ปีการศึกษา 2532

ปริญญานิพนธ์เรื่อง

การสร้างระบบสื่อสารข้อมูลในเครือข่ายไมโครคอมพิวเตอร์แบบบัส

A Construction On Table Communication In Bus Type Microcomputer Network

ผู้จัดทำ

นาย ปิยะ ตัณฑวิเชียร รหัส 29.1105

นาย ปิยะมิตร ฉัตรชาตรี รหัส 29.1111

นาย วรากร เกษมสุวรรณ์ รหัส 29.1179

อาจารย์ที่ปรึกษา

ผศ.ดร. รัตติกร วรากูลศิริพันธุ์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้า

เจ้าคุณทหารลาดกระบัง

ปริญญานิพนธ์ ปีการศึกษา 2532

เรื่อง การสร้างระบบสื่อสารข้อมูลในเครือข่ายไมโครคอมพิวเตอร์แบบบัส

A Construction On Table Communication.

In

Bus Type Microcomputer Network

ผู้จัดทำ

นาย ปิยะ ตัณฑวิเชียร รหัส 29.1105

นาย ปิยะมิตร ฉัตรชาตรี รหัส 29.1111

นาย วรากร เกษมสุวรรณ์ รหัส 29.1179

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหาร ลาดกระบัง

.....

(ผศ.ดร. รัตติกร วรากูลศิริพันธุ์)

อาจารย์ที่ปรึกษา

การสร้างระบบการสื่อสารข้อมูลในเครือข่ายไมโครคอมพิวเตอร์แบบบัส

A Construction On Data Communication

In

Bus type Microcomputer Network

นาย ปิยะ ตัณฑวิเชียร 29.1105

นาย ปิยะมิตร ฉัตรชาตรี 29.1111

นาย วรากร เกษมสุวรรณ์ 29.1179

อาจารย์ที่ปรึกษา:

ผศ.ดร. รัตติกร วรากูลศิรินันท์

ปีการศึกษา 2532

บทคัดย่อ

ปริญญานิพนธ์ นี้แสดงถึงการสร้างระบบสื่อสารข้อมูลไมโครคอมพิวเตอร์ IBM-PC รูปโครงข่ายแบบบัส เราทำการออกแบบสร้างฟังค์ชั่นการทำงานที่จำเป็นเพื่ออำนวยความสะดวกในการสื่อสารข้อมูลแก่ผู้ใช้ ระบบมีการใช้รหัสผ่าน มีการกำหนดระดับความสำคัญให้กับผู้ใช้แต่ละราย (USER LEVEL) นั่นคือเป็นการกำหนดความสามารถในการเข้าถึงแฟ้มข้อมูล ระบบถูกออกแบบให้มีความอ่อนตัวสูง ง่ายต่อการพัฒนาขีดความสามารถ สามารถเพิ่มฟังค์ชั่นการทำงานเพื่อเพิ่มสมรรถภาพการทำงานของระบบให้สูงขึ้นในอนาคตข้างหน้า อีกทั้งระบบนี้ง่ายต่อการประยุกต์ใช้งานที่เกี่ยวข้องกับการสื่อสารข้อมูล อาทิ ในระบบงานไปรษณีย์อิเลคโทรนิคส์ (ELECTRONIC MAIL) , ระบบคอมพิวเตอร์เพื่อช่วยการศึกษาในชั้นเรียน (MICROCOMPUTER FOR CLASS ROOM) ,การสื่อสารข้อมูลภายในสำนักงาน เป็นต้น

A Construction On Data Communication

In

Bus Type Microcomputer Network

Mr. Piya Tanthawichian 29.1105

Mr. Piyamit Chatchartree 29.1111

Mr. Varakorn Kasamsuwan 29.1179

Advisor:

Assistant Professor Dr.Ruttikorn Varakulsiripunth

Academic Year 1989

Abstract

This paper shows the construction of data communication in bus type microcomputer IBM-PC network . We built various necessary communication functions. This system uses password, user level. We designed the data communication system to be successful in high flexibility . This system is easy to develope the capability , by adding the function operation in the future. We present the operating system in a very simple form. It is easy to use and apply in data communication tasks for example ELECTRONICS MAIL ,. MICROCOMPUTER FOR CLASSROOM , THE DATA COMMUNICATION SYSTEM IN THE OFFICE .

