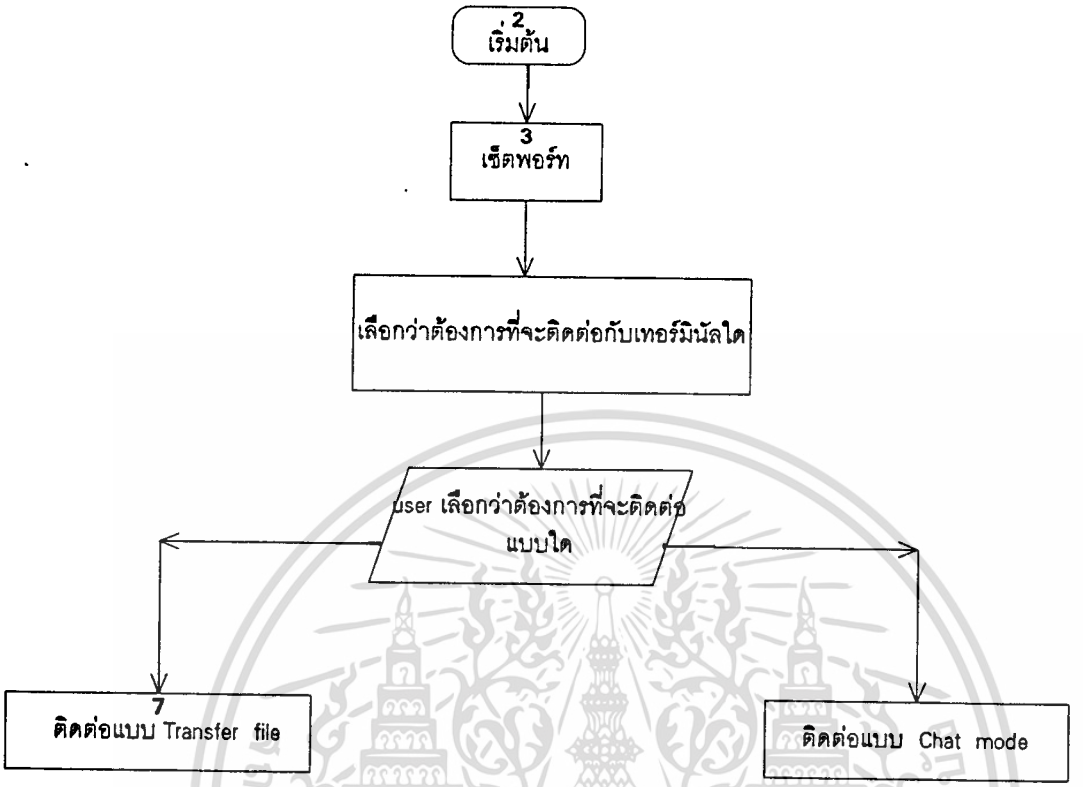


TopChart

Sunday, March 20, 1994

5:11 PM

5.3 การทำงานที่เทอร์มินัล

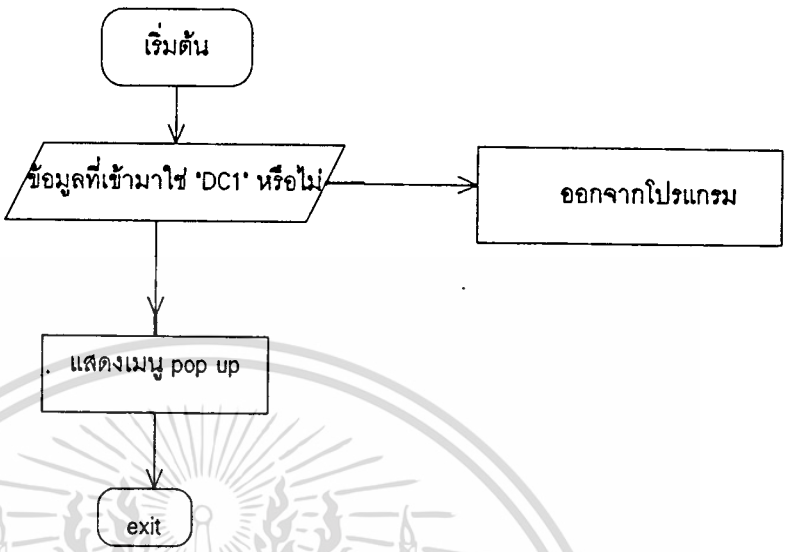


TopChart

Sunday, March 20, 1994

6:23 PM

5.4 อินเทอร์รัพท์ เซอร์วิท รูทีน ที่ใส่ใน INT 0B

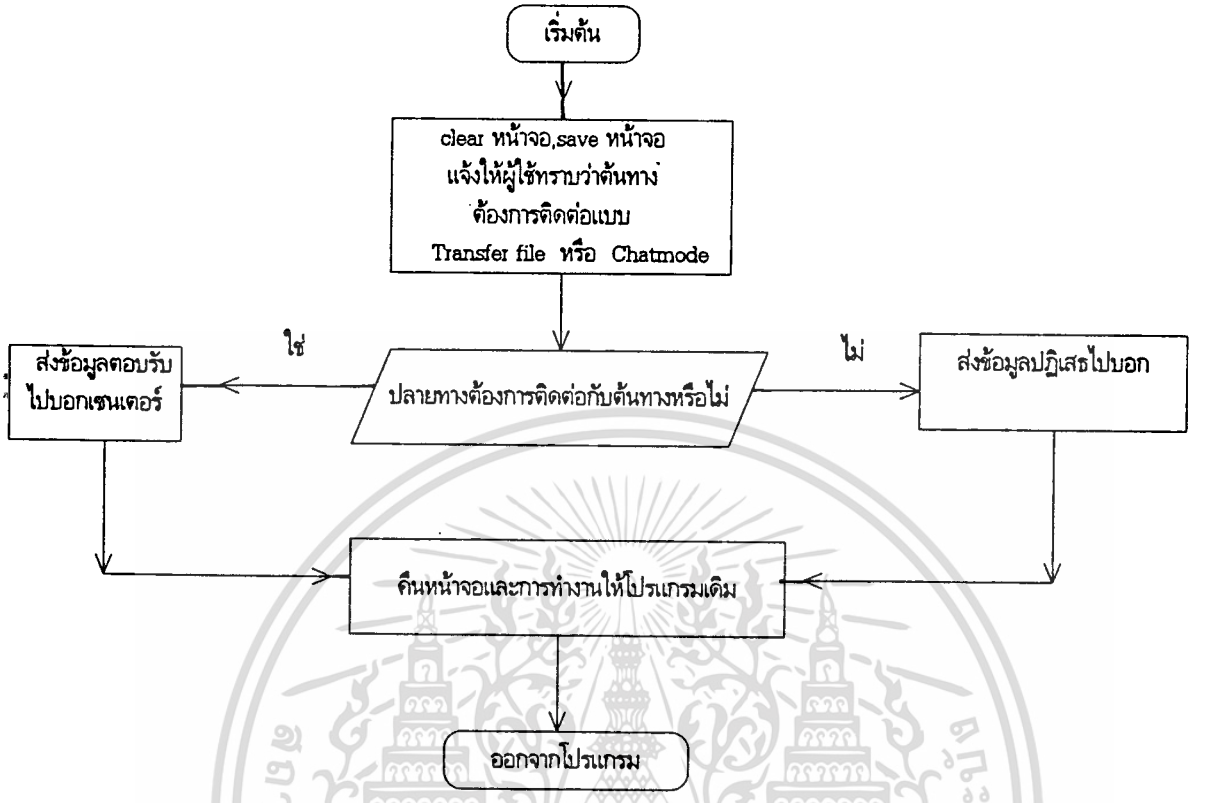


TopChart

Sunday, March 20, 1994

6:32 PM

5.5 pop up เมนู



บทที่ 6
ผลการทดลองและสรุป

ในการทดลองเราใช้เครื่องคอมพิวเตอร์ทั้งหมด 4 เครื่องโดยจะมีอยู่ 1 เครื่องทำหน้าที่เป็นเซนเตอร์ ซึ่งเครื่องนี้จะมีการ์ดที่ทำขึ้นมาเสียบอยู่ที่เอ็กซ์แพนชั่นสล็อต และที่เหลือจะทำตัวเป็นเทอร์มินัล 1,2 และ 3 ตามลำดับ การติดต่อจะผ่านทางพอร์ตอนุกรม

การทำงานเริ่มแรกจะต้องรันโปรแกรมควบคุมที่เซนเตอร์ก่อน ซึ่งเป็นการเซตค่าเริ่มต้นของการสื่อสาร มี ความเร็ว 1200 บอด ความยาวข้อมูล 8 บิท 1 สต็อปบิท ไม่มีพาริตี้บิท

สำหรับทางเทอร์มินัลต้นทางนั้นจะเริ่มต้นจากการเลือกเทอร์มินัลปลายทางและจะส่งแอดเดรสของเทอร์มินัลนั้นเป็นการร้องขอการติดต่อส่งไปยังเซนเตอร์ หลังจากนั้นจะเลือกวิธีการติดต่อแล้วส่งไปยังเซนเตอร์ การทำงานหลังจากนี้จะเป็นช่วงของการรอการตอบกลับ ซึ่งถ้าได้รับ 'ACK' ก็จะทำงานในส่วนของแอปปลิเคชั่นที่ได้เลือกไว้ ส่วนกรณีที่ได้รับ'NAK' ก็จะออกจากการทำงาน

เทอร์มินัลที่เหลือจะรันโปรแกรมฝังตัวเพื่อรอรับการติดต่อจากเซนเตอร์ในระหว่างนี้ทางเทอร์มินัลสามารถทำงานอื่นใดอยู่ก็ได้ เมื่อได้รับการติดค่อ ทางหน้าจอจะปรากฏข้อความเพื่อแจ้งให้ยูสเซอร์ทราบ ในโปรแกรมจะถามยูสเซอร์ว่าต้องการที่จะติดต่อหรือไม่ หากไม่ต้องการที่จะติดต่อจะส่ง'NAK' ไปยังเซนเตอร์แล้วทำงานขณะนั้นต่อไป แต่ถ้าต้องการที่จะติดต่อจะส่ง'ACK' ไปยังเซนเตอร์ ยูสเซอร์จะต้องรันโปรแกรมสำหรับการติดต่อขึ้นมาตามที่เซนเตอร์บอก

ในส่วนของฮาร์ดแวร์ได้ทำการต่อวงจรตามรูปในหน้าหลังสุดของรายงานนี้ โดยการต่อวงจรแบบพันวายแลบ ไอซีหลักๆที่ใช้คือ 8250 และส่วนกำเนิดสัญญาณคล็อกในการควบคุมการรับส่งที่ความถี่ 1.8432 เมกะเฮิร์ต จากผลการทดลองการ์ดนี้สามารถที่จะใช้งานตามแบบของพอร์ตอนุกรมสื่อสารได้อย่างครบถ้วน

แนวทางการพัฒนา

ขยายการทำงานในส่วนของแอปพลิเคชั่น เช่น เพิ่มให้มีการส่ง Email , Memory mapping และทำโปรแกรมแอปพลิเคชั่นเดิมเป็นแบบฝังตัว เป็นต้น ในกรณีของฮาร์ดแวร์ที่เพิ่มเข้าไปในเซนเตอร์สามารถที่จะเชื่อมต่อกับเทอร์มินัลได้มากกว่า 4 เครื่อง โปรโตคอลที่ใช้ในการติดต่อสื่อสารต้องเปลี่ยนแปลงไป ต้องมีการกำหนดแอดเดรสใหม่เพิ่มเข้าไป เพื่อที่เซนเตอร์สามารถดีเทคได้ว่า ข้อมูลที่ส่งมานั้นมาจากปลายทางไหน สำหรับในส่วนที่ได้ทำมาทั้งหมดนี้ จะเป็นพื้นฐานที่จะนำไปพัฒนาต่อได้เป็นอย่างดี

บทแทรก

สตาร์แลนที่ใช้รูปแบบ OSI โมเดล

STARLAN AT&T ได้ทำเป็นมาตรฐานอิงกับ OSI โมเดล ซึ่งไม่ทำให้เกิดปัญหาเมื่อใช้กับอุปกรณ์ตัวอื่นๆ สตาร์แลน(STARLAN)รุ่นที่3ได้เพิ่มมาตรฐาน OSIในชั้นทรานสปอร์ต(TRANSPORT) และเน็ตเวิร์ค(NETWORK) และยังมี แมนูแฟคเจอริ่ง ออโตเมชั่น โปรโตรคอล/เทคนิค ออฟฟิส โปรโตคอล(Manufacturing Automation Protocol/Technice Office Protocol MAP/TOP) ผลิตภัณฑ์ที่ได้มีความเร็ว 10 เมกะบิท/วินาที มาตรฐาน IEEE

โปรโตคอล พวกนี้สามารถใช้สำหรับ UNIX และ MS-DOS และยังใช้ฟังก์ชันในการสื่อสารต่างๆบนเน็ตไบออส ทรานสปอร์ต เลเยอร์ อินเตอร์เฟส(NETBIOS Transport Layer Interface (TLI)) สตาร์แลนเป็นโครงข่ายท้องถิ่นซึ่งมีรูปแบบการต่อแบบรูปดาวซึ่งทำให้ง่ายในการที่จะต่อกับสาย ทวิสต์แพร์(Twisted pair line) และเพิ่มความเร็วขึ้นเป็น 10 เมกะบิท/วินาที ในการที่ใช้โปรโตคอลของ OSI โมเดลนั้นทำให้เลือกการใช้งานได้อย่างกว้างขวางทั้ง ฮาร์ดแวร์ ตัวกลาง และตัวปฏิบัติการ

AT&T ผลิต สตาร์แลน รุ่นแรกความเร็ว 1 เมกะบิท/วินาที เพื่อที่จะทำเป็นโครงข่ายและสามารถใช้งานได้กับสายโทรศัพท์ทำให้ประหยัดกว่าที่จะใช้สายโคแอกเซียล และได้เพิ่มความเร็วเป็น 10 เมกะบิท/วินาที แบบรูปดาวนั้นสะดวกในการใช้กับโครงสร้างอาคารที่มีสายโยงอยู่แล้ว และยังทำเป็นรูปดาวลดหลั่นกันลงไปได้

ความเป็นมาของสตาร์แลน

สตาร์แลนอยู่ภายใต้มาตรฐาน IEEE 802.3 ซึ่งมาตรฐานนี้ใช้หลักการ CSMA ซึ่งใช้งานครั้งแรกที่มหาวิทยาลัยแห่งฮาวาย ในระบบของโครงข่ายวิทยุ Aloha Packet และทำการปรับปรุงโดยบริษัท Xerox Corp. และได้เติม CD เมื่อเกิดการชนกันของข้อมูล

สถานีที่ส่งจะต้องจำการชนกันด้วย เมื่อมีการชนกันเกิดขึ้นต้องออกจากการส่งและรอระยะเวลาเล็กน้อยในการเริ่มต้นส่งใหม่เรียกว่าแบ็ค ออฟ ไทม์(back-off time) ส่วนสถานีอื่นๆก็ต้องคอยจนกว่าสถานีแรกจะส่งข้อมูลเสร็จ เวลาแบ็คออฟ นั้นมีกำหนดไว้ในมาตรฐาน IEEE 802.3

อุปกรณ์ที่ใช้ในตอนต้นๆของ 802.3 ก็คล้ายๆกับอีเทอร์เน็ตแลน(Ethernet Lan) ที่ใช้มาตรฐาน IEEE 802.3 Base5 ซึ่งใช้กับสายโคแอกเซียลอย่างหนา ความยาวไม่เกิน500เมตร ในการเชื่อมต่อสายโคแอกเซียล 2 สาย ทำโดยผ่าน รีพีทเตอร์ ต้องต่อก่อนที่สัญญาณจะผิดเพี้ยนและลดทอนมากเกินไป

ในช่วงต้นของ ค.ศ.ที่ 84 IEEE ได้จัดทำเวิร์คกิงกรุป (WorkingGroup)เพิ่มเข้าไปในระบบ บัส (Bus) 10Base5 ซึ่งเพิ่มเข้าไปใน สตาร์แลน 1เมกะบิท/วินาที และยังมีการใช้คอร์มัลติเพิลรีพีทเตอร์ (Core Multiple repeater)

มาตรฐานและตัวกลาง

มาตรฐาน	ตัวกลาง	ความเร็ว (เมกกะบิท/วินาที)	ระยะทาง (เมตร)	หมายเหตุ
10 BASE5	สายโคแอคเชียล	10	500	เหมือนอีเทอร์เนต
10BASE 2	สายโคแอคบาง	10	185	ชีพเปอร์เนต
1BASE 5	สายโทรศัพท์	1	500	สตาร์แลน(รัศมี 250 เมตร)
10BROAD36	ระบบบรอดแบน	10	3600	1800เมตร-รัศมี
. FOIRL	ออบติคไฟเบอร์	10	1000	

เทคโนโลยีและองค์ประกอบ

มีตัวแปรอยู่สองตัวแปรที่สามารถทำให้เราใช้สายทวิสต์แพร์โดยตอนแรกจะใช้ส่งเสียงที่10เมกะบิท/วินาที

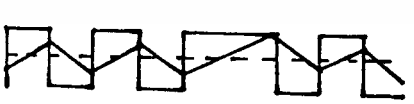
1 ทำให้ระยะระหว่างผู้ใช้สองจุดอยู่ห่างกันไม่เกิน 200 ฟุต

2 ทำการชดเชยสัญญาณก่อนที่จะมีการผิดเพี้ยนเนื่องจากเราส่งสัญญาณความถี่สูงไปในสื่อกลางความถี่แคบ ทำให้สัญญาณเลวลง

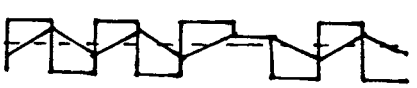
ดูจากรูปที่1ในระบบอีเทอร์เน็ต สตาร์แลน1และสตาร์แลน10 จะใช้การเข้ารหัสแบบแมนเชสเตอร์ (Manchester) ซึ่งมีการเปลี่ยนแปลงค่าทุกๆครึ่งบิทหรือทุกๆ1บิท แต่ถ้าเราส่งสัญญาณลงไปหนึ่งบิทเต็มทำให้จุดที่จับการเปลี่ยนแปลงสัญญาณเลื่อนออกไปทำให้วงจรด้านรับ รับข้อมูลแล้วแปลผิดพลาด

แต่การชดเชยสัญญาณก็ทำได้ในการส่งในระยะทางไม่เกิน 100 เมตร แต่ก็ยังทำได้ยากกว่าระยะสั้นๆอยู่ดี ในระบบ สตาร์แลน10 มีอุปกรณ์ในหลายๆส่วนประกอบไปด้วย

ไม่มีกลไกป้องกันการลดทอน



มีกลไกป้องกันการลดทอน



รูปที่ 1 รีดิวซิ่ง จิสเตอร์

ตัวกลางแบบเส้นลวด (Wire Hub)

แก่นแท้ของ สตาร์แลน1 อยู่ที่ ฮับ(Hub) ซึ่งเป็นตัวรวบรวมสัญญาณ ฮับ แต่ละตัวจะรวมสัญญาณข้อมูลมาจากแหล่งข้อมูลและส่งผ่านข้อมูลไปที่ฮับที่ระดับสูงกว่า หลังจากนั้นก็จะส่งข้อมูลเหล่านั้นไปที่สถานีต่างๆ

ฟังก์ชันโดยทั่วๆไปของ ฮับ จะเพิ่มในส่วนของการตรวจจับการชนกัน โดยที่ ฮับ จะส่งสัญญาณพิเศษบอกว่ามีการชนกันเกิดขึ้นแต่ละสถานีก็จะจำได้ สถานีส่งก็จะหยุดส่งและรอช่วงเวลาส่งใหม่

