โครงข่ายคอมพิวเตอร์รูปดาว

STAR NETWORK

โดย

นาย มาลิน น้อย อินทรสิทธิ์

นาย ภีญทรรศน์ ทับมณี

นาย วิชูรย์ เจียมศิริกาญจน์

ปริญญานิพนธ์เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2535

โครงข่ายคอมพิวเตอร์รูปดาว

นาย มาลิน น้อย อินทรสิทธิ์
นาย ภิญทรรศน์ ทับมณี
นาย วิฑูรย์ เจียมศิริกาญจน์

อาจารย์ที่ปรึกษา
ดร. สุวิพล สิทธิชีวภาค

ปีการศึกษา 2535

บทคัดย่อ

โครงงาน สตาร์เน็ตเวอร์ค (STAR NETWORK) นี้ เป็นโครงการจำลอง ซึ่งได้ศึกษาและนำหลักการสื่อสารของคอมพิวเตอร์เน็ตเวอร์ค (computer network) ที่มีใช้อยู่ทั่วไปมาประยุกต์ร่วมใช้กับโครงงาน โดยการจำลองเน็ตเวอร์ค (network) ที่มีโทโปโลจี (Topology) แบบสตาร์ (star) ซึ่งจะใช้คอมพิวเตอร์ พีซี(computer PC) เป็นอุปกรณ์พื้นฐานในเน็ตเวอร์ค โดยถูกแบ่งเป็น 2 ส่วน คือ ตัวควบคุมส่วนกลาง (HUB) และอุปกรณ์ปลายทาง (Terminal) โดยที่ HUB จะเป็นเครื่องไอบีเอ็ม-พีซี (IBM-PC) ที่มีฮาร์ดแวร์ (Hardware) ที่ทำขึ้นเองเพิ่มเติม ซึ่งทำเป็นการ์ด (card) เสียบในเอ็กซ์แพนชั่น สลอต (expansion slot) โดยจะทำหน้าที่เป็นสวิชชิ่ง (switching) เลือกที่จะเชื่อมต่อ Terminal เข้ากับ HUB ผ่านทวิส-แพร์ วาร์ย (Twist-pair wire) และใช้อินเตอร์เฟส (interface) ตามมาตรฐาน RS232C ที่มีการส่งแบบซีเรียล อะซิงโครนัส (Serial Asynchronous) ส่วน Terminal นั้นสามารถใช้ PC ที่ไม่ต้องมีอะไรเพิ่มเติมอีก สำหรับการสื่อสารกันของ HUB และ Terminal นั้น จะใช้โปรโตคอล (Protocol) ซึ่งเป็นปัจจัยสำคัญในการกำหนดวิธีการส่งข้อมูลผ่าน HUB แบบพิเศษ คือ เป็นรูปแบบที่คิดขึ้นเพื่อให้สามารถทำงานสอดคล้องกับ ฮาร์ดแวร์ ที่ทำขึ้น ซึ่งในโครงการนี้ได้ทำรูปแบบการแลกเปลี่ยนข้อมูลที่สามารถใช้กับ เน็ตเวอร์คนี้ไว้ 3 ลักษณะ ได้แก่ ไฟล์ ทรานซ์เฟอร์ (FILE TRANSFER) อีเลคโทรนิกส์ เมล์ (ELECTRONIC MAIL) และ ออน-ไลน์ คอมมิวนิเคชั่น (ON-LINE COMMUNICATION)

STAR NETWORK

Mr. Malin noi Intrasith

Mr. Piyatat Tahmanee

Mr. Witoon Jiemsirikarn

Advisor

Dr. Suvepon Sittichivapak

1992

ABSTRACT

The Purpose of the "STAR NETWORK" project is to study the principle of today computer network and apply the knowledge to simulate a small but practical network. The simulated network has a "star-like" topology. Every computer in the network is an IBM-PC. The network consists of HUB and Terminals. The IBM-PC HUB has an additional card for IBM-PC expansion slot. The card acts as a switching for the HUB to select a terminal for connection. Twisted-pair wire is used as a transmission line for serial asynchronous transmission. The IBM-PC terminal has no need for additional hardware. The HUB and terminals communicate through a special protocol which depens to the handmade hardware. Both protocol and addition hardware are the important factor for terminal to terminal communication.

The overall objective of this project is to make efficient exchange information across the network, providing 3 exchanging information service (File Transfer, Electronic Mail and On-line communication).

สารบัญ

บทที่ 1

บทนำ

ในโลกแห่งการสื่อสารปัจจุบัน การสื่อสารกันระหว่างคอมพิวเตอร์กับคอมพิวเตอร์ดูจะเป็นสิ่งที่ขาดเสียมิได้ ธุรกิจส่วนใหญ่มักจะใช้คอมพิวเตอร์และโมเด็ม (modem) ในการรับส่งแฟ้มข้อมูล รวมทั้งการใช้คอมพิวเตอร์ในเน็ตเวอร์คเพื่อทำงานอีกหลายหน้าที่ ในเรื่องที่เกี่ยวกับการโอนย้ายข้อมูล ไม่ว่าจะเป็นระบบบูลเลทินบอร์ด(Bulletin Board Systems) บริการออน-ไลน์ คอมมิวนิเคชั่น อีเลคโทรนิกส์ เมล์ และระบบการแบ่งใช้ข้อมูลพื้นฐาน ซึ่งเหล่านี้เป็นเพียงส่วนหนึ่งในหลายๆ แอปปลิเคชั่น (Application)ของคอมพิวเตอร์เน็ตเวอร์ค

ดังนั้น กลุ่มผู้จัดทำจึงมีความคิดที่จะสร้างระบบ คอมพิวเตอร์เน็ตเวอร์ค ขึ้นมา โดยไม่จำเป็นต้องพึ่งพาระบบ เน็ตเวอร์ค ที่มีขายอยู่ในท้องตลาด ซึ่งโครงการนี้ได้เริ่มขึ้นตั้งแต่พรีโปรเจกต์ (Preproject) ปีการศึกษา 2534 โดยเริ่มต้นศึกษาทฤษฎีและรูปแบบต่างๆ ของระบบเน็ตเวอร์ค ที่ใช้อยู่ทั่วไปในภาคการศึกษาที่ 1 และในภาคการศึกษาที่ 2 ได้เริ่มวางรูปแบบและนำเอาทฤษฎีที่ได้ศึกษามากำหนดลักษณะของเน็ตเวอร์คนี้ และเริ่มทดลอง ต่อมาในปีการศึกษา 2535 โปรเจกต์ (Project) ภาคการศึกษาที่ 1 ได้นำเอาปัญหาที่มีในการทดลองครั้งก่อน มาศึกษาและหาทางแก้ไขจนทำให้เน็ตเวอร์คสามารถเชื่อมต่อและรับส่งข้อมูลกันได้ สำหรับภาคการศึกษาที่ 2 ในส่วนของทฤษฎีนั้น ยังคงใช้หลักการและทฤษฎีเดิม แต่ยังได้ค้นคว้าหาทฤษฎีเพิ่มเติมในส่วนท้าย เพื่อจะได้นำไปเป็นแนวทางที่จะพัฒนาต่อไปได้ ส่วนในภาคปฏิบัตินั้น ได้มีการเปลี่ยนแปลงเล็กน้อยในส่วนของวงจรอาร์ดแวร์ และยังเขียนแอปปลิเคชั่นเพิ่มเติมในส่วนของ ซอร์ฟแวร์ (Software) ได้แก่ ไฟล์ ทรานซ์เฟอร์ อีเลคโทรนิกส์เมล์ และออน-ไลน์ คอมมิวนิเคชั่น

ในส่วนรายละเอียดของหลักการและรูปแบบการทำงานของเน็ตเวอร์คนั้น จะอยู่ในบทต่างๆ ของวิทยานิพนธ์ฉบับนี้ แต่ในบทนำนี้จะขอกล่าวถึงโดยสังเขป คือ ระบบเน็ตเวอร์คสำหรับโครงงานนี้ เป็นเน็ตเวอร์คที่เป็น star topology กล่าวคือจะมีรูปร่างคล้ายดาว โดยจะมีตัวควบคุมการทำงานของระบบอยู่ตรงกลาง 1 ตัว และมีอุปกรณ์ปลายทางหลายๆ ตัวเชื่อมต่ออยู่กับศูนย์กลางการควบคุม สำหรับ star network ที่ทำขึ้นนี้จะใช้คอมพิวเตอร์ พีซี 1 เครื่อง ทำหน้าที่เป็นศูนย์กลางการควบคุม ซึ่งถูกเรียกว่า HUB และมีอุปกรณ์ปลายทางซึ่งขอเรียกว่า Terminal เชื่อมต่ออยู่ โดยในเน็ตเวอร์คนี้ได้ใช้พีซีเป็น Terminal เช่นกัน

สำหรับการติดต่อสื่อสารของเน็ตเวอร์คนั้น สามารถมองได้อย่างกว้างๆ เป็น 2 แบบคือ การต้องการติดต่อจาก Terminal หนึ่งกับอีก Terminal หนึ่ง โดยจะต้องผ่าน HUB และการต้องการติดต่อของ Terminal หนึ่งกับ HUB ซึ่งหากมองกันลึกลงไปแล้ว การติดต่อของทั้ง 2 แบบจะมีพื้นฐานเหมือนกันคือ การสื่อสารกันระหว่าง Terminal กับ HUB (แต่สำหรับ ไฟล์ ทรานซ์เฟอร์ และออน ไลน์ คอมมิวนิเคชั่น ซึ่งมีลักษณะการทำงานที่ดูเหมือนว่า Terminal ติดต่อกับอีก Terminal โดยตรงนั้น ที่จริงก็คือ ทรานซ์แพเรนซี (Transparency) ของเน็ตเวอร์คนั่นเอง) เพราะฉะนั้น การอธิบายรูปแบบการทำงานทั้งหมดของเน็ตเวอร์ค จึงกล่าวถึงลักษณะการสื่อสารของ Terminal กับ HUB เท่านั้น ซึ่งก็คือ โปรโตคอล ในแบบที่คิดขึ้นเพื่อให้ทำงานร่วมกับฮาร์ดแวร์ที่ทำขึ้นได้อย่างสมบูรณ์ เพราะหากว่า Terminal กับ HUB คู่หนึ่งสามารถติดต่อสื่อสารกันได้ นั่นย่อมหมายความว่า มีความเป็นไปได้ที่เน็ตเวอร์คจะทำงานได้ตามต้องการ จะเหลือก็แต่เพียงการคัดเลือกว่า Terminal ใดจะเป็นตัวติดต่อกับ HUB เท่านั้น ซึ่งปัญหานี้ได้ถูกแก้ไขโดยทำการสร้างวงจรฮาร์ดแวร์ขึ้นมาเพิ่มเติมจากวงจรพื้นฐานที่มีอยู่ใน พีซี แล้ว ฮาร์ดแวร์นี้จะทำหน้าที่เป็นสวิชชิ่ง โดยจะคัดเลือก Terminal ที่ต้องการจะติดต่อกับ HUB ซึ่งทุก Terminal จะมีลักษณะการร้องขอการเชื่อมต่อในทฤษฎีของ มีเดีย แอคเซส คอนโทรล (Media Access Control) แบบเซ็นทรัลลี่ คอนโทรล ดีมานด์ แอสไซน์เมนท์ เทคนิก (Centrally Control Demand Assignment Technique) จะกล่าวอีกครั้งในส่วนของทฤษฎี และในส่วนของโปรโตคอลที่ใช้ในเน็ตเวอร์คนี้ จะมีลักษณะที่นำเอาทฤษฎีต่างๆ มาประยุกต์และคิดเพิ่มเติมขึ้นด้วยโดยได้นำทฤษฎีแบบ ไบท์-โอเรียน โปรโตคอล (Byte-oriented Protocol) มาใช้ในโฟล คอนโทรล (Flow Control) แต่จะใช้เพียง 4 รหัสของรหัสแอสกี (ASCII) เท่านั้น (จะกล่าวอีกครั้งในส่วนของทฤษฎี)

สำหรับรายละเอียดของวงจร ฮาร์ดแวร์ ที่ทำขึ้นและขั้นตอนของ โปรโตคอลที่ใช้ในเน็ตเวอร์คนี้ จะถูกอธิบายอย่างละเอียดในวิทยานิพนธ์ฉบับนี้ โดยเนื้อหาทั้งหมดจะถูกแบ่งเป็นบทๆ ดังนี้

บทที่ 1 บทนำ

บทที่ 2 ทฤษฎีและหลักการ

บทที่ 3 วิธีการทำงานของเน็ตเวอร์ค

บทที่ 4 ผลการทดลองและสรุป

บทที่ 5 บทแทรก

บทที่ 2
ทฤษฎีและหลักการ

2.1 คาร์แรกเตอร์-โอเรียน (CHARACTER-ORIENTED)

โปรโตคอล สำหรับการสื่อสารทั่วไป สามารถแบ่งได้เป็น 2 ประเภท คือ

- บิท-โอเรียน โปรโตคอล (Bit-Oriented Protocol)
- ไบท์-โอเรียน โปรโตคอล (Byte-Oriented Protocol) หรือ คาร์แรกเตอร์-โอเรียน โปรโตคอล (Character-Oriented Protocol)

*** สำหรับเน็ตเวอร์คที่ทำขึ้นได้ใช้ประเภท Byte-Oriented Protocol ตัวอย่างหนึ่งของ Byte-Oriented Protocol ที่ใช้กันทั่วไปบนพีซี ก็คือ ASCII (American Standard Code for Information Interchange) ดังนั้น การสื่อสารของเน็ตเวอร์คนี้จึงนำรหัสแอสกีมาใช้ (รูปที่ 1) โดยรหัสแอสกีที่สำคัญที่ถูกนำมาใช้ในโฟล คอนโทรล ของเน็ตเวอร์คนี้จะใช้ 4 รหัส เท่านั้น คือ ACK, ENQ, ETX, SYN ซึ่งมีคำอธิบายดังนี้

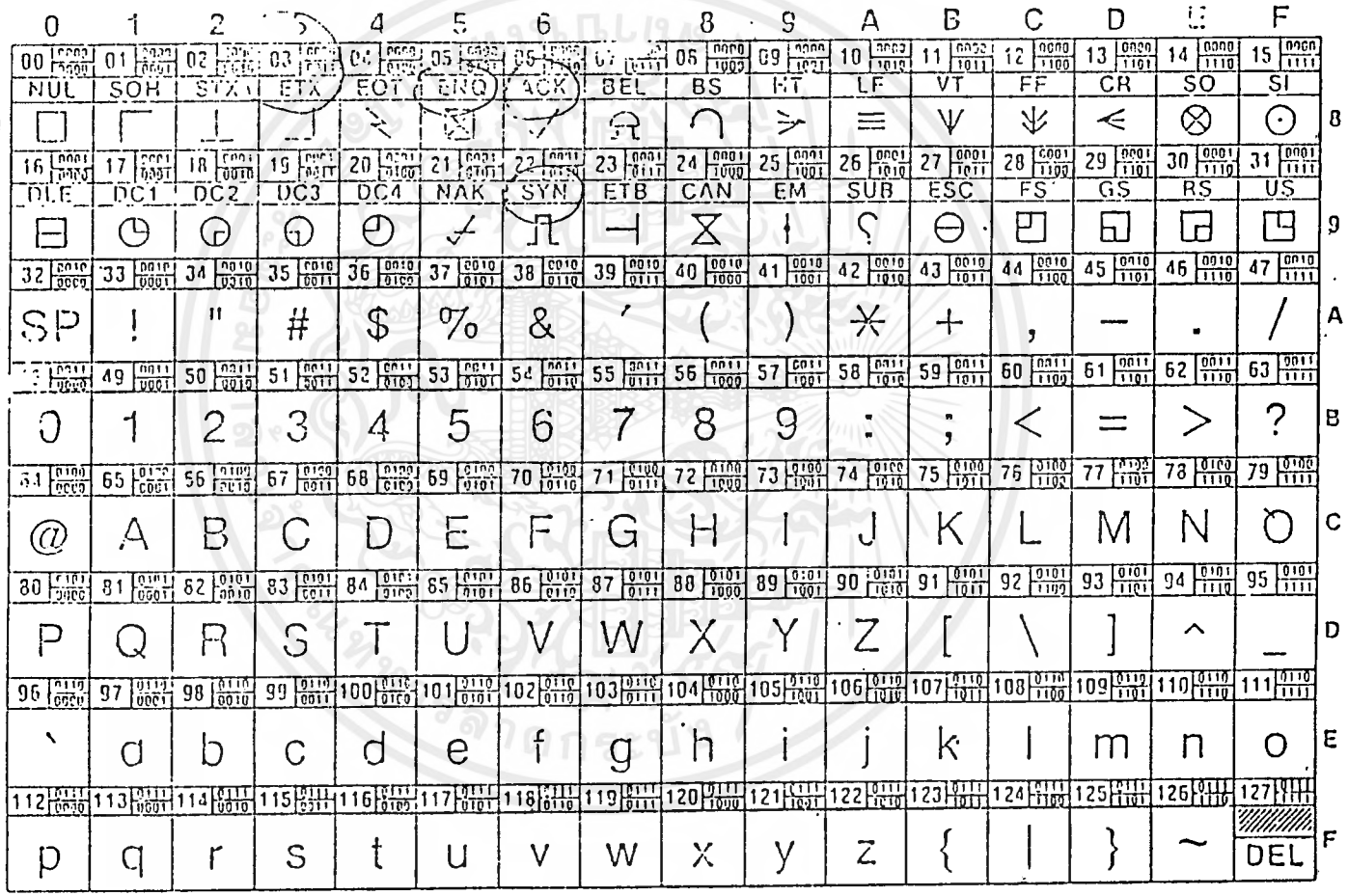
ACK (Acknowledge) : คาแรกเตอร์นี้จะถูกใช้โดยทางด้านรับเพื่อบอกให้ทราบว่า ได้รับข้อมูลที่ไม่ผิดพลาดแล้ว

ENQ (Enquiry) : เมื่อคอมพิวเตอร์ 2 เครื่อง เริ่มที่จะติดต่อสื่อสารกัน เครื่องทางด้านส่งจะส่ง ENQ เพื่อให้ทราบว่า เครื่องปลายทางอีกด้านยังเชื่อมต่ออยู่หรือไม่

ETX (End of Text) : คาแรกเตอร์นี้เป็นสัญลักษณ์ว่าสิ้นสุดบล็อกข้อมูล

SYN (Synchronous) : คาแรกเตอร์นี้ถูกใช้ในระบบซิงโครนัสเพื่อที่จะเริ่มหรือรักษาการซิงโครไนซ์ ในเส้นทางการสื่อสารในขณะที่ไม่มีข้อมูลไหลอยู่

Figure 1 The ASCII chart.



2.2 มีเดีย แอคเซส โปรโตคอล (MEDIA ACCESS PROTOCOL)

ในระบบแลน (LAN) ที่มีอยู่ในปัจจุบัน จะมีส่วนของมีเดีย แอคเซส คอนโทรล โปรโตคอล (Media Access Control Protocols) ซึ่งสามารถแบ่งได้เป็น 5 กลุ่ม ดังนี้

1. ฟิกซ์ แอสไซน์เมนท์ เทคนิก (Fixed Assignment Technique)
2. แรนดอม แอคเซส เทคนิก (Random Access Technique)
3. เซ็นทรัลลี่ คอนโทรล ดีมานด์ แอสไซน์เมนท์ เทคนิก (Centrally Controlled Demand Assignment Technique)
4. ดีมานด์ แอสไซน์เมนท์วิดธ์ ดิสทริบิวท์ คอนโทรล (Demand Assignment With Distributed Control)
5. เทคนิกอื่นๆ (Other Techniuqe)

อนึ่ง สำหรับ สตาร์ เน็ตเวอร์ค ที่ทำขึ้นได้ใช้ เซ็นทรัลลี่ คอนโทรล ดีมานด์ แอสไซน์เมนท์ เทคนิก ซึ่งเทคนิกนี้มีความยืดหยุ่นมากกว่า 2 เทคนิกแรก โดยจะมีส่วนควบคุมศูนย์กลางเป็นตัวกำหนดความต้องการ ซึ่งจะแตกต่างกันที่เทคนิกแรกจะไม่สามารถปรับเปลี่ยนตามความต้องการของผู้ใช้ และถ้าระบบต้องการให้มีดีเลย์สั้นๆ ระบบอาจจะเสียหายได้ ส่วนเทคนิกที่ 2 จะมีปัญหาการชนกันของข้อมูล ดังนั้น เทคนิกที่ 3 ผู้ใช้จะสามารถแสดงความต้องการการส่งข้อมูล ก่อนที่ช่องสัญญาณจะถูกแบ่งให้สำหรับข้อมูลนั้น ซึ่งกรรมวิธีแบ่งช่องสัญญาณจะทำได้ 2 วิธี คือ วิธี พอลลิ่ง (polling) ซึ่งส่วนควบคุมศูนย์กลางจะสอบถามไปยังเครื่องปลายทางว่ามีข้อมูลจะส่งหรือไม่ ส่วนอีกวิธีซึ่งเป็นวิธีที่ใช้ในเน็ตเวอร์คนี้ คือ วิธี รีเซิร์ฟ (Reserve) ซึ่ง Terminal จะเป็นผู้ร้องขอการใช้ช่องสัญญาณ เมื่อต้องการส่งข้อมูล วิธีนี้จะมีความเหมาะสมที่สุดในการใช้พีซีเป็นมีเดีย และยังมีทรูพุท (Throughput) ที่สูงอีกด้วย แต่ถ้าเป็น ทีดีเอ็มเอ (TDMA) หากมีการปิดหรือไม่ใช้ในบาง Terminal ก็จะเป็นการสูญเสียเวลาที่จัดไว้ให้ของแต่ละ Terminal ไป

บทที่ 3

วิธีการทำงานของเน็ตเวอร์ค

ในบทนี้ จะอธิบายวิธีทดลองการทำงานของระบบเน็ตเวอร์ค ซึ่งถูกแบ่งออกเป็น 2 ส่วน คือ ส่วนที่เป็นซอร์ฟแวร์หรือเน็ตเวอร์ค โปรโตคอล และส่วนที่เป็นวงจรฮาร์ดแวร์ ซึ่งทั้ง 2 ส่วนจะต้องทำงานอย่างสอดคล้องกัน

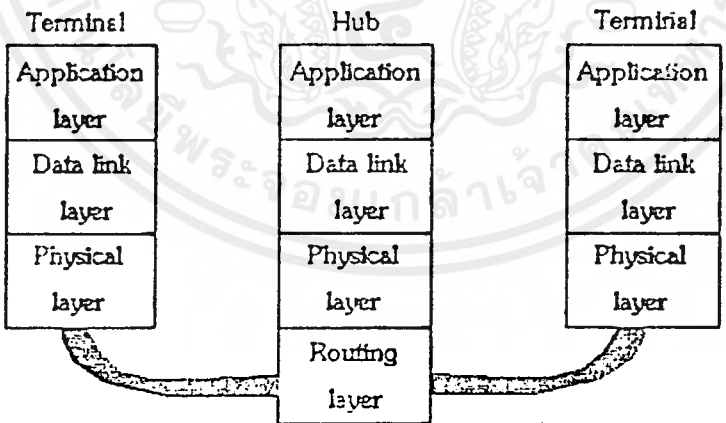
3.1 เน็ตเวอร์ค โปรโตคอล (NETWORK PROTOCOL)

ส่วนต่อไปนี้จะกล่าวถึงการทำงานโดยรวมของเน็ตเวอร์ค ซึ่งจะอธิบายให้เข้าใจในรูปของกฎการสื่อสารโปรโตคอล (PROTOCOL) ที่คิดขึ้นสำหรับใช้ในเน็ตเวอร์คนี้ โดยจะใช้หลักการของเลเยอร์ โปรโตคอล (LAYERED PROTOCOL)*

สตาร์ เน็ตเวอร์ค นี้ สามารถแบ่งการทำงานของ Terminal ได้เป็น 3 ชั้น คือ

1. ฟิสิคัล เลเยอร์ (Physical Layer)
2. ดาต้าลิงค์ เลเยอร์ (Datalink Layer)
3. แอปปลิเคชั่น เลเยอร์ (Application Layer)

ส่วนการทำงานของ HUB สามารถแบ่งได้เป็น 4 ชั้น โดยมีการเพิ่มรูทติ้ง เลเยอร์ (Routing Layer) เข้าไปในชั้นล่างของ ฟิสิคัล เลเยอร์ อีกชั้นหนึ่ง ลักษณะของการแบ่งเลเยอร์ของเน็ตเวอร์ค แสดงดังรูป 1



รูปที่ 1 ลักษณะของเนตเวอร์คที่แบ่งเป็นชั้น

---

* หมายเหตุ การใช้หลักของ Layer Protocol มาอธิบายก็เพื่อให้เข้าใจการทำงานของเน็ตเวอร์คได้ง่ายขึ้น แต่มิได้หมายความว่าเน็ตเวอร์คนี้ เป็นไปตามมาตรฐาน OSI ของ ISO ทั้งหมด

การทำงานของ ฟิสิคัล เลเยอร์ และ ดาต้าลิงค์ เลเยอร์ นั้น จะเหมือนกันทั้งใน Terminal และ HUB แต่แอปปลิเคชั่นใน HUB และ Terminal จะต่างกันเล็กน้อย และชั้นรูทติ้งจะมีแต่ใน HUB ดังนั้นจึงขออธิบายการทำงานของชั้น ฟิสิคัล และ ดาต้าลิงค์ ก่อน

ฟิสิคัล เลเยอร์ (PHYSICAL LAYER)

ชั้นนี้จะเป็นชั้นเดียวใน 4 ชั้นที่เครื่อง ไอบีเอ็ม-พีซี มีอุปกรณ์ที่จะต้องใช้อยู่แล้ว คือ ยูอาร์ต (UART) 8250 ซึ่งเป็นอุปกรณ์หลักในการทำงานของชั้นนี้ โดยนำฟังก์ชั่นเหล่านี้มาใช้ คือ การเปลี่ยนข้อมูลไปมาระหว่าง แบบอนุกรม อะซิงโครนัส กับแบบขนาน เปลี่ยน บอดเรต (Baud Rate) เลือกรูปแบบของข้อมูลอะซิงโครนัส ส่งสัญญาณ ฮาร์ดแวร์ อินเทอร์รัพท์ (hardware interrupt) เมื่อข้อมูลพร้อม ตรวจสอบ พาริตี้ เออเรอร์ (Parity Error) วิธีการควบคุมการทำงานฟังก์ชั่นเหล่านี้ในเน็ตเวอร์คจะอยู่ในส่วนของโปรแกรม

หน้าที่ของชั้นฟิสิคัลใน สตาร์ เน็ตเวอร์ค นี้ คือ

- เปลี่ยนข้อมูลระหว่างแบบอนุกรม ใน ทวิสท์-แพร์ วาร์ย กับข้อมูลแบบขนาน และใช้ความเร็วข้อมูลใน ทวิสท์-แพร์ วาร์ย ที่ 38,400 บอด
- ทำการสื่อสารแบบ ฮาร์ฟ ดูเพล็ก (Half Duplex)
- ตรวจพาริตี้ (Parity)
- เซท แฟลก (Set Flag) เพื่อบอกว่ามีข้อมูลเข้ามา

ดาต้าลิงค์ เลเยอร์ (DATALINK LAYER)

ส่วนที่ทำหน้าที่เป็นดาต้าลิงค์ในชั้นนี้ก็คือ อินเทอร์รัพท์ เซอร์วิส รูทีน (Interrupt Service Routine) ที่เขียนขึ้น ซึ่งจะอำนวยความสะดวกต่อแอปปลิเคชั่นโปรแกรม (Application Program) และในฐานะที่ดาต้าลิงค์เป็นมีเดีย แอคเซส โปรโตคอล จึงมีหน้าที่ดังนี้

- คอยรับไบท์ข้อมูลจากชั้นฟิสิคัล มาประกอบเป็นแพคเก็ท (Packet) เก็บไว้ในบัฟเฟอร์ (Buffer) และรอให้แอปปลิเคชั่นมาอ่าน
- ตรวจสอบ พาริตี้ เออเรอร์ ของข้อมูลจากชั้นฟิสิคัล ถ้าหากมี ฝ่ายรับจะส่งแนค แพคเก็ท(NAK Packet)กลับไป และจะรอรับแพคเก็ทที่ถูกต้อง แต่หากไม่มีเออเรอร์ ชั้นนี้จะเซท แฟลกบอกแอปปลิเคชั่นมาอ่าน
- รับแพคเก็ทจากแอปปลิเคชั่นและส่งต่อไปให้ฟิสิคัลทีละไบท์

- ควบคุมไบท์ ซิงโครนัส (Byte Sychronous) โดยใช้หลักการสตอป แอนด์ เวท (stop and wait) เพื่อป้องกัน โอเวอร์รัน เออเรอร์ (overrun error)

หลักในการรับส่งข้อมูลระหว่าง ดาต้าลิงค์ เลเยอร์ กับ ฟิสิคัล เลเยอร์

ดาต้าลิงค์ เลเยอร์ จะเป็นตัวทำ มิเดีย แอคเซส ซึ่งในเน็ตเวอร์คนี้จะเป็นแบบ เซ็นทรัลลี่ คอนโทรล ดีมานด์ แอสไซน์เมนท์ เทคนิก โดย Terminal ที่มีแพคเก็ทจะส่งจะต้องแย่งกับขชันแนลจาก HUB โดยฮาร์ดแวร์ในชั้นรูทติ้งจะเป็นตัวบังคับให้ต้องใช้เทคนิกนี้ ซึ่งมีลักษณะดังนี้

การรับแพคเก็ท ก่อนที่จะเริ่มรับแพคเก็ทข้อมูล ชั้นฟิสิคัลจะทำการขอการเชื่อมต่อ (connection) มายังชั้นดาต้าลิงค์ ด้วยการส่ง เอ็นไควร์ลี่ (ENQ) มาอย่างต่อเนื่อง ชั้นดาต้าลิงค์จะต้องตอบรับโดยการส่งแอคโนวเลดจ์ (ACK) ไปให้ชั้นฟิสิคัล หลังจากนั้น รหัสตัวต่อไปจากชั้นฟิสิคัลจึงถือเป็นข้อมูล และเมื่อชั้นฟิสิคัลเซท แฟลก บอกว่าได้รับข้อมูลจากชั้นฟิสิคัลของทางด้านส่งแล้ว ชั้นดาต้าลิงค์ของด้านรับจึงจะอ่านข้อมูล 1 ไบท์นั้น พร้อมทั้งส่งรหัส ACK ไปให้ฟิสิคัล 1 ไบท์ จากนั้น ชั้นดาต้าลิงค์จึงจะได้รับตัวต่อไป ขบวนการจะเป็นเช่นนี้ไปจนกว่าจะรับหมดทั้งแพคเก็ท

