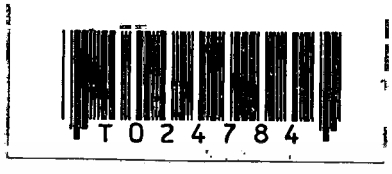
โครงข่ายคอมพิวเตอร์แบบท้องถิ่นราคาถูกโดยใช้เทคนิคเนกาทีฟพัลส์

LOW COST LAN BY NEGATIVE PULSE TECHNIQUE



นายวีระ ธรรมจรัส

MR. WERA THAMJARAT

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชาวิศวกรรมไฟฟ้า

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2538

ISBN 974-621-455-1

LOW COST LAN BY NEGATIVE PULSE TECHNIQUE

WERA THAMJARAT

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE DEGREE

MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING

GRADUATE SCHOOL

KING MONGKUT 'S INSTITUTE OF TECHNOLOGY LADKRABANG

1995

ISBN 974-621-455-1

หัวข้อวิทยานิพนธ์	โครงข่ายคอมพิวเตอร์แบบท้องถิ่นราคาถูกโดยใช้เทคนิคเนกาทีฟพัลส์
นักศึกษา	นายวีระ ธรรมจรัส
อาจารย์ผู้ควบคุมวิทยานิพนธ์	ผศ.พลผดุง ผดุงกุล
ระดับการศึกษา	วิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมไฟฟ้า
ภาควิชา	อิเล็กทรอนิกส์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหาร ลาดกระบัง
พ.ศ.	2538

บทคัดย่อ

การสื่อสารข้อมูลดิจิตอลแบบอนุกรมโดยเทคนิคเนกาทีฟพัลส์ ทำให้รับส่งข้อมูลได้ระยะทางไกล เทคนิคนี้ทำให้ข้อมูลมีความผิดพลาดลดลง เนื่องจากมีสัญญาณนาฬิกาส่งไปพร้อมกับสัญญาณข้อมูลในทุกๆ บิทโดยใช้เทคนิคเนกาทีฟพัลส์ ขยายสัญญาณนาฬิกาให้อยู่ในช่วง 0 ถึง +12 โวลต์ พร้อมกับแปลงสัญญาณข้อมูลให้อยู่ในช่วง 0 ถึง -5 โวลต์ ผสมสัญญาณทั้งสองส่งไปในสายส่งแบบคู่บิดเกลียว (Twisted Pair) หรือใช้สายโทรศัพท์ทั่วไปซึ่งมีราคาถูกกว่าสายส่งสัญญาณชนิดอื่น เมื่อนำมาประยุกต์ใช้กับการสื่อสารข้อมูลดิจิตอลแบบอนุกรม เพื่อสร้างระบบโครงข่ายคอมพิวเตอร์แบบท้องถิ่น (Local Area Network : LAN) จึงได้กำหนดโทโปโลจีแบบบัสมีโหลดเทอร์มินอล 100 โอมห์ มีสถานีได้สูงสุด 256 สถานีและเพื่อให้ประสิทธิภาพในการสื่อสารเพิ่มขึ้น เราจึงใช้อุปกรณ์วงจรรวม INTEL 82510 Asynchronous Serial Controller แทนอุปกรณ์วงจรรวม INTEL 8251 ทำให้รับส่งข้อมูลดิจิตอลแบบอนุกรมอะซิงโครนัสมีอัตราส่งข้อมูลสูงถึง 288000 บิทต่อวินาที โปรโตคอลที่ใช้เป็นแบบที่คอยการเรียกจากศูนย์กลาง(Centralized Polling) โดยมีสถานีบริการแฟ้มข้อมูล(File Server Station) เป็นศูนย์กลางทำหน้าที่วนถามความต้องการของแต่ละสถานีผู้ใช้ (User Station Scanning) เมื่อพบสถานีผู้ใช้ที่ต้องการติดต่อก็จะมีรูปแบบของการติดต่อ เช่น การส่งแฟ้มข้อมูลไปยังสถานีอื่น การขอแฟ้มข้อมูลจากสถานีอื่น การส่งข้อความให้ปรากฏบนจอภาพของสถานีอื่น การควบคุมแป้นพิมพ์ของสถานีอื่นและนำภาพบนจอภาพของสถานีอื่นมาปรากฏบนจอภาพของสถานีผู้ควบคุม และการสั่งพิมพ์บนเครื่องพิมพ์ที่ต่อกับสถานีอื่นเป็นต้น เพื่อที่จะให้สถานีอื่นใช้สายส่งเสมือนว่าใช้สายส่งอยู่ตลอดเวลา จึงแบ่งเวลาในการติดต่อ(Time Sharing) โดยควบคุมจากสถานีบริการแฟ้มข้อมูล การทดลองในส่วนโปรแกรมโครงข่ายเป็นไปตามโปรโตคอลที่กำหนดไว้ทุกประการ ส่วนวงจรเนกาทีฟพัลส์และวงจรอินเตอร์เฟสได้ทดลองรับส่งข้อมูล สามารถส่งข้อมูลบนสายส่งความยาว 1 กิโลเมตร ด้วยอัตราส่งข้อมูล 288000 บิทต่อวินาที เมื่อทำการทดลองเปรียบเทียบกับอุปกรณ์รับส่งที่เป็นมาตรฐาน RS-485 ได้ผลการทดลองเป็นที่น่าพอใจ

Thesis Title	Low Cost LAN by Negative Pulse Technique
Student	Mr. Wera Thamjarat
Thesis Advisor	Assistant Professor Polphaduang Phadungkul
Level of Study	Master of Engineering in Electrical Engineering
Department	Electronics King Mongkut's Institute of Technology Ladkrabang
Year	1995

ABSTRACT

Digital data communication by Negative Pulse Technique is a data communication in serial format, by amplifying voltage and current of clock signal in positive voltage and combine it with the digital data which is inverted in negative voltage. The combined signal is sent by a twisted pair cable. This transmission data is served for a long distance. Every bit of transmission data signal is synchronoused with the clock signal, it can reduce a signal error. These good specifications are suited for using this technique in LAN. For more communication efficiency, we use the INTEL 82510 Asynchronous Serial Controller to process the specific protocol controller. This LAN is bus topology, the terminal loads 100 ohm, max number of station are 256 stations. We choose the Centralized Polling Protocols by setting the File Server Station scans the command from the User Station(such as : SEND FILE, RECEIVE FILE, MAIL, REMOTE and PRINT). The File Server Station uses the Time Sharing to give the User Station a change to communicate. The Time Sharing makes the user feels continuously in the communication. In the experiment, the interface circuits and software parts follow a specified protocal. In a Negative Pulse circuit can transfer data in 1 kilometres of the telephone wires and the baudrate is 288000. We test this Negative Pulse Technique and compare it with the RS-485 standard.

กิตติกรรมประกาศ

ขอขอบคุณ ผู้ช่วยศาสตราจารย์ พลผดุง ผดุงกุล เป็นอย่างยิ่งที่ท่านเป็นผู้วางหัวข้อวิทยานิพนธ์ ให้คำปรึกษา ชี้นำ ในการศึกษา ค้นคว้า ทดลอง จนวิทยานิพนธ์ฉบับนี้สำเร็จ

ขอขอบคุณ รองศาสตราจารย์ ดร. รัตติกร วรากูลศิริพันธุ์ ผู้ให้ความรู้พื้นฐานที่สำคัญในระบบโครงข่ายคอมพิวเตอร์ จนสามารถนำมาประยุกต์ใช้งานได้

ขอขอบคุณ อาจารย์ ทรงชัย วีระทวีมาศ ที่ให้ความรู้ในการเขียนและวิธีแก้ไขโปรแกรม M-WARE

ขอขอบคุณ คุณผานิตย์ และ คุณกิติ ธรรมจรัส ที่ให้กำลังใจและสนับสนุนการศึกษามาโดยตลอด

ขอขอบคุณ คุณขวัญจิต ธรรมจรัส ที่ช่วยจัดพิมพ์วิทยานิพนธ์ฉบับนี้จนเสร็จสมบูรณ์

นายวีระ ธรรมจรัส

สารบัญ

หน้า

สารบัญตาราง

หน้า

ตารางที่

สารบัญภาพ

รูปที่ หน้า

รูปที่ หน้า

บทที่ 1
บทนำ

1.1 ความจำเป็นในการสื่อสารข้อมูลดิจิตอล

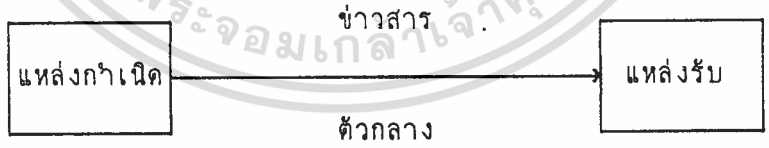
ปัจจุบันการสื่อสารระหว่างมนุษย์ด้วยกันหรือระหว่างเครื่องคอมพิวเตอร์ด้วยกันมีบทบาทมากขึ้น และมีแนวโน้มที่จะเป็นสังคมข่าวสาร(Information Society)อย่างชัดเจนเป็นเหตุให้เกิดบริการทางการสื่อสารมากชนิดกว่าแต่ก่อน อย่างเช่น

- การให้บริการทางโทรศัพท์ระบบดิจิตอล
- การบริการลักษณะและรูปแบบของข่าวสารผ่านโครงข่ายโทรศัพท์
- การให้บริการสื่อสารแบบเคลื่อนที่
- การให้บริการด้านข่าวสารข้อมูลทางคอมพิวเตอร์

ความหมายเดิม คือ การติดต่อระหว่างมนุษย์ด้วยกัน การอ่านหนังสือ การสนทนาทางโทรศัพท์ หรือการดูภาพต่างๆ จุดประสงค์หลักเพื่อให้เกิดการรับและส่งข่าวสารระหว่างกัน เมื่อพิจารณาถึงคำว่า"ระบบสื่อสาร"(Communication System)อย่างแรกต้องมีข่าวสาร(Message)ที่จะส่ง ถ้าไม่มีส่วนนี้ก็ถือว่าไม่เกิดการสื่อสารขึ้นอย่างเช่น ต้องการส่งข้อมูลพยากรณ์อากาศ เป็นต้น

อย่างที่สองต้องมีแหล่งกำเนิดและแหล่งรับ โดยแหล่งกำเนิดทำหน้าที่สร้างข่าวสาร ในขณะที่มีแหล่งรับทำการรับข่าวสารนั้นผ่านทางสื่อกลาง ซึ่งเป็นตัวนำพาข่าวสารมา อย่างเช่นการกระจายเสียงวิทยุให้ประชาชนโดยสถานีวิทยุโดยมีคลื่นความถี่ในอากาศเป็นสื่อกลาง

เพื่อให้เกิดความง่ายในการพิจารณา เราแทนความหมายของระบบสื่อสารด้วยแผนภาพได้ดังรูป 1



รูปที่ 1. แสดงแผนภาพแต่ละส่วนของระบบสื่อสาร

สิ่งหนึ่งที่เราควรคำนึงถึงเมื่อพูดถึงระบบสื่อสาร คือ "ประสิทธิภาพ"(Performance) ซึ่งขึ้นอยู่กับเงื่อนไข 3 ประการ

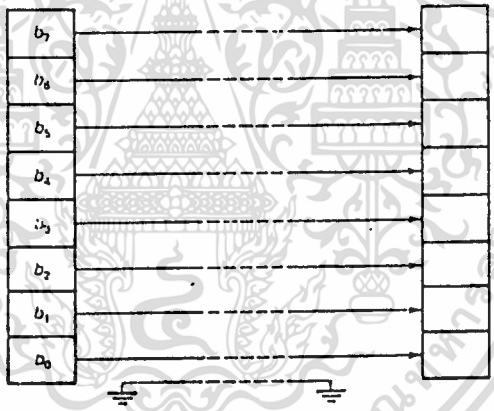
1. เพื่อให้การสื่อสารมีประสิทธิผล ข่าวสารที่ใช้ต้องเป็นที่เข้าใจกัน อย่างเช่นภาษาในการพูดควรเป็นภาษาเดียวกัน

2. รูปแบบของข่าวสารควรเข้ากันได้กับแหล่งกำเนิด, สื่อกลาง และแหล่งรับ อย่างเช่นการสื่อสารระหว่างคอมพิวเตอร์ด้วยกัน ข่าวสารต้องอยู่ในรูปเลขฐานสองซึ่งเป็นรหัสที่วงจรภายในทราบความหมายได้

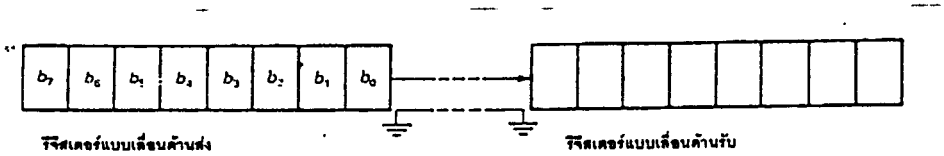
3. ในระบบสื่อสารโอกาสที่จะเกิดการรบกวนจากภายนอกหรือภายในระบบเอง ก็มีได้เช่นกัน จึงต้องมีการป้องกันไว้ส่วนหนึ่งด้วย

1.2 การสื่อสารข้อมูลดิจิตอลทั่วไป(Data Communication)

การสื่อสารข้อมูลเกี่ยวกับการส่งรหัสเลขฐานสอง(Binary code) ซึ่งเป็นรหัสที่สร้างและดำเนินการโดยคอมพิวเตอร์ การติดต่อในการสื่อสารข้อมูลมีลักษณะเชิงดิจิตอลที่สามารถกำหนดสภาวะได้ 2 สถานะ คือค่าตรรกะ(Logic)เท่ากับ 1 หรือ 0 (ส่วนเชิงอนาลอกมีได้ไม่จำกัดสถานะ) การสื่อสารข้อมูลดิจิตอลแบ่งได้เป็นสองแบบคือ การสื่อสารข้อมูลแบบอนุกรมและแบบขนาน (Parallel and Serial Transmission) พิจารณารูปที่ 2



ก. แบบขนาน



ข. แบบอนุกรม

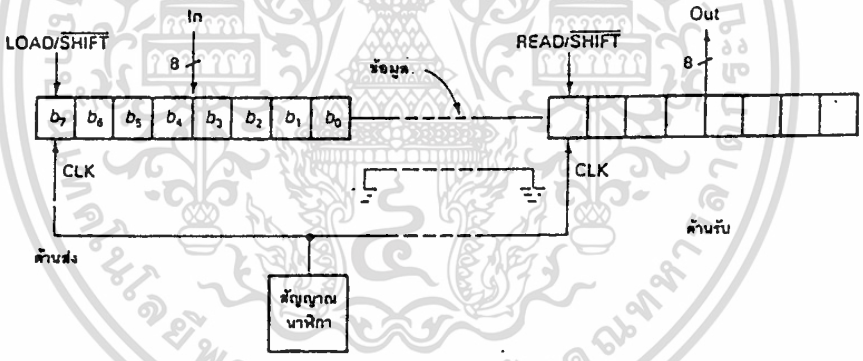
รูปที่ 2. แสดงการสื่อสารข้อมูลแบบขนานและอนุกรม

กรณีแบบขนาน ทุกบิทของข้อมูลถูกส่งแยกไปบนช่องทางติดต่อของแต่ละบิทในเวลาเดียวกัน กรณีแบบอนุกรม แต่ละบิทของข้อมูลถูกส่งไปบนทางติดต่อ 1 ช่องและครั้งละหนึ่งบิท

เปรียบเทียบทั้ง 2 วิธีพบว่าแบบขนานใช้ค่าใช้จ่ายสูงกว่าแบบอนุกรม เนื่องจากกรณีแบบขนานต้องใช้สายนำข้อมูลจำนวนมาก และยังต้องมีความเร็วในการส่งสูงกว่าด้วย เพราะทุกบิทส่งในเวลาเดียวกัน มีผลให้การใช้งานแบบขนานเหมาะสำหรับกรณีด้านรับติดตั้งใกล้ด้านส่ง เช่น ภายในบริเวณห้องเดียวกัน

อย่างเช่นข้อมูลในระบบคอมพิวเตอร์นั้นจะมีความยาว 8 บิทหรือ 16 บิทต่อ 1 คำ จึงต้องมีสายให้พอกับจำนวนบิทที่ต้องส่งออกไปพร้อมกัน เช่นข้อมูล 8 บิทต้องมีสายส่ง 8 เส้น ข้อมูล 16 บิทก็ต้องมีสายส่ง 16 เส้น เป็นต้น

ในการสื่อสารข้อมูลแบบอนุกรมมีการแบ่งรูปแบบในการส่งข้อมูลเป็นสองแบบคือ การส่งข้อมูลแบบสัมพันธ์และแบบไม่สัมพันธ์(Synchronous and Asynchronous Transmission) พิจารณาจากรูปที่ 3

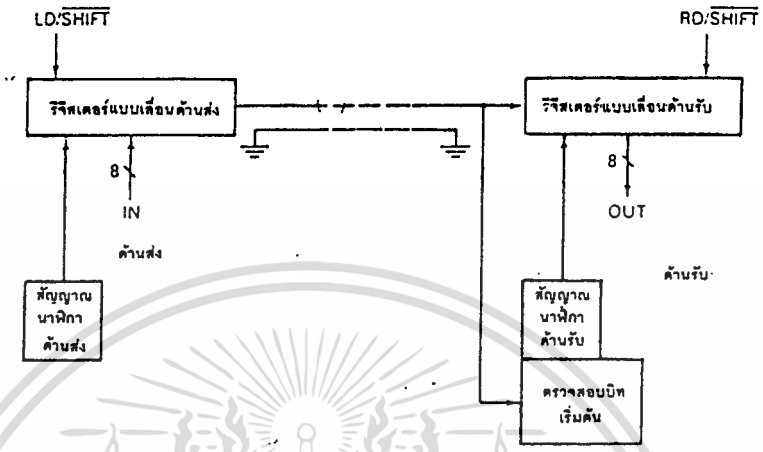


รูปที่ 3. แสดงการส่งข้อมูลแบบสัมพันธ์

การส่งข้อมูลแบบสัมพันธ์(Synchronous Transmission) หมายถึงการที่ด้านรับอ่านข้อมูลในจังหวะเดียวกันกับด้านส่งโดยใช้สัญญาณนาฬิกาเป็นตัวกำหนดจังหวะเดียวกับด้านส่ง สัญญาณนาฬิกาเป็นตัวกำหนดจังหวะการทำงานของรีจีสเตอร์ทั้งสองให้ทำงานสัมพันธ์กัน(วงจรกำเนิดสัญญาณนาฬิกาจะติดตั้งภายในด้านส่ง) นอกจากนี้ เมื่อจังหวะเวลาถูกตั้งให้สัมพันธ์กับด้านรับได้แล้ว ข้อมูลจะถูกส่งไปบนทางติดต่อในแบบบิทต่อบิทต่อเนื่องกันไปอาศัยช่วงเวลาระหว่างบิทต่อบิทมีค่าเท่ากัน โดยไม่ต้องมีบิทเริ่มส่งหรือบิทจบคอยกำกับ ทำให้ส่งข้อมูลได้มากในเวลาเท่ากัน

ข้อเสียของการส่งแบบสัมพันธ์คือ การที่ต้องมีสัญญาณนาฬิกาขนานไปกับข้อมูลทำให้ต้องการทางติดต่อช่องที่สองเพิ่ม โดยเฉพาะกรณีระยะทางไกลๆเป็นการยากมากที่จัดหาทางติดต่อแยกต่างหากสำหรับสัญญาณนาฬิกา นอกจากนี้ทางด้านรับต้องมีวงจรเฟสล็อคลูป(PLL)เพิ่มทำหน้าที่รับข้อมูลจังหวะ

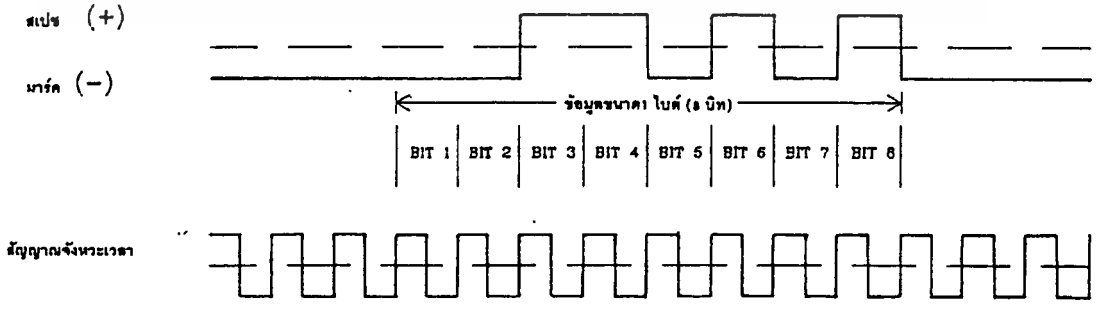
เวลาจากด้านส่งและสร้างสัญญาณนาฬิกาขึ้นใหม่ในด้านรับเพื่อให้สัมพันธ์ขึ้นอาจกล่าวได้ว่าการส่งแบบสัมพันธ์ มีค่าใช้จ่ายสูงกว่าแบบไม่สัมพันธ์



รูปที่ 4. แสดงรูปแบบข้อมูลแบบสัมพันธ์

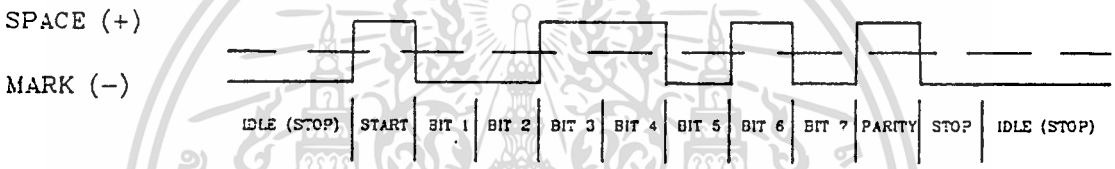
สิ่งที่ควรรู้เพิ่มเติมในการส่งข้อมูลแบบสัมพันธ์มีดังนี้ ข้อมูลในแบบสัมพันธ์นี้จะถูกจัดการให้อยู่ในรูปของชุดข้อมูล(Block of data) ที่มีลักษณะพิเศษคือ ช่วงระยะเวลาระหว่างตัวอักษรด้วยกันจะไม่มี ทำให้การส่งข้อมูลเป็นไปอย่างต่อเนื่องซึ่งตัวอักษรจะแทนด้วยรหัสเลขฐานสองเช่น รหัสแอสคีดังรูปที่ 4

การส่งข้อมูลแบบไม่สัมพันธ์(Asynchronous Transmission)



รูปที่ 5. แสดงการส่งข้อมูลแบบไม่สัมพันธ์

วิธีนี้ไม่จำเป็นต้องมีสัญญาณนาฬิกาสัมพันธ์กับสัญญาณข้อมูลตลอดเวลา โดยจะมีสัมพันธ์ก็ต่อเมื่อมีข้อมูลที่จะรับส่งเท่านั้น จึงต้องมีการใช้บิทเริ่มต้น(Start Bit) เพื่อให้ทางด้านรับตรวจจับการเริ่มส่งของข้อมูลและมีผลให้วงจรกำเนิดสัญญาณนาฬิกาภายในด้านรับทำงาน เพื่อเกิดความสัมพันธ์ในการรับส่งข้อมูล สำหรับความถี่ในการทำงานของวงจรกำเนิดสัญญาณนาฬิกาทั้งด้านส่งและด้านรับมีค่าเท่ากันและขึ้นอยู่กับอัตราบิท(Bit Rate) ที่ใช้ นอกจากนี้มีการเพิ่มบิทจบ(Stop Bit) เพื่อบอกการสิ้นสุดของข้อมูลที่รับส่ง โดยมีขนาด 1-2 บิท เนื่องจากมีการเพิ่มบิทลงในข้อมูลทำให้ความเร็วในการส่งข้อมูลช้ากว่าแบบสัมพันธ์แต่ค่าใช้จ่ายต่ำกว่า



รูปที่ 6. แสดงรูปแบบข้อมูลแบบไม่สัมพันธ์

1.3 แนวความคิดในการสื่อสารข้อมูลดิจิตอลแบบเนกาทีฟพัลส์

เป็นการประยุกต์การสื่อสารข้อมูลแบบไม่สัมพันธ์ ที่นำเอาข้อดีของการสื่อสารแบบสัมพันธ์มาใช้ โดยนำความสัมพันธ์ของสัญญาณนาฬิกามาใช้ในการรับส่งสัญญาณข้อมูล โดยไม่ต้องเพิ่มสายส่งสัญญาณนาฬิกาทำให้การรับข้อมูลมีโอกาสผิดพลาดน้อยลง เมื่อเทียบกับการสื่อสารข้อมูลดิจิตอลแบบไม่สัมพันธ์ และมีการใช้สายส่งที่น้อยกว่าเมื่อเทียบกับการสื่อสารข้อมูลดิจิตอลแบบสัมพันธ์

1.4 การประยุกต์ใช้การสื่อสารข้อมูลแบบเนกาทีฟพัลส์

เริ่มแรกในการออกแบบเทคนิคเนกาทีฟพัลส์ ใช้เพื่อสื่อสารและควบคุมในระบบรักษาความปลอดภัยของพื้นที่บริเวณกว้างเช่น ขอบเขตของโรงงาน โรงแรม โรงเรียน มหาวิทยาลัย หรือสถาบันจึงต้องออกแบบให้รับส่งข้อมูลได้ระยะทางไกล และมีรูปแบบของสัญญาณเฉพาะ ทำให้ไม่สามารถอ่านข้อมูลด้วยวิธีทั่วไปได้เป็นการเพิ่มความปลอดภัยให้ข้อมูล เทคนิคนี้ใช้ควบคุมตามจุดต่างๆ เช่น ประตูรั้ว ประตูอาคาร โดยใช้สายสัญญาณ 2 เส้น ส่งสัญญาณเตือนภัยไปยังเครื่องคอมพิวเตอร์ที่ศูนย์รักษาความปลอดภัย ถึงแม้สายสัญญาณถูกตัดก็จะทราบตำแหน่งของจุดที่ตัดได้

แต่ข้อเสียของวงจรเนกาทีฟพัลส์ คือ อัตราส่งข้อมูลประมาณ 1000 บิทต่อวินาทีซึ่งต่ำมากจึงมีการแก้ไขปรับปรุงวงจรมาจนสามารถรับส่งสัญญาณข้อมูลสูง คือมีอัตราส่งข้อมูล 288000 บิทต่อวินาที ทำให้วงจรเทคนิคเนกาทีฟพัลส์มีประสิทธิภาพสูงขึ้น และที่สำคัญความสัมพันธ์ระหว่างสัญญาณนาฬิกากับสัญญาณข้อมูลทำให้ข้อมูลผิดพลาดลดลง จึงเหมาะที่จะนำมาประยุกต์ใช้ในการสื่อสารข้อมูลระหว่างคอมพิวเตอร์

1.5 แนวคิดในการออกแบบโครงข่ายคอมพิวเตอร์แบบท้องถิ่น

โครงข่ายคอมพิวเตอร์แบบท้องถิ่น หรือที่เรียกสั้นๆ ว่า LAN ย่อมาจาก Local Area Network คือการเชื่อมต่อคอมพิวเตอร์ตั้งแต่ 2 เครื่องขึ้นไป ให้สามารถรับส่งข้อมูลได้ในบริเวณวงกว้างขึ้น เช่นภายในห้อง ระหว่างห้อง ระหว่างชั้น หรือระหว่างอาคาร แต่ต้องอยู่ในระยะไม่เกิน 10 กิโลเมตร หรืออยู่ในบริเวณท้องถิ่นเดียวกัน เราจึงเรียกการเชื่อมต่อโครงข่ายนี้ว่าโครงข่ายคอมพิวเตอร์แบบท้องถิ่น สำหรับโครงข่ายในวิทยานิพนธ์นี้ เราเรียกว่าโครงข่าย M-NET

องค์ประกอบของโครงข่าย M-NET มี 2 ส่วน ส่วนฮาร์ดแวร์ เรียกว่า M-NET และ ส่วนซอฟแวร์ เรียกว่า M-WARE

ข้อกำหนดของโครงข่าย M-NET

1. คอมพิวเตอร์ที่ใช้ต้องเป็น IBM-PC รุ่น XT หรือ AT(หรือเทียบเท่า)
2. ระบบจัดการต้องเป็น DOS
3. สายส่งสามารถใช้กับสายส่งคุณภาพต่ำ หรือที่ใช้กันทั่วไปในการเดินสายโทรศัพท์ภายในบ้าน
4. ควรจะมี HARD DISK จะทำให้การทำงานรวดเร็วขึ้น
5. ความยาวของสายส่งทั้งโครงข่ายไม่เกิน 1 กิโลเมตร
6. อัตราส่งข้อมูลสามารถกำหนดได้ตั้งแต่ 300 ถึง 288000 บิทต่อวินาที

1.6 วัตถุประสงค์และขอบเขตการวิจัย

ต้องการวางระบบโครงข่ายคอมพิวเตอร์แบบท้องถิ่นโดยใช้เทคนิคการรับส่งข้อมูลแบบเนกาทีฟพัลส์ ที่สามารถใช้ฟังก์ชั่นพื้นฐานของโครงข่าย เช่น รับส่งข้อความ รับส่งแฟ้มข้อมูลและใช้ทรัพยากรร่วมกันระหว่างเครื่องคอมพิวเตอร์ที่ติดตั้งอยู่ในโครงข่ายด้วยอัตราส่งข้อมูลสูง 288000 บิทต่อวินาที ระยะทางสูงสุด 1 กิโลเมตร มีจำนวนสถานีสูงสุด 256 สถานี

บทที่ 2
หลักการสื่อสารข้อมูลดิจิตอลเบื้องต้น

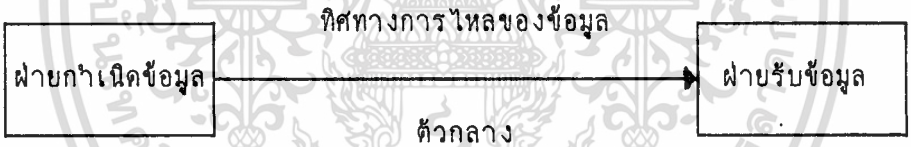
ในบทนี้จะกล่าวถึงรายละเอียดเกี่ยวกับการติดต่อสื่อสารข้อมูลเบื้องต้น และความรู้ที่เป็นพื้นฐานที่เกี่ยวข้องกับระบบโลคัลแอเรียเนตเวอร์ค(LAN) ทั้งในด้านโทโปโลยีในรูปแบบต่างๆ สื่อกลางในการติดต่อ(Media) วิธีการเข้าถึงข้อมูล(Access Method) ตลอดจนรูปแบบของ(Packet) ที่ใช้ในการติดต่อสื่อสารข้อมูล ซึ่งจะนำไปใช้เป็นแนวทางในการออกแบบโครงงานนี้ทั้งส่วนของฮาร์ทแวร์และซอฟแวร์

2.1 การติดต่อสื่อสารข้อมูลเบื้องต้น

ส่วนประกอบเบื้องต้นในการสื่อสารข้อมูลแบ่งได้ออกเป็น 3 ส่วน คือ

- ฝ่ายกำเนิดข้อมูล(Transmitter)
- ตัวกลางในการส่งผ่านข้อมูล(Medium)
- ฝ่ายรับข้อมูล(Receiver)

ซึ่งแต่ละส่วนมีความสัมพันธ์กันดังรูปที่ 7

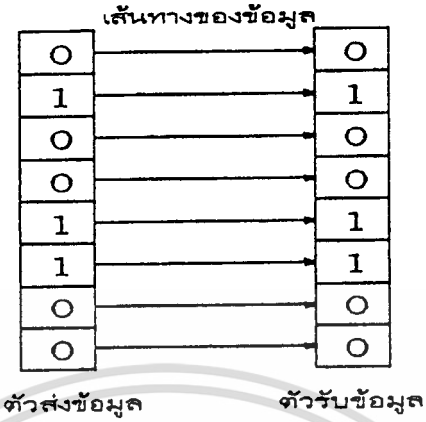


รูปที่ 7. แสดงความสัมพันธ์ของส่วนประกอบหลักในการสื่อสารข้อมูล

วิธีการสื่อสารข้อมูลในการติดต่อสื่อสารข้อมูลของเครื่องคอมพิวเตอร์นั้นแบ่งเป็น 2 ชนิด

2.1.1 การติดต่อข้อมูลแบบขนาน(Parallel)

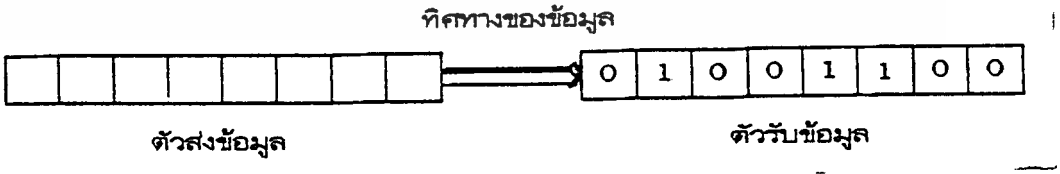
เป็นการติดต่อสื่อสารโดยส่งข้อมูลออกมาครั้งละ 1 ไบต์ คือ ครั้งละ 8 บิท จากอุปกรณ์ส่งไปยังอุปกรณ์รับ ดังนั้นตัวกลางระหว่างเครื่องคอมพิวเตอร์ทั้งสองเครื่องจะต้องมีช่องทางให้ข้อมูลเดินทางอย่างน้อย 8 ช่องทาง โดยมากจะใช้เป็นสายขนานแต่เนื่องจากมีสัญญาณสูญหายไปกับอิมพิแดนซ์ของสาย ซึ่งมีความสัมพันธ์กับระยะทางของสาย ดังนั้นระยะทางระหว่างเครื่องสองเครื่อง ไม่ควรจะเกิน 100 ฟุต ทำให้การส่งแบบนี้ได้ระยะทางไม่ไกลนักแต่สามารถส่งข้อมูลได้รวดเร็วเพราะส่งออกมาพร้อมกันทีละ 8 บิท อุปกรณ์ที่ติดต่อแบบขนานกับคอมพิวเตอร์ เช่น เครื่องพิมพ์ เป็นต้น



รูปที่ 8. การส่งข้อมูลแบบขนาน

2.1.2 การติดต่อข้อมูลแบบอนุกรม(Serial)

ในการติดต่อแบบนี้ ข้อมูลจะถูกส่งออกมาทีละบิทระหว่าง จุดส่งและจุดรับ ซึ่งการส่งข้อมูลแบบนี้จะช้ากว่าการส่งแบบขนาน แต่ใช้ตัวกลางในการสื่อสารเพียงช่องเดียวหรือสายเพียงคู่เดียว ทำให้ค่าใช้จ่ายสำหรับสื่อกลางถูกกว่าแบบขนาน การส่งแบบนี้ใช้สำหรับส่งในระยะทางไกลมากกว่า 100 ฟุต ข้อมูลจากจุดส่งจะต้องถูกเปลี่ยนให้เป็นอนุกรมก่อนแล้วค่อยทยอยส่งออกไปทีละบิทไปยังจุดรับ ที่จุดรับจะต้องมีรูปแบบในการเปลี่ยนข้อมูลที่ถูกส่งมาให้เป็นแบบขนาน ในการแปลงต้องมีรูปแบบที่เหมาะสมเพื่อป้องกันการผิดพลาดของข้อมูล



รูปที่ 9. การส่งข้อมูลแบบอนุกรม

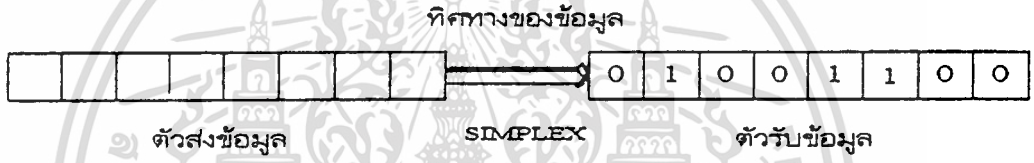
2.2 ทิศทางการรับส่งข้อมูล

แบ่งได้ออกเป็น 3 แบบ คือ

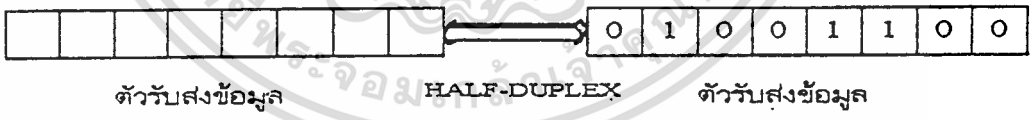
ก. การส่งแบบทิศทางเดียว(Simplex) การส่งแบบนี้ทิศทางการส่งจะคงที่โดยกำหนดในครั้งแรกว่าฝ่ายใดเป็นฝ่ายส่ง ฝ่ายใดเป็นฝ่ายรับ การส่งแบบนี้แม้ไม่คล่องตัวแต่เหมาะสมในการใช้งานบางประเภท

ข. การส่งแบบฮาล์ฟดูเพล็กซ์(Half Duplex) เป็นการส่งข้อมูลในทิศทางใดทิศทางหนึ่งในเวลาใดเวลาหนึ่ง คือ ทั้ง 2 สถานีสามารถผลัดกันส่งได้แต่จะส่งพร้อมๆกันไม่ได้

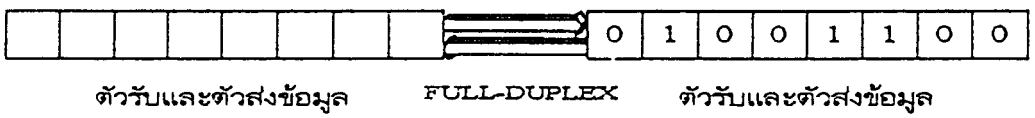
ค. การส่งแบบฟูลดูเพล็กซ์(Full Duplex) เป็นการส่งข้อมูลในทิศทางใดๆก็ได้ในเวลาหนึ่งๆ คือทั้ง 2 สถานีสามารถรับแล้วส่งข้อมูลได้ในเวลาเดียวกัน



การส่งแบบซิมเพล็กซ์ The Simplex Connection



การส่งแบบฮาล์ฟดูเพล็กซ์ The Half-Duplex Connection



การส่งแบบฟลูดูเพล็กซ์ The Full-Duplex Connection

2.3 ชนิดของสายส่งสัญญาณข้อมูล(Transmission Line)

ในระบบLAN มีตัวกลางหลายอย่างที่ใช้ เช่น ทองแดง หรือใยแก้ว ตัวอย่างตัวกลางทองแดงเช่น สายคู่บิดเกลียว(Twisted pair) หรือ สายโคแอกเชียล(Coaxial cable) เป็นต้น ตัวอย่างตัวกลางใยแก้วเช่น สายใยแก้วนำแสง(Optical fiber) เป็นต้น

ลักษณะของสายส่งชนิดต่าง ๆ

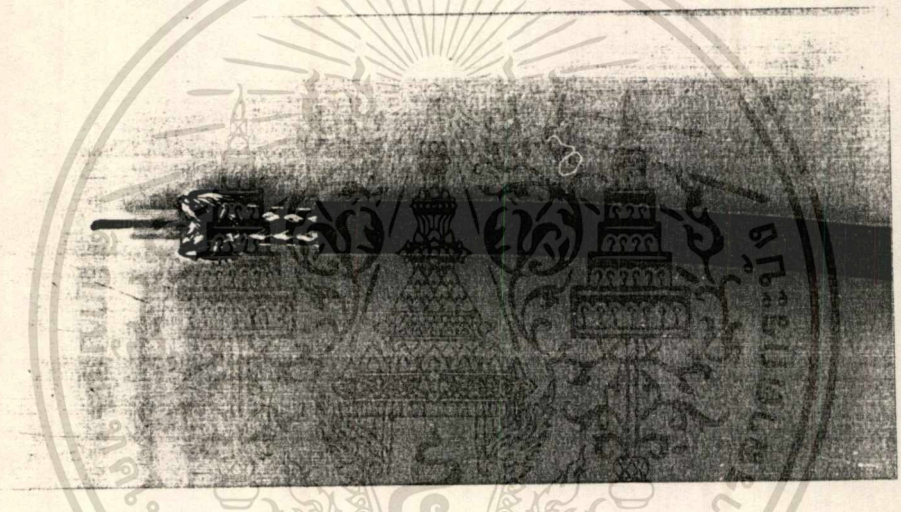
ก. สายส่งแบบคู่บิดเกลียว(Twisted Pair Cable)