ฮับ ของ สตาร์แลน10 อยู่บนพื้นฐานของ มัลติพอร์ทรีพิทเตอร์ ซึ่งเป็นอุปกรณ์ที่ใช้ในการเชื่อมต่อภายในลักษณะที่ไม่เหมือนกับ ฮับ ของ สตาร์แลน1 อับกอลิทึมของ มัลติพอร์ทรีพิทเตอร์ จะส่งผ่านข้อมูลไปที่ทุกๆพอร์ทยกเว้นแหล่งที่มาของข้อมูล ทุกๆสถานีจะจำได้ว่าการทีมีขอ้มูลเข้าและออกพร้อมๆกันคือการชนกัน

รูปแบบของ ฮับ ของ สตาร์แลน10 จะมีสาย 10 เส้นที่ใช้ในการต่อกับผู้ใช้หรือ ฮับ ตัวอื่นๆสายเส้นที่11จะบอก IN/OUT ซึ่งเป็นตัวบอกว่าสายสองเส้นนั้นจะรับข้อมูลเข้ามาหรือนำออกไป เป็นการลดความต้องการในเรื่อง เอกซ์เทอร์นัล อแดปเตอร์ นัลโมเด็ม(External อแดปเตอร์ Null Modem) หรือกระตุ้นให้รับหรือส่ง นอกจากนี้ ฮับ ยังมี พอร์ท AUI ซึ่งสามารถที่จะเชื่อมต่อกับ MAU ใดๆก็ได้เช่น สายโคแอกเซียล อย่างหนา โคแอกเซียวอย่างบาง ออฟติคอลไฟเบอร์ บรอดแบนด์เซมิริกิดโคแอกเซียล(broadband semirigid coax).และยังสามารถที่จะเชื่อมต่อสายทวิสต์แพร์ อีกด้วย

ในคำจำกัดความของ IEEE แล้ว ฮับเป็นรีพีทเตอร์ที่มีหลายๆพอร์ทเชื่อมต่อกับทวิสต์แพร์MAUs สำหรับ 11 พอร์ท

พีซี/หน่วยที่เข้าถึงโครงข่าย (The PC/Network Access Unit)

NAU ของสตาร์แลน 10 ถูกติดตั้งเข้าไปใน MS-DOS ตั้งแต่เครื่องคอมพิวเตอร์รุ่น XT ขึ้นไป เช่น AT&T PC-6300 จะมีพอร์ทเอาท์ที่ใช้ในการเชื่อมต่อกับฮับภายในท้องถิ่นนั้นหรือโดยผ่าน AT&T Premises Distribution System ไปเชื่อมต่อกับฮับตัวที่อยู่ใกล้ที่สุด

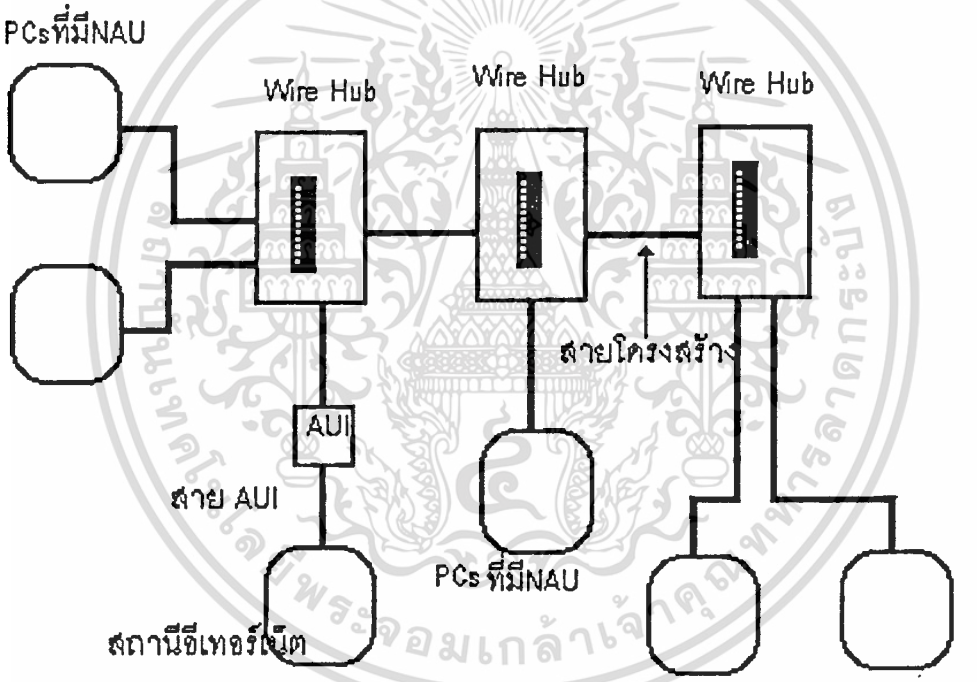
ในคำจำกัดความของ IEEE แล้ว NAU จะบรรจุทวิสต์แพร์ MAU ที่เชื่อมต่อภายใน

อแดปเตอร์ เอยูไอ(The AUI Adapter)

AUIอแดปเตอร์ อนุญาตให้สถานี bridge และจุดปลายทางต่างๆในโครงข่ายซึ่งมี AUI เพื่อที่จะเชื่อมต่อพันสาย ทวิสต์แพร์ ไปที่ฮับของสตาร์แลน 10 ในคำจำกัดความของ IEEE นั้น AUI อแดปเตอร์ จะถูกพิจารณาตาม ทวิสต์แพร์ MAU ซึ่ง AUI มีทั้งอินและเอาท์ และยังเชื่อมต่อกับ AUI พอร์ทของวายน์ฮับตัวอื่นๆหรือออฟติคอลฮับ ในการที่จะเพิ่มพอร์ทของสายทวิตแพร์

รูปที่2 สตาร์แลน-10แบบพื้นฐาน

เป็นพื้นฐานการเชื่อมต่อที่ใช้สายเท่านั้นของสตาร์แลน-10 ซึ่งสายและสถานีอีเทอร์ต่อเข้ากับฮับแบบรูปดาว



ตัวศูนย์กลางแบบเส้นใยแสง(The Optical Fiber Hub)

หน้าที่การทำงานของออฟติคอลไฟเบอร์ฮับจะคล้ายคลึงกับการทำงานของวายน์ฮับและ มัลติพอร์ต รีพิทเตอร์ วีแอลเอสไอ ชิบ(Multiport Repeater VLSI Chip AT&T) อย่างเป็นหลักที่ใช้ในออฟติคอลและวายน์ฮับ ออฟติคอลไฟเบอร์ฮับมี 6 ไฟเบอร์พอร์ท 1อินหรือเอาท์พอร์ท
ของทวิตแพร์ และ 1 AUI พอร์ต

หน่วยเข้าถึงระบบอีเอ็น100 (EN 100 Network Access Unit)

EN100 เหมือนกับPC NAU แทนที่จะต่อกับสายทวิตแพร์ จะต่อกับสาย อีเทอร์เน็ตซีฟเปอร์เน็ต(Ethernet Cheapernet) (10 Base5/10Base2) มันมีพอร์ทของ AUI ใช้ต่อกับ MAU ภายนอก และ AUI ทำให้ EN 100 สามารถจะเชื่อมต่อกับสายทวิตแพร์ได้

หน่วยเข้าถึงระบบเอ็มซี100 (MC100 Network Access Unit)

MC100 จัดให้มีการเชื่อมต่อกับพีซี ซึ่งมีการเชื่อมต่อแบบ ไมโครแชลเนล(micro channel) จะมี AUI พอร์ต ซึ่งจะต้องต่อกับ MAU ภายนอก AUI พอร์ต ทำให้ MC100เชื่อมต่อกับทวิสต์แพร์ได้

อแดปเตอร์แบบเส้นใยแสง(The Optical Fiber Adapter)

มันเหมือนเป็นการเพิ่มความยาวของเส้นลวดถ้าตัวติดตั้งการเชื่อมต่อไม่สามารถที่จะต่อแบบจุดต่อจุดได้ไกลเกิน 100 เมตร ก็จะใช้การเชื่อมต่อแบบออฟติคอลไฟเบอร์ ซึ่งออฟติคอลไฟเบอร์อแดปเตอร์ ทำให้วายน์ฮับติดต่อกับไฟเบอร์ฮับที่อยู่ไกลออกไป ออฟติคอลไฟเบอร์ยังใช้ในพื้นที่ ที่มีการรบกวนทางไฟฟ้าสูง และการเชื่อมต่อจากเส้นลวดไปยังออฟติคอลต้องแปลงเป็นแสงเสียก่อน

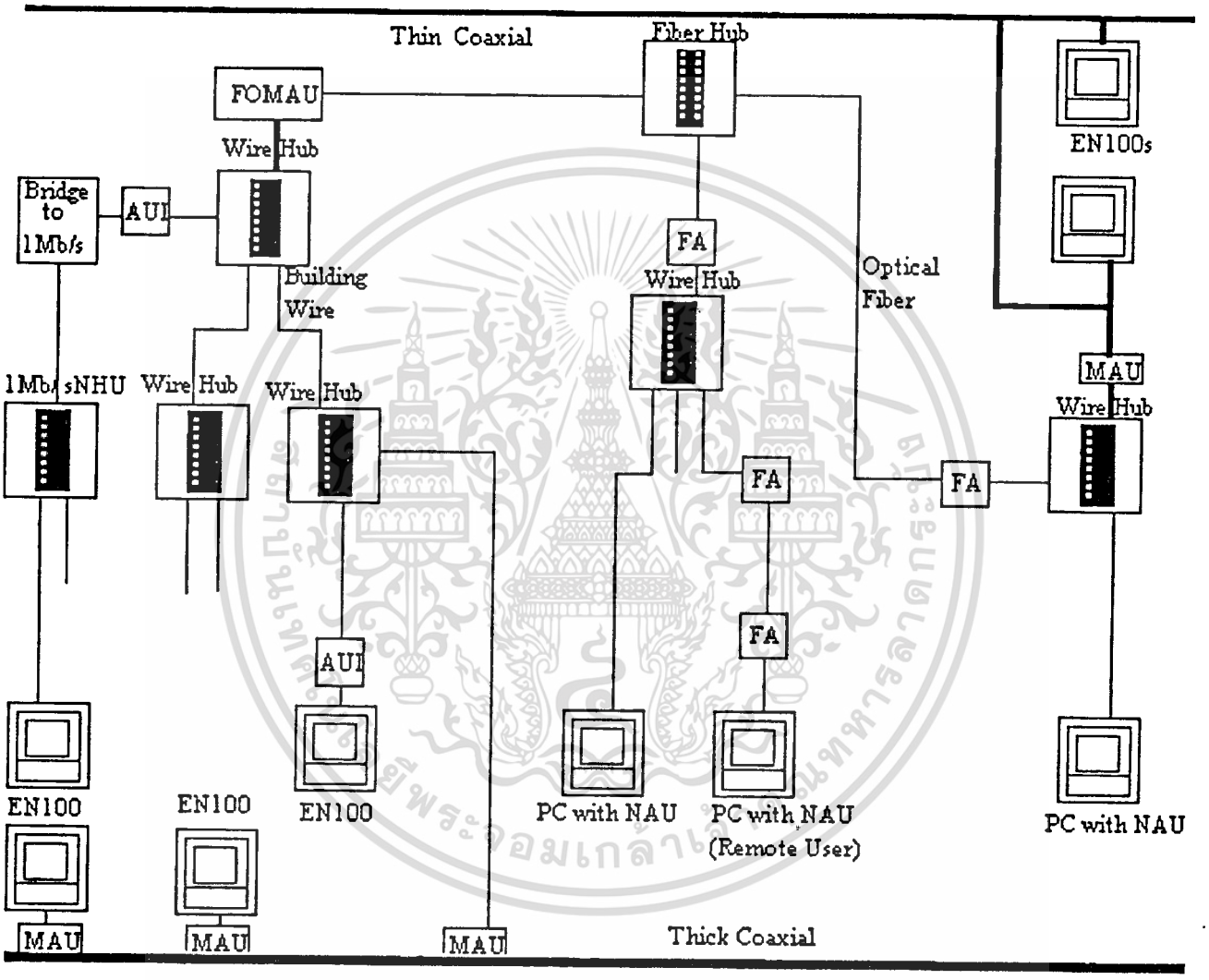
หนึ่งต่อสิบบริดจ์(The 10:1 Bridge)

บริดจ์ สามารถเชื่อมต่อโครงข่าย 1 เมกะบิท/วินาที กับโครงข่าย 10เมกะบิท/วินาที รูปที่ 3 แสดงการเชื่อมต่อแบบต่างๆในโครงข่ายเดียว มันดูไม่เหมือนเป็นโครงข่ายเป็นรูปแบบการต่อทั้งสตาร์แลนและอุปกรณ์ 802.3

สำหรับระยะทางที่เกิน 100 เมตร จะใช้ไฟเบอร์แดปเตอร์ในการที่จะเชื่อมต่อกับๆไฟเบอร์ฮับ ซึ่งทำให้สามารถขยายระยะทางได้ถึง 2 กิโลเมตร

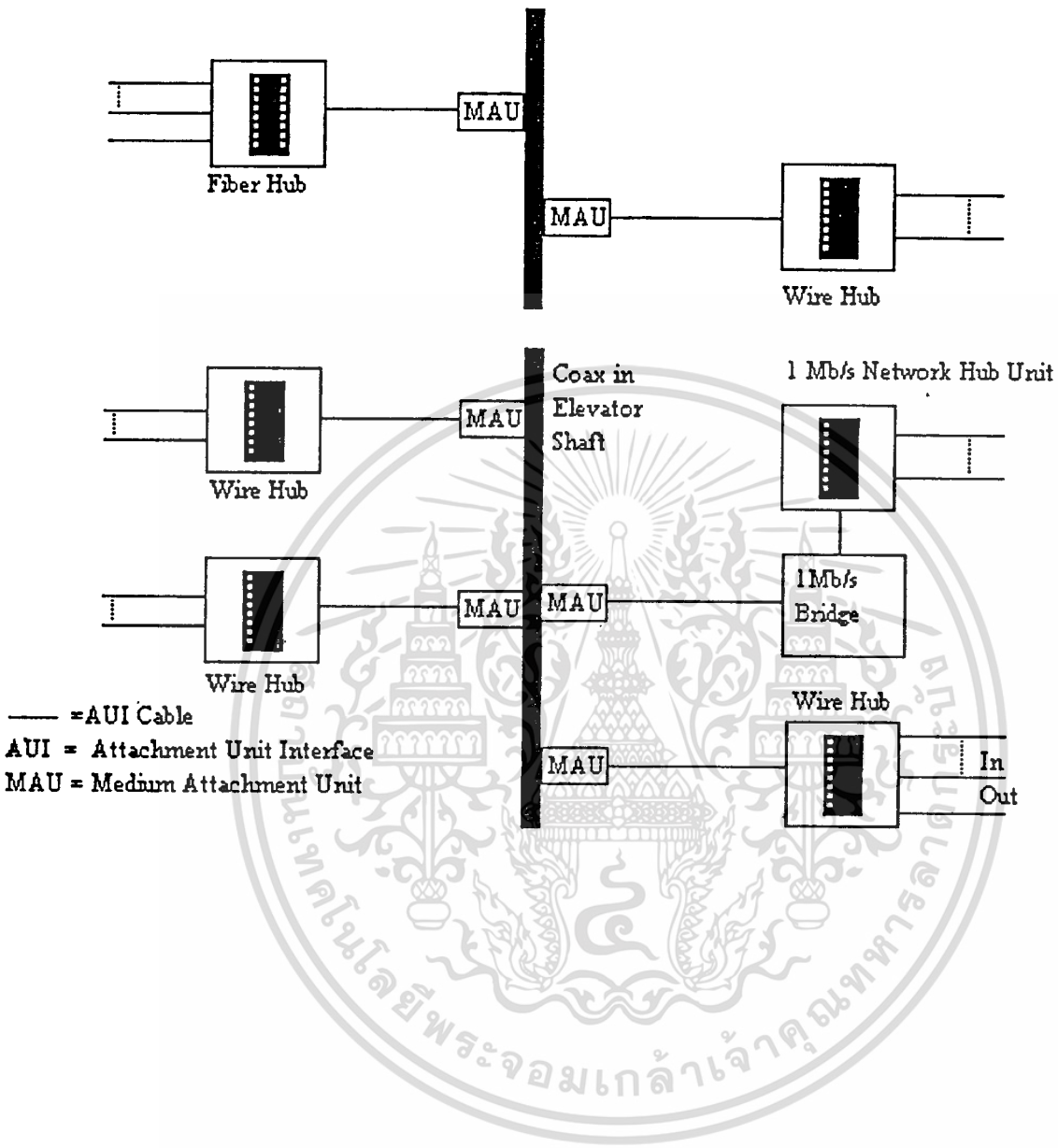
โครงข่ายของEN100 และMc100 ถูกเชื่อมต่อเข้ากับเอยูไออแดปเตอร์ธิคโคแอคเชียลเคเบิลเอมเอยู (AUI Adapter Thick coaXial Cable MAUs) หรือต่อโดยตรงกับสายโคแอกเชียลอย่างบาง โครงสร้างที่มากไปกว่านี้ โครงข่ายที่ต้องการสำหรับการติดตั้งใหม่จะเน้นในส่วนของ The AT&T Premises Distribution System พร้อมด้วยสายลวดและไฟเบอร์ ลำดับขั้นตอนในการจัดเตรียมต้องเคร่งครัดลดการตรวจตราและดูแลรักษาลำบาก

รูปที่ 4 เป็นรูปแบบสุดท้ายซึ่งเป็นกรณีพิเศษที่ใช้โคแอกเชียลเคเบิลที่มีอยู่แล้ว สายที่เป็นตัวจ่ายเป็นสายแบลกโบน ข้อได้เปรียบคือ ลดการดีเลย์ของสัญญาณ ข้อสังเกตผู้ใช้ในสตาร์แลนจะติดต่อกันต้องผ่าน



รูปที่ 3 การเชื่อมต่อแบบ Star Lan

รูปที่ 4 สายหลักของโคแอกเชียล(COAX BACKBONE)



ยับ

ซอร์ฟแวร์ในการเชื่อมต่อชั้นเน็ตเวิร์คและทรานสปอร์ต(LINKNETWORKAND TRASPORT LAYER SOFTWARE) ในส่วนนี้จะอธิบายถึง โปรโตคอลของชั้นที่เหลือ รูปที่ 5 แสดง Starlan ISO Protocol Network Naming and Addressing ในระบบ สตาร์แลนผู้ใช้ไม่จำเป็นต้องรู้เกี่ยวกับ ฮาร์ดแวร์ในระบบ อุปกรณ์จะถูกเรียกชื่อโดยแทนด้วยหมายเลข 16 ตัวอักษร ซึ่งใช้ได้กีมาตรฐานการแปลงชื่อของ NETBIOS โปรแกรมของโครงข่ายสามารถที่จะเปลี่ยน แอดเดรสมากกายภาพได้โดยอัตโนมัติ

ชื่อในสตาร์แลนสามารถใช้ร่วมกันได้จากเครื่องต่างๆในโครงข่ายเดียวกัน ซึ่งรูปแบบนี้มีประโยชน์มาก เช่น การส่งดาต้าแกรมไปที่ปลายทางหลายๆที่ในเวลาเดียวกันและมีการใช้ทรัพยากร
ร่วมกันได้อีก