การส่งแพคเก็ท ก่อนจะส่งแพคเก็ทข้อมูลจะต้องทำการแย่งชันแนลของ HUB กับ Terminal อื่นๆ โดยการส่ง ENQ ไปให้ชั้นฟิสิคัลอย่างต่อเนื่องจนกว่าจะได้รับ ACK จากชั้นฟิสิคัล จึงสามารถส่งข้อมูลไบท์แรกไปให้ชั้นฟิสิคัลได้ ต่อจากนั้นจะต้องรอรับ ACK อีกจึงจะสามารถส่งข้อมูลไบท์ต่อไปให้ชั้นฟิสิคัลได้ ขบวนการจะเป็นเช่นนี้จนกว่าจะส่งได้หมดทั้งแพคเก็ท

เมื่อส่งแพคเก็ทไปให้ชั้นฟิสิคัลแล้ว จะต้องเก็บแพคเก็ทนั้นสำรองไว้ชุดหนึ่ง เผื่อในกรณีที่ได้รับ NAK Packet กลับมา ก็จะสามารถส่งแพคเก็ทสำรองนั้นไปให้ชั้นฟิสิคัลอีกครั้งหนึ่งได้

แอปปลิเคชั่น เลเยอร์ (Application Layer)

ชั้นนี้จะเป็นหน้าที่ของแอปปลิเคชั่น ซอร์ฟแวร์ ลักษณะการสื่อสารที่เราจัดให้ในชั้นนี้มี

1. ไฟล์ ทรานซเฟอร์ โปรแกรม (File Transfer Program (FTP)) เป็นการสื่อสารข้อมูลแบบทางเดียวระหว่าง Terminal แต่ละเครื่อง โดยต้องเตรียมข้อมูลที่จะส่งให้เป็นไฟล์เสียก่อน

2. ออน-ไลน์ คอมมิวนิเคชั่น (On-Line Communication) หรือ ชั้ท โหมด (chat mode) เป็นการสื่อสารแบบ อาร์ฟ ดูเพล็กซ์ ระหว่าง Terminal แต่ละเครื่องโดยไม่ต้องเตรียมข้อมูลก่อน กล่าวคือ สามารถพิมพ์ข้อมูลตอบโต้กันในทันที

3. อิเลกทรอนิกส์ เมล์ (Electronic Mail) เป็นการสื่อสารข้อมูลระหว่างผู้ใช้เครื่อง โดยที่ผู้ส่งจะต้องเตรียมข้อมูลเป็นไฟล์ แล้วส่งไฟล์นั้นไปฝากไว้กับ HUB ภายใต้ชื่อของผู้รับ จากนั้นผู้รับจะสามารถตรวจสอบกับ HUB ได้ว่ามีข้อมูลฝากมาหรือไม่ ถ้ามีก็จะส่งสัญญาณควบคุมไปให้ HUB ส่งข้อมูลนั้นกลับมาให้ได้

หน้าที่ของ แอปปลิเคชั่น เลเยอร์ มี 3 หน้าที่ คือ

1. รอรับแพคเก็ท จากชั้นดาต้าลิงค์ แล้วนำข้อมูลนั้นมาแยกจัดการตามความเหมาะสม เช่น เก็บลงแผ่นข้อมูล แสดงผลทางจอ หรือส่งแพคเก็ทชนิดอื่นๆ ตอบโต้กลับไป

2. นำข้อมูลที่จะส่งมาแบ่งเป็นแพคเก็ท แล้วส่งให้ชั้นดาต้าลิงค์

3. หน้าที่ที่ 3 นี้ จะเป็นแอปปลิเคชั่นของ HUB เท่านั้น ซึ่งมีหน้าที่ คือ ต้องพิจารณาว่าแพคเก็ทที่รับมานั้นเป็นของตัวเอง หรือเป็นแพคเก็ทที่จะส่งไปให้ Terminal อื่น เพื่อที่จะจัดการกับข้อมูลนั้นได้ถูกต้อง หรือจะส่งแพคเก็ทนั้นกลับไปให้ชั้นดาต้าลิงค์ ส่งไปให้ Terminal ต่อไป

อนึ่ง ข้อมูลที่แอปปลิเคชั่นส่งไปให้ชั้นดาต้าลิงค์นั้น จะต้องอยู่ในรูปแพคเก็ท ซึ่งสามารถแบ่งออกตามลักษณะข้อมูลใน แพคเก็ท เป็น 2 ชนิด คือ

1. คอนโทรล แพคเก็ท (Control Packet) ประกอบไปด้วย

- ไฟล์ ทรานซเฟอร์ ริเควส แพคเก็ท (File Transfer Request Packet)
- ไฟล์ ทรานซเฟอร์ คอนเฟิร์ม แพคเก็ท (File Transfer Confirm Packet)
- ชัท โหมด ริเควส แพคเก็ท (Chat Mode Request Packet)
- ชัท โหมด คอนเฟิร์ม แพคเก็ม (Chat Mode Confirm Packet)
- อี-เมล์ ริเควส แพคเก็ท (E-mail Request Packet)
- อี-เมล์ คอนเฟิร์ม แพคเก็ท (E-mail Confirm Packet)
- "ฮู" แพคเก็ท ("WHO" Packet)
- "ลิสท์" แพคเก็ท ("List" Packet)
- ดิสคอนเนค แพคเก็ท (Diaconnect Packet)

- นอน-แอคโนวเลดจ์ แพคเก็ท (Non-acknowledge Packet)
- พอล อี-เมล์ แพคเก็ท (Poll E-mail Packet)

2. ดาต้า แพคเก็ท (Data Packet) ประกอบด้วย

- ชัท แพคเก็ท (Chat Packet)
- อี-เมล์ แพคเก็ท (E-mail Packet)
- ไฟล์ ทรานซเฟอร์ แพคเก็ท (File Transfer Packet)

จะเห็นได้ว่ามีแพคเก็ทอยู่หลายชนิด แต่แท้จริงแล้วแพคเก็ทเหล่านี้จะมีรูปแบบของแต่ละแพคเก็ทอยู่ที่ไบท์ที่ 5 ซึ่งเป็นฟิล์ด (Field) ที่บอกชนิดของแพคเก็ทนั่นเอง สำหรับรายละเอียดของรูปแบบแพคเก็ทจะรวมอยู่กับโฟล ชาร์ท (Flow Chart) ด้านหลัง

เพื่อที่จะให้ง่ายต่อการเข้าใจวิธีการใช้แพคเก็ทแต่ละชนิด จะขอแบ่งแพคเก็ทออกตามลักษณะหน้าที่ของแพคเก็ท ซึ่งมีดังนี้

1. รีเควส แพคเก็ท (Request Packet)
   - ไฟล์ ทรานซเฟอร์ รีเควส (File Transfer Request)
   - ชัท โหมด รีเควส (Chat Mode Request)
   - อี-เมล์ รีเควส (E-mail Request)
   - "ลิสท์" แพคเก็ท ("List" Packet)
   - "ฮู" แพคเก็ท ("WHO" Packet)
2. คอนเฟิร์ม แพคเก็ท (Confirm Packet)
   - ไฟล์ ทรานซเฟอร์ คอนเฟิร์ม (File Transfer Confirm)
   - ชัท โหมด คอนเฟิร์ม (Chat Mode Confirm)
   - อี-เมล์ คอนเฟิร์ม (E-mail Confirm)
3. ดาต้า แพคเก็ท (Data Packet)
   - ไฟล์ ทรานซเฟอร์ แพคเก็ท (File Transfer Packet)
   - ชัท แพคเก็ท (Chat Packet)
   - อี-เมล์ แพคเก็ท (E-mail Packet)
4. Flow Control Packet
   - นอน-แอคโนวเลดจ์ แพคเก็ท (Non-acknowledge Packet)
   - ดิสคอนเนค แพคเก็ท (Disconnect Packet)
   - พอล อี-เมล์ แพคเก็ท (Poll E-mail Packet)

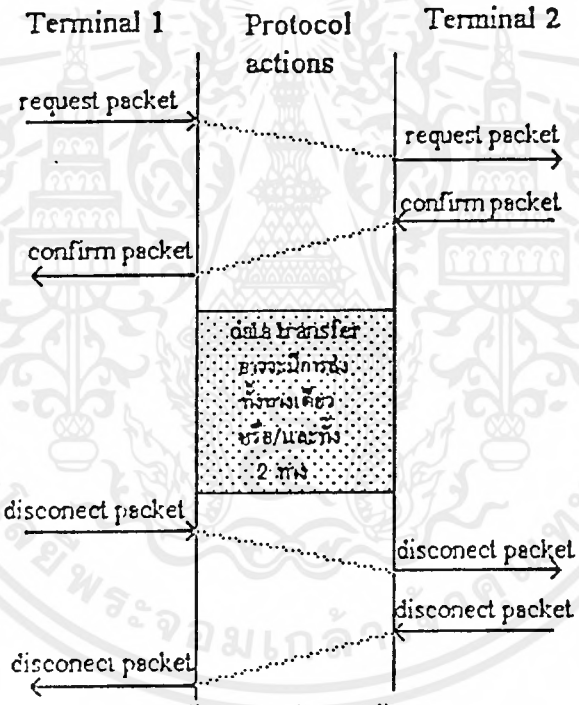
รีเควส แพคเก็ท เป็นแพคเก็ทที่ถูกส่งโดยแอปปลิเคชั่นหนึ่งไปยังอีกแอปปลิเคชั่นหนึ่งใน Terminal ปลายทาง เพื่อแจ้งการขอบริการในการส่งข้อมูลในลักษณะ

ต่างๆ ที่กล่าวแล้วข้างต้น

คอนเฟิร์ม แพคเก็ท เป็นแพคเก็ทที่ถูกส่งเพื่อเป็นการตอบรับรีเควสของการสื่อสารชนิดนั้นๆ

ดิสคอนเนค แพคเก็ท จะถูกส่งออกมาเมื่อ Terminal ใดๆ ก็ตาม ต้องการจะเลิกการแลกเปลี่ยนข้อมูลที่ทำกันอยู่

สำหรับการแลกเปลี่ยนข้อมูลของแพคเก็ททุกชนิดนั้น จะมีหลักการเหมือนกันดังรูปที่ 2



รูปที่ 2 แสดงขั้นตอนการสื่อสารในชั้น Application

ในแพคเก็ทหลายๆ ชนิดที่กล่าวมาแล้วนั้น จะมีอยู่บางชนิดที่มีหน้าที่การทำงานเป็นแบบเฉพาะของมันเอง ซึ่งมีดังนี้

1. "ฮู"แพคเก็ทเป็นแพคเก็ทที่ใช้สำหรับรีเควสไปยัง HUB เพื่อขอให้เตรียมการทรานซเฟอร์ข้อมูลเกี่ยวกับผู้ใช้เครื่องในเน็ตเวอร์คขณะนั้น
2. "ลิสท์" แพคเก็ทเป็นแพคเก็ทที่ใช้สำหรับรีเควสไปยัง HUB เพื่อขอให้เตรียมการทรานซเฟอร์ข้อมูลเกี่ยวกับไฟล์ที่อยู่ในเมล์ บอกซ์(mail box : การส่งอี-เมล์นั้นผู้ส่งจะส่งไฟล์มาฝากเก็บไว้กับ HUB ในไดเร็กทอรี (directory)หนึ่งที่เรียกว่า เมล์ บอกซ์ ซึ่งผู้รับจะมาเปิดดูว่ามีไฟล์ส่งมาถึงหรือไม่ โดยการใช้ List Packet)
3. พอล อี-เมล์ สำหรับมีเดีย แอคเซส โปรโตคอล ของเน็ตเวอร์คที่เป็นแบบเซ็นทรัลลี่ คอนโทรล ดีมานด์ แอสไซน์เมนท์ เทคนิก Terminalใดต้องการจะส่งแพคเก็ทชั้นคาต้าลิงค์ต้องส่งสัญญาณไปแย่งชันแนลที่ HUB กับ Terminal อื่นๆ การที่ต้องเลือกมีเดีย แอคเซส โปรโตคอลเช่นนี้เพราะฮาร์ดแวร์ในชั้นรูทติ้งกำหนดให้เป็นเช่นนั้น แต่ฮาร์ดแวร์ในชั้นรูทติ้ง ไม่ได้จัดให้มีการแย่งชันแนลกันได้ระหว่าง Terminal กับ HUB ดังนั้น เมื่อ Terminal ใดต้องการให้ HUB ส่งข้อมูลจึงต้องส่งข้อมูลควบคุมไปแย่งชันแนลให้ HUB ด้วย ซึ่งข้อมูลที่ว่านี้ถูกเรียกว่า พอล อี-เมล์ แพคเก็ท ที่เป็นเช่นนี้ก็เพราะว่า ข้อมูลส่วนใหญ่ที่ Terminal ต้องการจาก HUB ก็คือ อี-เมล์ แต่ความเป็นจริงแล้ว พอล อี-เมล์ ใช้ส่งทุกครั้งเมื่อ Terminal ต้องการแพคเก็ทข้อมูลข่าวสารจาก HUB ซึ่งมีข่าวสารเกี่ยวกับผู้ใช้เน็ตเวอร์คและไฟล์ในเมล์ บอกซ์
4. อี-เมล์ แพคเก็ท เช่นเดียวกับ พอล อี-เมล์ แพคเก็ท ไม่ได้มีเอาไว้ใส่ข่าวสาร อี-เมล์ อย่างเดียว แต่เอาไว้ใส่ข่าวสารทุกชนิดที่มีต้นกำเนิดมาจาก HUB แล้วส่งมายัง Terminal จากแพคเก็ทชนิดที่ 3 และ 4 พอจะสรุปลักษณะของมันได้ว่า ทุกครั้งที่ Terminal ส่ง พอล อี-เมล์ มาให้ HUB HUB จะส่งข่าวสารชนิดที่ Terminal นั้นส่งรีเควสมาขอไว้ก่อนแล้วลงไปใน อี-เมล์ แพคเก็ท
5. รีจิสเตรชั่น แพคเก็ท (Registration Packet) เป็นแพคเก็ทที่ Terminal ส่งไปบอก HUB ว่า
   - ขณะนี้ใครเป็นผู้ใช้ Terminal อยู่
   - ใช้เซทหรือรีเซท อินเทอร์รัพท์ แฟลก
   - ใช้บอกเลิกออกจากเน็ตเวอร์ค

6. แนค แพคเก็ท (NAK Packet) ความจริงแล้วแอปปลิเคชั่นไม่มีหน้าที่จัดการ แนค แพคเก็ท แต่จะขอกล่าวไว้ในที่นี้ด้วย คือ เมื่อชั้นดาต้าลิงค์รับแพคเก็ท มาได้แล้ว ชั้นดาต้าลิงค์จะเช็คดูว่าข้อมูลที่รับมาได้มีพาริตี้เออเรอร์ หรือไม่จากบิทที่2ของ ไลน์ สเตตัส รีจิสเตอร์ (Line Status Register) (ทั้งนี้ต้องมีการเอนาเบิล อินเทอร์รัพท์ (enable interrupt) ที่ 8250 ที่ถูกต้อง ซึ่งรายละอียดจะอยู่ในโปรแกรม) ถ้ามีเออเรอร์ ชั้นดาต้าลิงค์จะส่ง NAK ออกไป และรอรับแพคเก็ทเดิมที่สมบูรณ์อีกครั้ง ในทางกลับกัน ถ้าชั้นดาต้าลิงค์ได้รับ NAK ก็จะส่งแพคเก็ทเดิมที่เพิ่งจะส่งไปอีกครั้งหนึ่ง

ในการทำงานลักษณะนี้ จะเห็นได้ว่า แอปปลิเคชั่นจะมั่นใจได้แน่นอนว่าแพคเก็ทที่ส่งไปให้ชั้นดาต้าลิงค์จะถึงปลายทางอย่างสมบูรณ์แน่นอน และแพคเก็ทที่มาจากชั้นดาต้าลิงค์ ก็จะเป็นเออเรอร์ ฟรี แพคเก็ท (error free packet) แน่นอนเช่นกัน

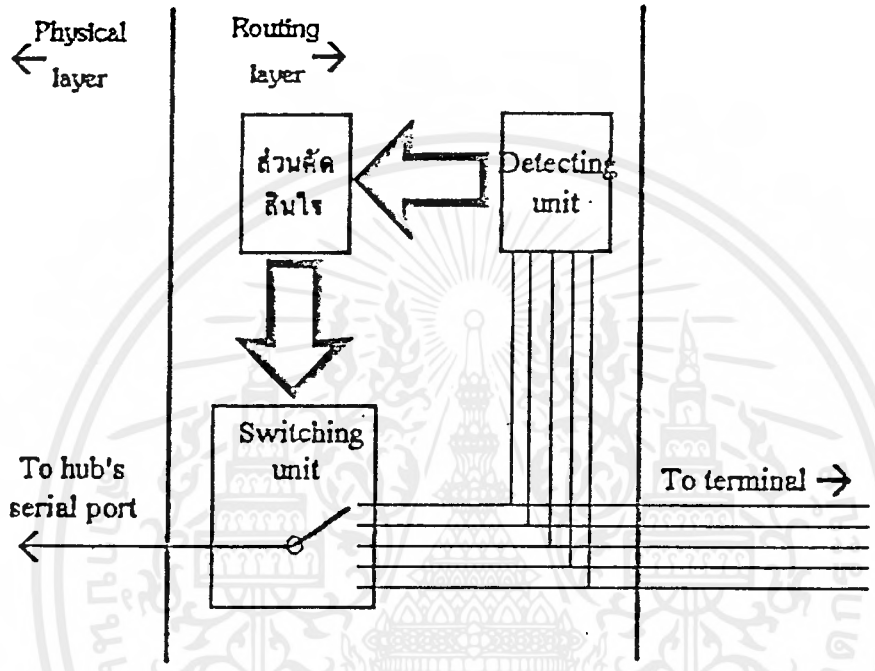
สำหรับวิธีการโปรแกรมและการทำงานโดยละเอียดของแอปปลิเคชั่น เช่น ไฟล์ ทรานซเฟอร์ อี-เมล์ และออน-ไลน์ คอมมิวนิเคชั่นของ Terminal นั้น จะอยู่ในโฟล ชาร์ทและโปรแกรมด้านหลัง ส่วนวิธีการที่ HUB จัดการกับแพคเก็ทต่างๆ ก็จะอยู่ในโฟล ชาร์ทของ HUB เช่นกัน ส่วนวิธีการใช้รีจิสเตรชั่น แพคเก็ทนั้น จะอยู่ในโฟล ชาร์ทที่ชื่อ ลอกอิน (Login) และ ลอกเอาท์ (Logout)

รูทติ้ง เลเยอร์ (ROUTING LAYER (FOR HUB))

ชั้นนี้เป็นเอกลักษณ์พิเศษของ HUB (Terminal จะไม่มีชั้นนี้) HUB จะเปรียบเสมือนกับมีเดียของเน็ตเวอร์ค และ Terminal คู่ใดต้องการสื่อสารกันจะต้องส่งข้อมูลผ่าน HUB ในช่วงขณะหนึ่งๆ จะมีเพียง 1 Terminal เท่านั้นที่สามารถเชื่อมต่อเพื่อส่งหรือรับของมูลกับ HUB หน้าที่ของชั้นรูทติ้ง ก็คือ เลือกเชื่อมต่อ Terminal เข้ากับ HUB เพราะฉะนั้น ชั้นอื่นๆ ของ HUB ก็จะไม่ต้องสนใจกับการสวิชชิ่งได้ เนื่องจากว่า เมื่อแพคเก็ทผ่านชั้นแอปปลิเคชั่นมาจนถึงชั้นฟิสิคัลแล้ว ชั้นรูทติ้งจะเป็นชั้นที่ดูแลให้ข้อมูลไปถึง Terminal ที่ถูกต้อง และชั้นรูทติ้งยังจะคอยดูแลเลือกรับข้อมูลจาก Terminal ที่สามารถส่งข้อมูลตามกฎของมีเดีย แอคเซส โปรโตคอลที่กำหนดขึ้น ซึ่งได้กล่าวไว้แล้วในชั้นดาต้าลิงค์

ชั้นรูทติ้งนี้ เป็นส่วนของฮาร์ดแวร์ที่ทำขึ้น ซึ่งจะทำงานร่วมกับซอร์ฟแวร์ ในฮาร์ดแวร์จะมีส่วนหนึ่งที่เรียกว่า ดีเทคติ้ง ยูนิท (Detecting Unit) ซึ่งจะทำหน้าที่เป็นตัวบอกซอร์ฟแวร์ว่า Terminal ใดสามารถได้ชันแนลสื่อสาร จากนั้น ซอร์ฟแวร์

จึงจะสั่งให้อาร์ดแวร์อีกส่วนหนึ่งที่เรียกว่าสวิชชิ่งเลือกเชื่อมต่อและรับส่งข้อมูลกับ Terminal นั้นๆ ดังรูป 3



รูปที่ 3 แสดงการหน้าที่ต่างๆ ของชั้นรูทติ้ง

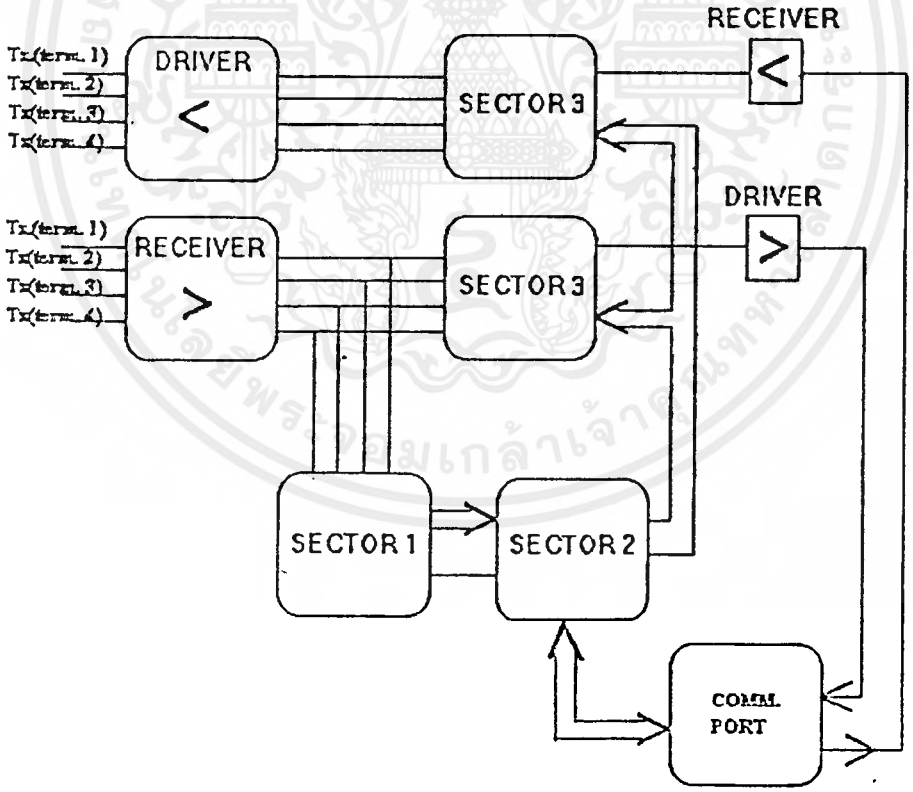
สำหรับการอินเตอร์เฟส (Interface) ระหว่างชั้นรูทติ้งกับชั้นฟิสิคัลนั้น จะใช้ซีเรียล พอร์ต(Serial Port) เพราะฉะนั้น ข้อมูลที่ชั้นรูทติ้งจัดให้ชั้นฟิสิคัลของ HUB ก็จะเป็นข้อมูลแบบอะซิงโครนัสเช่นเดียวกับชั้นฟิสิคัลใน Terminal ส่วนวิธีการโปรแกรมควบคุมชั้นรูทติ้งนี้จะอยู่ในโปรแกรมและโฟล ชาร์ทของ HUB

3.2 หลักการทำงานของฮาร์ดแวร์

เนื่องจากการที่เลือกใช้สัญญาณเพียง 3 เส้นคือ เส้นส่ง (Tx), เส้นรับ (Rx) และกราวนด์ (Gnd) ส่งผลให้การรับรู้ถึงการขอใช้ช่องสัญญาณ (ch) ที่ตัว HUB มีได้ทางเดียว คือ ต้องส่งสัญญาณมาบอกทาง Tx เท่านั้น และการที่ Terminal ใดๆ จะขอใช้ช่องสัญญาณ จะต้องส่งสัญญาณมาบอกโดยการส่ง ENQ มา ซึ่งจะทำให้ทาง HUB สามารถตรวจจับได้ แล้วจึงทำให้การสวิทช์ช่องสัญญาณถูกต้อง ดังนั้น ฮาร์ดแวร์จึงถูกแบ่งเป็น 3 ส่วนด้วยกัน คือ

1. ส่วนตรวจจับการขอใช้ช่องสัญญาณ
2. ส่วนควบคุมการใช้ช่องสัญญาณ
3. ส่วนที่ทำหน้าที่สวิทช์ Terminal เข้ากับ HUB

ลักษณะความสัมพันธ์ของแต่ละส่วนจะเป็นดังรูปด้านล่าง



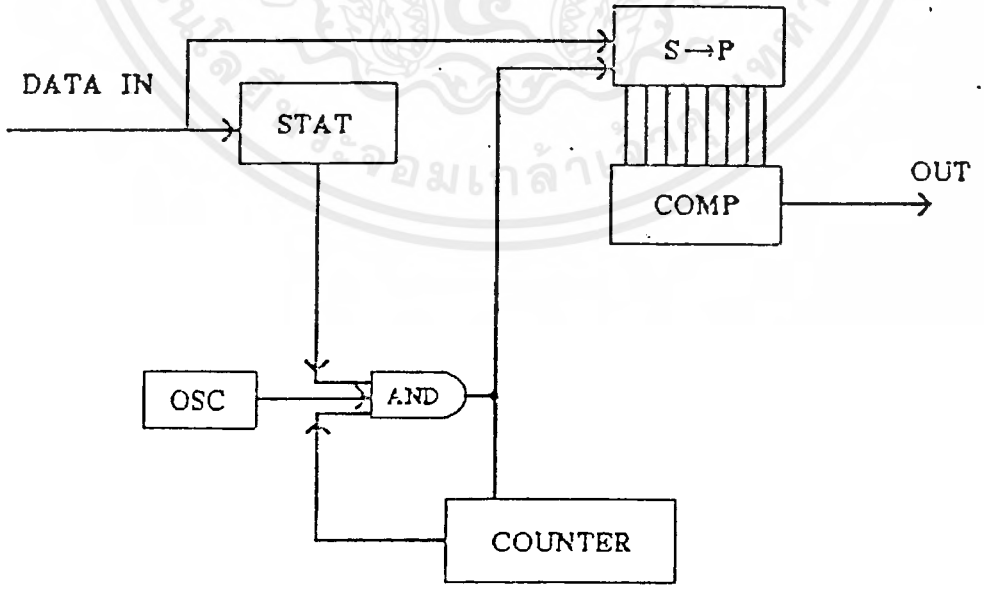
รูปที่ 1 แสดงบล็อค ไดอะแกรม รวมของฮาร์ดแวร์

ลักษณะหรือขั้นตอนการทำงานแต่ละส่วนเป็นดังต่อไปนี้ คือ

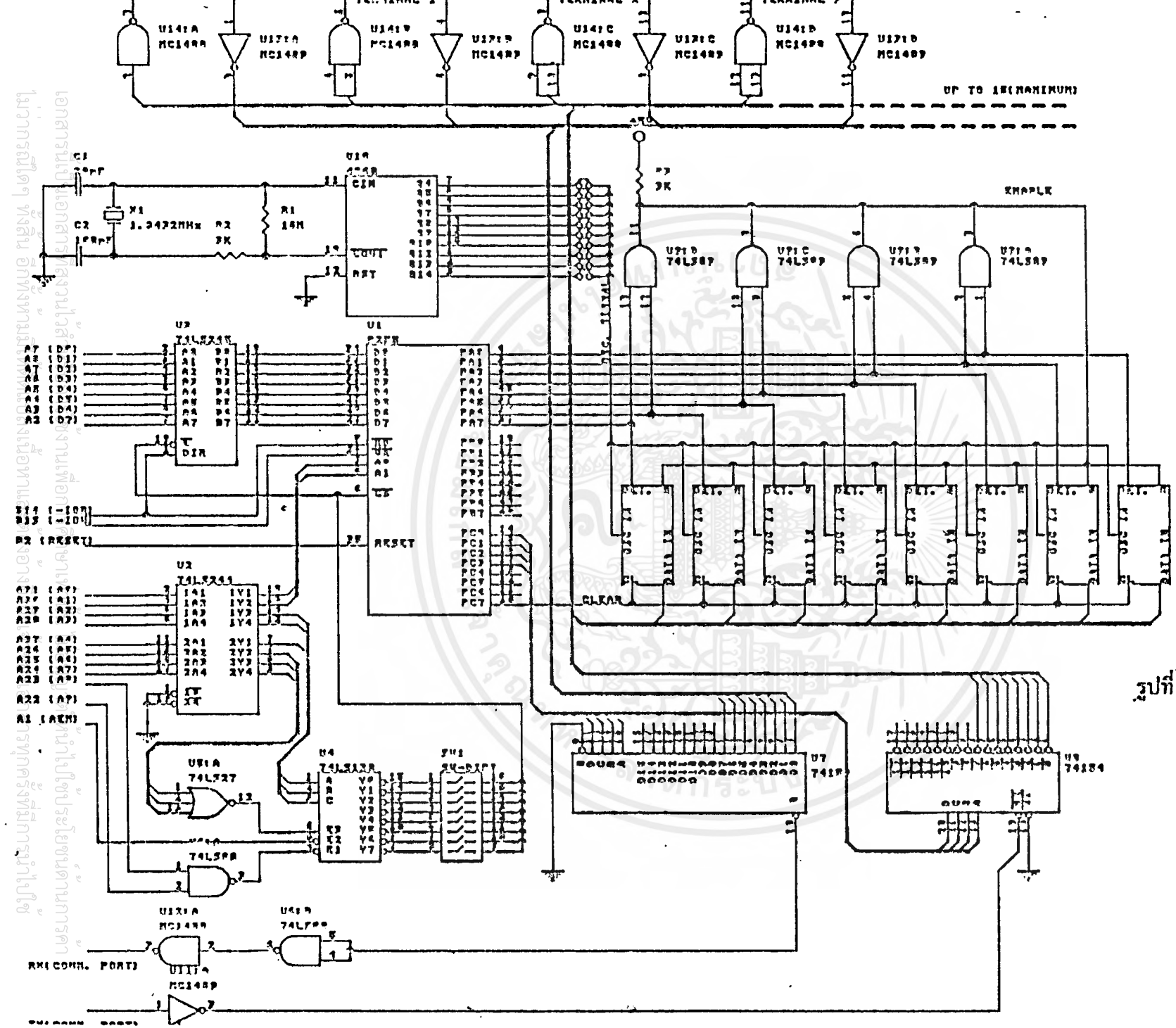
1. ส่วนตรวจจับการขอใช้ ช่องสัญญาณ.