ส่วนใหญ่จะใช้ในการต่อสายโทรศัพท์ภายใน ซึ่งถือเป็นสายแบบ Voice Grade Medium (VGM) จะประกอบด้วยสายสองสายพันไขว้กันเป็นไป และแต่ละสายควรมีอิมพิแดนซ์เท่ากัน ในระบบ LAN จะใช้สายที่มีคุณภาพดีกว่าเรียกว่า Data Grade Medium(DGM) ข้อดีของสายแบบนี้คือ ติดตั้งง่าย ราคาถูก แต่จะมีค่าการลดทอนสัญญาณสูง จำเป็นต้องใช้ Repeater ต่อเป็นระยะๆ เมื่อส่งในระยะทางไกลๆ มี Bandwidth ต่ำและไม่เหมาะที่จะใช้ในสถานที่ที่มีสัญญาณรบกวนสูง เช่นในโรงงาน อย่างไรก็ตามสายชนิดนี้ก็ยังเหมาะที่จะใช้ในการส่งด้วยความเร็วต่ำและระยะทางสั้นเนื่องจากมีราคาถูกกว่าแบบอื่นมาก

รูปที่ 10. สายส่งแบบคู่บิดเกลียว

ข. สายโคแอกเชียล (Coaxial Cable)

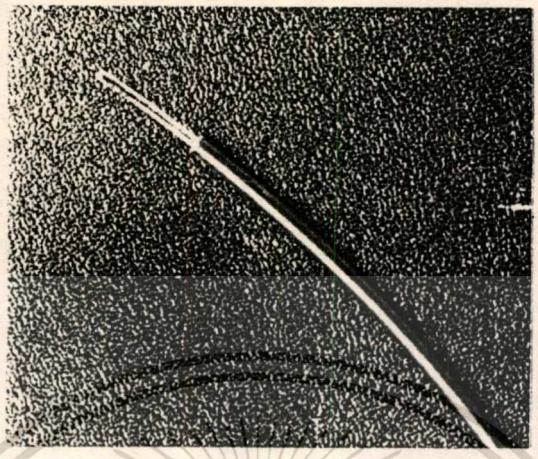
สายชนิดนี้จะประกอบด้วยสายเส้นในเส้นหนึ่ง แล้วล้อมรอบด้วยแผ่นอลูมินั่มบางๆ หรือสายถักโดยจะถูกแยกจากกันด้วยฉนวน สายเส้นในจะเป็นตัวนำสัญญาณ ส่วนสายรอบๆจะเป็นสายกราวนด์ ส่วนใหญ่นิยมใช้เป็น Community Antenna Television Cable(CATV) สายชนิดนี้มีคุณสมบัติในการส่งข้อมูลดีกว่าสาย Twisted Pair และยังสามารถใช้กับการส่งแบบ Boardband Transmission ได้ด้วย เนื่องจากมี Bandwidth ถึง 300 เมกกะเฮริตซ์ นอกจากนี้ยังสามารถป้องกันสัญญาณรบกวน(Noise) ได้ดีกว่าสายคู่บิดเกลียว



รูปที่ 11. สายส่งแบบโคแอกเชียล

ค. สายใยแก้วนำแสง(Optical Fiber)

จะเป็นแท่งแก้วหรือสารที่คล้ายแก้วบางๆ ซึ่งทำหน้าที่เป็นตัวนำแสงจากตัวต้นกำเนิดไปยังตัวรับโดยต้นกำเนิดแสงจะเป็น LED หรือ Laser Diode(LD) ข้อมูลที่จะส่งจะถูกมอดดูเลทแบบความสี่(FM) กับแสง ในทางด้านรับจะเป็น Pin Field Effect Transistor (Pin FET) แล้วทำการดีมอดดูเลทกลับมาข้อมูล ปัญหาที่สำคัญในการใช้ Optical Fiber คือการติดตั้งซึ่งสายแบบนี้มีความเปราะบางมากและในการต่อเชื่อมสายทำได้ยากต้องใช้กรรมวิธีพิเศษอย่างไรก็ตาม Optical Fiber นี้จะเป็นเทคโนโลยีที่สำคัญต่อไป เนื่องจากสามารถส่งข้อมูลด้วยความเร็วสูง มีค่าการลดทอนต่ำ มีความต้านทานต่อสัญญาณรบกวนดีมาก และวัตถุดิบที่ใช้ทำสายคือ ซิลิกา(Silica)นั้นหาได้ทั่วไปทำให้ในอนาคตจะมีราคาถูกลง

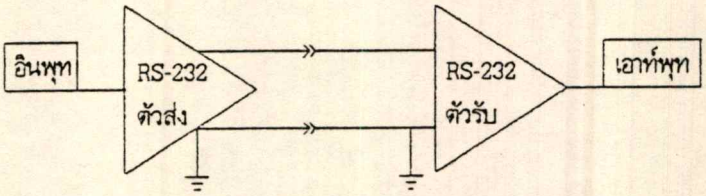


รูปที่ 12. สายส่งแบบใยแก้วนำแสง

2.4 มาตรฐานในการสื่อสารข้อมูลแบบอนุกรม

2.4.1 การสื่อสารข้อมูลแบบอนุกรมตามมาตรฐาน RS-232C

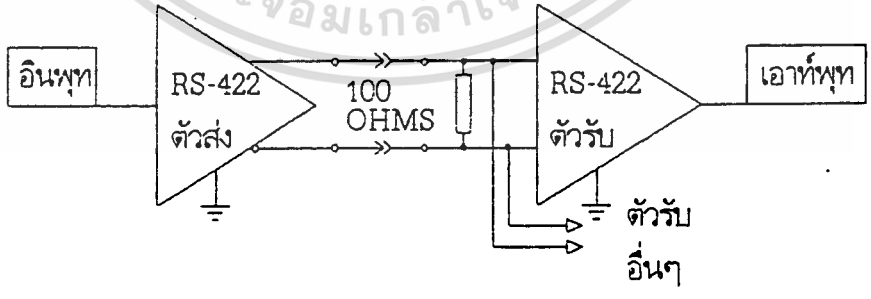
การสื่อสารข้อมูลแบบอนุกรมที่ใช้งานอยู่ในปัจจุบันนั้น ได้มีการกำหนดมาตรฐานการรับส่งข้อมูลไว้หลายแบบด้วยกัน แต่ที่ได้รับความนิยมนำมาใช้งานอย่างมาก คือ การสื่อสารข้อมูลแบบอนุกรมตามมาตรฐาน RS-232C และที่มาตรฐานนี้เป็นที่นิยมเนื่องจากเป็นระบบการสื่อสารข้อมูลที่ใช้ในเครื่องไมโครคอมพิวเตอร์ IBM PC ซึ่งเป็นคอมพิวเตอร์ที่มีใช้อย่างแพร่หลายมากจากอดีตจนถึงปัจจุบัน มาตรฐานการสื่อสารนี้ในการออกแบบเบื้องต้นได้ออกแบบ สำหรับการเชื่อมต่อกับเครื่องโมเด็ม(MODEM) ซึ่งเป็นอุปกรณ์ที่ใช้การสื่อสารข้อมูลระหว่างคอมพิวเตอร์ผ่านทางสายโทรศัพท์ ซึ่งทำให้อัตราการรับส่งข้อมูลถูกจำกัดให้มีค่าที่ค่อนข้างต่ำ มาตรฐาน RS-232C นี้ได้ออกแบบให้มีโครงสร้างการสื่อสารเป็นแบบจุดต่อจุดเท่านั้น โดยมีลักษณะสมบัติทางไฟฟ้าและทางกายภาพ ดังแสดงในตารางที่ 1 และรูปที่ 13



รูปที่ 13. แสดงโครงสร้างของการสื่อสารข้อมูลแบบอนุกรมตามมาตรฐาน RS-232C

2.4.2 การสื่อสารข้อมูลแบบอนุกรมมาตรฐาน RS-422A

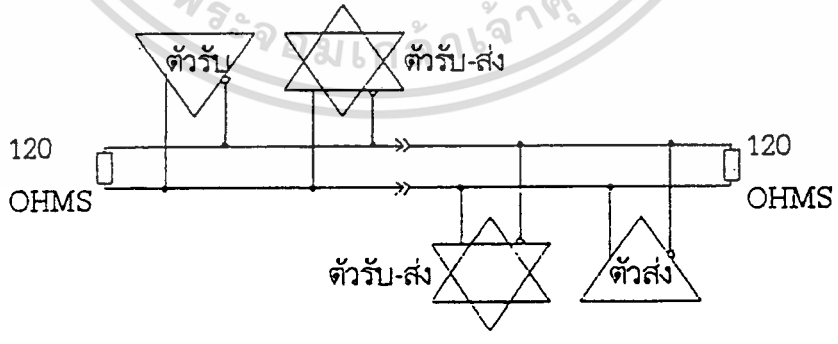
ในการออกแบบระบบสื่อสารข้อมูลจากที่ผ่านมาจนถึงปัจจุบัน ได้มีการพยายามที่จะออกแบบให้การสื่อสารข้อมูลได้รวดเร็วขึ้นและมีระยะในการสื่อสารที่มากขึ้นด้วย ซึ่งที่ผ่านมาการสื่อสารข้อมูลตามมาตรฐาน RS-232C ได้ออกแบบเพื่อใช้เชื่อมต่อกับโมเด็มเท่านั้น จึงไม่ได้คำนึงถึงความเร็วและระยะทางในการสื่อสาร ต่อมาได้มีมาตรฐานในการสื่อสารข้อมูลใหม่ที่ได้ออกแบบมาเพื่อรองรับความต้องการของผู้ใช้งานที่ต้องการให้การรับส่งข้อมูลได้ระยะทางไกลและรวดเร็ว มาตรฐานนี้คือ RS-422A ซึ่งการที่มาตรฐานสื่อสารนี้สามารถรับส่งข้อมูลได้ไกลและรวดเร็วขึ้นเนื่องมาจากหลักการที่ใช้สัญญาณเป็นแบบดิฟเฟอเรนเชียลดังแสดงในรูปที่ 14 ซึ่งหลักการก็คือสัญญาณที่รับส่งจะเป็นการเปรียบเทียบระหว่างสัญญาณ 2 เส้นเทียบกับมาตรฐาน RS-232C ที่สัญญาณทุกสัญญาณจะเทียบกับกราวด์ ซึ่งในการสื่อสารในระยะทางไกลๆแล้วสัญญาณจะถูกลดทอนไปและเมื่อสัญญาณถูกลดทอนถึงจุดๆหนึ่งสัญญาณนั้นก็จะผิดพลาดไปจากความเป็นจริง ทำให้การรับส่งข้อมูลเกิดความผิดพลาดขึ้น แต่สำหรับสัญญาณแบบดิฟเฟอเรนเชียล การลดทอนของสัญญาณก็จะไปลดทอนทั้งสองสายด้วยค่าที่เท่ากันหรือใกล้เคียงกันและความแตกต่างของสัญญาณระยะสัญญาณทั้ง 2 เส้น จากตัวส่งไปยังตัวรับก็ยังมีค่าเท่าเดิมหรือเปลี่ยนแปลงน้อย จึงทำให้ผลของการลดทอนต่อสัญญาณที่ระยะการสื่อสารที่ไกลมีผลต่อสัญญาณดิฟเฟอเรนเชียลมีค่าน้อยกว่า การสื่อสารข้อมูลแบบนี้จึงสามารถส่งข้อมูลได้ไกลกว่าและอัตราการสื่อสารข้อมูลสูงกว่าพร้อมทั้งสามารถติดต่อตัวรับได้ถึง 10 ตัว ดังแสดงตารางเปรียบเทียบมาตรฐานการสื่อสารข้อมูลในตารางที่ 1



รูปที่ 14. แสดงโครงสร้างของการสื่อสารข้อมูลแบบอนุกรมตามมาตรฐาน RS-422A

2.4.3 การสื่อสารข้อมูลแบบอนุกรมตามมาตรฐาน RS-485

การสื่อสารข้อมูลตามมาตรฐานที่กล่าวมาข้างต้นคือ RS-232C นั้นเป็นมาตรฐานการสื่อสารข้อมูลในแบบที่ใช้สื่อสารระหว่างอุปกรณ์ต่ออุปกรณ์ หรือ จุดต่อจุด(Point-to-Point) ส่วน RS-422A นั้นเป็นมาตรฐานที่พัฒนามาจาก RS-232C ให้ได้ระยะทางไกลขึ้นและอัตราการสื่อสารเพิ่มขึ้น แต่ก็ยังเป็นการสื่อสารข้อมูลจากอุปกรณ์หนึ่งตัวไปยังอุปกรณ์อื่นๆ ได้สูงสุด 10 ตัวเท่านั้นไม่สามารถส่งย้อนกลับจากอุปกรณ์ตัวรับมาตัวส่งได้ หรือกล่าวได้ว่าการสื่อสารข้อมูลตามมาตรฐาน RS-422A นั้นเป็นการสื่อสารข้อมูลแบบ Simplex คือทิศทางของข้อมูลเป็นแบบทางเดียวตลอดเวลา ดังนั้นถ้าต้องการออกแบบระบบให้เป็นลักษณะโครงข่ายข้อมูลก็จะไม่สามารถทำได้ จึงมีการพัฒนามาตรฐานการสื่อสารข้อมูลขึ้นใหม่เพื่อรองรับความต้องการนี้ คือ มาตรฐาน RS-485 ซึ่งเป็นมาตรฐานที่อาศัยหลักการของสัญญาณแบบดิฟเฟอเรนเชียลเช่นเดียวกับมาตรฐาน RS-422A แต่สามารถสื่อสารข้อมูลได้ทั้ง 2 ทิศทางในสายสัญญาณเพียงคู่เดียว ซึ่งก็คือการสื่อสารข้อมูลแบบ Half-Duplex จากผลของการใช้สัญญาณในลักษณะดิฟเฟอเรนเชียลนี้ ทำให้ระยะทางและความเร็วในการสื่อสารข้อมูลมีค่าสูง เช่นเดียวกับมาตรฐานการสื่อสารข้อมูล RS-422A แต่มาตรฐาน RS-485 สามารถที่จะสื่อสารระหว่างอุปกรณ์ทั้งการรับและการส่งของอุปกรณ์ได้สูงสุด 32 ตัว หรืออาจกล่าวได้ว่าการสื่อสารตามมาตรฐาน RS-485 เป็นการสื่อสารแบบหลายจุด(Multipoint Communication) ดังแสดงค่าเปรียบเทียบในตารางที่ 1 และแสดงโครงสร้างในรูปที่ 15



รูปที่ 15. แสดงโครงสร้างของการสื่อสารข้อมูลแบบอนุกรมตามมาตรฐาน RS-485

ตารางที่ 1 แสดงค่าเปรียบเทียบคุณสมบัติทางไฟฟ้าของ RS-232C, RS-422A และ RS-485

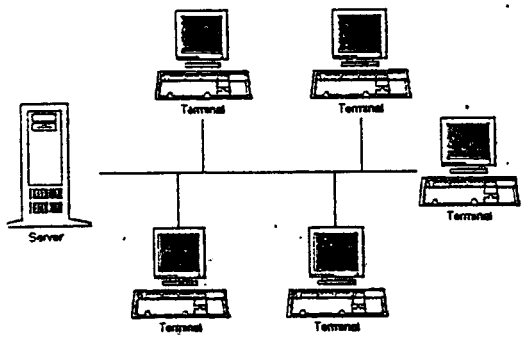
พารามิเตอร์	RS-232-C	RS-422-A	RS-485
โหมดการทำงาน จำนวนของตัวรับและส่ง ที่ยอมรับได้	Single-ended 1 ตัวส่ง 1 ตัวรับ	Diferrential 1 ตัวส่ง 10 ตัวรับ	Diferrential 32 ตัวส่ง 32 ตัวรับ
ความยาวของคู่สายสูงสุด	50 ฟุต	4000 ฟุต	4000 ฟุต
อัตราการส่งข้อมูลสูงสุด	20k bps	10 M bps	10 M bps
maximum common mode voltage	2.5 V	6 V -2.5 V	12 V -7 V
Driver output	5V ต่ำสุด 15V สูงสุด	2V ต่ำสุด	1.5 V ต่ำสุด
Driver load (ohm)	3K ถึง7K	100 ต่ำสุด	60 ต่ำสุด
Driver slew rate	30V/usสูงสุด	NA	NA
กระแสลิมิต เมื่อเอาท์พุทลัดวงจร	500 mA ลัดวงจร กับ VCC หรือ GND	150 mA ลัดวงจรกับ GND	150 mA ลัดกับ GND 250 mA ลัดกับ 12V
ค่าความต้านทานเอาท์พุท ของตัวส่ง (โอหม์)	NA (power on) 300 (power off)	NA (power on) 60k (power off)	120k (power on) 120k (power off)
ค่าความต้านทานอินพุทตัวรับ	3k ถึง 7k ohm	4k ohm	12k ohm
ความไวของตัวรับ	3V	200 mV	200 mV

2.5 หลักการและโครงสร้างการสื่อสารข้อมูลเป็นโครงข่าย(Network Topology)

ในการสื่อสารข้อมูลที่ต้องการให้อุปกรณ์หลายๆตัวสามารถติดต่อและสื่อสารกันได้จะเป็นลักษณะโครงสร้างที่มีชื่อเรียกเฉพาะว่าโครงข่าย(Network) โดยอุปกรณ์ที่ต่ออยู่ภายในโครงข่ายนั้นจะมีชื่อเรียกว่า(Node) และเนื่องจากการที่มีโหนดหลายตัวอยู่ในโครงข่าย จึงต้องมีการจัดวางตำแหน่งโหนดเพื่อให้เหมาะกับการนำไปใช้งานตามความต้องการของระบบซึ่งก็คือ การจัดโครงสร้างของโครงข่าย(Network Configuration) หรือ โทโปโลยี(Topology) โดยโทโปโลยีหรือโครงสร้างของโครงข่ายนั้นสามารถจัดได้หลายแบบ ซึ่งมีข้อดีและข้อเสียที่ต่างกันดังจะได้กล่าวต่อไป

2.5.1 การสื่อสารข้อมูลเป็นโครงข่ายแบบบัส

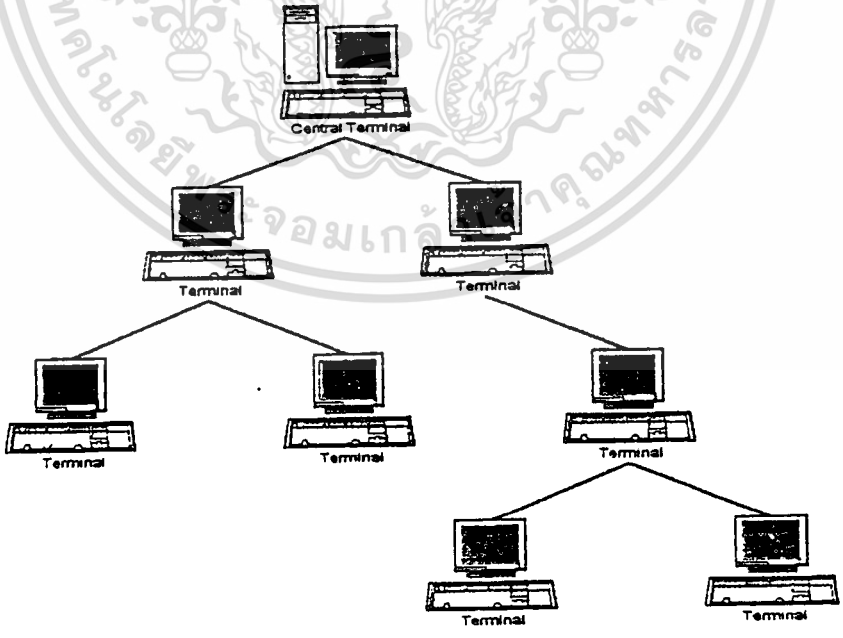
โครงสร้างการโครงข่ายในการสื่อสารข้อมูลแบบบัส(Bus Network) นี้เป็นโครงสร้างของโครงข่ายข้อมูลที่ได้รับความนิยมอย่างมากในปัจจุบัน โดยเฉพาะอย่างยิ่งกับโครงข่ายข้อมูลแบบท้องถิ่น เนื่องจากความสะดวกและง่ายในการควบคุมการทำงานของระบบเพราะจากโครงสร้างของโครงข่ายข้อมูลแบบนี้ จะใช้สายสัญญาณเพียงชุดเดียวในการรองรับการสื่อสารข้อมูลของระบบดังในรูปที่ 16 ซึ่งจากคุณสมบัติที่กล่าวนี้ยังทำให้การส่งข้อมูลจากศูนย์กลางข้อมูลสามารถที่จะกระจายไปยังอุปกรณ์ที่ต่ออยู่กับโครงข่ายได้พร้อมกันด้วยการส่งเพียงครั้งเดียว ทำให้สิ้นเปลืองสายสัญญาณน้อยกว่า และสามารถสื่อสารข้อมูลใน 2 ทิศทางแบบสลับเวลา(Half-Duplex) และในกรณีที่ต้องการให้การสื่อสารข้อมูลรวดเร็วขึ้นยังสามารถจะขยาย ระบบให้การสื่อสารข้อมูลพร้อมกันได้ใน 2 ทิศทางแบบตลอดเวลา(Full-Duplex) โดยการเพิ่มสายหรือช่องสัญญาณที่ขนานไปกับสายหรือช่องสัญญาณเดิมอีก 1 ชุด จากที่กล่าวจะแสดงให้เห็นถึงข้อดีของโครงสร้างของโครงข่ายแบบนี้แต่โครงข่ายแบบนี้ก็มีข้อเสียที่สำคัญคือ ในกรณีระบบเกิดทำงานผิดปกติการตรวจหาจุดที่เกิดปัญหานั้นทำได้ยากและอาจทำให้โครงข่ายทั้งระบบทำงานไม่ได้ เพราะสายหรือช่องสัญญาณในการสื่อสารข้อมูลมีเพียงชุดเดียว ซึ่งในการพัฒนาต่อมาได้ทำให้โหนดที่เกิดทำงานผิดปกติสามารถตัดออกจากโครงข่ายได้โดยไม่รบกวนต่อระบบ



รูปที่ 16. แสดงโครงสร้างของโครงข่ายแบบบัส

2.5.2 การสื่อสารข้อมูลเป็นโครงข่ายแบบทรี

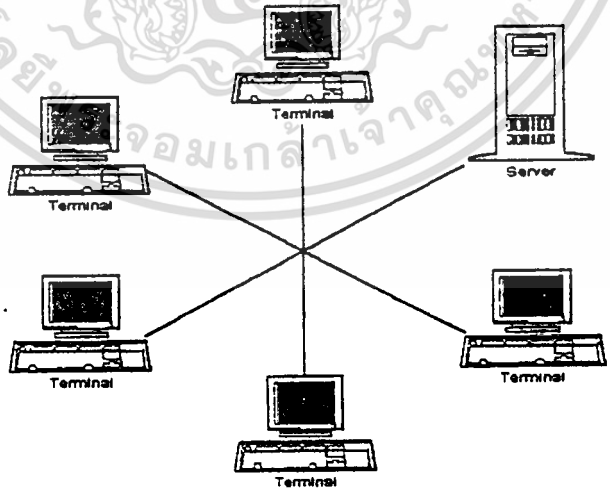
โครงข่ายการสื่อสารข้อมูลแบบทรี(Tree Network) เป็นโครงสร้างของโครงข่ายข้อมูลที่สามารถควบคุมระบบได้ง่าย โดยโครงสร้างของโครงข่ายข้อมูลแบบนี้มีการกระจายศูนย์กลางควบคุมออกไปอย่างเป็นลำดับชั้นคล้ายกับการแผ่กิ่งก้านของต้นไม้ ดังในรูปที่ 17 ซึ่งที่ทุกจุดแยกจะมีศูนย์กลางย่อยเพื่อทำหน้าที่ในการควบคุมการทำงานของโหนดย่อยที่แยกออกไปจากจุดนี้ ทำให้การควบคุมการหาจุดผิดปกติและการแก้ไขสามารถทำได้สะดวกรวดเร็วกว่า และยังเป็นการลดความหนาแน่นของงานที่ศูนย์กลางหลักของระบบ อย่างไรก็ตามโครงสร้างของโครงข่ายแบบนี้ก็ยังมีข้อเสีย ที่มีโอกาสที่ข้อมูลจะกระจุกเป็นคอขวดอยู่ที่ตำแหน่งศูนย์กลางได้และถ้าโหนดที่มีลำดับชั้นที่สูงกว่าเกิดมีปัญหาขึ้นมา จะมีผลต่อโหนดที่ลำดับชั้นต่ำกว่าด้วยและจะมีผลกระทบมากยิ่งขึ้นถ้าโหนดที่ทำงานผิดปกติ ผลจากการเกิดเหตุการณ์นี้อาจมีผลให้ข้อมูลเกิดความสูญหายหรือผิดพลาด และเป็นการลดความเชื่อถือได้ของระบบอีกด้วย ยิ่งไปกว่านั้นในการที่จะสื่อสารกันระหว่างโหนดที่มีลำดับชั้นต่างกันจะทำได้ยาก เพราะจะต้องผ่านข้อมูลหลายขั้นตอนตามลำดับชั้นของโครงข่าย แต่อย่างไรก็ตามโครงสร้างการสื่อสารแบบนี้ก็ยังได้รับความนิยมนำไปใช้งาน เนื่องจากการที่สามารถเพิ่มจำนวนโหนดทำได้ง่าย



รูปที่ 17. แสดงโครงสร้างของโครงข่ายแบบทรี

2.5.3 การสื่อสารข้อมูล เป็นโครงข่ายแบบสตาร์

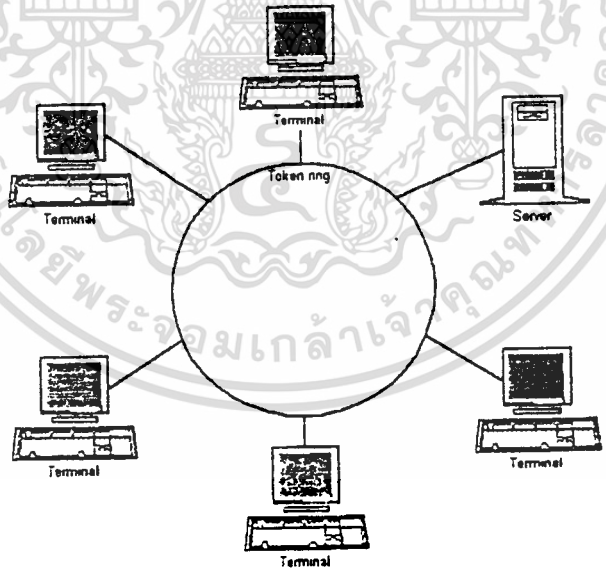
โครงสร้างของโครงข่ายการสื่อสารข้อมูลแบบสตาร์(Star Network) เป็นโครงสร้างที่ได้รับความนิยมนำมาใช้อย่างมากทั้งในอดีตจนปัจจุบัน เนื่องจากความง่ายและความสะดวกในการควบคุม รวมทั้งโปรแกรมควบคุมก็ไม่ซับซ้อน โดยโครงสร้างของโครงข่ายการสื่อสารข้อมูลแบบนี้จะต้องเป็นการสื่อสารกันผ่านศูนย์กลางหลักของระบบเท่านั้น และสายสัญญาณของแต่ละโหนดจะกระจายออกไปจากศูนย์กลางข้อมูลเพียงจุดเดียว ดังในรูปที่ 18 ซึ่งจากโครงข่ายแบบที่กล่าวมาจะคล้ายกับโครงสร้างของโครงข่ายแบบทรี แต่โครงข่ายแบบสตาร์จะมีข้อจำกัดในแง่ของจำนวนโหนดที่มาต่อเข้ากับระบบ และยังมีข้อเสียที่คล้ายกับโครงข่ายแบบทรีคือ มีโอกาสที่จะเกิดการกระจุกของข้อมูลรวมทั้งการเสียหายหรือการทำงานผิดปกติของศูนย์กลางของระบบ จะมีผลให้โครงข่ายทำงานไม่ได้หรือทำให้เกิดความผิดพลาดของข้อมูลขึ้นได้ จึงเป็นการลดความน่าเชื่อถือของข้อมูลภายในระบบซึ่งในปัจจุบันได้มีการพยายามเพิ่มความน่าเชื่อถือ โดยเพิ่มศูนย์กลางสำรองที่จะทำงานแทนศูนย์กลางข้อมูลหลักในกรณีที่ไม่สามารถทำงานได้ อย่างไรก็ตามโครงข่ายแบบนี้ก็มีข้อดีที่สามารถแยกโหนดที่ทำงานผิดปกติได้ง่ายกว่า และเส้นทางในการสื่อสารของแต่ละโหนดจะแยกจากกันเป็นอิสระ ทำให้โหนดแต่ละตัวภายในโครงข่ายแบบนี้สามารถสื่อสารข้อมูลได้แม้ว่าจะมีอัตราการสื่อสารข้อมูล มาตรฐานการสื่อสารข้อมูล และโครงสร้างของข้อมูลที่ต่างกัน



รูปที่ 18. แสดงโครงสร้างของโครงข่ายแบบสตาร์

2.5.4 การสื่อสารข้อมูลเป็นโครงข่ายแบบริง

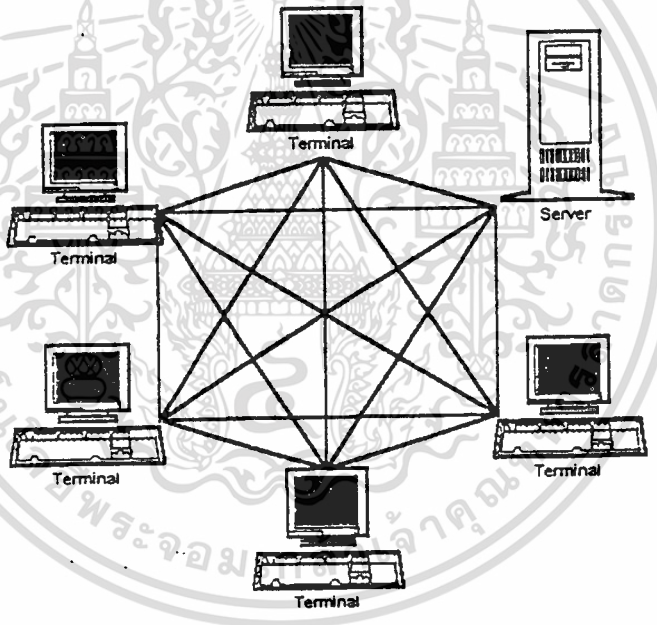
โครงสร้างของโครงข่ายข้อมูลแบบริง(Ring Network) นี้ก็เป็นโครงข่ายอีกแบบหนึ่งที่ได้รับความนิยมนำไปใช้งาน โดยสาเหตุที่เรียกโครงข่ายแบบนี้ว่าเป็นริง เนื่องจากโครงสร้างทางกายภาพของระบบที่การไหลเวียนของข้อมูล จะวนอยู่ในลักษณะเป็นวงกลมและข้อมูลจะไหลไปในทิศทางเดียว ดังแสดงในรูปที่ 19 โดยการทำงานเริ่มจากเมื่อข้อมูลเข้ามาที่โหนดก็จะทำการตรวจสอบว่าเป็นข้อมูลของโหนดที่กำลังติดต่อสื่อสารกันอยู่หรือไม่ ถ้าใช่ก็จะรับข้อมูลนี้เข้าไป แต่ถ้าไม่ใช่ก็จะทำการส่งข้อมูลไปยังโหนดถัดไป และทำในลักษณะเดียวกันต่อไปภายในโครงข่ายโดย ข้อดีของโครงข่ายแบบนี้ คือจะไม่เกิดการกระจุกกันของข้อมูล และมีโครงสร้างที่ง่ายต่อการควบคุมเพราะโหนดทุกโหนดจะทำงานในลักษณะเหมือนกัน แต่ก็มีข้อเสียที่สำคัญคือถ้าสายสัญญาณภายในโครงข่ายเกิดเสียหรือขาดก็จะทำให้โครงข่ายของระบบหยุดทำงาน และมีโอกาสที่ข้อมูลซึ่งไม่ต้องการหรือไม่ใช้งานแล้วยังคงวิ่งวนอยู่ภายในโครงข่าย ทำให้ความสามารถในการสื่อสารข้อมูลของระบบลดลง



รูปที่ 19. แสดงโครงสร้างของโครงข่ายแบบริง

2.5.5 การสื่อสารข้อมูลเป็นโครงข่ายแบบเมช

โครงสร้างของโครงข่ายข้อมูลเมช(Mesh Network) เป็นโครงข่ายที่จะให้โหนดแต่ละตัวสามารถจะสื่อสารข้อมูลไปยังโหนดอื่นทุกตัวได้โดยไม่ผ่านโหนดอื่นๆ ดังในรูปที่ 20 โดยโครงข่ายแบบนี้มีข้อดีที่สำคัญ คือสามารถแก้ปัญหาการกระจุกของข้อมูลและหัญหาการที่อุปกรณ์และช่องสัญญาณเกิดเสียหรือผิดปกติ อย่างไรก็ตามโครงข่ายแบบนี้ก็มีข้อเสียที่การวางโครงข่ายจะมีความสลับซับซ้อน การควบคุมระบบทำได้ยากมีช่องสัญญาณที่ไม่ได้ใช้งาน(ใช้งานช่องสัญญาณไม่คุ้มค่า) และความสิ้นเปลืองสายสัญญาณเป็นจำนวนมากถ้ามีจำนวนโหนดมากขึ้นจึงทำให้ไม่ค่อยได้รับความนิยมนำไปใช้งาน แต่ถึงกระนั้นก็ตามถ้าผู้ใช้งานโครงข่ายที่ต้องการความเชื่อถือได้ของข้อมูลสูงแล้วก็ควรจะเลือกใช้โครงข่ายแบบเมช



รูปที่ 20. แสดงโครงสร้างของโครงข่ายแบบเมช

ในวิทยานิพนธ์ได้ใช้โครงสร้างของโครงข่ายข้อมูลแบบบัสในการสื่อสารข้อมูล เนื่องจากโครงสร้างพื้นฐานของวิทยาพนธ์นั้นจะมีศูนย์กลางข้อมูลเพียงจุดเดียวจึงไม่เหมาะที่จะใช้โครงข่ายแบบทรีและแบบเมชสำหรับโครงข่ายแบบริงนั้นจะมีขั้นตอนและวิธีการควบคุมที่ยุ่งยากและสลับซับซ้อนมาก และสำหรับโครงข่ายแบบสตาร์จะมีข้อจำกัดที่จำนวนของโหนดที่จะมาต่อ, สายสัญญาณที่ใช้ก็จะต้องใช้มากกว่า(เทียบกันเมื่อจำนวนโหนดเท่ากัน), มีการกระจุกของข้อมูลที่ศูนย์กลางและในการสื่อสาร

แต่ละครั้งจะสามารถติดต่อกับโหนดได้เพียงครั้งละโหนดเท่านั้น และเมื่อนำมาเทียบกับโครงข่ายแบบบัสซึ่งจะใช้สายสัญญาณน้อย, การควบคุมระบบทำได้ง่ายกว่า, สามารถสื่อสารข้อมูลกับโหนดได้หลายตัวพร้อมกันด้วยส่งสัญญาณเพียงครั้งเดียวโดยใช้สายสัญญาณเพียงชุดเดียว และเป็นโครงข่ายที่เหมาะสำหรับการที่จะมีศูนย์กลางข้อมูลเพียงจุดเดียว

2.6 โปรโตคอล

องค์ประกอบสำคัญสำหรับการเป็นโครงข่ายข้อมูล นอกจากหลักการและโครงข่ายที่ได้กล่าวถึงแล้ว ยังมีองค์ประกอบที่สำคัญและขาดไม่ได้สำหรับการเป็นโครงข่าย คือ กฎหรือข้อกำหนดในการข้อมูล หรือที่นิยมเรียกว่าโปรโตคอล(Protocols) ซึ่งเป็นส่วนที่จะกำหนดมาตรฐานในการควบคุมและจัดการระบบการสื่อสารข้อมูลในส่วนต่างๆ ซึ่งสามารถแบ่งโปรโตรคอลได้เป็นหลายลำดับชั้น (layer) ซึ่งมาตรฐานการแบ่งระดับนั้นได้มีองค์กรต่างๆ แบ่งไว้หลายแบบ แต่ที่ได้รับความนิยมมากที่สุด คือการแบ่งลำดับขั้นตามมาตรฐาน OSI(Open System Interconnection) ซึ่งจะแบ่งออกเป็น 7 ลำดับชั้นดังนี้

1. Physical layer
2. Data-link layer
3. Network layer
4. Transport layer
5. Session layer
6. Presentation layer
7. Application layer

โดยในลำดับชั้นที่ 1-3 จะเป็นการควบคุมและจัดการการสื่อสารข้อมูลทางกายภาพ ส่วนในลำดับชั้น 5-7 เป็นลำดับชั้นที่ควบคุมและจัดการในส่วนที่ติดต่อกับผู้ใช้งาน และสำหรับลำดับชั้นที่ 4 เป็นส่วนที่เชื่อมโยงระหว่าง 3 ลำดับชั้นล่าง(ชั้น 1-3) กับ 3 ลำดับชั้นบน(ชั้น 5-7)

สำหรับรายละเอียดที่กล่าวในที่นี้ จะเกี่ยวข้องเฉพาะในส่วนของโปรโตคอลการควบคุมการเชื่อมโยงข้อมูล(Data Link Control Protocols หรือ DLCP) ซึ่งจัดการในส่วนของขั้นตอนและหลักการต่างๆคือโครงสร้างและรายละเอียดของข้อมูล วิธีในการสื่อสารข้อมูลการตรวจสอบแก้ไขความผิดพลาดของข้อมูล และขนวนการในการควบคุมการติดต่อสื่อสาร โดย DLCP แบ่งได้ตามโครงสร้างของข้อมูล 2 แบบ คือ Byte-Oriented Protocols และ Bit-Oriented Protocols

2.6.1 ไบท์โอเรียนท์โปรโตคอล(Byte-oriented protocols)

โปรโตคอลแบบนี้เป็นโปรโตคอลที่การสื่อสารข้อมูล และการควบคุมการทำงานจะทำโดยใช้ลักษณะข้อมูลที่เป็นตัว(character) หรือ ไบท์(byte)

ก. อะซิงโครนัสโปรโตคอล(Asynchronous protocols)

โปรโตคอลในการสื่อสารข้อมูลนี้จะใช้การสื่อสารข้อมูลแบบ Half-Duplex ที่มีลักษณะการสื่อสารข้อมูลแบบอะซิงโครนัส ซึ่งเป็นการสื่อสารข้อมูลแบบพื้นฐานที่ใช้งานมาเป็นเวลานานแล้ว จึงมีรายละเอียดและขั้นตอนในการสื่อสารข้อมูลที่ทำให้มีโอกาสเกิดความผิดพลาดได้น้อย ยังมีข้อดีที่การสื่อสารข้อมูลแบบนี้มีโครงสร้างการทำงานที่ง่าย อุปกรณ์ที่ใช้ในการสื่อสารข้อมูลก็ไม่สลับซับซ้อนและมีราคาถูก โปรโตคอลแบบนี้จึงเหมาะสำหรับใช้ในระบบขนาดเล็ก

ข. ไบนารี่ซิงโครนัสโปรโตคอล(Binary synchronous protocols)