ทรานสปอร์ตแอดเดรส มีอยู่ 3 ส่วน

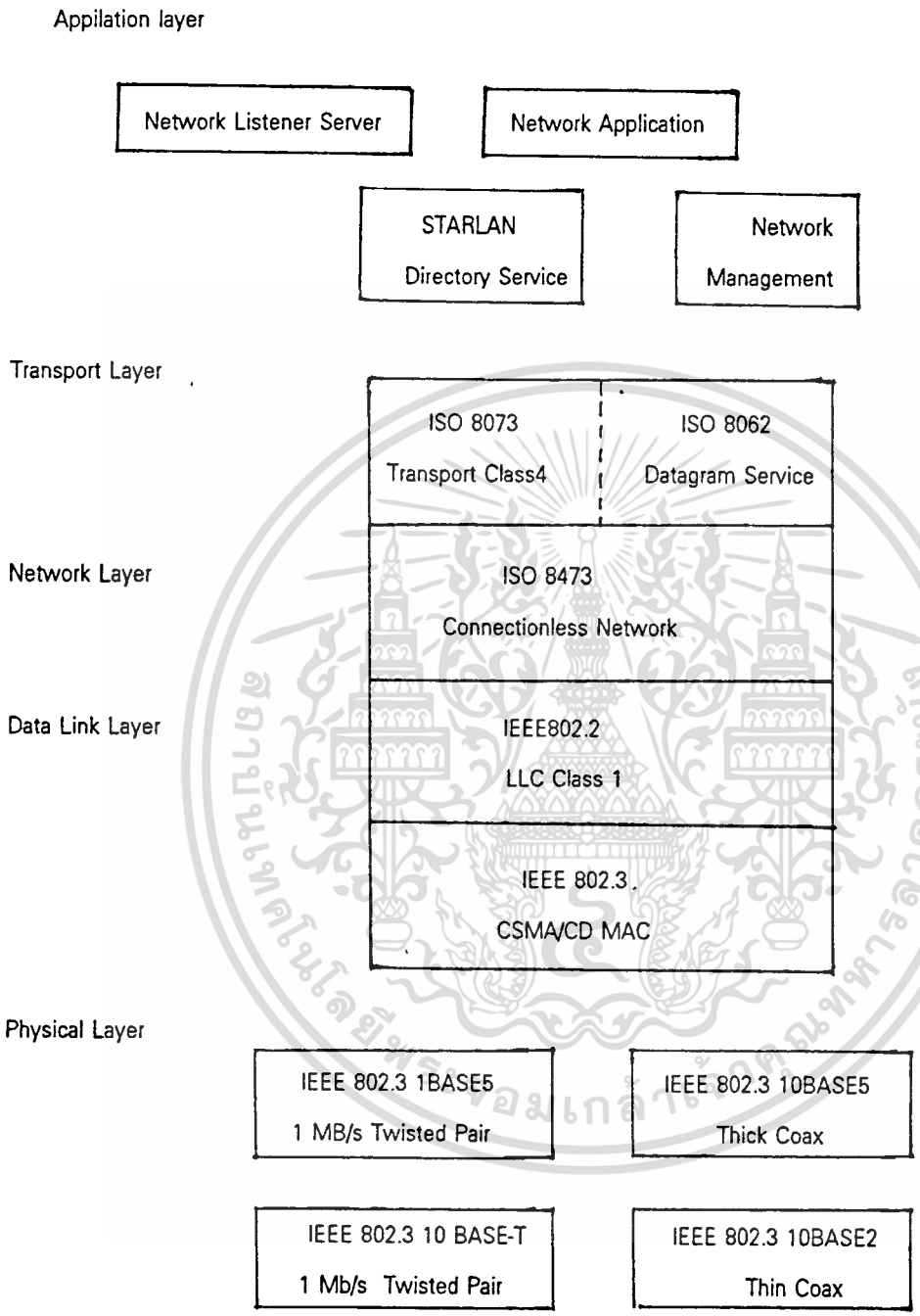
- เน็ตเวิร์คนัมเบอร์(Network Number) ซึ่งเป็นตัวบอกชื่อแลน
- ฟิสิคัลฮาร์ดแวร์แอดเดรส(Physical hardware Address)เป็นตัวลอกหน่วยที่สามารถเข้าจัดการกับระบบ สตาร์แลนได้
- เอ็นโค้ดเน็ตไบออสเนม(Encode NETBIOS Name) ใช้โปรแกรมประยุกต์ได้ตรงภายในเครื่องจุดหมาย ปลายทาง

การบริการไดเรกทอรีของสตาร์แลน (Starlan Directory Service)

เนื่องจากชื่อ NETBIOS และแอปลิเคชั่น อยู่ในเมโมรีแทนที่จะอยู่ในฐานข้อมูล ทำให้เกิดการเปลี่ยนแปลงง่าย และมันจะหายไปเมื่อปิดเครื่อง หรือ เคลื่อนย้ายไปโครงข่ายใหม่เมื่อแอฟลิเคชั่นย้าย ดังนั้นในการจัดบริการสร้างไดเรกทอรีของชื่อเป็นการแก้ปัญหาและเก็บค่าจุดปลายทางต่างๆใช้ในฐานข้อมูลชื่อ

ยกตัวอย่าง ถ้าลูกค้าผู้ใช้เวิร์คสเตชั่น ต้องการติดต่อกับ ไฟล์เซิร์ฟเวอร์1(fileserver1) มันจะถามชื่อและเซิร์ฟเวอร์ตัวนั้นก็จะโต้ตอบกลับมาพร้อมด้วย ทรานสปอร์ตแอดเดรส ของมัน
ชื่อของเซิร์ฟเวอร์ จะถูกลงทะเบียนไว้ตอนที่เปิดเครื่องใหม่ๆโดยที่จะส่งกระจายไปที่เครื่องต่างๆให้รับรู้ในโครงข่าย) เวิร์คกิงกรุป ของ TOP ได้พัฒนามาตรฐานของ NETBIOS ในการสร้างไดเรกตอรีของชื่อ เพื่อที่จะทำให้มีการเชื่อมต่อกันระหว่างส่วนเสริม NETBIOS ISO สตาร์แลนที่รันบนยูนิกซ์จะใช้แอดเดรสมากกว่าจะใช้ชื่อ

รูปที่ 5 โปรโตคอลของISO



LLC = Logical Control

MAC = Media Access Control

CSMA/CD = Carrier Sense Multiple Access/Collosion Detection

โปรโตคอลโอเอสไอ(OSI Protocol)

สตาร์แลนได้ใช้มาตรฐาน ISO หลายมาตรฐานเช่น

- ISO 8073 ซึ่งเป็นการส่งข้อมูลแบบเวอร์ชวลเซอร์กิต มีโฟลคอลโทรล เออเรอดีเทคชันและการกู้คืนสัญญาณ
- ISO 8062 คอนเน็คชั่นเลส ทรานสปอร์โปรโตคอล สำหรับการส่งดาต้าแกรมแบบง่ายๆ
- ISO 8470 คอนเน็คชั่นอินเตอร์เน็ตโปรโตคอลรวมทั้งมีกลไกลการหาเส้นทางทั้งของตอนท้ายของระบบและตอนกลางของระบบ(กลไกนี้อนุญาตให้มีการส่งข้อมูลจากจุดปลายของสตาร์แลนผ่าน แวน(WAN) ไปที่จุดปลายทางที่อยู่ห่างไกลออกไป
- ISO 802.2 การบริการด้านการเชื่อมแบบลอจิกคอล และมีการ ดีมัลติเพลกซ์เพคเกจ(demultipkex packet)ที่มาถึง

การเชื่อมต่อกับโปรแกรมระดับสูง (Application Programming Interfaces)

ในโครงสร้างของสตาร์แลนในปัจจุบันมีการสร้างแอปลิเคชั่นโปรแกรมอินเตอร์เฟส 2แบบคือในชั้น ทรานสปอร์ต ภายใต้ระบบปฏิบัติการยูนิกซ์ และNETBIOS ภายใต้ MS-DOS ซึ่งมันเราสามารถใช้ตัวไหนขึ้นอยู่#กับเครื่องที่ใช้ ถึงแม้ว่าการอินเตอร์เฟสจะแตกต่างกัน แต่ก็มีความสามารถคล้ายๆกันเช่นการส่งแบบเวอร์ชวล เซอร์กิต การส่งผ่านข้อมูล ดาต้าแกรม การจัดการโครงข่าย พูดให้สั้นๆคือฟังก์ชันของ NETBIOS ก็คือฟังก์ชันของยูนิกซ์ระบบTLIและลิซีเนอร์ของโครงข่ายก็มีอยู่ในยูนิกซ์ช่วยพัฒนาแอปลิเคชั่นของลูกค้าและเซิร์ฟเวอร์

ระบบยูนิกซ์ทีเอลไอ (Unix system TLI)

ระบบยูนิกซ์วี3 ได้สร้างมาตรฐานการเชื่อมต่อTLIซึ่งได้จัดสร้างฟังชันก์ต่างๆที่จะเข้าจัดการระบบยูนิกซ์และทรานสปอร์ตและฟังก์ชันของทรานสปอร์ตนั้นจะอิงมาตรฐานISO 8072 และสามารถที่จะยังมีการจัดหาตัวทรานสปอร์ตแบบอื่นเช่นทรานสมิทชั่นคอนโทรลโปรโตคอล(transmission Control Protocol TCP)และX.25 ซึ่งฟังก์ชันของTLI มีอยู่3ฟังก์ชัน

1 ฟังก์ชันในการจัดการในท้องถิ่น ซึ่งเป็นตัวอนุญาตในการปิดเปิดตัวทรานสปอร์ตในเครื่องในท้องถิ่น ลงทะเบียนแอปลิเคชั่นแอเดรส ตกลงการทำงานของทราสปอร์ตและตรวจดูการทำงานต่างๆ

2 ฟังก์ชันการบริการการเชื่อมต่ออนุญาตให้แอปลิเคชันโปรแกรมเรียกโปรแกรมอื่น รับการร้องขอ รับและส่งข้อมูลผ่านการเชื่อมต่อ และยกเลิกการเชื่อมต่อ

3 ฟังก์ชันการไม่เชื่อมต่อ(Connectionless) สนับสนุนการส่งข้อมูลรับข้อมูลแบบดาต้าแกรม ของแต่ละ

บุคคลหรือกลุ่ม

เอ็มเอสดอส เน็ตไบออส (MS-DOS NETBIOS)

เอ็มเอสดอส ลีลิส3 สนับสนุนการเชื่อมต่อแบบ NETBIOS ที่รู้กันในนาม INT 2A หรือINT 5C ซึ่งใช้ตอบโต้การอินเทอร์รัพต์ทางซอร์ฟแวร์ ที่ใช้โดย MIcrosoft MS-NET และIBM PC-NETซึ่งหน้าที่หลักๆของNETBIOSก็คล้ายคลึงกับTLIเปรียบเทียบกับตารางที่2ที่แสดงความคล้ายคลึง
ของฟังก์ชัน TLI และ NETBIOS

NETBIOS ทำให้แอปลิเคชันต่างๆสามารถที่จะรันบนสตาร์แลนได้นอกจะมีการบริการพื้นฐานทางด้านไฟล์และพิมพ์แล้วยังมีการจำลองเทอร์มินัล เกจเวย์ ที่เป็นซิงโครนัสและอซิงโครนัส บริการฐานข้อมูล และการส่งเมล์ของโครงข่าย(AT&T Mail)

ตัวอย่างTLIและกรเปรียบเทียบเนตไบออส

ฟังค์ชั่น	TLI Call	เนตไบออสฟังค์ชั่น
Name registration	t_bind	add name
Connection establishment	t_connect	call
Listen for connection	t_listen	listen
Send data	t_snd	send
Receive data	t_reive	receive
Disconnect	t_snddis	hang up

เน็ตเวิร์คลิซีเนอร์(Network Lister)

ระบบยูนิกซ์จะมีเน็ตเวิร์คลิสซีเนอร์เนื่องจากเซิร์ฟเวอร์ไม่รู้ว่าเมื่อไหร่จะมีลูกค้าเรียกเข้ามาดังนั้นมันต้องเตรียมพร้อมอยู่เสมอ

แอปลิเคชันของระบบจะต้องฟังและรับการร้องขอการเชื่อมต่อ มันเป็นการไม่ประหยัดถ้าจะรันเซิร์ฟเวอร์โดยที่ไม่มีลูกค้าคนไหนทำงาน TLI ลิซีเนอร์รันภายใต้ระบบปฏิบัติการของยูนิกซ์

ลิซีเนอร์ของระบบยูนิกซ์จะลงทะเบียนชื่อและแอดเดรสของ TLI ทรานสปอร์ตที่ม่ปละรอการร้องขอการเชื่อมต่อ ถ้ามีการร้องขอเข้ามา ลิซีเนอร์ก็จะดูที่เป็นการร้องขอติดต่อกับเซิร์ฟเวอร์ตัวไหนลิซีเนอร์ก็จะทำให้เซิร์ฟเวอร์ตัวนั้นปฏิบัติการ และรอฟังการร้องขอครั้งต่อไป ความสามารถของลิซีเนอร์ถูกใช้โดยสตาร์แลนเอ็มเอสดอสไฟล์เซิร์ฟ ระบบยูนิกซ์ยูยูซีพี และเน็ตเวอร์ลอคอินเซอรวิส และระบบยูนิกซ์อาร์เอฟเอส

ส่วนเพิ่มเติมของโปรโตคอลไอเอสโอ(ISO PROTOCOL IMPLEMENTATION)

รูปที่7 การแสดงส่วนเพิ่มเติมของสตาร์แลนไอเอสโอโปรโตคอล ในการทีจะให้การพัฒนาให้เล็กลงการทดสอบและราคาการซ่อมบำรุง ความสำเร็จหลักๆของสตาร์แลนไอเอสโอโปรโตคอลคือยังคงมีความเข้ากันได้ของยูนิกซ์และเอ็มเอสดอสโปรโตคอล ยังรวมไปถึงการเข้ากันได้ของฟังก์ชันและโค้ด ในการสำเร็จครั้งนี้คือการพัฒนายูนิกซ์รันบนเอ็มเอสดอส

สตรีมซัฟพอร์ทแอปลิเคชันซึ่งรูปแบบของโปรโตคอลสามารถส่งรับและปฏิบัติการข้อมูลและควบคุมข้อมูลในลักษณะที่ต้องการ และยังทำให้มีความยืดหยุ่นในการเชื่อมต่อ และยังสามารถขยายการจัดการฟเฟอร์และยังสนับสนุนในทางด้านเครื่องมืออีกด้วย

สตาร์แลนไอเอสโอโปรโตคอลถูกเสริมเพิ่มแบ่งเป็น3ส่วนที่เอามารวมกัน

- ทรานสปอร์ต
- เน็ตเวิร์ค
- ดาต้าลิงเลเยอร์

ซึ่งมันแบ่งเป็นชั้นๆและความสามารถในการรวมทำให้ได้รูปแบบที่ยืดหยุ่นมากขึ้นการเชื่อมต่อกับยูนิกซ์มันขึ้นอยู่บนพื้นฐานของการเชื่อมต่อ TLI ระดับผู้ใช้และมาตรฐานของยูนิกซ์ซิสเต็มสตรีมเฮด การเชื่อมต่อกับกับเอ็มเอสดอสนั้น โดยผ่านINT 5C ในการอินเทอร์รัพและแบ่งการเชื่อมต่อNETBIOS ซึ่งต้องการฟังก์ชันของ NETBIOS

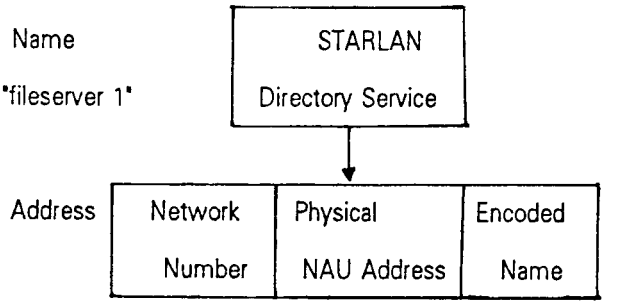
ในส่วนเสริมของสตาร์แลนเน็ตไบออส จัดให้มีการอินเตอร์เฟสแต่ไม่ใช่แบ่งเป็นชั้นหนึ่งไปเลยของ IBM PC NET ส่วนเสริมสตาร์แลนจัดให้มีประโยชน์ของแอปลิเคชันของ NETBIOS ไม่ต้องการเพิ่มเติมในส่วนบนของชั้นISO ทรานสปอร์ต

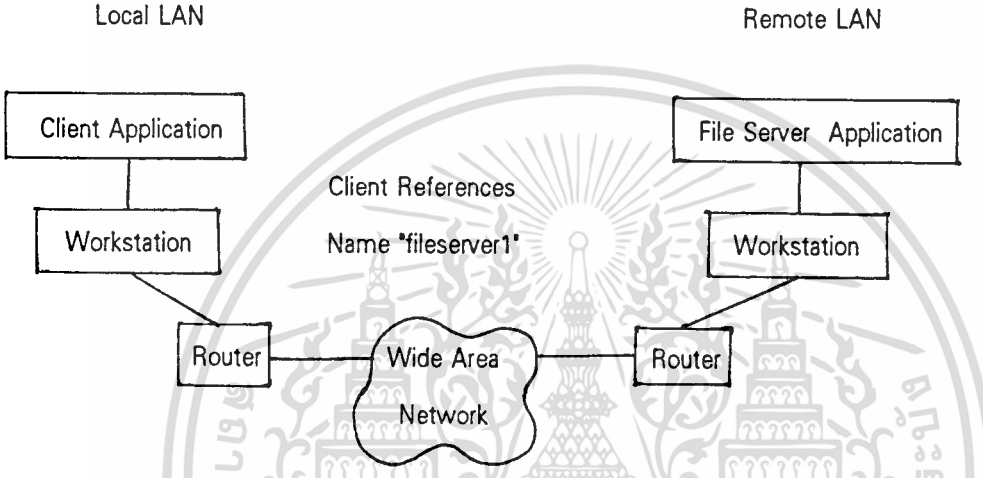
สตาร์แลนไดเรกทอรีเซอร์วิส ซึ่งมีการบริการจัดชื่อบน NETBIOS ได้แยกออกมาเป็นส่วนหนึ่งภายใต้ซอร์ฟแวร์ของยูนิกซ์ มันสามารถที่จะเข้าถึงดาต้าลิงค์เลเยอร์ได้โดยตรงเป็นข้อได้เปรียบของความสามารถในการกระจายข่าว

การปรับการทำงานของโครงข่าย (Tuning Network Performance)