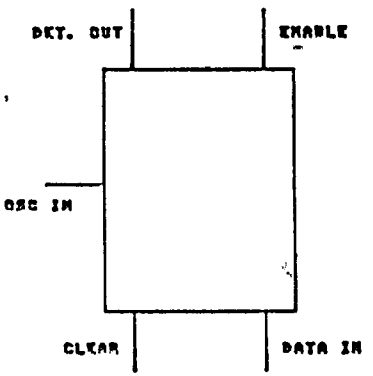
ส่วนตรวจจับ (Detector) จะเริ่มต้นทำงานด้วยการเคลียร์การทำงานของทุก ยูนิต (unit) ก่อน ซึ่งผลจากการเคลียร์จะทำให้ ผลของการตรวจจับ (Detect Out) ของแต่ละยูนิตมีค่าเป็น ไฮ เป็นผลให้สถานะของสัญญาณเอนาเบิล ซึ่งเกิดจากตรรกของการต่อเกท (gate) แบบโอเพ็น คอลเลกเตอร์ (Open Collector) (U9A ถึง U9D)มีค่าเป็น ไฮ ส่งผลให้ขาเอนาเบิลของ ดิเทค ยูนิต (Detect Unit) ทุกยูนิตมีค่าเป็น ไฮ ซึ่งหมายถึงว่า เมื่อทำการเคลียร์ส่วนตรวจจับจะมีผลให้ส่วนตรวจจับพร้อมที่จะทำงานทันที และการทำงานของส่วนตรวจจับก็คือ เมื่อมีข้อมูลเข้าไปในส่วนตรวจจับยูนิตใดๆ และส่วนตรวจจับยูนิตนั้นๆ สามารถตรวจจับได้ว่า ข้อมูลที่เข้ามานั้นเป็นไปตามที่กำหนด (ในกรณีนี้กำหนดให้เป็นเลข 5 ฐาน16 (5_{16}) ซึ่งเป็นรหัสเอ็นไควร์รี่ (ENQ)) ก็จะทำให้ผลการตรวจจับของหน่วยตรวจจับนั้นๆ มีค่าเป็นโล และส่งผลให้สัญญาณเอนาเบิลเป็น โล ทันที มีผลให้หน่วยตรวจจับทุกหน่วย ดิสเอเบิล (Disable) ซึ่งจะหมายความว่า ที่ขณะเวลาหนึ่ง หากมีการส่ง ENQ ออกมาจาก Terminal ใดๆ ก็ตามเป็น Terminal แรกและหน่วยตรวจจับสามารถดิเทคได้แล้ว Terminal อื่นๆ จะถูกไม่สนใจอีก

ลักษณะวงจร



รูปที่ 2 แสดงบล็อค ไดอะแกรม ของส่วนตรวจจับสัญญาณเอนาเบิล





ADDRESS DETAIL ON DETECT UNIT

รูปที่ 3 แสดงวงจรทั้งหมดของฮาร์ดแวร์

RCH/MP

ลักษณะของวงจรสามารถแบ่งเป็นส่วนต่างๆ ดังนี้

-1 ส่วนกำเนิดความถี่ที่มีความสัมพันธ์กับความเร็วของการส่งข้อมูล

-2 ส่วนตรวจจับ สตาร์ท บิท (Start Bit)

-3 ส่วนแปลงข้อมูลจากแบบอนุกรม(SERIAL)เป็นแบบขนาน(PARALLEL) (S-P)

-4 ส่วนเปรียบเทียบข้อมูล

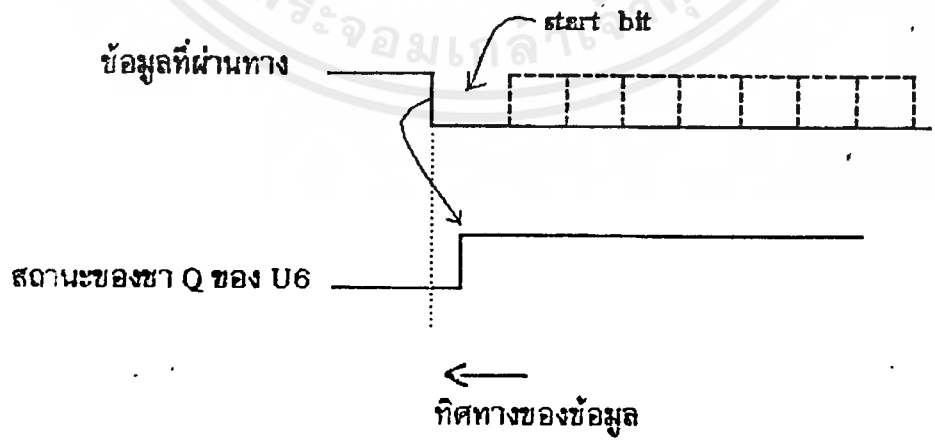
-5 ส่วนวงจรนับ

-1 ส่วนกำเนิดความถี่

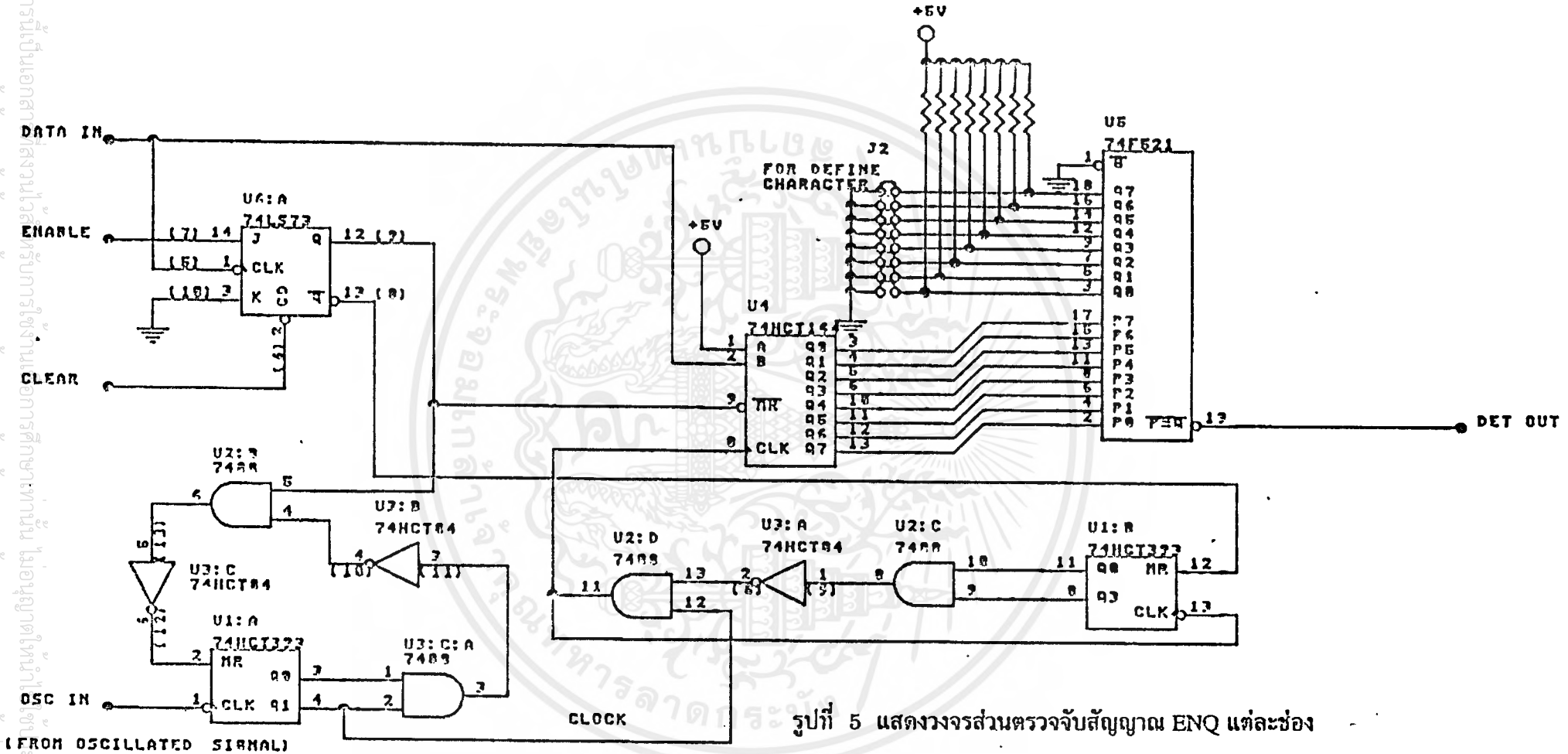
ในส่วนนี้จะใช้ ไอซี(IC) 4060 (พิจารณาวงจร STAR บริเวณ U10 ประกอบคำอธิบาย) ซึ่งมีหน้าที่เป็นทั้ง เคาเตอร์ (counter) และ ออสซิเลเตอร์ (Oscillator (OSC)) และให้ ไอซี ตัวนี้ออสซิเลทที่ความถี่ 1.8432 เมกกะเฮิร์ท (MHz) และผลจากการหารความถี่ของไอซี จะทำให้ออสซิเลทสัญญาณความถี่ 115.2 กิโลเฮิร์ท (KHz) ออกมาที่ขา Q4(ขา7) ซึ่งความถี่ที่ได้นี้จะถูกนำไปหาร 3 อีก ด้วยไอซี 74HCT 393 (U1A) โดยมี U3A, U3A, U2B และ U3C เป็นตัวควบคุมการทำงาน

-2 ส่วนตรวจจับ สตาร์ท บิท

ในส่วนนี้จะใช้ (U6A) เป็นตัวตรวจจับโดยกำหนดให้ขา เค(K) เป็นโล ส่วนขา เจ(J) นั้นจะถูกต่อกับเอนาเบิล ซึ่งเมื่อเริ่มทำงานครั้งแรกสุดจะมีค่าเป็นไอ ดังนั้น เมื่อมีสตาร์ทบิทเข้ามาที่ขาคล็อค จะทำให้ คิว(Q) มีสถานะเป็น ไอ ดังรูปที่ 4



รูปที่ 4 แสดงการตรวจจับ start bit



รูปที่ 5 แสดงวงจรส่วนตรวจจับสัญญาณ ENQ แต่ละช่อง

DETECT UNIT

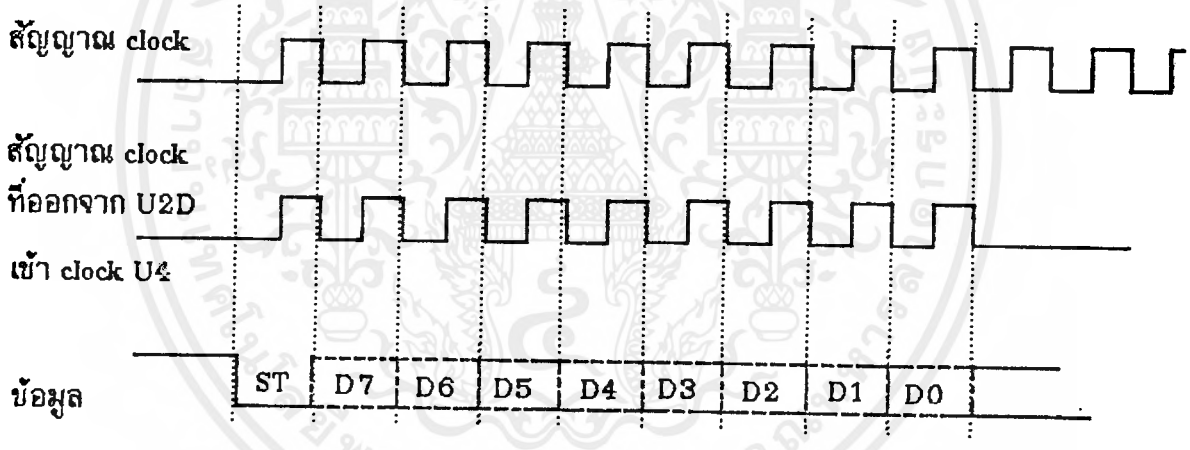
การเปลี่ยนแปลงสถานะของ Q ใน U6A จะมีผลให้ออสซิเลทเต็ท ซิกแนล อินพุท (Oscillated Signal Input) ถูกหารโดย U1A และได้สัญญาณ คล็อค (clock) ออกมาทางขา Q1

-3 ส่วนแปลงข้อมูลจากแบบอนุกรมมาเป็นแบบขนาน

สัญญาณคล็อคจาก U1A จะผ่าน U2D ไปยัง U4 ซึ่งทำหน้าที่แปลงข้อมูลจากแบบอนุกรมเป็นแบบขนาน และในขณะเดียวกันสัญญาณคล็อคที่เข้ามาจะถูกควบคุมจำนวนลูกคลื่นด้วยส่วนวงจรนับ

-4 ส่วนวงจรนับ

สัญญาณคล็อคที่ออกจาก U2D ส่วนหนึ่งจะถูกป้อนกลับนำไปเข้า U1B ซึ่งจะทำหน้าที่นับจำนวนลูกคลื่นให้ได้ 9 ลูกคลื่น (เนื่องจาก เริ่มทำการนับตั้งแต่บิทเริ่มต้น 1 บิทและบิทข้อมูลอีก 8 บิท) โดย U1B จะทำงานร่วมกับ U2C และ U3A ซึ่งเมื่อนับได้ครบ 9 แล้วจะทำให้ U2D เป็นโล ทำให้ไม่มีสัญญาณคล็อคผ่านเข้าไปที่ U4 อีก ดังรูป



รูปที่ 6 แสดง สัญญาณ clock ที่ออกจาก U2D เข้า clock U4

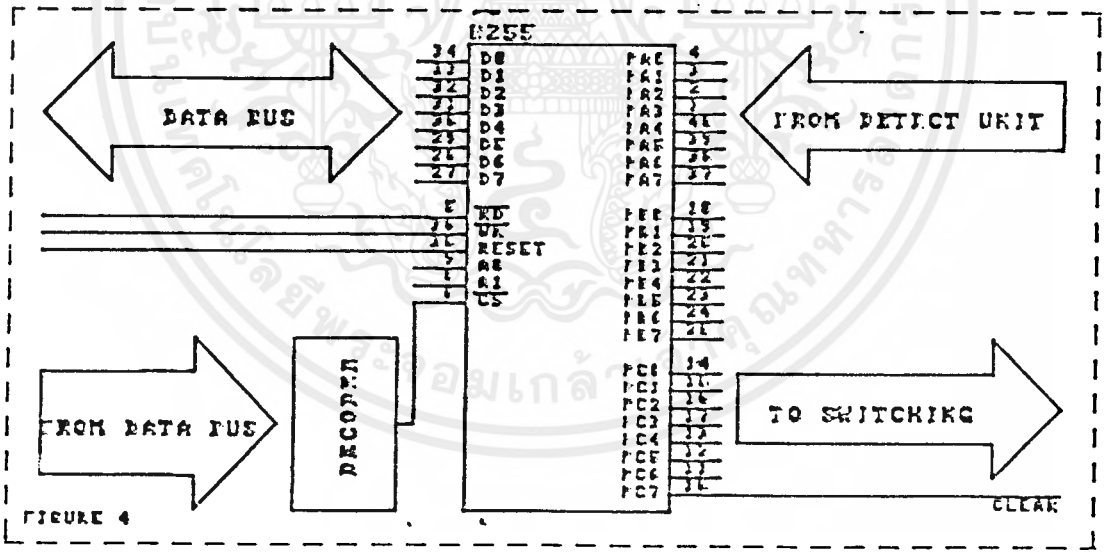
เมื่อถึงขั้นนี้แล้ว จะเป็นการสิ้นสุดของการแปลงข้อมูลจากแบบอนุกรมไปเป็นแบบขนาน ซึ่งการทำงานหลังจากนี้จะเป็นหน้าที่ของ

-5 ส่วนเปรียบเทียบข้อมูล

โดยมี U5 ซึ่งทำหน้าที่เป็น คอมพาราเตอร์ (comparator) จะทำการเปรียบเทียบข้อมูลที่ได้จากส่วนที่ 3 กับข้อมูลที่ถูกกำหนดไว้โดย จัมเปอร์2 (Jumper2) และจะให้ผลการเปรียบเทียบออกมา เพื่อให้ส่วนควบคุมอ่านเข้าไปวิเคราะห์ต่อไป

2. ส่วนควบคุมการทำงาน

ส่วนนี้จะประกอบไปด้วยไอซี 8255 เป็นหลัก โดยไอซีจะถูกตั้งให้ทำงานโดยกำหนดให้พอร์ต เอ(port A) และ บี (B) เป็นอินพุท พอร์ต ส่วนพอร์ต ซี (C) เป็นเอาท์พุท พอร์ต โดยพอร์ต A และ พอร์ต B เป็นพอร์ตที่ทำการรับข้อมูลจากส่วนตรวจจับการขอใช้ช่องสัญญาณเข้ามา เพื่อให้ซอร์ฟแวร์ของชั้นรูทติ้งทำการคัดเลือก Terminal ที่มีสิทธิใช้ช่องสัญญาณ แต่หากเกิดกรณีที่มีการขอใช้ช่องสัญญาณมากกว่า 1 Terminal แล้ว ชั้นรูทติ้งจะทำการจัดลำดับตามไพออร์ริตี้ (Piority) โดยรับจัดการให้กับ Terminal ที่มีค่าแอดเดรส(Address) น้อยกว่า จากนั้น ชั้นรูทติ้งก็จะส่งผลที่เป็นค่าของ Terminal ที่ได้รับเลือกให้ใช้ช่องสัญญาณก่อนออกมาทางพอร์ต C ด้านต่ำกว่า และข้อมูลทางพอร์ตนี้จะถูกส่งไปให้ส่วนสวิทช์ต่อไป นอกจากนี้ บิทที่ 7 ของพอร์ต C ยังทำหน้าที่ในการส่งสัญญาณเคลียร์ ไปให้กับส่วนตรวจจับช่องสัญญาณในการที่จะเริ่มต้นการตรวจจับการขอใช้ช่องสัญญาณใหม่ด้วย ลักษณะวงจรของส่วนนี้ถูกแสดงอยู่ด้านล่าง



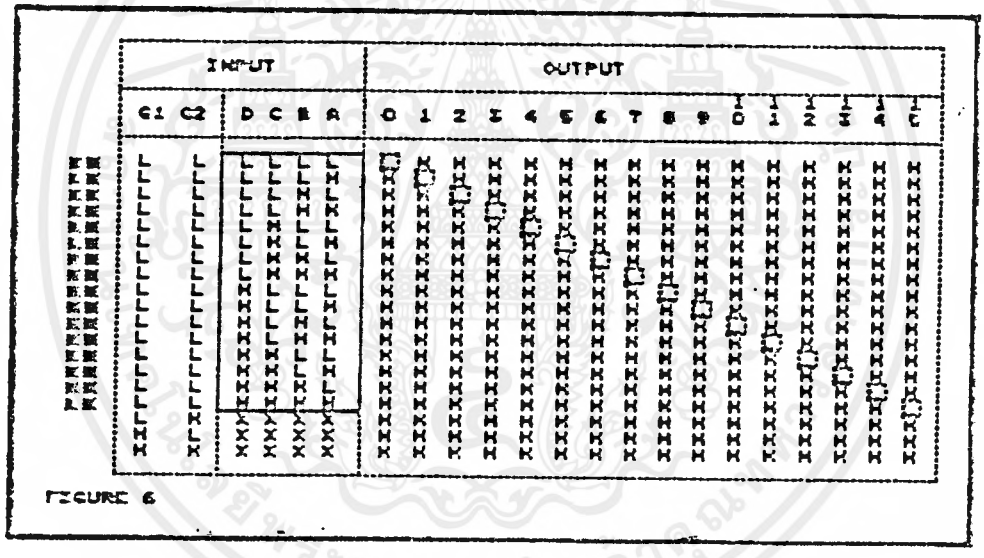
รูปที่ 7 แสดงลักษณะของวงจรส่วนควบคุมการทำงาน

3. ส่วนสวิทช์ Terminal เข้ากับคอม พอร์ต(comm port) ของ HUB

ในส่วนนี้จะมีสวิทช์ที่ควบคุมการไหลของข้อมูลอยู่ 2 แบบ คือ

- การไหลของข้อมูลจาก Terminal เข้าสู่ HUB
- การไหลของข้อมูลจาก HUB เข้าสู่ Terminal

ในแบบแรกนั้นไม่มีปัญหา เนื่องจากได้เลือกใช้ ไอซี เบอร์ 150 ซึ่งเป็น 16 line to 1 line Data Selector แต่สำหรับแบบที่ 2 นั้นไม่สามารถหา ไอซี ที่ทำหน้าที่เป็น 1 line to 16 line Data Selector ได้ ดังนั้น จึงได้ดัดแปลงโดยนำไอซี เบอร์ 154 ซึ่งมีหน้าที่ 4 line to 16 line Decoder มาใช้แทน โดยพิจารณาจากฟังก์ชั่น เทเบิล (Function Table) คือ



รูปที่ 8 แสดงฟังก์ชั่น เทเบิล

หากสภาวะของ G_1 และ G_2 เป็น โล ทั้งคู่ (*) จะเห็นได้ว่าเอาท์พุทที่ถูกชี้โดยอินพุทจะมีสภาพเป็น โล และหาก G_1 หรือ G_2 ก็ตามเป็น ไฮ จะเห็นได้ว่าเอาท์พุททุกตัว ก็จะกลายเป็น ไฮ ทันที

เนื่องจากในสภาพปกติของสัญญาณ Tx จะมีสภาพเป็น ไฮ ตลอด ดังนั้นหากให้อินพุทชี้ไปที่ค่าใดค่าหนึ่ง สมมติว่าเป็น 8 จะได้ผลดังรูป

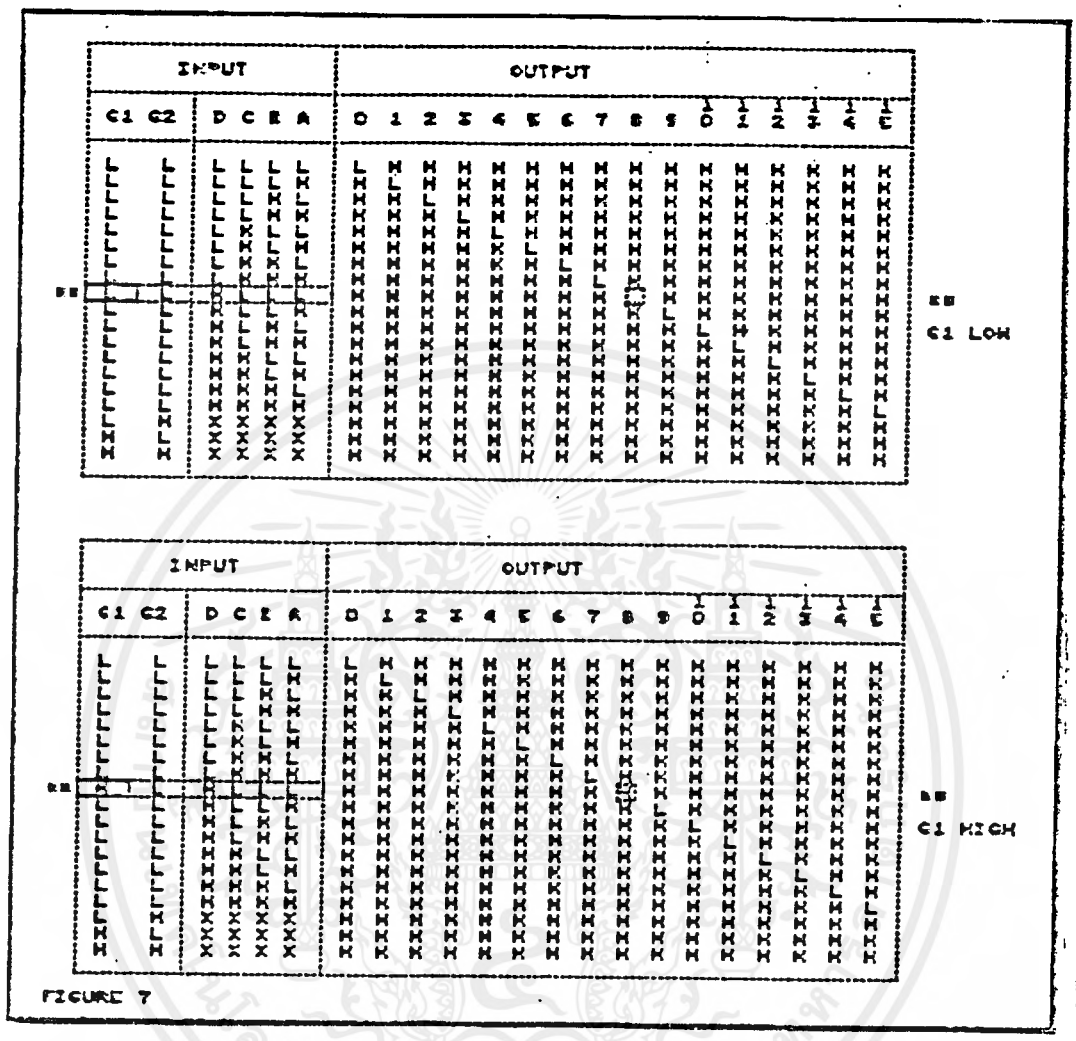
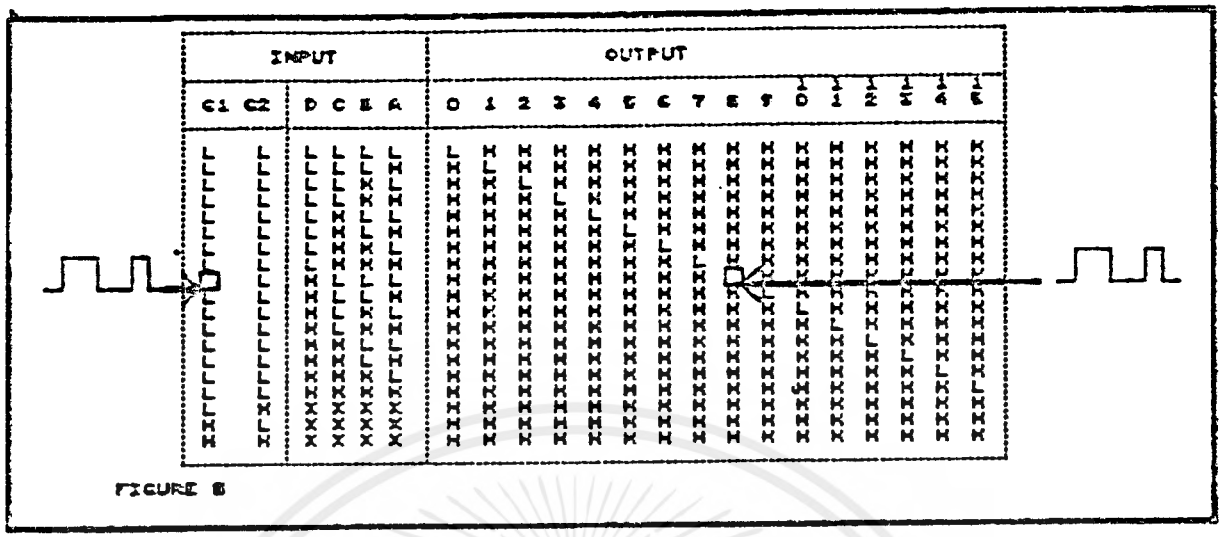


FIGURE 7

รูปที่ 9 แสดงฟังก์ชั่น เทเบิล ที่เอาท์พุทที่ 8

กำหนดให้ G_2 เป็น โล และ G_1 เป็น โล ด้วย จะเห็นว่าเอาท์พุท 8 เป็น โล ด้วย และในทางกลับกัน ให้ G_1 เป็น ไฮ เอาท์พุท 8 จะกลายเป็น ไฮ ด้วย ซึ่งทั้งๆที่ปกติแล้วกรณีนี้จะเป็นกรณีที่เกิดการดิสเอเบิลของ ไอซี นี้ ตามหน้าที่ปกติของมัน แต่เนื่องจากภาวะนี้กลายเป็นภาวะปกติของสัญญาณ Tx ด้วย คือ ไม่มีผลกระทบต่อสัญญาณ Tx ของ Terminal เครื่องอื่นๆ ดังนั้น หากทำการเปลี่ยนสถานะของ G_1 เป็นอะไรก็ตาม เอาท์พุทที่ถูกชี้โดยอินพุทจะถูกเปลี่ยนแปลงตามไปด้วย ดังเหตุผลดังกล่าวข้างต้น



รูปที่ 10 แสดงฟังก์ชั่น เทเบิล ที่เปลี่ยนแปลง

ดังนั้น ไอซี ตัวนี้ จึงถูกนำมาทำหน้าที่เป็น 1 line to 16 line Data Selector โดยจะรับหมายเลขของ Terminal ที่จะถูกสวิทช์มาจากส่วนควบคุมอีกที

เพราะฉะนั้น สามารถกล่าวโดยสรุปได้ว่า การทำงานของฮาร์ดแวร์มีลักษณะ คือ

ส่วนที่ 1 หรือส่วนตรวจจับจะทำการตรวจจับการขอใช้ช่องสัญญาณและจะคงผลไว้ตราบเท่าที่ส่วนควบคุมมาทำการอ่านผลนี้ไปและเคลียร์ส่วนตรวจจับ หลังจากนั้น ส่วนตรวจจับพร้อมที่จะทำงานอีกต่อไป

ส่วนที่ 2 จะนำผลจากส่วนแรกไปทำการประมวลผลว่าจะตัดสินให้ Terminalใดได้ใช้ช่องสัญญาณ แล้วจะส่งผลไปให้ส่วนที่ 3

ส่วนที่ 3 จะทำการสวิทช์ Terminal นั้นๆ เข้ากับ HUB เพื่อทำการส่งข้อมูลเข้ามายัง HUB ได้

ผลการทดลองและสรุป

เน็ตเวอร์คนี้ ได้ถูกทดลองโดยนำเครื่องพีซี 3 เครื่อง ซึ่งเป็น HUB 1 เครื่องอยู่ตรงกลาง และ Terminal 2 เครื่อง เชื่อมต่อกับ HUB ด้านละเครื่อง และดำเนินการทดลองดังนี้

1. ให้ Terminal เครื่องที่ 1 ส่งข้อมูลในลักษณะ ไฟล์ ทรานซเฟอร์ ผ่าน HUB ไปให้ Terminal ที่ 2
2. ทำการส่ง อี-เมล์ ด้วยการให้ Terminal เครื่องที่ 1 ส่งข่าวสารไปให้ Terminal เครื่องที่ 2 โดยฝากไว้ที่ HUB ก่อน แล้วให้ Terminal ที่ 2 ส่งข้อมูลควบคุมมาขอข่าวสารเกี่ยวกับไฟล์กับ HUB ว่ามี เมล์ มาถึงตนหรือไม่ ถ้ามี Terminal 2 จะส่งข้อมูลมาขอรับเมล์นั้นไป
3. ให้ Terminal ที่ 1 และ 2 ส่งข้อมูลแบบออน-ไลน์ คอมมิวนิเคชั่น
4. ให้Terminal1 ส่งข้อมูลควบคุมมาขอข่าวสารเกี่ยวกับผู้ใช้เครื่องขณะนั้นกับHUB