โปรโตคอลแบบนี้จะมีลักษณะการทำงานที่ใช้งานข้อมูลเป็นในลักษณะไบท์ และยังคงใช้การสื่อสารข้อมูลแบบ Half-Duplex ซึ่งจะคล้ายกับอะซิงโครนัสโปรโตคอล แต่ที่ต่างกันคือจะใช้การสื่อสารข้อมูลแบบซิงโครนัส มีรายละเอียยดและขั้นตอนในการสื่อสารข้อมูลที่ให้ความน่าเชื่อถือมากกว่า อีกทั้งยังสามารถใช้อัตราเร็วในการสื่อสารข้อมูลที่สูงกว่าโดยตัวอย่างของการสื่อสารข้อมูลแบบนี้ที่ได้กำหนดเป็นมาตรฐานแล้วคือ การสื่อสารข้อมูลตามมาตรฐาน BSC (Binary Synchronous Communications) ซึ่งเป็นโปรโตคอลที่มีลักษณะของข้อมูลแบบไบท์ที่ได้รับความนิยมนำไปใช้งาน

2.6.2 บิทโอเรียนท์โปโตคอล(Bit-oriented protocols)

โปรโตคอลแบบนี้เป็นโปรโตคอลที่การสื่อสารข้อมูล และการควบคุมการทำงานจะทำโดยใช้ลักษณะข้อมูลที่เป็นบิท(bit) โดยมีตัวอย่างของการสื่อสารข้อมูลในลักษณะนี้ที่มีการกำหนดขึ้นเป็นมาตรฐานแล้ว คือ HDLC(High-level Data Link Control) โดยมีโครงสร้างของข้อมูลแบบซิงโครนัสเช่นเดียวกับ BSC แต่ต่างกันที่มีลักษณะของข้อมูลเป็นแบบบิท ซึ่งโปรโตคอลแบบนี้ข้อดีที่สามารถสื่อสารข้อมูลแบบ full-duplex ได้ทำให้การสื่อสารข้อมูลได้รวดเร็วกว่า แต่โปรโตคอลแบบนี้ก็มีรายละเอียดและโครงสร้างในการสื่อสารข้อมูลที่สลับซับซ้อนมากทำให้การควบคุมการทำงานทำได้ยากและต้องใช้อุปกรณ์ที่มีราคาสูง จึงไม่เหมาะที่จะนำไปใช้งานกับระบบขนาดเล็ก

2.6.3 แพ็กเก็ท(Packet of Information)

ในระบบโครงข่ายต่าง ๆ ยกเว้นแบบสตาร์ ตัวกลางในการส่งจะมีการแบ่งกันใช้งานระหว่างแต่ละโหนด ซึ่งในการที่มีการใช้งานตัวกลางร่วมกันนี้ทำให้มีปัญหาเกิดขึ้นก็คือจะต้องมีการควบคุมการใช้ตัวกลางไม่ให้การส่งข้อมูลของโหนดต่างๆมีการรบกวนกันและอีกสิ่งหนึ่งก็คือ จะต้องมีการสร้างข้อกำหนดต่างๆเช่น วิธีการส่งและวิธีการรับข้อมูล

มีวิธีการหลายอย่างที่จะให้โหนดต่างๆติดต่อกันได้ในโครงข่าย วิธีที่ง่ายก็คือ มีการสร้างเส้นทางที่แน่นอนให้แก่ด้านส่งและด้านรับ โดยวิธีที่เรียกว่า Circuit Switching ซึ่งใช้กันมากในระบบโทรศัพท์ แต่ในการส่งข้อมูลของคอมพิวเตอร์นั้นแตกต่างจากสัญญาณเสียง คอมพิวเตอร์จะติดต่อกันด้วยความเร็วสูงและเป็นเพียงชั่วขณะหนึ่งเท่านั้น ดังนั้นเวลาที่เหลืออยู่จะเป็นการสูญเสียไปโดยเปล่าประโยชน์จึงมีเทคนิคที่จะเพิ่มประสิทธิภาพในการใช้ตัวกลาง เทคนิคนี้เรียกว่า Packet Switching เทคนิคนี้จะไม่มีการกำหนดเส้นทางที่แน่นอนจากด้านส่งไปยังด้านรับไว้ก่อน แต่ข้อมูลจะถูกส่งโดยแบ่งเป็นส่วนย่อย ๆ (Packet) แล้วส่งเข้าไปในโครงข่าย วิธีการนี้ด้านส่งและด้านรับจะใช้ตัวกลางเฉพาะขณะส่งแพ็กเก็ทเท่านั้น ส่วนเวลาที่เหลือจะสามารถให้โหนดอื่นๆใช้ได้อีก

รูปแบบของแพ็กเก็ทในระบบโครงข่ายจะประกอบด้วยส่วนต่าง ๆ ดังนี้

ก. HEADER

จะประกอบด้วย

- Preamble or start of packet indicator เป็นส่วนเริ่มแรกของ Packet และในบางระบบอาจใช้ในการซิงค์กับสัญญาณนาฬิกาของตัวส่งและตัวรับด้วย

- Addressing information node ต่างๆ ในโครงข่ายจะมีแอดเดรส(Address) ที่แน่นอนอยู่ ข้อมูลในส่วนนี้จะประกอบด้วยแอดเดรส ของตัวส่งและตัวรับทำให้โหนดต่าง ๆ รู้ได้ว่าแพ็กเก็ทนี้ต้องการส่ง ให้โหนดใดและส่งมาจากโหนดใด เพื่อจะได้รับข้อมูลและตอบรับ หรือส่งต่อข้อมูลไปยังโหนดที่ถูกต้องได้

- Control information ส่วนนี้เป็นข้อมูลที่บอกถึงวัตถุประสงค์ของแพ็กเก็ทนั้นว่าใช้ทำอะไรเช่นเพื่อการจัดการระบบเพื่อดู Status ของ node หรืออื่น ๆ

นอกจากส่วนต่าง ๆ เหล่านี้แล้ว ในส่วน Header อาจจะมีส่วนที่เป็น Sequencial Number เป็นส่วนที่บอกให้ทราบถึงลำดับของแพ็กเก็ทในกรณีที่ข้อมูลมีความยาวหลายแพ็กเก็ท

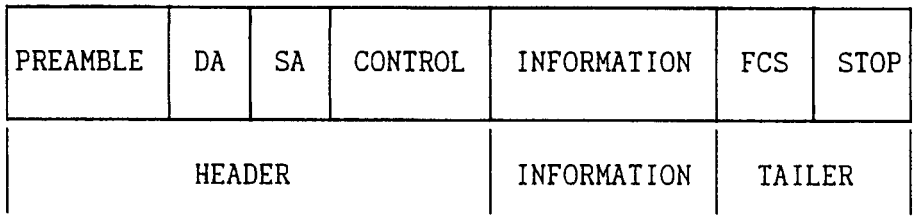
ข. INFORMATION

- Data field เป็นส่วนของข้อมูลจริงที่ต้องการจะส่ง

ค. TAILER

- Frame Check Sequence(FCS) เป็นส่วนที่ใช้ตรวจสอบความถูกต้องของข้อมูล ซึ่งอาจเป็น parity bit, Check sum หรือ CRC เป็นต้น

- End of Packet Indicator เป็นส่วนที่บอกให้ทราบว่าสิ้นสุดของข้อมูลแล้ว



รูปที่ 21 แสดงโครงสร้างของ Packet

2.6.4 **วิธีเข้าออกระบบโครงข่าย(Network Access Method)**

ในระบบสื่อสารข้อมูลทั่วไปจะใช้ตัวกลางร่วมกัน โดยใช้วิธีการของ Time Division Multiplexer ซึ่งข้อเสียก็คือเราจะไม่สามารถรู้ได้ว่าเวลาที่จัดให้แต่ละnodeนั้น ตัวมันมีข้อมูลที่ต้องการจะส่งหรือไม่ ซึ่งวิธีการที่ดีขึ้นก็คือการจัดให้แต่ละโหนดได้ใช้ตัวกลางในขณะที่ตัวมันมีข้อมูลจะส่งเท่านั้น และสิ่งสำคัญก็คือแต่ละโหนดจะต้องมีความเท่าเทียมกันในการที่จะรอใช้ตัวกลางนั้น

ในระบบโครงข่ายก็มีปัญหาเช่นเดียวกัน วิธีการที่ใช้กันอยู่ในการควบคุมการใช้ตัวกลางแบ่งได้ 2 ประเภทคือ

ก. **วิธีเข้าออกระบบโครงข่ายแบบไม่รอจังหวะ(Contention access)**

คือ แต่ละโหนดจะรอโอกาสที่จะส่งข้อมูลเมื่อโครงข่ายนั้นว่างอยู่ แล้วจึงทำการส่งข้อมูลออกไป วิธีนี้จะเกิดปัญหาการชน(Collision) คือ มีโหนดมากกว่าหนึ่งโหนดส่งข้อมูลออกมาพร้อมๆ กันทำให้ข้อมูลชนกันและถูกทำลายไป ซึ่งวิธีนี้อาจจะเรียกว่า Random Access การเข้าถึงข้อมูลแบบนี้แบ่งออกได้เป็น 3 แบบคือ

1. ALOHA มีทั้งแบบ Pure ALOHA และ Slotted ALOHA
2. CSMA(Carrier Sense Multipie Access)
3. CSMA/CD(Carrier Sense Multipie Access / Collision Detect)

ข. **วิธีเข้าออกระบบโครงข่ายแบบรอจังหวะ(Noncontention access)**

บางทีอาจเรียกว่า Controlled Access หรือ Polling วิธีนี้แต่ละโหนดในโครงข่ายสามารถส่งข้อมูลออกไปได้ก็ต่อเมื่อได้รับอนุญาตเท่านั้น ซึ่งวิธีนี้จะไม่มีการเกิดการชนกันของสัญญาณ สามารถแบ่งได้ออกเป็น 2 แบบ คือ

1. Centralized Polling
2. Distributing Polling ซึ่งยังแบ่งได้เป็น 2 แบบ คือ
   - Empty Slot มีทั้งแบบ Single Slot และ Multi-Slot
   - Token Passing มีทั้งแบบ Token Bus และ Token Ring

2.7 โครงข่ายคอมพิวเตอร์แบบท้องถิ่น(LOCAL AREA NETWORK)

โครงข่ายคอมพิวเตอร์ หมายถึงการเชื่อมต่อกันของกลุ่มคอมพิวเตอร์ซึ่งแต่ละเครื่องสามารถที่จะแลกเปลี่ยนข่าวสารกันได้ โดยแต่ละเครื่องมีการทำงานเป็นอิสระต่อกัน โครงข่ายการเชื่อมต่อแบ่งออกเป็น 3 แบบ ตามขนาดของพื้นที่ที่มีโครงข่ายครอบคลุมถึงคือ

2.7.1 **โครงข่ายคอมพิวเตอร์ระหว่างประเทศ : Wide Area Network(WAN)** เป็นโครงข่ายที่ครอบคลุมพื้นที่ขนาดใหญ่มาก อาจถึงขนาดหลาย ๆ ทวีป

2.7.2 **โครงข่ายคอมพิวเตอร์แบบท้องถิ่น : Local Area Network(LAN)** เป็นโครงข่ายที่ครอบคลุมพื้นที่เล็ก ๆ เช่น ภายในตึก หรือหลายๆ ตึก เช่น ในมหาวิทยาลัย

2.7.3 **โครงข่ายคอมพิวเตอร์ระหว่างระหว่างเมือง : Metropolitan Area Network(MAN)** มีขนาดระหว่าง WAN กับ LAN เช่น ขนาดเมืองเล็กๆ เมืองหนึ่ง

คุณสมบัติทั่วไปของการสื่อสารแบบ LAN คือ

- อัตราการส่งข้อมูลได้ 0.1 ถึง 100 Mbps
- ระยะทางการส่ง 0.1 ถึง 25 กิโลเมตร
- มีอัตราการผิดพลาดของข้อมูลต่ำ(10^{-6} ถึง 10^{-11})

การแบ่งชนิดของ LAN จะต้องพิจารณาถึงสิ่งต่อไปนี้

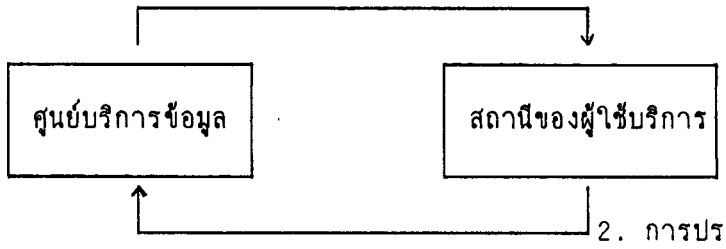
1. ตัวกลางที่ใช้ในการเชื่อม Node ต่าง ๆ
2. รูปแบบของการเชื่อมต่อ หรือ Topology
3. Algorithm ที่ใช้ควบคุมการเข้าถึงตัวกลาง

2.8 ความสามารถของระบบเครือข่ายท้องถิ่น

ระบบเครือข่ายท้องถิ่นมีข้อดีเหนือมินิคอมพิวเตอร์หรือเมนเฟรม กล่าวคือมีการประมวลผลแบบกระจายงาน(Distributed Processing) ความรวดเร็วในการติดต่อสื่อสาร การติดต่อระหว่างสถานีผู้ใช้งาน(Interconnected Stations) ใช้โปรแกรมและข้อมูลร่วมกันได้ใช้ฮาร์ดแวร์และทรัพยากรที่มีอยู่ร่วมกันได้

2.8.1 การประมวลผลแบบกระจายงาน

ในการประมวลผลแบบกระจายเมื่อสถานีของผู้ใช้บริการขอโปรแกรม และข้อมูลจากศูนย์บริการข้อมูล(file server) โปรแกรมคำสั่งจะถูกสำเนาจากศูนย์บริการข้อมูลไปยังหน่วยความจำที่สถานีของผู้ใช้ เมื่อเสร็จสิ้นการใช้งานโปรแกรมและไฟล์ข้อมูลแล้ว สถานีผู้ใช้จะส่งข้อมูลกลับไปยังศูนย์บริการข้อมูลเพื่อใช้งานต่อไป



รูปที่ 22 ลักษณะการประมวลผลที่กระจายงานให้แต่ละสถานีของผู้ใช้บริการ

การประมวลผลแบบกระจายงานจะช่วยให้สถานีหลายๆ สถานีใช้งานร่วมกันในระบบได้โดยไม่ไปลดความสามารถในการประมวลผลข้อมูลของแต่ละสถานี ในเครื่องคอมพิวเตอร์แบบเมนเฟรมความสามารถในการประมวลผล จะถูกแบ่งออกไปในแต่ละสถานี ถ้ายิ่งมีสถานีมากก็จะทำให้ประสิทธิภาพการทำงานลดลง ในการกระจายการประมวลผลข้อมูลจะทำให้เพิ่มความเร็วและประสิทธิภาพของระบบได้

2.8.2 ความเร็วในการติดต่อสื่อสาร

ระบบเครือข่ายท้องถิ่นมีการใช้สื่อการส่งข้อมูลของตนเอง (โดยปกติจะเป็นสายส่งข้อมูล) มากกว่าที่จะใช้สื่อสาธารณะ เช่น สายโทรศัพท์ทำให้การติดต่อสื่อสารในระบบเครือข่ายมีความเร็วสูงมากกว่า 1 ล้านบิทต่อวินาทีขึ้นไปในสายส่งข้อมูลความเร็วสูง นอกจากนี้ระบบเครือข่ายท้องถิ่นยังมีช่องทางที่เรียกว่าประตูสื่อสาร(Gateways) เชื่อมต่อกับเครือข่ายแบบ WAN ในการติดต่อสื่อสารอื่นๆอีกด้วย การสื่อสารที่มีความรวดเร็วช่วยให้ได้ข้อมูลทันต่อเหตุการณ์ ลดความซ้ำซ้อนของข้อมูลและเพิ่มความถูกต้องแม่นยำของข้อมูล

2.8.3 การติดต่อระหว่างสถานีผู้ใช้งาน

เมื่อแต่ละสถานีเชื่อมต่อเข้าด้วยกันแล้วจะทำให้ผู้ใช้ติดต่อหรือส่งข้อมูลให้แก่กันได้ เช่น การใช้จดหมายอิเล็กทรอนิกส์(Electronic Mail) และช่วยให้ผู้ใช้จัดการข้อมูลจากที่อื่นได้

2.8.4 การใช้โปรแกรมร่วมกัน

ในระบบเครือข่ายผู้ใช้สามารถจะใช้ข้อมูลและซอฟต์แวร์ร่วมกันได้ เพื่อลดความซ้ำซ้อนของข้อมูล สื่อบันทึกข้อมูล ตลอดจนการค้นหาตรวจสอบข้อมูลได้รวดเร็วและถูกต้อง การปรับปรุงโปรแกรมที่ใช้งานร่วมกันก็สามารถกระทำ ณ จุดเดียว

2.8.5 การใช้ทรัพยากรร่วมกัน

เนื่องจากระบบเครือข่ายท้องถิ่นอนุญาตให้ผู้ใช้ระบบเครือข่าย ใช้อุปกรณ์ร่วมกัน (เช่นเครื่องพิมพ์ โมเด็ม ประตูสื่อสาร อุปกรณ์ด้านกราฟิกและอุปกรณ์การเก็บข้อมูลเป็นต้น) ซึ่งจะช่วยลดความซ้ำซ้อนของอุปกรณ์ทำให้ประหยัดค่าใช้จ่าย นอกจากนี้ระบบเครือข่ายท้องถิ่นยังช่วยให้ผู้จัดการระบบสามารถตรวจสอบการใช้อุปกรณ์ต่างๆ เหล่านี้จากผู้ใช้ในระบบได้อีกด้วย

เมื่อระบบการประมวลผลแบบกระจายได้เริ่มขยายตัวไปมากขึ้น ประกอบกับราคาของเครื่องไมโครคอมพิวเตอร์ถูกลง ในขณะที่ประสิทธิภาพของเครื่องเพิ่มขึ้นด้วย ดังนั้นการเชื่อมต่อไมโครคอมพิวเตอร์ให้เป็นระบบเครือข่ายท้องถิ่น จะช่วยเพิ่มประสิทธิภาพของการใช้งานอุปกรณ์ต่างๆ ในสำนักงานได้ดียิ่งขึ้น

ตัวกลางในการส่งข้อมูลหรือสายส่งข้อมูล มีอยู่หลายแบบ เช่น โคแอคเชียล (coaxial) สายโทรศัพท์(twisted pair) และเส้นใยนำแสง(fiber optic) เป็นต้น ซึ่งสามารถส่งข้อมูลได้ด้วยความเร็วสูง และอุปกรณ์ที่ต่อเชื่อมกับสายส่งข้อมูลเหล่านี้ไม่จำเป็นต้องมีศูนย์สลับสาย(Switching Center) ซึ่งมีราคาแพง

ระบบเครือข่ายท้องถิ่น อาจนำมาประยุกต์ใช้งานได้หลายแบบ เช่น

1. นำมาใช้ควบคุมการผลิตในโรงงานอุตสาหกรรม

2. เทอร์มินัลในการบริการ ณ จุดขายในซูเปอร์มาร์เก็ตกับเครื่องควบคุมสต๊อกสินค้า และระบบบัญชี

3. เชื่อมต่อไมโครคอมพิวเตอร์ต่างๆและเครื่องพิมพ์ในสำนักงานเพื่อจัดทำงานด้านปฏิบัติงานได้รวดเร็วยิ่งขึ้น เช่นระบบบัญชีพื้นฐานซึ่งรวมบัญชีแยกประเภท บัญชีลูกหนี้ บัญชีสินค้า บัญชีเงินเดือนค่าจ้างเข้าด้วยกัน ระบบทะเบียน และระบบวัดผลนักศึกษาในสถาบันการศึกษา เป็นต้น

4. เชื่อมต่อเครื่องคอมพิวเตอร์และเครื่องพิมพ์ในสำนักงาน เพื่อให้เป็นสำนักงานอิเล็กทรอนิกส์(Electronic Office) หรืออาคารระบบอัตโนมัติ(Intelligent Building)

บทที่ 3
การออกแบบและการทำงานของอุปกรณ์โครงข่าย M-NET

เมื่อเลือกการสื่อสารข้อมูลดิจิตอลอนุกรมแบบไม่สัมพันธ์บนโครงข่ายคอมพิวเตอร์ IBM-PC ดังนั้นจึงต้องศึกษาช่องทางติดต่ออุปกรณ์ภายนอกของเครื่อง IBM-PC เนื่องจากเครื่อง IBM-PC ก็มีช่องต่อเชื่อมข้อมูลอนุกรมแบบไม่สัมพันธ์อยู่แล้ว แต่มีอัตราบอดในการรับส่งข้อมูลต่ำ คือมีอัตราบอดมาตราฐานสูงสุดเพียง 9.6 K ซึ่งเป็นอัตราบอดที่ต่ำกว่าที่โครงข่าย M-NET ต้องการจึงได้ค้นหาอุปกรณ์ที่สามารถแปลงสัญญาณข้อมูลแบบขนานไปเป็นสัญญาณข้อมูลอนุกรมที่มีอัตราบอดสูงที่สุดเท่า ที่จะหาข้อมูลได้จึงได้อุปกรณ์ของบริษัท Intel เบอร์ 82510 ซึ่งเป็นอุปกรณ์ควบคุมการแปลงสัญญาณข้อมูลแบบขนาน 8 บิทให้เป็นสัญญาณข้อมูลอนุกรมที่มีอัตราบอดสูงสุด 288K ซึ่งเป็นอัตราบอดที่ส่งบิทของข้อมูลที่ 288000 บิท ต่อวินาทีจริงๆ โดยไม่ต้องมีการบีบลดขนาดของข้อมูลก่อนการส่งข้อมูล

เมื่อได้อุปกรณ์ Intel 82510 จึงออกแบบวงจรอินเตอร์เฟสกับเครื่อง IBM-PC ผ่านทางสล๊อต(ช่องต่อเชื่อมของเครื่อง IBM-PC) ในรูปแบบของแผ่นวงจรพิมพ์ที่ใช้เสียบบนสล๊อตของเครื่อง IBM-PC เรียกแผ่นวงจรพิมพ์นี้ว่า M-NETการ์ด (M-NET Card)

3.1 วงจร M-NETการ์ด

วงจรM-NETการ์ดมีอุปกรณ์Intel 82510 เป็นอุปกรณ์หลักทำหน้าที่รับหรือส่งข้อมูลกับเครื่อง IBM-PCแบบขนาน โดยใช้อุปกรที่ทีแอล 74LS245 เป็นบัฟเฟอร์คอยจังหวะรับหรือส่งสัญญาณข้อมูล การให้จังหวะการรับหรือส่งถูกกำหนดที่ขา DIR โดยมีที่ทีแอล 74LS682 เป็นตัวกำหนดทิศทางสัญญาณที่ขา DIR จากคำสั่งรับหรือส่งที่อุปกรณ์ประมวลผลของเครื่อง IBM-PC ส่งคำสั่งไปยังแอดเดรสที่ตรงกับแอดเดรสของM-NETการ์ดที่ตั้งไว้ล่วงหน้าด้วยดิพสวิทช์ ในที่นี้ตั้งแอดเดรสไว้ที่ 2E8h

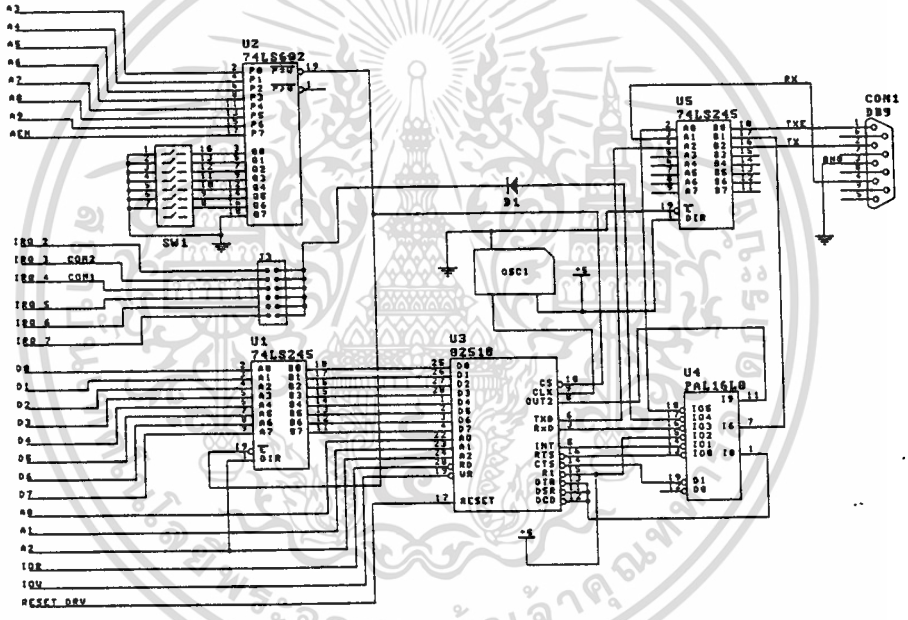
เมื่ออุปกรณ์ Intel 82510 ได้สัญญาณข้อมูลที่จะรับหรือส่งแล้วก็จะแปลงสัญญาณข้อมูลแบบขนานเป็นอนุกรม โดยใช้สัญญาณนาฬิกาจากอุปกรณ์สร้างสัญญาณนาฬิกาที่ 18.432 MHz(OSC1)เมื่อ Intel 82510 ต้องการส่งข้อมูลด้วยอัตราบอดเท่าไรก็ต้องใช้ซอฟแวร์ส่งค่าที่เป็นตัวหารจากความถี่ของสัญญาณนี้ให้ลดลงเท่าอัตราบอดที่ต้องการใช้ การคำนวณค่าตัวหาร

ใช้สูตร อัตราบอด = 18.432M/ตัวหาร

ตัวอย่างเช่น อัตราบอด 288K = 18.432M/ตัวหาร

ตัวหาร = 64

เมื่อได้อัตราบอดก็จะรับหรือส่งข้อมูลด้วยอัตราบอดนั้น โดยรับหรือส่งสัญญาณข้อมูลผ่านทีทีแอล 74LS 245 ก่อนที่จะรับหรือผ่านส่งคอนเน็กเตอร์แบบ 9 ขาไปยังอุปกรณ์ส่วนอื่นๆ โดยส่งสัญญาณข้อมูล(TX) ที่ขา 2 และรับสัญญาณข้อมูล(RX)ที่ขา 4 และมีขา 3 เป็นขากราวนด์ของสัญญาณ นอกจากนี้ยังมีสัญญาณ TXE ที่ขา 1 ซึ่งเป็นสัญญาณควบคุมการส่งข้อมูล ที่ขา TXE จะเปลี่ยนสัญญาณ 0 โวลต์เป็น +5 โวลท์ เมื่อมีการส่งข้อมูลและตกลงเป็น 0 โวลท์เหมือนเดิมเมื่อหยุดส่งข้อมูลออกจาก Intel 82510 สัญญาณนี้ถูกส่งมาจากอุปกรณ์ PAL16L8 ซึ่งเป็นอุปกรณ์เฉพาะที่ต้องซื้อเพิ่มเมื่อต้องการให้ Intel 82510 มีสัญญาณควบคุมการส่งข้อมูล



รูปที่ 23. วงจรการ์ด M-NET

3.2 วงจรเนกาทีฟพัลส์

การส่งสัญญาณในสายส่งที่มีความยาวมากต้องเตรียมสัญญาณให้เหมาะสม ถ้าส่งสัญญาณข้อมูลจากอุปกรณ์ทีทีแอล โดยตรงจะไม่สามารถส่งสัญญาณผ่านสายส่งที่มีความยาวมากได้ เพราะสายส่งจะมีการลดทอนสัญญาณสูงขึ้นตามความยาวของสาย การที่จะส่งสัญญาณให้ได้ในสายส่งยาวคือจะต้องรักษารูปสัญญาณที่ปลายสายให้คงรูปสัญญาณเดิมมากที่สุด ถ้ารูปสัญญาณต่างจากเดิมมากกว่าที่อุปกรณ์

ปลายทางจะรับได้ ข้อมูลที่รับได้ก็จะไม่ตรงกับค่าข้อมูลที่ส่งไป หลักการของตัวส่งสัญญาณของวงจรเนกาทีฟพัลส์ ก็คือขยายสัญญาณข้อมูลที่จะส่งทั้งทางด้านแรงดันและกระแส เพื่อให้เหมาะกับสายส่งที่ใช้ทดสอบ การกำหนดค่าตัวต้านทานที่ปลายสายส่ง(R-Terminated) ก็มีความสำคัญในการดึงกระแสให้มากพอที่สัญญาณจะรักษารูปสัญญาณที่ปลายสายให้คงเดิมมากที่สุดได้

การส่งสัญญาณ 2 สัญญาณในสายส่งคู่เดียวสามารถทำได้หลายวิธี เช่นการแบ่งจังหวะเวลาในการส่งหรือการแบ่งระดับแรงดันในการส่ง แต่เทคนิคเนกาทีฟพัลส์ส่งแบบแบ่งระดับแรงดันโดยส่งสัญญาณนาฬิกาในระดับแรงดันบวกและสัญญาณข้อมูลในระดับแรงดันลบ วงจรที่ใช้ส่งสัญญาณนาฬิกาในระดับแรงดันบวก คือวงจรสร้างสัญญาณสวิตชิ่ง(Switching) วงจรที่ใช้ส่งสัญญาณข้อมูลในระดับแรงดันลบคือ วงจรสร้างสัญญาณแรงดันลบ(Negative Pulser) วงจรที่ใช้ส่งสัญญาณนาฬิกาในระดับแรงดันบวก คือวงจรสร้างสัญยาณสวิตชิ่ง(Switching) วงจรที่ใช้ส่งสัญญาณข้อมูลในระดับแรงดันลบ คือวงจรสร้างสัญญาณแรงดันลง(Negative Pulser)

การรับสัญญาณเนกาทีฟพัลส์ที่ปลายทางต้องมีวงจรแยกสัญญาณที่มีระดับแรงดันต่างกัน โดยใช้วงจรแยกสัญญาณนาฬิกาที่มีความสัมพันธ์กับสัญญาณข้อมูล โดยแยกสัญญาณนาฬิกาที่ระดับแรงดันบวกออกเรียกวงจรนี้ว่าวงจรแยกสัญญาณชิงโครนัสและใช้วงจรแยกสัญญาณข้อมูลระดับแรงดันลบว่าวงจรตรวจจับสัญญาณแรงดันลบ(Negative Detector)

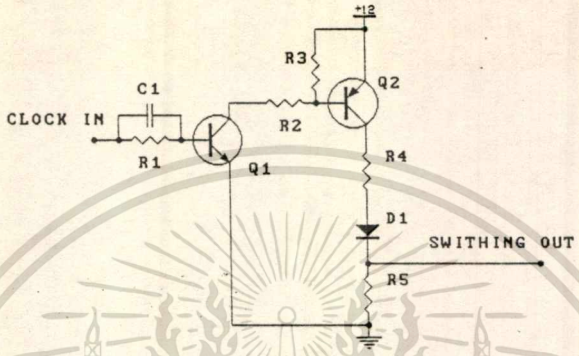
3.2.1 วงจรสร้างสัญญาณสวิตชิ่ง

วงจรนี้ใช้สัญญาณนาฬิกาที่มีความถี่เท่ากับอัตราบอดเป็นอินพุท โดยสัญญาณนี้ต้องเป็นคลื่นสี่เหลี่ยมแบบมีช่วงแรงดันสูงและช่วงแรงดันต่ำเท่ากัน โดยวงจรนี้จะขยายแรงดันจากสัญญาณอินพุท 0 ถึง +12 โวลต์ และวงจรนี้สามารถจ่ายกระแสให้กับโหลด ซึ่งเกิดขึ้นจากอิมพีแดนซ์ของสายส่งอิมพิแดนซ์ของอินพุทของวงจรรับสัญญาณ และค่าความต้านทานของตัวต้านทานที่ปลายสาย (R-Terminate) ได้สูงถึง 250 มิลลิแอมป์

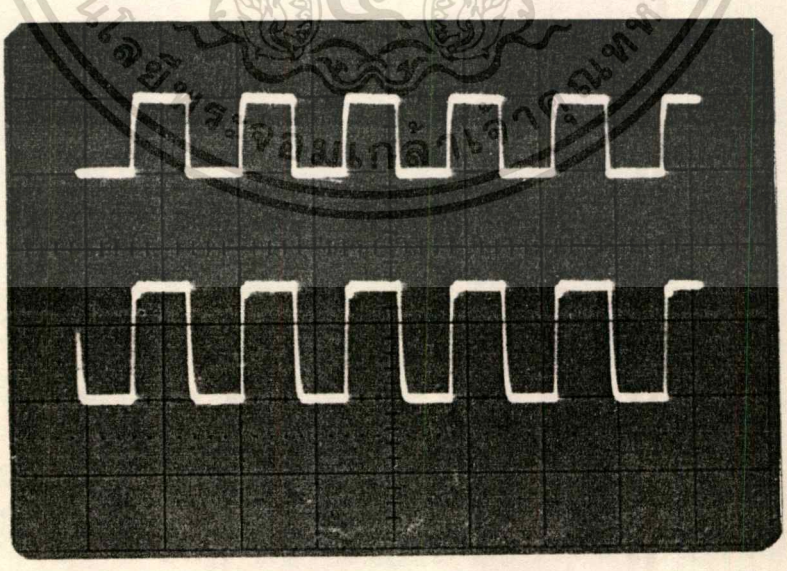
จากรูปที่ 24 เมื่อสัญญาณนาฬิกาที่อินพุททำให้มีระดับแรงดัน +5 โวลต์ ทำให้ทรานซิสเตอร์ Q1 นำกระแสจากแหล่งจ่าย +5 โวลต์ไหลผ่านตัวต้านทาน R3 ลงกราวด์ ทำให้ทรานซิสเตอร์ Q2 นำกระแส กระแสไหลจากขาอิมิตเตอร์ออกขาคอลเลคเตอร์ผ่านตัวต้านทาน R4, ไดโอด D1 และตัวต้านทาน R5 ลงกราวด์ ทำให้เกิดแรงดันตกคร่อมที่อุปกรณ์ทั้ง 3 ตัว ประมาณ 12 โวลต์ แต่จุดที่เลือกเป็นเอาท์พุทคือจุดคร่อมตัวต้านทาน R5 ซึ่งมีแรงดันตกคร่อมประมาณ 9 โวลต์ R4 เป็นอุปกรณ์จำกัดกระแสกรณีสายส่งลัดวงจร และ D1 ทำหน้าที่ป้องกันกระแสไหลเข้าขาคอลเลคเตอร์ของ Q2 เมื่อระดับแรงดันที่อินพุทเป็น 0 โวลท์ทรานซิสเตอร์ Q1 ไม่มีกระแสไบอัสก็จะหยุดนำกระแสทำให้ไม่มีกระแสไบอัสทรานซิสเตอร์ Q2 ทำให้ไม่มีกระแสไหลผ่าน R5 จึงไม่มีแรงดันตกคร่อม

รูปสัญญาณเอาท์พุทจึงมีเฟสตรงกับสัญญาณอินพุทแต่มีระดับแรงดันสูงกว่า และสามารถจ่าย

กระแสได้สูงกว่าเพราะทรานซิสเตอร์ Q2 สามารถจ่ายกระแสได้สูงสุด 800 มิลลิแอมป์ แต่วงจรจำกัดกระแสด้วย R4 (47 โอมห์) ไว้ที่ประมาณ 250 มิลิแอมป์ดังรูปที่ 25



รูปที่ 24. วงจรสร้างสัญญาณสวิตชิ่ง



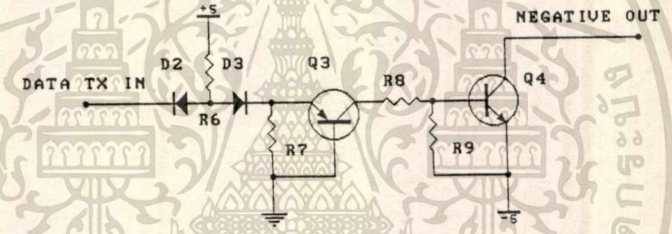
รูปที่ 25. สัญญาณอินพุทและ เอาท์พุทของวงจรสร้างสัญญาณสวิตชิ่ง

3.2.2 วงจรสร้างสัญญาณช่วงแรงดันลบ

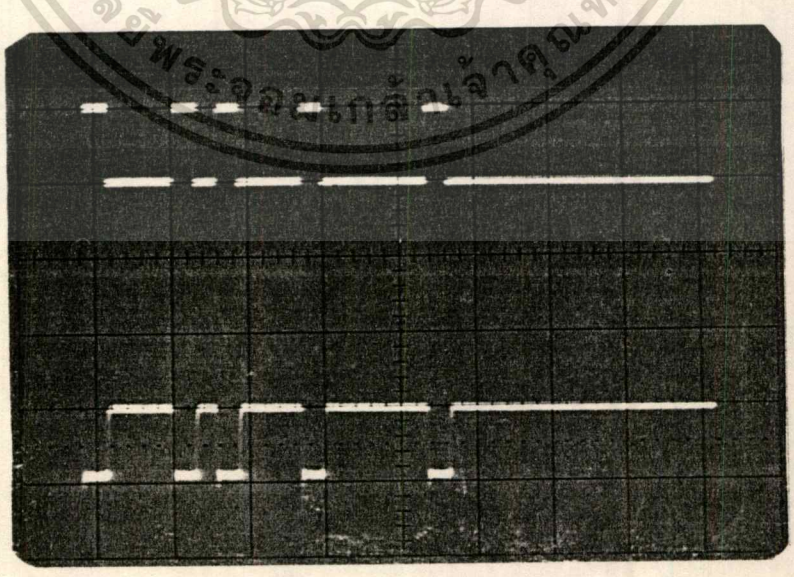
จากรูปที่ 26 วงจรสร้างสัญญาณแรงดันลบ เมื่อสัญญาณอินพุทมีระดับแรงดัน +5 โวลต์ ก็จะมีกระแสไหลจากแหล่งจ่าย +5 โวลท์ ผ่าน R6, D3 และ R7 ลงกราวนด์ทำให้เกิดแรงดันตกคร่อมที่ R7 เมื่อแรงดันตกคร่อมมีค่ามากกว่า 0.7 โวลท์ ทรานซิสเตอร์ Q3 ก็จะนำกระแส ทำให้มีกระแสไหลผ่าน R8 และ R9 ลงแหล่งจ่าย -5 โวลท์ จึงมีแรงดันตกคร่อม มากกว่า 0.7 โวลท์ Q4 ก็จะนำกระแสจากแหล่งจ่าย -5 โวลท์

เมื่อสัญญาณข้อมูลที่อินพุทเป็น 0 โวลต์ ขาอินพุทเหมือนถูกลัดวงจรทำให้กระแสไหลผ่าน R6 และ D2 ลงกราวด์ ทำให้ไม่มีกระแสไหลผ่าน R7 จึงไม่เกิดแรงดันตกคร่อมทำให้ Q3 หยุดนำกระแสเมื่อไม่มีการไหลผ่าน R8 และ R9 ทำให้ไม่เกิดแรงดันตกคร่อม R9 ดังนั้น Q4 จึงหยุดนำกระแส

เมื่อนำตัวต้านทานต่อที่เอาท์พุทลงกราวนด์ ก็จะได้สัญญาณเอาท์พุทมีเฟสตรงกันข้ามกันกับสัญญาณอินพุทและระดับแรงดันอยู่ระหว่าง 0 ถึง -5 โวลท์ ดังรูปที่ 27