สารบัญ

บทที่ 1

บทนำ

ในปัจจุบัน ไมโครคอมพิวเตอร์มีการใช้อย่างแพร่หลายในกิจการทั่วไป กล่าวคือ นับตั้งแต่เกิดเทคโนโลยีทาง VLSI(VERY LARGE SCALE INTEGRATE CIRCUIT) ซึ่งทำให้สามารถผลิตชิพไมโครโปรเซสเซอร์ที่มีประสิทธิภาพดีกว่าเครื่องเมนเฟรมในด้านความเร็วประมาณ 10 เท่า ในขณะที่ราคาถูกกว่านับ 1000เท่า

จากแนวโน้มในระยะหลังนี้พบว่าเริ่มมีการนำไมโครคอมพิวเตอร์มาเชื่อมต่อกันเป็น NETWORK เป็นการเพิ่มประสิทธิภาพการทำงานของไมโครคอมพิวเตอร์เพิ่มมากขึ้น กล่าวคือเป็นการใช้งานในลักษณะที่ผู้ใช้หลายฝ่ายสามารถที่จะใช้เครื่องมือที่มีราคาแพงร่วมกันได้ แทนที่จะเป็นระบบ STAND ALONE ในระบบเก่าซึ่งมีข้อได้เปรียบอันได้แก่

ด้านความเร็ว เนื่องจากสัญญาณของคอมพิวเตอร์เป็นสัญญาณทางวิเลคโทรนิคส์ซึ่งมีความเร็วซึ่งเทียบเท่ากับแสงดังนั้นในการสื่อสารข้อมูลจำนวนมากๆ จะใช้เวลาสั้นมากเมื่อเทียบกับการสื่อสารวิธีอื่นๆ

ด้านความเชื่อถือ ในระบบไมโครคอมพิวเตอร์นั้นมีวิธีการตรวจสอบข้อมูลว่ามีข้อผิดพลาดเกิดขึ้นหรือไม่ เช่น มีการตรวจเช็คพาริตี้และการตรวจเช็คผลรวมเป็นต้น ถ้ามีการตรวจสอบข้อมูลพบว่ามีข้อผิดพลาด ไมโครคอมพิวเตอร์จะมีวิธีทำการแก้ไขข้อมูลให้ถูกต้อง

ด้านการจัดเก็บข้อมูลและความสะดวกในการค้นหาข้อมูล ในระบบไมโครคอมพิวเตอร์จะมีการบันทึกข้อมูลในตัวกลางที่มีความหนาแน่นสูงมาก ทั้งมีขนาดเล็ก เช่น แผ่นจานแม่เหล็ก , แผ่นแม่เหล็ก เป็นต้น (floppy disk ขนาด ห้านิ้วครึ่ง สามารถบันทึกข้อมูลนับ ล้านตัวอักษร) และไมโครคอมพิวเตอร์สามารถนำข้อมูลที่ได้เหล่านั้นมาทำการประมวลผล เช่นในการค้นหาข้อมูลเป็นต้น

ด้านค่าใช้จ่าย ไมโครคอมพิวเตอร์สามารถทำงานได้ในหลายๆด้าน และใช้เวลาสั้นมาก เมื่อเปรียบเทียบกับวิธีการอื่นๆ

จากข้อได้เปรียบที่กล่าวมาแล้วเบื้องต้น จึงมีการนำไมโครคอมพิวเตอร์มาต่อเป็นโครงข่ายเพื่อใช้ในการติดต่อสื่อสารข้อมูล แต่อย่างไรก็ตามการจัดตั้งโครงข่ายไมโครคอมพิว