จากการทดลองทั้ง 4 ขั้นตอน และเนื่องจากเน็ตเวอร์คได้ใช้หลักการ เลเยอร์ โปรโตคอล ดังนั้น จึงต้องมาดูผลงานของเลเยอร์เหล่านี้ ซึ่งได้ผล คือ ในแต่ละเลเยอร์สามารถทำงานในเลเยอร์ของมันเองได้สมบูรณ์ แต่สำหรับการสื่อสารกันระหว่างเลเยอร์นั้น ยังมีการสัมพันธ์ไม่ดีนัก กล่าวคือ เมื่อมีข้อมูลถูกส่งมาจากแอปปลิเคชั่นเลเยอร์ จาก Terminal หนึ่ง ข้อมูลนั้นจะถูกจัดการไปถึงชั้นดาต้าลิงค์ของ Terminal รับและถูกกักไว้ ทำให้ชั้นดาต้าลิงค์ไม่สามารถติดต่อกับชั้นแอปปลิเคชั่นทางด้านรับได้

สรุป

เนื่องจากการทำงานเพื่อทดลองเน็ตเวอร์คนี้ สามารถแบ่งกล่าวเป็น 2 ส่วน คือ ส่วนของฮาร์ดแวร์ และส่วนของแอปปลิเคชั่น ซอร์ฟแวร์ ซึ่งจะขอสรุปเป็นะ

- ส่วนของฮาร์ดแวร์

การทำงานของฮาร์ดแวร์ในเน็ตเวอร์คนี้ได้เปลี่ยนแปลงวงจรบางส่วนในส่วนตรวจจับสัญญาณ จากเดิมนั้น การตรวจจับจะตรวจจับจากช่องสัญญาณเพียงบิทเดียว ซึ่งถ้าหากมีนอยซ์ (noise) หรือสัญญาณที่ไม่ต้องการเข้ามาในบิทใดๆ จะทำให้ส่วนตรวจจับสามารถตรวจจับสัญญาณได้ทันที ซึ่งเป็นข้บกพร่องที่สำคัญ ดังนั้น จึงได้ทำการเปลี่ยนแปลงวงจรส่วนนี้โดยให้มีการตรวจจับสัญญาณทั้ง 8 บิท ซึ่งจะทำให้เป็ฯการตรวจจับที่แน่นอนกว่าเดิมมาก

- ส่วนแอปปลิเคชั่นซอร์ฟแวร์

การเขียนแอปปลิเคชั่นในเน็ตเวอร์คนี้ จะเน้นไปในส่วนของการสื่อสารของระบบมากกว่าส่วนยูสเซอร์ อินเตอร์เฟส อีกทั้งโปรแกรมที่เขียนนี้ ได้ใช้ภาษาซีและแอสแซมบลี แต่มิได้อธิบายหลักการเขียนของภาษาทั้ง 2 ซึ่งหากสนใจก็สามารถศึกษาได้จาก ซอร์สโค้ด (source code) ที่อยู่ในบทแทรกและจากหนังสือที่เกี่ยวกับภาษาทั้ง 2 ที่กล่าวไว้ในบรรณานุกรมได้ นอกจากนี้ โปรแกรมยังอาจถูกเขียนด้วยภาษาอื่นๆ ได้ เพียงแต่ต้องมีโปรโตคอลที่ถูกต้องตามที่ได้อธิบายมาแล้วเท่านั้น

บทที่ 5

บทแทรก

วงจรหลีกเลี่ยงการชนสำหรับโครงข่ายรูปดาว

(STAR NETWORK WITH COLLISION-AVOIDANCE CIRCUITS)

แคร์เรียร์-เซนส์ มัลติเปิล แอคเซส โลคัล แอเรีย เน็ตเวอร์ค วิดธ์ คอลลิชั่น ดีเทคชั่น (Carrier-Sense Multiple-Access local-area network with collision detection (CSMA-CD)) ได้นำเทคโนโลยีของคลื่นแสงมาใช้ แต่การใช้ตัวนำแสงยังเป็นปัญหาอยู่ เนื่องจากเน็ตเวอร์คไม่สามารถใช้ประโยชน์จากฟูลแบนด์วิดธ์ (full bandwidth) ของตัวนำแสง นอกจากนี้ ข้อเสียอีกอย่างของซีเอสเอ็มเอ-ซีดี (CSMA-CD)เน็ตเวอร์ค คือ ต้องการแพคเก็ทที่ใหญ่ เมื่อทำงานที่อัตราบิทสูงๆ เช่น สตาร์ เน็เวอร์ค ที่ยาว 1 Km ทำงานที่ 150 Kb/s หากใช้ 256 b/p จะใช้ประสิทธิภาพเพียง 5% หากใช้ 4096 b/p จะใช้ประสิทธิภาพ 44% และหากใช้ 65,536 b/p จะใช้ประสิทธิภาพ 93%

ในกรณีที่ใช้ประสิทธิภาพต่ำ จะมีปัญหาเรื่องความไม่เสถียร ซึ่งสามารถเห็นได้เมื่อเน็ตเวอร์คมีปริมาณการใช้เท่ากับความจุชั้นแนลภายใต้เงื่อนไขที่แทรฟฟิก (Traffic) หนาแน่น แต่ถ้าปราศจากข้อจำกัดของการจราจรแล้วเน็ตเวอร์คจะไม่เสถียร และเกิดการชนขึ้น

ซีเอสเอ็มเอ-เอดี (CSMA-AD) นี้ เป็นวิธีใหม่ที่จะจำกัดการเกิดการชน ถึงแม้ว่าจะมีการส่งหลายแพคเก็ทพร้อมๆ กันภายในเน็ตเวอร์ค กรรมวิธีก็คือ จะมีวงจรที่จะคอยทำหน้าที่ไม่ให้มีการชนเกิดขึ้นที่ เซ็นเตอร์ โนด (Center Node) โดยมันจะทำตัวคล้ายตำรวจจราจรซึ่งจะคอยควบคุมให้แพคเก็ทผ่านไปถึงโนด (node) เมื่อโนดอยู่ในช่วง "ไอเดิล" ("idle") และจะกั้นแพคเก็ทที่มาเมื่ออยู่ในช่วง "บิซี่" ("busy") แพคเก็ทที่ผ่านตำรวจจราจรจะถูกส่งไปยังเครื่องปลายทางทุกเครื่องรวมทั้งเครื่องที่ส่งด้วย และแพคเก็ทที่ถูกกั้นจะต้องส่งใหม่จนกว่าแพคเก็ทนั้นจะผ่านไปถึงโนด จึงถือเป็นการส่งสำเร็จ

การเพิ่มวงจรที่หลีกเหลี่ยงการชนมีข้อดีคือ เน็ตเวอร์คจะมีสภาพเหมือน ซีโร-เลนธ์ ซีเอสเอ็มเอ-ซีดี (zero-length CSMA-CD) และจะมีประสิทธิภาพและความเสถียรมากกว่า เนื่องจากคุณลมบัติเหล่านี้จะช่วยสนับสนุนเน็ตเวอร์คที่ใช้อัตราบิทสูงๆ และยังเหมาะกับการใช้คลื่นแสงในแลน (LAN) ด้วย

สถาปัตยกรรมรองเน็ตเวอร์ค

จากรูปที่1 แสดงถึงแผนที่ผู้ใช้ทุกคนถูกเชื่อมต่อที่ เซ็นทรัล โนด ในรูปแบบดาว โดยวงจรเชื่อมต่อ ยูไอซี (UIC) การเชื่อมต่อนี้จะประกอบด้วย ตัวส่ง (T), ตัวรับ (R) และวงจรโลจิก (logic) ซึ่งจะเอนาเบิล (enable) ตัวรับที่รับแพคเก็ทแรกในขณะที่ ไอเดิล ส่วนโนดจะประกอบไปด้วยบัส (bus) ที่มีความยาวสูงสุดเท่ากับความยาวที่ทำให้เวลาบัส ราวด์ ทริป (bus round trip) เท่ากับ ช่วงเวลา 1 บิท เครื่องปลายทางแต่ละเครื่องจะถูกเชื่อมต่อเข้ากับยูไอซี โดยยูสเซอร์ ลิงค์ (USER'S LINK)

โปรโตคอล (PROTOCOL)

โปรโตคอล ระหว่างผู้ใช้กับเน็ตเวอร์คสามารถสรุปขั้นต้นได้ดังนี้ :

วงจร ยูไอซี จะกั้นแพคเก็ทที่มาถึง เซ็นทรัล โนด ก่อนที่มันจะทำให้เกิดการชน แล้วผู้ใช้ก็จะส่งแพคเก็ทนั้นใหม่จนกว่าจะส่งผ่านโนดได้สำเร็จ

รายละเอียดของโปรโตคอล ได้ถูกแบ่งเป็นยูสเซอร์ โปรโตคอล (User Protocol) และ โนด โปรโตคอล (Node Protocol)

- ยูสเซอร์ โปรโตคอล (User Protocol)

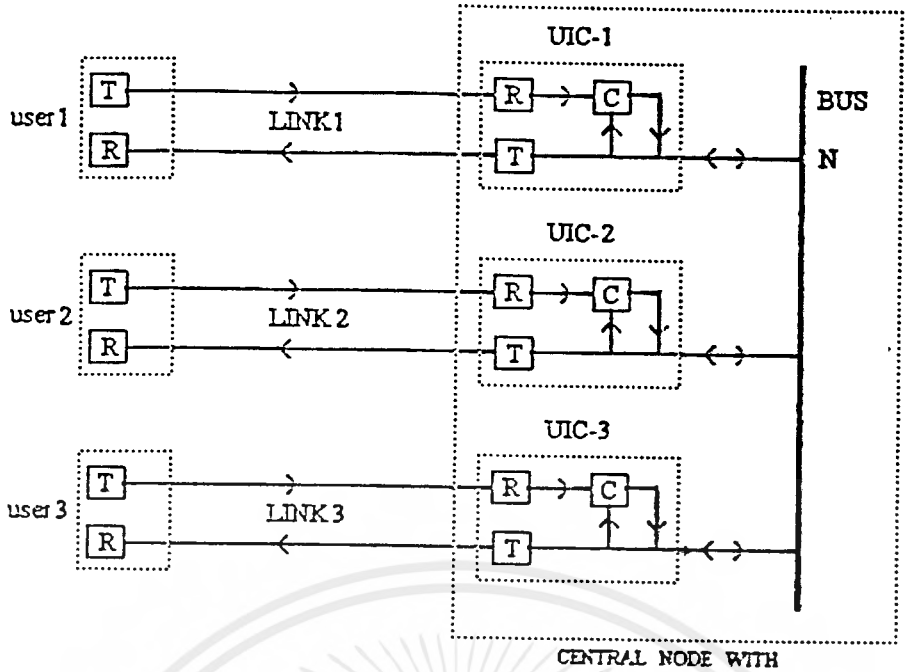
ผู้ใช้จะมี อินพุท (i/p) และ เอาท์พุท บัฟเฟอร์ (o/p Buffer) ที่จะเก็บแพคเก็ทที่จะส่งและรับมา เมื่อผู้ใช้มีแพคเก็ทที่เอาท์พุท บัฟเฟอร์ เพื่อที่จะส่งมีการปฏิบัติดังนี้

1. จะส่งแพคเก็ทออกไป
2. มันจะรอ T_{RT} ยูสเซอร์ ราวด์ ทริป ไทม์ (User's Round Trip Time) : เวลาที่ใช้ในการเดินทาง 1 รอบ
3. มันจะตรวจที่ อินพุท บัฟเฟอร์ (i/pbuffer) ว่า หากได้รับแพคเก็ทจะวิเคราะห์ว่ามาจากไหน และจะไปไหน
4. ถ้าไม่ได้รับแพคเก็ท หรือ แพคเก็ทที่รับมาไม่เหมือนกับแพคเก็ทที่ส่ง หมายความว่า แพคเก็ทที่ส่งนั้นถูกกั้นที่โนด และแพคเก็ทที่รับได้นั้นเป็นของผู้ใช้อื่น จากนั้น ขบวนการทั้งหมดจะเกิดขึ้นซ้ำๆ จนกว่าจะส่งผ่านโนดสำเร็จ

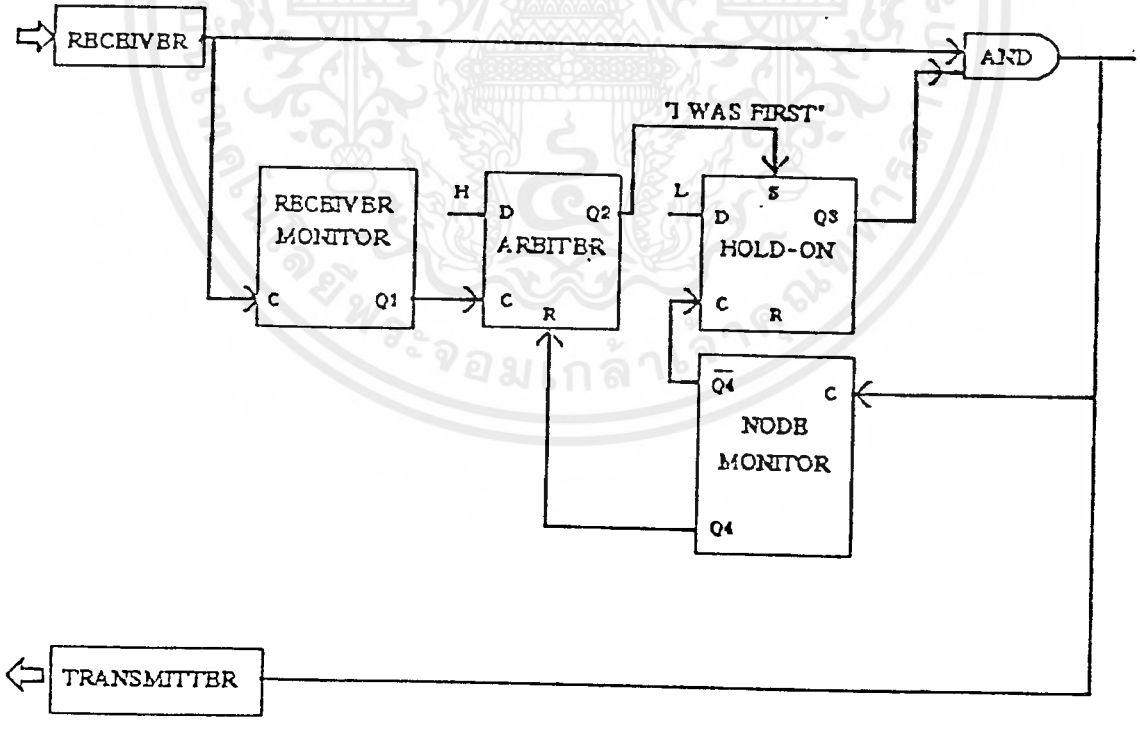
- โนด โปรโตคอล (NODE PROTOCOL)

ตัวรับและตัวส่งที่ เซ็นทรัล โนด ถูกควบคุมการทำงานดังนี้ :

1. ตัวส่งทุกตัวจะถูกเชื่อมต่อกับโนด N ตลอดเวลา เพราะฉะนั้น ผู้ใช้ทุกคนควรจะได้รับแพคเก็ทที่ถูกส่งกระจายจากโนดตลอดเวลา



รูปที่ 1 แสดงการเชื่อมต่อของเน็ตเวอร์ค



รูปที่ 2 แสดงวงจร ยูสเซอร์ อินเตอร์เฟส

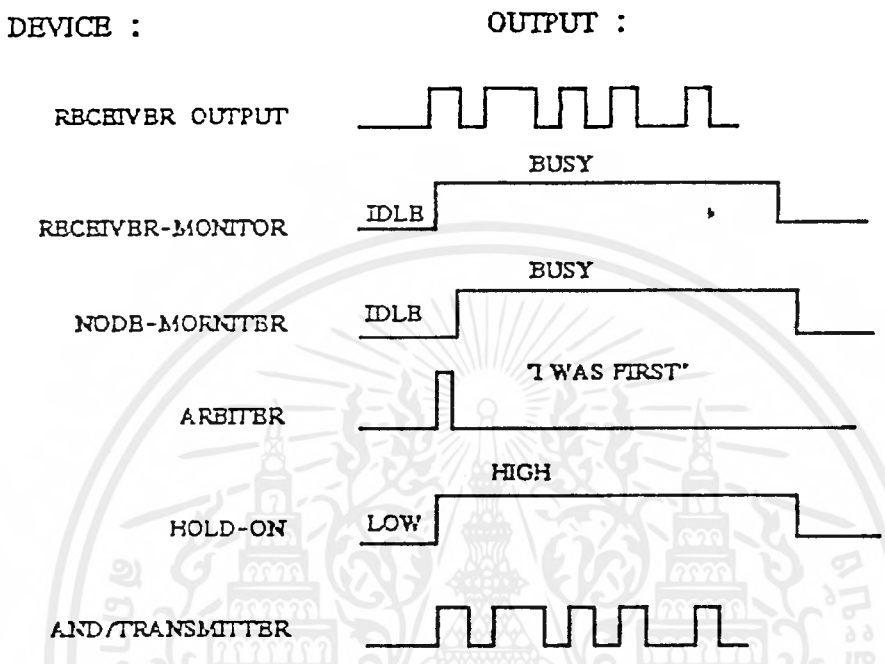
เช่นกันด้วย

2. ตัวรับทุกตัวจะไม่ได้ถูกต่อกับโนด
3. ตัวรับและโนด จะตรวจสอบสถานะ "ไอเดิล" หรือ "บิซี่" ตลอดเวลา
4. เมื่อแพคเก็ทเดินทางมาถึงตัวรับ ตัวรับจะเปลี่ยนเป็น "บิซี่" การเปลี่ยนสถานะจาก "ไอเดิล" เป็น "บิซี่" ในขณะที่โนด"ไอเดิล" ตัวรับจะถูกเชื่อมต่อกับโนด แต่การเปลี่ยนสถานะจาก "ไอเดิล" เป็น "บิซี่" ในขณะที่โนด มีสถานะ "บิซี่" จะไม่ได้รับความสนใจ และตัวรับยังคงไม่ถูกเชื่อมต่อกับโนด ทั้งหมดสรุปได้ว่าแพคเก็ทแรกที่ป่านเข้ามาเท่านั้นที่ตัวรับถูกเชื่อมต่อ นอกจากนั้น จะไม่ถูกเชื่อมต่อกับโนด
5. สถานะของโนดจะถูกเปลี่ยนกลับเป็น "ไอเดิล" เมื่อแพคเก็ทที่ถูกส่งกระจายออกไปสิ้นสุดและการเชื่อมต่อของตัวรับจะถูกตัด

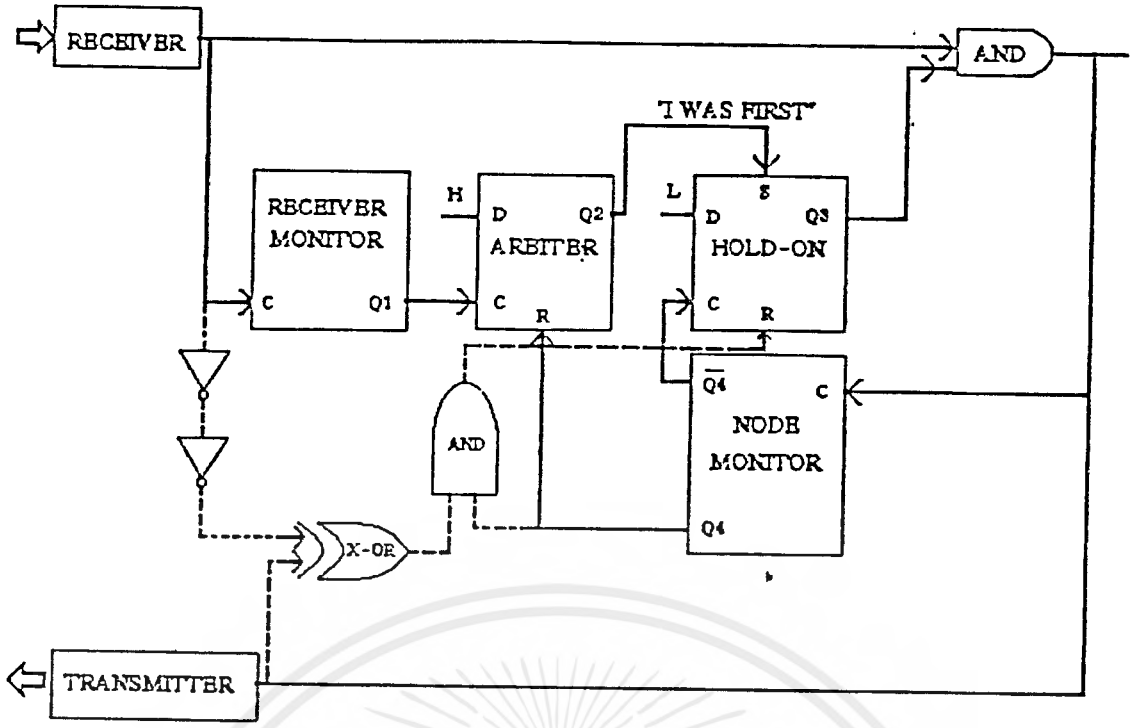
โปรโตคอล นี้ จะถูกจัดการโดยวงจรโลจิก ซึ่งแสดงโดยรูปที่ 2 ตัวรับและตัวส่งเป็นส่วนหนึ่งของ ยูสเซอร์ ลิงค์ ซึ่งจะแปรตามประเภทของลิงค์(link) วงจรโลจิกนี้ ประกอบไปด้วย แอนด์ เกท (AND gate) และ ฟลิปฟลอป (F/F) 4 ตัว ซึ่งถูกต่อถึงกันเป็น รีซีฟเวอร์ มอนิเตอร์ (Receiver Monitor), โนด มอนิเตอร์ (Node-Moiter), อาร์บิเตอร์ (ARBITER), โฮล์ด-ออน (HOLD-ON) ตัว แอนด์ เกท จะเป็นตัวเชื่อมหรือไม่เชื่อมตัวรับกับโนด N ตามสถานะของตัว โฮล์ด-ออน รูปที่ 3 จะแสดง ไทม์มิ่ง ไดอะแกรม (Timing Diagram) ของแต่ละส่วนในวงจร

ตัวรีซีฟเวอร์ มอนิเตอร์ และโนด มอนิเตอร์ เป็นวงจร รีทริกเกเบิล ซิงเกิล ชอต (retriggable single shot) ซึ่งจะตรวจสอบว่ามีแพคเก็ทหรือไม่ โดยการเซนส์ซิ่ง เคอะ แคร์เรียร์ (sensing the carrier) คือ จะเป็น "ไฮ" ("high") เมื่อมีแพคเก็ทมาถึงและจะเป็น "โล" ("low") เมื่อหมดแพคเก็ท

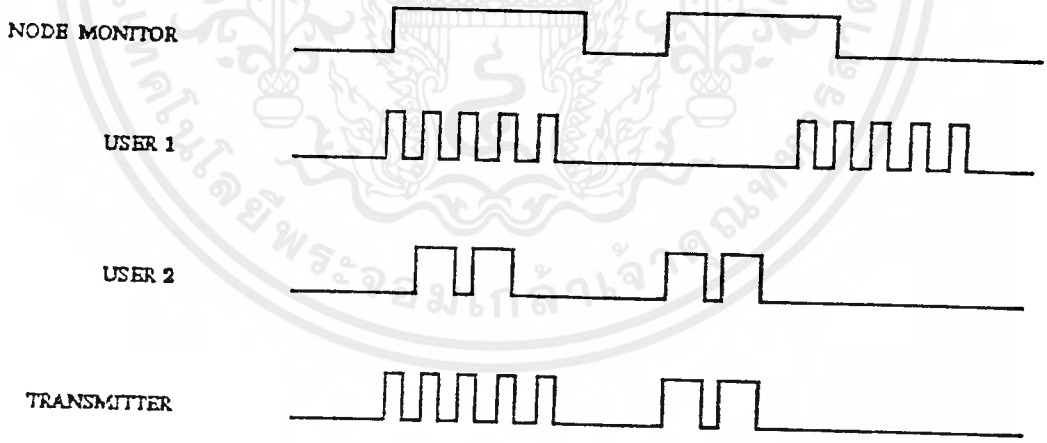
ตัว อาร์บิเตอร์ เป็น ดี ฟลิปฟลอป (D F/F) ซึ่งจะทำหน้าที่ผลิตพัลส์ "ไอ วอส เฟิร์ส " ("I WAS FIRST") เมื่อรีซีฟเวอร์ มอนิเตอร์ เปลี่ยนเป็น "บิซี่" ในขณะที่โนด มอนิเตอร์ เป็น "ไอเดิล" พัลส์ "ไอ วอส เฟิร์ส" จะเป็นตัวชี้ว่าตัวรับมีแพคเก็ทแรกมาถึงและจะเซท โฮล์ด-ออน ฟลิปฟลอป เป็น "ไฮ" และทุกครั้งที่ รีซีฟเวอร์ มอนิเตอร์ เปลี่ยนจาก "ไอเดิล" เป็น "บิซี่" ในช่วงที่ โนด มอนิเตอร์ เป็น "บิซี่" อยู่นั้น จะไม่มีผล และ โฮล์ด-ออน ฟลิปฟลอป ก็จะไม่ถูกรีเซท



รูปที่ 3 แสดงไทม์มิ่ง ไดอะแกรม ของ วงจรบัสเซอร์ อินเตอร์เฟส



รูปที่ 4 แสดงวงจรเพิ่มเติม ยูสเซอร์ อินเตอร์เฟส



รูปที่ 5 แสดงไทม์มิ่ง ไดอะแกรม ของผู้ใช้ 2 เครื่อง เมื่อบัส ไอเดิล

ตัว โอล์ด-ออน ฟลิปฟลอป จะเป็น "ไอ" ตามพัลส์ "ไอ วอส เพิร์ส"และจะเปลี่ยนเป็น "โล" เมื่อ โนด มอนิเตอร์ เป็น "ไอเดิล" เอาท์พุทของ โอล์ด-ออน ฟลิปฟลอป จะไปทำให้ แอนด์ เกท ทำงาน โดยจะส่งผลให้ตัวรีซีฟเวอร์เชื่อมหรือไม่เชื่อมกับโนด (N)

ลักษณะการทำงานดังที่กล่าวมาแล้วนั้นอาจจะทำให้สงสัยว่า หากมีแพคเก็ทมาถึงตั้งแต่ 2 แพคเก็ทขึ้นไปพร้อมๆ กัน ซึ่งที่จริงแล้วมีโอกาสเกิดขึ้นน้อยมาก เพราะการตัดสินว่า แพคเก็ทใดเป็นแพคเก็ทแรกนั้น จะใช้เวลาเพียง 20 ns แต่ถ้าจะสนใจว่าเกิดขึ้นภายในช่วง 20 ns นั้น จะก่อให้เกิดปัญหาอย่างไร ซึ่งอาจจะเป็นการชนกันของแพคเก็ททั้ง 2 ก็ได้ เพราะรีซีฟเวอร์ทั้ง 2 ตัวจะถูกเชื่อมกับโนด N พร้อมๆ กัน

จากรูปที่ 4 จะเห็นว่ามีการเพิ่มวงจรเอ็กซ์คลูซีฟ ออร์ (exclusive OR) ซึ่งจะช่วยแก้ปัญหาที่มีโอกาสเกิดขึ้นน้อยนี้ได้ โดยจะรีเซท โอล์ด-ออน ฟลิปฟลอป เป็น "โล" ตลอดเวลาที่สัญญาณจาก รีซีฟเวอร์ แตกต่างจากสัญญาณ ทรานซมิตเตอร์ (Trans mitter) และช่วงที่โนดเป็น "บิซี่" รีซีฟเวอร์ 2 ตัวยังคงเชื่อมต่ออยู่จนกระทั่ง สัญญาณจาก รีซีฟเวอร์ ตัวใดตัวหนึ่งต่างจากสัญญาณ ทรานซมิตเตอร์ ของตัวมันเอง การเชื่อมต่อของ รีซีฟเวอร์ ตัวนั้นก็จะสิ้นสุดลง รีซีฟเวอร์ที่มีบิท "โล" ก่อน รีซีฟเวอร์ ตัวอื่นจะถูกตัดก่อน รีซีฟเวอร์ ตัวอื่น รูปที่ 5 จะแสดงไทม์มิ่ง ไดอะแกรม ของเหตุการณ์สุ่ม 2 ยูสเซอร์ ส่งข้อมูลมาพร้อมๆ กัน

สรุป

กรรมวิธีใหม่ของสตาร์ เน็ตเวอร์ค ที่คิดขึ้นในรูปของ เซ็นทรัล โนด สามารถรองรับการปฏิบัติการที่ 50 Kb/s โนด สามารถจัดการชนโดยใช้หลักการที่ว่า ทางใครทางมัน หรือใครถึงก่อนมีสิทธิ์ก่อน เมื่อมีแพคเก็ทมากมายมาที่โนด โนดจะให้แพคเก็ท ที่มาถึงก่อนผ่าน และจะกั้นแพคเก็ทอื่นๆ ไม่ให้มาชนแพคเก็ทแรก ในขณะเดียวกัน ผู้ใช้ก็จะต้องคอยส่งแพคเก็ทใหม่ (retransmitting) จนกว่าจะผ่าน

This paper define every kind of packet used in the network.

HEADER — each field of header is one byte long

DC1	DC2	DC3	DC4	packet type	sorce	destination		no. of byte to follow

This field tell the receive routine if there are anything to display. If there are, the following bytes will be the characters to be displayed

This field tells the number of bytes that follow the header If the packet has only the header with no follow infomation, this field contains value = 0.

Packet type field: This field indicates the type of packet. There are a number of packet types.

1. File transfer request
2. File transfer confirm
3. File transfer
4. Registration packet
5. NAK packet
6. Disconnect packet
7. Chat mode request
8. Chat mode confirm
9. Chat
10. E-mail request
11. E-mail confirm
12. E-mail
13 Poll E-mail
14 Who
15 List

The last field will be terminated by "newline"

File transfer request packet

header (9-byte)	total packet (2-byte)	file size (3-byte)	filename (12-byte)	data to display

The value in packet type field in the header equal to ...."Bx0000-0001".

File transfer confirm packet This packet format is the same as FT request, but in the packet type field contains ..."Bx0000-0010".

File transfer packet This packet has data field follow header ,not exceed 247 bytes, and in the packet type field contains ..."Bx0000-0011".

header	username(3			In the packet type field

contains..."Bx0000-0101".

Regis. packet

This last field contains char. "I","C" or "O" for login, toggle mode or logoff respectively.