รูปที่ 26. วงจรสร้างสัญญาณช่วงแรงดันลบ

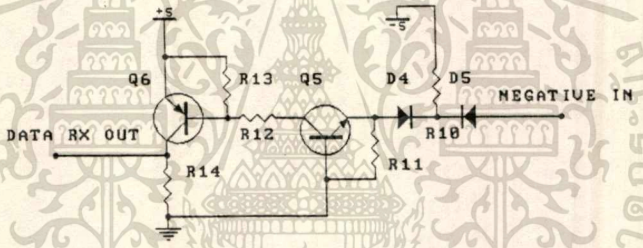


รูปที่ 27. สัญญาณอินพุทและ เอาท์พุทของวงจรสร้างสัญญาณช่วงแรงดันลบ

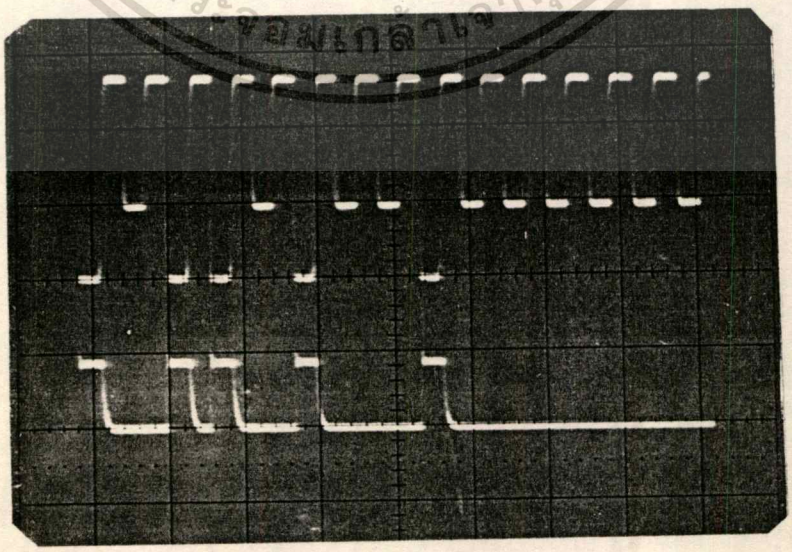
3.2.3 วงจรตรวจจับสัญญาณช่วงแรงดันลบ

จากรูปที่ 28 เมื่อมีสัญญาณระดับแรงดันลบที่อินพุท กระแสก็จะไหลจากแหล่งจ่ายแรงดัน -5 โวลท์ ผ่านไดโอด D4 และ R11 ลงกราวนด์ทำให้เกิดแรงดันตกคร่อม R11 เมื่อแรงดันตกคร่อมสูงกว่า 0.7 โวลท์ ทรานซิสเตอร์ Q5 จะนำกระแสทำให้กระแสจากแหล่งจ่าย +5 โวลท์ ผ่าน R13, R12, Q5 และ D4 ลงแหล่งจ่าย -5 โวลท์ ทำให้เกิดแรงดันตกคร่อม R13 เมื่อแรงดันสูงกว่า 0.7 โวลท์ R6 จึงนำกระแสไหลผ่าน R14 ลงกราวนด์ ทำให้เกิดแรงดันตกคร่อม R14 สัญญาณเอาท์พุทก็จะมีระดับแรงดัน +5 โวลท์

เมื่อสัญญาณอินพุทมีระดับแรงดัน 0 โวลท์ทำให้ไม่มีกระแสไหลผ่าน R11 แรงดันตกคร่อม R11 น้อยกว่า 0.7 โวลท์ Q5 หยุดนำกระแสทำให้ไม่มีกระแสไหลผ่าน R13 และ R12 ทำให้แรงดันตกคร่อม R13 น้อยกว่า 0.7 โวลท์ Q6 หยุดนำกระแสสัญญาณที่เอาท์พุทก็จะมีระดับแรงดันเป็น 0 โวลท์ สัญญาณเอาท์พุทจึงมีเฟสตรงข้ามกับสัญญาณอินพุท และมีระดับแรงดันระหว่าง 0 ถึง +5 โวลท์ ดังรูปที่ 29



รูปที่ 28. วงจรตรวจจับสัญญาณช่วงแรงดันลบ



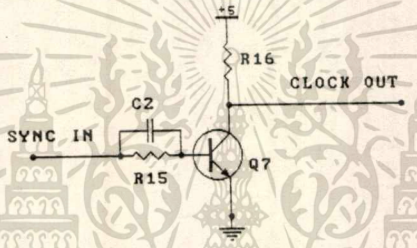
รูปที่ 29. สัญญาณอินพุทและเอาท์พุทของวงจรตรวจจับสัญญาณช่วงแรงดันลบ

3.2.4 วงจรแยกสัญญาณซิงโครนัส

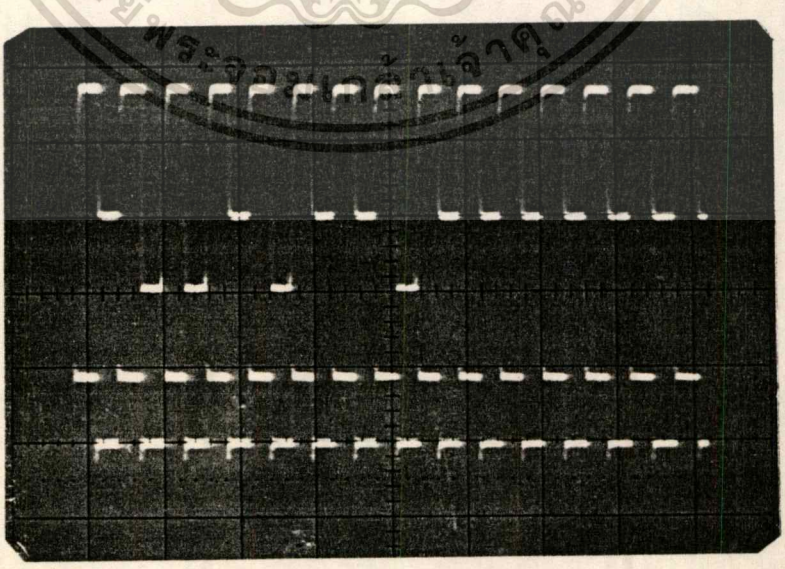
วงจรรูปที่ 30 วงจรแยกสัญญาณซิงโครนัส วงจรนี้จะทำงานเมื่อสัญญาณอินพุทมีระดับแรงดันสูงกว่า 0 โวลท์ เมื่อแรงดันสูงจนไบอัส Q7 ให้นำกระแส กระแสก็จะไหลจากแหล่งจ่ายไฟ +5 โวลท์ ผ่าน R16 ลงกราวนด์ ทำให้สัญญาณเอาท์พุทมีระดับแรงดันเป็น 0 โวลท์

เมื่ออินพุทเป็น 0 หรือต่ำกว่า 0 โวลท์ ก็จะไม่มีกระแสไบอัส Q7 ทำให้ Q7 หยุดนำกระแส สัญญาณเอาท์พุท จึงมีระดับแรงดันเป็น +5 โวลท์

สัญญาณเอาท์พุทนี้จะเฟสตรงกันข้ามสัญญาณอินพุท แต่มีระดับแรงดัน 0 ถึง +5 โวลท์ ดังรูปที่ 31



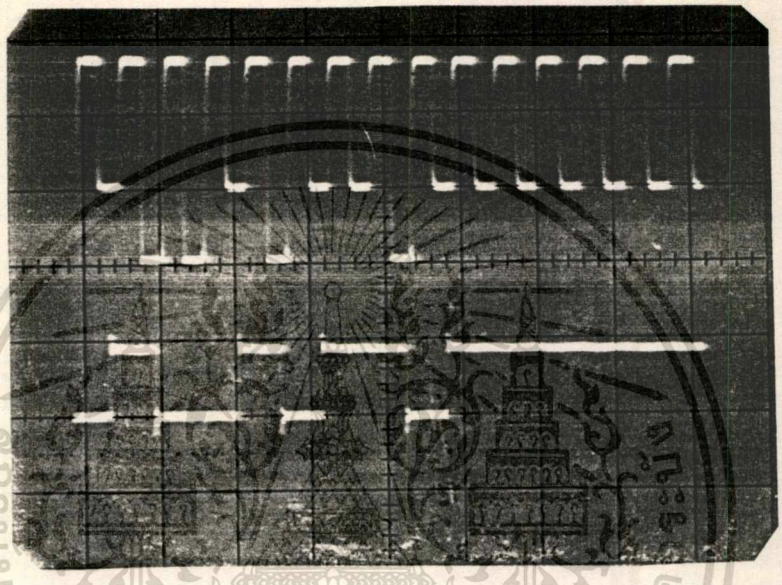
รูปที่ 30 วงจรแยกสัญญาณซิงโครนัส



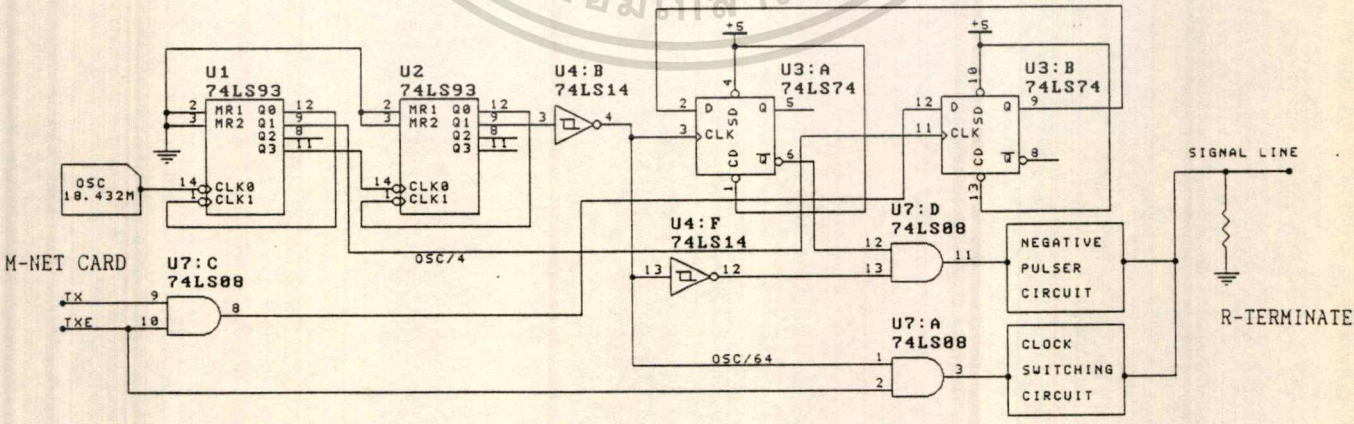
รูปที่ 31. สัญญาณอินพุทและเอาท์พุทของวงแยกสัญญาณซิงโครนัส

3.3 วงจรแต่งสัญญาณ

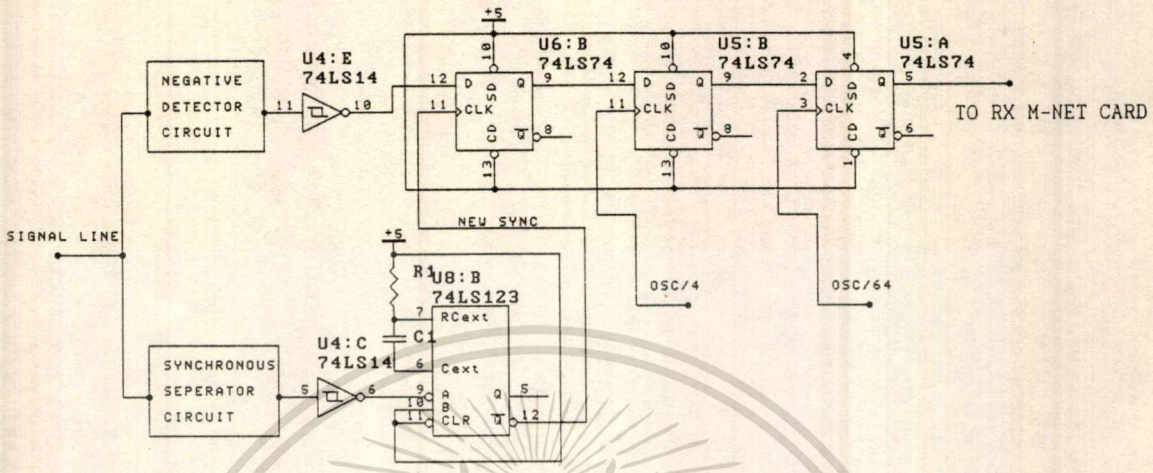
วงจรเนกาทีฟพัลส์ต้องการสัญญาณนาฬิกาที่สัมพันธ์ด้วยสัญญาณข้อมูล และต้องการสัญญาณข้อมูลที่แอน (AND) กับค่าอินเวอร์สของสัญญาณนาฬิกา เพื่อให้สัญญาณในสายส่งเป็นไปตามรูปแบบของเทคนิคเนกาทีฟพัลส์ ดังรูปที่ 32 จึงต้องมีวงจรแต่งสัญญาณเพื่อให้เข้าใจวงจรง่ายจึงต้องแบ่งวงจรแต่งสัญญาณออกเป็นสองวงจรคือ วงจรแต่งสัญญาณที่ต้องการส่งและวงจรแต่งสัญญาณที่ได้จากการรับ



รูปที่ 32. สัญญาณเนกาทีฟพัลส์



รูปที่ 33. รูปวงจรแต่งสัญญาณส่ง



รูปที่ 34. รูปวงจรแต่งสัญญาณรับ

3.3.1 วงจรแต่งสัญญาณก่อนส่ง วงจรนี้จะหารความถี่ของสัญญาณนาฬิกาจากออสซิเลเตอร์ 18.432 เมกกะเฮิรทซ์ออกมาเป็นความถี่ 2 ความถี่ คือหารความถี่ด้วย 4 ให้ได้สัญญาณความถี่ 4.608 เมกกะเฮริทซ์ และหารความถี่ด้วย 64 เพื่อให้ได้สัญญาณความถี่ 288 กิโลเฮิรทซ์ การหารใช้อุปกรณ์ทีทีแอล 74LS93 2 ตัว แต่ละตัวประกอบด้วย ดี-ฟลิปฟลอป 4 ตัว ดี-ฟลิปฟลอบแต่ละตัวทำหน้าที่หารสัญญาณด้วย 2 สัญญาณที่หารได้มีช่วงระดับแรงดันสูงและต่ำมีค่าเท่ากันเสมอ

นำสัญญาณนาฬิกาความถี่ 288 กิโลเฮริตซ์ เข้าวงจรอินเวอร์เตอร์แบบสมิธทริกเกอร์(74LS 14) เพื่อให้รูปสัญญาณของขาขึ้นและลงของสัญญาณมีความชันขึ้น

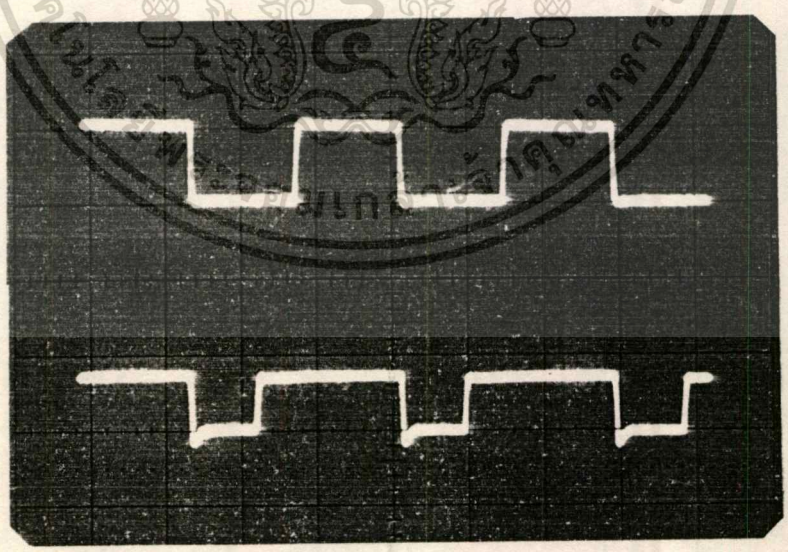
สัญญาณข้อมูลจะถูกส่งมาที่ขา TX สัญญาณควบคุมการส่งก็ถูกส่งมาที่ขา TXE แอนสัญญาณทั้งสองเพื่อให้แน่ใจว่าสัญญาณข้อมูลนั้นเป็นสัญญาณข้อมูลที่ต้องการส่งจริง สัญญาณควบคุมการส่งยังถูกนำไปแอนกับสัญญาณนาฬิกาความถี่ 288 กิโลเฮริตซ์ เพื่อให้มีสัญญาณนาฬิกาส่งออกไปในสายส่งขณะที่มีการส่งสัญญาณข้อมูลเท่านั้น สัญญาณนาฬิกาที่ได้จากการแอนจะถูกส่งไปวงจรสวิตชิ่ง เมื่อสัญญาณนาฬิกาที่ 288 กิโลเฮริตซ์ จะถูกอินเวอร์สอีกครั้งเพื่อให้มีเฟสตรงข้ามกับสัญญาณนาฬิกาในสายส่งและแอนกับสัญญาณข้อมูลที่มีการเลื่อนเฟสให้สัมพันธ์กับสัญญาณนาฬิกา แล้วจึงส่งให้วงจรสร้างสัญญาณแรงดันลบ

สัญญาณจากวงจรสวิตชิ่งและวงจรสร้างสัญญาณแรงดันลบถูกส่งออกไปพร้อมกันในสายส่ง ซึ่งมีรูปสัญญาณดังรูป 32

การเลื่อนเฟสของสัญญาณข้อมูลให้สัมพันธ์สัญญาณนาฬิกา ใช้ทีทีแอลเบอร์ 74LS74 ดี-ฟลิปฟลอบสองตัว ดี-ฟลิปฟลอบตัวแรกใช้สัญญาณความถี่ 4.608 เมกกะเฮริตซ์ส่งไปที่ขาสัญญาณนาฬิกา (CLK) และส่งสัญญาณข้อมูลไปที่ขาข้อมูล(D) ขอบขาขึ้นของสัญญาณความถี่ 4.608 เมกกะเฮริตซ์ขอบแรกที่เกิดหลังขอบขาขึ้นของสัญญาณข้อมูล จะเป็นจังหวะที่สร้างสัญญาณข้อมูลขึ้นมาใหม่และมีเฟสสัมพันธ์กับความถี่ 4.608 เมกกะเฮริตซ์ ทำนองเดียวกันนำสัญญาณที่ได้นี้ไปเลื่อนเฟสอีกครั้งกับดี-ฟลิปฟลอบอีกครั้งแต่ใช้ความถี่ 288 กิโลเฮริตซ์ ก็จะได้สัญญาณข้อมูลใหม่ที่มีค่าเท่ากับข้อมูลเดิมและมีเฟสสัมพันธ์กับสัญญาณนาฬิกาความถี่ 288 กิโลเฮริตซ์ ตามต้องการ

3.3.2 วงจรแต่งสัญญาณที่รับจากสายส่ง

เมื่อได้สัญญาณนาฬิกาที่แยกออกมาจากสัญญาณในสายส่งด้วยวงจรแยกสัญญาณซิงโครนัสสัญญาณที่ได้มีขอบขาขึ้นและลงลาดเอียงต้องแต่งสัญญาณผ่านอินเวอร์เตอร์แบบสมิธทริกเกอร์ให้มีขอบขาขึ้นและลงชันขึ้นนำสัญญาณนี้ไปเลื่อนขอบขาขึ้นให้ไปขึ้นที่จุดกึ่งกลางในระหว่างช่วงแรงดัน 0 โวลท์ โดยใช้ทีทีแอล 74LS123 ทำหน้าที่กำหนดช่วงเวลาให้ขอบขาขึ้นขึ้นหลังจากถูกทริกด้วยขอบขาขึ้น ช่วงเวลาถูกกำหนดด้วยค่าความต้านทาน R1 และตัวเก็บประจุ C1 ในที่นี้ 1 คาบเวลาของสัญญาณนาฬิกาความถี่ 288 กิโลเฮริตซ์เท่ากับ 3.472 ไมโครวินาที ช่วงหนึ่งในสี่ของคาบเวลาเท่ากับ 0.868 ไมโครวินาที



รูปที่ 35. สัญญาณนาฬิกาที่เลื่อนขอบขาขึ้นออกไป

ใส่สูตร $T = 0.30 \star R1 \star C1$ (กำหนดค่า R1=25K)

เพราะฉนั้น $C1 = \frac{T}{0.7(R1)} = \frac{868 \times 10^{-9}}{0.7(25 \times 10^{3})} = 49.60$ pF

สัญญาณนาฬิกาที่เลื่อนของขาขึ้นเพื่อใช้ในการเลือกอ่านสถานะกลางสัญญาณของบิทสัญญาณข้อมูลเพื่อลดความผิดพลาดในการอ่านสถานะที่เกิดจากความลาดเอียงของสัญญาณ ที่เกิดมากขึ้นตามความยาวของสายส่งที่เพิ่มขึ้น และการเลือกอ่านตรงกลางบิทสัญญาณซึ่งเป็นจุดที่สูงที่สุดหรือต่ำที่สุดของสัญญาณก็ช่วยให้ตัดสินใจได้ว่าบิทนี้มีค่าเป็นหนึ่งหรือเป็นศูนย์แม้สัญญาณจะถูกลดทอนขนาดลงไปดังรูปที่ 3.14

สัญญาณข้อมูลที่อ่านได้ก็จะถูกนำไปเลื่อนเฟส โดยใช้ฟลิปฟลอป 2 ตัว ตัวแรกเลื่อนเฟสด้วยความถี่ 4.608 เมกกะเฮริตซ์ ตัวที่สองเลื่อนเฟสด้วยความถี่ 288 กิโลเฮริตซ์ ทำให้สัญญาณข้อมูลที่อ่านได้มีอัตราบอด 288 K และสัมพันธ์กับสัญญาณนาฬิกา

เมื่อเปรียบเทียบสัญญาณนี้กับสัญญาณที่วัดก่อนส่งมีรูปสัญญาณใกล้เคียงกันมาก เมื่อเทียบกับสัญญาณที่ได้จากการรับส่งของอุปกรณ์มาตรฐาน RS-485 ที่ระยะและอัตราบอดเท่ากันรูปสัญญาณที่ได้แตกต่างกันอย่างเห็นได้ชัด

บทที่ 4
การออกแบบโปรแกรม M-WARE

โปรแกรม M-WARE เป็นโปรแกรมที่เขียนด้วยภาษาซีใช้กับเครื่องคอมพิวเตอร์ส่วนบุคคลของบริษัทไอบีเอ็ม (IBM-PC) รุ่น XT หรือ AT หรือใช้กับเครื่องคอมพิวเตอร์ที่เทียบเท่า ใช้ระบบจัดการของดอส (DOS)

โปรแกรม M-WARE เป็นโปรแกรมรวมที่สามารถจัดตั้งเครื่องคอมพิวเตอร์เครื่องใดบนโครงข่ายเป็นสถานีบริการแฟ้มข้อมูลหรือเป็นสถานีผู้ใช้ได้อย่างอิสระ (สถานีคือเครื่องคอมพิวเตอร์ที่ติดตั้งอุปกรณ์และมีสายส่งของโครงข่าย M-NET เชื่อมโยงอยู่)

รายละเอียดของโปรแกรม M-WARE

4.1 โปรแกรมควบคุม Intel 82510

การใช้งาน Intel 82510 ต้องมีการกำหนดค่าในโปรแกรมเพื่อควบคุมการทำงานของ Intel 82510 ค่าที่ต้องกำหนดมีดังนี้

1. BASE ADDRESS = 2E8 /*COM4=2E8*/
2. 82510 REGISTER ADDRESSES /*bank 0,1,2,3*/

การกำหนดค่าทั้ง 2 หัวข้อนี้ให้ดูที่ตัวอย่างจากโปรแกรม M-WARE ในภาคผนวก ง.นอกจากการกำหนดค่าในโปรแกรมแล้วต้องมีโปรแกรมฟังก์ชั่นย่อยที่ใช้ควบคุม Intel 82510 ที่สำคัญมีดังนี้

1. Transmit Mode
2. Receive Mode
3. Set Ninth Bit
4. Clear Ninth Bit
5. Send Char
6. Recv Char
7. Set Band Rate

ให้ดูตัวอย่างการเขียนโปรแกรมฟังก์ชั่นทั้ง 7 นี้ จากตัวโปรแกรม M-WARE ในภาคผนวก ง.

4.2 รูปแบบของเฟรม (FRAME)

การส่งข้อมูลออกไปในสายส่งสัญญาณโปรแกรม M-WARE มีการกำหนดรูปแบบบเพื่อให้ข้อมูลเหล่านั้นเกิดความหมายแตกต่างกัน และที่สำคัญโปรแกรม M-WARE สามารถควบคุมขนาดของไบท์ได้สองขนาดคือขนาดข้อมูล 8 บิทต่อไบท์ และขนาดข้อมูล 9 บิทต่อไบท์

ขนาดข้อมูล 8 บิทต่อไบท์ โปรแกรม M-WARE ให้แสดงความหมายเป็นไบท์ซิงโครนัส(SYNC) ไบท์ข้อมูล(DATA) ไบท์ชื่อแฟ้ม(FILE NAME) และไบท์ขนาดแฟ้ม(FILE NAME)

ขนาดข้อมูล 9 บิทต่อไบท์โปรแกรม(M-WARE) ใช้แสดงความหมายเป็นไบท์แสดงตำแหน่งสถานีที่รับส่งข้อมูล ไบท์แสดงตำแหน่งสถานีที่ส่งข้อมูลและไบท์คำสั่ง

นอกจากนี้โปรแกรม M-WARE ยังนำไบท์เหล่านี้มารวมเป็นรูปแบบเฟรม (FRAME) เพื่อให้เกิดความหมายที่แตกต่างและเป็นการแบ่งข้อมูลที่มีขนาดยาวมาก เช่นแฟ้มข้อมูลให้ส่งไปครั้งละเฟรม ไม่ได้ส่งไปทั้งแฟ้มข้อมูล เป็นการแบ่งเวลาในการใช้สายส่งสัญญาณ (Time sharing) ทำให้สถานีผู้ใช้เสมือนใช้สายส่งอยู่ตลอดเวลาและยังเป็นการประหยัดเวลาในการใช้สายส่ง ในกรณีรับข้อมูลผิดการส่งข้อมูลใหม่ (Retransmission) ก็ส่งเพียงเฟรมเดียวไม่ต้องส่งใหม่ทั้งแฟ้ม

โปรแกรม M-WARE กำหนดรูปแบบของเฟรม 3 รูปแบบดังนี้

1. เฟรมคำสั่ง เฟรมนี้ประกอบด้วยไบท์ซิงโครนัส 10 ไบท์ ไบท์หมายเลขสถานีผู้รับ 1ไบท์ ไบท์หมายเลขสถานีผู้ส่ง 1 ไบท์ และไบท์คำสั่ง 1 ไบท์ เฟรมคำสั่งมีความหมายให้สถานีผู้รับทราบว่าสถานีหมายเลขใดเป็นสถานีผู้รับและสถานีหมายเลขใดเป็นสถานีผู้ส่ง และสถานีผู้ส่งมีความต้องการให้สถานีผู้รับดำเนินการอย่างไร ส่วนความหมายของไบท์คำสั่งจะอธิบายอีกครั้งในหัวข้อที่ 4.4

ไบท์ซิงโครนัส 10 ไบท์ ประกอบด้วยรหัสแอสกีของตัวเลข 0 ถึง 9 เพื่อกำหนดความสัมพันธ์ระหว่างสถานีผู้รับและสถานีผู้ส่งให้ทราบตำแหน่งไบท์ข้อมูลถัดไป และเข้าใจความหมายของไบท์นั้นจึงต้องมีการซิงโครนัส สาเหตุอีกประการหนึ่งที่ต้องมีการซิงโครนัสเพราะการส่งสัญญาณข้อมูลไบท์แรกๆ ออกไปในสายส่งสัญญาณ วงจรรับสัญญาณข้อมูลส่วนมากจะยังจัดความสัมพันธ์สัญญาณข้อมูลกับสัญญาณนาฬิกาไม่ได้ทำให้ข้อมูลไบท์แรกๆ อ่านได้ค่าผิด จึงต้องมีการส่งไบท์ซิงโครนัสไปก่อนจนกว่าวงจรรับจะอ่านค่าไบท์ซิงโครนัสได้ถูกต้องจึงกำหนดความหมายของไบท์ถัดไปได้

การซิงโครนัสในโปรแกรม M-WARE มีใช้อยู่ 2 วิธี วิธีแรกไบท์ซิงโครนัส 10 ไบท์ จะประกอบด้วยรหัสแอลกีค่า 0 ถึง 9 โปรแกรมในส่วนรับจะเทียบค่าที่อ่านได้ตั้งแต่ไบท์แรกไปจนถึงไบท์ที่ 10 ว่าไบท์ใดมีค่าเป็นตัวเลข 8 เมื่อพบไบท์ที่มีค่าเป็นตัวเลข 8 แล้ว ไบท์ถัดไปต้องมีค่าเป็นตัวเลข 9 เมื่อพบทั้ง 2 ไบท์ เรียงกันมีค่าเป็นตัวเลข 8 และ 9 ถือว่าวงจรรับซิงโครนัสสัญญาณข้อมูลได้ถูกต้อง ฉะนั้นไบท์ถัดไปคือไบท์ที่มีตำแหน่งเป็นไบท์แรกที่เริ่มมีความหมายในเฟรมคำสั่ง ไบท์แรกที่เริ่มมีความหมายก็คือไบท์หมายเลขสถานีผู้รับนั่นเอง พิจารณาที่รูป 36 และทำให้ทราบว่าไบท์ถัดไปคือ

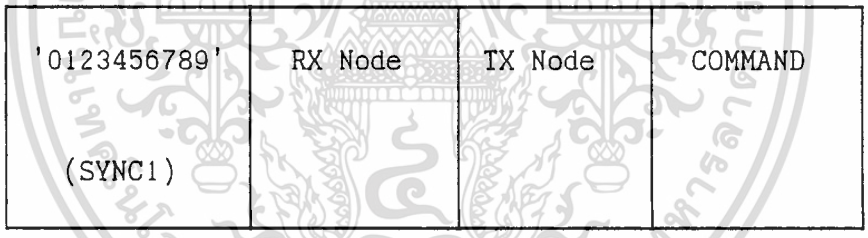
ไบท์หมายเลขสถานีผู้ส่ง และไบท์คำสั่งตามลำดับ

วิธีซิงโครนัสแบบที่สองคือ ส่งค่าแอสกีของตัวอักษร A ถึง Z จำนวน 26 ไบท์ ในขนาด 8 บิทต่อไบท์ ตามด้วยไบท์แรกที่มีความหมายในขนาด 9 บิทต่อไบท์ แล้วใช้โปรแกรมตรวจว่าเมื่อไรที่วงจรรับอ่านค่าได้ไบท์ขนาด 9 บิทต่อไบท์ ให้ถือว่าไบท์นั้นเป็นไบท์แรก วิธีที่สองมีโอกาสอ่านค่าผิดพลาดน้อยกว่าแต่จะเสียเวลาตรวจมากกว่า เพราะจำนวนไบท์ซิงโครนัสมากกว่าวิธีแรก

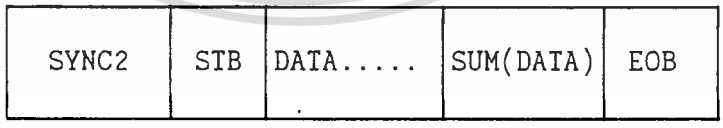
2. **เฟรมข้อมูล** เฟรมนี้ประกอบด้วยไบท์ซิงโครนัส 26 ไบท์ ขนาด 8 บิท ไบท์เริ่มบล็อกข้อมูล (STB) 1ไบท์ ขนาด 9 บิท ไบท์ข้อมูลจำนวนไม่เกิน 1 บล็อก(กำหนดค่า 1 บล็อกไว้เท่ากับ 4096 ไบท์) ไบท์ผลบวกของข้อมูล 1ไบท์ และไบท์สิ้นสุดบล็อกข้อมูล (EOB)1 ไบท์ขนาด 9 บิท

เฟรมนี้ใช้วิธีซิงโครนัสแบบที่สอง ทำให้ทราบว่าไบท์แรกคือไบท์เริ่มบล็อกข้อมูล ไบท์ถัดไปคือไบท์ข้อมูลจนไปตรวจพบไบท์ 9 บิทอีกครั้งคือไบท์สิ้นสุดบล็อกข้อมูล ดังนั้นไบท์สุดท้ายของไบท์ข้อมูลก็คือไบท์ผลบวกของข้อมูลนั้นเอง ดังรูปที่ 36

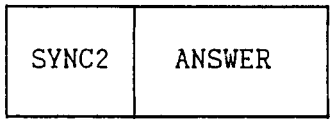
3. **เฟรมตอบกลับ** เฟรมนี้ประกอบไบท์ซิงโครนัส 26 ไบท์ขนาด 8 บิท และไบท์คำสั่งตอบกลับมีอยู่ 3 คำสั่งคือคำสั่งตอบกลับจากสถานีผู้ใช้(ANS) คำสั่งระบบผิดพลาด(ERROR) และคำสั่งเกินเวลารับคำสั่งหรือข้อมูล(TIME OUT) โดยใช้การซิงโครนัสแบบที่สอง



ก) เฟรมคำสั่ง



ข) เฟรมข้อมูล



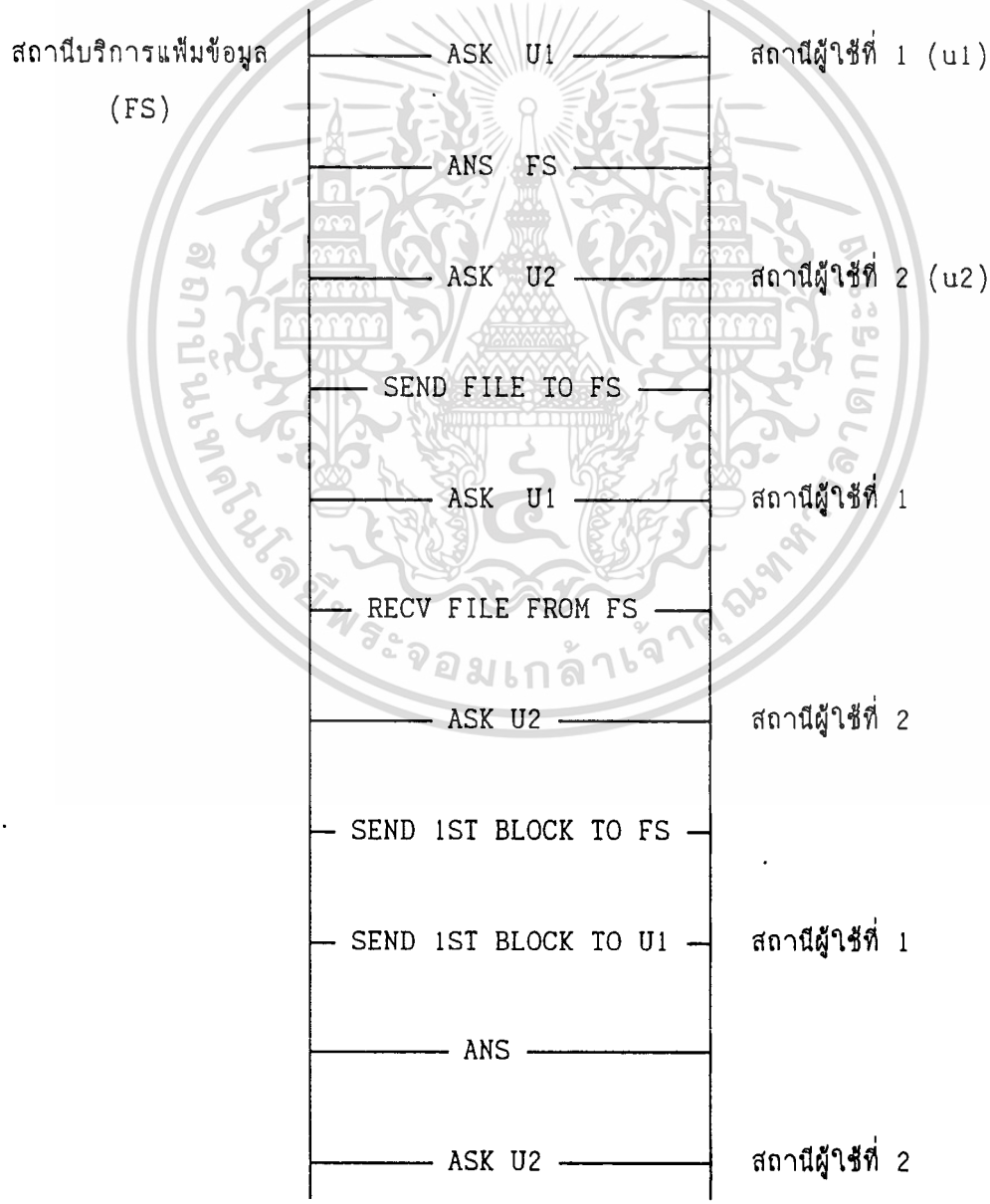
ค) เฟรมตอบกลับ

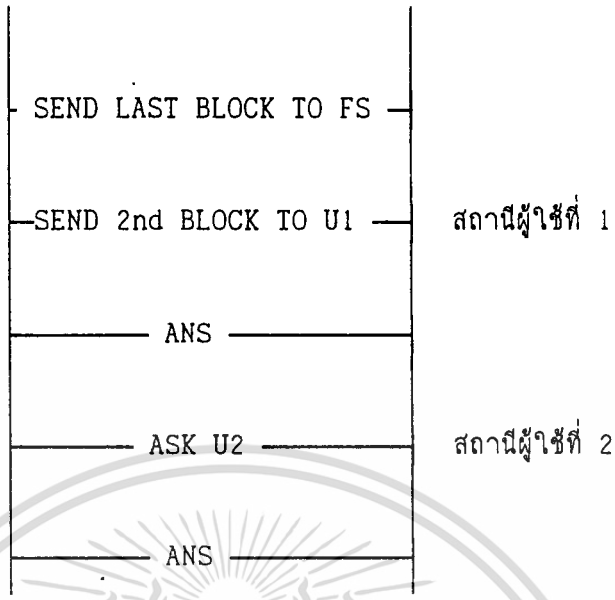
รูปที่ 36. รูปแบบของเฟรม ก) เฟรมคำสั่ง ข) เฟรมข้อมูล ค) เฟรมตอบกลับ

4.3 โปรแกรมหลักของโปรแกรม M-WARE

โปรแกรมหลักของโปรแกรม M-WARE มีขั้นตอนซับซ้อนมากการเขียนโฟลว์ชาร์ทและคำอธิบายจะเข้าใจได้ยาก จึงขอใช้การอธิบายการทำงานของโปรแกรมหลักของโปรแกรม M-WARE ด้วยโปรโตคอลชาร์ท(Protacal Chart) โปรแกรมหลักเขียนขึ้นพื้นฐานทฤษฎีการวนเรียกจากศูนย์กลาง(Centralized Polling) โดยมีสถานีแฟ้มข้อมูลเป็นศูนย์กลางในการวนถามความต้องการของสถานีผู้ใช้วิธีนี้เป็นการแบ่งเวลาในการใช้สายส่งสัญญาณ (Time Sharing) เพื่อไม่ให้เกิดการชนกันของข้อมูล

โปรโตคอลชาร์ทของสถานีบริการแฟ้มข้อมูลในการวนถามคำสั่งจากสถานีผู้ใช้





คำอธิบายโปรโตคอลชาร์ท

(1) สถานีบริการแฟ้มข้อมูล ส่งรหัสคำถาม ASK ไปถาม สถานีผู้ใช้ที่ 1

(2) สถานีผู้ใช้ที่ 1 ส่งรหัสคำตอบ ANS เพื่อรับทราบการติดต่อไปยังสถานีบริการแฟ้มข้อมูล

(3) สถานีบริการแฟ้มข้อมูลส่งรหัสคำถาม ASK ไปถามสถานีผู้ใช้ที่ 2

(4) สถานีผู้ใช้ที่ 2 ส่งคำสั่งการส่งแฟ้มข้อมูลตามด้วยชื่อแฟ้มและขนาดของแฟ้มข้อมูลให้ สถานีบริการแฟ้มข้อมูล จากนั้นเตรียมบล็อกของแฟ้มข้อมูลที่จะส่งบล็อกแรก

(5) สถานีบริการแฟ้มข้อมูล ส่งรหัสคำถาม ASK ไปตามสถานีผู้ใช้ที่ 1

(6) สถานีผู้ใช้ที่ 1 ส่งคำสั่งการขอรับแฟ้มข้อมูลตามด้วยชื่อแฟ้มจากสถานีบริการแฟ้มข้อมูล สถานีบริการจะเปิดแฟ้มข้อมูลแล้วเตรียมข้อมูลบล็อกแรกไว้ให้