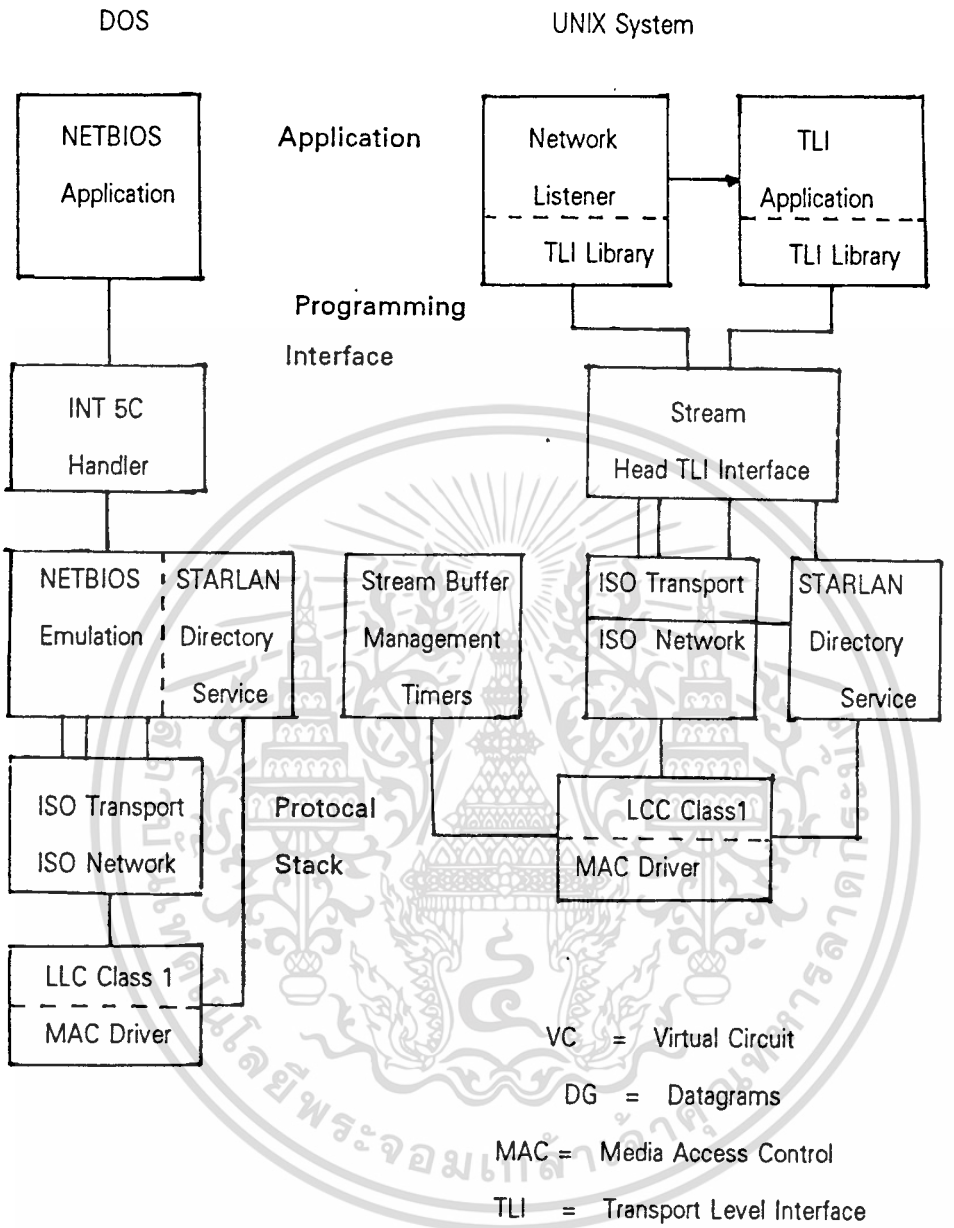
สตาร์แลนลีลิส3 ปฏิบัติการเป็น2เท่าของรุ่นก่อนๆทั้ง ทรูพุท(Throughput) และเลเทนซี่(Latency) ทรูพุทอธิบายในรูปไบต์/วินาที เป็นการจัดประสิทธิภาพของการส่งและการใช้งานโปรแกรม เลเทนซีเป็นการหน่วงเวลาโครงข่ายและใช้วัดประสิทธิภาพในการส่งข้อมูลเป็นบล็อคเล็กๆในการที่จะเพิ่มเนื้องานและการหน่วงเวลาใช้น้อยลงแต่ยังคงฟังก์ชันการทำงานต่างๆไว้นั้นได้ใช้เทคนิคหลายๆเทคนิคเข้ามาช่วยโดยใช้ ISO

รูปที่6 นามมิงและ แอดเดรสซิง





รูปที่7 ส่วนเพิ่มเติมโปรโตคอล



ทรานสปอร์ตและเน็ตเวิร์คโปรโตคอลและลดจำนวนส่วนหัวของข้อมูลลง เหล่านี้รวมทั้ง

1 โค้ดที่มีขนาดใหญ่ก็ทำการตรวจดูใหม่และทำการเปลี่ยนแปลงโค้ดโดยใช้หลักการออฟติไมเซชัน

2 ขยายและทำซ้ำในส่วนเนื้อหาในการวัด เพื่อที่จะบอกซื่อบอทเทิลเน็ค

3 การใช้ประโยชน์ISOทรานสปอร์ตโปรโตคอล เป็นการลดเพ็คเกจของเอคโนเลจด์

4 ใช้อัลกอริทึมซึ่งลดเครดิตที่เสนอให้(ขนาดวินโด) ซึ่งจัดให้มีเวอร์ชวลเซอร์กิตจำนวนมากต้องการในสภาวะของเซิร์ฟเวอร์พร้อมกับผลกระทบในการใช้เหน่วยความจำน้อยที่สุดและการปฏิบัติการ

ข้อพิจารณาพิเศษ(Special Consideration)

ถึงแม้จะพยายามที่จะทำการใช้ร่วมกันให้มากที่สุดระหว่างยูนิกซ์และเอ็มเอสดอส ความแตกต่างระหว่างความต้องการของผลิตภัณฑ์และเงื่อนไขต่างๆตามแต่จำนวนของความแตกต่างยูนิกซ์มีการจัดหาหลายรูปแบบที่เอ็มเอสดอสไม่มี

1 โครงร่างการเชื่อมต่อแบบเวอร์ชวลเซอร์กิตจำนวนมาก โดยเฉพาะอย่างยิ่งสำหรับเซิร์ฟเวอร์ขนาดใหญ่และโครงร่างของเกจเวย์

2 โครงร่างของการมัลติเฟลกซ์ที่สลับซับซ้อนที่จัดให้เช่น มัลติดาต้าลิงค์ภายใต้ชั้นเน็ตเวิร์คเลเยอร์ในการค้นหาเส้นทาง

3 สนับสนุนสำหรับรันไทม์คอนฟิกิวเรชัน (Run-time Configulation)เช่น โมดูลของกระบวนการทางเทอร์มินัลจะถูกหลักไปอยู่ในชั้นโปรโตคอลของเน็ตเวอร์คเพื่อที่จะสนับสนุนเน็ตเวอร์ค ลอคอิน UUCP

4 มีความง่ายในการจัดการระบบ สำหรับรวบรวมสถิติและข้อมูลในเหตุการณ์การลอคอินทำให้เกิดประโยชน์ในการจัดการ

โปรโตคอลMS-DOS เสนอรูปแบบมากมายในการปฏิบัติที่ดีที่สุดและการใช้ที่ว่างในสถานะของ MS-DOS

5 ลดความต้องการหน่วยของ MS-DOS ซอร์ฟแวร์และปรับปรุงการปฏิบัติงาน การแปลงรหัสและอัลกอลิทึมเป็นโปรโตคอลและสตรีม

6 อนุญาตไดรว์เวอร์ของโครงข่ายสตาร์แลนเพื่อที่จะทำงานในส่วนความที่ขยายทำการใช้ประโยชน์ของหน่วยความจำเดิม ส่วนเสริมสามารถทำงานร่วมได้กับมาตรฐานอุตสาหกรรม โลตัส อินเทล ไมโครซอร์ฟ4.0 สำหรับหน่วยความจำที่ขยายและสามารถที่ทำงานกับหน่วยความจำที่เพิ่มเข้ามาในบอร์ดเพื่อรูปแบบนั้นๆ

การเชื่อมต่อการทำงานกับโครงข่าย เอทีแอนที สตาร์แลน

(Internetworking with AT&T Star Lan Network)

ความสามารถของสตาร์แลนรูทเตอร์(Starlan Router)สามารถที่จะอธิบายได้ดีที่สุดให้ดูจากการเชื่อมต่อแบบต่างๆของแลน กลไกในการเชื่อมต่อของแลนมี บริดจ์(Bridges), รูทเตอร์(Routers), รีเลย์(Releys) และเกจเวย์(Gageways) ซึ่งแต่ละแบบก็มีความเฉพาะตัวและความสำคัญต่างกันไป ขึ้นอยู่กับโปรโตคอลที่ใช้ ตัวสื่อกลางและความต้องการในการที่จะเชื่อมต่อระหว่างสองจุด แต่ทุกชั้นต้องขึ้นอยู่กับสถาปัตยกรรมของ OSI

สถาปัตยกรรมของโอเอสไอ(OSI Architecture)

ISO โมเดล 7 ชั้น เรียกสถาปัตยกรรมของโอเอสไอ(OSI Architecture)ซึ่งเป็นโปรโตคอลมาตรฐานของการสื่อสารข้อมูลเพื่อลดความยุ่งยากต่างๆ และผู้ใช้ไม่จำเป็นต้องมารู้เกี่ยวกับตัวอุปกรณ์

ชั้นต่ำที่สุดชั้นที่ 1ฟิสิคัลเลเยอร์(Physical Layer)เป็นการพิจารณาเกี่ยวกับการรับและส่งบิทข้อมูลไปที่ตัวกลางหรือจากตัวกลางที่ใช้ในการสื่อสาร

ชั้นที่ 2 ดาต้า ลิงค์ เลย์เยอร์(Data Link Layer)เป็นการส่งแพคเก็จที่ไม่มีการผิดพลาดระดับเฟรมไปสู่ชั้นเนตเวิร์กเลเยอร์

ชั้นที่ 3 เนตเวอร์คเลเยอร์(Network Layer) เป็นรูปแบบการเชื่อมต่อมีแอดเดรสของโครงข่าย การหาเส้นทางในการเชื่อมต่อซึ่งจะต่อตรงหรือมีการเชื่อมต่อผ่านหลายๆจุด

ชั้นที่ 4ทราสปอร์ตเลเยอร์(Transport Layer) เป็นการสื่อสารจากจุดปลายทางถึงจุดปลายทางหรือจากโฮสท์(Host)ไปถึง โฮสท์ เป็นท่อลำเลียงข้อมูลสำหรับ เซสชั่นเลเยอร์(Session Layer)

ชั้นที่ 5เซสชั่นเลเยอร์(Session Layer)เป็นการเชื่อมต่อโครงข่ายสำหรับผู้ใช้ในโครงข่าย ผู้ใช้ต้องการเชื่อมต่อเข้ากับโครงข่ายเพื่อที่จะทำงานบางอย่าง เช่น รับส่งอีเลคทรอนิกส์เมร์ หรือส่งไฟล์ หรือการร้องขอชั้นข้อมูล

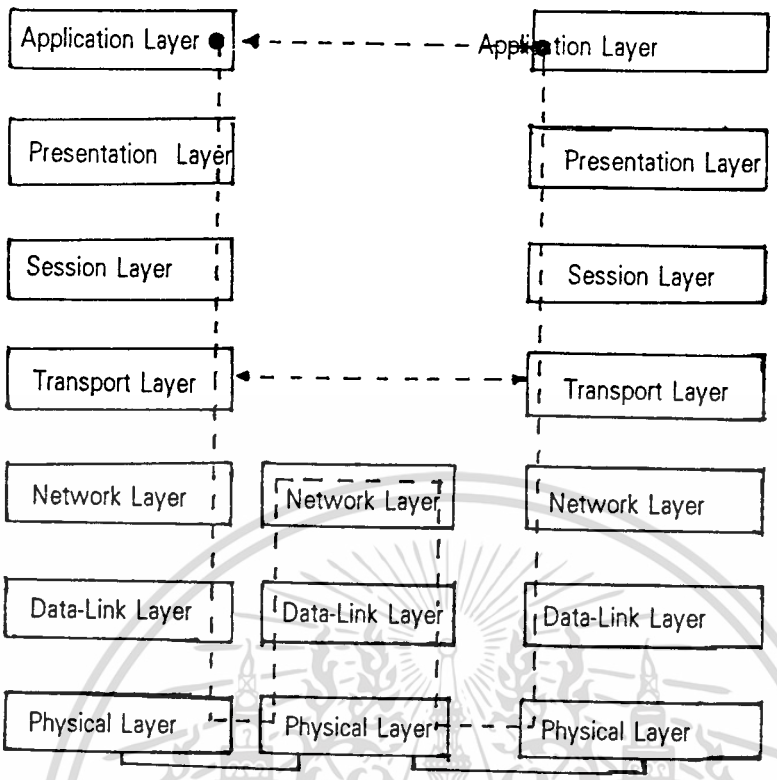
ชนที่ 6 พรีเซนเทชั่น เลเยอร์(Presentation Layer)เป็นการปรับเปลี่ยนข้อมูลของเราเพื่อที่จะใช้ในการส่ง เช่น แปลงเป็น ASCII หรือ EBCDIC

ชั้นที่ 7 แอปลิเคชั่นเลเยอร์(Application layer)ขึ้นอยู่กับรูปแบบที่เราต้องการติดต่อ

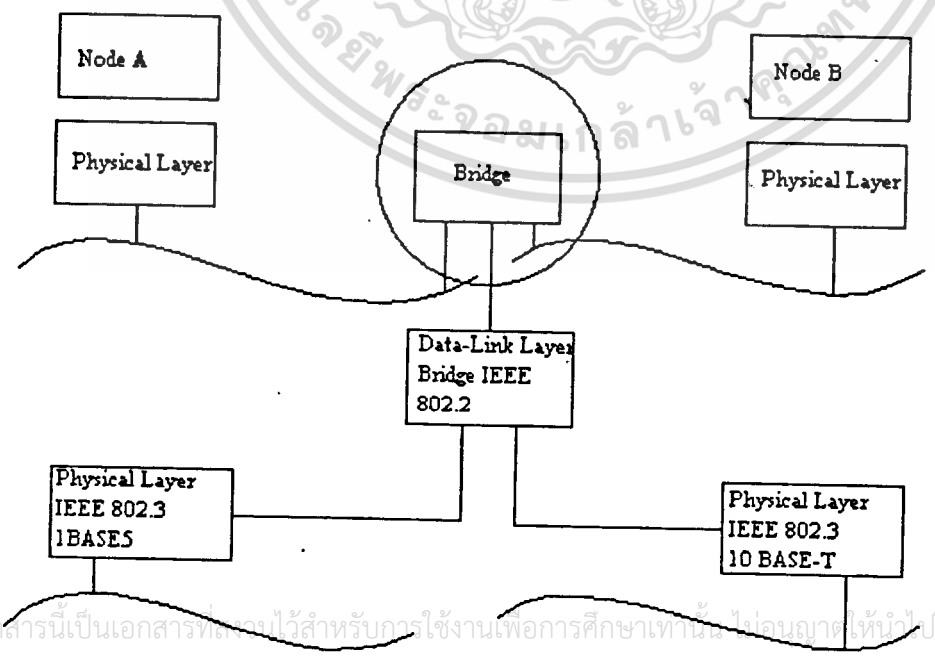
บริดจ์(Bridge)

บริดจ์ จะเชื่อมต่อเข้าในชั้นของดาต้าลิงค์ และในการเชื่อมกันต่างตัวกลางต้องโปร่งใส บริดจ์อย่างง่ายที่สุดเป็นแบบ 2พอร์ท ซึ่งเชื่อมต่อแลน 2ตัวกลาง บริดจ์จะรับทุกเพคเก็จที่ส่งมาและจะเลือกที่จะส่งออกไปใหม่บางตัว ตามที่อยู่ที่หมายปลายทาง บริดจ์เป็นการเชื่อมต่อจากแลนที่มี ตัวกลางต่างกัน ความเร็วต่างกัน เราจะเชื่อมแลนหลายๆระบบ มาเป็นโครงข่ายเดียวกันก็ได้

รูปที่1 สถาปัตยกรรมOSIโปรโตคอล



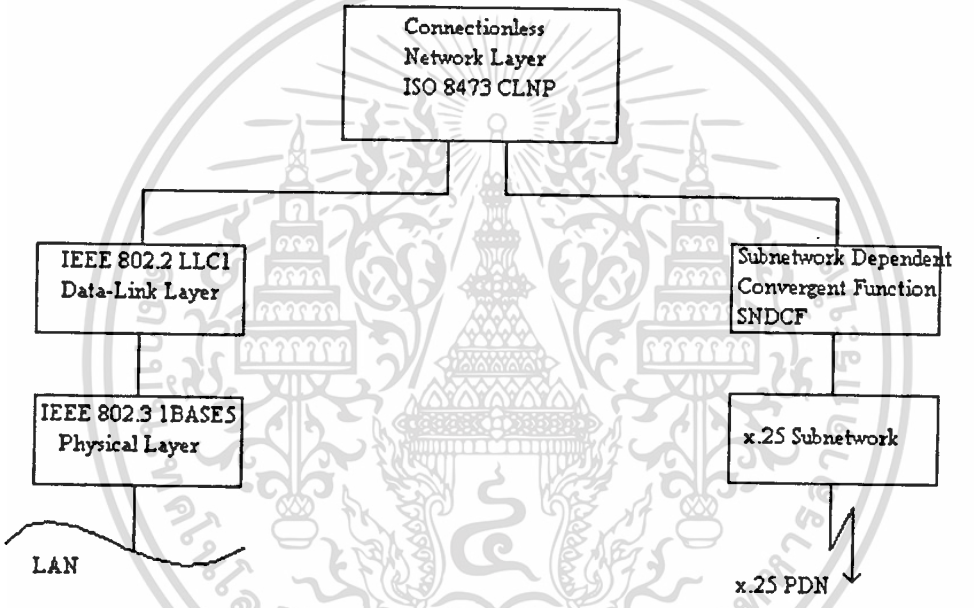
รูปที่ 2 ตัวอย่างโครงสร้างการต่อแบบ Bridge



รูเตอร์(Router)

รูทเตอร์ จะถูกเชื่อมต่อเข้ากับชั้นเน็ตเวิร์ค รูปแบบของรูทเตอร์ แสดงตามรูปที่ 3 ซึ่งชั้นเน็ตเวอร์คจะมีฟังก์ชันรูทติงขึ้นอยู่กับเน็ตเวิร์คแอดเดรส ในชั้นเน็ตเวิร์คข้อมูลที่สำคัญที่สุดคือ เน็ตเวิร์คแอดเดรส ของจุดหมายปลายทาง ดังนั้นเน็ตเวิร์ค เลเยอร์ รูทเตอร์ ใช้ตารางรูทติงของมัน ทำให้สามารถส่งข้อมูลไปถึงปลายทางที่ถูกต้องอาจจะผ่าน หนึ่งหรือหลายๆสถานี

รูทเตอร์สามารถที่จะเชื่อมต่อกับหลายๆโครงข่าย และหลายๆตัวกลางเช่นรูทเตอร์สามารถที่จะเชื่อมต่อกับโครงข่าย 802.3 รูทเตอร์สามารถแทน บริดจ์ได้ มันเป็นประโยชน์ในการที่จะเชื่อมต่อกันภายในโครงข่ายสำหรับเหตุผลในแง่ความปลอดภัย มันไม่จำเป็นต้องเข้าถึงผู้ใช้ทุกคน ซึ่งการเข้าถึงนี้โดยใช้รูทติงเทเบิล