NAK There is only header and in the packet type field contains..."Bx0001-0101".

Disconnect There is only header and in the packet type field contains..."Bx0000-0100".

E-mail request

header (9-bytes)	filename (247-bytes)

in the packet type field contains value 6.

E-mail confirm

header (9-byte)	total packet (2-byte)	file size (3-byte)	filename (12-byte)	data to display

in the packet type field contains value 7.

E-mail This packet has data field follow header ,not exceed 247 bytes, and in packet type field contains value 8.

P E-mail There is only header and in the packet type field contains value 9.

Chat request packet There is only header and in the packet type field contains value 10.

Chat confirm packet There is only header and in the packet type field contains value 11.

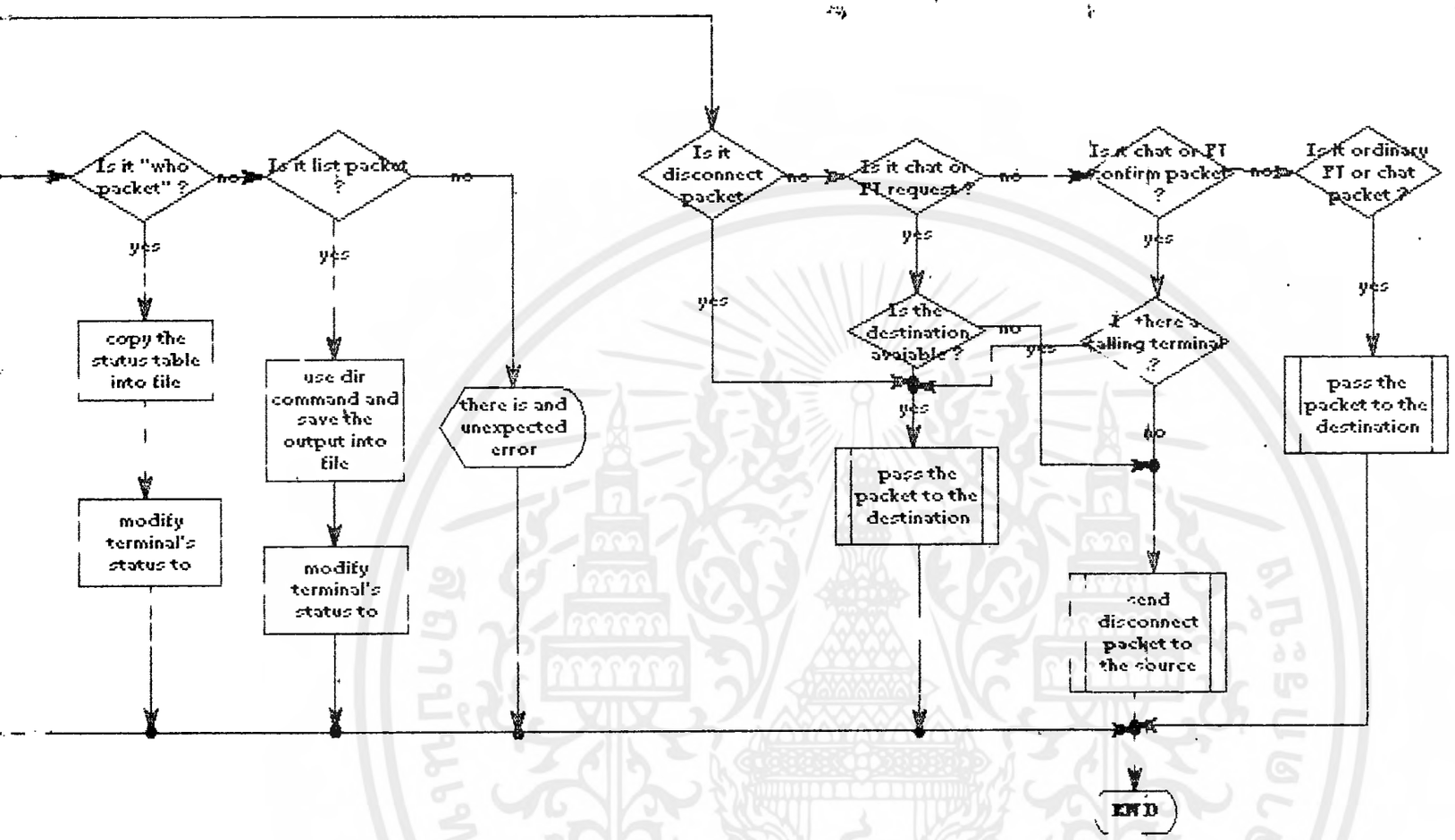
Chat packet This packet has data field follow header ,not exceed 247 bytes, and in packet type field contains value 12.

Who packet There is only header and in the packet type field contains value 13.

List packet

header (9-bytes)	pathname (247-bytes)

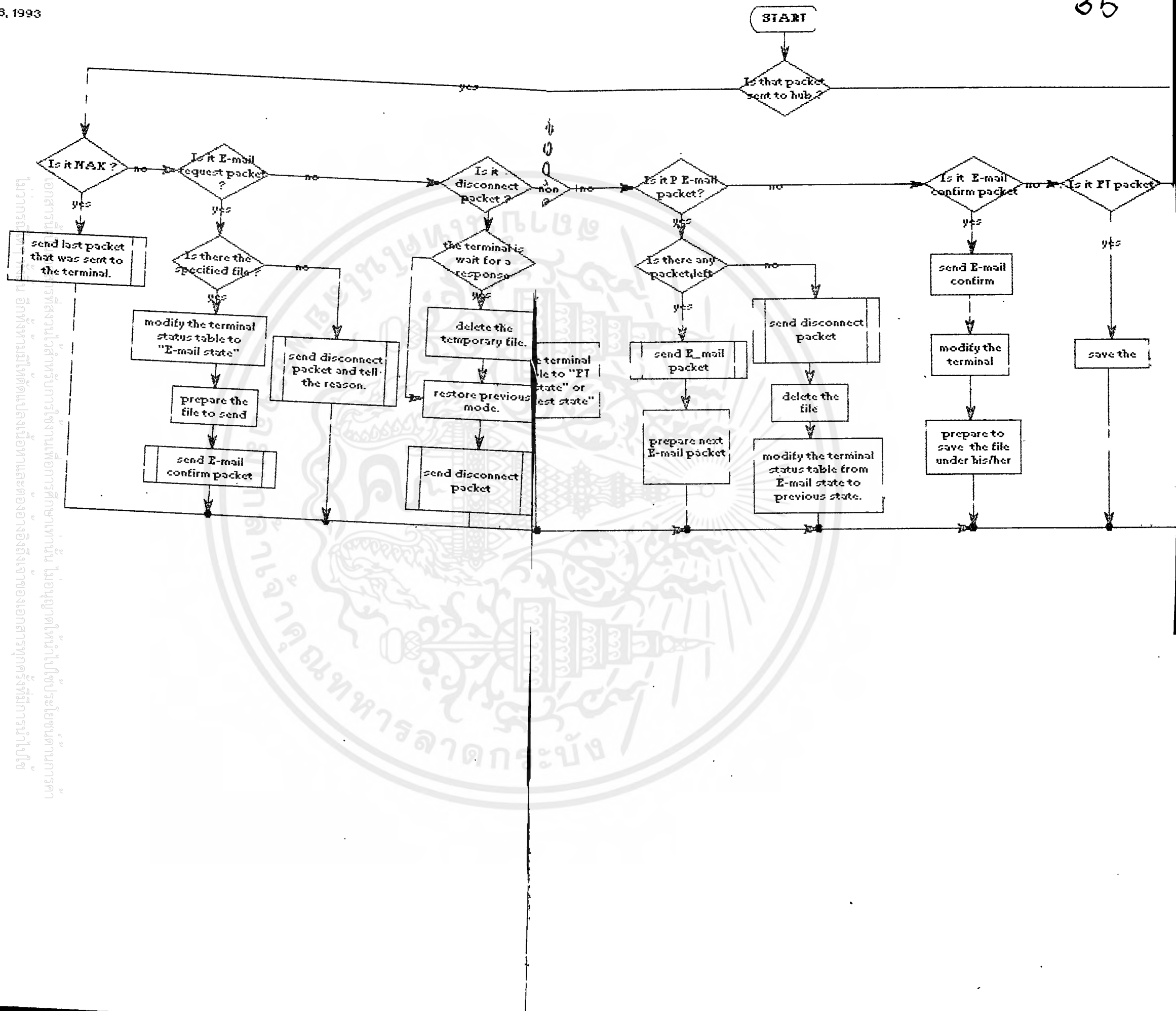
In the packet type field contains value 14.



determine packet's type

Mon, Jan 18, 1993

10:30 AM



Login

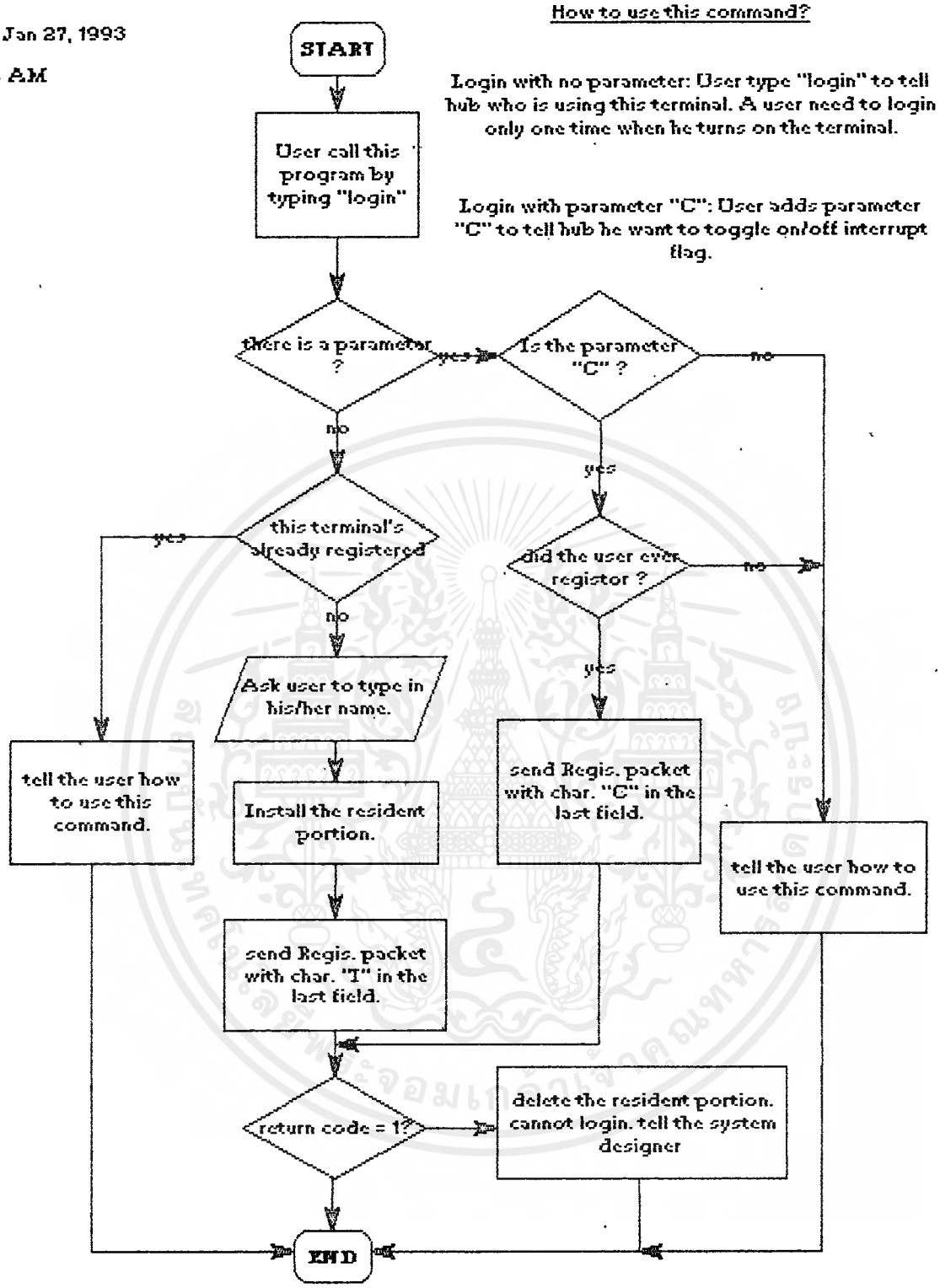
Wed, Jan 27, 1993

10:54 AM

How to use this command?

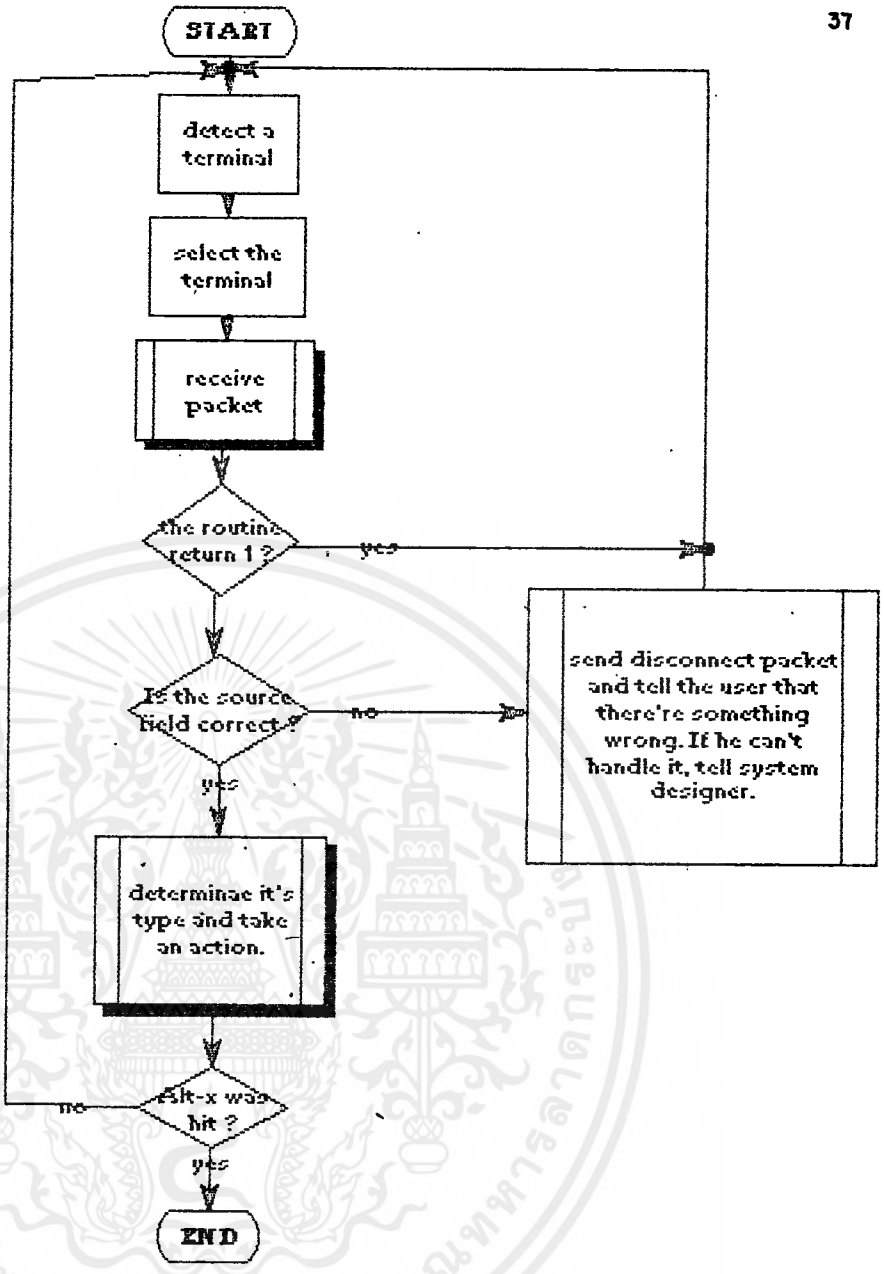
Login with no parameter: User type "login" to tell hub who is using this terminal. A user need to login only one time when he turns on the terminal.

Login with parameter "C": User adds parameter "C" to tell hub he want to toggle on/off interrupt flag.



Regis. packet

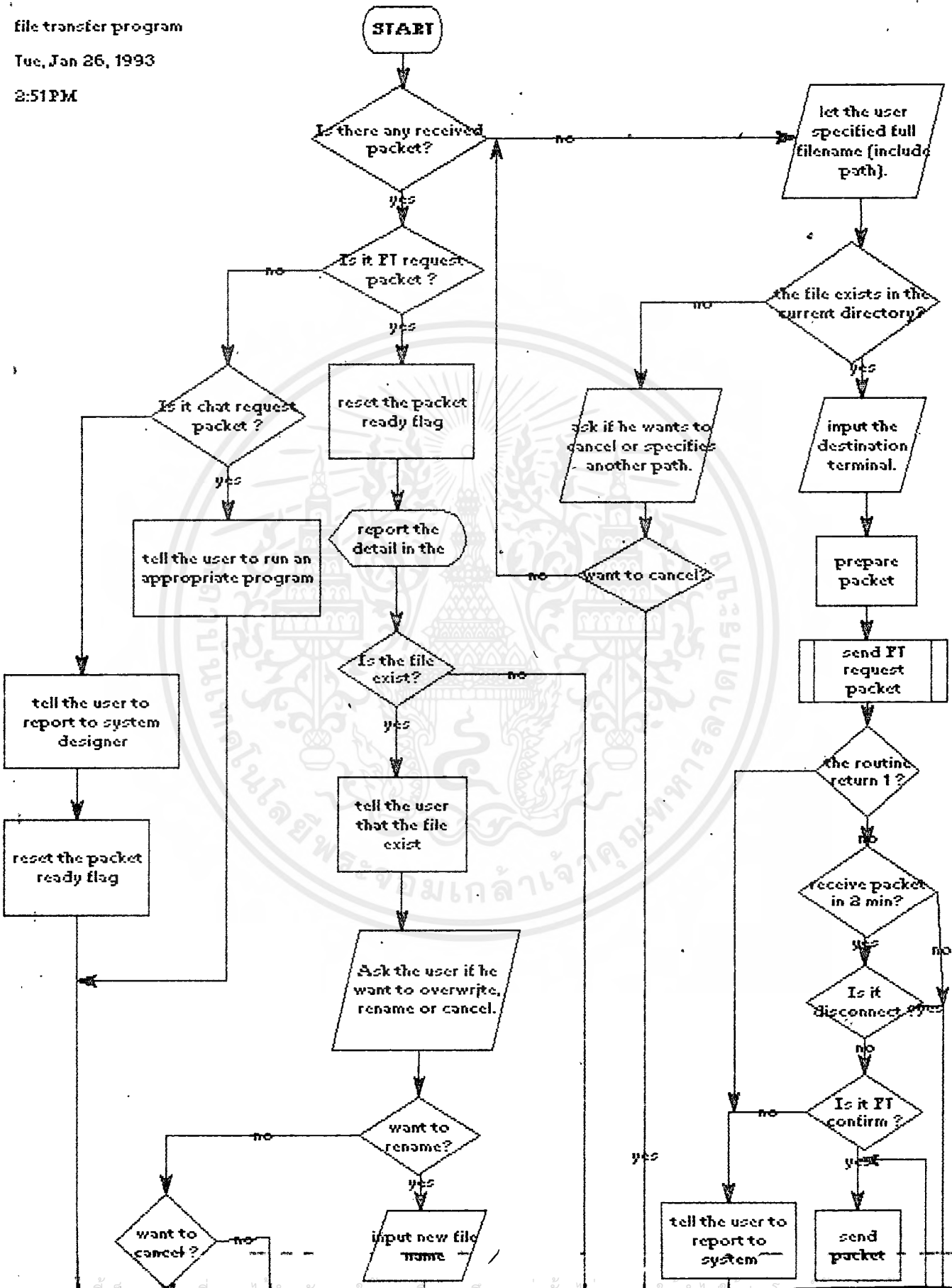
This last field contains char. "I","C" or "O" for login, toggle mode or logoff respectively.