(7) สถานีบริการแฟ้มข้อมูลส่งรหัสคำถาม ASK ไปถามสถานีผู้ใช้ที่ 2

(8) สถานีผู้ใช้ที่ 2 ก็จะส่งข้อมูลบล็อกแรกให้สถานีบริการแฟ้มข้อมูล

(9) สถานีบริการแฟ้มข้อมูล ส่งข้อมูลบล็อกแรกให้ สถานีผู้ใช้ที่ 1

(10) สถานีผู้ใช้ที่ 1 ตอบรับข้อมูลถูกต้อง

(11) สถานีบริการแฟ้มข้อมูล ส่งรหัสคำถาม ASK ให้สถานีผู้ใช้ที่ 2

(12) สถานีผู้ใช้ที่ 2 ส่งบล็อกสุดท้ายให้สถานีบริการแฟ้มข้อมูล สถานีบริการแฟ้มข้อมูลตรวจขนาดของแฟ้มและ EOF ก็จะเปิดแฟ้มข้อมูลนั้นยกเลิกลำดับบล็อกแฟ้มข้อมูล

(13) สถานีบริการแฟ้มข้อมูล ส่งข้อมูลบล็อกสองให้ สถานีผู้ใช้ที่ 1

(14) สถานีผู้ใช้ที่ 1 ตอบรับข้อมูลถูกต้อง

(15) สถานีบริการแฟ้มข้อมูล ส่งรหัสคำถาม ASK ให้สถานีผู้ใช้ที่ 2

(16) สถานีผู้ใช้ที่ 2 ส่งรหัสคำตอบ ANS เพื่อรับทราบการติดต่อ

ถ้าสถานีบริการแฟ้มข้อมูลวนถาม แต่สถานีผู้ใช้ไม่ได้ส่งคำสั่งตอบกลับให้ทันช่วงเวลาที่ตั้งไว้ (Time Out) ถือว่าสถานีขาดการติดต่อต้องรอการวนถามครั้งถัดไป เพื่อรับทราบการติดต่อกันใหม่

ถ้าฝ่ายรับตรวจข้อมูลที่รับได้ปรากฏว่าข้อมูลผิดพลาด ฝ่ายรับจะส่งคำสั่ง ERROR ไปยังฝ่ายส่ง ฝ่ายส่งจะส่งข้อมูลบล็อกเดิมให้ใหม่ในการวนถามครั้งถัดไป

สถานีผู้ใช้สามารถติดต่อระหว่างกันได้ โดยส่งคำสั่ง DIRECT ให้สถานีบริการแฟ้มข้อมูลทราบ เมื่อมีการวนถาม สถานีบริการแฟ้มข้อมูลจะเปิดโอกาสให้สถานีผู้ใช้ติดต่อกันไม่เกินหนึ่งบล็อกข้อมูล หรือมีคำสั่ง END ปรากฏในสายส่งสถานีบริการแฟ้มข้อมูลก็จะวนถามสถานีถัดไป

บล็อกข้อมูลคือ จำนวนข้อมูลมีหน่วยเป็นไบท์ สามารถกำหนดขนาดของบล็อกได้โดยกำหนดจำนวนไบท์ของข้อมูลในโปรแกรม M-WARE ปรกติตั้งขนาดของบล็อกเท่ากับ 4096 ไบท์

4.4 คำสั่งโปรแกรม M-WARE

ในโปรแกรม M-WARE มีคำสั่ง 2 ประเภท ประเภทแรกคือคำสั่งที่ใช้ในตัวโปรแกรม และประเภทที่สองคือคำสั่งที่ผู้ใช้พิมพ์สั่งบนเครื่องคอมพิวเตอร์ ซึ่งจะอธิบายในบทที่ 5

คำสั่งประเภทแรกเป็นคำสั่งที่จะอธิบายในส่วนนี้เพื่อจะทำให้อ่านวิทยานิพนธ์เล่มนี้เข้าใจง่ายขึ้น

คำสั่งที่ใช้ในโปรแกรม M-WARE มีดังนี้

ค่า(DEC)	สัญญาลักษณ์	ความหมาย
0	ASK	= การถามจากสถานีแฟ้มข้อมูล
1	ANS	= การตอบจากสถานีผู้ใช้
2	SEND	= การขอส่งแฟ้มข้อมูล
3	RECV	= การขอรับแฟ้มข้อมูล
4	QUIT	= การขอออกจากโปรแกรม
5	ERROR	= การแสดงว่ามีความผิดพลาด
6	END	= การจบแฟ้มข้อมูล
7	MAIL	= การส่งข่าวสาร
8	DIRECT	= การขอติดต่อตรงระหว่างสถานี
9	PRINT	= การขอติดต่อเครื่องพิมพ์
10	STATUS	= การขอตรวจสถานะการติดต่อ
11	HELP	= การขอดูเมนูการใช้คำสั่ง
12	SYS	= การเรียกระบบตัวเอง

13 REMOTE = การเรียกระบบสถานีอื่น

14 SENDNEXT = การส่งเฟรมถัดไป

15 RECVNEXT = การรับเฟรมถัดไป

16 STB = การเริ่มต้นบล็อกข้อมูล

17 EOB = การจบบล็อกข้อมูล

18 TIMEOUT = การแสดงความผิดพลาดในการใช้เวลาเกินจริง

19 DISPLAY = การสั่งให้แสดงผลบนสถานีอื่น

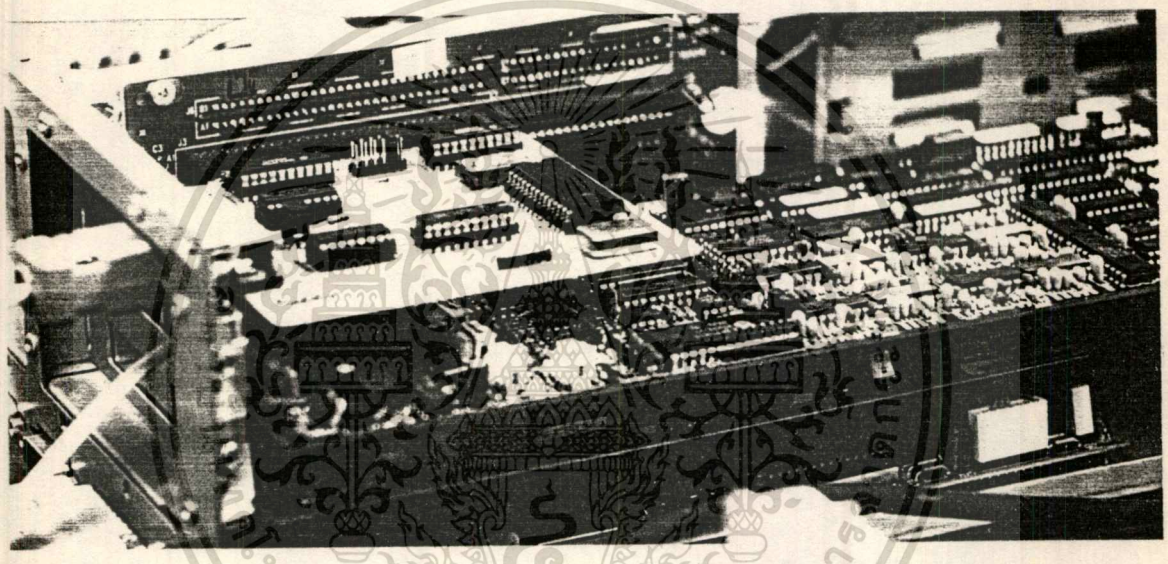
20 LOGIN = การขอเข้าโครงข่าย

21 TEST = การทดสอบส่งค่าซ้ำเมื่อดูว่าสถานีปลายทางรับค่าผิดบางหรือไม่

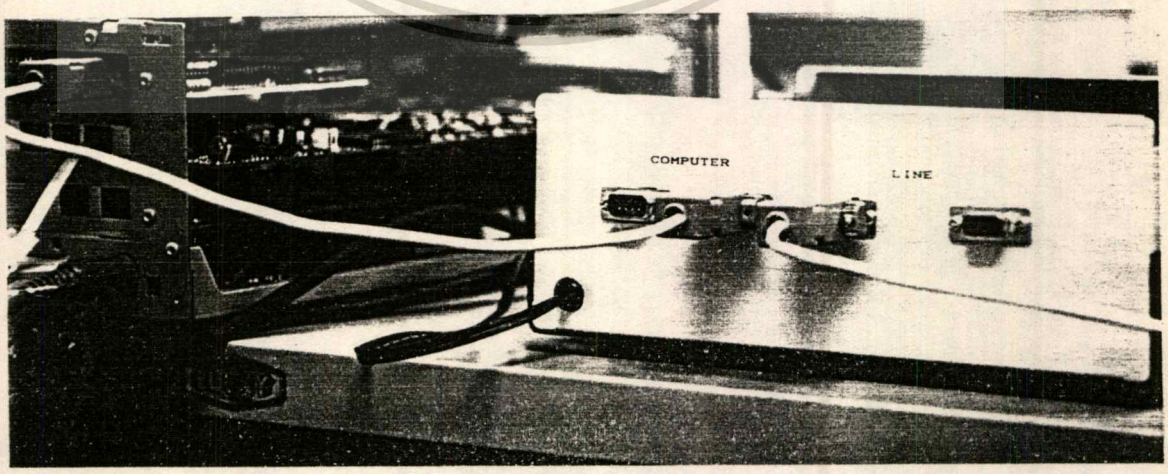
บทที่ 5
การติดตั้งและการใช้โครงข่าย M-NET

5.1. การติดตั้งสถานี

การติดตั้งเครื่องคอมพิวเตอร์ให้เป็นสถานีบนโครงข่าย M-NET จะต้องมี M-NET การ์ดเสียบที่สล็อตดังรูปที่ 37 ต่อสายเชื่อมคอนเนกเตอร์แบบ 9 ขาจาก M-NET การ์ดไปยังคอนเนกเตอร์ (COMPUTER)ของกล่องวงจรรับส่งเนกาทีฟพัลส์(NEGATIVE PULSE TRANSCEIVER) และต่อสายส่งสัญญาณเข้าที่คอนเนกเตอร์(LINE) ดังรูปที่ 38



รูปที่ 37. การติดตั้ง M-NET การ์ดบนสล็อต

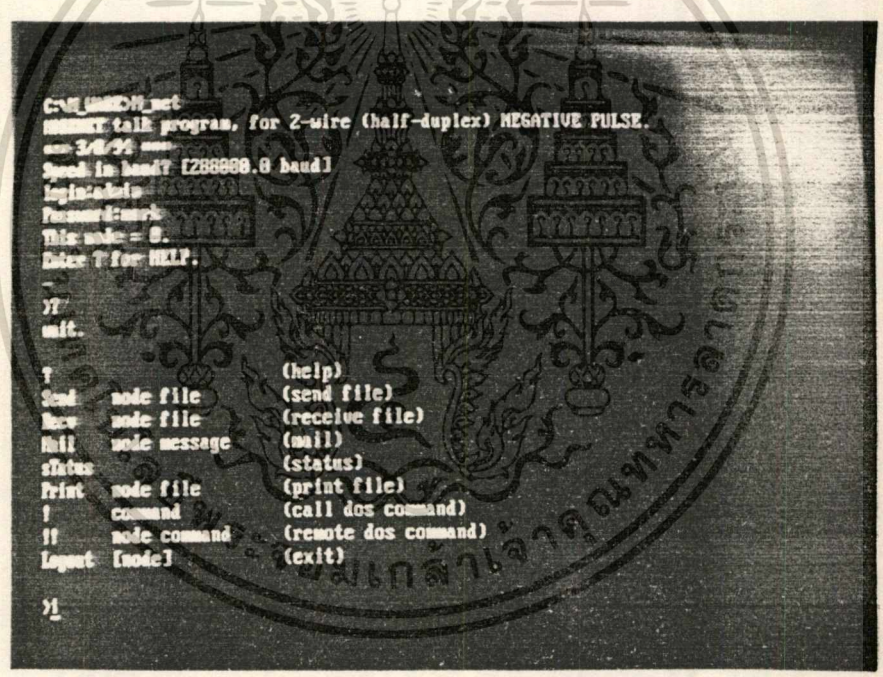


รูปที่ 38. การติดตั้งสายส่งสัญญาณ

เมื่อติดตั้งสถานีได้ครบตามจำนวนสถานีที่ต้องการแล้ว ต่อไปเป็นการติดตั้งโปรแกรม M-WARE โดยนำแผ่นดิสเก็ทที่มีโปรแกรม M-WARE ก็อปปี้โปรแกรม M-NET.EXE และแฟ้ม USER.TAB ลงฮาร์ทดิสค์ของสถานีที่กำหนดให้เป็นสถานีบริการแฟ้มข้อมูล ส่วนสถานีผู้ใช้ให้กอปปี้เฉพาะโปรแกรม M-NET.EXE เป็นการเสร็จสิ้นการติดตั้งโครงข่าย M-NET ต่อไปเป็นการเปิดใช้โครงข่าย M-NET

5.2 การเปิดใช้โครงข่าย M-NET

การเริ่มโครงข่าย M-NET จะต้องเปิดใช้สถานีแฟ้มข้อมูลก่อนสถานีผู้ใช้เสมอ โดยรันโปรแกรม M-NET.EXE โปรแกรมจะแสดงข้อความบนจอภาพ ให้กำหนดอัตราบอดสูงสุดคือ 288K กดปุ่ม ENTER โปรแกรมจะถามว่าท่านคือโหนดใดป้อนชื่อโหนด(เท่ากับชื่อสถานี) ถ้าเป็นสถานีแฟ้มข้อมูลให้พิมพ์คำว่า ADMIN กดปุ่ม ENTER โปรแกรมจะถามรหัสผ่านของท่านให้พิมพ์คำว่า MORK กดปุ่ม ENTER โปรแกรมจะเริ่มทำการวนถามหรือรับคำสั่งจากแป้นพิมพ์ต่อไป ดังรูปที่ 39



รูปที่ 39. แสดงจอภาพขณะรันโปรแกรม M-NET.EXE

สถานีผู้ใช้ก็ให้ติดตั้งโปรแกรมเดียวกันแต่การพิมพ์ชื่อโหนดและรหัสผ่าน ต้องให้ตรงกับชื่อโหนดและรหัสผ่านในแฟ้ม USER.TAB ดังนั้นในฮาร์ดดิสท์ของสถานบริการแฟ้มข้อมูลต้องมีแฟ้มนี้อยู่ตลอดเวลา สถานบริการแฟ้มข้อมูลต้องตรวจสอบชื่อโหนดและรหัสผ่านของสถานีผู้ใช้ทุกครั้งที่เข้าโครงข่ายเสมอ เป็นการป้องกันไม่ให้บุคคลที่ไม่มีสิทธิ์เข้าโครงข่ายได้

```
ADMIN mork
AAA aaa
BBB bbb
CCC ccc
DDD ddd
EEE eee
FFF fff
```

รูปที่ 40. แฟ้ม USER.TAB

ชื่อโหนดหรือรหัสผ่านสามารถแก้ไขเพิ่มเติมได้โดยใช้โปรแกรม Text Editer ทั่วไป ตัวอย่างการเปลี่ยนชื่อโหนดและรหัสผ่านจากบรรทัดที่ 2 ในรูปที่ 40

```
AAA aaa
```

ความหมาย AAA คือชื่อโหนด aaa คือรหัสผ่านถ้าต้องการเปลี่ยนชื่อโหนดเป็น WERA และรหัสผ่านเป็น MYSON ก็ให้แก้บรรทัดที่ 2 เป็น

```
WERA MYSON
```

แล้วบันทึกลงแฟ้ม USER.TAB คราวต่อไปในการเข้าโครงข่ายอีกต้องใช้ชื่อโหนดและรหัสผ่านที่แก้ไขแล้วเท่านั้น

5.3 การใช้โครงข่าย M-NET

เมื่อสถานีบริการแฟ้มข้อมูลเปิดบริการโครงข่ายและสถานีผู้ใช้ได้เข้าระบบโครงข่ายแล้ว ต่อไปเป็นการใช้โครงข่าย M-NET การใช้โครงข่ายก็คือการพิมพ์คำสั่งในการติดต่อระหว่างสถานีหรือใช้คำสั่งเข้าออกโครงข่าย

5.3.1 การใช้คำสั่ง HELP

คำสั่งนี้แสดงรูปแบบการพิมพ์คำสั่งและพารามิเตอร์ของทุกคำสั่งที่ใช้ในโครงข่าย M-NET ให้ปรากฏบนจอภาพ เพื่อช่วยให้ผู้ใช้ไม่จำเป็นต้องจำคำสั่งทั้งหมด หรือช่วยผู้ใช้ที่จำรูปแบบและพารามิเตอร์ของคำสั่งไม่ได้ การเรียกคำสั่งนี้ทำได้สองรูปแบบ

รูปแบบคำสั่ง >HELP

หรือ >?

ผลของคำสั่งนี้จะปรากฏบนจอภาพดังรูปที่ 41

รูปที่ 41. แสดงจอภาพขณะใช้คำสั่ง HELP

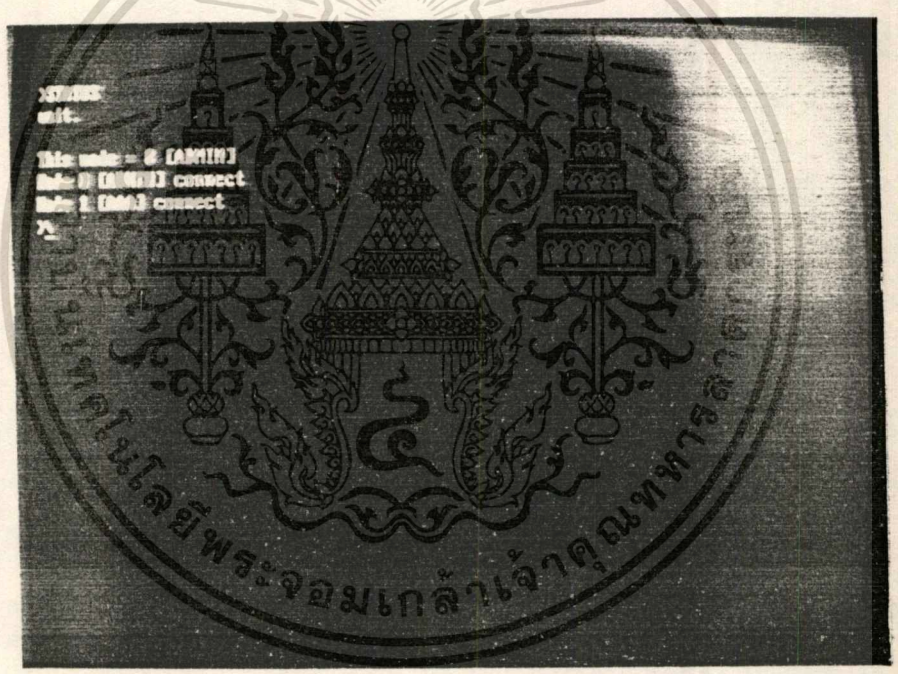
5.3.2 การใช้คำสั่ง STATUS

คำสั่งนี้ใช้เมื่อต้องการทราบว่าสถานีใด LOGIN เข้ามาในโครงข่ายโดยผลของคำสั่งจะแสดงชื่อโหนด(ชื่อสถานี)ตามด้วยคำว่า CONNECT บนจอภาพดังรูปที่ 42

รูปแบบคำสั่ง >STATUS

หรือ >T

เพื่อให้ทราบว่าสถานีนั้น LOGIN เข้ามาและยังติดต่อโครงข่ายอยู่



รูปที่ 42. แสดงจอภาพขณะใช้คำสั่ง STATUS

5.3.3 การใช้คำสั่ง MAIL

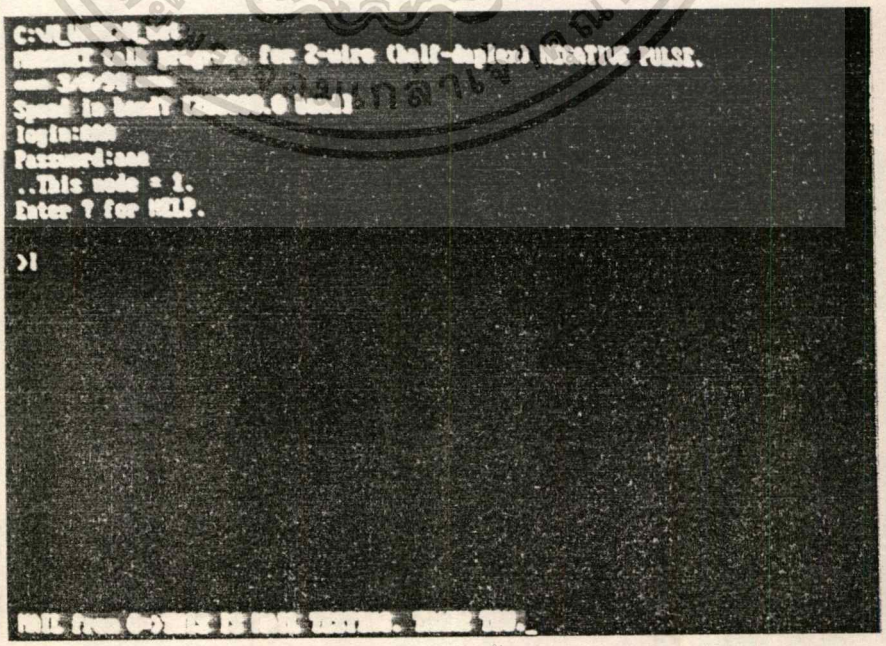
คำสั่งนี้ใช้ส่งข้อความไปปรากฏตรงบรรทัดล่างของจอภาพของสถานีที่ต้องการติดต่อ เพื่อใช้ติดต่อกันระหว่างสถานีด้วยข้อความที่ยาวไม่เกินหนึ่งบรรทัด 80 ตัวอักษรซึ่งนิยมใช้โต้ตอบข้อความกัน

รูปแบบคำสั่ง >MAIL [NODE][MESSAGE < 80 CHAR]

>M [NODE][MESSAGE < 80 CHAR]

[NODE] คือหมายเลขของสถานีที่ต้องการให้ข้อความไปปรากฏตัวอย่างการใช้คำสั่ง MAIL ดังรูปที่ 43 และการแสดงข้อความตรงบรรทัดล่างของจอภาพดังรูปที่ 44

รูปที่ 43. แสดงจอภาพของผู้ใช้คำสั่ง MAIL



รูปที่ 44. แสดงจอภาพของผู้รับคำสั่ง MAIL

5.3.4 คำสั่ง SEND และ RECV

คำสั่งนี้ใช้ส่งและขอแฟ้มข้อมูลระหว่างสถานี

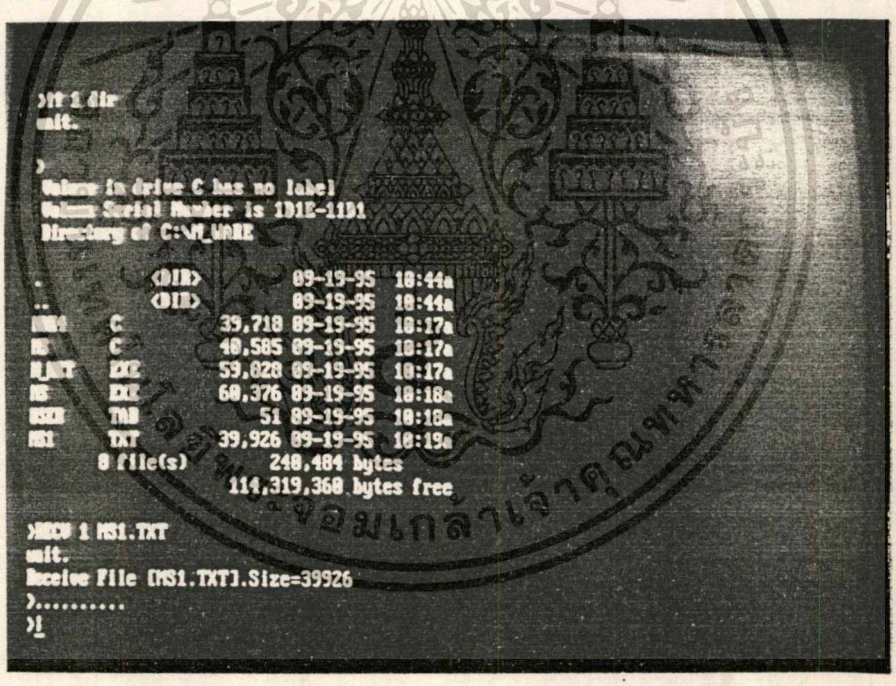
รูปแบบคำสั่ง >SEND [NODE] [PATH:FILENAME]

หรือ >S [NODE] [PATH:FILE NAME]

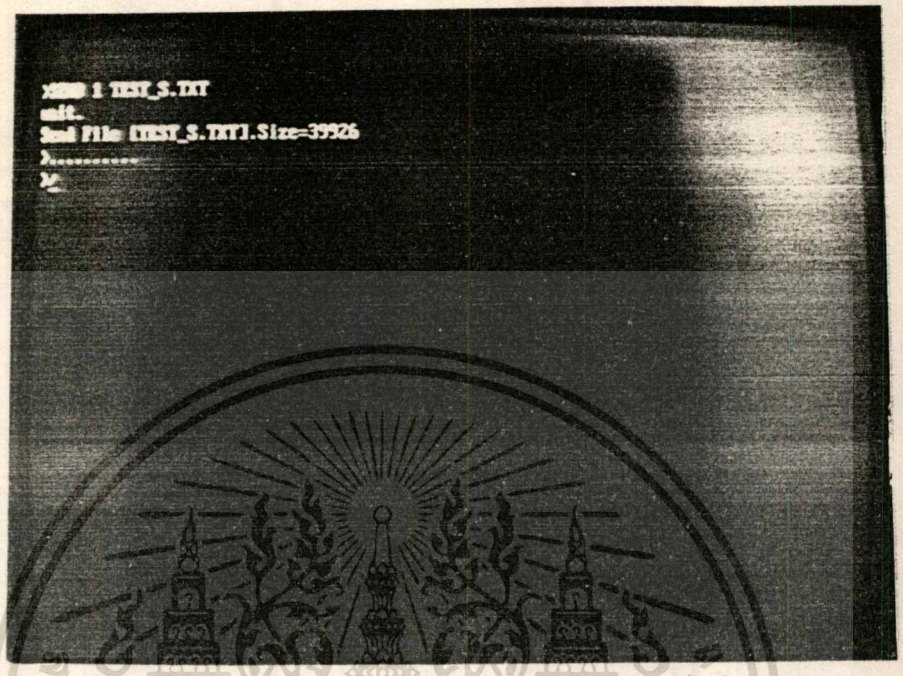
รูปแบบคำสั่ง >RECV [NODE] [PATH:FILENAME]

หรือ >R [NODE] [PATH:FILENAME]

การส่งหรือขอแฟ้มข้อมูลนั้น แฟ้มข้อมูลจะถูกแบ่งเป็นบล็อกข้อมูลการส่งหรือรับหนึ่งบล็อกข้อมูลจะปรากฏจุด "." บนจอภาพหนึ่งจุด เมื่อได้แฟ้มข้อมูลจนครบทุกบล็อกก็จะปรากฏ > ดังรูปที่ 45 และ รูปที่ 46



รูปที่ 45. แสดงจอภาพขณะใช้คำสั่ง RECV



รูปที่ 46. แสดงจอภาพขณะใช้คำสั่ง SEND

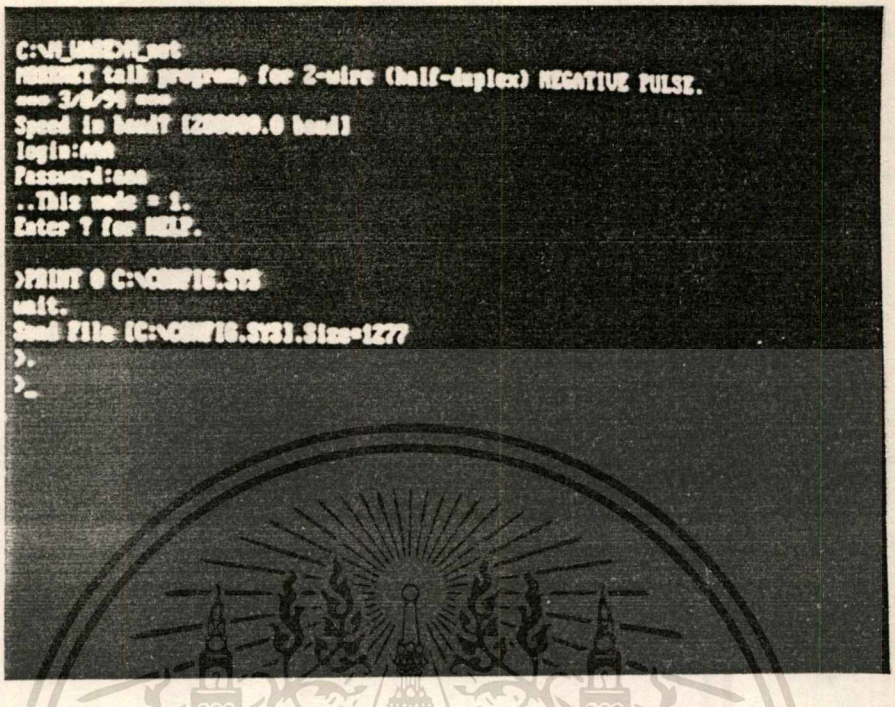
5.3.5 คำสั่ง PRINT

คำสั่งนี้ใช้พิมพ์แฟ้มข้อมูลออกทางเครื่องพิมพ์ที่ติดตั้งอยู่กับสถานีผู้ใช้สถานีอื่น คำสั่งนี้เป็นการแสดงให้เห็นถึงการใช้ทรัพยากรร่วมกัน ซึ่งเป็นจุดประสงค์หนึ่งของระบบโครงข่าย M-NET

รูปแบบคำสั่ง >PRINT [NODE] [PATH:FILE NAME]
หรือ >p [NODE] [PATH:FILE NAME]

[NODE] คือหมายเลขของสถานีที่ติดตั้งเครื่องพิมพ์และต้องการให้พิมพ์ข้อมูล

ข้อจำกัดคือเครื่องพิมพ์ควรใช้กระดาษต่อเนื่องและเปิดเครื่องไว้พร้อมพิมพ์ต่อเวลา ตัวอย่างใช้คำสั่งดังรูปที่ 47

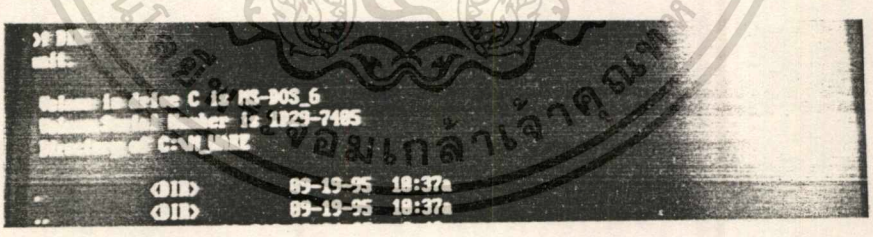


รูปที่ 47. แสดงจอภาพขณะใช้คำสั่ง PRINT

5.2.6 การใช้คำสั่ง DOS

การใช้คำสั่ง DOS บนสถานีของผู้ใช้เองขณะยังอยู่ในระบบโครงข่ายมีรูปแบบคำสั่งดังนี้

รูปแบบคำสั่ง >! [DOS COMMAND]

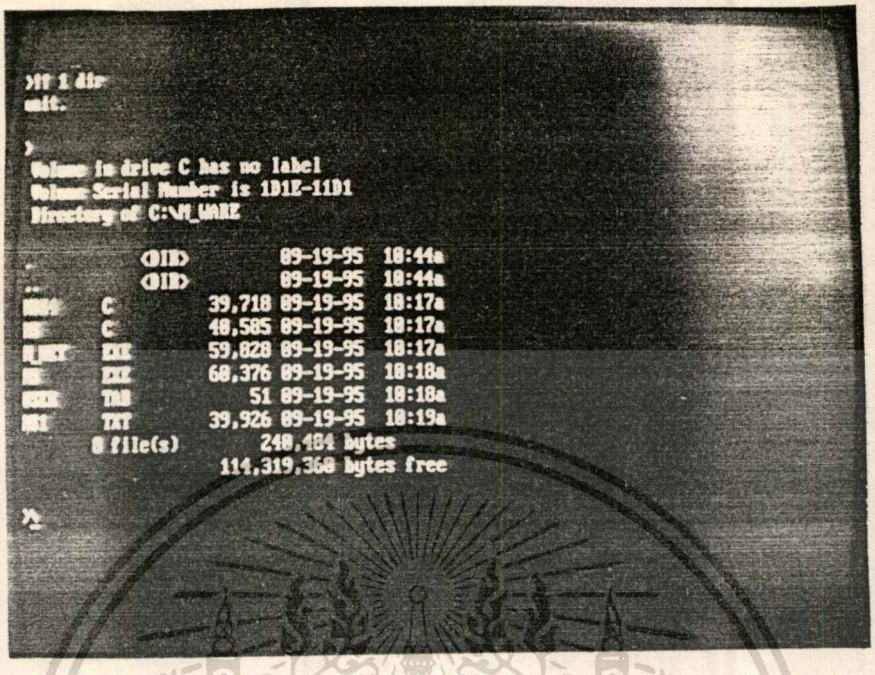


รูปที่ 48. แสดงจอภาพขณะใช้คำสั่ง !

การใช้คำสั่ง DOS บนสถานีของผู้อื่น เพื่อเป็นการรีโมทไม่ใช้คำสั่งดอสบนเครื่องของสถานีอื่น แต่ให้ส่งภาพมาปรากฏบนจอภาพของสถานีผู้ใช้คำสั่งนี้

ตัวอย่างการขอดูแฟ้มไดเร็กทอรีของสถานีผู้ใช้ที่ 1 พิจารณาจากรูปที่ 49

รูปแบบคำสั่ง >!! [NODE] [DOS COMMAND]



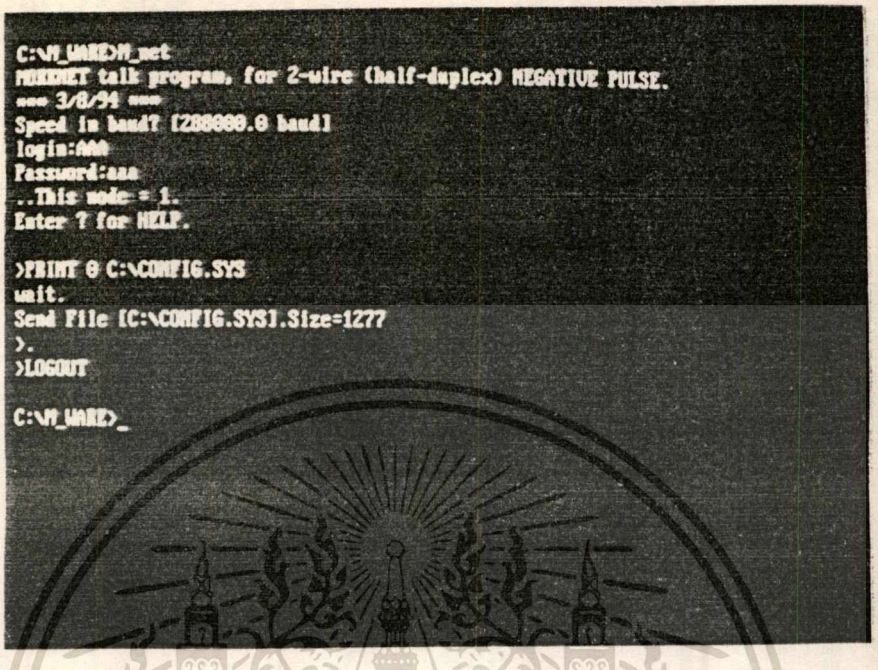
รูปที่ 49. แสดงจอภาพขณะใช้คำสั่ง !!