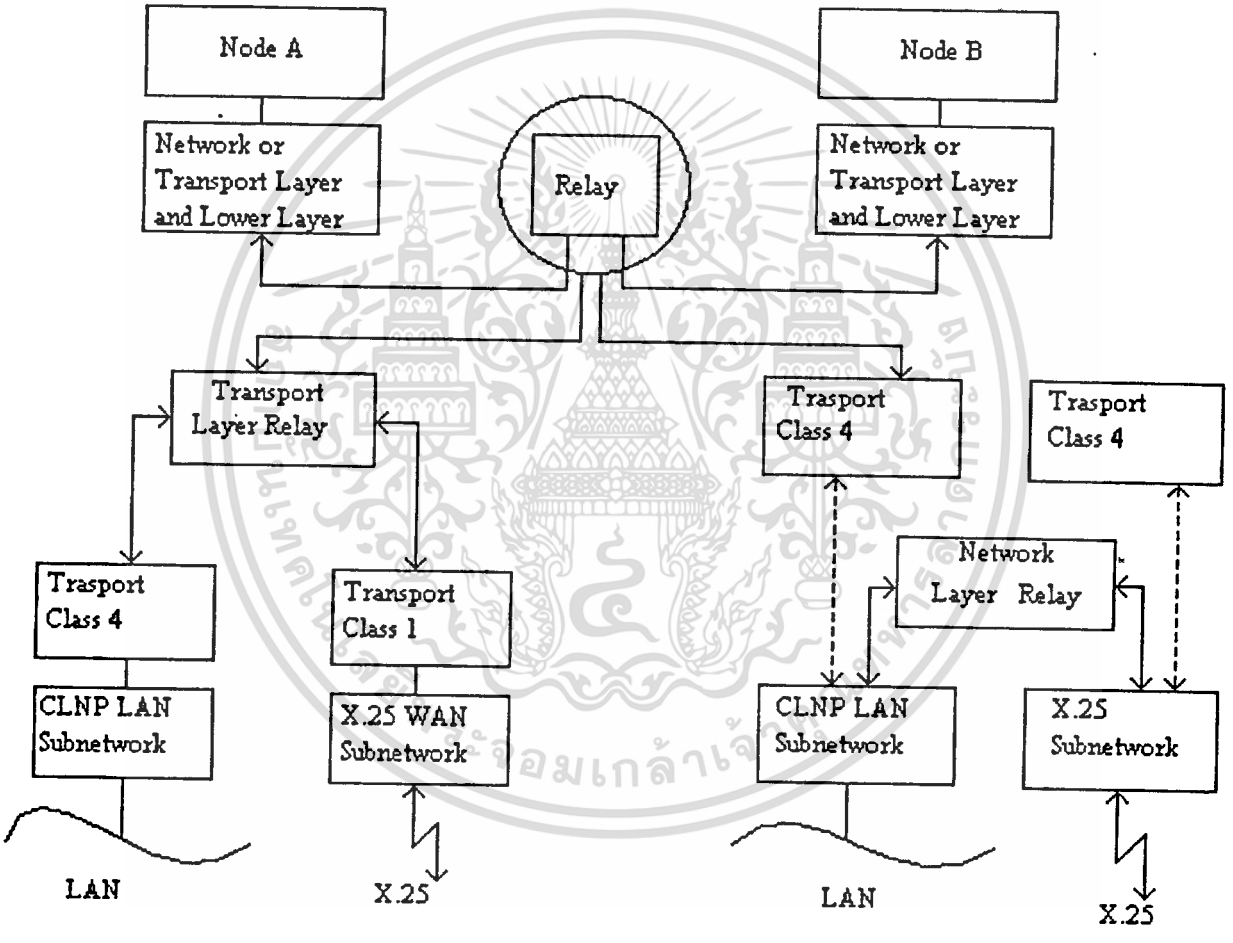


รูปที่ 3 สตาร์แลน X.25รูทเตอร์

รีเลย์(Relays)

รีเลย์ ก็คล้ายๆกับบริดจ์แต่ต่างกันในจุดที่เชื่อมต่อขึ้นกับระดับของรีเลย์เช่น เน็ตเวิร์ค เลเยอร์ รีเลย์ ก็จะเชื่อมต่อเข้ากับชั้นเน็ตเวิร์ค ทรานสปอร์ต เลเยอร์ รีเลย์ ก็จะเชื่อมต่อเข้ากับชั้นทรานสปอร์ต รูปที่ 4 เน็ตเวิร์ค เลเยอร์ รีเลย์ เชื่อมต่อโครงข่าย ซึ่งใช้โปรโตคอล เดียวกับชั้น ทรานสปอร์ต หรือชั้นที่สูงกว่า แต่แตกต่างจากโปรโตคอลของชั้น เน็ตเวิร์ค และชั้นที่ต่ำกว่าเช่น เน็ตเวิร์ค เลเยอร์ รีเลย์ จะต้องเชื่อมต่อเข้ากับชั้นเน็ตเวิร์คของคอมพิวเตอร์ที่มีการใช้ โปรโตคอลจองการไม่มีการเชื่อมต่อ และมีการเชื่อมต่อ

รูปที่ 4 สถาปัตยกรรมของ Relay



ทรานสปอร์ต เลเยอร์ รีเลย์ เชื่อมต่อกับโครงข่ายโดยใช้กลุ่มของโปรโตคอลที่สูงกว่าชั้นทรานสปอร์ต แต่แตกต่างจากกลุ่มของโปรโตคอลที่ชั้นทรานสปอร์ต หรือชั้นที่ต่ำกว่า ตัวอย่างเช่น ทรานสปอร์ต เลเยอร์ รีเลย์ จะเชื่อมต่อกับคอมพิวเตอร์ บนโครงข่ายที่ใช้ OSI ทรานสปอร์ต คลาส1 และคอมพิวเตอร์บนโครงข่ายที่ใช้ OSI ทรานสปอร์ต คลาส4

เนื่องจาก รีเลย์มีฟังก์ชันในการแปลงโปรโตคอลระหว่างโครงข่ายที่ไม่เหมือนกัน รีเลย์ไม่ได้กระจายสำหรับการปฏิบัติการเชื่อมต่อขั้นสูง ดังนั้น รีเลย์ต้องการพื้นที่หรือบัฟเฟอร์จำนวนมาก เพราะว่ามันทำฟังก์ชัน สตอร์และฟอร์เวิร์ด(Store and forward)

เกจเวย์(Gateway)

เกจเวย์ เป็นลักษณะทั่วๆไปของ รีเลย์ซึ่งจัดให้มีการเชื่อมต่อ ในชั้นแอปลิเคชั่นสำหรับระบบที่แตกต่างกัน และมีตัวแปลงโปรโตคอลเช่นในการเชื่อมต่อระหว่างโครงข่าย SNA OSI,TCP และ OSI Network

เกจเวย์มีประสิทธิภาพต่ำกว่ารีเลย์เพราะมีการ แมบปิง(Mapping)ระหว่างโปรโตคอลอันหนึ่งไปยังอีกอันหนึ่ง และในการจัดการกับระบบนั้นทำได้ยากกว่า รีเลย์ เนื่องจากมีความแตกต่างกันมากในโครงข่าย เกจเวย์จะใช้เฉพาะงานเท่านั้น เช่นความต้องการสื่อสารข้ามโครงข่ายที่ต่างกัน

สถาปัตยกรรมของสตาร์แลนรูทเตอร์(Starlan Router Architecture)

ในการเปลี่ยนการเชื่อมต่อของแลนนั้นและกลายมาเป็นสิ่งที่สำคัญของ ISO, X.25 Router ถูกเลือกมาใช้ในการเชื่อมต่อระหว่างแลนกับแวน (WAN)

Starlan X.25 Router เป็นส่วนเสริมในโครงข่ายอยู่บนพื้นฐานคอนเนคชั่นเลสเนตเวอร์เลเยอร์โปรโตคอล (Connectionless Network Layer Protocol)สถาปัตยกรรมแสดงไว้ในรูปที่ 3

The OSI Connectionless Network Layer Protocol จัดให้มีการบริการแบบไม่มีการเชื่อมต่อมันต้องการเพียงโปรโตคอลชั้นมูลฐานแบบไม่มีการเชื่อมต่อ เมื่อโครงข่ายย่อยใช้การเชื่อมต่อ X.25ซับเนตเวอร์ค ดีเพนเดนท์ คอนเวอร์เจนท์ฟังค์ชั่น (A Subnetwork Dependent Convergence Function (SNDCF))ถูกใช้ในการปรองดองความแตกต่างระหว่างชั้นเน็ตเวิร์ค และจัดให้มีโครงข่ายย่อยชั้นมูลฐาน โครงข่ายสตาร์แลน(The Starlan Network Layer)ใช้ร่วมกับ ระบบสิ้นสุด(END System)ไปถึง อินเตอร์มิเดียท์(Intermediate System Protocol)ซึ่งสร้างแอดเดรสแบบลอจิกที่อาจเป็นไปได้และปรับปรุงฟังก์ชันในการหาเส้นทาง

แอดเดรสของเน็ตเวิร์คมีอยู่ 2 แบบ

- ฟิสิคัล(Physical)
- ลอจิคคอล(logical)

Physical address,The Sub Network Point of Attachment (SNPA) Address (หรือ ลิงค์เลเยอร์แอดเดรสเช่น

อีเทอร์เน็ต แอดเดรส) ถึงฝังติดกับเน็ตเวิร์คแอดเดรส ลอจิกแอดเดรส อิสระจาก SNPA แอดเดรส การใช้ลอจิกแอดเดรสโครงข่ายกลายเป็นอิสระจาก SNPA แอดเดรสหรือฟิสิกส์คัลแอดเดรส (Meadia Access Control Address หรือ MAC) ดังนั้นมันจึงเป็นไปได้ที่จะมีความสามารถ เรื่อง มัลติโฮมมิง (Multi-homing)คือ1ลอจิกคัลแอดเดรสแทนหลายฟิสิกส์คัลแอดเดรส แต่ยังคงเป็นแอดเดรสของเน็ตเวิร์คเดียว ในการเปลี่ยนแปลงฮาร์ดแวร์นั้นก็หมายถึง MAC ค่าใหม่ แต่ไม่มีผลกระทบต่อลอจิกคัลแอดเดรสหรือเน็ตเวิร์คแอดเดรส

ลอจิกคัล แอดเดรสชิง มีความยืดหยุ่นในโครงสร้างของแอดเดรสโดยขยายการบริหารและปรับปรุงการค้นหาเส้นทาง

การทำงานของสตาร์แลน X.25รูทเตอร์ (Starlan X.25 Router Operation)

เมื่อสตาร์แลนX.25 ถูกทำให้เป็นโครงร่างขึ้น โครงร่างของเน็ตเวอร์คและข้อมูลในไดเรกทอรีถูกใส่ไว้ในรูทติงเทเบิล ซึ่งข้อมูลเหล่านี้จัดให้รูทเตอร์แต่ละตัวด้วยข้อมูลในการเชื่อมต่อกับรูทเตอร์ตัวอื่นๆเท่าๆกับแอดเดรสของผู้ใช้ปลายทางหรือการบริการซึ่งผู้ใช้ในท้องถิ่นต้องเข้าถึงผ่านรูทเตอร์ แต่ละรูทเตอร์(ปกติจะเรียก Intermidiate System-IS) จะรู้ เอ็นชิสเต็ม(ES)ทุกๆตัวที่อยู่ในโครงข่ายในเวลานั้นซึ่งบรรดาคอมพิวเตอร์ไม่ใช่รูทเตอร์ ทุกๆรูทติงโหนดให้ข่าวสารทุกคอมพิวเตอร์บนโครงข่ายในขณะนั้น โดยใช้ ES-IS โปรโตคอลเหมือนกัน

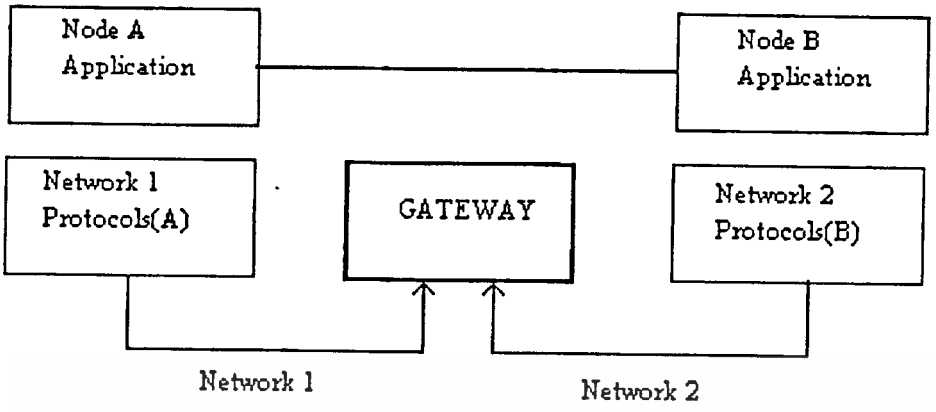
โปรโตคอลบริการชื่อใช้เพื่อที่จะบรรจุเน็ตเวร์คแอดเดรสของปลายทาง ES ถ้าปลายทางES อยู่ในLAN เดียวกัน.แพ็คเกตก็จะถูกส่งไปโดยตรงถ้าไม่ใช่โปรโตคอลบริการชื่อจะจัดเตรียมไว้ให้รูทเตอร์ ในการที่จะจัดการกับแพ็คเกจ รูทเตอร์จะตรวจดูแพ็คเกตที่รับมาเพื่อที่จะดูจุดหมายปลายทางของอีเอส ถ้าเป็นแอดเดรสของมัน มันก็จะส่งผ่านชั้นทรานสปอร์ตขึ้นไป ถ้าเป็นของคนอื่น รูทเตอร์ก็จะใช้แอดเดรสปลายทาง ของอีเอส เพื่อที่จะค้นหาเส้นทาง ในตารางรูทติ้ง และส่งแพ็คเกตไปที่ปลายทางนั้น หรือรูทเตอร์ตัวอื่นๆ

การสื่อสารกันระหว่างคู่ของรูทเตอร์ ถูกสนับสนุนโดย X.25 เวอร์ชวล เซอร์กิต เมื่อ SNDLF รับแพ็คเกตสำหรับจะส่ง มันจะตรวจดูปลายทางของ SNPA แอดเดรสกับวงจรที่มี ถ้ามีวงจร (เส้นทาง) มันก็จะส่งแพ็คเกตนั้นไปบนเส้นทางนั้น มิฉะนั้นมันก็จะสร้างเส้นทางไปถึงปลายทาง แล้วก็ส่งแพ็คเกต SNDLF ก็จะตรวจดูจราจรของเส้นทางนั้นด้วย ถ้าวงจรมันว่างสำหรับช่วงเวลาก่อนการตรวจสอบมันจะไม่ทำการเชื่อมต่อวงจร

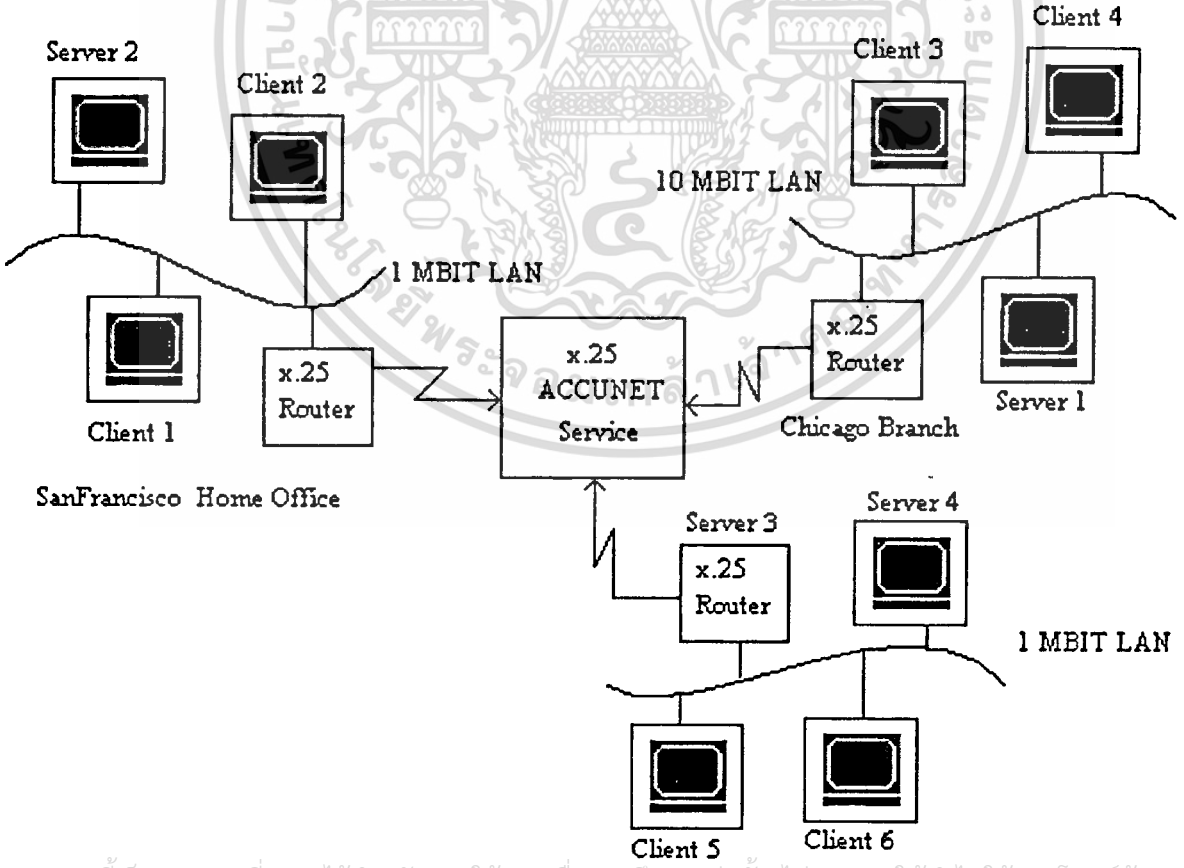
รูปแบบของสตาร์แลนรูทเตอร์ (Starlan Router Feature)

สตาร์แลนรูทเตอร์จะมีไว้สำหรับ AT&T 3B2 และAT&T 6386 เวิร์คสเตชันรันบน ยูนิกซ์ ระบบ5 รุ่น3 คอมพิวเตอร์เหล่านี้ไม่จำเป็นต้องทำเป็นเหมือนรูทเตอร์สามารถใช้เป็นบริการไฟล์และการพิมพ์ การจัดการ

รูปที่ 5 ตัวอย่าง GATEWAY



รูปที่ 6 ตัวอย่างการเชื่อมต่อต่างถิ่น



ระบบและใช้งานทั่วๆไปของยูนิกซ์

ครั้งหนึ่งรูทเตอร์ถูกติดตั้งเข้าไปและทำให้ออนไลน์ คอมพิวเตอร์ทุกเครื่องในโครงข่ายสามารถที่จะเข้าโครงข่ายอื่น การติดตั้งซอร์ฟแวร์จะปรากฏที่โหนดของรูทเตอร์ ปรกติผลิตภัณฑ์ของสตาร์แลนรุ่น3 จะรวม MS-DOS ที่เป็นพื้นฐานของพีซี จะมีทุกอย่างที่จำเป็นต้องใช้ในการสื่อสารกับรูทเตอร์ รูปแบบนี้จะเริ่มทำงานเมื่อรูทเตอร์มีอยู่ในเนตเวอร์ค

แอปลิเคชันหลายๆตัวของ MS-DOS ไม่มีความยุ่งยากในการที่จะใช้กับรูทเตอร์ เนื่องจากรูทเตอร์มีบริการไดเรกทอรีซึ่งปฏิบัติการอย่างเดียวกับการบริการแยกย่อยชื่อ ในการเชื่อมต่อภายในโครงข่าย ถึงแม้ว่ารูทเตอร์จะสนับสนุนข้อมูลทุกๆตัวที่ไหลเข้ามาและออกจากแลนในท้องถิ่นไปสู่โครงข่ายอื่นมันก็ไม่จำเป็นที่จะต้องใช้ในโครงข่ายท้องถิ่นเดียวกันเพราะว่า รูทเตอร์ใช้ในการเชื่อมต่อข้ามโครงข่าย