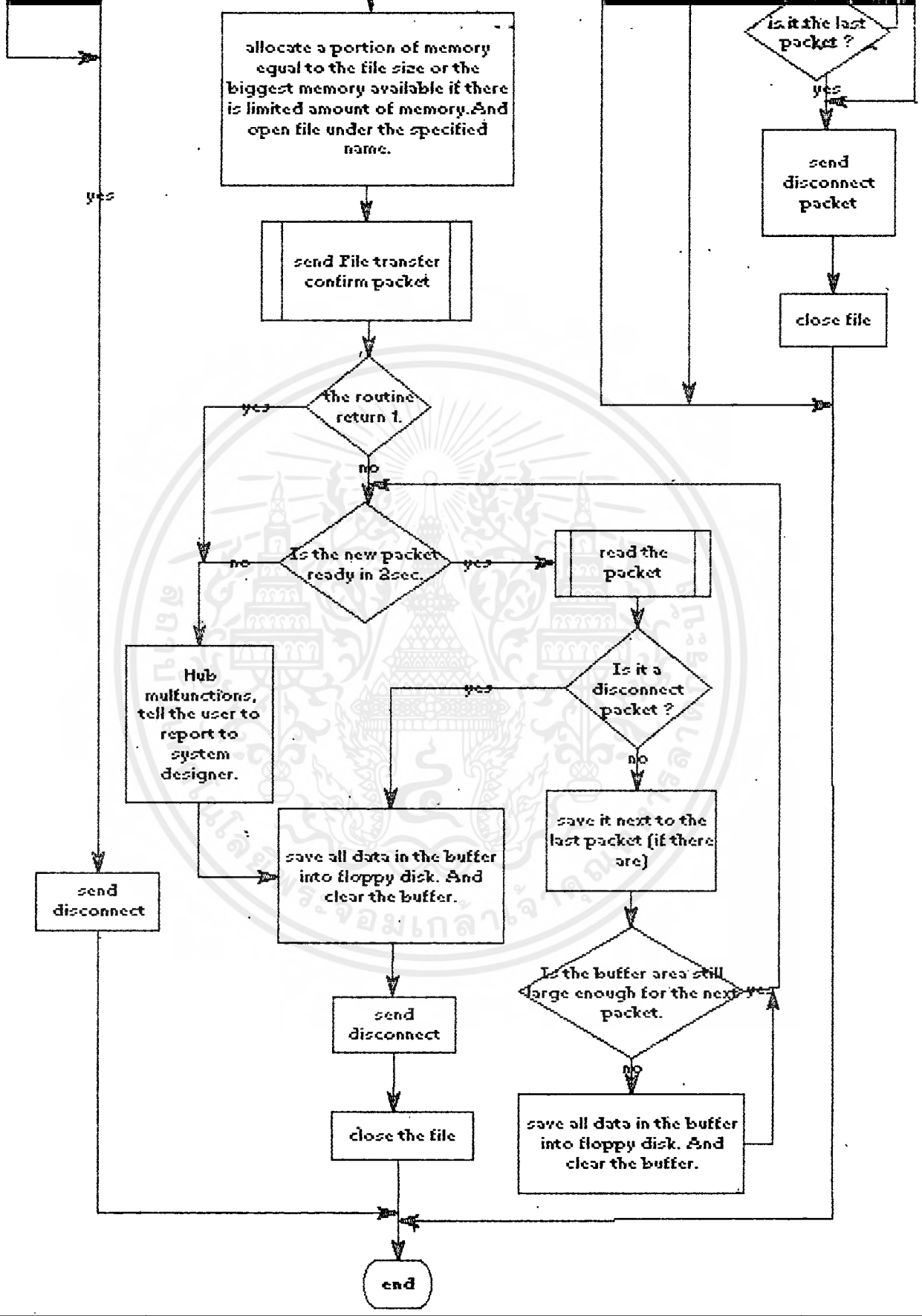


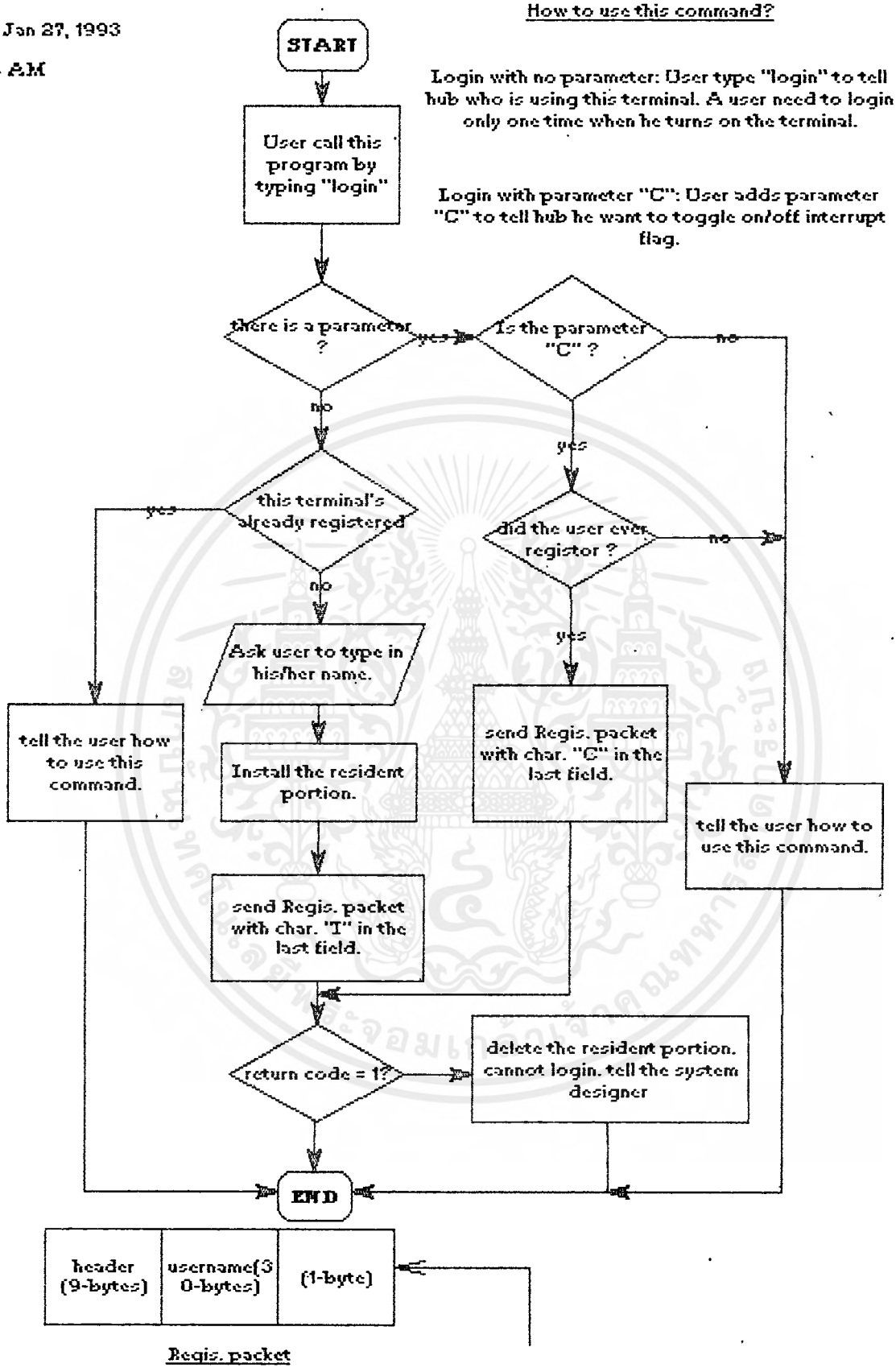
file transfer program

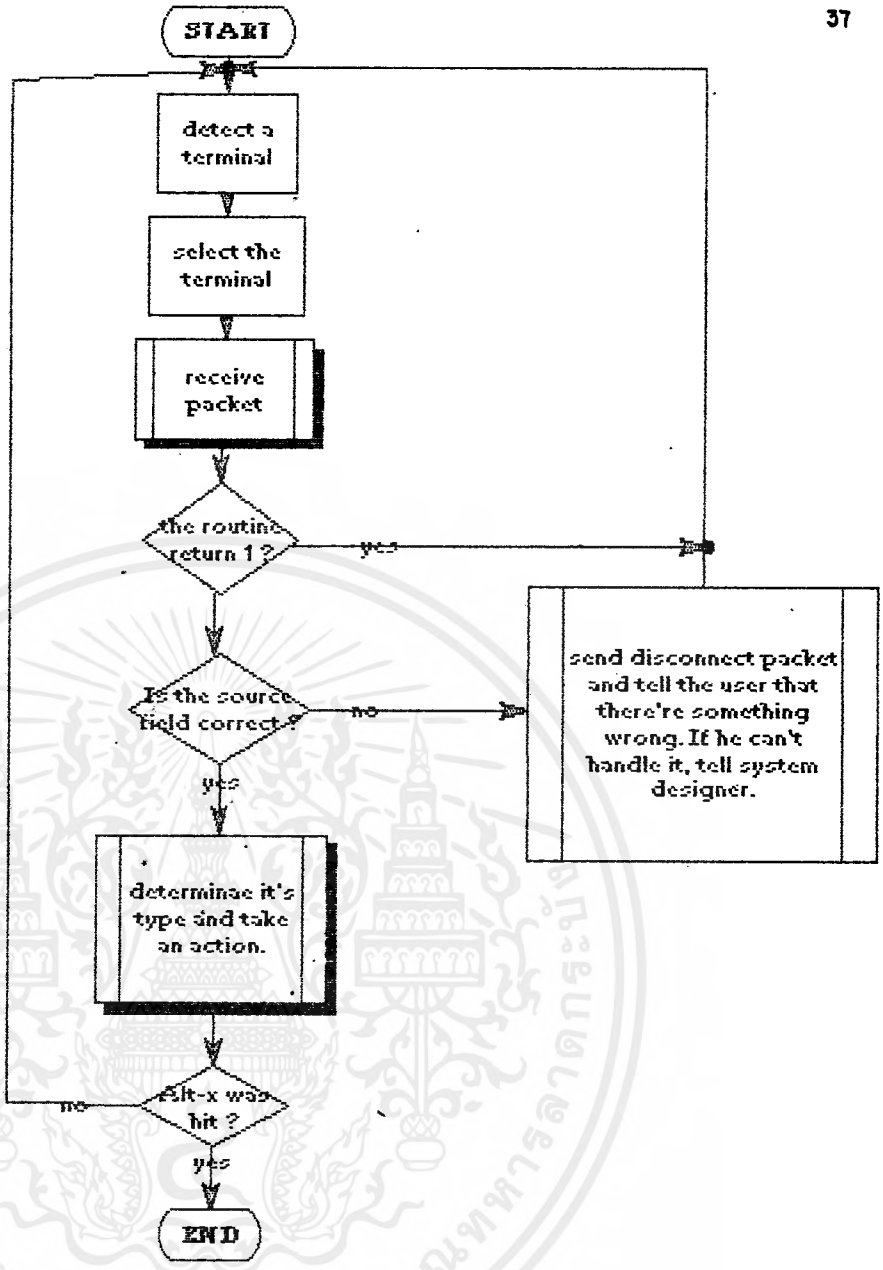
Tue, Jan 26, 1993

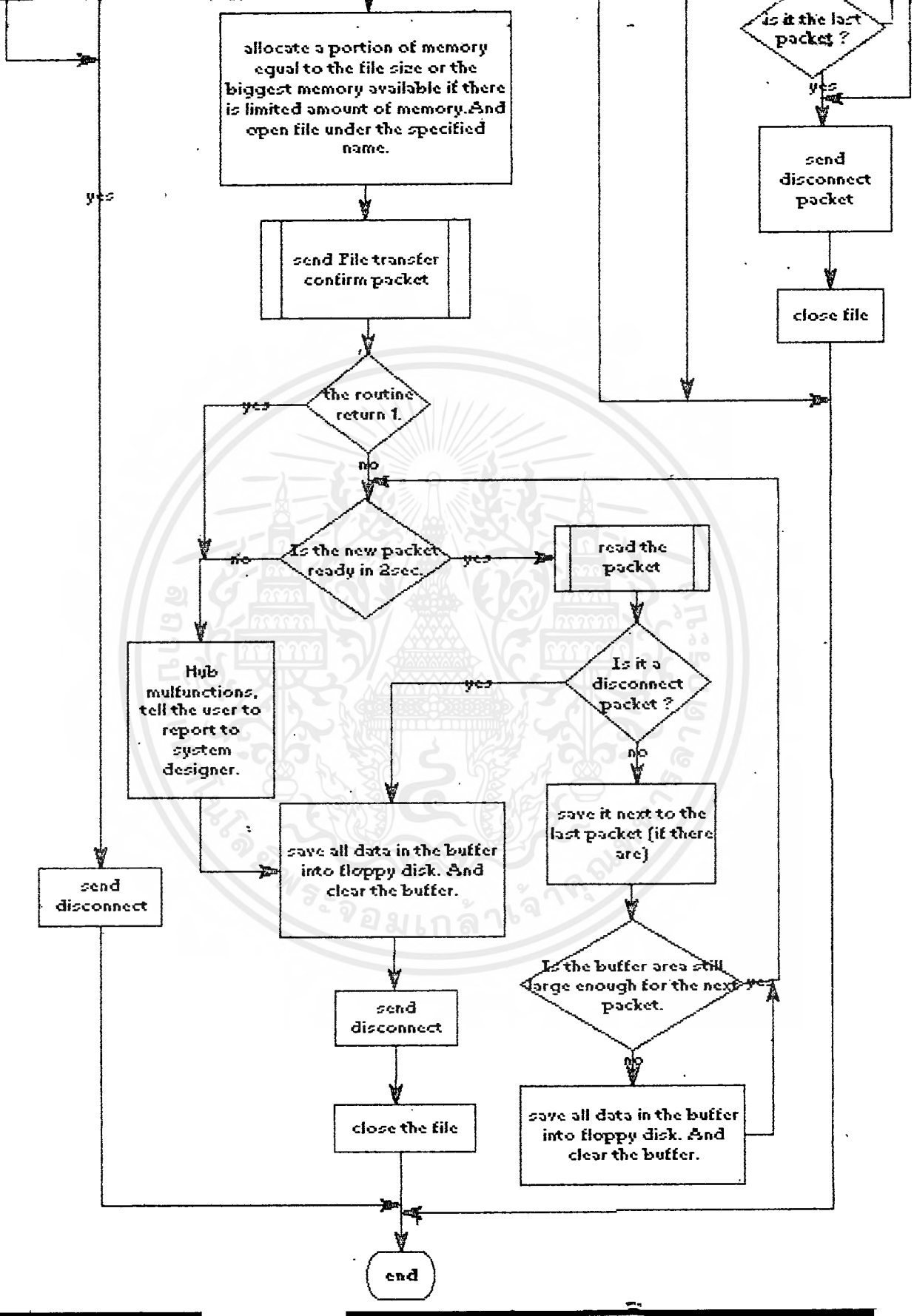
2:51PM




allocate a portion of memory equal to the file size or the biggest memory available if there is limited amount of memory.And open file under the specified name.
send File transfer confirm packet
the routine return 1.
yes
no
Is the new packet ready in 2sec.
no
yes
read the packet
Is it a disconnect packet ?
yes
no
Hub malfunctions, tell the user to report to system designer.
save all data in the buffer into floppy disk. And clear the buffer.
save it next to the last packet (if there are)
Is the buffer area still large enough for the next packet.
yes
no
save all data in the buffer into floppy disk. And clear the buffer.
send disconnect
close the file
send disconnect
yes
Is it the last packet ?
yes
send disconnect packet
close file
end

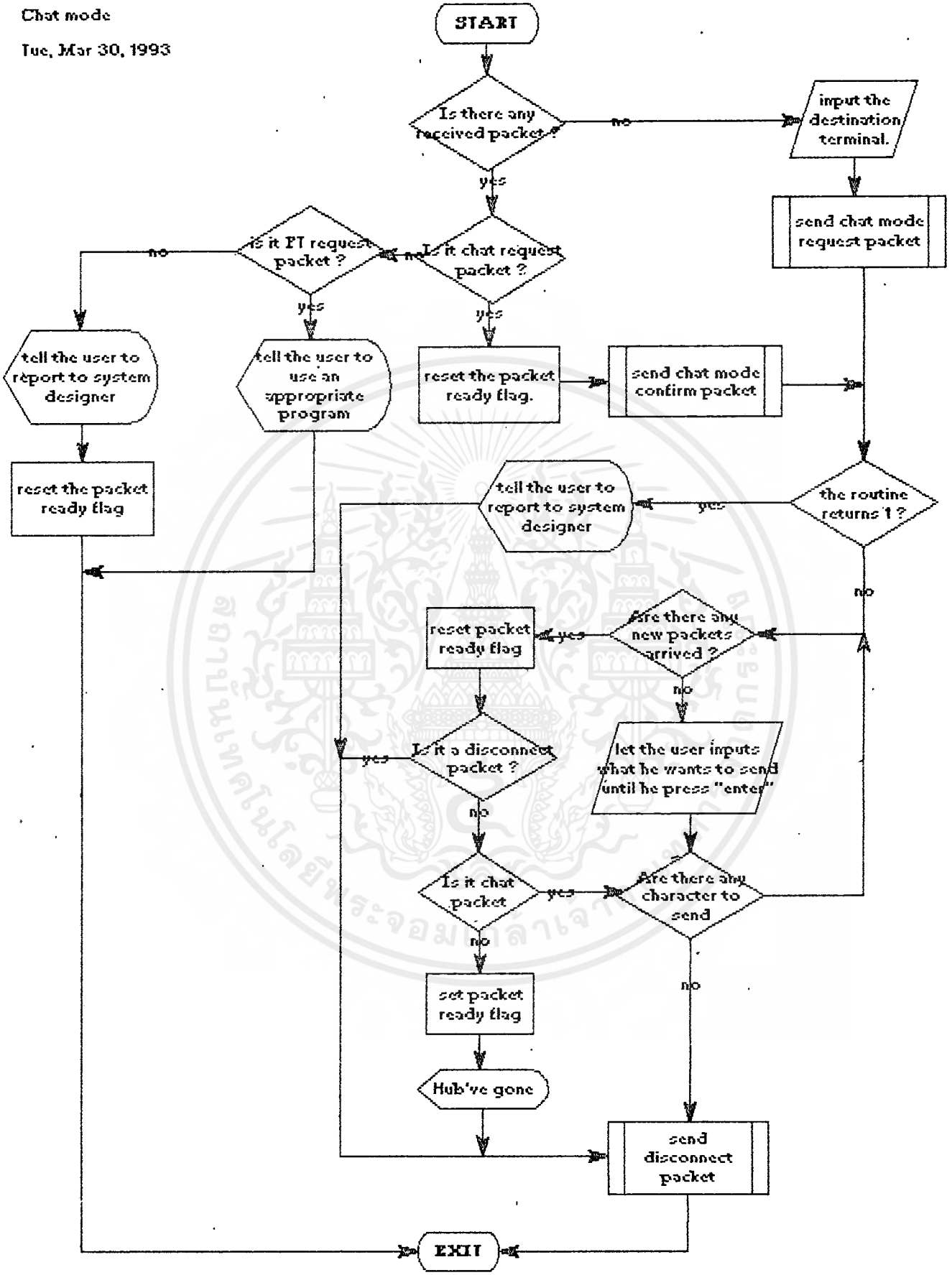


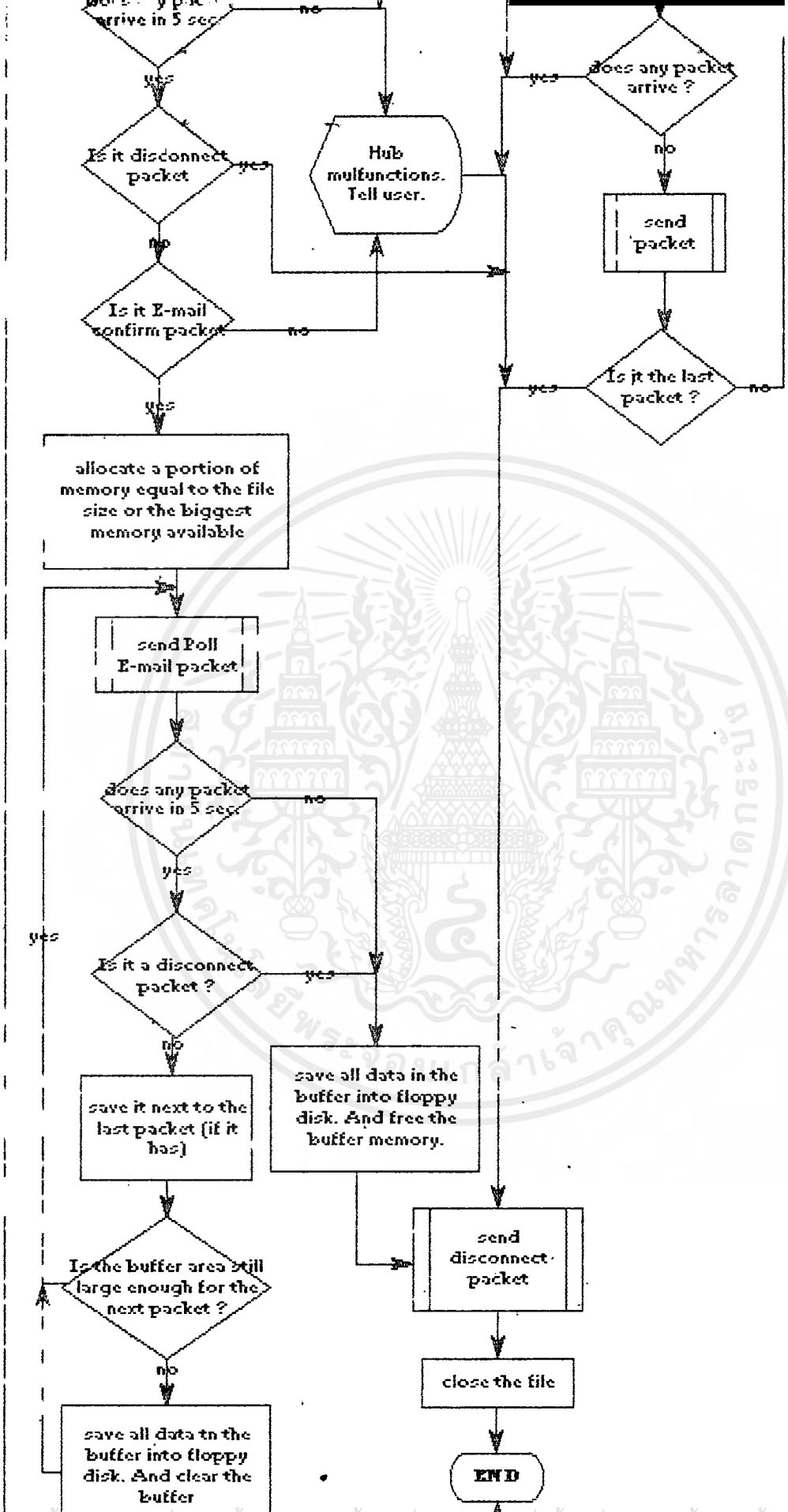



allocate a portion of memory equal to the file size or the biggest memory available if there is limited amount of memory.And open file under the specified name.
send File transfer confirm packet
the routine return 1.
yes
no
Is the new packet ready in 2sec.
no
yes
read the packet
Is it a disconnect packet ?
yes
no
Hub mulfunctions, tell the user to report to system designer.
save all data in the buffer into floppy disk. And clear the buffer.
save it next to the last packet (if there are)
Is the buffer area still large enough for the next packet.
yes
save all data in the buffer into floppy disk. And clear the buffer.
send disconnect
close the file
send disconnect
close file
is it the last packet ?
yes
send disconnect packet
yes
end

Chat mode

Tue, Mar 30, 1993

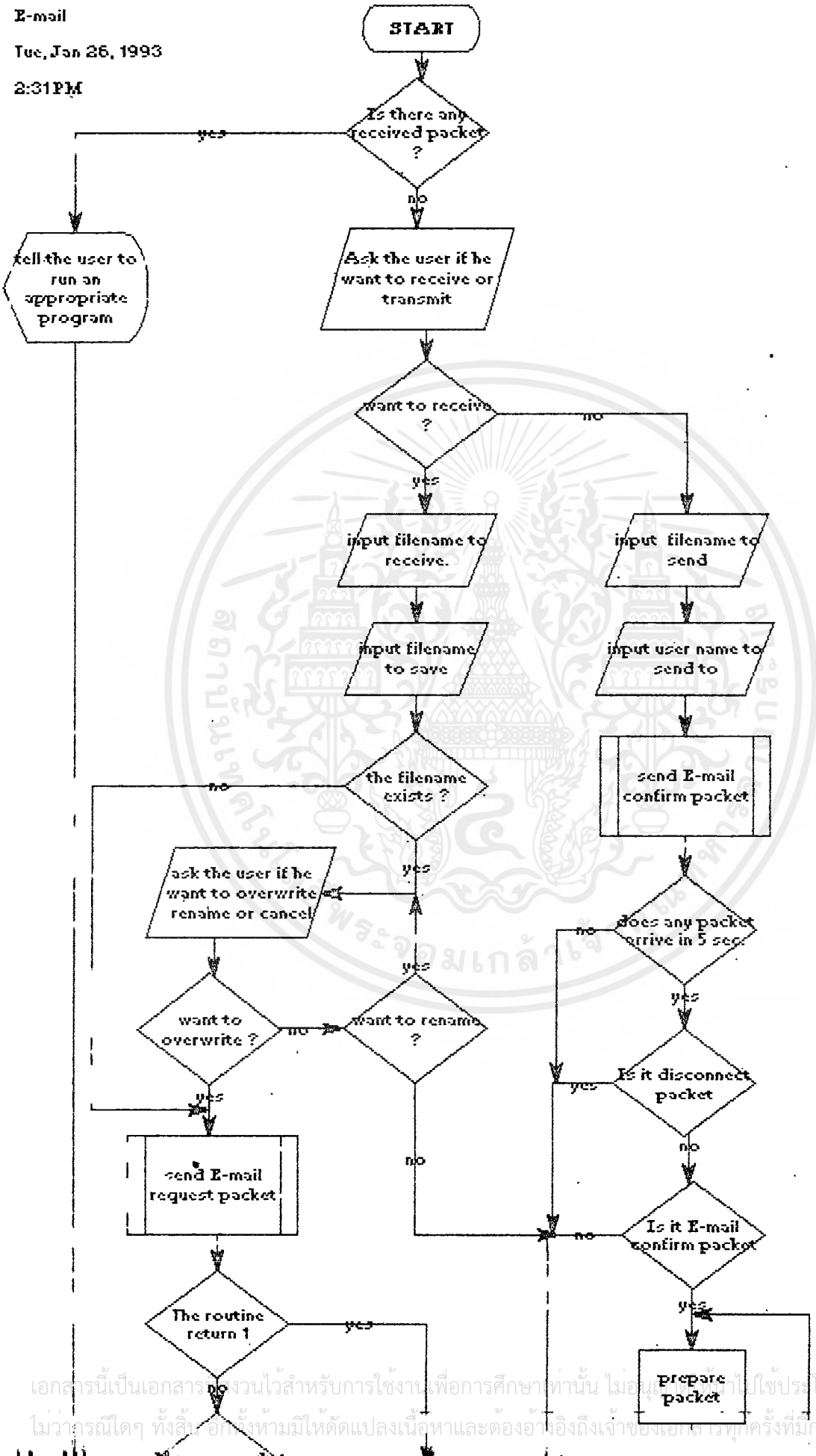




E-mail

Tue, Jan 26, 1993

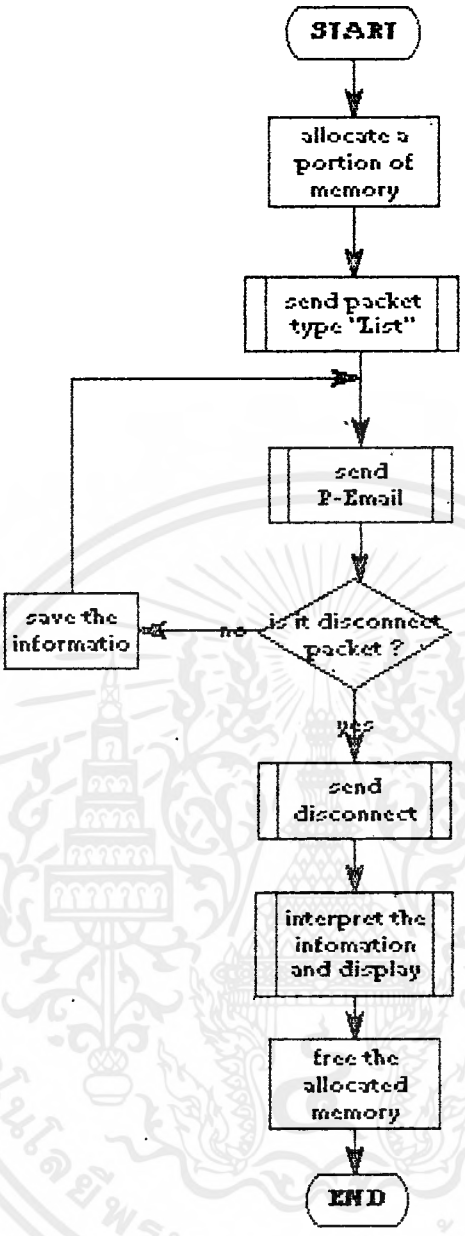
2:31PM

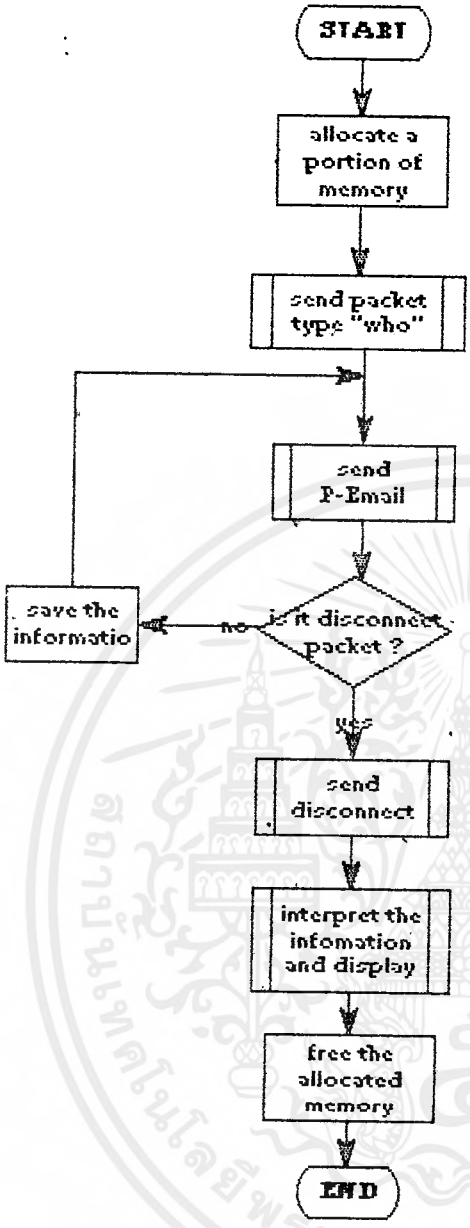


LIST

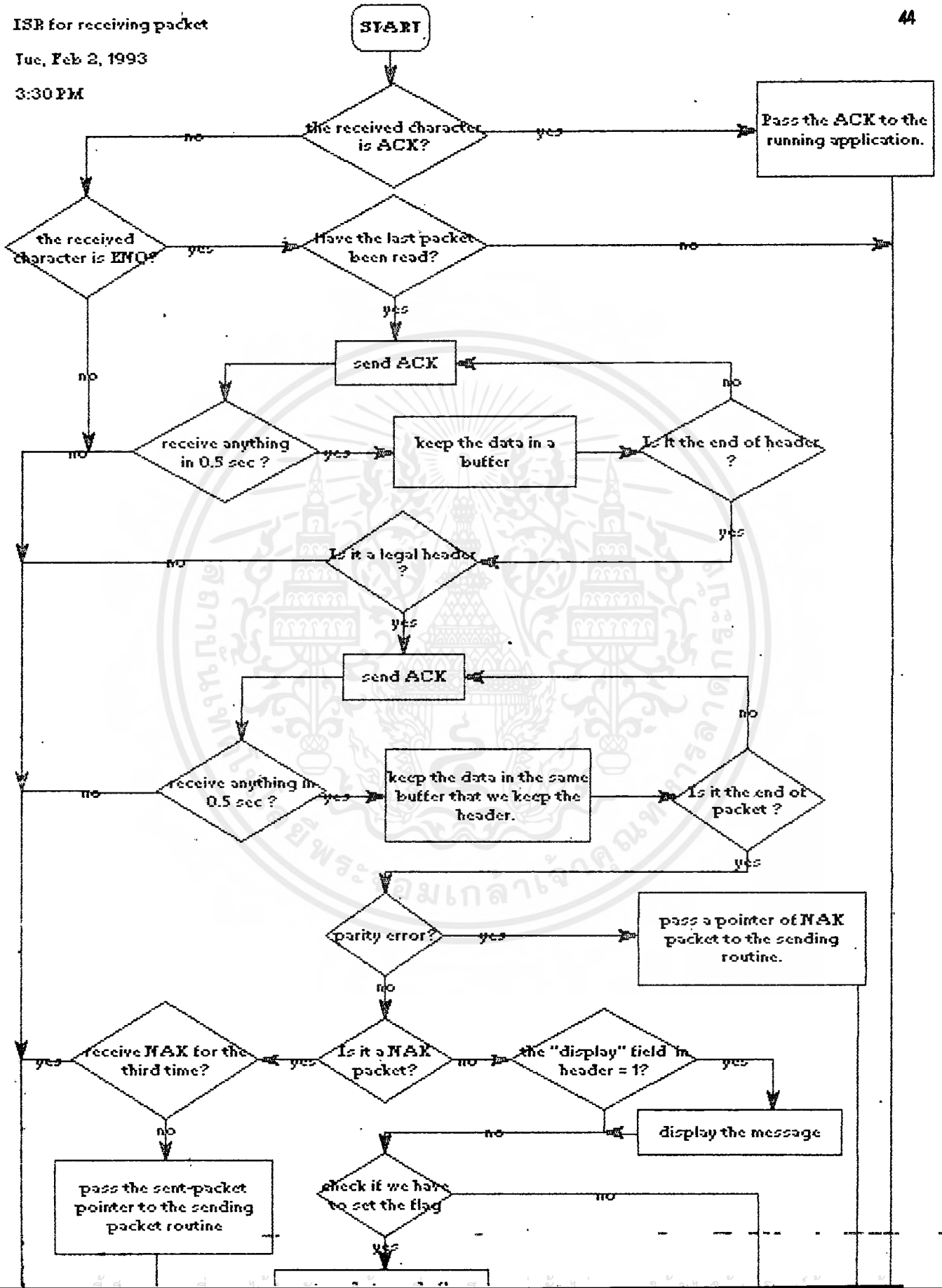
Tue, Jan 26, 1993

3:15 PM





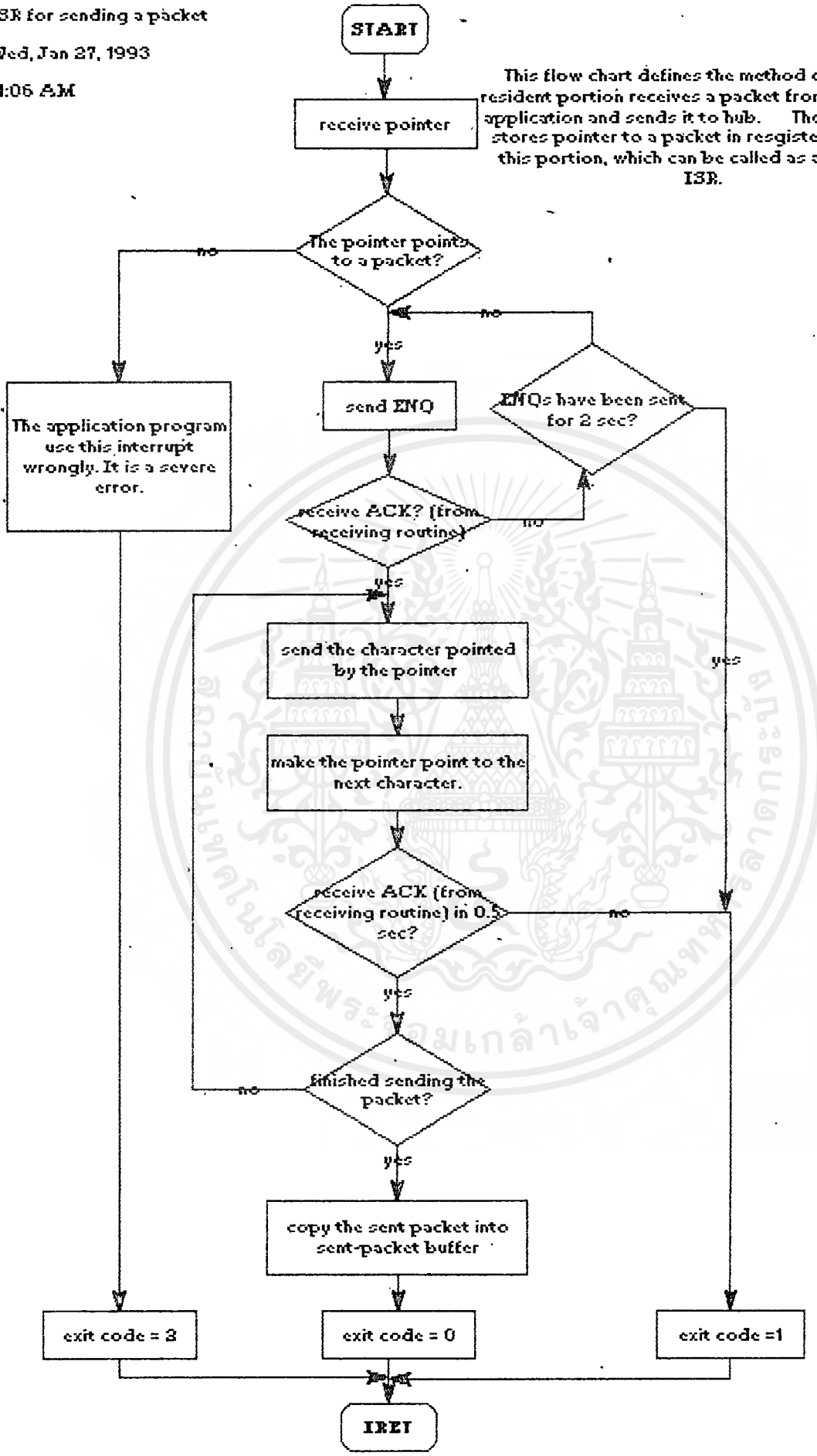

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

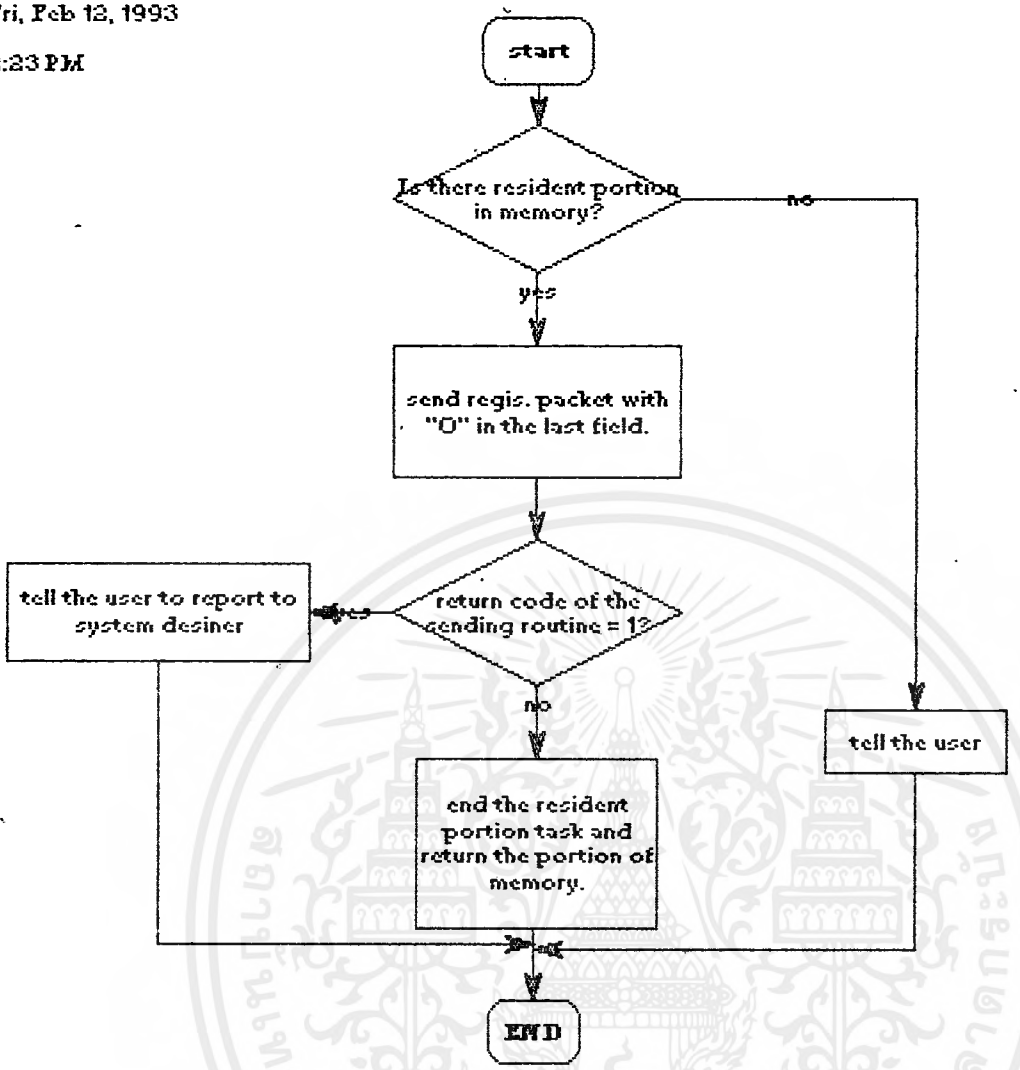

START
the received character is ACK?
yes
Pass the ACK to the running application.
no
the received character is ENQ?
yes
Have the last packet been read?
no
yes
no
send ACK
no
receive anything in 0.5 sec ?
yes
keep the data in a buffer
Is it the end of header ?
no
yes
Is it a legal header ?
no
yes
send ACK
no
receive anything in 0.5 sec ?
no
yes
keep the data in the same buffer that we keep the header.
Is it the end of packet ?
no
yes
parity error?
yes
pass a pointer of NAK packet to the sending routine.
no
receive NAK for the third time?
yes
Is it a NAK packet?
yes
no
the "display" field in header = 1?
yes
display the message
no
no
pass the sent-packet pointer to the sending packet routine
check if we have to set the flag
no
yes