ข้อจำกัด DOS COMMAND ต้องเป็นคำสั่งดอสที่คำเนินการเสร็จภายในการติดต่อครั้งเดียว ตัวอย่างเช่นคำสั่งดอส DIR ใช้ได้แต่ DIR/P ที่มีชื่อแฟ้มข้อมูลเกิดหนึ่งหน้าจอภาพใช้ไม่ได้ เพราะคำสั่งดอสนี้จะมีคำสั่งให้ทำต่อเช่นกดปุ่มใดๆเพื่อดูหน้าถัดไป ซึ่งลักษณะนี้คือคำสั่งดอสที่ไม่เสร็จสิ้นในการติดต่อครั้งเดียว คำสั่งลักษณะนี้จะใช้ไม่ได้

5.3.7 คำสั่ง LOGOUT

คำสั่งนี้ใช้เมื่อผู้ใช้ต้องการออกจากโครงข่าย M-NET อาจจะเพื่อหยุดใช้สถานีหรือออกจากโครงข่ายเท่านั้นแต่ยังใช้คอมพิวเตอร์เครื่องนั้นอยู่เพื่อใช้เครื่องแบบส่วนตัว

รูปแบบคำสั่ง >LOGOUT



รูปที่ 50. แสดงจอภาพขณะใช้คำสั่ง LOGOUT

5.4 รูปแบบการแจ้งข้อผิดพลาด

- Time out error ใช้แจ้งเมื่อสถานีหรือเครื่องพิมพ์ไม่ได้เปิดเครื่อง ไม่ได้ติดตั้งหรือไม่สามารถติดต่อได้
- Can't open file ใช้แจ้งเมื่อมีการรับส่งหรือพิมพ์แฟ้มข้อมูลแต่ไม่สามารถเปิดแฟ้มข้อมูลได้
- Printer not ready ใช้แจ้งเมื่อเครื่องพิมพ์ไม่พร้อมหรือถูกใช้งานอยู่

บทที่ 6

การทดสอบอุปกรณ์โครงข่าย M-NET และเปรียบเทียบผล

6.1 โปรแกรมทดสอบ

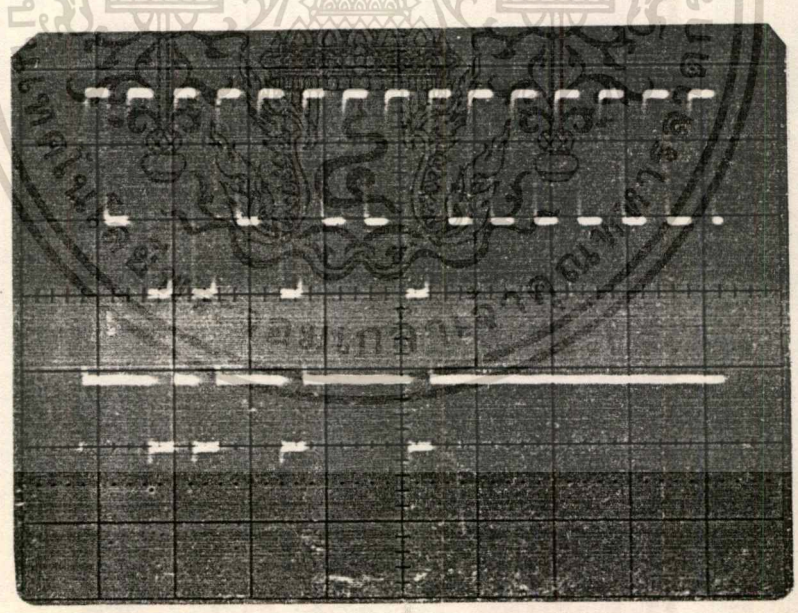
โปรแกรมในการทดสอบ TESINET มีสองโปรแกรมย่อยดังนี้

1. โปรแกรมส่งข้อมูลแอสกีที่ได้จากการป้อนเข้าทางแป้นพิมพ์ และวนส่งซ้ำไปจนกว่าจะกดปุ่มใดๆบนแป้นพิมพ์เพื่อยกเลิกการส่ง หรือตั้งจำนวนข้อมูลที่ต้องการส่ง

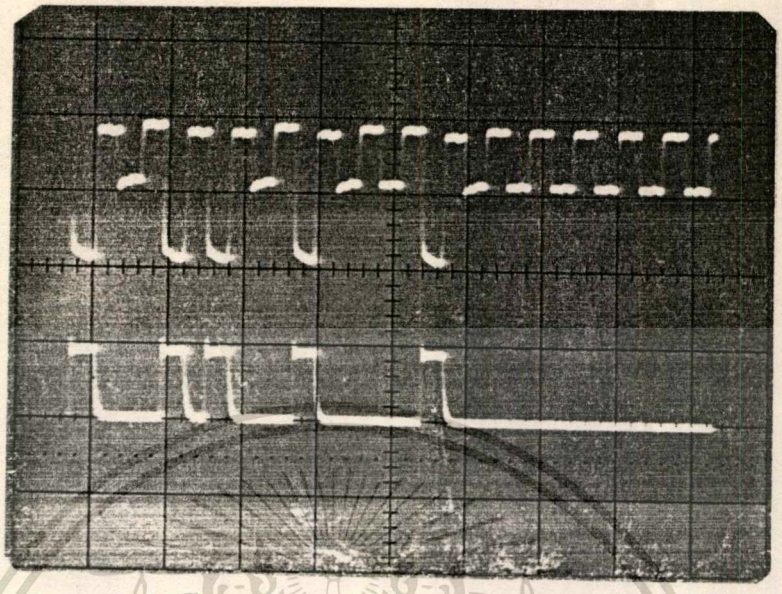
2. โปรแกรมรับก็จะรับข้อมูลแล้วตรวจว่ามีข้อมูลที่ส่งมาถูกต้องหรือไม่ ถ้ามีไบท์ผิดก็จะนับเป็นจำนวนไบท์ผิดไว้

การกำหนดตัวแปรระยะทางมีการกำหนดความยาวของสายส่งสัญญาณเป็นช่วงที่ 200, 500 และ 1000 เมตร

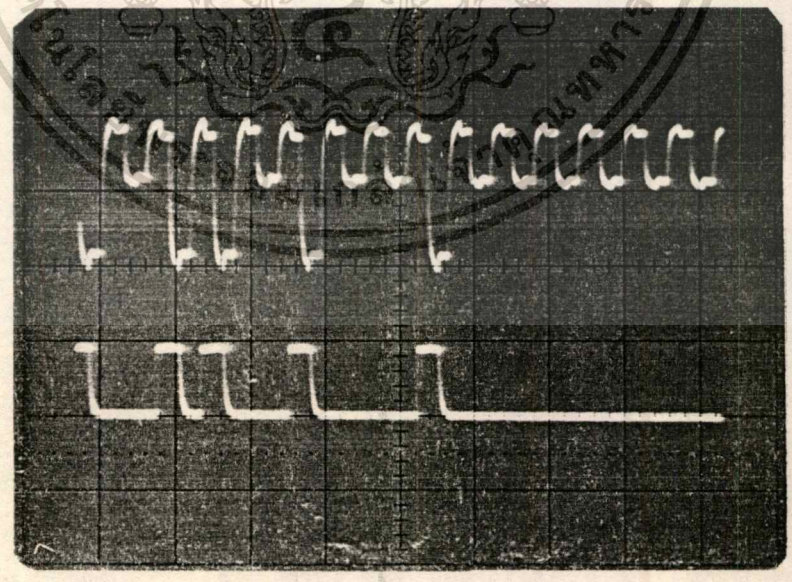
การกำหนดตัวแปรความเร็วมีการกำหนดอัตราส่งข้อมูล 72000, 144000 และ 288000 บิทต่อวินาที



ก) 200 เมตร



ข) 500 เมตร



ค) 1000 เมตร

รูปที่ 51 สัญญาณที่วัดได้ที่สายส่งความยาวต่างกัน

6.2 ผลการทดสอบการรับส่งสัญญาณโดยใช้เทคนิคเนกาทีฟพัลส์

6.2.1 การกำหนดระยะทางเป็นตัวแปร

ผลของการทดสอบด้วยอัตราส่งข้อมูลสูงสุดที่ 288K โดยใช้สายโทรศัพท์ยาว 200, 500 และ 1000 เมตร ได้ผลของรูปสัญญาณดังรูปที่ 6.1 และไม่มีจำนวนไบท์ผิดในทุกระยะของความยาวสายที่ทดสอบดังตารางที่ 6.1

6.2.2 การกำหนดความเร็วเป็นตัวแปร

ผลของการทดสอบระยะทางด้วยสายโทรศัพท์ความยาว 1000 เมตร ส่งข้อมูลด้วยอัตรา 72000, 144000 และ 288000 บิทต่อวินาที ผลปรากฏไม่มีจำนวนไบท์ผิดดังตารางที่ 6.1

6.3 ผลการทดสอบการรับส่งสัญญาณโดยใช้มาตรฐาน RS-485

6.3.1 การกำหนดระยะทางเป็นตัวแปร

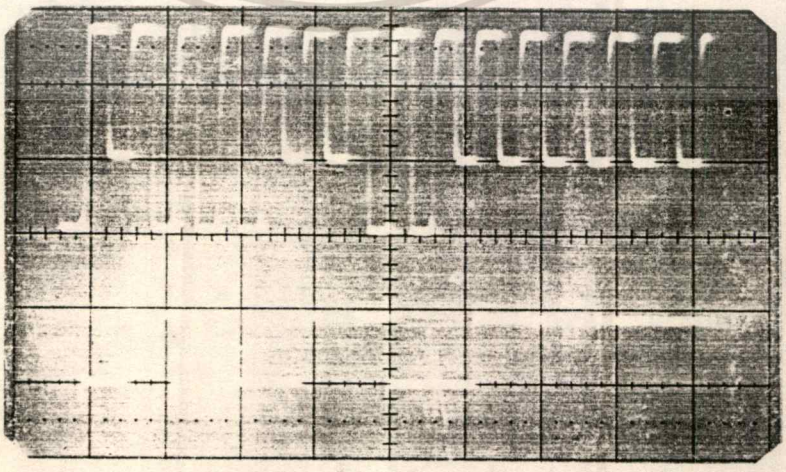
ผลการทดสอบด้วยอัตราส่งข้อมูลสูงสุดที่ 288000 บิทต่อวินาทีโดยใช้สายโทรศัพท์เป็นสายส่งที่ความยาวแตกต่างกันดังนี้ 200, 500 และ 1000 เมตร ผลปรากฏว่าเริ่มมีจำนวนไบท์ผิดที่ระยะ 500 เมตร และที่ระยะ 1000 เมตรผลปรากฏว่าไม่สามารถรับข้อมูลได้เลยดังตารางที่ 6.2

6.3.2 การกำหนดความเร็วเป็นตัวแปร

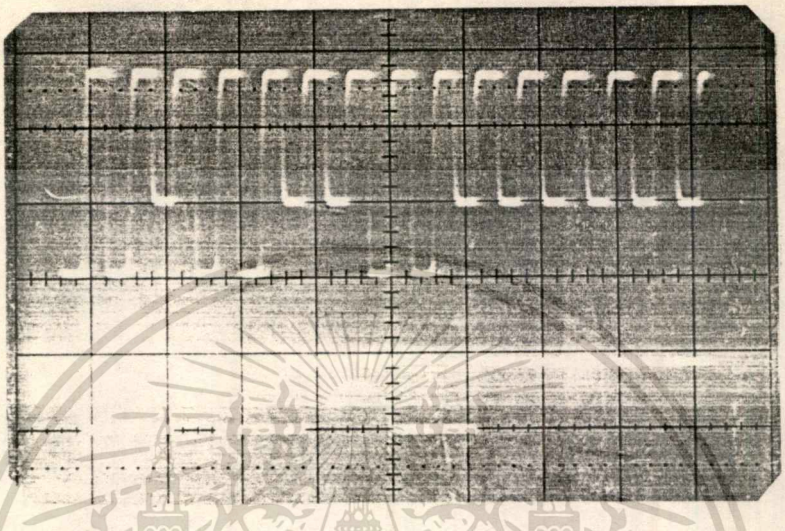
ผลการทดสอบด้วยการกำหนดระยะทางด้วยสายโทรศัพท์ความยาว 1000 เมตร รับส่งข้อมูลด้วยอัตราส่งข้อมูล 72000, 144000 และ 288000 บิทต่อวินาที ผลปรากฏว่าเริ่มมีจำนวนไบท์ผิดที่อัตราส่งข้อมูล 144000 บิทต่อวินาที และที่อัตราส่งข้อมูล 288000 บิทต่อวินาที ผลปรากฏว่าไม่สามารถรับข้อมูลได้เลยดังตารางที่ 6.2

6.3.3 การอ่านข้อมูลจากสัญญาณในสายส่ง

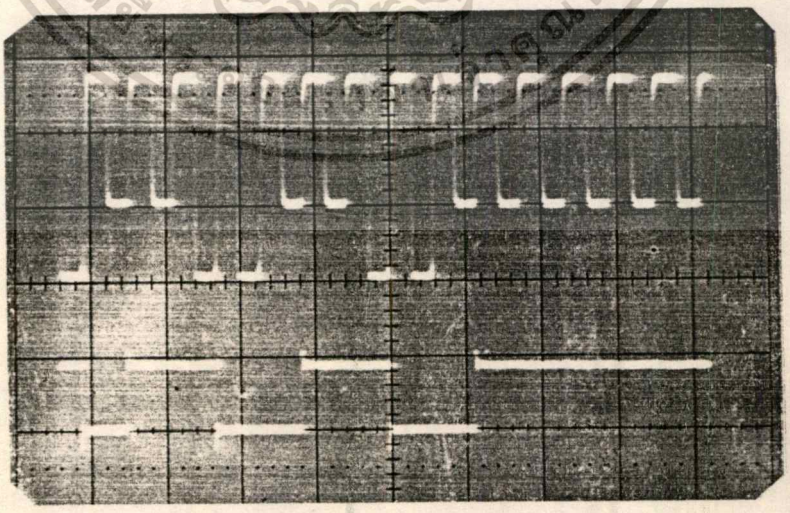
ทดสอบส่งตัวอักษรด้วยค่ารหัสแอสกี แล้วใช้ออสซิเลสโคปวัดสัญญาณตัวอักษรที่ทดสอบ A=41, B=42, c=43, 1=31, 2=32 และ 3=33 ภาพที่วัดได้จากออสซิเลสโคปเป็นดังรูป 6.2



ก.ตัวอักษร A



ข.ตัวอักษร B



ค.ตัวอักษร C

รูปที่ 52 ภาพสัญญาณข้อมูลในสายส่ง

ตารางที่ 2 ตารางแสดงจำนวนไบท์ที่อ่านผิดในการรับส่งข้อมูลด้วย เทคนิค เนกาทีฟพัลส์

ทดสอบด้วยโปรแกรม IESINET กำหนดค่าของตัวแปรตามตาราง ได้ผลลัพธ์แสดงจำนวนไบท์ที่อ่านผิด(หน่วยเป็นไบท์)ในการส่งข้อมูลต่อเนื่องจำนวน 10 เมกกะไบท์

ความยาวสายส่ง(เมตร) อัตราส่งข้อมูล(บิทต่อวินาที)	200	500	1000
72000	0	0	0
144000	0	0	0
288000	0	0	0

ตารางที่ 3 ตารางแสดงจำนวนไบท์ที่อ่านผิดในการรับส่งข้อมูลด้วยอุปกรณ์มาตรฐาน RS-485

ทดสอบด้วยโปรแกรม IESINET กำหนดค่าของตัวแปรตามตาราง ได้ผลลัพธ์แสดงจำนวนไบท์ที่อ่านผิด(หน่วยเป็นไบท์)ในการส่งข้อมูลต่อเนื่องจำนวน 10 เมกกะไบท์

ความยาวสายส่ง(เมตร) อัตราส่งข้อมูล(บิทต่อวินาที)	200	500	1000
72000	0	0	0
144000	0	0	0
288000	0	1727	**อ่านข้อมูลไม่ได้**

6.4 เปรียบเทียบผลการทดสอบระหว่างเทคนิคเนกาทีฟพัลส์(NPT) กับ RS-485

การที่เลือกเปรียบเทียบกับมาตรฐาน RS-485 เพราะมีโครงสร้างในการเชื่อมโครงข่ายแบบเดียวกัน คือการเชื่อมโครงข่ายแบบหลายจุด(Multipoints) ส่วนมาตรฐานอื่นไม่มีโครงสร้างลักษณะนี้ จึงไม่ได้นำมาเปรียบเทียบ

6.4.1 ผลการเปรียบเทียบด้านความเร็ว

การทดลองรับส่งข้อมูลแสดงให้เห็นว่า การใช้เทคนิคเนกาทีฟพัลส์รับส่งข้อมูลเทียบกับมาตรฐาน RS-485 ที่ระยะทาง 500 เมตรและ 1000 เมตร จะเห็นว่าการรับส่งข้อมูลด้วยการใช้เทคนิคเนกาทีฟพัลส์รับส่งข้อมูลด้วยอัตราส่งข้อมูล(บิทต่อวินาที)ที่สูงกว่ามาตรฐาน RS-485

6.4.2 ผลการเปรียบเทียบด้านระยะทาง

การทดลองที่อัตราส่งข้อมูล 144000 และ 288000 บิทต่อวินาที จะเห็นว่าการรับส่งข้อมูลด้วยการใช้เทคนิคเนกาทีฟพัลส์รับส่งข้อมูลได้ระยะทางที่ไกลกว่ามาตรฐาน RS-485

6.5 การเทียบคุณสมบัติโครงข่าย M-NET กับโมเดล OSI

DATA LINK LAYER and APPLICATION LAYER

M-NET	-	Character Format	Startbit, 8 bit
		Command Format	Startbit, 9 bit
		Access Control Method	Centralize polling
		Protocal	M-NET
		Error Detection	Check summing of data field
		Error Correction	Retransmission
		Message Lenght	1-4096 Bytes
		OSI Layers	Data link and application
		Network Nodes	Upto 256 nodes

PHYSICAL LAYER

NPT	-	Bus Topology	Multidrop
		Cable	Twisted pair
		Cable Length	Up to 1000 metres
		Max Speed at Max Lenght	288K baud rate
		Network Nodes	Up to 256 nodes

PHYSICAL LAYER

RS-485	-	Bus Topology	Multidrop
		Cable	Twisted pair
		Cable Lenght	Up to 1200 metres
		Max Speed at Max Lenght	100K baud rate
		Network Nodes	Up to 32 nodes

บทที่ 7

สรุปผลการวิจัยและข้อเสนอแนะ

7.1 สรุปและวิจารณ์ผลการทดลองในเชิงทฤษฎี

ในหลักการรับส่งข้อมูลโดยใช้เทคนิคเนกาทีฟพัลส์จะมีการส่งสัญญาณชิงโคนัสไปพร้อมกับข้อมูลเสมอ ดังรูปที่ 6.1 และ 6.2 ดังนั้นการดึงข้อมูลกลับออกมาจากสายส่งสัญญาณจะอาศัยจังหวะเวลา(Timing)ของสัญญาณชิงโคนัส ซึ่งจะทำให้เกิดความถูกต้องของข้อมูลมากกว่า เมื่อเปรียบเทียบกับการส่งข้อมูลตามมาตรฐาน RS-485 เพราะในการส่งข้อมูลตามมาตรฐาน RS-485 นั้นมีเฉพาะสัญญาณข้อมูลเท่านั้น ดังนั้นเมื่อมีการส่งข้อมูลโดยใช้อัตราส่งสูงเช่น 144000 และ 288000 บิทต่อวินาทีไปในสายส่งระยะไกลเช่น 1000 เมตร จะทำให้รูปของสัญญาณข้อมูลในสายส่งสัญญาณที่ปลายทางจะมีการผิดเพี้ยนไปในส่วนของช่วงเวลาขาขึ้นและขาลง(Rise Time และ Fall Time) ทำให้การดึงข้อมูลกลับจากสายส่งสัญญาณนั้นเกิดการผิดพลาดในขนาดของพัลส์ข้อมูล จึงเป็นเหตุให้เกิดข้อมูลผิดพลาดดังผลการทดลองในตารางที่ 6.2

7.2 ข้อเสนอแนะที่ควรพัฒนาต่อไป

7.2.1 ความเร็ว(Baudrate)

อัตราส่งข้อมูลในการรับส่งข้อมูล ควรมีการพัฒนาให้มีค่าอัตราส่งข้อมูลสูงขึ้นเพื่อให้การรับส่งข่าวสารได้เร็ว และให้ทันกับการพัฒนาความเร็วของเครื่องคอมพิวเตอร์ การใช้อุปกรณ์แปลงสัญญาณขนานเป็นอนุกรมที่มีประสิทธิภาพสูง หรือการเลือกใช้ตัวกลางที่มีความกว้างแถบสูงก็เป็นทางหนึ่งในการพัฒนา ถึงจุดนี้ก็ควรพิจารณาควบคู่ไปกับราคาของอุปกรณ์ และตัวกลางที่ใช้รับส่งสัญญาณ

7.2.2 ระยะทาง(Distance)

การเพิ่มระยะทางก็เป็นจุดที่ควรพัฒนา การเลือกใช้ตัวกลางในการรับส่งสัญญาณข้อมูล การรับส่งผ่านระบบสายโทรศัพท์ก็เป็นการสื่อสารระยะทางไกล โดยไม่ต้องลงทุนลงแรงกับการติดตั้งสาย แต่ก็ควรพิจารณาเรื่องค่าบริการ

7.2.3 ความเป็นมาตรฐาน(Standard)

ขณะนี้โปรแกรมสำเร็จรูปเริ่มมีบทบาทในระบบจัดการข้อมูลมากขึ้น เช่น โปรแกรมวินโดวส์ ซึ่งมีระบบจัดการเกี่ยวกับระบบการสื่อสารในรูปของโครงข่ายเช่น โปรแกรมวินโดวส์สำหรับเวอร์คกรุ๊ป(Windows for Workgroup) หรือโปรแกรมวินโดวส์เทคนิคใหม่(Windows NT) ที่คาดว่าจะเป็นที่นิยมในอนาคตอันใกล้ ซึ่งมีระบบจัดการโครงข่ายที่มีประสิทธิภาพสูง และมีความอ่อนตัวให้ผู้ใช้มากจนผู้ใช้สามารถเขียนโปรแกรมขับอุปกรณ์ได้เอง(Device Driver) ผู้ทำวิจัยได้เริ่มศึกษาและพยายามเขียนโปรแกรมขึ้นสำหรับ M-NET card ที่ใช้เทคนิคเนกาทีฟพัลส์ให้ใช้ร่วมกับโปรแกรมวินโดวส์อยู่คาดว่าจะมีผลงานในช่วงต่อไป

บรรณานุกรม

[1] Ahuja, Vijay, "Design and Analysis of Computer Communication Networks", McGraw-Hill,Inc., 1985.

[2] Jesty, P.H. "Networking with Microcomputers", Blackwell Scientific Publications, 1985.

[3] Kuo, Franklin F.,"Protocols and Techniques for Data Communication Networks", Prentice-Hall, Inc., 1981.

[4] Kleinrock, Leonard, "Queueing Systems", Volume I, John Wiley & Sons, Inc., 1975

[5] Schidt, Herbert," C Power User's Guide", McGraw-Hill, Inc., 1988.

[6] Tanenbaum, Andrew S., "Computer Networks", Prentice-Hall, Inc., 1988.

[7] Williams, Arthur B., "Designer's Handbook of Integrated Circuits "McGraw-Hill,Inc., 1984.

[8] "IBM PC AT Technical Reference", IBM Corp., 1984

[9] "Interface Databook", National Semiconductor, 1990.

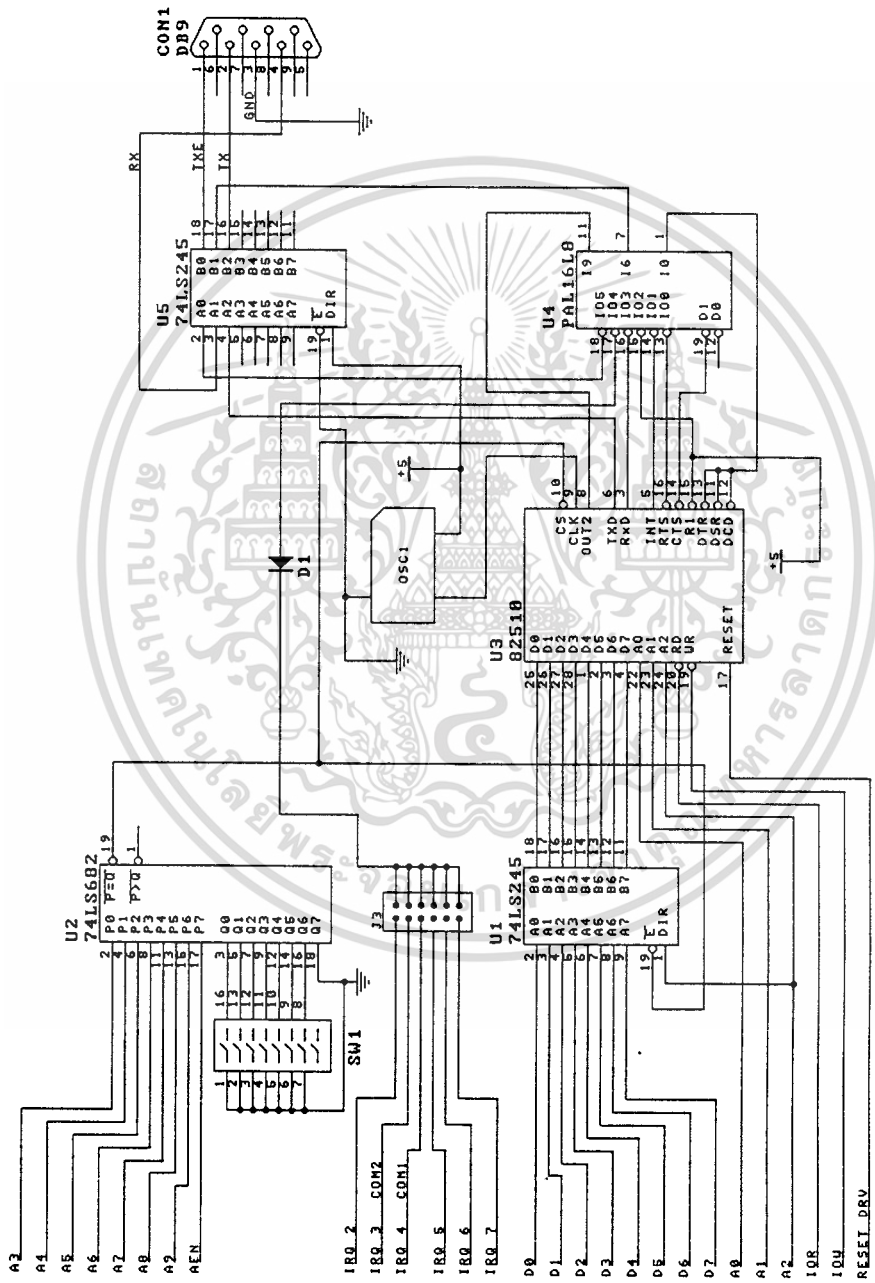
[10] วีระ ธรรมจรัส และพลผดุง ผดุงกุล, "โครงข่ายคอมพิวเตอร์แบบท้องถิ่นโดยใช้เทคนิคเนกาทีฟพัลส์", การประชุมใหญ่วิชาการทางวิศวกรรม, วิศวกรรมสถานแห่งประเทศไทยในพระบรมราชูปถัมภ์, พ.ศ. 2533.

ภาคผนวก

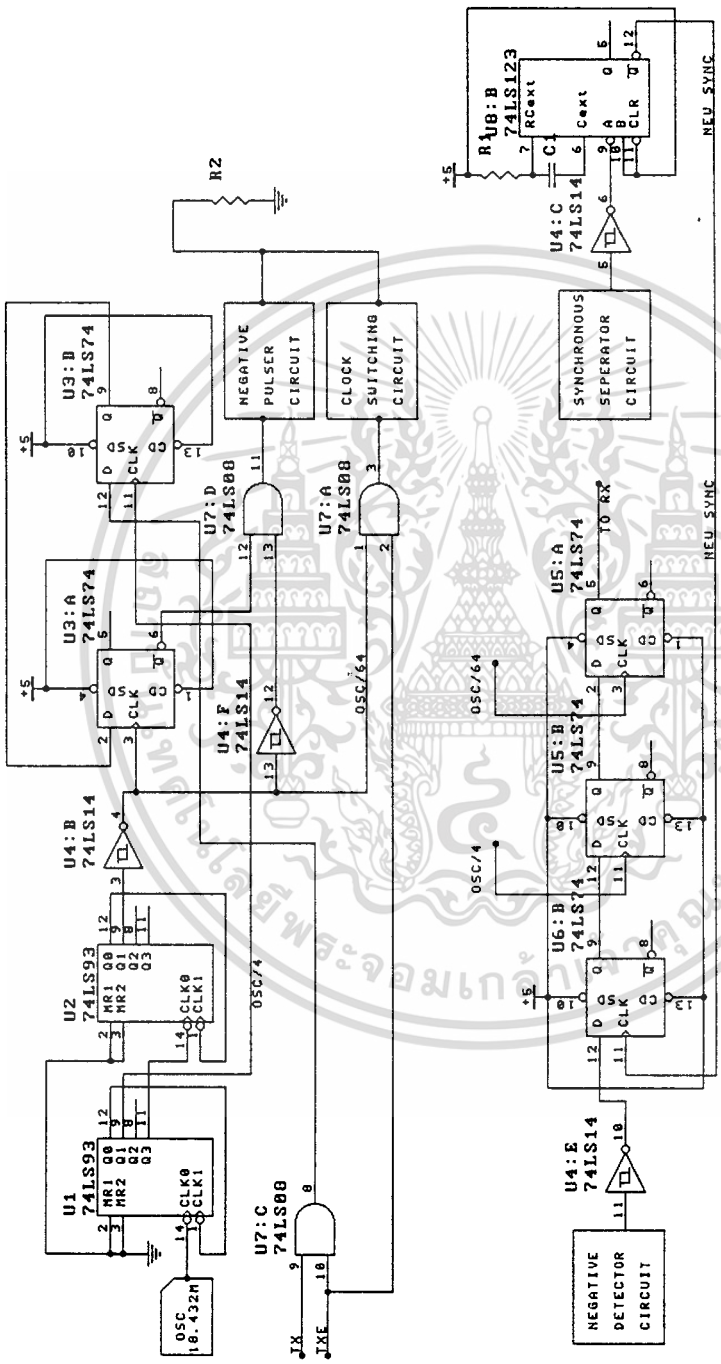
ภาคผนวก ก.

วงจรรวมของระบบโครงข่าย M-NET

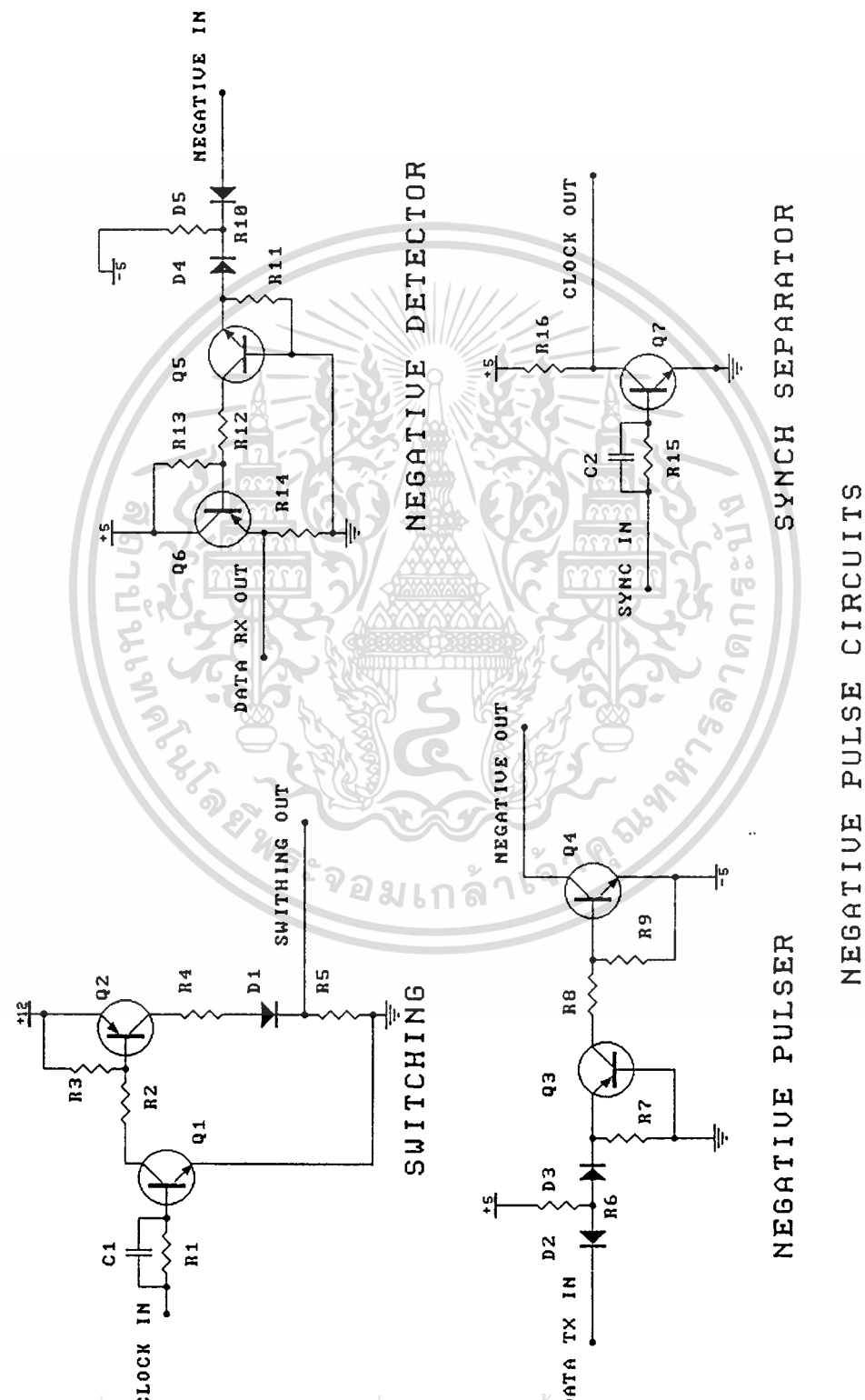
ก.1 วงจรรวมของการ์ด M-NET



ก.2 วงจรรวม เทคนิค เนกาทีฟพัลส์



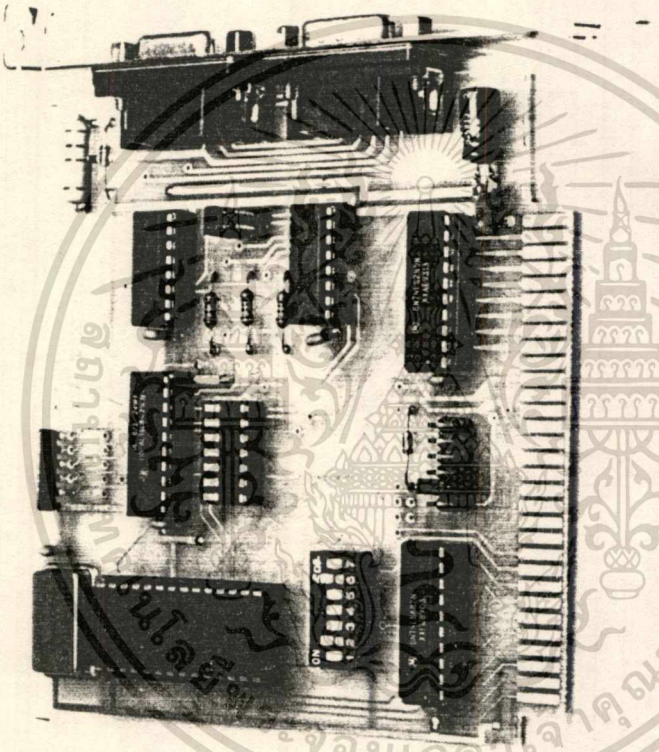
ก.3 วงจรย่อยในวงจร เทคนิค เนกาทีฟพัลส์



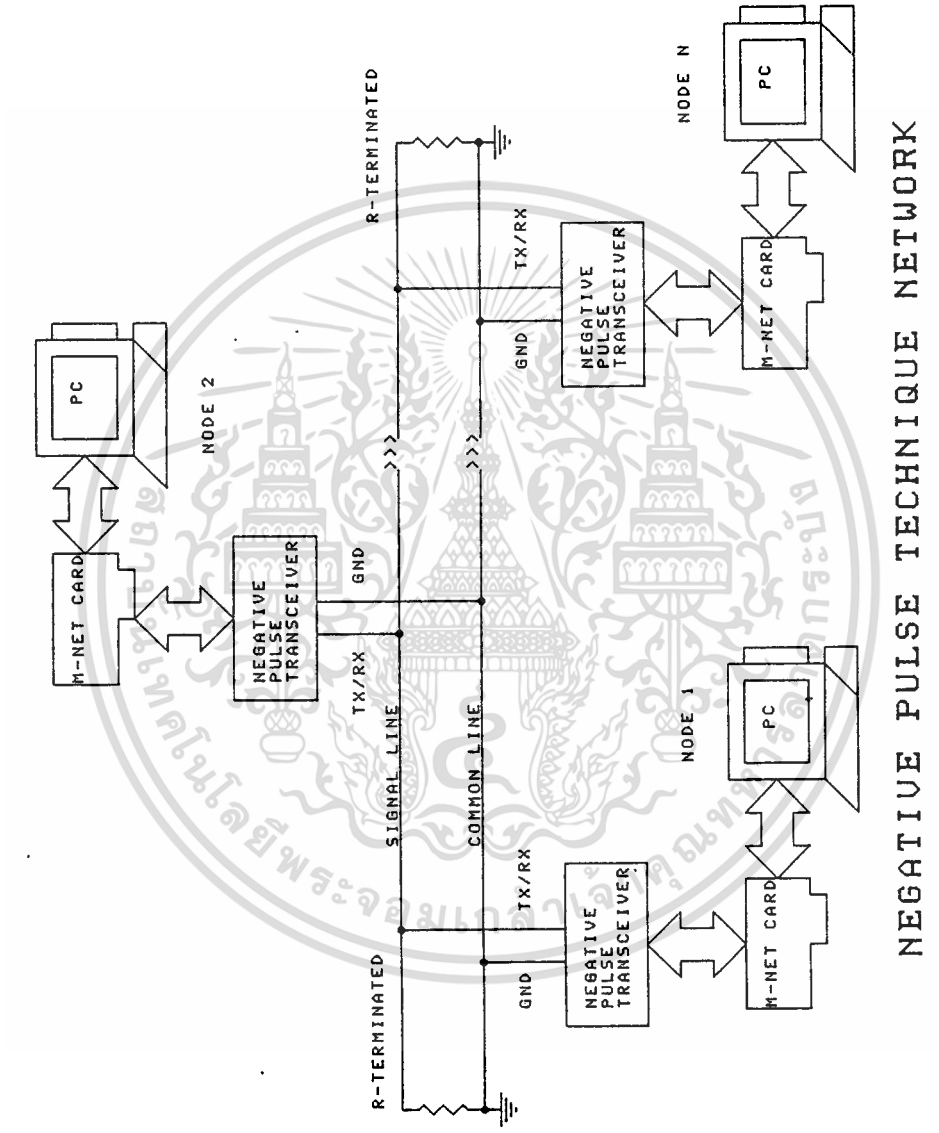
ภาคผนวก ข.

รูปภาพอุปกรณ์ของระบบโครงข่าย M-NET

ข.1 รูปภาพ M-NET การ์ด



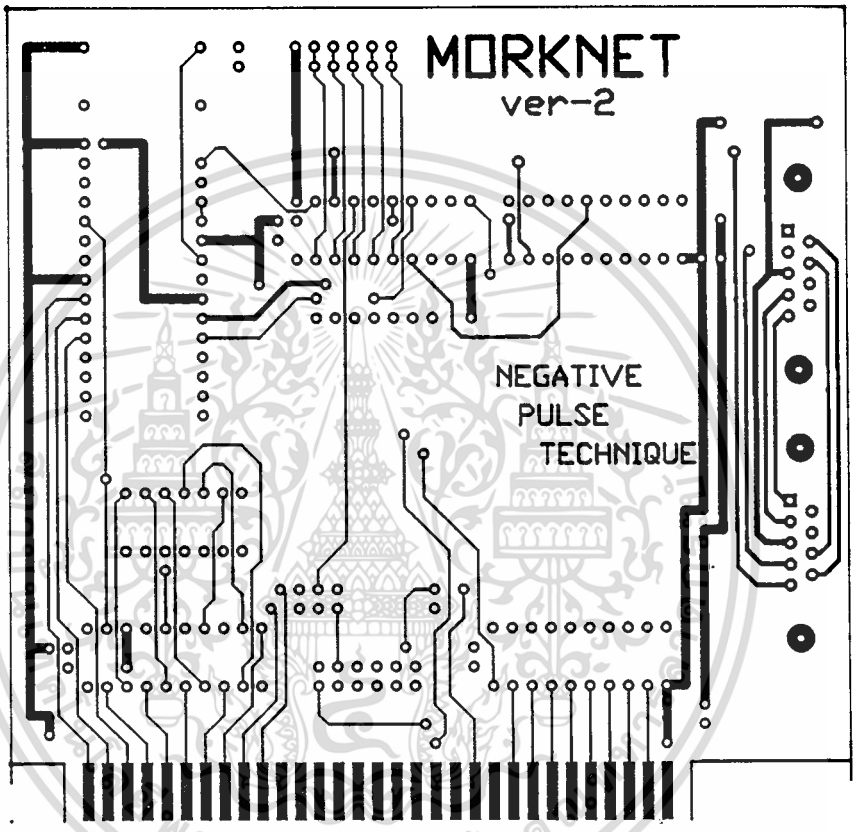
ข.2 ภาพแสดงการติดตั้งอุปกรณ์โครงข่าย M-NET



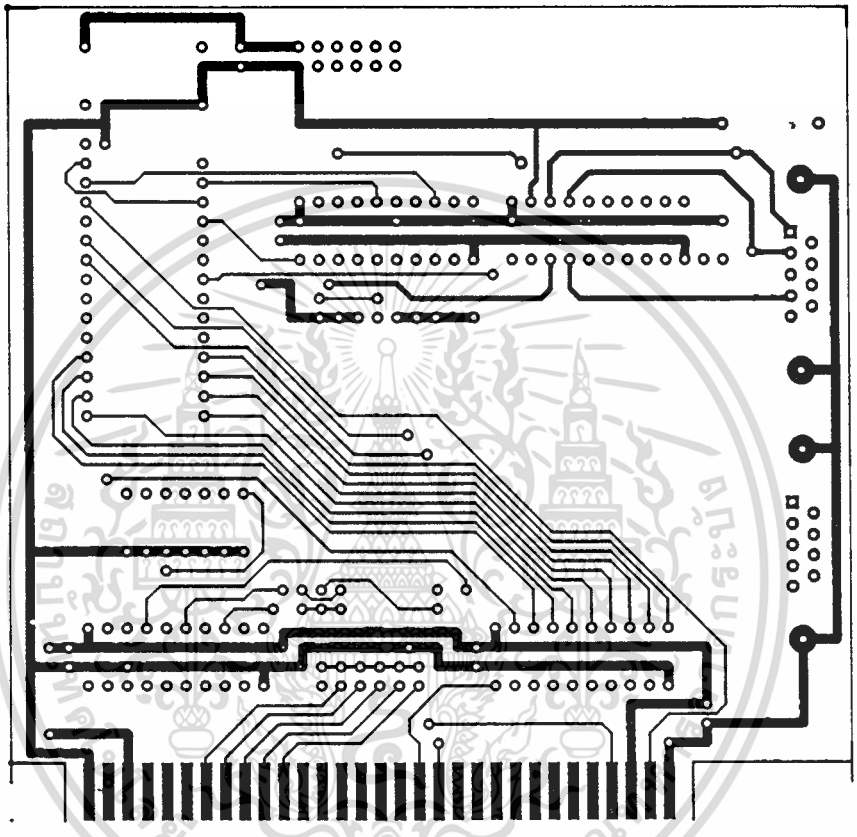
ภาคผนวก ค.