เทคโนโลยีต่างๆที่มีอยู่ในการสื่อสารข้อมูลสามารถที่จะใช้ต่อกับรูทเตอร์ เช่นต่อกับแลนที่รูทเตอร์บริการ อันนี้รวมทั้งดิจิตอลและอนาลอกไปรเวทไลน์ DDD เซอร์กิตสวิทชิงและการบริการ X.25 แพ็คเกจสวิทชิง รูทเตอร์ถูกใส่เข้าไปใน AT&T 3B2 จะสนับสนุนการเชื่อมต่อ10การเชื่อมต่อ X.25 ที่ความเร็วถึง19.2 กิโลบิท /วินาที สำหรับ 6386 WGS รูทเตอร์จะสนับสนุน 4 การเชื่อมต่อที่ความเร็ว 19.2 กิโลบิท/วินาที

การเชื่อมต่อแลนหลายๆตัวสามารถที่จะทำได้ดังในรูปที่ 6 สาขาทั้งหลายสามารถเชื่อมต่อเข้ากับหน่วยหลักได้ หน่วยหลักก็จะเป็นพวกแหล่งทรัพยากร เช่น ไฟล์เซิร์ฟเวอร์1 ซึ่งสาขาย่อยๆ สามารถที่จะเข้ามาจัดการกับข้อมูลได้ เช่น ลูกค้าของเซิร์ฟเวอร์ จะต่ออยู่ที่สาขาย่อย และเส้นทางการเข้าถึงผ่านรูทเตอร์

เส้นทางการเข้าถึงที่เพิ่มขึ้นอนุญาตให้ลูกค้าที่สาขาย่อยหนึ่งไปใช้เซิร์ฟเวอร์อันอื่นได้ ซึ่งเส้นทางนี้ไม่ต้องเห็นโดยลูกค้า แต่รูทเตอร์จะหาเส้นทางเอาเองโดยอัตโนมัติ โดยจะถูกติดตั้งโดยการบริหารโครงข่าย

แลนกลายมาเป็นบทบาทสำคัญเพิ่มขึ้นในหน่วยของธุรกิจ ซึ่งมันอนุญาตให้ผู้ใช้หลายคนแบ่งข้อมูลและแหล่งทรัพยากรกันใช้ การเชื่อมต่อระหว่างโครงข่ายที่เพิ่มเข้าไปสามารถที่จะหลอมธุรกิจในหลายๆที่เข้าด้วยกัน

AT&T สตาร์แลนเป็นผลิตภัณฑ์ที่เสนอเทคโนโลยีในการเชื่อมโครงข่ายเป็นแวนที่คุณพอใจ

```
#include <conio.h>          /* โปรแกรม control ที่ center */
#include <stdio.h>
#include <dos.h>
#include "process.h"
#include "string.h"
#define ENTER  13
#define SYN    22
#define COM1   0x3f8
#define COM2   0x2f8
#define COM3   0x3e8
#define COM4   0x2e8
int x,y;
int x1,y1,x2,y2;

 transfer(int com)
{
    char c;
    clrscr();
    outportb(com+3,0x80);
    outportb(com,0x60);
    outportb(com+1,0x00);
    outportb(com+3,0x03);
    printf("\nWhat is your requirement.(send[s] /receive[r] file)");
    c=getche();

     switch(c)
     {
            case's':  send_f(com); break;
            case'r':  rec_f(com);  break;
            default:  printf("Press only s or r");
                      exit(0);
     }
}
 send_f(int com)
{
  FILE *fp;
  int p,j;
  char k;
  unsigned char name[10];
```

```
union {
        char x[2];
        unsigned int t;
        }ct;

 printf("\nEnter file's name which you want to send.");
 gets(name);
 j=strlen(name);
 while(inportb(com)!='.')
 {
   outportb(com,SYN);
 }
 printf("\nsyn\n");
 outportb(com,j);/*******send size of file's name**********/
 wait(com);

 for(p=0;p<j;p++)    /*****send file's name******/
  {
    outportb(com,name[p]);
    wait(com);
  }

fp=fopen(name,"rb");
if(fp==NULL)
  {
    printf("Can't open file");
    exit(0);
  }

ct.t=filesize(fp);
printf("\nfilesize=%d\n",ct.t);
outportb(com,ct.x[0]);
wait(com);
outportb(com,ct.x[1]);
wait(com);
for(p=1;p<=ct.t;p++)
{
    k=getc(fp);
    outportb(com,k);
```

```
        printf("%c",k);
        wait(com);
    }
    outportb(com,27); /*****use for tell center to give up connecting*****/
    fclose(fp);
}
rec_f(int com)
{
    FILE *fp;
    int p,j,n;
    char c;
     unsigned char name[10]={"          "};
    union {
            char x[2];
            unsigned int c;
            }ct;
    printf("\nplease wait\n");
    outportb(com,SYN);

    while(inportb(com)!=SYN);  /*********syn**********/
    outportb(com,'.');
    printf("\nsyn");
    while(inportb(com)==SYN);
    j=inportb(com);
    printf("\nj=%d\n",j);
    outportb(com,'.');
    for(p=0;p<j;p++)
    {   while(check(com)==0);
            name[p]=inportb(com);
            outportb(com,'.');
            printf("%c",name[p]);
    }
    name[p]=NULL;
    remove(name);
    fp=fopen(name,"wb");
    if(fp==NULL)
    {  printf("\nCan't open file");
       exit(0);
    }
```

```
    while(check(com)==0);
    ct.x[0]=inportb(com);
    outportb(com,'.');
    while(check(com)==0);
    ct.x[1]=inportb(com);
    outportb(com,'.');
    printf("\nfilesize=%d\n",ct.c);
    for(n=1;n<=ct.c;n++)
    {
        while(check(com)==0);
        c=inportb(com);
        printf("%c",c);
        putc(c,fp);
        outportb(com,'.');
    }
    fclose(fp);
}
    wait(int com)
{
        while(check(com)==0);
        while(inportb(com)!='.');
}
 filesize(FILE  *fp)
{
  int i=0;
  do{
    getc(fp);
    i++;
  }while(!feof(fp));
  rewind(fp);
  return i-1;
}

check(int com)
{
      int a,b,flag;
      flag = 0;
      a  = inportb(com+5);
      b = a&2;
```

```
        a  = a&1;
        if(b==1) printf("OVERRUN ERROR!");
        if(a==0) flag=0;
        else flag = 1;
        return flag;
}
 chatmode(int com)
{
int  ch;
int  a,flag;
    flag = 0;
    x1=x2=y1=y2=1;
    outportb(com+3,0x80);
    outportb(com,0x60);
    outportb(com+1,0x00);
    outportb(com+3,0x03);
    clrscr();
    buildbox(1,1,79,10);
    gotoxy(34,1);
    textcolor(5);
    textbackground(15);
       cprintf("LOCAL");
       normvideo();

    buildbox(1,12,79,21);
    gotoxy(34,12);
    textcolor(5);
    textbackground(15);
       cprintf("REMOTE");
       normvideo();

    gotoxy(1,25);
    textcolor(1);
    textbackground(10);
       cprintf("ESC->EXIT");
       normvideo();

    for(;;) {
            if(kbhit()) {
```

```
                window(3,3,77,9);
                ch=getch();
                switch(ch) {
                        case 13 : if(y1==7) {
                                        movetext (3,4,77,9,3,3);
                                        x1=1;
                                        y1=7;
                                        gotoxy(x1,y1);
                                        delline();
                                        }
                                if(y1<7) {
                                        y1+=1;
                                        x1=1;
                                        }
                                gotoxy(x1,y1);
                                outportb(com,13);
                                break;
                        case 27 : break;
                        default : send(com,ch,x1,y1);
                        }
                if(ch==27) break;
                }
        else {
            a=inportb(com+5);
                    a=a&1;
            if(a==0)  flag=0;
            else flag=1;
            if(flag) rec(com,x2,y2);
           }
        }
}

send(int com,int c,int x,int y)
{
  gotoxy(x,y);
  cprintf("%c",c);
  outportb(com,c);
  x+=1;
  if(x==75)  {
```

```
                if(y==7){
                        movetext(3,4,77,9,3,3);
                        x=1;
                        y=7;
                        delline();
                    }
                else {
                      y+=1;
                      x=1;
                      }
                gotoxy(x,y);
               }
   x1=x;
   y1=y;
}

rec(int com,int x,int y)
{
   int ch;
   window(3,14,77,20);
   ch=inportb(com);
   switch(ch) {

              case 13 : if(y<7){
                                x=1;
                                y+=1;
                                gotoxy(x,y);
                               }
                        if(y==7) {
                                  movetext(3,15,77,20,3,14);
                                  x=1;
                                  y=7;
                                  gotoxy(x,y);
                                  delline();
                                 }
                        break;
                default : gotoxy(x,y);
                          cprintf("%c",ch);
                          x+=1;
```

```
                    if(x==75) {
                        if(y<7) {
                                x=1;
                                y+=1;
                                gotoxy(x,y);
                                }
                        if(y==7) {
                                movetext(3,15,77,20,3,14);
                                x=1;
                                y=7;
                                gotoxy(x,y);
                                delline();
                                }
                            }
                    }
  x2=x;
  y2=y;
 }

buildbox(int l,int t,int r,int b)
{
  int i;
  gotoxy(l,t);
  putch(218);
  gotoxy(r,t);
  putch(191);
  for(i=l+1;i<=r-1;i++)
  {
    gotoxy(i,t);
    putch(196);
    gotoxy(i,b);
    putch(196);
  }
  gotoxy(l,b);
  putch(192);
  gotoxy(r,b);
  putch(217);
  for(i=t+1;i<=b-1;i++)
  {
```

```
                outportb(com_b,a); }
            }while((inportb(com_a)!=27)||(inportb(com_b)!=27));
            break;
        }
        clrscr();               /****before back to main loop*****/
        printf("\nCenter's  waiting\n");

}

setport(int com)
{
        outportb(com+3,0x80);
        outportb(com,0x60);
        outportb(com+1,0x00);
        outportb(com+3,0x03);
}

void main()
{
        clrscr();
        setport(COM1);
        setport(COM2);
        setport(COM3);
        setport(COM4);
  printf("\nCenter's  waiting\n");
 for(;;)
  {
     if(check(COM1))
         {    switch(inportb(COM1))
                  {
                   case '0': cdecide(COM1); break;
                   case '2': connect(COM1,COM2);  break;
                   case '3': connect(COM1,COM3);  break;
                   case '4': connect(COM1,COM4);  break;
                  }

         }
     if(check(COM2))
         {    switch(inportb(COM2))
```

```
            {
                case '0': cdecide(COM2);break;
                case '1': connect(COM2,COM1); break;
                case '3': connect(COM2,COM3);break;
                case '4': connect(COM2,COM4); break;
            }
        }
        if(check(COM3))
        { switch(inportb(COM3))
          {
                case '0': cdecide(COM3);break;
                case '1': connect(COM3,COM1); break;
                case '2': connect(COM3,COM2); break;
                case '4': connect(COM3,COM4); break;
          }
        }
    if(check(COM4))
        { switch(inportb(COM4))
          {
                case '0': cdecide(COM4); break;
                case '1': connect(COM4,COM1);break;
                case '2': connect(COM4,COM2);break;
                case '3': connect(COM4,COM3);break;
          }
        }
  }
}
```

```
#include <conio.h>                    /* โปรแกรมรับส่งข้อมูลของเทอร์มินัล */
#include <stdio.h>
#include <dos.h>
#include <process.h>
#include <string.h>
#define ENTER  13
#define SYN    22
#define COM    0x2f8
#define ACK    6
#define NAK    21
int x1,y1,x2,y2;
 transfer(int com)
{
    char c;
    clrscr();
    outportb(com+3,0x80);
    outportb(com,0x60);
    outportb(com+1,0x00);
    outportb(com+3,0x03);

    printf("\nWhat is your requirement.(send[s] /receive[r] file)");
    c=tolower(getche());
    do{
      outportb(com,c);     /*  send  request(send or receive) to  center*/
    }while(inportb(com)!=ACK && inportb(com)!=NAK);
    if(inportb(com)==NAK)
      {  printf(" Sorry ! you are denied ");
            outportb(com,27);/* tell center finish  switching*/
            exit(1);
      }
     switch(c)
     {
            case's':  send_f(com); break;
            case'r':  rec_f(com);  break;
            default:  printf("Press only s or r");
                      exit(0);
     }
}
send_f(int com)
```

```
{
  FILE *fp;
  int p,j;
  char k;
  unsigned char name[10];
  union {
          char x[2];
          unsigned int t;
          }ct;

  printf("\nEnter file's name which you want to send.");
  scanf("%s",name);
  j=strlen(name);
  while(inportb(com)!='.')
  {
    outportb(com,SYN);
  }
  printf("\nsyn\n");
  outportb(com,j);/*******send size of file's name**********/
  wait(com);
  for(p=0;p<j;p++)    /*****send file's name******/
   {
     outportb(com,name[p]);
     wait(com);
   }
 fp=fopen(name,"rb");
 if(fp==NULL)
  {
    printf("Can't open file");
    exit(0);
  }
 ct.t=filesize(fp);
 printf("\nfilesize=%d\n",ct.t);
 outportb(com,ct.x[0]);
 wait(com);
 outportb(com,ct.x[1]);
 wait(com);
 for(p=1;p<=ct.t;p++)
 {
```

```
        k=getc(fp);
        outportb(com,k);
        printf("%c",k);
        wait(com);
    }
    fclose(fp);
}
rec_f(int com)
{
    FILE *fp;
    int p,j,n;
    char c;
     unsigned char name[10]={"          "};
    union {
            char x[2];
            unsigned int c;
            }ct;
    printf("\nplease wait\n");
    outportb(com,SYN);
    while(inportb(com)!=SYN);  /*********syn***********/
    outportb(com,'.');
    printf("\nsyn");
    while(inportb(com)==SYN);
    j=inportb(com);
    printf("\nj=%d\n",j);
    outportb(com,'.');
    for(p=0;p<j;p++)
    {   while(check(com)==0);
            name[p]=inportb(com);
            outportb(com,'.');
            printf("%c",name[p]);
    }
    name[p]=NULL;
    remove(name);
    fp=fopen(name,"wb");
    if(fp==NULL)
    {  printf("\nCan't open file");
        exit(0);
    }
```

```
  while(check(com)==0);
  ct.x[0]=inportb(com);
  outportb(com,'.');
  while(check(com)==0);
  ct.x[1]=inportb(com);
  outportb(com,'.');
  printf("\nfilesize=%d\n",ct.c);
  for(n=1;n<=ct.c;n++)
  {
     while(check(com)==0);
     c=inportb(com);
     printf("%c",c);
     putc(c,fp);
     outportb(com,'.');
  }
  fclose(fp);
}
  wait(int com)
{
     while(check(com)==0);
     while(inportb(com)!='.');

}
 filesize(FILE  *fp)
{
  int i=0;
  do{
    getc(fp);
    i++;
  }while(!feof(fp));
  rewind(fp);
  return i-1;
}
int check(int com)
{
     int a,flag;
     flag = 0;
     a = inportb(com+5);
     a = a&1;
```

```
    if(a==0) flag=0;
    else flag = 1;
    return flag;
}
 chatmode(int com)
{
int  ch;
int  a,flag;
  flag = 0;
  x1=x2=y1=y2=1;
  outportb(com+3,0x80);
  outportb(com,0x60);
  outportb(com+1,0x00);
  outportb(com+3,0x03);
  clrscr();
  buildbox(1,1,79,10);
  gotoxy(34,1);
  textcolor(5);
  textbackground(15);
    cprintf("LOCAL");
    normvideo();

  buildbox(1,12,79,21);
  gotoxy(34,12);
  textcolor(5);
  textbackground(15);
    cprintf("REMOTE");
    normvideo();

  gotoxy(1,25);
  textcolor(1);
  textbackground(10);
    cprintf("ESC->EXIT");
    normvideo();

  for(;;) {
        if(kbhit())  {
                window(3,3,77,9);
                ch=getch();
```

```
                switch(ch) {
                        case 13 : if(y1==7) {
                                        movetext (3,4,77,9,3,3);
                                        x1=1;
                                        y1=7;
                                        gotoxy(x1,y1);
                                        delline();
                                       }
                                  if(y1<7) {
                                        y1+=1;
                                        x1=1;
                                        }
                                  gotoxy(x1,y1);
                                  outportb(com,13);
                                  break;
                        case 27 : exit(0);
                        default : send(com,ch,x1,y1);
                        }
                }
        else {
                a=inportb(com+5);
                        a=a&1;
                if(a==0)  flag=0;
                else flag=1;
                if(flag) rec(com,x2,y2);
               }
        }
}

send(int com,int c,int x,int y)
{
   gotoxy(x,y);
   cprintf("%c",c);
   outportb(com,c);
   x+=1;
   if(x==75)  {
                if(y==7){
                        movetext(3,4,77,9,3,3);
                        x=1;
```

```
                    y=7;
                    delline();
                  }
                else {
                    y+=1;
                    x=1;
                    }
                gotoxy(x,y);
               }
   x1=x;
   y1=y;
}

rec(int com,int x,int y)
{
   int ch;
   window(3,14,77,20);
   ch=inportb(com);
   switch(ch) {

              case 13 : if(y<7){
                          x=1;
                          y+=1;
                          gotoxy(x,y);
                          }
                        if(y==7) {
                          movetext(3,15,77,20,3,14);
                          x=1;
                          y=7;
                          gotoxy(x,y);
                          delline();
                         }
                        break;
                default : gotoxy(x,y);
                        cprintf("%c",ch);
                        x+=1;
                        if(x==75) {
                                if(y<7) {
                                        x=1;
```

```
                                        y+=1;
                                        gotoxy(x,y);
                                        }
                                if(y==7) {
                                         movetext(3,15,77,20,3,14);
                                         x=1;
                                         y=7;
                                         gotoxy(x,y);
                                         delline();
                                        }
                                    }
                            }
   x2=x;
   y2=y;
 }

buildbox(int l,int t,int r,int b)
{
  int i;
  gotoxy(l,t);
  putch(218);
  gotoxy(r,t);
  putch(191);
  for(i=l+1;i<=r-1;i++)
  {
    gotoxy(i,t);
    putch(196);
    gotoxy(i,b);
    putch(196);
  }
  gotoxy(l,b);
  putch(192);
  gotoxy(r,b);
  putch(217);
  for(i=t+1;i<=b-1;i++)
  {
    gotoxy(l,i);
    putch(179);
    gotoxy(r,i);
```

```
    putch(179);
  }
}
void main()
{
   char m,temp,temp1;
    int temp2;

    clrscr();
    outportb(COM+3,0x80);
    outportb(COM,0x60);
    outportb(COM+1,0x00);
    outportb(COM+3,0x03);
    printf("\n\nEnter terminal's number which you are—> ");
    temp = getchar();
    temp1=temp;
    printf("\nChoose terminal which you want to connect");
    printf("\n      center      press  [0]  ");
    if(temp=='4') temp='0';
    temp++;
    printf("\n      terminal%c   press  [%c]  ",temp,temp);
    if(temp=='4') temp='0';
    temp++;
    printf("\n      terminal%c   press  [%c]  ",temp,temp);
    if(temp=='4') temp='0';
    temp++;
    printf("\n      terminal%c   press  [%c]  ",temp,temp);

    m = getche();
    if(m!=temp1) {
          switch(m) {
                              case '0': outportb(COM,'0');
                                        break;
                              case '1': outportb(COM,'1');
                                        break;
                              case '2': outportb(COM,'2');
                                        break;
                              case '3': outportb(COM,'3');
                                        break;
```

```
                    case '4': outportb(COM,'4');
                                        break;
                    default : exit(0);
            }
    } else {  printf("\nYou can't connect yourself."); exit(0);}
  printf(" \nchoose transfer file [t] or chatmode [c]\n");
 switch(getch())
 {   case 't':  outportb(COM,'t');
                    transfer(COM);break;

     case 'c': outportb(COM,'c');
                    chatmode(COM);break;
 }
outportb(COM,NULL);
}
```

```
#include "stdio.h"                    /* โปรแกรมฝังตัวที่เทอร์มินัลปลายทาง */
#include "dos.h"
#include "bios.h"
#include "conio.h"
#include "stdlib.h"
#include "string.h"
#include "mem.h"

#define  COM2        0x2f8
#define  INDOS       0x34
#define  CRIT_ERR    0x5d06
#define  ISR         0x0b
#define videoram ((int far *)0xb8000000)
#define  DC1         17
#define  DC2         18
#define  ACK         6
#define  NAK         21

void InitIntDos(void);
void clrbuf(void);
void enb8259(void);
void disenb8259(void);
void far interrupt tsr();
void far interrupt (*old_isr)();
void save_video(int,int,int,int,unsigned int *);
void restore_video(int,int,int,int,unsigned int *);
void display(char *menu[],int,int);
void write_string(char *,int,int);
void goto_xy(int,int);
void cls(void);
void get_ans(int,int);
void popup(char *ch[]);
void init_port(void);
void com_cleanup( void );
void set_cur(int,int);
void read_cur(int *,int *);
void clr_win( int x0, int y0, int x1, int y1 );
int DosBusy(void);
int Int28DosBusy(void);
```

```
int busy(void);
char far *indosp=0;
char far *crit_errp=0;
int row,col;
char *ch1[]={" Now! There is someone who want to communicate with you by transfer file.",
                        " Do you want to connect with  him ?",
                        " If you want please press 'y' and finish your work.",
                        " Go to Dosprompt and run program SR_FILE.exe.",
                        " If you don't want to connect please press 'n'.",
                        };
char *ch2[]={"Now! There is someone who want to communicate with you by chatmode.",
                        "Do you want to connect with him ?",
                        "If you want please press 'y' and finish your work.",
                        "Go to Dosprompt and run program Chatmode.exe.",
                        "If you don't want to connect please press 'n'.",
                        };
unsigned int videosave[80*10];//=(unsigned int *) malloc((80) * (25));
void main()
{
  static unsigned  keep_size;
//  InitIntDos();        /* get Intdos flag and crittical error flag */
  old_isr=getvect(ISR);    /* move interrupt vector 0b */
  disable();
  init_port();         /*set 8250 I 1200 bit/sec,data 8 bit,1 stop bit */
  setvect(ISR,tsr);
  enb8259();           /* enable 8250 */
  clrbuf();           /*clear buffer com2 */
  enable();
  printf("Program is stayed resident. \n");
  keep_size= _SS+(_SP/16)-_psp+50;
  keep(0,keep_size);
}

int busy()
{
   int flag;
   flag=(DosBusy() || Int28DosBusy());
   if(!flag)
        return 0;
```

```
    else
            return 1;
}

void  enb8259()
{
    outportb(0x21,(inportb(0x21) & 0xf7));
}

void  disenb8259()
{
    outportb(0x21,(inportb(0x21) | 0x08));
}

void clrbuf()
{
  int c;

   c=inportb(COM2+2);    /* clear interrupt idenfication register */
   c=inportb(COM2+5);  /* clear line status register */
   c=inportb(COM2+6);   /* clear modem status register */
   c=inportb(COM2);        /* clear buffer com2 */
}

void InitIntDos()
{
    union  REGS regs;
    struct SREGS sregs;
    regs.h.ah=INDOS;
    intdosx(&regs,&regs,&sregs);  /*pointer to flag is returned in ES:BX */
    indosp=MK_FP(sregs.es,regs.x.bx);   /* get address Intdos flag  */

    if(_osmajor<3)            /* version Dos less than 3 */
          crit_errp=indosp+1;      /* flag is one byte after Indos */
    else  if(_osmajor==3 && _osminor==0)
                      crit_errp=indosp-1;      /* flag is one byte before */
                  else {
                                regs.x.ax=CRIT_ERR;
                                intdosx(&regs,&regs,&sregs);
```

```
                                /* pointer to flag is returned in DS:SI */
                                crit_errp=MK_FP(sregs.es,regs.x.bx);
                               }  /*get address crittical error flag */
}

int DosBusy(void)       /*This function will non-zero if DOS is busy */
{
    if(indosp && crit_errp)
            return(*crit_errp || *indosp);
    else
            return 0xffff;   /*return dos busy if flags are not set */
}
/*  This function will return non-zero if the Intdos flag is >1 or
    the critical error flag is non zero. To be used inside of an
    INT 28 loop. Note that inside INT 28, Intdos==1 is normal, and
    indicates DOS is not busy ; Intdos >1 inside INT 28 means it is. */
int  Int28DosBusy(void)
{
    if(indosp && crit_errp)
            return(*crit_errp || (*indosp > 1));
    else
            return  0xffff;  /* return dos busy if flag are not set */
}

void far interrupt tsr()
{
  int a,flag;
  char c;
 c=inportb(COM2+2)&7;      /* check already recieve interrupt */
  if (c == 4)
  {
          c=inportb(COM2);
          if( c==DC1){
 //                 if (!busy())   /* check Dos busy or not */
                    popup(ch1);
            }
          if(c==DC2)  popup(ch2);
  }
  outport(0x20,0x20);  /* end of interrupt */
```

```
}

void popup(menu)
char *menu[];
{
   char *keys;
   int x,y;
   int  i,len;
   x=0;
   y=0;

   read_cur(&row,&col);          /* read position cursor */
   save_video(0,0,79,9,videosave);     /* save old screen */
   clr_win(0,0,79,9);          /* clear your old screen */
   x=0;
   y=0;
   display(menu,x+1,y+1);     /* display menu */
   x=2;
   y=7;
   get_ans(x,y+1);
   restore_video(0,0,79,9,videosave);     /*  restore your old screen */
   set_cur(row,col);                /* set position cursor  */
}

void display(menu,x,y)
char *menu[];
int x,y;
{
  int i;
          for(i=0;i<5;i++)
          write_string(menu[i],x,y+i);
}

void clr_win( int x0, int y0, int x1, int y1 )
{
 int far *vram,wd;

 vram = videoram+y0*80+x0;
 wd = (x1-x0+1)*2;
```