ISR for sending a packet

Wed, Jan 27, 1993

11:06 AM





```
/*------File name TTEST2.C---the program is a TSR program.----*/
/*------It is work in Data link layer-------------------------*/
#include (dos.h)
#include (mem.h)
#include (stdio.h)
#include (stdlib.h)

#pragma inline

/******
        Control Character
******/
#define ENQ             0x05
#define ACK                     0x06
#define NAK                     0x21
#define SOU                     0x00
#define ZERO_DIV        0x00
#define COMM            0x08
#define TIME                    0x1C
#define KEY_LOOP        0x28
#define DISK            0x13
#define CTRL_BRK        0x1B
#define CTRL_C          0x23
#define TABLE                   0x60
#define SERVICE         0x61
#define GET_INDOS               0x34
#define GET_CRIT_ERR            0x5D06
#define POS_MAX         80

/*--------Prototype---------*/
void interrupt new_time();
void interrupt new_comm();
void interrupt new_key_loop();
void interrupt new_disk();
void interrupt new_break();
void interrupt new_ctrlc();
void interrupt new_service();

unsigned get_psp();
unsigned alloc_mem(int size);
unsigned free_mem(unsigned seg);

int   request();
int   p_err();
char  r_port();
char  rr_port();
int   DosBusy(void);
int   Int28DosBusy(void);
int      data_ready();

void  s_port(char data);
void  s_non_ack();
void  trn_data();
void  rec_data();
void  wait_key();
void  popup();
void  clear_func();
void     clear_err();
void  disp_n_message();
```

```
void  disp_o_message();
void  clear_ready();
void  init_block();
void  set_psp(unsigned psp);
void  set_popup();
void  init_comm();
void  clear_buff();
void  end_of_intr();
void  enable_8259();
void  set_seg_block();
void  disable_8259();
void  InitInDos(void);
void  s_last_packet();
void  put_char(unsigned row,unsigned col,unsigned char data);
void  put_message(unsigned row,unsigned col,int message_num);

/*--------Global variable--------*/
void interrupt (*old_comm)();
void interrupt (*old_key_loop)();
void interrupt (*old_disk)();
void interrupt (*old_break)();
void interrupt (*old_ctrlc)();
void interrupt (*old_table)();
void interrupt (*old_time)();
void interrupt (*old_service)();
void interrupt (*new_table)();

union REGS regs;
struct SREGS sregs;

static int func1;
static int func2;
static int disp;

extern unsigned _heaplen=2048;
extern unsigned _stklen=1024;

int non_ack_pac[]={17,18,19,20,21,SOU,0,0,0};    /*----NAK packet format----*/

unsigned segment[2];
unsigned char message[][30]=
                                {"Press F11 to continue","Now detected data ready"};
                                /* Max 22 character */
unsigned char ins_check[8]={"STAR NET"};
unsigned int far *old_message;
unsigned char far *new_message;

unsigned table;
unsigned o_mess;
unsigned n_mess;
unsigned int far *screen;
unsigned char far *con_block;
unsigned char far *packet;
unsigned char far *block[2];
char far *indos_ptr=0;
char far *crit_err_ptr=0;

void main()
{
      disp=0;
```

```
        func1=0;
        func2=0;
        InitInDos();                    /* Function:Init InDos Pointers */

        screen=MK_FP(0xB800,0);
        init_comm();                        /* Initialize communication port */

        segment[0]=alloc_mem(256);
        if(!segment[0])
                exit(1);

        block[0]=MK_FP(segment[0],0);

        segment[1]=alloc_mem(256);
        if(!segment[1])  {
                free_mem(segment[1]);
                exit(1);
        }
        block[1]=MK_FP(segment[1],0);

        table=alloc_mem(16);
        if(!table)  {
                free_mem(segment[0]);
                free_mem(segment[1]);
                exit(1);
        }
        new_table=MK_FP(table,0);
        con_block=MK_FP(table,0);

        o_mess=alloc_mem(80);
        if(!o_mess)  {
                free_mem(segment[0]);
                free_mem(segment[1]);
                free_mem(table);
                exit(1);
        }
        old_message=MK_FP(o_mess,0);

        n_mess=alloc_mem(80);
        if(!n_mess)  {
                free_mem(segment[0]);
                free_mem(segment[1]);
                free_mem(table);
                free_mem(o_mess);
                exit(1);
        }
        new_message=MK_FP(n_mess,0);

        init_block();

    set_seg_block();

/*****
        Read old interrupt vector
*****/
        old_comm=getvect(COMM);                 /* return segment:offset ES:AX */
        old_time=getvect(TIME);
        old_service=getvect(SERVICE);
        old_table=getvect(TABLE);
```

```
/*****
        Set new interrupt vector
*****/
        setvect(COMM,new_comm);
        setvect(TIME,new_time);
        setvect(SERVICE,new_service);
        setvect(TABLE,new_table);

        clear_buff();
        enable_8259();
        enable();

        printf("\nSEGMENT 1 :%04X",segment[0]);
        printf("\nSEGMENT 2 :%04X",segment[1]);
        printf("\nCONTROL BLOCK :%04X",table);

/*****
        Terminate and stay resident by using function 31H of DOS
*****/
        keep(0,_SS+(_SP/16)-_psp);
}

void interrupt new_time()
{
        int byte;
        int lst;

        (*old_time)();
        if((!func2) && (!func1) && disp)  {
                lst=block[0][8];
                for(byte=0;byte < POS_MAX;byte++)  {
                        if(byte < lst)
                                new_message[byte]=block[0][9+byte];
                        else
                                if(byte > 58)
                                        new_message[byte]=message[0][byte-59];
                                else  {
                                        new_message[byte]=0;
                                        block[0][byte]=0;
                                }
                }
                disp_n_message(25);
                wait_key();
                disp_o_message(25);
                disp--;
        }
}
void interrupt new_service()
{
        switch(con_block[1])  {
                case 0  : break;
                case 1  : disable_8259();
                          popup();
                          enable_8259();
                          break;                /*tran. packet*/
                case 2  : if(con_block[5])  {
                                  set_seg_block();
                                  clear_ready();  /* if data ready set first segment */
                                  }                     /* to con_block else initialize con*/
```

```
                        clear_func();
                        break;          /*receive packet*/
                case 3  : break;
                default : break;
        }
}

void interrupt new_comm()
{
        func2++;
        con_block[1]=2;
        disable_8259();
        popup();
        end_of_intr();
        enable_8259();
        func2--;
}

void init_block()
{
        int byte;
        con_block[0]=0xCF;
        for(byte=1;byte<16;byte++)
                con_block[byte]=0x00;
}

void popup()
{
        switch(con_block[1]) {
                case 0  : break;
                case 1  : clear_func();
                          trn_data();
                          set_seq_block();
                          break;
                case 2  : clear_func();
                          rec_data();
                          break;
                default : break;
        }
}

void rec_data()
{
        int byte;
        int max;
        max=256;
        init_block();
        if(rr_port() == ENQ)  {
                s_port(ACK);
                for(byte=0;byte<max;byte++)  {
                        block[0][byte]=r_port();
                        s_port(ACK);
                        if(p_err())
                                con_block[4]=0xFF;
                        if(byte==8)
                                max=block[0][byte]+8;
                }
                if(con_block[4]==0xFF)  {
                        non_ack_pac[6]=block[0][5];     /*destination of NAK packet*/
                        s_non_ack();
```

```
                        con_block[5]=0;
                }
                else {
                        con_block[5]=0xFF;
                }
                if(block[0][4] == NAK)  {
                        s_last_packet();
                }
                if(block[0][7] ).0)  {
                        disp++;
                }
        }
}

void trn_data()
{
        int byte;
        int max;
        max=256;
        max=block[0][8]+8;
        do  {
                s_port(ENQ);
        }while(r_port() != ACK);
        for(byte=0;byte(max;byte++)  {
                s_port(block[0][byte]);
                do  {
                }while(r_port() != ACK);
                block[1][byte]=block[0][byte];
        }
}

void init_comm()
{
        outportb(0x2FB,0x80);                                   /* Set DLAB */
        outportb(0x2F9,0x00);
        outportb(0x2F8,0x03);                   /* Setup baud rate to 38.4 kBit */
        outportb(0x2FB,0x1B);                   /* Reset DLAB and set to 8E1 */
        outportb(0x2F9,0x01);       /* Generate signal Intr. when received data */
        outportb(0x2FC,(inportb(0x2FC) | 0x08));     /* Set OUT2 for Intr. signal */
}

void set_seg_block()
{
        int byte;
        unsigned con;
        con=segment[0];
        con=(con & 0xFF00)))8;
        *(con_block+2)=con;
        con=segment[0];
        con=(con & 0x00FF);
        *(con_block+3)=con;
        for(byte=8;byte ( 16;byte++)
                con_block[byte]=ins_check[byte-8];
}

unsigned get_psp()
{

        unsigned psp;
        asm  {
```

```
                mov ah,62h
                int 21h
                mov psp,bx
        }
        return psp;
}

void set_psp(unsigned psp)
{
        asm  {
                mov ah,50h
                mov bx,psp
                int 21h
        }
}
/*****
        Check parity error
        return : 0 if not error
                 1 if error detected
******/
int p_err()
{
        return ((inportb(0x2FD) & 0x04))>>2);
}

void clear_func()
{
        con_block[1]=0;
}

void clear_err()
{
        con_block[4]=0;
}

void clear_ready()
{
        con_block[5]=0;
}

/*****
        Read data at buffer of comm. port to clear it
*****/
void clear_buff()
{
        inportb(0x2F8);
}

void enable_8259()
{
        outportb(0x21,(inportb(0x21) & 0xF7));
}

void disable_8259()
{
        outportb(0x21,(inportb(0x21) | 0x08));
}

void end_of_intr()
{
```

```
        outportb(0x20,0x20);
}

void s_port(char data)
{
        outportb(0x2F8,data);
}

int request()
{
        do  {
        }while(r_port() != ACK);
        return 1;
}

void s_last_packet()
{
        int byte;
        int max;
        do  {
                s_port(ENQ);
        }while(r_port() != ACK);
        max=block[1][8]+8;
        for(byte=0;byte<max;byte++)  {
                s_port(block[1][byte]);
                request();
        }
        con_block[6]=0xFF;
}

void s_non_ack()
{
        int byte;
        do  {
                s_port(ENQ);
        }while(r_port() != ACK);

        for(byte=0;byte<8;byte++)  {
                s_port(non_ack_pac[byte]);
                request();
        }
}

int data_ready()
{
        return (inportb(0x2FA) & 0x01);
}

char rr_port()
{
        return inportb(0x2F8);
}

char r_port()
{
        do  {
        }while(data_ready() != 0);
        return inportb(0x2F8);
}
```

```
unsigned alloc_mem(int size)
{
        unsigned seg;
        size=size/16;
        asm  {
                mov ah,48h
                int 21h
                mov seg,ax
        }
        if((seg==0x07)||(seg==0x08))  {
                puts("Memory allocation error");
                return 0;
        }
        else
                return seg;
}

unsigned free_mem(unsigned seg)
{
        int err;
        asm  {
                mov ah,49h
                mov es,seg
                int 21h
                mov err,ax
        }
        if((seg==0x07)||(seg==0x09))  {
                puts("Memory block is destroyed");
                return 0;
        }
        else
                return 1;
}

void wait_key()
{
        do  {
        }while(inportb(0x60) != 0x57);
}

void disp_n_message()
{
        int pos;
        screen=MK_FP(0xB800,0);
        screen=screen+1920;
        for(pos=0;pos < POS_MAX;pos++)  {
                old_message[pos]=screen[pos];
                screen[pos]=0xBE00 + new_message[pos];
        }
}

void disp_o_message()
{
        int pos;
        screen=MK_FP(0xB800,0);
        screen=screen+1920;
        for(pos=0;pos < POS_MAX;pos++)
                screen[pos]=old_message[pos];
}
```

```
void put_char(unsigned row,unsigned col,unsigned char data)
{
        screen=MK_FP(0xB800,0);
        screen=screen+(((row-1)*80)+(col-1));
        *screen= 0xBE00 + data;
}

void put_message(unsigned row,unsigned col,int message_num)
{
        int i;
        screen=MK_FP(0xB800,0);
        screen=screen+(((row-1)*80)+(col-1));
        for(i=0;i(30;i++)
                screen[i] = 0xBE00 + message[message_num-1][i];
}


/* INDOS.C - Functions to manage DOS flags */
/*****
Function: Init InDos Pointers
Initialize pointers to InDos Flags
*****/
void InitInDos(void)
{
        union  REGS regs,
        struct SREGS segregs;

        regs.h.ah = GET_INDOS;
        intdosx(&regs,&regs,&segregs);
        /* pointer to flag is returned in ES:BX */
        FP_SEG(indos_ptr) = segregs.es;
        FP_OFF(indos_ptr) = regs.x.bx;

        if (_osmajor ( 3)                       /* flag is one byte after InDos */
                crit_err_ptr = indos_ptr + 1;
        else if (_osmajor==3 && _osminor == 0)      /* flag is one byte before */
                crit_err_ptr = indos_ptr - 1;
        else
        {
                regs.x.ax = GET_CRIT_ERR;
                intdosx(&regs,&regs,&segregs);
                /* pointer to flag is returned in DS:SI */
                FP_SEG(crit_err_ptr) = segregs.ds;
                FP_OFF(crit_err_ptr) = regs.x.si;
        }
}

/*****
Function: DosBusy
This function will non-zero if DOS is busy
*****/
int DosBusy(void)
{
        if (indos_ptr && crit_err_ptr)
                return (*crit_err_ptr || *indos_ptr);
        else
                return 0xFFFF;              /* return dos busy if flags are not set */
}
```

```
/*****
Function: Int28DosBusy
This function will return non-zero if the InDos 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, InDOS == 1 is normal, and
indicates DOS is *not* busy; InDOS ) 1 inside INT 28 means it is.
*****/
int Int28DosBusy(void)
{
        if (indos_ptr && crit_err_ptr)
                return (*crit_err_ptr || (*indos_ptr ) 1));
        else
                return 0xFFFF;          /* return dos busy if flags are not set */
}
```

```
/*----standard include file---*/
#include (sys\timeb.h)
#include (stdio.h)
#include (string.h)
#include (io.h)
#include (math.h)
#include (errno.h)
#include (dir.h)

/*----home made include----*/
#include (hub.h)
#include (network.h)

/*-----Global variable -------*/
/*---buffer for backup packet of each terminal.----*/
        extern unsigned char bakpak[][256];
        extern int  time_re_nak[];
/*---pointer to disconnect packet---*/
        extern unsigned char disc[];
/*---variable for holding each terminal's status---*/
extern struct each_term term[];

void set_comm_var (void)
{
        asm {
                mov dx, COMM2
                inc dx
                inc dx          /*---set line control register---*/
                mov al, 8Bh     /*---set 8 data bits, odd parity---*/
                out dx, al      /*---and set DLAB(divisor latel access bit)---*/
                dec dx
                mov al, 0h      /*---set divisor LSB---*/
                out dx, al
                dec dx
                mov al, 03h     /*---set divisor MSB---*/
                out dx, al      /*---set baud rate = 38,400 baud---*/
                inc dx
                inc dx
                mov al, 0Bh
                out dx, al      /*---reset DLAB---*/
                dec dx          /*---select interrupt enable register---*/
                mov al,01h      /*---enable data ready interrupt---*/
                out dx,al
        }
}

receive (unsigned char *buf, unsigned char win)
{
        unsigned char *moving;
        const char head[4]={ DC1, DC2, DC3, DC4 };
        char ready, chrx, p_error;
        int  gen;               /*---general variable---*/

        /*-------time variable for compute timeout error------*/
        struct timeb t;
        long double std, ctd;
        float time_past;        /*---How long has the time past?---*/
```

```
/*-----NAK packet pointer-------*/
unsigned char nak[9] = { DC1, DC2, DC3, DC4, NAK, 16, 0, 0, 0};

asm {
        mov dx  , CNTLP         /*---select the terminal---*/
        mov al  , win
        and al  , 80h       /*---leave the MSB set---*/
        out dx  , al
}
moving = buf;
asm {
        mov dx  , COMM2   /*---read the char receive from the terminal.---*/
        in  al  , dx
        mov chrx, al
}
if (chrx != ENQ)   /*if the received character isn't ENQ goto main loop*/
        return (1);
asm {
        mov dx, COMM2             /*---send ACK to the selected terminal---*/
        mov al, ACK
        out dx, al
}
for (gen=1; gen(=9; gen++) {                        /*---start receive header---*/
        ftime(&t);std=t.time+t.millitm/1000.0;/*-get start time into std--*/
        do {
                asm {
                        mov dx, COMM2
                        add dx, 02h/*--select interrupt identification  register*/
                        in  al, dx          /*---read it.---*/
                        and al, 04h         /*---if the 1st bit was set,---*/
                        mov ready,al        /*--it means that a data arrived.---*/
                }                         /*--get current time into ctd--*/
                ftime (&t); ctd = t.time + t.millitm/1000.0;
                time_past = ctd-std;      /*---compute timeout error---*/
                if (time_past ) 0.5)
                        return (1);   /*it's longer than 0.5 sec and get nothing*/
        } while (!ready); /*---program passes this loop when get a char---*/
        asm {
                mov dx, COMM2
                in  al, dx                  /*---read data from COMM2---*/
                mov chrx,al
        }
        *moving = chrx;                 /*---store it in buffer---*/
        moving++;
        asm {
                mov dx, COMM2           /*---send ACK to the selected terminal---*/
                mov al, ACK
                out dx, al
        }
} /*--At this point, "moving" is pointing the 10th byte in the array--*/
if ( memcmp(buf, head, 4)!=0 || (int)buf[8])247)
        return (1);             /*--If it is an illegal header, ignore it.--*/
for (gen=1; gen(=buf[8]; gen++) {                   /*---start receive header---*/
        ftime(&t);std=t.time+t.millitm/1000.0;/*--get start time into std-*/
        do {
                asm {
                        mov dx, COMM2
                        add dx, 02h/*--select interrupt identification register-*/
                        in  al, dx              /*---read it.---*/
                        and al, 04h             /*---if the 1st bit was set,---*/
                        mov ready,al            /*--it means that a data arrived.---*/
                }                               /*--get current time into ctd--*/
```

```
                ftime (&t); ctd = t.time + t.millitm/1000.0;
                time_past = ctd-std;            /*---compute timeout error---*/
                if (time_past ) 0.5)
                        return (1);   /*it's longer than 0.5 sec and get nothing*/
        } while (!ready); /*---program passes this loop when get a char---*/
        asm {
                mov dx, COMM2
                in  al, dx                      /*---read data from COMM2---*/
                mov chrx,al
        }
        *moving = chrx;                 /*---store it in buffer---*/
        moving++;
        asm {
                mov dx, COMM2           /*---send ACK to the selected terminal---*/
                mov al, ACK
                out dx, al
        }
    }/*--------------------At this point we got a whole packet.-----------*/
    asm {
            mov dx, COMM2
            add dx, 05h                                 /*---select line status register---*/
            in  al, dx                                          /*---read the port---*/
            and al, 04h                         /*---the 2th bit indicate parity error---*/
            mov p_error, al
    }
    if (p_error) {              /*---if there is a parity error send NAK-*/
            nak[6] = win;                               /*---assign destination field---*/
            send (nak, win);
            return (1);
    }
    if (buf[4]==NAK) {                  /*---if the packet is NAK, send BAcKup PAcKet*/
            if(time_re_nak[win]++)3) return(1);/*How many TIME did we REceive NAK*/
            send (bakpak[win], win);
    }
    return (0);
}

int send (unsigned char *buf, unsigned char win)
{
    const char head[4] = { DC1, DC2, DC3, DC4 };

    /*-------time variable for compute timeout error------*/
    struct timeb t;
    long double std, ctd;
    float time_past;      /*---How long has the time past?---*/

    /*-------loop control variable------*/
    int ready, gen;

    /*---general variables---*/
    char chrx, chtx;

    if ( memcmp(head,buf,4) || (int)buf[8])247 )
            return (2);    /*-----if it's an illegal header ,then return.-----*/
    asm {
            mov dx  , CNTLP         /*---select the terminal---*/
            mov al  , win
            and al  , 80h       /*---leave the MSB set---*/
            out dx  , al
    }
```

```
        ftime (&t); std = t.time + t.millitm/1000.0;  /*--STart counting time--*/
        do {
                do {
                        asm {
                                mov dx, COMM2   /*---select COMM2, send ENQ---*/
                                mov al, ENQ
                                out dx, al
                                add dx, 02h     /*---select interrupt identifb

q*on
                        time_past = ctd-std;
                        if (time_past ) 2.0)
                                return (1);/*-if didn't get respond within 2 sec, return*/
                } while (!ready);    /*-if didn't get respond, keep on sending ENQ*/
                asm {
                        mov dx,   COMM2                 /*--read the character--*/
                        in  al,   dx
                        mov chrx, al
                }
        } while (chrx!=ACK);            /*--if the responding char isn't ACK, return*/
        for (gen=0; gen(9+buf[8]; gen++) {
                chtx = buf[gen];
                asm {
                        mov dx, COMM2                   /*--send a character--*/
                        mov al, chtx
                        out dx, al
                }
                ftime (&t); std = t.time + t.millitm/1000.0;/*--STart counting time--*/
                do {
                        asm {
                                add dx, 02h             /*---select interrupt identification  register---*/
                                in  al, dx              /*---read it.---*/
                                and al, 04h             /*---if the 1st bit was set,---*/
                                mov ready,al            /*--it means that a data arrived.---*/
                        }
                        ftime (&t); ctd =t.time +t.millitm/1000.0;/*--get Current time--*/
                        time_past = ctd-std;
                        if (time_past ) 0.5)
                                return(1);/*if didn't get next byte within .5 sec, return*/
                } while (!ready);               /*--keep on checking for incoming data--*/
        }
        if (buf[4]!=NAK) {/*---if the sent packet isn't NAK, backup the packet for retransmittion---*/
                time_re_nak[win] = 0;           /*--stop counting NAK packet---*/
                memcpy (bakpak[win] , buf, 256);
        }
        if ( buf[6]==16 && buf[7])0)
                puts ((char *)&buf[9]);                 /*---display message---*/
        return (0);                     /*---the routine has finished its job.---*/
}

void Emailre (unsigned char *buf, unsigned char win)
{
        union four      a;
        int             handle;
        unsigned char Econf[27]={DC1,DC2,DC3,DC4,EMCON,HUB,0,0,16};
        long            big;
        char          g[66];
        char            ext[MAXEXT];
        int             b;                      /*---general variable---*/
```

```
        Econf[6] = win;
        term[win].fp =fopen((signed char *)&buf[9], "rb");/*---open the file---*/
        if(term[win].fp==NULL){/*if there's a problem opening the file,disconnect*/
                disc[6] =win;
                disc[7] =1;             /*---there is message for the terminal---*/
                strcpy ((signed char *)&disc[9], strerror(errno));/*-store the message next to the header-*
                disc[8] = strlen((signed char *)&disc[9]);/*--store length of the following message field---
                send (disc, win);               /*---send disconnect packet---*/
        }
        term[win].pst=term[win].cst;    /*----store Previous terminal's STatus*/
        term[win].cst=buf[4];           /*---modify Current  terminal's STatus*/
        handle = fileno(term[win].fp);              /*---begin to packetize---*/
        big       = filelength(handle);
        a.word[0] = ceil(big/247.0);       /*calculate number of total packets*/
        Econf[9]  = a.byte[0];                    /*fill in total packet field*/
        Econf[10] = a.byte[1];
        a.dword   = big;
        Econf[11] = a.byte[0];                    /*fill in packet size field*/
        Econf[12] = a.byte[1];
        Econf[13] = a.byte[3];
        fnsplit((signed char *)&buf[9],g,g,(signed char *)&buf[14],ext);
        b=strlen((signed char *)&buf[14]);            /*fill in filename field*/
        strcpy((signed char *)&buf[14+b],ext);/*write extension next to the filename*/
        send (Econf, win);                      /* send E-mail confirm packet */
}

void Pemail (unsigned char win)
{
        unsigned char buf[256]={DC1,DC2,DC3,DC4,EMnorm,HUB,0,0,0};/*---header--*/
        long int big, pos, left;
        int handle;
        unsigned char willrd;

        buf[6]=win;
        pos = ftell (term[win].fp);             /*---pos=no. of bytes sent.--*/
        handle = fileno(term[win].fp);
        big = filelength (handle);                      /*---big=file's size---*/
        left = big-pos;                         /*---left=no. of bytes left---*/
        if (left==0) {          /*---if there're no bytes left, disconnect---*/
                send (disc, win);
                return ;
        }
                if ( left(247 )
                        willrd = (unsigned char)left;
                else
                        willrd = 247;
                buf[8]=willrd;   /*---set no. of information byte in the packet---*/
                fread (&buf[9], 1, willrd, term[win].fp); /*---store infomation---*/
        send (buf, win);                                /*---send packet---*/
}

void reslist (unsigned char *buf, unsigned char win)
{
        int handle, done;
        struct ffblk blk;

        term[win].pst = term[win].cst;          /*---store current status---*/
        term[win].cst = WAIT;
        strcpy (term[win].fname, TMPATH); /*-path name to create temporary file*/
        handle = creattemp (term[win].fname, 0);/*--write file name into status table--*/
        term[win].fp = fdopen (handle, "wb");
        done = findfirst ((signed char *)&buf[9], &blk, 0); /*--write file's list into the temporary file--*
```

```
        while(!done) {
                fprintf (term[win].fp,"\n%s\t%ld",blk.ff_name,blk.ff_fsize);
                done = findnext (&blk);
        }                                                              /*--finish writing-
}

void reswho (unsigned char win)
{
        int handle, gen;    /*---handle and general variable---*/

        term[win].pst = term[win].cst;                    /*---store current status---*/
        term[win].cst = WAIT;

        strcpy (term[win].fname, TMPATH);
        handle = creattemp (term[win].fname, 0);
        term[win].fp = fdopen(handle,"wb");
        for (gen=0; gen(16; gen++)        /*---examine each terminal---*/
                if (term[gen].cst!=OFF)/*-report which terminal is in the network-*/
                        fprintf (term[win].fp,"%s%d",term[gen].user,term[gen].cst);
}

void resre (unsigned char *buf, unsigned char win, int stt)
{
        if(term[buf[6]].cst==BROWSE) {/*---if the destination is available, pass the packet---*/
                term[win].pst=term[win].cst;          /*---store previous status---*/
                term[win].cst=stt;                    /*---modify current status---*/
                send (buf, buf[6]);     /*---pass the packet to the destination---*/
        }
        else { /*---the destination is busy, send the disconnect packet to the source---*/
                disc[6] = buf[6];                      /*---prepare disconnect packet---*/
                disc[7] = 1;
                strcpy ((signed char *)&disc[9],"the destination terminal is now busy, plese wait and try a
                disc[9] = (unsigned char) strlen ((signed char *)&disc[9]);
                send (disc,buf[6]);
        }
}

void rescon (unsigned char *buf, unsigned char win, int stt)
{
        if( (term[buf[6]].cst!2) == CHATre ) {/*--if the destination is waiting the confirm, pass the confi
                term[win].pst = term[win].cst;          /*---store previous status---*/
                term[win].cst = stt;
                term[buf[6]].cst = stt;                 /*---modify current status---*/
                send (buf, buf[6]);     /*---pass the packet to the destination---*/
        }
        else {          /*---the destination doesn't expecting a confirmation---*/
                disc[6] = win;
                disc[7] = 1;
                strcpy ((signed char *)&disc[9], "Your terminal has gone nut.");
                disc[8] = (unsigned char) strlen ((signed char *)&disc[9]);
                send (disc, win);
        }
}
```

```
void emcon (unsigned char *buf, unsigned char)
{
        char name[10];
        unsigned char conf[256];

        strcpy (name, &buf[15], 12);
        term[win].fp = fopen(name,"wb");
        memcpy (conf,buf,256);
        conf[5]=HUB;
        conf[6]=buf[5];
        term[win].pst=term[win].cst;     /*----store Previous terminal's STatus*/
        term[win].cst=buf[4];            /*---modify Current  terminal's STatus*/
        send (conf, win);                    /*---send E-mail confirm packet---*/
}
```

```
/*----Program Chat.c for online communication between two terminals---*/
#include (stdio.h)
#include (stdlib.h)

#include "network.h"
#include "term.h"

void main ()
{
        int answer;           /*---character to hold answer "yes" or "no"---*/
        char local[256];          /*---local buffer for holding a packet---*/

        /*---packet pattern of various---*/
        char chatcon[9]={DC1,DC2,DC3,DC4,CHATcon,SOURCE,0,0,0};
        char disc[9]   ={DC1,DC2,DC3,DC4,DISC   ,SOURCE,0,0,0};
        char chat[256] ={DC1,DC2,DC3,DC4,CHAT   ,SOURCE,0,1,0};
        char chatre    ={DC1,DC2,DC3,DC4,CHATre ,SOURCE,0,0,0};

        unsigned char length;               /*---No. of data in the packet---*/

        /*---segment and offset of far packet buffer---*/
        unsigned int segment, offset;
        void far *packet;         /*---far pointer to the received packet---*/
        int called;

        getpaccpy (&segment,&offset);/*get segment, offset of far packet buffer*/
        packet = MK_FP (segment, offset);/*make far pointer to far packet buffer*/
        if (pacred()) {                                 /*---!!!!!!!!!!!!!!!!---*/
                cpypac(packet,local);/*--copy received packet into local buffer--*/
                if (local[4]!=CHATre){/*---if there's a packet exist in buffer---*/
                        if (local[4]==FTre)/*-if the packet is file transfer packet--*/
                                printf ("\nYou've received file transfer packet, please run file transfer program");
                        else {/*if existing packet isn't FT packet, it's an error*/
                                printf ("\nPiyatad's stupidity cuase this to happen.");
                                resetpack ();
                        }
                        exit (EXIT_FAILURE);        /*---operation failed---*/
                }
                resetpack ();                   /*---reset packet ready flag---*/
                chatcon[6]=local[5];          /*---set destination terminal---*/
                send (chatcon);                         /*---!!!!!!!!!!!!!!!!---*/
                if (_AL==1) {/*---If the sent routine cannot complete its task---*/
                        printf ("\nCannot send packet properly");
                        setpack ();           /*---leave the packet unread---*/
                        exit (EXIT_FAILURE);          /*---operation failed---*/
                }
        }else{/*-there's no packet in buffer while calling program do following*/
                printf ("\nWhat terminal number do you want to talk to ?");
                scanf ("\n%d",&called);   /*---input destination terminal---*/
                while (called>15) {
                        printf("\nThere is less than 16 terminals in our network");
                        printf("\nPlease enter the number again");
                        scanf ("\n%d",&called);
                }