แผ่นพิมพ์ลายวงจรของระบบโครงข่าย M-NET

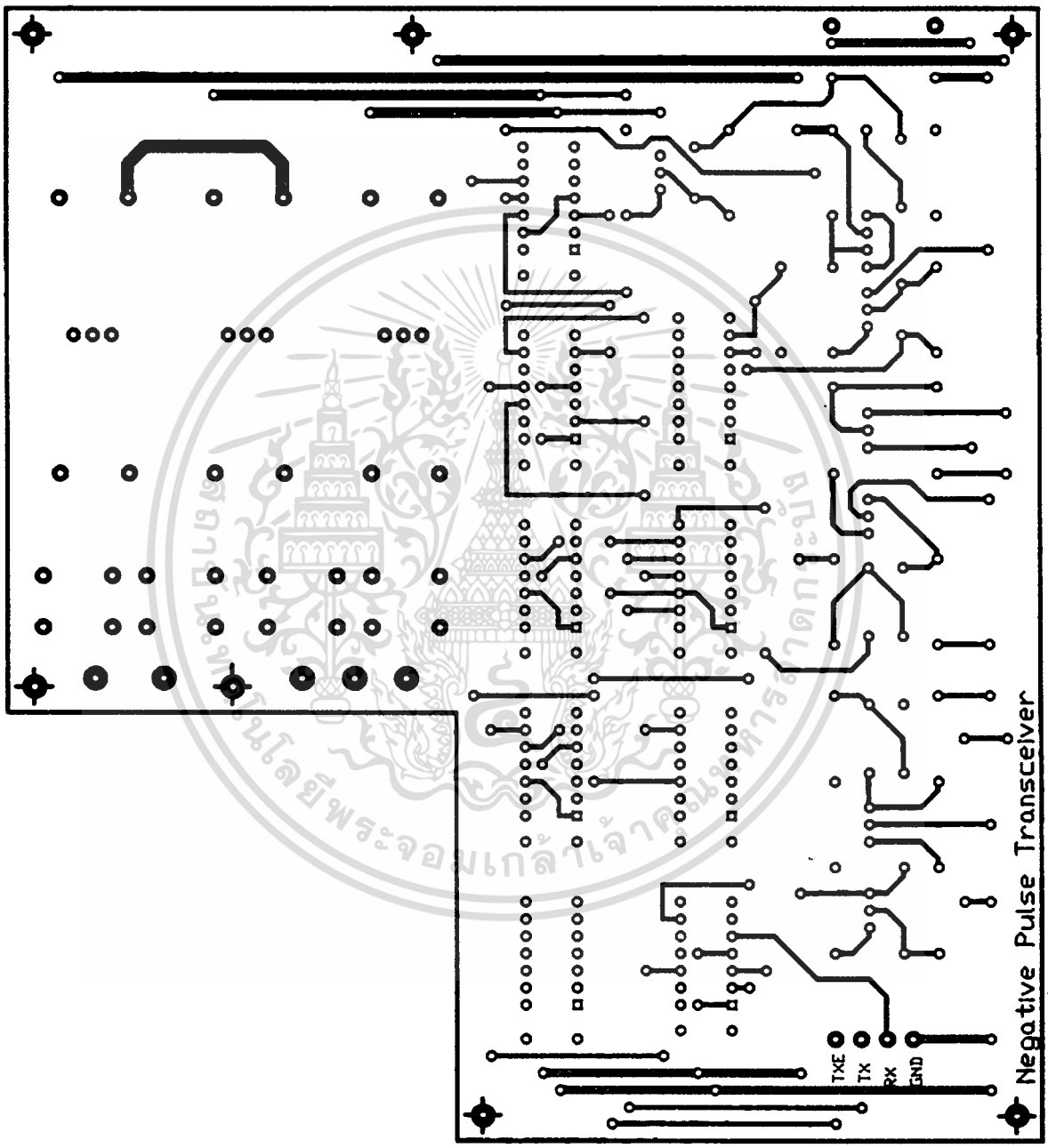
ค.1 แผ่นพิมพ์ลายวงจรของ M-NET การ์ด ด้านอุปกรณ์



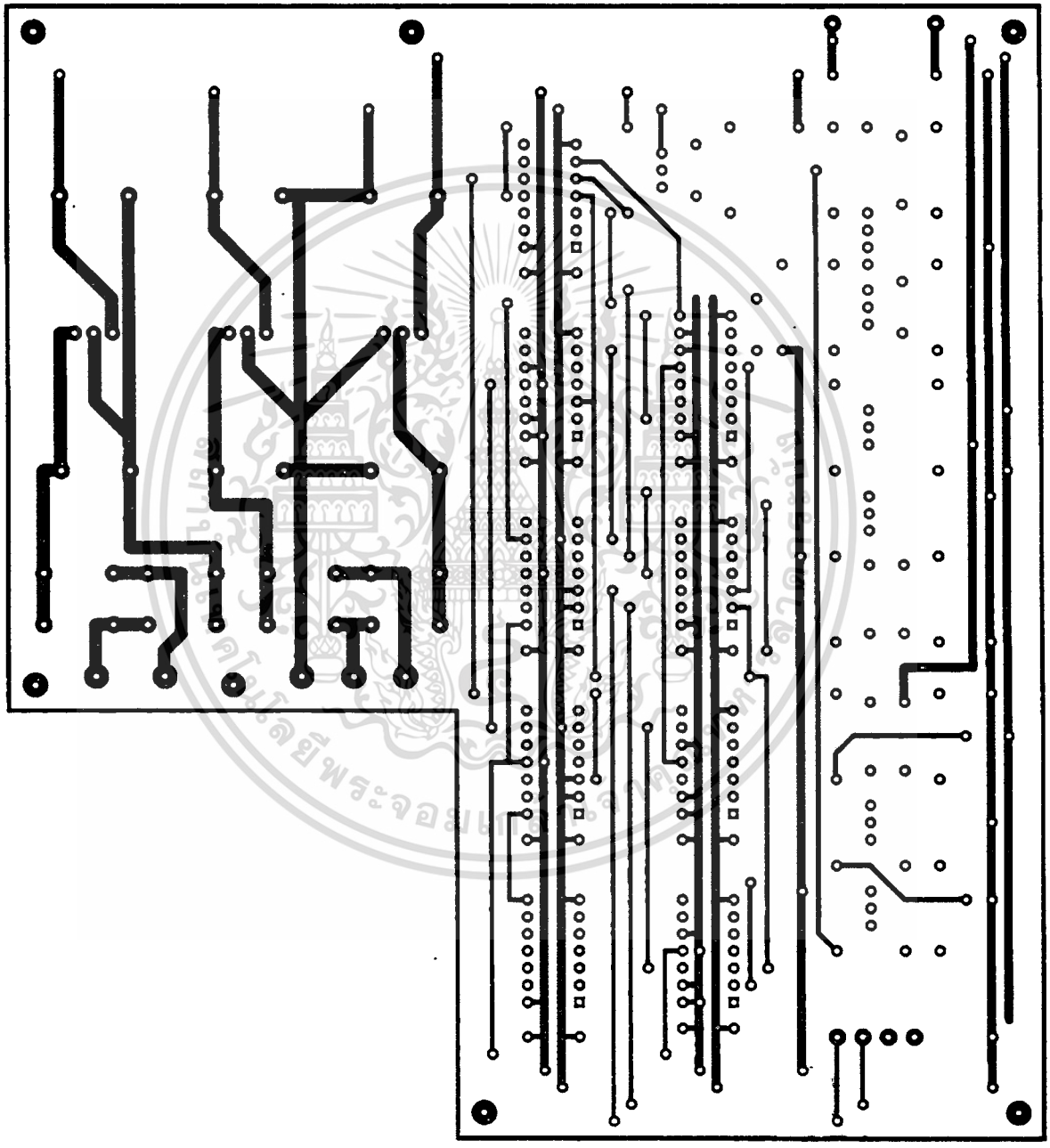
ค.2 แผ่นพิมพ์ลายวงจรของ M-NET การ์ด ด้านล่าง



ค.3 แผ่นพิมพ์ลายวงจร Negative Pulse Transceiver ด้านอุปกรณ์



ค.4 แผ่นพิมพ์ลายวงจร Negative Pulse Transceiver ด้านล่าง



ภาคผนวก ง.

โปรแกรมระบบโครงข่าย M-NET

```
/*
 *    MORKNET program:
 *    Send messages between PCs
 *    an simple CENTRALIZED POLLING protocol.
 *
 *    SW1 settings: (for COM2, address 2F8h: see BASE_ADDRESS)
 *              1-5, 7 ON
 *              6 OFF
 *
 *    Interprocessor message format:
 *              one address character (ninth-bit set)
 *              Data characters (ninth-bit clear)
 */

char *version = "*** 3/8/94 ***";

#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <math.h>
#include <dos.h>
#include <alloc.h>
#include <dir.h>
#include <process.h>
#include <stdlib.h>
#include <bios.h>

#define BASE_ADDRESS 0x2E8          /* COM4 = 2E8 */
#define TX_ENABLE_BIT RTS_BIT       /* RTS_BIT or DTR_BIT */

/*** type definitions ***/
typedef unsigned char uchar;
typedef int boolean;

/*** 82510 register addresses ***/

      /* bank 0 */
#define EMULATE_16450_BANK 0
#define DIVISOR_LOW_ADDRESS (BASE_ADDRESS+0)
#define DIVISOR_HIGH_ADDRESS (BASE_ADDRESS+1)
#define TRANSMIT_BUFFER_ADDRESS (BASE_ADDRESS+0)  /* also bank 1 */
#define RECEIVE_BUFFER_ADDRESS (BASE_ADDRESS+0)          /* also bank 1 */
#define INTERRUPT_ENABLE_ADDRESS (BASE_ADDRESS+1)
#define INTERRUPT_ID_ADDRESS (BASE_ADDRESS+2)      /* any bank */
#define BANK_ADDRESS (BASE_ADDRESS+2)              /* any bank */
#define LINE_CONTROL_ADDRESS (BASE_ADDRESS+3)
#define MODEM_CONTROL_ADDRESS (BASE_ADDRESS+4)     /* also bank 1 */
#define LINE_STATUS_ADDRESS (BASE_ADDRESS+5)
#define MODEM_STATUS_ADDRESS (BASE_ADDRESS+6)
#define SCRATCH_ADDRESS (BASE_ADDRESS+7)

      /* bank 1 */
#define GENERAL_WORK_BANK 1
#define TRANSMIT_FLAGS_ADDRESS (BASE_ADDRESS+1)
#define TRANSMIT_FLAGS_BANK 1
#define RECEIVE_FLAGS_ADDRESS (BASE_ADDRESS+1)
#define RECEIVE_FLAGS_BANK 1
#define GENERAL_STATUS_ADDRESS (BASE_ADDRESS+7)
#define GENERAL_STATUS_BANK 1
#define FIFO_LEVEL_ADDRESS (BASE_ADDRESS+4)
```

```
#define FIFO_LEVEL_BANK 1
#define INTERNAL_COMMAND_ADDRESS (BASE_ADDRESS+7)
#define INTERNAL_COMMAND_BANK 1
      /* bank 2 */
#define TRANSMIT_MACHINE_MODE_ADDRESS (BASE_ADDRESS+3)
#define TRANSMIT_MACHINE_MODE_BANK 2
#define INTERNAL_MODE_ADDRESS (BASE_ADDRESS+4)
#define INTERNAL_MODE_BANK 2
#define FIFO_MODE_ADDRESS (BASE_ADDRESS+1)
#define FIFO_MODE_BANK 2

      /* bank 3 */
#define CLOCKS_CONFIGURE_ADDRESS (BASE_ADDRESS+0)
#define CLOCKS_CONFIGURE_BANK 3
#define BRGB_CONFIGURATION_ADDRESS (BASE_ADDRESS+3)
#define BRGB_CONFIGURATION_BANK 3

/*** useful UART register bits ***/
#define RTS_BIT 0x02            /* modem control register */
#define DTR_BIT 0x01
#define TxIR_BIT 0x10           /* general status register */
#define RFIR_BIT 0x01
#define RESET_BIT 0x10          /* internal command register */

/*** allocate ring buffer ***/
#define RING_BUFFER_SIZE 100
char ringBuffer[RING_BUFFER_SIZE];
int ringInPointer = 0;
int ringOutPointer = 0;

/*** allocte other buffers ***/
char tmpBuff [300];
char *RecvBuff;
char *SendBuff;
char *PntBuff;
char *Table;
/*** miscel laneous constants and variables ***/
#define ESC_KEY 27
#define FALSE 0
#define TRUE (!FALSE)
#define MAX_XTAL_FREQ 20.0
#define MIN_XTAL_FREQ 1.0
double baudRate = 288000.0;          /* default, in bits per second (daud) *
double crystalFrequency = 18.432;  /* default, in MHz */
unsigned int thisNodeAddress;

#define RegisterBank(b) outp(BANK_ADDRESS,(b)<<5);

/*** function prototypes: ***/
void main(void);
void NetworkMonitor(void);
int ReceiveChar(int *ninthBit);
void sendMessage(uchar nodeAddress, unsigned int messagelength);
void TransmitMode(void);
void ReceiveMode(void);
void SetNinthBit(void);
void ClearNinthBit(void);
void SendChar(uchar lowerEightBits);
void EnableFifoMode(void);
void SetBaudRate(double rate);
```

```
unsigned short ComputeDivisor(double baudRate, boolean suppressErrorMsg);
unsigned RecvChar();

#define ASK    0    /* ask from master */
#define ANS    1    /* answer from slave */
#define SEND   2    /* require to SEND file */
#define RECV   3    /* require to RECEIVE file */
#define LOGOUT   4    /* exit program */
#define ERROR  5    /* error such as file can't open */
#define END    6    /* END of FILE */
#define MAIL   7    /* MAIL information */
#define DIRECT 8    /* connect SLAVE <---> SLAVE    (request MASTER) */
#define PRINT  9    /* printer */
#define STATUS 10   /* status node connect */
#define HELP   11   /* help */
#define SYS    12   /* call system */
#define REMOTE 13   /* call system another node */
#define SENDNEXT 14 /* send next */
#define RECVNEXT 15 /* recv next */
#define STB    16   /* start of block */
#define EOB    17   /* end of block */
#define TIMEOUT 18  /* time out for receive data or command */
#define DISPLAY 19  /* display form node to node like mail but not "MAIL=>"*
#define LOGIN   20  /* Login */
#define TEST    21

#define BLOCK 4096
#define BUFFERSIZE BLOCK+100
#define MaxNode 3
#define DelayCmd 30
#define DelayCmdSyn 10
#define PrintName "MORKNET.TMP"
int stat_send[MaxNode];
int stat_recv[MaxNode];

FILE *streamSend[MaxNode];
FILE *streamRecv[MaxNode];

char UserName[8*MaxNode];    /* 8 character * MaxNode */
char node[30];
char NodeAsk;

char FlagPrint=FALSE;
char FlagAns=TRUE;
char FlagReq=FALSE;
char TimeOutErr;                    /* TRUE , FALSE */
int ninthbit;
unsigned int DestNodeAddr;
char FlagRecvPrint=FALSE;
#define Tx 0
#define Rx 1
char ModeTxRx=Rx;       /* check mode display */


int tablesize;
int cntbuff;
int keydata;
char filename[50];
char BufSyn[20];
struct Syn {char SynIn[10];
```

```
          char NodeRx;
          char NodeTx;
          char Instruc;};
struct Syn SynChk;
char rotate[]={"/|\\-"};
char cntr=0;
FILE *streamPnt;
union sizef { char ary[4]; unsigned long lon;};
union sizef FileSizeTx,FileSizeRx;
unsigned long int count,cntRemote;
int cntTime;
char flagTrick;
/*********************************/
/*           MAIN                */
/*********************************/
void main()
{
 char temp[10];
 int cnt,ttt;

 thisNodeAddress=MaxNode-1;

 for(cnt=0;cnt<sizeof(SynChk.SynIn);cnt++)
  SynChk.SynIn[cnt]='0'+cnt;
 for(cnt=0;cnt<MaxNode;cnt++)
 {
  stat_send[cnt]=0;
  stat_recv[cnt]=0;
 }
 NodeAsk=0;
 node[0]=1;

 /*--------allocate ----------*/
 RecvBuff= (char *) malloc((unsigned)BUFFERSIZE);
 if (RecvBuff== NULL)
    terminate("Allocate failed RecvBuff\n");

 /*--------allocate ----------*/
 SendBuff= (char *) malloc((unsigned)BUFFERSIZE);
 if (SendBuff== NULL)
    terminate("Allocate failed SendBuff\n");

 /*--------allocate ----------*/
 PntBuff= (char *) malloc((unsigned)BUFFERSIZE);
 if (PntBuff== NULL)
    terminate("Allocate failed PntBuff\n");

 puts("MORKNET talk program, for 2-wire (half-duplex) NEGATIVE PULSE.");
 puts(version);
 RegisterBank(INTERNAL_COMMAND_BANK);
 outp(INTERNAL_COMMAND_ADDRESS,RESET_BIT);    /* reset 825100 */
 RegisterBank(EMULATE_16450_BANK);

 /*** SETUP UART REGISTERS ***/
/* do
 {
  printf("Crystal Frequency in MHz? [%.4lf MHz] ",crystalFrequency);
  gets(tmpBuff);
  sscanf(tmpBuff,"%lf",&crystalFrequency);
 }
```

```
 while (crystalFrequency < MIN_XTAL_FREQ && crystalFrequency > MAX_XTAL_FREQ
*/
 do
 {
  printf("Speed in baud? [%.11f baud] ",baudRate);
  gets(tmpBuff);
  sscanf(tmpBuff,"%lf",&baudRate);
 }
 while (ComputeDivisor(baudRate,FALSE) == 0);


 SetBaudRate(baudRate);
 outp(MODEM_CONTROL_ADDRESS, 0);
 /* receive mode, disable IRQ, disable loopback */
 /* nine-bit mode: */
 RegisterBank(EMULATE_16450_BANK);
 outp(LINE_CONTROL_ADDRESS,0);
 RegisterBank(TRANSMIT_MACHINE_MODE_BANK);
 outp(TRANSMIT_MACHINE_MODE_ADDRESS,0x20);
 EnableFifoMode();
 outp(INTERRUPT_ENABLE_ADDRESS,0);        /* disable all interrupts */
 inp(LINE_STATUS_ADDRESS);              /* clear out any garbage */
 inp(RECEIVE_BUFFER_ADDRESS);

 /*** OPERATOR TYPES IN ADDRESS NUMBER FOR THIS NODE ***/

 RegisterBank(GENERAL_WORK_BANK);

 do
 {
  printf("login:");
  gets(tmpBuff);
  strupr(tmpBuff);
  printf("Password:");
  gets(tmpBuff+50);
  if(strcmp(tmpBuff,"ADMIN")==0)
  {
   if(strlen(tmpBuff+50)==0) {printf("Password ADMIN error.");terminate();}
   ReadTable();
   if(strcmp(tmpBuff+50,Table)==0) /* check password ADMIN */
   {
    thisNodeAddress=0;
    strcpy(UserName,tmpBuff);
   }
  }
  else
  {
   ttt=0;
   do
   {
    RecvSyn();
    printf(".");
    if((MaxNode-1)==SynChk.NodeRx)      /* This Node */
    {
     if(ninthbit==1&&SynChk.Instruc==ASK)
     {
      SendCommand(0,LOGIN);
      delay(50);
      SendMessage(tmpBuff,strlen(tmpBuff)+1); /* send Login name */
      SendMessage(tmpBuff+50,strlen(tmpBuff+50)+1); /* send password name */
```

```
        ReceiveMode();
        if(RecvCmdSyn()==ERROR)
        {
         printf("password error.\n");
         ttt=1;
        }
        else
         cnt=RecvMessage(RecvBuff);
        ttt=RecvChar1();
         if(cnt==1)
         {
        ttt=1;
        thisNodeAddress=RecvBuff[0];
         }
        }
       }
      }
      while(ttt==0);
     }
  }
  while (thisNodeAddress >= (MaxNode-1));
  printf("This node = %d.\n",thisNodeAddress);
  printf("Enter ? for HELP.\n");
  Network();
         /*** CLEANUP UART REGISTERS BEFORE TERMINATING ***/
  terminate();
}
/*******************************/
ReadTable()
{
 int cnt;

 if((streamPnt=fopen("user.tab","rt"))==NULL)
 {
  printf("Not have user table file.\n");
  terminate();
 }
 else
 {
  fseek(streamPnt,0,SEEK_END);
  cntbuff=ftell(streamPnt);
  fseek(streamPnt,0,SEEK_SET);
  /*--------allocate ----------*/
  Table = (char *) malloc(cntbuff);
  if (Table== NULL)
   terminate("Allocate failed Table\n");
  tablesize=fread(Table,1,cntbuff,streamPnt);
  if(tablesize==0)
  {
   printf("Read error on user table file.\n");
   terminate();
  }
  else
  {
   fclose(streamPnt);
   for(cnt=0;cnt<tablesize;cnt++)
    if(Table[cnt]=='\n')
     Table[cnt]=0;
  }
 }
```

```
}
/*****************************/
terminate()
{
 RegisterBank(INTERNAL_COMMAND_BANK);
 outp(INTERNAL_COMMAND_ADDRESS,RESET_BIT);     /* reset 82510 */
 free(RecvBuff);
 free(SendBuff);
 free(Table);
 free(PntBuff);
 exit(1);
}
/*****************************/
SendEnd()
{
 TransmitMode();
 SendCmd(END);
 ReceiveMode();
}
/*****************************/
SendCmd(uchar node)
{
 SetNinthBit();
 SendChar(node);
 ClearNinthBit();
}
/*****************************/
RecvSyn()
{
 unsigned temp,cnt=0;
 char flag=0;
 unsigned count=0;
 do
 {
  if(FlagReq==FALSE)
   KeyFunc();
  else
   delay(3);
  temp=RecvChar();          /* hand check */
  if(temp!=0xffff)
  {
   if(ninthbit==1)
   {
    switch(flag)
    {
     case 0:if(temp=='8') flag++; count=0;       break;
     case 1:if(temp=='9') flag++; count=0;       break;
     case 2:switch(cnt)
          { case 0:SynChk.NodeRx=temp;  cnt++; count=0;       break;
            case 1:SynChk.NodeTx=temp;  cnt++; count=0;       break;
            case 2:SynChk.Instruc=temp; cnt++; count=0;
                 /*if(SynChk.NodeRx==thisNodeAddress)
                  printf("%d ",temp);*/
                 break;}
    }
   }
  }
  else
  {
   count++;
```

```
    if(count==1000)
    {
     count=0;
     ninthbit=0;
     return;
    }
   }
  }
  while(cnt!=3);
  delay(50);
}
/*****************************/
unsigned RecvChar()
{
 if(!(inp(GENERAL_STATUS_ADDRESS) & RFIR_BIT))
  return(0xffff);
 else
  return(ReceiveChar(&ninthbit));
}
/*****************************/
int RecvChar1(int t)
{
 int temp;
 unsigned  cnt1=0,cnt2=0;

 TimeOutErr=TRUE;
 while(!(inp(GENERAL_STATUS_ADDRESS) & RFIR_BIT))
 {
  cnt1++;
  if(cnt1==65535)
  {
   cnt2++;
   if(cnt2==t)
   {
    TimeOutErr=FALSE;
  /*  printf("Time out error.\n");*/
    ninthbit=0;
    return(ERROR);
   }
  }
 }
 return(ReceiveChar(&ninthbit));
}
/*****************************/
ScanNode()
{
 int cnt;

 for(cnt=1;cnt<MaxNode;cnt++)
  node[cnt]=1;
 for(cnt=1;cnt<MaxNode;cnt++)
 {
  SendCommand(cnt,ASK);
  ReceiveMode();
  delay(500);
  RecvSyn();
  if(ninthbit==1)
  {
   if(SynChk.NodeRx==0) /* master */
   {
```

```
    if(SynChk.Instruc==ANS) /* instruction */
     node[cnt]=1;                    /* found */
    else
     node[cnt]=0;       /* not found */
   }
   else
    node[cnt]=0;        /* not found */
  }
  else
   node[cnt]=0;        /* not found */
  if(node[cnt]==1)
   printf("\nnode:%d connect  ",cnt);
  else
   printf("%c%c",8,rotate[cnt&3]);
 }
 NextNode();
}
/*****************************/
NextNode()              /* search node connect */
{
 NodeAsk++;
 if(NodeAsk==MaxNode)
  NodeAsk=1;
}
/*****************************/
/*           NETWORK         */
/*****************************/
Network()
{
 cntbuff=0;
 if(thisNodeAddress==0)
 {
  ScanNode();           /* master node */
  AK_ans();
 }
 printf("\n>");
 for(;;)
  OneCycle();
}
/*.........................*/
OneCycle()
{
 RecvSyn();          /* Receive 1 Instruction  (Synce,Node,Cmd,Synce)*/
 if(ninthbit==1)
 {
  if(thisNodeAddress==SynChk.NodeRx)     /* This Node */
    RunAK();
  else
  {
   switch(SynChk.Instruc)       /* not this node */
   {
    case SEND:
    case RECV:
    case MAIL:
    case SENDNEXT:
    case RECVNEXT:
    case PRINT:
    case STATUS:
    case DISPLAY:
    case REMOTE:
```

```
    case LOGIN:
    case LOGOUT:
    case TEST:
            printf("*%c",8);FlagAns=TRUE;cntbuff=0;
            do {KeyFunc();} while(RecvChar1(10)!=END||ninthbit!=1);
            printf(" %c",8);break;
   }
  }
 }
 else
 {
  if(thisNodeAddress==0)
  {
   node[NodeAsk]=0;
   AK_ans();     /* case => Node Disconnect */
  }
 }
}
/*.....................*/
RunAK()
{
 switch(SynChk.Instruc)
 {
  case ASK :                                  /* for SLAVE   (ANS) */
          printf("%c%c",rotate[cntr&3],8);cntr++;
          AK_ask(); FlagAns=TRUE; break;
  case ANS :                          /* for MASTER  (ASK) */
          node[NodeAsk]=1;
          printf("%c%c",rotate[cntr&3],8);cntr++;
          AK_ans(); FlagAns=TRUE; break;
  case SEND:    /* SEND from master : slave RECV */
          AK_recv();
          FlagAns=TRUE; break;
  case RECV:    /* RECV from master : slave SEND */
          AK_send();
          FlagAns=TRUE; break;
  case MAIL:
          AK_mail();
          FlagAns=TRUE; break;
  case DIRECT:AK_direct();break;
  case PRINT: AK_print();
          FlagAns=TRUE; break;
  case STATUS:node[SynChk.NodeTx]=1;
          AK_status();
          FlagAns=TRUE; break;
  case REMOTE:AK_remote();
          FlagAns=TRUE; break;
  case SENDNEXT:RecvFile(SynChk.NodeTx);
          FlagAns=TRUE; break;
  case RECVNEXT:SendFile(SynChk.NodeTx);
          FlagAns=TRUE; break;
  case DISPLAY:AK_display();
          FlagAns=TRUE; break;
  case LOGIN:AK_login();                      /* Master only  */
          FlagAns=TRUE; break;
  case LOGOUT:     /* LOGOUT */
          terminate(); break;
  case TEST: AK_test();
          FlagAns=TRUE; break;
  defualt: cntbuff=0; printf("Not Instruction \n");break;
```

```
 }
}
/************************************************/
/* CONDITION if   NO KEY or KEY != '\r' return FALSE
          else return cmdcode[] */
int InKey()
{
 char *cmd[] =  {"SEND","RECV","MAIL","LOGOUT","PRINT","STATUS","?" ,"!","!!
             "S"   ,"R"   ,"M"   ,"L"     ,"P"     ,"T"      , "TEST"};
 int cmdcode[]= { SEND,  RECV,  MAIL,  LOGOUT , PRINT , STATUS ,
HELP,SYS,REMOTE,
              SEND,  RECV,  MAIL,  LOGOUT , PRINT , STATUS ,TEST};
 char temp,flag,cntmenu,cnt;

 if(kbhit()!=0)
 {
  tmpBuff[cntbuff]=getch();
  if(tmpBuff[cntbuff]=='\r') printf(" ");
  printf("%c",tmpBuff[cntbuff]);
  if(tmpBuff[cntbuff]==8)
  {
   if(cntbuff>0)
   {
    printf("  %c%c",8,8);
    cntbuff--;
   }
   else
    printf(">",8);
  }
  else
   cntbuff++;
  if(tmpBuff[cntbuff-1]!='\r')
   return(FALSE);
  else
  {
   printf("\n");
   tmpBuff[cntbuff-1]=0;
   for(cnt=0,temp=0;temp<cntbuff&&cnt<2;temp++)
   {
    if(tmpBuff[temp]==' ')
    {
     tmpBuff[temp]=0;                    /* clear space */
     cnt++;
    }
   }
   cntmenu=0;
   strupr(tmpBuff);
   do
   {
    flag=strcmp(cmd[cntmenu],tmpBuff);
    cntmenu++;
   }
   while(cntmenu< sizeof(cmd) &&flag!=0);

   if(flag!=0||cntbuff==1)
   {
    printf("Cmd Er\n>");
    return(ERROR);
   }
   else
```

```
    {
     return(cmdcode[cntmenu-1]);              /* return code command */
    }
   }
  }
  else
   return(FALSE);
}
/***************************/
KeyFunc()
{
 if(FlagPrint==TRUE)
  PrintData();
 keydata=InKey();
 if(keydata!=FALSE)
 {
  if(keydata==ERROR)
   cntbuff=0;
  else
  {
   FlagReq=TRUE;
   if(keydata==LOGOUT&&GetPointBuf(1)==0)      /* check function LOGOUT */
    terminate();
   printf("wait.\n");
  }
 }
}
/************************/
PrintData()
{
 int temp;

 if(prnready()) /* get status printer */
 {
  temp=getc(streamPnt);
  if(temp!=EOF)
   biosprint(0,temp,0);     /* send to print */
  else
  {
   FlagPrint=FALSE;
   fclose(streamPnt);
   bdos(0x41,FP_OFF(PrintName),0); /* delete file */
  }
 }
}
/*.......................*/
int prnready()
{
 int status;
 status=biosprint(2,NULL,0);
 return(status&0x10)?1:0;
}

/***************************/
/*   ACKNOWLAGE FUNCTION   */
/***************************/
AK_test()
{
 int cnt,cnt1;
 float total;
```

```
 if(RecvCmdSyn()!=STB) return;
 total=0;
 cnt1=RecvChar1(20);
 cnt=cnt1-1;
 do
 {
  cnt1=RecvChar1(20);
  if(cnt1!=-1)
  {
   if(cnt!=cnt1-1)
    if(cnt1!=0)
     printf("%d",cnt1);
    else
     total++;
   cnt=cnt1;
  }
 }
 while(cnt1!=0xff);
 printf("total =%f\n",(float)total*256);
 ReceiveMode();
 }
/*************************/
AK_print()
{
 int temp;

 TransmitMode();
 if(FlagPrint==TRUE)
 {
  SendCmdSyn(ERROR);  /* printer not ready */
  return;
 }
 if(!prnready())
  SendCmdSyn(ERROR);  /* printer not ready */
 else
 {
  SendCmdSyn(ANS);
  ReceiveMode();
  FlagRecvPrint=TRUE;
  strcpy(filename,PrintName);
  ModeTxRx=Tx;
  RecvFile(SynChk.NodeTx);
 }
 ReceiveMode();
}
/*************************/
AK_recv()
{
 int temp;

 TransmitMode();
 SendCmdSyn(ANS);
 ReceiveMode();
 ModeTxRx=Tx;
 temp=RecvMessage(filename);
 if(temp==TIMEOUT) {printf("Time out error\n");return;}
 RecvFile(SynChk.NodeTx);
}
/*************************/
```

```
AK_send()
{
 int temp;

 TransmitMode();
 SendCmdSyn(ANS);
 ReceiveMode();
 ModeTxRx=Rx;
 temp=RecvMessage(filename);
 if(temp==TIMEOUT) {printf("Time out error\n");return;}
 TransmitMode();
 delay(100);
 SendFile(SynChk.NodeTx);
 ReceiveMode();
}
/***************************/
AK_mail()
{
 unsigned temp;
 char x,y;

 temp=RecvMessage(RecvBuff);
 if(temp!=0)
 {
  for(temp=0;temp<5;temp++)
  {
   sound(temp*1000);
   delay(150);
  }
  nosound();
  x=wherex();
  y=wherey();
  gettext(1,25,79,25,RecvBuff+500);
  gotoxy(1,25);
  textattr(0x9f);
  cprintf("MAIL from %d=>",SynChk.NodeTx);
  textattr(0x1f);
  cprintf("%s",RecvBuff);
  getch();
  puttext(1,25,79,25,RecvBuff+500);
  gotoxy(x,y);
 }
}
/***************************/
AK_display()
{
 unsigned temp;

 temp=RecvMessage(RecvBuff);
 if(temp!=0)
  printf("%s",RecvBuff);
}
/***************************/
AK_ans()                        /* only MASTER */
{
 if(FlagReq==TRUE)
 {
  ReqFunc();
  FlagReq=FALSE;
 }
```

```
 NextNode();
 if(stat_send[NodeAsk]!=0)
 {
  SendCommand(NodeAsk,SENDNEXT);
  SendFile(NodeAsk);              /* Send File */
 }
 else
 {
  if(stat_recv[NodeAsk]!=0)
  {
   SendCommand(NodeAsk,RECVNEXT);    /* RECV File */
   ReceiveMode();
   RecvFile(NodeAsk);
  }
  else
   SendCommand(NodeAsk,ASK);      /* ASK */
 }
 ReceiveMode();
}
/***************************/
AK_ask()                          /* only SLAVE */
{
 if(FlagReq==TRUE)
 {
  ReqFunc();
  FlagReq=FALSE;
 }
 NodeAsk=0;
 while(stat_send[NodeAsk]==0&&NodeAsk<MaxNode){NodeAsk++;}
 if(stat_send[NodeAsk]!=0)
 {
  ChkDirect();
  SendCommand(NodeAsk,SENDNEXT);
  SendFile(NodeAsk);
 }
 else
/* {
  if(stat_recv[NodeAsk]!=0)
  {
   SendCommand(NodeAsk,RECVNEXT);
   ReceiveMode();
   RecvFile(NodeAsk);
  }
  else*/
   SendCommand(0,ANS);
/* }*/
 ReceiveMode();
}
/***************************/
AK_direct()                       /* only SLAVE */
{
 printf("*\%c",8);FlagAns=TRUE;
 do {KeyFunc();} while(RecvChar1(10)!=END||ninthbit!=1);
 printf(" \%c",8);
}
/***************************/
AK_status()                       /* only Master */
{
 unsigned cnt=0,cntbf;
```

```
 TransmitMode();
 sprintf(SendBuff,"\nThis node = %d [%s]",SynChk.NodeTx,UserName+(8*
SynChk.NodeTx));
 cntbf=strlen(SendBuff);
 for(cnt=0;cnt<MaxNode;cnt++)
  if(node[cnt]==1)
  {
   sprintf(SendBuff+cntbf,"\nNode %d [%s] connect",cnt,UserName+(8*cnt));
   cntbf=cntbf+strlen(SendBuff+cntbf);
  }
 delay(50);
 SendMessage(SendBuff,cntbf+1);
 ReceiveMode();
 do {KeyFunc();} while(RecvChar1(10)!=END||ninthbit!=1);
}
/***************************/
void interrupt(*oldvecSCR)();

AK_remote()                          /* */
{
 unsigned cnt=0;
 void interrupt runvecSCR();

 TransmitMode();
 SendCmdSyn(ANS);
 ReceiveMode();
 printf("*%c",8);
 cnt=RecvMessage(RecvBuff);
 if(cnt==0) return;
 oldvecSCR=getvect(0x10);  /* DISPLAY */
 setvect(0x10,runvecSCR);
 cntRemote=0;
 system(RecvBuff);
 setvect(0x10,oldvecSCR);
 if(cntRemote<BLOCK-2)
 {
  RecvBuff[cntRemote]='\n';
  RecvBuff[cntRemote+1]='>';
  cntRemote++;cntRemote++;
 }
 else
 {
  remote_display();
  RecvBuff[0]='\n';
  RecvBuff[1]='>';
  cntRemote=2;
 }
 delay(700);
 remote_display();
 printf(" %c",8);
}
void interrupt runvecSCR()
{

 if(_AH==0xe||_AH==0xa||_AH==0x9)
 {
  RecvBuff[cntRemote]=_AL;
  if(RecvBuff[cntRemote]!=0x3)   /* don't use code */
  {
   if(cntRemote<BLOCK)
```

```
    cntRemote++;
   else
    remote_display();
  }
 }
 else
  (*oldvecSCR)();
}
/*************************/
remote_display()
{
 int temp,flag=0;

 if(cntRemote!=0)
  do
  {
   if(thisNodeAddress==0)
   {
    delay(500);
    SendCommand(NodeAsk,DISPLAY);
    RecvBuff[cntRemote]=0;
    SendMessage(RecvBuff,cntRemote+1);
    delay(50);
    cntRemote=0;
    flag=1;
   }
   else
   {
    RecvSyn();          /* Receive 1 Instruction  (Synce,Node,Cmd,Synce)*/
    if(ninthbit==1)
     if(thisNodeAddress==SynChk.NodeRx)      /* This Node */
      if(SynChk.Instruc==ASK)
       {
      SendCommand(0,DISPLAY);
      RecvBuff[cntRemote]=0;
      SendMessage(RecvBuff,cntRemote+1);
      cntRemote=0;
      flag=1;
       }
   }
   ReceiveMode();
  }
  while(flag==0);
}
/*************************/
AK_login()
{
 int temp,cnt,cntlogin;
 char flag;

 temp=RecvMessage(RecvBuff);     /* receive LOGIN */
 if(temp!=0)
 {
  temp=strlen(RecvBuff);
  cntlogin=temp;
  RecvBuff[temp]=' ';
  temp=RecvMessage(RecvBuff+temp+1);  /* receive PASSWORD */
  if(temp!=0)
  {
```

```
    temp=0;
    do
    {
     flag=strcmp(RecvBuff,Table+temp);
     do{temp++;}while(temp<tablesize&&Table[temp]!=0);
     temp++;
    }
    while(temp<tablesize&&flag!=0);
    delay(100);
    TransmitMode();
    if(flag==0)
    {
     SendCmdSyn(ANS);
     temp=1;
     while(temp<MaxNode&&node[temp]!=0){temp++;}
     SendMessage(&temp,1);
     RecvBuff[cntlogin]=0;
     strcpy(UserName+(8*temp),RecvBuff);
     delay(100);
    }
    else
     SendCmdSyn(ERROR);
    SendEnd();
    ReceiveMode();
   }
  }
}
/*****************************/
/*      REQUIRE FUNCTION     */
/*****************************/
ReqFunc()
{
 GetFileName();
 GetNode();
 ChkDirect();
 switch(keydata)
 {
  case SEND:RQ_send();  SendEnd();break;
  case RECV:RQ_recv();  SendEnd();break;
  case MAIL:RQ_mail();  SendEnd();break;
  case PRINT:RQ_print();SendEnd();break;
  case REMOTE:RQ_remote();SendEnd();break;
  case STATUS:RQ_status();        break;
  case HELP:RQ_help();            break;
  case SYS:RQ_sys();              break;
  case LOGOUT:RQ_quit();          break;
  case TEST:RQ_test();            break;
 }
 cntbuff=0;
 FlagAns=TRUE;
 printf("\n>");
}
/*****************************/
int GetPointBuf(int num)
{
 int temp,cnt;

 cnt=0;
 temp=0;
 while(cnt<num&&temp<cntbuff)
```

```
 {
  for(;tmpBuff[temp]!=0;temp++)
   {}
  temp++;
  cnt++;
 }
 if(temp<cntbuff)
  return(temp);
 else
  return(0);
}
/*****************************/
GetFileName()
{
 int temp;

 temp=GetPointBuf(2);
 if(temp==0)
  filename[0]=0;
 else
 {
  if(strlen(tmpBuff+temp)<40)
   strcpy(filename,tmpBuff+temp);
  else
   filename[0]=0;
 }
}
/*****************************/
GetNode()
{
 int temp;

 temp=GetPointBuf(1);
 if(temp==0)
  DestNodeAddr=0xff;
 else
  DestNodeAddr=atoi(tmpBuff+temp);
/* if(DestNodeAddr>MaxNode)
  printf("Node error\n"); */
}
/*****************************/
RQ_test()
{
 char cnt;

 SendCommand(1,TEST);
 cnt=0;
 SendCmdSyn(STB);
 delay(50);
 do
 {
  SendChar(cnt);
/*  delay(1);*/
  cnt++;
  if(cnt==0xff) cnt++;
/*  printf("%d ",cnt);*/
 }
 while(kbhit()==0);
  SendChar(0xff);
 ReceiveMode();
```

```
}
/***************************/
RQ_print()
{
 int temp;

 SendCommand(DestNodeAddr,PRINT);
 ReceiveMode();
 temp=RecvCmdSyn();                    /* receive */
 switch(temp)
 {
  case TIMEOUT: return;
  case ERROR  : printf("Printer Not Ready.\n"); break;
  case ANS    : TransmitMode();
            ModeTxRx=Tx;
            SendFile(DestNodeAddr);
            ReceiveMode(); break;
 }
}
/***************************/
RQ_send()
{
 int temp;

 SendCommand(DestNodeAddr,SEND);
 ReceiveMode();
 temp=RecvCmdSyn();                    /* receive */
 switch(temp)
 {
  case TIMEOUT:
  case ERROR  : printf("Destination node not ready.\n"); break;
  case ANS    : TransmitMode();
            ModeTxRx=Tx;
            SendMessage(filename,strlen(filename)+1);   /* send FILE NAME */
            SendFile(DestNodeAddr);
            ReceiveMode();
  }
}
/***************************/
RQ_recv()
{
 int temp;

 SendCommand(DestNodeAddr,RECV);
 ReceiveMode();
 temp=RecvCmdSyn();                    /* check error*/
 switch(temp)
 {
  case TIMEOUT:
  case ERROR  : printf("Destination node not ready.\n"); break;
  case ANS    : TransmitMode();
            ModeTxRx=Rx;
            SendMessage(filename,strlen(filename)+1); /* send FILE NAME */
            ReceiveMode();
            RecvFile(DestNodeAddr);
  }
}
/***************************/
RQ_mail()                             /* "1234"+mail */
{
```

```
 int temp,strsize;

 SendCommand(DestNodeAddr,MAIL);
 temp=GetPointBuf(2);
 SendMessage(tmpBuff+temp,strlen(tmpBuff+temp)+1);    /* send MAIL */
 ReceiveMode();
}
/***************************/
RQ_quit()
{
 if(DestNodeAddr==0xff)
  terminate();
 else
 {
  SendCommand(DestNodeAddr,LOGOUT);
  ReceiveMode();
 }
}
/***************************/
ChkDirect()
{
 if(thisNodeAddress!=0&&DestNodeAddr!=0&&keydata!=STATUS)   /* check direct
mode access */
 {
  SendCommand(0,DIRECT);              /* send DIRECT to MASTER */
  ReceiveMode();
 }
}
/***************************/
RQ_status()
{
 unsigned temp,cnt=0;

 if(thisNodeAddress==0)
 {
  printf("\nThis node = %d [%s]",thisNodeAddress,UserName+(8*thisNodeAddress))
;
  for(cnt=0;cnt<MaxNode;cnt++)
   if(node[cnt]==1)
    printf("\nNode %d [%s] connect",cnt,UserName+(8*cnt));
 }
 else
 {
  SendCommand(0,STATUS);
  ReceiveMode();
  delay(50);
  temp=RecvMessage(RecvBuff);
  if(temp!=0)
   printf("%s\n",RecvBuff);
  SendEnd();
 }
}
/***********************/
RQ_help()
{
 printf("\n?\t\t\t (help)\n");
 printf("Send \tnode file\t (send file)\n");
 printf("Recv \tnode file\t (receive file)\n");
 printf("Mail \tnode message\t (mail)\n");
 printf("sTatus\t\t\t (status)\n");
```

```
 printf("Print \tnode file\t (print file)\n");
 printf("! \tcommand\t\t (call dos command)\n");
 printf("!! \tnode command\t (remote dos command)\n");
 printf("Logout \t[node]\t\t (exit)\n");
}
/***********************/
RQ_sys()
{
 int temp,cnt;

 temp=GetPointBuf(1);
 for(cnt=temp;cnt<cntbuff-temp;cnt++)
  if(tmpBuff[cnt]==0)
   tmpBuff[cnt]=' ';
 SysClock(tmpBuff+temp);
}
/*.........................*/
void interrupt(*oldvecTIME)();

SysClock(char *buf)                    /* */
{
 void interrupt runvecTIME();

 oldvecTIME=getvect(8);    /* TIMER   */
 setvect(8,runvecTIME);
 cntTime=0;
 flagTrick=0;
 system(buf);
 setvect(8,oldvecTIME);
}
void interrupt runvecTIME()
{
 (*oldvecTIME)();

 cntTime++;
 if(cntTime>5)
 {
  cntTime=0;
  if(flagTrick==0)
   Get1Cycle();
 }
}
/*.................*/
Get1Cycle()
{
 unsigned temp,cntRecvCmd;
 char flagok=0;

 if(thisNodeAddress==0) return;
 cntRecvCmd=0;
 flagTrick=1;
 do
 {
  temp=RecvChar();
  if(temp!=0xffff)
  {
   if(ninthbit==1)
   {
    switch(cntRecvCmd)
    {
```

```
      case 0:if(temp=='8') {cntRecvCmd++; flagok=0;} break;
      case 1:if(temp=='9') {cntRecvCmd++; flagok=0;}
           else flagok=1; break;
      case 2:SynChk.NodeRx=temp;  cntRecvCmd++;  flagok=0;       break;
      case 3:SynChk.NodeTx=temp;  cntRecvCmd++;  flagok=0;       break;
      case 4:SynChk.Instruc=temp; cntRecvCmd++;  flagok=1;       break;
      default:flagok=1;
     }
    }else flagok=1;
   }
  }
  while(flagok==0);
  if(SynChk.NodeRx==thisNodeAddress)
  {
   switch(SynChk.Instruc)
   {
    case ASK:TransmitMode();
           SendCommand(0,ANS);
           ReceiveMode(); break;
    case MAIL:AK_mail();
            break;
    case SEND:
    case RECV:
    case PRINT:
    case REMOTE:
           TransmitMode();
           SendCmdSyn(ERROR);
           ReceiveMode(); break;
   }
  }

  flagTrick=0;
 }
 /*************************/
 RQ_remote()
 {
  unsigned temp,cnt=0;

  if(thisNodeAddress==DestNodeAddr)
   printf("Can't call in the same node\n");
  else
  {
   SendCommand(DestNodeAddr,REMOTE);
   ReceiveMode();
   temp=RecvCmdSyn();                  /* check error*/
   switch(temp)
   {
    case TIMEOUT:
    case ERROR  : printf("Destination node not ready.\n"); break;
    case ANS    : TransmitMode();
              temp=GetPointBuf(2);
              for(cnt=temp;cnt<cntbuff-temp;cnt++)
               if(tmpBuff[cnt]==0)
              tmpBuff[cnt]=' ';
              SendMessage(tmpBuff+temp,strlen(tmpBuff+temp)+1);
              ReceiveMode();
   }
  }
 }
 /*************************/
```

```
/*      SUBROUTINE          */
/***************************/
/* FORMAT  +------+--------+--------+-------------+
           | sync | Rx node | Tx node | Instruction |
           +------+--------+--------+-------------+ */
SendCommand(int node,int cmd)
{
 int cnt;

 SynChk.NodeTx=thisNodeAddress;
 SynChk.NodeRx=node;
 SynChk.Instruc=cmd;
 TransmitMode();
 SetNinthBit();
 for(cnt=0;cnt<sizeof(SynChk);cnt++)
 {
  SendChar(SynChk.SynIn[cnt]);
  delay(DelayCmd);
 }
 ClearNinthBit();
 delay(50);
}
/************************/
/*  +---------------------------+-----+
     |012345... (data sync)      | Cmd |
     +---------------------------+-----+ */
SendCmdSyn(int Cmd)
{
 int cnt;

 for(cnt=0;cnt<25;cnt++)
 {
  SendChar(cnt+'A');                    /* Sync */
  delay(DelayCmdSyn);
 }
 SendCmd(Cmd);
 delay(50);
}
/************************/
/*  +---------------------------+-----+
     |012345... (data sync)      | Cmd |
     +---------------------------+-----+
                output:return TIMEOUT if time out error
                     another return Commad */
int RecvCmdSyn()
{
 int code;

 do{code=RecvChar1(20);/*printf("%c",code);*/}while(!(ninthbit==1&&code<'0')&&
TimeOutErr!=FALSE);
 if(TimeOutErr!=TRUE) return(TIMEOUT);
 else                 return(code);
}
/************************/
/* FORMAT  +---------+-----------+
           | filesize| data....  |  for file open success
           +---------+-----------+
           +------------+
           | Error Code |          for file can't open
           +------------+
```

```
        */
SendFile(int NodeSend)
{
 int temp,cnt,code;
 char sum;

 switch(stat_send[NodeSend])
 {
  case 0:if(ModeTxRx==Tx) printf("Send File [%s].",filename);
       if((streamSend[NodeSend]=fopen(filename,"rb"))==NULL)
       {
        if(ModeTxRx==Tx) printf("File (%s) can't open...\n",filename);
        SendCmdSyn(ERROR);
        ReceiveMode();
        return;
       }
       SendCmdSyn(ANS);
       fseek(streamSend[NodeSend],0,SEEK_END);
       FileSizeTx.lon=ftell(streamSend[NodeSend]);        /* file length */
       fseek(streamSend[NodeSend],0,SEEK_SET);
       do
       {
        SendMessage(FileSizeTx.ary,4);
        ReceiveMode();
        delay(50);
        code=RecvCmdSyn();
        TransmitMode();
       }
       while(code!=ANS);
       if(ModeTxRx==Tx) printf("Size=%lu",FileSizeTx.lon);
       stat_send[NodeSend]=1;
       ReceiveMode();
       delay(200);
       break;
  case 1:temp = fread(SendBuff,1,BLOCK,streamSend[NodeSend]);
       TransmitMode();
       SendMessage(SendBuff,temp);
       ReceiveMode();
       delay(30);
       code=RecvCmdSyn();
       TransmitMode();
       if(ninthbit==1)
       {
        switch(code)
        {
         case ERROR: /* when ERROR */
                  fseek(streamSend[NodeSend],-BLOCK,SEEK_CUR);/* seek back for
resend again */
                  break;
         case TIMEOUT:
                  ReceiveMode();
                  code=RecvCmdSyn();
                  TransmitMode();
                  if(code==ERROR)
                  {
                   if(ModeTxRx==Tx) printf("Node Don't respond.");
                   stat_send[NodeSend]=0;
                   break;
                  }
                  break;
```

```
          case ANS:     if(ModeTxRx==Tx) printf("."); break;
          default :     if(ModeTxRx==Tx) printf("Send file Error\n");
                        stat_send[NodeSend]=0;
                        temp=0;
         }
        }
        if(temp!=BLOCK&&code!=ERROR)          /* End Of File */
        {
         if(ModeTxRx==Tx) printf("\n>");
         fclose(streamSend[NodeSend]);
         ReceiveMode();
         stat_send[NodeSend]=0;
        }
  }
 ReceiveMode();
}
/*****************************/
RecvFile(int NodeRecv)
{
 unsigned temp,temp1;

 switch(stat_recv[NodeRecv])
 {
  case 0:if(ModeTxRx==Rx) printf("Receive File [%s].",filename);
        temp=RecvCmdSyn();
        if(temp!=ANS) return;
        if(temp==TIMEOUT) {if(ModeTxRx==Rx) printf("File Error.");return;}
        if((streamRecv[NodeRecv]=fopen(filename,"wb"))==NULL)
        {
         if(ModeTxRx==Rx) printf("File (%s) can't open...\n",filename);
         return;
        }
        else
        {
         do
         {
          temp=RecvMessage(FileSizeRx.ary);
          TransmitMode();
          delay(50);
          if(temp==0)    SendCmdSyn(ERROR);
          else           SendCmdSyn(ANS);
          ReceiveMode();
         }
         while(temp==0);
         if(ModeTxRx==Rx) printf("Size=%lu",FileSizeRx.lon);
         stat_recv[NodeRecv]=1;
         count=0;
        }
        break;
   case 1:if(ModeTxRx==Rx) printf(".");
        temp=RecvMem(RecvBuff);
        if(temp!=0)
        {
         temp1=fwrite(RecvBuff,1,temp,streamRecv[NodeRecv]);
         if(temp1!=temp) {if(ModeTxRx==Rx) printf("Write file error.\n");}
         count=count+temp;
        }
        if(count>=FileSizeRx.lon)
        {
         if(ModeTxRx==Rx) printf("\n>");
```

```
        fclose(streamRecv[NodeRecv]);
        if(count!=FileSizeRx.lon)
         {if(ModeTxRx==Rx) printf("Size Error\n");}
        stat_recv[NodeRecv]=0;
        if(FlagRecvPrint==TRUE)
        {
         if((streamPnt=fopen(PrintName,"rb"))==NULL)
          {if(ModeTxRx==Rx) printf("File (%s) can't open...\n",filename);}
         else
          FlagPrint=TRUE;
         FlagRecvPrint=FALSE;
        }
       }
 }
}
/***************************/
int RecvMem(char Buffer[])          /* receive file save in memory (Buffer)
*/
{
 unsigned temp;

 delay(50);
 temp=RecvMessage(Buffer);
 if(temp==0)
 {
  if(ModeTxRx==Rx) printf("Er %lu\n",count);          /* data error */
  TransmitMode();
  delay(50);
  SendCmdSyn(ERROR);
  ReceiveMode();
 }
 else
 {
  TransmitMode();
  SendCmdSyn(ANS);
  ReceiveMode();
 }
 return(temp);
}
/*********@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*/
int ReceiveChar(int *ninthBit)
/*** RETURNS THE LOWER EIGHT BITS, AND SETS VARIABLE ninthBit***/
{
        /* assumes GENERAL_WORK_BANK */
        *ninthBit = inp(RECEIVE_FLAGS_ADDRESS) & 1;
        return inp(RECEIVE_BUFFER_ADDRESS);
}
/*****************************/
SendMessage(char buf[],unsigned int messageLength)
{
 unsigned int i;
 char sum=0;

 SendCmdSyn(STB);
 delay(50);
 for(i=0;i<messageLength;i++)  /*DATA CHARACTERS: 9TH BIT IS CLEAR*/
 {
  SendChar(buf[i]);
  sum=sum+buf[i];
 }
```

```
 SendChar(sum);
 SendCmd(EOB);
 delay(50);
}
/*****************************/
/* +---------+-----+---------------+----------+-----+
   | sync... | STB | data...       | sum(data) | EOB |
   +---------+-----+---------------+----------+-----+
/* input : data in buf[]
   output: format incomplete return 0
         format complete   return length of data */
int RecvMessage(char buf[])
{
 unsigned int i;
 char sum=0;

 if(RecvCmdSyn()!=STB) return(0);
 i=0;
 do
 {
  buf[i]=RecvChar1(20);
  if(ninthbit!=1) sum=sum+buf[i];
  i++;
 }
 while(ninthbit!=1&&TimeOutErr!=FALSE);
 if(ninthbit==1)
  if(buf[i-1]!=EOB) return(0);
  else
  {
   sum=sum-buf[i-2];
   if(sum==buf[i-2])
    return(i-2);
   else
    return(0);
  }
 else
  return(0);
}
/*****************************/
void TransmitMode(void)
/***ENABLE RS-485 DRIVERS***/
{
        /* assumes EMULATE_16450_BANK or GENERAL_WORK_BANK*/
        outp(MODEM_CONTROL_ADDRESS, TX_ENABLE_BIT);
             /* set transmit enable bit */
}
/*****************************/
void ReceiveMode(void)
/***AFTER TRANSMISSION IS COMPLETE, DISABLE RS-485 DRIVERS ***/
{
        /* assumes GENERAL_WORK_BANK */
        while((inp(GENERAL_STATUS_ADDRESS) & TxIR_BIT) == 0){}
             /* wait until transmission is complete */
        outp(MODEM_CONTROL_ADDRESS, 0);
             /* clear transmit enable bit, whatever it is */
}

void SetNinthBit(void) /* for transmission */
{
        /* assumes GENRAL_WORk_BANK*/
```