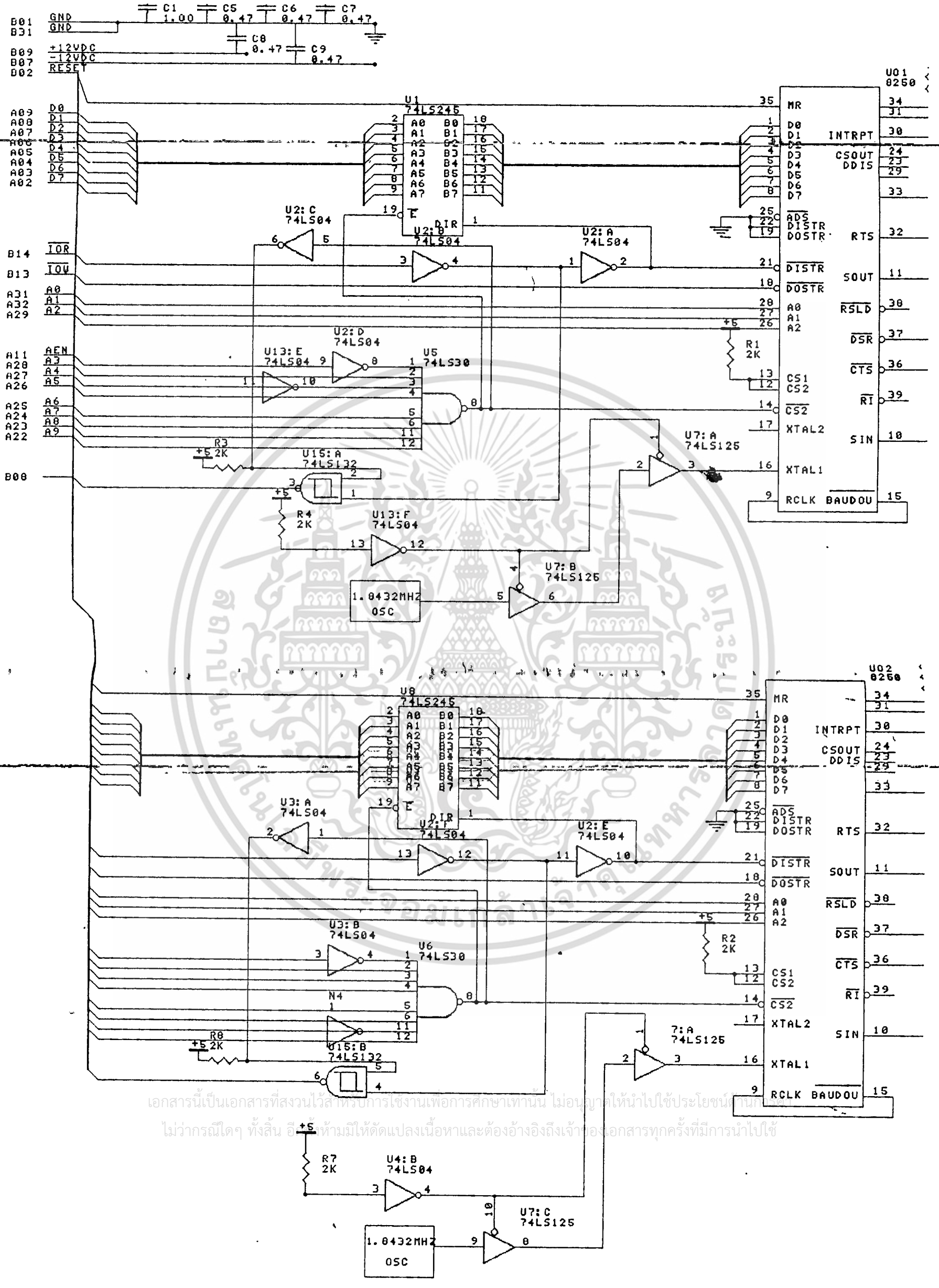
```
 for (;y0<=y1;y0++,vram+=80)
   _fmemset(vram,0x20,wd);
}

void save_video(startx,starty,endx,endy,buf_ptr)
int startx,endx,starty,endy;
unsigned  int *buf_ptr;
{
 int far *vram,wd;

 vram = videoram+starty*80+startx;
 wd = (endx-startx+1)*2;
 for (;starty<=endy;starty++,vram+=80,buf_ptr+=80)
 {
   _fmemcpy(buf_ptr,vram,wd);
 }
}

void get_ans(x,y)
int x,y;
{
char *q[]={ "Now! already. You can go to run nal.exe. ",
            "Ok! You can continue your work. ",
          };
char  key;

 while(bioskey(1)==0);
 key=bioskey(0);
 goto_xy(x,y);
 switch(key) {
 case 'y' : outportb(COM2,ACK);
   write_string(q[0],x,y);
   while(!bioskey(1));

   break;
case 'n' : outportb(COM2,NAK);
   write_string(q[1],x,y);
   while(!bioskey(1));
   break;
```

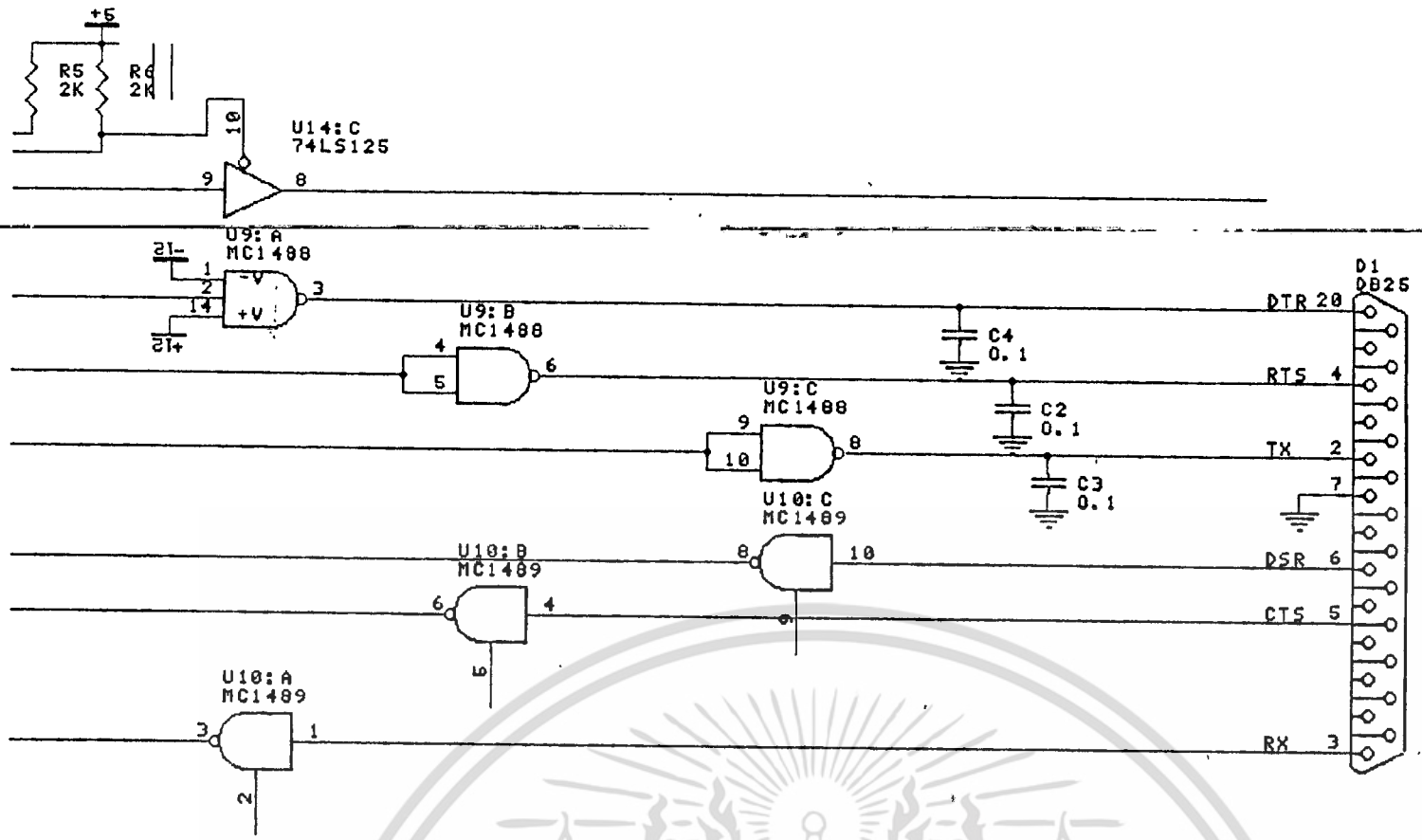
```
 default :  break;
 }
}

void restore_video(startx,starty,endx,endy,buf_ptr)
int  startx,endx,starty,endy;
unsigned  int *buf_ptr;
{
 int far *vram,wd;

 vram = videoram+starty*80+startx;
 wd = (endx-startx+1)*2;
 for (;starty<=endy;starty++,vram+=80,buf_ptr+=80)
   _fmemcpy(vram,buf_ptr,wd);
}

void write_string(p,x,y)
char *p;
int  x,y;
{
  int i,l;
  union  REGS r;

  l=strlen(p);
          for(i=x;i<=l;i++)
          {
           goto_xy (i,y);
// r.h.ah=0x0a;
// r.h.bh=0;
// r.x.cx=1;
// r.h.al=*p++;
// int86(0x10,&r,&r);
            asm {
              mov bx,word ptr p
              mov al,byte ptr [bx]
              mov ah,0x0a
              mov bh,0;
              mov cx,1
              int 0x10
```

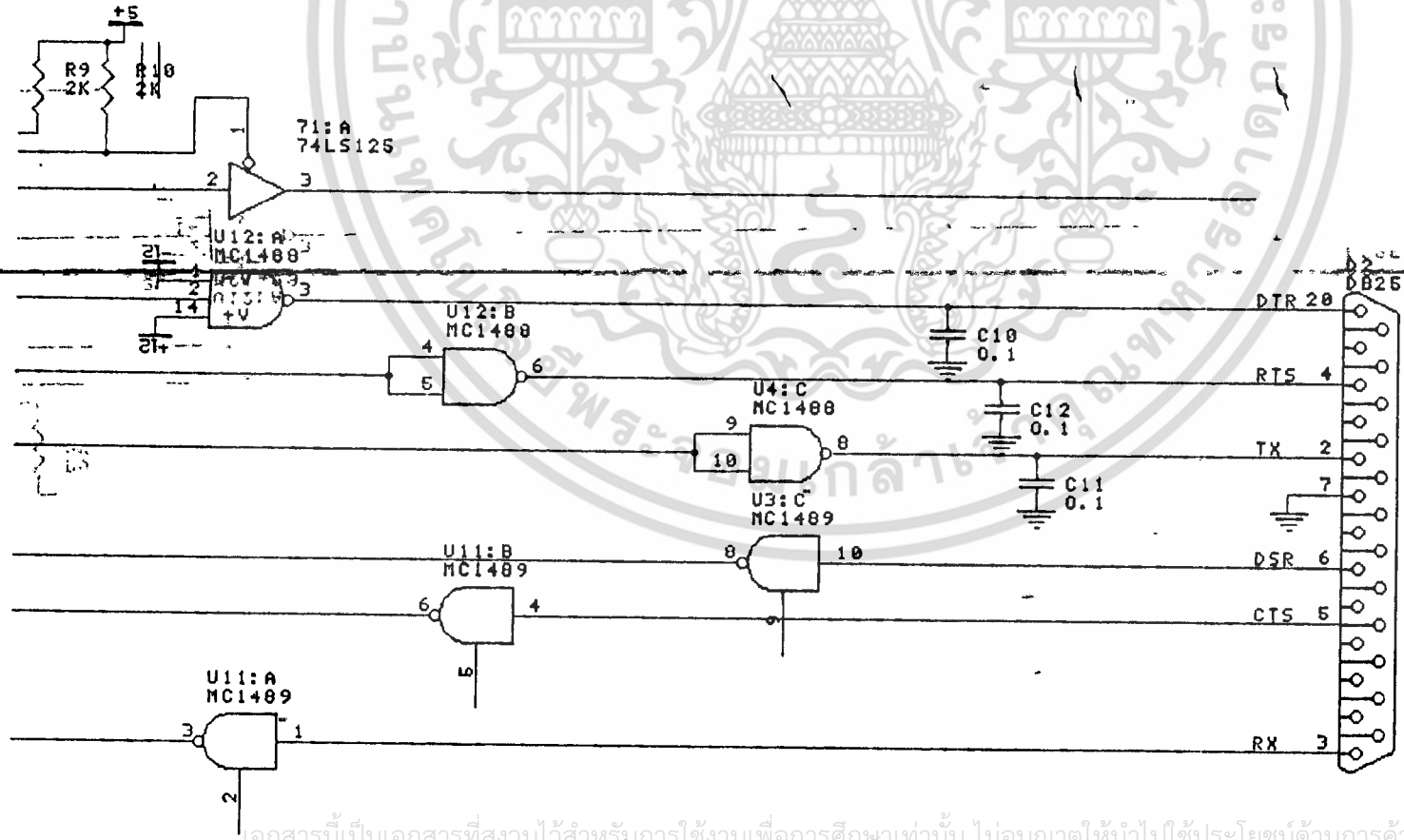
```
            }
            p++;
          }
 }

 void  cls()
 {
   union  REGS r;
   r.h.ah=6;
   r.h.al=0;
   r.h.ch=0;
   r.h.cl=0;
   r.h.dh=24;
   r.h.dl=79;
   r.h.bh=7;

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

 void  goto_xy(x,y)
 int x,y;
 {
 //   union  REGS r;
 //   r.h.ah=2;
 //   r.h.dl=x;
 //   r.h.dh=y;
 //   r.h.bh=3;

 //   int86(0x10,&r,&r);
    asm {
            mov ah,2
            mov bh,0
            mov dh,byte ptr y
            mov dl,byte ptr x
            int 0x10
    }
 }

 void init_port()
```

```
{
   bioscom(0,0x03l0x00l0x00l0x80,1);  /* set COM2 to 1200,8,n,1 */
   outportb(COM2+4,0x0c);                  /* set OUT2=1  */
   outportb(COM2+1,0x01);
}

void read_cur(int *x,int *y)
{
//  union REGS r;
//  r.h.ah=3;
//  r.h.bh=3;
//  int86(0x10,&r,&r);
//  *x=r.h.dl;
//  *y=r.h.dh;
   asm {
          mov ah,3
          mov bh,0
          int 0x10
          mov ah,0
          mov al,dl
          mov bx,x
          mov word ptr [bx],ax
          mov al,dh
          mov bx,y
          mov word ptr [bx],ax
   }
}

void set_cur(x,y)
int x,y;
{
// union REGS r;
// r.h.ah=2;
// r.h.bh=3;
// r.h.dh=y;
// r.h.dl=x;
// int86(0x10,&r,&r);
   asm {
          mov ah,2
```

```
        mov bh,0
        mov dh,byte ptr y
        mov dl,byte ptr x
        int 0x10
    }
}
```