                chatre[6]=called;/*---fill in destination field in the packet---*/
                send (chatre);          /*---send chat mode request packet---*/
                if (_AL==1) {          /*---if cannot send packet properly---*/
                        printf ("\nStupid programer cause this error.");
                        exit (EXIT_FAILURE);
                }
```

```
    }
    for (;;) {
        if (pacred()) {            /*---check if a packet arrive---*/
            resetpack();           /*---reset packet ready flag---*/
            cpypac(packet,local);/*---copy it to local buffer---*/
            if (lobuf[4]==DISC) {     /*---if it is disconnect packet---*/
                disc[6]=lobuf[5];/*---store destination field in disc---*/
                send (disc);        /*---!!!!!!!!!!!!!!!!!---*/
            }
            if (lobuf[4]==CHAT)                  /*---if it's chat packet---*/
                continue;                       /*---go detect another packet---*/
            else {
                printf ("\nHub've gone nut");
                disc[6]=lobuf[5];
                send (disc);
                setpack();
                exit (EXIT_FAILURE);
            }
        }
        gets (&chat[9]);           /*---let user input the data to talk to---*/
        if (&chat[9]=='\x0') {              /*---if press pure enter---*/
            printf ("\nYou have not type anything except ENTER.");
            printf ("\nDo you want to quit?(Y/N)\n");
            for (;;) {                          /*---force him to answer---*/
                answer = toupper(getch());
                switch (answer) {
                    case 'Y': send (disc);
                              exit (EXIT_SUCCESS);
                    case 'N': break;
                    default : continue;
                }
                break;
            }

        }
        length = strlen(&chat[9]);      /*---tell the length of data---*/
        if (length)(char)247)    /*ignored the data longer than 247 bytes*/
            length=247;
        send (chat);                            /*---send chat packet---*/
    }
}



/*---Program for sending disconnect packet and reset packet ready flag---*/
#include "term.h"
#include "network.h"

void main ()
{
    char disc[9]={DC1,DC2,DC3,DC4,DISC   ,SOURCE,HUB,0,0};

    resetpack();
    send (disc);
}
```

```
/*---Program for sending Electronic mail---*/
#include (stdio.h)
#include (stdlib.h)
#include (string.h)

#include "term.h"
#include "network.h"

void main ()

{
        char local[256];            /*---local buffer for holding a packet---*/
        int gen;                                                    /*---general variable---*/

        /*----time variable for compute timeout error----*/
        struct timeb t;
        long double std,ctd;
        float time_past;

        /*---segment and offset of far packet buffer---*/
        unsigned int segment, offset;
        void far *packet;           /*---far pointer to the received packet---*/

        char answer;

        /*---variable holding detail about file---*/
        union two  total;                                   /*---total packet---*/
        union four size;
        char hubname[80]={0,0};
        char tername[80]={0,0};

        /*---packet---*/
        char emreq [256] ={DC1,DC2,DC3,DC4,EM   ,SOURCE,HUB,0,0};
        char emconf[256] ={DC1,DC2,DC3,DC4,EMCON,SOURCE,HUB,0,18};
        char pem   [256] ={DC1,DC2,DC3,DC4,PEM ,SOURCE,HUB,0,0};
        char disc  [9]   ={DC1,DC2,DC3,DC4,DISC,SOURCE,HUB,0,0};
        chat ftnom [256] ={DC1,DC2,DC3,DC4,FTnorm ,SOURCE,HUB,0,0};

        FILE *fp;
        int handle;                                 /*---file pointer and file handle---*/

        getpaccpy (&segment,&offset);/*get segment, offset of far packet buffer*/
        packet = MK_FP (segment,offset);/*make far pointer to far packet buffer*/
        if (pacred()) {                 /*---if there is a packet in buffer---*/
                printf ("\nYou've receive a packet.Run 'discon' if you want to ignore it.");
                exit (EXIT_SUCCESS);
        }
        printf ("\nyou want to Receive or Transmit a file with hub\n");
        for (;;) {                                          /*---force him to answer---*/
                answer = toupper(getch());
                switch (answer) {
                        case 'T': break;
                        case 'R': break;
                        default : continue;
                }
                break;
        }
        if (answer=='R') {                          /*---user want to receive---*/
                printf ("\nEnter filename you want to receive.\n");
                scanf ("%s",hubname);
```

```
for (;;) {
        printf ("\nEnter name in which we will save the file.\n");
        scanf  ("%s",tername);
        if (!access(name, 0)) {
                printf ("\nThe file exist.");
                printf ("\nWant to Overwrite, Rename or Cancel.(O,R,C)");
                for (;;) {                      /*---force him to answer---*/
                        answer = toupper(getch());
                        switch (answer) {
                                case 'O': break;
                                case 'R': break;
                                case 'C': exit (EXIT_SUCCESS);
                                default : continue;
                        }
                        break;
                }
        } else answer=1;
        if (answer=='R') continue;
        break;
}
emreq[8]=strlen(hubname);           /*---prepare E-mail request packet---*/
strcpy (&emreq[9],hubname);
fopen (tername,"wb");
send (emreq);                                   /*---!!!!!!!!!!!!!!!---*/
if (_AL) {
        printf ("\nErroe:sending E-mail request.");
        exit (EXIT_FAILURE);
}
ftime(&t);std=t.time+t.millitm/1000.0;          /*-STart counting time-*/
do {
        ftime(&t);ctd=t.time+t.millitm/1000.0;          /*get Current time*/
        time_past = ctd-std;                    /*---wait for new packet---*/
        if (time_past > 120) {          /*-can't wait longer than 2 sec.-*/
                printf ("\ncannot receive confirm packet in 2 min.");
                send (disc);
                fclose (fp);
                exit (EXIT_FAILURE);
        }
} while (!pacred())
cpypac(packet,local);       /*--copy received packet into local buffer--*/
resetpack ();                           /*---reset packet ready flag---*/
if (local[4]==DISC) {                   /*---Is it the last packet---*/
        send (disc);
        fclose (fp);
        exit (EXIT_SUCCESS);
}
if (local[4]!=EMCON) {
                printf ("\nError:receiving inappropriate file.");
                send (disc);
                fclose (fp);
                exit (EXIT_SUCCESS);
}
for (;;) {
        send (pem);
        ftime(&t);std=t.time+t.millitm/1000.0;    /*-STart counting time-*/
        do {
                ftime(&t);ctd=t.time+t.millitm/1000.0; /*get Current time*/
                time_past = ctd-std;            /*---wait for new packet---*/
                if (time_past > 5) {    /*-can't wait longer than 5 sec.-*/
                        printf ("\ncannot receive any packet in 5 sec.");
```

```
        scanf ("%s",name);
        if (access(name,0)) {     /*---check for file's existence---*/
                printf ("\nthe file doesn't exist, Cancel or Retry(C/R)");
                for (;;) {              /*---force him to answer---*/
                        answer = toupper(getch());
                        switch (answer) {
                                case 'R': break;
                                case 'C': exit (EXIT_SUCCESS);
                                default : continue;
                        }
                        break;
                }
        } else break;
   }
printf ("\nPlease input terminal number\n");
   scanf ("%d",&called);            /*---input destination terminal---*/
   while (called>15) {
        printf("\nThere is less than 16 terminals in our network");
        printf("\nPlease enter the number again\n");
        scanf ("%d",&called);
   }
   fp = fopen (name,"rb");
   ftreq[6] = called;                        /*---prepare packet FTreq---*/
   ftnom[6] = called;
   ftreq[8] = 17;
   handle = fileno(fp);                      /*---find file's size---*/
   size.li = filelength(handle);
   ftreq[11]=size.i[0];
   ftreq[12]=size.i[1];
   ftreq[13]=size.i[2];
   total.b = ceil((double)size.li/247);
   ftreq[9] =total.c[0];
   ftreq[10]=total.c[1];
   fnsplit(name,&name[40],&name[40],&ftreq[14],&name[60]);
   gen = strlen (&ftreq[14]);
   strcpy (&ftreq[14+gen],&name[60]);
   send (ftreq);
   if (_AL) {
        printf ("\nsending FT request failure.");
        exit (EXIT_FAILURE);
   }
   ftime(&t);std=t.time+t.millitm/1000.0;       /*-STart counting time-*/
   do {
        ftime(&t);ctd=t.time+t.millitm/1000.0;    /*get Current time*/
        time_past = ctd-std;              /*---wait for new packet---*/
        if (time_past > 120) {     /*-can't wait longer than 2 sec.-*/
             printf ("\ncannot receive confirm packet in 2 min.");
             send (disc);
             fclose (fp);
             exit (EXIT_FAILURE);
        }
   } while (!pacred())
   cpypac(packet,local);   /*--copy received packet into local buffer--*/
   resetpack ();                        /*---reset packet ready flag---*/
   if (local[4]==DISC) {/*-if called terminal didn't want to receive file-*/
        disc[6]=local[5];
        send(disc);
   }
   if (local[4]!=FTcon) {   /*---the received packet should be FTcon---*/
        printf ("\nError :received inappropriate packet");
```

```
} while (!pacred())
copypac(packet,local);                    /*--copy received packet into local buffer--*/
resetpack ();                                        /*---reset packet ready flag---*/
if (local[4]!=EMCON) {                    /*---the received packet should be FTcon-  */
        printf ("\nError :received inappropriate packet");
        exit (EXIT_FAILURE);
}
for (;;) {                              /*---this loop is for sending packet until EOF---*/
        for (gen=0;gen(247;gen++) {                  /*---put data into the packet---*/
                ftnom[gen+9]=getc(fp);
                if (ftnom[gen+9]==EOF) break;
        }
        ftnom[8]=gen+1;
        send (ftnom);
        if (ftnom[gen+9]==EOF) {          /*---if it's the last packet,disconnect---*/
                disc[6]=local[5];
                send (disc);
                break;
        }
}
fclose(fp);                                                  /*---close file---*/
}
```

```
/*---Program for transfering file---*/
#include (stdlib.h)
#include (stdio.h)
#include (string.h)
#include (io.h)
#include (math.h)

#include "term.h"
#include "network.h"

void main ()
{
        char local[256];            /*---local buffer for holding a packet---*/

        int gen;                                        /*---general variable---*/

        /*---segment and offset of far packet buffer---*/
        unsigned int segment, offset;
        void far *packet;          /*---far pointer to the received packet---*/
        unsigned char called;

        /*---variable holding detail about file---*/
        union two  total;                               /*---total packet---*/
        union four size;
        char name[80]={0,0};
        char onlyname[13];

        /*---packet---*/
        char disc[9]   ={DC1,DC2,DC3,DC4,DISC   ,SOURCE,0,0,0};
        char ftreq[256]={DC1,DC2,DC3,DC4,FTreq  ,SOURCE,0,0,0};
        chat ftnom[256]={DC1,DC2,DC3,DC4,FTnorm ,SOURCE,0,0,0};
        char ftcon[256];

        char answer;

        void far* point;              /*- -pointer to file transfer buffer---*/

        FILE *fp;                       /*---file pointer and file handle---*/
        int   handle;

        /*----time variable for compute timeout error----*/
        struct timeb t;
        long double std,ctd;
        float time_past;

        getpaccpy (&segment,&offset);/*get segment,.offset of far packet buffer*/
        packet = MK_FP (segment,offset);/*make far pointer to far packet buffer*/
        if (pacred()) {                 /*---if there is a packet in buffer---*/
                cpypac(packet,local); /*--copy received packet into local buffer--*/
                if (local[4]!=FTre) {
                   if (local[4]==CHATre) {
                           printf ("\nPlease run 'chat' program");
                           exit (EXIT_FAILURE);
                        } else {
                           printf ("\nError:receive an inappropriate packet");
                           resetpack ();
                           exit (EXIT_FAILURE);
                        }
                }
```

```
        /*--value from packet---*/
        size.i[0]=local[11]; size.i[1]=local[12];
        size.i[2]=local[13]; size.i[3]=0;
        total.c[0]=local[9]; total.c[1]=local[10];
        memcpy (name, &local[15], 12);
        printf ("\ntotal packet \t= %d",total.b);/*display file's detail*/
        printf ("\nfile size \t= %ld"  ,size.li);
        printf ("\nfile name \t= %s"   ,name);
        for (;;) {
                if (!access(name, 0)) {
                        printf ("\nThe file exist.");
                        printf ("\nWant to Overwrite, Rename or Cancel.(O,R,C)");
                        for (;;) {       /*---force him to answer---*/
                                answer = toupper(getch());
                                switch (answer) {
                                        case 'O': break;
                                        case 'R': printf ("\nplease input new filename\n");
                                                  scanf ("%s",name);
                                                  break;
                                        case 'C': disc[6]=local[5];
                                                  send (disc);    /*---!!!!!!!!!!!!!---*/
                                                  exit (EXIT_SUCCESS);
                                        default : continue;
                                }
                                break;
                        }
                } else answer=1;
                if (answer=='R') continue;
                break;
        }
        fp = fopen(name,"wb");                    /*---open file---*/
        memcpy (ftcon,local,256);  /*---prepare FT confirm packet---*/
        ftcon[5]=SOURCE;
        ftcon[6]=local[5];
        send (ftcon);                  /*---send FT confirm packet---*/
        if (_AL)
                printf("\nunsuccessful sending FT confirm");
        for (;;) {
                ftime(&t);std=t.time+t.millitm/1000.0;/*-START counting time-*/
                do {
                        ftime(&t);ctd=t.time+t.millitm/1000.0;/*get Current time*/
                        time_past = ctd-std;/*---wait for new packet---*/
                        if (time_past > 2) {/*-can't wait longer than 2 sec.-*/
                                printf ("\ncannot receive another packet in2 sec.");
                                send (disc);
                                fclose (fp);
                                exit (EXIT_FAILURE);
                        }
                } while (!pacred())
                cpypac(packet,local);/*--copy received packet into local buffer--*/
        resetpack();
                if (local[4]==DISC) {/*---Is it the last packet---*/
                        send (disc);
                        fclose (fp);
                        exit (EXIT_SUCCESS);
                }
                fwrite (&local[9],1,local[8],fp);/*---write data into file---*/
        }
}
for (;;) {
        printf ("\nPlease enter filename(and pathname)\n");
```

```
                                send (disc);
                                fclose (fp);
                                exit (EXIT_FAILURE);
                        }
                } while (!pacred())
                copypac(packet,local);    /*copy received packet into local buffer*/
                resetpack ();                    /*---reset packet ready flag---*/
                if (local[4]==DISC) {              /*---Is it the last packet---*/
                        send (disc);
                        fclose (fp);
                        exit (EXIT_SUCCESS);
                }
                fwrite (&local[9],1,local[8],fp);    /*---write data into file---*/
        }
    }
    for (;;)
        printf ("\nPlease enter filename(and pathname)\n");
        scanf ("%s",ternane);
        if (access(ternane,0)) {              /*---check for file's existence---*/
                printf ("\nthe file doesn't exist, Cancel or Retry(C/R)");
                for (;;) {                              /*---force him to answer---*/
                        answer = toupper(getch());
                        switch (answer) {
                                case 'R': break;
                                case 'C': exit (EXIT_SUCCESS);
                                default : continue;
                        }
                        break;
                }
        } else break;
    }
fp = fopen (ternane,"rb");
    printf ("\nplease Enter name of the person you want to mail to\n");
    scanf ("%s",hubname);
    memcpy (&emconf[14],hubname,7);
    emconf[6]=HUB;
    emconf[8]=17;
    handle = fileno(fp);
    size.li = filelength(handle);
    emconf[11]=size.i[0];
    emconf[12]=size.i[1];
    emconf[13]=size.i[2];
    total.b = ceil((double)size.li/247);
    emconf[9] =total.c[0];
    emconf[10]=total.c[1];
    send (emconf);
    if (_AL) {
        printf ("\nsending E-mail request failure.");
        exit (EXIT_FAILURE);
    }
    ftime(&t);std=t.time+t.millitm/1000.0;                    /*-STart counting time-*/
    do {
        ftime(&t);ctd=t.time+t.millitm/1000.0;              /*get Current time*/
        time_past = ctd-std;                            /*---wait for new packet---*/
        if (time_past > 30) {                    /*-can't wait longer than 30 sec.-*/
            printf ("\ncannot receive confirm packet in 30 sec.");
            send (disc);
            fclose (fp);
            exit (EXIT_FAILURE);
        }
```

```
        exit (EXIT_FAILURE);
    }
    for (;;) {          /*---this loop is for sending packet until EOF---*/
        for (gen=0;gen<247;gen++) { /*---put data into the packet---*/
            ftnom[gen+9]=getc(fp);
            if (ftnom[gen+9]==EOF) break;
        }
        ftnom[8]=gen+1;
        send (ftnom);
        if (ftnom[gen+9]==EOF) {/*---if it's the last packet,disconnect---*/
            disc[6]=local[5];
            send (disc);
            break;
        }
    }
    fclose(fp);                                          /*---close file---*/
}
```

```
          /*---Program for asking about files in the hub---*/
#include (stdlib.h)
#include (stdio.h)
#include (mem.h)

#include "term.h"
#include "network.h"

void main ()
{
        void *buffer;                   /*---buffer to store the information---*/
        char local[256];                                /*---local packet buffer---*/

        /*---packet pattern of various---*/
        char lis[9]   ={DC1,DC2,DC3,DC4,LIST,SOURCE,HUB,0,0};
        char pemail[9]={DC1,DC2,DC3,DC4,PEM, SOURCE,HUB,0,0};
        char dis[9]   ={DC1,DC2,DC3,DC4,DISC,SOURCE,HUB,0,0};

        /*---segment and offset of far packet buffer---*/
        unsigned int segment, offset;
        void far *packet;          /*---far pointer to the received packet---*/

        int next = 0;       /*---int tell how much information we receive---*/
        int gen;                                        /*---general variable---*/

        /*----time variable for compute timeout error----*/
        struct timeb t;
        long double std,ctd;
        float time_past;

        getpaccpy(&segment,&offset);/*get segment, offset of far packet buffer*/
        packet = MK_FP (segment,offset);/*make far pointer to far packet buffer*/
        buffer = malloc (3000);/*--allocate memmory for buffering information--*/
        send (lis);                                     /*---send "list" packet---*/
        for (;;) {
          /*--send poll E-mail packet to ask hub to give information packet-*/
                send (pemail);
                ftime(&t);std=t.time+t.millitm/1000.0;/*--STart counting time--*/
                do {
                        ftime (&t);ctd=t.time+t.millitm/1000.0;/*--get Current time--*/
                        time_past = ctd-std;        /*---wait for a packet---*/
                        if (time_past > 3) {
                                printf ("\nI cannot wait any more.");
                                free (buffer);/*---free allocated memory before exit---*/
                                exit (EXIT_FAILURE);/*-can't wait longer than 3 sec.-*/
                        }
                } while(!pacred())                          /*---!!!!!!!!!!!!!---*/
                cpypac(packet,local);/*copy the arrived packet to the local buffer*/
                resetpack();                    /*---reset packet ready flag---*/
                if (local[4]!=DISC) {
                        memcpy(local,&buffer[next],local[8]);/*-save the information-*/
                        next+=local[8];         /*---information increase---*/
                        continue;/*---if the packet isn't "disc",wait for the next---*/
                }
                send (dis);                                                 /*---!!!!!!!!!!!!!---*/
        }
        for (gen=0; gen<next; gen++)
                putchar (buffer[gen]);
        free(buffer);
}
```

```
/*---Program for asking Hub about WHO is working in the network---*/

#include (stdlib.h)
#include (stdio.h)
#include (mem.h)

#include "term.h"
#include "network.h"

void main ()
{
        void *buffer;                   /*---buffer to store the information---*/
        char local[256];                        /*---local packet buffer---*/

        /*---packet pattern of various---*/
        char who[9]   ={DC1,DC2,DC3,DC4,WHO ,SOURCE,HUB,0,0};
        char pemail[9]={DC1,DC2,DC3,DC4,PEM, SOURCE,HUB,0,0};
        char dis[9]   ={DC1,DC2,DC3,DC4,DISC,SOURCE,HUB,0,0};

     /*---segment and offset of far packet buffer---*/
        unsigned int segment, offset;
        void far *packet;               /*---far pointer to the received packet---*/

        int next = 0;           /*---int tell how much information we receive---*/
        int gen;                /*---general variable---*/

        /*----time variable for compute timeout error----*/
        struct timeb t;
        long double std,ctd;
        float time_past;

        getpaccpy(&segment,&offset); /*get segment, offset of far packet buffer*/
        packet = MK_FP (segment,offset);/*make far pointer to far packet buffer*/
        buffer = malloc (3000);/*--allocate memmory for buffering information--*/
        send (who);                                             /*---send "list" packet---*/
        for (;;) {
                /*--send poll E-mail packet to ask hub to give information packet-*/
                send (pemail);
                ftime(&t);std=t.time+t.millitm/1000.0;   /*--START counting time--*/
                do {
                        ftime (&t);ctd=t.time+t.millitm/1000.0;/*--get Current time--*/
                        time_past = ctd-std;            /*---wait for a packet---*/
                        if (time_past > 3) {
                                printf ("\nI cannot wait any more.");
                                free (buffer); /*---free allocated memory before exit---*/
                                exit (EXIT_FAILURE);   /*-can't wait longer than 3 sec.-*/
                        }
                } while(!pacred());                             /*---!!!!!!!!!!!!!---*/
                cpypac(packet,local);/*copy the arrived packet to the local buffer*/
                resetpack();                    /*---reset packet ready flag---*/
                if (local[4]!=DISC) {
                        memcpy(local,&buffer[next],local[8]);/*-save the information-*/
                        next+=local[8];                 /*---information increase---*/
                        continue;/*---if the packet isn't "disc",wait for the next---*/
                }
                send (dis);                                                     /*---!!!!!!!!!!!!!---*/
        }
        for (gen=0; gen<next; gen++)
                putchar (buffer[gen]);
        free(buffer);
}
```

```
/*----Header file that defines important constant for using in the network--*/
/*---define name to important ASCII character---*/
#define ENQ 5
#define ACK 6
#define DC1 17
#define DC2 18
#define DC3 19
#define DC4 20
#define NAK 21

#define HUB    16

/*---packet type---*/
#define FTcon   2                   /*---File transfer confirm packet---*/
#define FTnorm  3                   /*---Normal file transfer packet---*/
#define DISC    4                   /*---Disconnect packet      ---*/
#define REGI    5                   /*---Registration packet    ---*/
#define EMCON   7                   /*---E-mail confirm packet  ---*/
#define EMnorm  8                   /*---ordinary E-mail packet ---*/
#define PEM     9                   /*---Poll E-mail packet     ---*/
#define CHATcon 11                  /*---Chat mode confirm packet---*/
#define WHO     13                  /*---Who('s in the network) packet---*/
#define LIST    14                  /*---List packet            ---*/

#define OFF     0                   /*---0. OFF mode            ---*/
#define FTre    1                   /*---1. FT request mode     ---*/
#define FT      2                   /*---2. FT mode             ---*/
#define CHATre 10                   /*---3. chat request mode   ---*/
#define CHAT   12                   /*---4. chat mode           ---*/
#define UNINT   5                   /*---5. uninterruptable mode---*/
#define EM      6                   /*---6. receive E-mail mode ---*/
#define BROWSE  7                   /*---7. Browsing mode       ---*/
#define WAIT    8                   /*---8. WAITing for a response-*/
```

```
/*---This file contains prototype of the functions and constant
       used in program "hub1.c"---*/

/*---define communication port---*/
#define COMM2 0x2F8
#define CNTLP 0x303

#define TMPATH "d:\\"          /*---path name to create temporary file---*/

int  receive     (unsigned char *, unsigned char);
int  send        (unsigned char *, unsigned char);
void take_action (unsigned char *, unsigned char);
void Emailre     (unsigned char *, unsigned char);
void reslist     (unsigned char *, unsigned char);
void resre       (unsigned char *, unsigned char, int);
void rescon      (unsigned char *, unsigned char, int);
void emcon       (unsigned char *, unsigned char);
void reswho      (unsigned char);
void Pemail      (unsigned char);

void set_comm_var (void);

/*---variable for holding each terminal's status---*/
/*---terminal status table---*/
/*---| username | current mode | previous mode | connection with |---*/
struct each_term {
                char user[40];
                char fname[128];
                int cst;
                int pst;
                FILE *fp;
};

union four {
                char byte[4];
                int  word[2];
                long int dword;
};
```

```
/*----Program for hub. It calls a lot of funtions in file HUB_SUP.C---*/
/*----define 8255 port a, b, c, & control port---*/
#define PORTA 0x300
#define PORTB 0x301
#define PORTC 0x302

/*----standard include file---*/
#include (stdio.h)
#include (string.h)

/*----home made include file----*/
#include "hub.h"
#include "network.h"

/*-----Global variable -------*/
/*---buffer for backup packet of each terminal.----*/
        unsigned char bakpak[16][256];
        int  time_re_nak[16]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
/*---pointer to disconnect packet---*/
        unsigned char disc[256] = {DC1,DC2,DC3,DC4,DISC,HUB};

/*---variable for holding each terminal's status---*/
struct each_term term[16];

void main ()
{
/*----------------------------local variable-----------------------*/
        union four a;

/*---buffer that stores just received packet---*/
        unsigned char rebuf[256];

/*-----win=terminal number that wins competition in each loop.-----*/
        unsigned char win;
        int gen1, gen2;                                 /*---general variable---*/
/*------ending all declarations of every variable in the program-------*/
/*--------------------------------------------------------------------*/
        for (gen1=0; gen1 (16; gen1 ++) {       /*----assign initial value-----*/
                strcpy (term[gen1].user, "\0");
                term[gen1].cst = 0;
                term[gen1].pst = 0;
        }
        set_comm_var ();                        /*---set communication variable---*/
        for (;;) {                              /*-----main loop is here-----*/
                asm {
                        mov dx,CNTLP
                        mov al,0Fh
                        out dx,al                       /*---reset 8255 port A & B---*/
                        mov al,8Fh
                        out dx,al                       /*--------let in ENQ---------*/
                        mov dx,PORTA            /*-----detect a terminal-----*/
                        in  ax,dx                       /*!!!this instruction may cause error!!!*/
                        mov a.byte[0],al
                        mov a.byte[1],ah
                }
                for (gen1=0; gen1(16; gen1++)   /*---report the detected terminals--*/
                        if ( !((a.word[0])>(gen1+1)) & 0xFFFE) ) {
                                printf ("\nTerminal %d get the connection.", gen1+1);
                                win = (char) gen1;
```

```
            asm {
                    mov dx  , CNTLP         /*---select the terminal---*/
                    mov al  , win
                    and al  , 80h       /*---leave the MSB set---*/
                    out dx  , al
            }
            if (receive (rebuf, win)) continue;/*if can't receive, detect new terminal*/
            if (term[win].cst==OFF) {
                    send (disc, win);
                    continue;
            } /*if the terminal=off, ignore the packet and detect new terminal*/
            take_action (rebuf, win);
    }                                       /*----------main loop end here-----------*/
}

void take_action (unsigned char *buf, unsigned char win)
{
    unsigned char whopac[256];

    if (buf[6]==16)
            switch (buf[4]) {/*if the packet was sent to hub,do the following command*/
                    case NAK    : send (bakpak[win], win);
                                  break;
                    case EM     : Emailre(buf, win);
                                  break;
                    case DISC   : if (term[win].cst==EM)
                                        fclose (term[win].fp);/*--if transfering a file, stop it.-
                                  if (term[win].cst==WAIT){/*--if finish receiving the response, de
                                        fclose (term[win].fp);
                                        unlink (term[win].fname);
                                  }
                                  term[win].cst=term[win].pst; /*-reset to prevoius status-*/
                                  break;
                    case REGI   : if (buf[39]=='0') term[win].cst=OFF;
                                  if (buf[39]=='C') term[win].cst=?;
                                  term[win].cst=BROWSE;
                                  break;
                    case PEM    : Pemail (win);
                                  break;
                    case WHO    : reswho (win);
                                  break;
                    case LIST   : reslist (buf, win);
                                  break;
                    case FTnorm : fwrite(&buf[9],1,buf[8],term[win].fp);
                                  break;
                    case EMCON  : emcon (buf);
                                  break;
                    case FTnorm : fwrite (&buf[9],1,&buf[8],term[win].fp;
                                  break;
                    default     : printf ("\nbug");
                                  break;
            }
    else
            switch (buf[4]) {/*if the packet wasn't sent to hub, execute one of the following command*/
                    case DISC   : term[win].cst = term[win].pst;
                                  if ( (term[buf[6]].cst!2) !=BROWSE)
                                                send (buf, buf[6]);
                                  break;
                    case FTre   : resre (buf, win, FTre);
```

```
                        break;
        case CHATre : resre (buf, win, CHATre);
                        break;
        case FTcon  : rescon (buf, win, FTcon);
                        break;
        case CHATcon: rescon (buf, win, CHATcon);
                        break;
        case FTnorm : send ( buf, buf[6]);
                        break;
        case CHAT   : send ( buf, buf[6]);
                        break;
        default     : printf ("\nbug");
                        break;
    }
}
```

บรรณานุกรม

[1] ยืน ภู่วรรณ, "เทคโนโลยี ฮาร์ดแวร์ IBM PC" ,ซีเอ็ด

[2] ธานินทร์ ถาวรศาสนวงศ์ และ ทินกร ดุ๊ก, " การอินเตอร์เฟส IBM PC", ฟิสิค เซ็นเตอร์

[3] ANDREW SCHULMAN, RAYMOND J.MICHEALS, JIM KYLE, TIM PATERSON, DAVID MAXEY, AND RALF BROW "UNDOCUMENTED DOS", Addison Wesley, 1990 .

[4] JOE CAMPELL "C Programmer's Guide to Serial Communications" ,Howard W. Sams and Company a division of Macmillan,Inc.