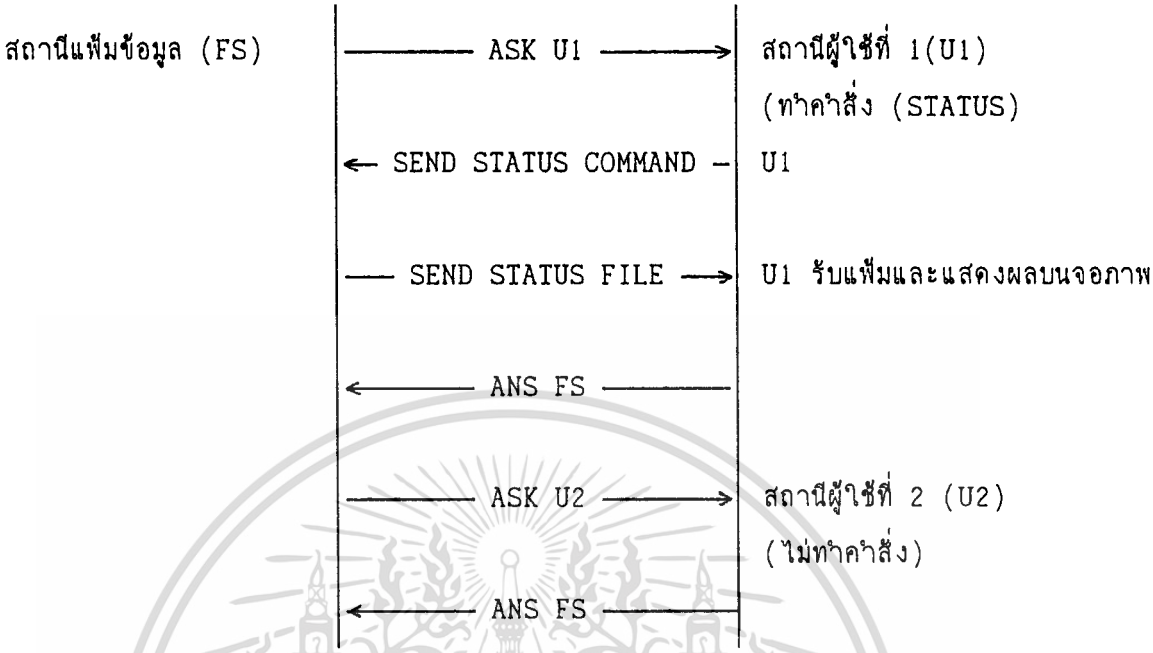
```
      outp(TRANSMIT_FLAGS_ADDRESS, 0x20);
}

void ClearNinthBit(void) /* for transmission */
{
        /* assumes GENERAL_WORK_BANK */
        outp(TRANSMIT_FLAGS_ADDRESS, 0);
}

void SendChar(uchar lowerEightBits)
/*SET OR CLEAR NINTH BIT BEFORE CALLING THIS PROCEDURE*/
{
     /*assumes GENERAL WORK _WORK_BANK*/
     while ((inp(FIFO_LEVEL_ADDRESS) & 0x07) == 4);
        /* wait until there is room in the transmitter FIFO*/
     outp(TRANSMIT_BUFFER_ADDRESS, lowerEightBits);
}

void EnableFifoMode(void)
{
     RegisterBank(INTERNAL_MODE_BANK);
     outp(INTERNAL_MODE_ADDRESS,0x08);       /*RX FIFO depth = bytes*/
     RegisterBank(FIFO_MODE_BANK);
     outp(FIFO_MODE_ADDRESS,0);     /*RX FIFO threshold = 1 byte*/
     RegisterBank(EMULATE_16450_BANK);
/*     printf("82510 FIFO mode enabled; trigger at 1 byte\n\r");*/
}
void SetBaudRate(double rate)
/* rate is in baud; maximum and minimum depend on the crystal frequency*/
{
     unsigned short divisor;
     uchar lcrValue;

     divisor=ComputeDivisor(rate, TRUE);
     if (divisor !=0)  {
        /* disable second divisor: */
        RegisterBank(BRGB_CONFIGURATION_BANK);
        outp(BRGB_CONFIGURATION_ADDRESS,0);
        /* set clock source to first divisor: */
        RegisterBank(CLOCKS_CONFIGURE_BANK);
        outp(CLOCKS_CONFIGURE_ADDRESS,0x50);
        RegisterBank(EMULATE_16450_BANK);
        /* set value of first divisor */
        lcrValue = (uchar)inp(LINE_CONTROL_ADDRESS);
        outp(LINE_CONTROL_ADDRESS,lcrValue | 0x80);   /* DLAB on */
        outp(DIVISOR_LOW_ADDRESS,(uchar)(divisor & 0x00FF));
        outp(DIVISOR_HIGH_ADDRESS,(uchar)((divisor>>8) & 0x00FF));
        outp(LINE_CONTROL_ADDRESS,lcrValue);    /* DLAB restored*/
     }
 }
 unsigned short ComputeDivisor(double baudRate, boolean suppressErrorMsg)
 {
     double rawDivisor, divisor, error;
     rawDivisor = crystalFrequency / (2 * 16e-6 * baudRate);
             /* factor of 2 is because we are using an external oscillator */
     divisor = floor(rawDivisor + .5);  /* round to nearest integer */
     if (!suppressErrorMsg) {
        if (divisor > 65535.0) {
           printf("A baud rate of %.11f is too low.\n",baudRate);
           return 0;
```

```
        }
        if (divisor < 1.0) {
             printf("A baud rate of %.1lf is too high.\n",baudRate);
             return 0;
        }
        error = fabs((divisor/rawDivisor) - 1.0);
        if (error > 0.02)
             printf("\aWARNING: the baud rate generation error will be %.2
lf%%.\n",
                  100.0 * error);
     }
     return (unsigned short)divisor;
}
```

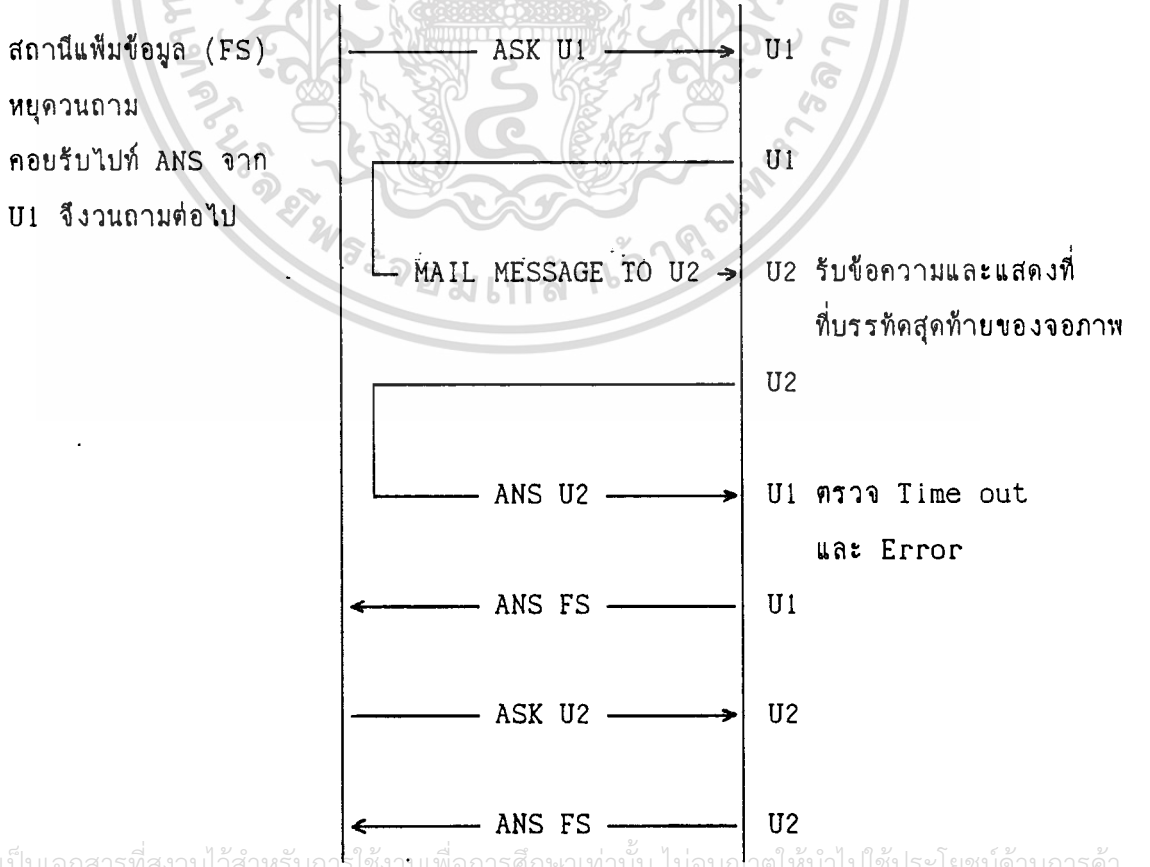
ภาคผนวก จ.

โปรโตคอลชาร์ทของโปรแกรม M-WARE

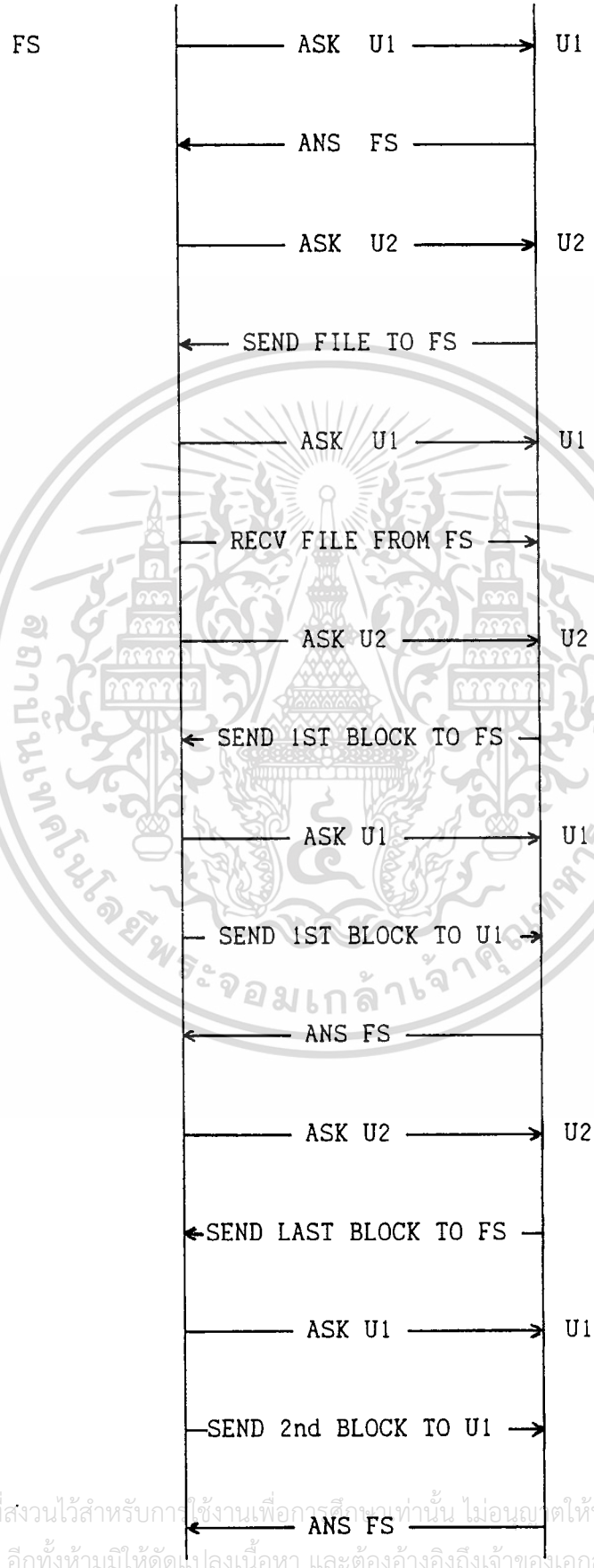
จ.1 โปรโตคอลชาร์ทของโปรแกรม M-WARE ในการทำคำสั่ง STATUS



จ.2 โปรโตคอลชาร์ทของโปรแกรม M-WARE ในการทำคำสั่ง MAIL

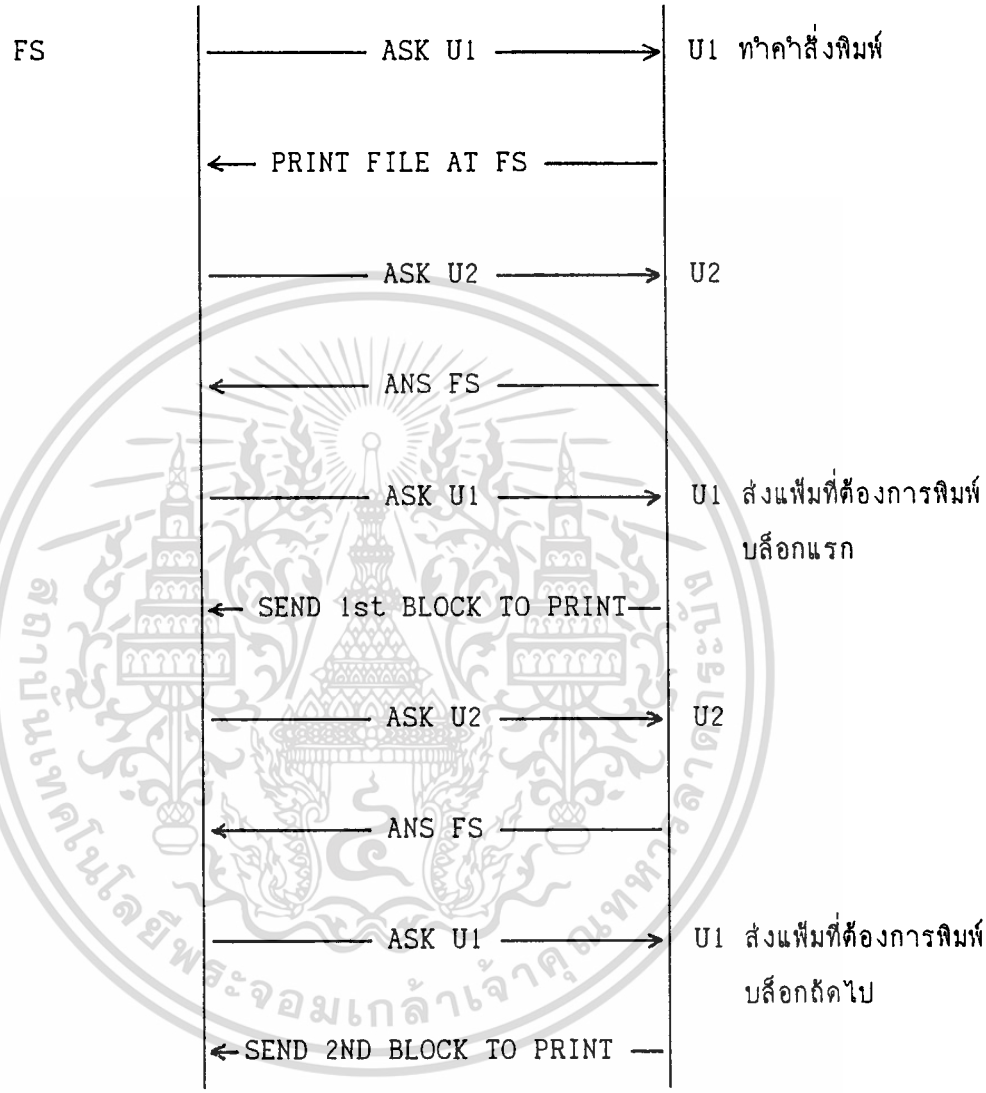


จ.3 โปรโตคอลชาร์ทของโปรแกรม M-WARE ในการทำคำสั่ง SEND และ RECV



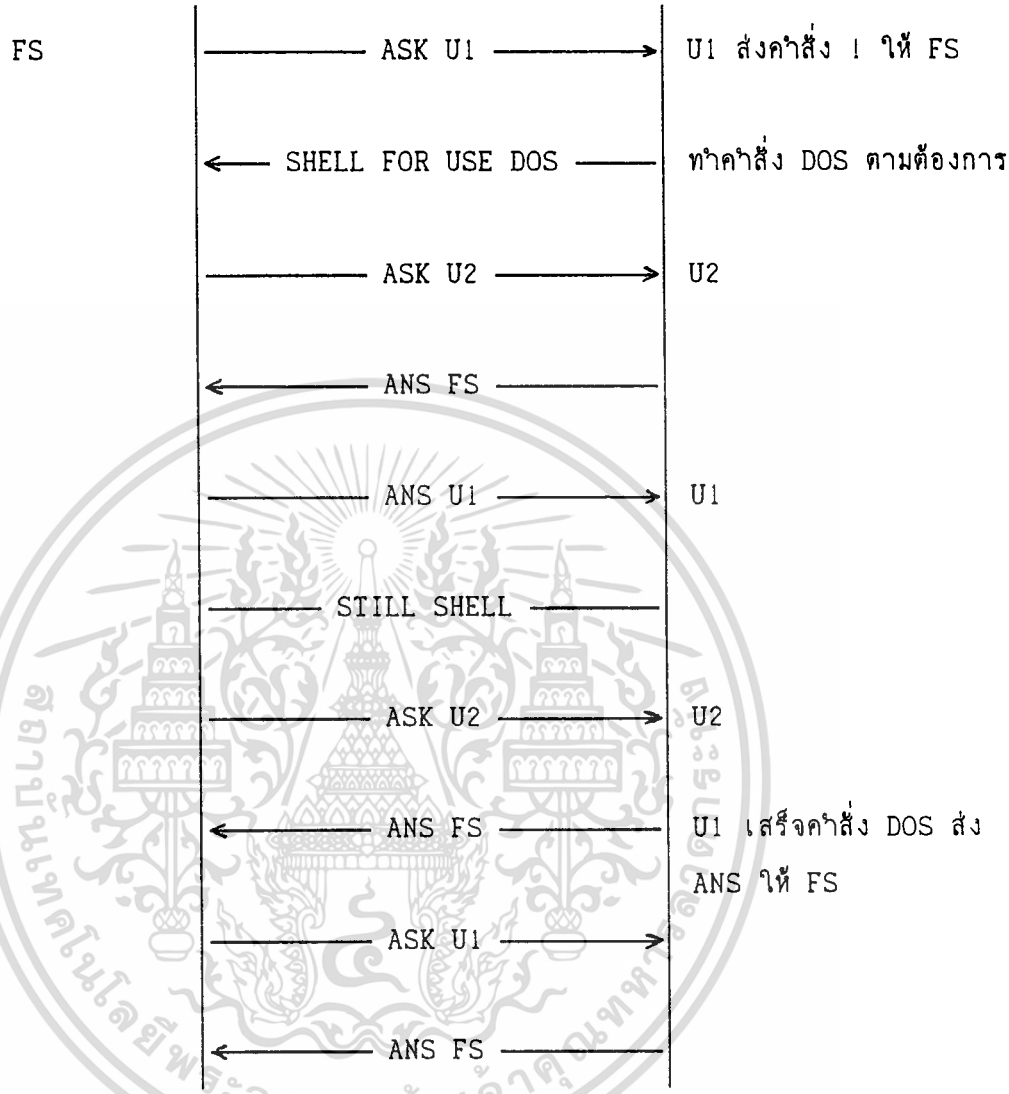
จ.4 โปรโตคอลชาร์ทของโปรแกรม M-WARE ในการทำคำสั่ง PRINT

การทำคำสั่ง PRINT ใช้โปรโตคอลเดียวกับการทำคำสั่ง SEND

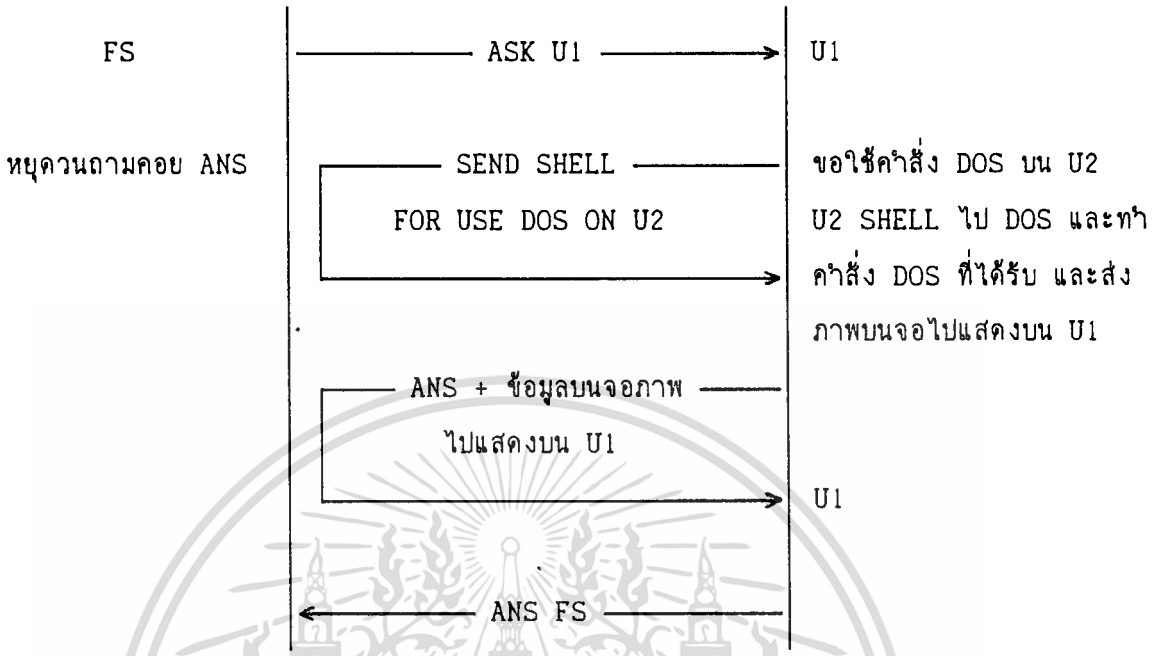


สถานีที่รับแฟ้มข้อมูลที่ต้องการพิมพ์ก็จะพิมพ์แฟ้มข้อมูลบล็อกนั้นออกที่เครื่องพิมพ์ที่ติดตั้งอยู่ ถ้าเครื่องพิมพ์ไม่พร้อมก็จะส่ง Error กลับไปถ้าเครื่องพิมพ์พร้อมก็จะส่ง ANS กลับไปและรอรับพิมพ์แฟ้มบล็อกต่อไป

จ.5 โปรโตคอลชาร์ทของโปรแกรม M-WARE ในการทำคำสั่ง DOS บนสถานีผู้ใช้เอง



จ.6 โปรโตคอลชาร์ทของโปรแกรม M-WARE ในการทำคำสั่ง DOS บนสถานีผู้ใช้อื่น



ประวัติผู้เขียน

ชื่อผู้เขียน	นายวีระ ธรรมจรัส
วันเดือนปีเกิด	วันที่ 6 มีนาคม พ.ศ. 2507
สถานที่เกิด	จังหวัด กรุงเทพมหานคร
วุฒิการศึกษาระดับปริญญาตรี	วิทยาศาสตรบัณฑิต สาขาวิชาฟิสิกส์
สถานที่สำเร็จการศึกษา	มหาวิทยาลัยศิลปากร
ปีที่สำเร็จการศึกษา	ปีการศึกษา 2530
ผลงานทางวิชาการที่ได้รับการตีพิมพ์	เรื่องโครงข่ายคอมพิวเตอร์แบบท้องถิ่นโดยใช้เทคนิคเนกาทีฟพัลส์ ตีพิมพ์ในเอกสารการประชุมใหญ่วิชาการทางวิศวกรรมของวิศวกรรมสถานแห่งประเทศไทยในพระบรมราชูปถัมภ์ ปี พ.ศ. 2533
ประสพการณ์การทำงาน	อาจารย์พิเศษภาควิชาอิเล็กทรอนิกส์และภาควิชาคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
อาชีพปัจจุบัน	ดำเนินธุรกิจ ตำแหน่งกรรมการผู้จัดการ บริษัท ปฐมยูนิท็อป จำกัด