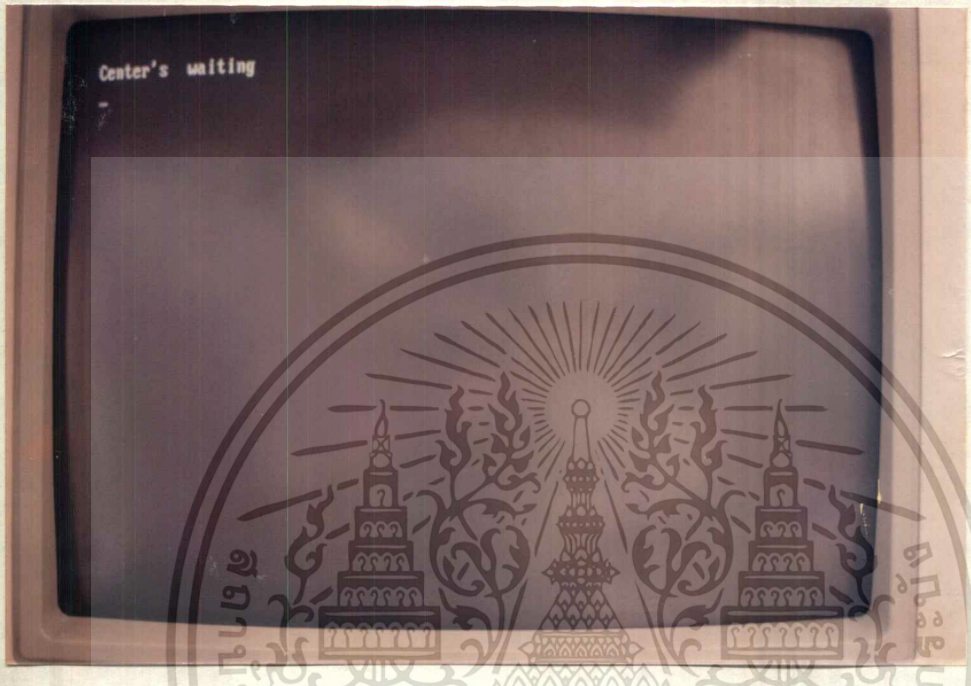




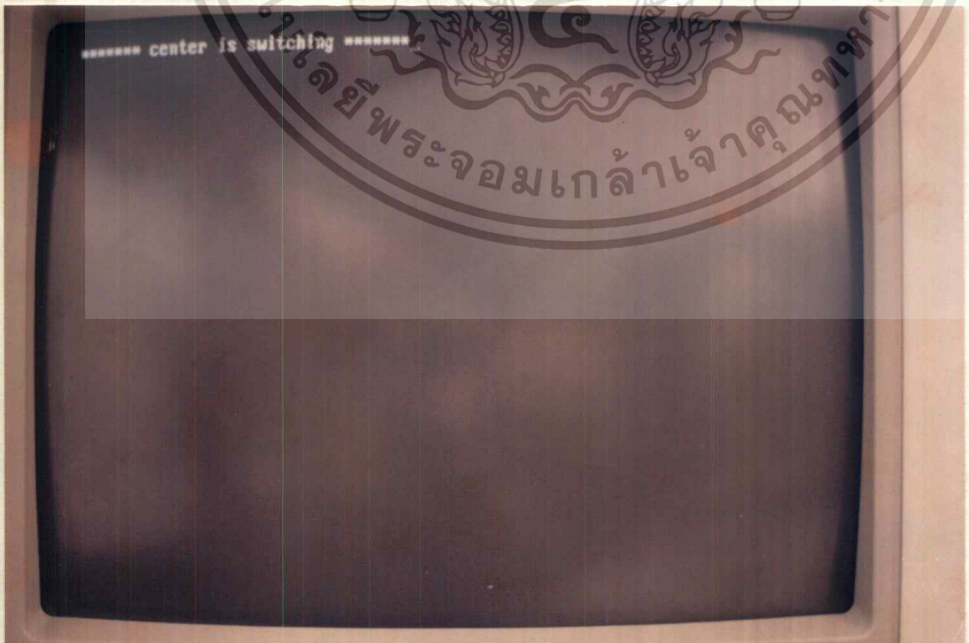
ภาคผนวกรูปภาพ

ขั้นตอนการทดลองและแสดงผล
เมื่อ เทอร์มินัลตัวที่1 ต้องการจะติดต่อกับตัวที่2

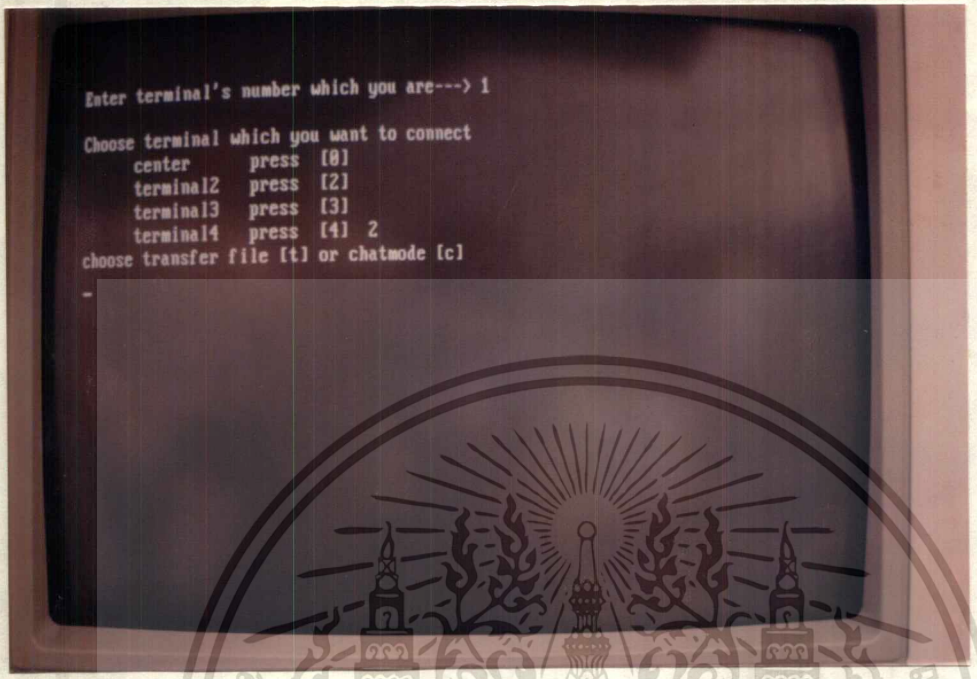
- คอมพิวเตอร์ตัวกลาง จะต้องรันโปรแกรม midjt.exe ไว้รอ ซึ่งจะแสดงผลเป็น



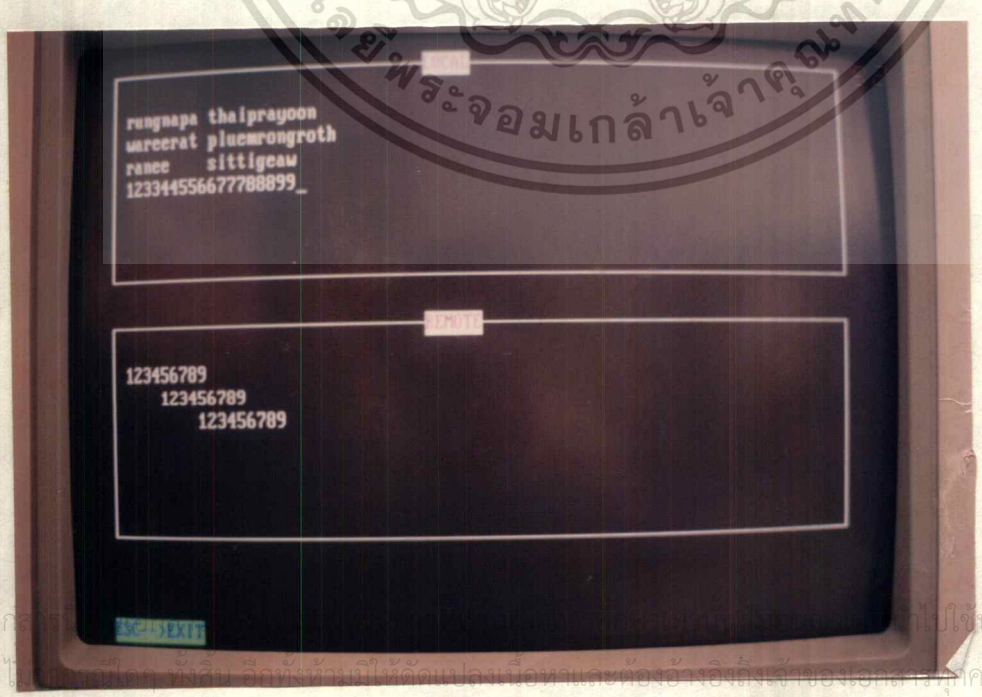
เมื่อมีการร้องขอเข้ามาจะแสดงผลเป็น



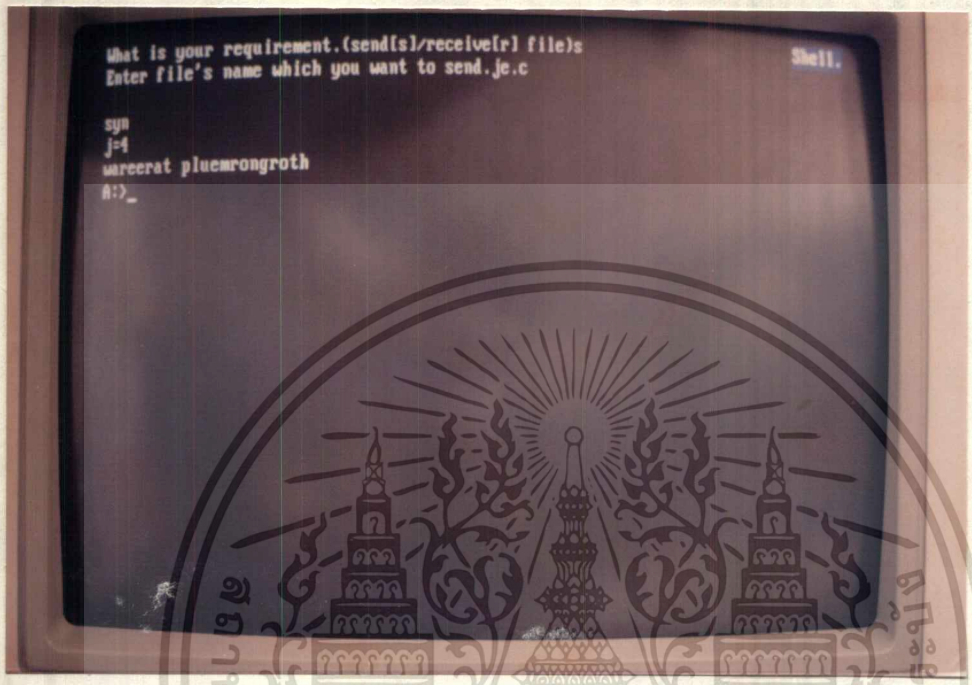
- เทอร์มินัลตัวที่1 ต้องรันโปรแกรม nal3.exe ซึ่งจะแสดงผลเป็น

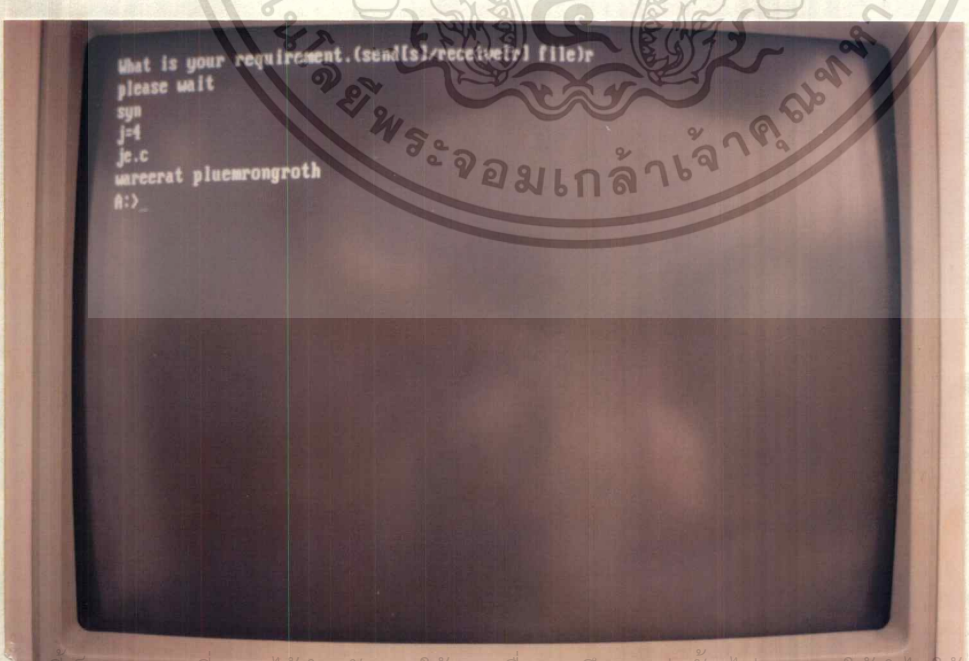


ถ้าเลือก chatmode ก็สามารถที่จะเขียนข้อมูลพูดคุยกันได้แสดงผลเป็น

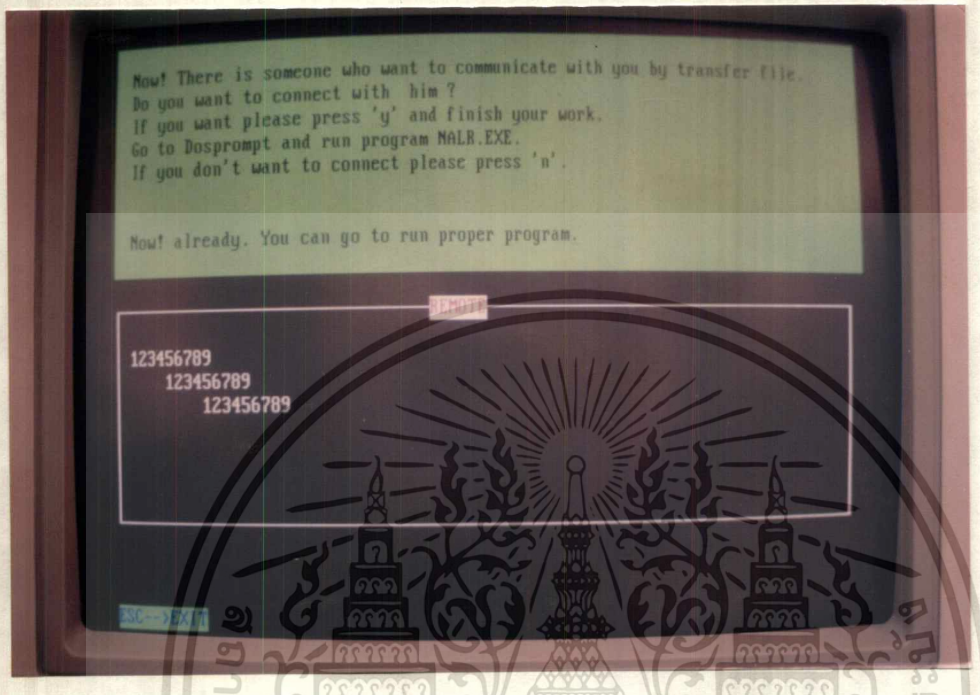


ถ้าเลือก transfer file จะแสดงผลเป็น





-ที่เทอร์มินัลที่2 เริ่มแรกต้องเรียกโปรแกรม resident.exe และมีการร้องขอติดต่อเข้ามาก็จะแสดงข้อความบนหน้าจอเป็น



และให้ทำตามคำสั่งที่แสดงบนหน้าจอนั้น

หนังสืออ้างอิง

1.) HAMMOND, J OSEPH L.. *PERFORMANCE ANALYSIS OF LOCAL COMPUTER NETWORKS* ADDDISON-WERLEY PUBLISHING COMPANY ,1988.

2.) HERBERT SHILDT *C POWER USER' S GUIDE* OSBORNE Mc. GRAW-HILL BERKERY CALIFORNIA,USA 1988.

3.) TREVOR HOUSELEY *DATA COMMUNICATIONS & TELEPROCESSING SYSTEMS* PRENTICE-HALL INTERNATIONAL.

4.) HALL , DOUGLAS V. *MICROPROCESSOR AND INTERFACING PROGRAMING AND HARDWARE* Mc. GRAW HILL ,SINGAPORE 1986.

5.) WILLIAM STALLINGS *MAXWELL DATA AND COMPUTER COMMUNICATIONS* MACMILLAN INTERNATIONAL SINGAPORE ,1991.

6.) ROBERT JOURDAIN *PROGRAMMER ' S PROBLEM SOLVER FOR THE IBM, XT &AT,* NEWYORK 1986.

7.) ธันวา ศรีประโมง *การโปรแกรมภาษาซี สำหรับวิศวกรรม* วิทยาลัยมหานคร กรุงเทพ, 2536.

กิตติกรรมประกาศ

โครงงานฉบับนี้สามารถสำเร็จลุล่วงไปด้วยดี ก็เพราะได้รับคำแนะนำและความช่วยเหลือจากอาจารย์หลายท่าน ทางคณะผู้จัดทำต้องขอขอบพระคุณอาจารย์สุวิพล สิทธิชีวภาค ,อาจารย์ปราโมทย์ วาดเขียน อาจารย์เกรียงไกร วงศ์โรจน์ภรณ์ ,อาจารย์สมยศ จุณณะปิยะ และคุณจรรยา คงเซ็นต์

นอกจากนี้ ต้องขอขอบคุณนายจิรศักดิ์ เหลืองอุไร ,นายมาลินน้อย อินทรสิทธิ ,นายกิตติเดช นิ้มวงษ์เจริญสุข ,นายไพสิฐ ว่องส่งสาร,นายนภัทร สระเอี่ยม และเพื่อนๆทุกคนที่ให้คำแนะนำเป็นอย่างดี

คณะผู้จัดทำ

21มีนาคม2537