เตอร์นั้นมีหลายรูปแบบ อาทิเช่น แบบวงแหวน(RING TYPE), แบบดาว (STAR TYPE), แบบบัส(BUS TYPE) เป็นต้น ดังนั้น จึงต้องมีการพิจารณาเลือกรูปแบบของโครงข่าย ให้เหมาะสมกับงานที่ประยุกต์ใช้ เช่น ด้านราคา, ด้านความเร็ว, ความสะดวกในการขยายระบบ เป็นต้น

ซึ่งข้อได้เปรียบ ข้อเสียเปรียบของรูปแบบโครงข่ายแต่ละชนิดจะกล่าวไว้ในบทหลัง สำหรับวิทยานิพนธ์ฉบับนี้ กล่าวถึงการนำโครงข่ายไมโครคอมพิวเตอร์ มาประยุกต์ใช้งานในด้านการสื่อสาร

บทที่ 2

ทฤษฎีของการสื่อสารข้อมูล

ในบทนี้จะเป็นการกล่าวถึงหลักการเบื้องต้นของการสื่อสารข้อมูลโดยอาศัยไมโครคอมพิวเตอร์ ซึ่งผู้อ่านสามารถหาอ่านเพิ่มเติมได้จากหนังสืออ้างอิงท้ายเล่ม

2.1 องค์ประกอบเบื้องต้นในการสื่อสารข้อมูล

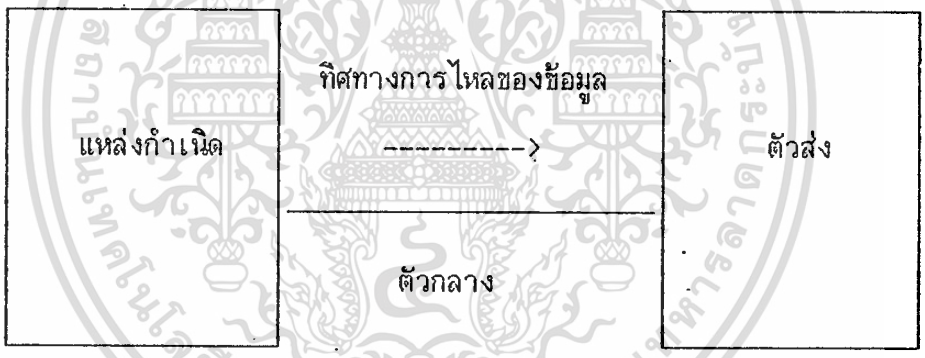
ในการสื่อสารข้อมูลต้องมีปัจจัยอย่างน้อย 3 ประการคือ

1) แหล่งกำเนิด (source)

2) ตัวกลาง (medium)

3) ตัวรับ (receiver)

สามารถแสดงความสัมพันธ์ได้ดังรูป



รูป 2.1 แสดงองค์ประกอบในการสื่อสาร

2.2 รูปแบบการถ่ายโอนข้อมูล วิธีการโอนถ่ายข้อมูลระหว่างไมโครคอมพิวเตอร์สามารถแบ่งออกได้เป็นสองวิธีคือ

2.2.1) การถ่ายโอนแบบขนาน

ในระบบไมโครคอมพิวเตอร์ทั่วไป การส่งข้อมูลระหว่างอุปกรณ์ต่างๆ ที่อยู่บนแผ่นวงจรพิมพ์ล้วนแล้วแต่เป็นการส่งแบบขนานทั้งสิ้น การส่งในลักษณะนี้ทำได้โดยส่งข้อมูลออกมาพร้อมๆ กันทุกบิท ดังนั้นช่องการเดินทางของข้อมูลจึงมีเท่ากับจำนวนบิท เช่นถ้าเป็นไมโครคอมพิวเตอร์

8 บิท สายส่งข้อมูลก็จะมี 8 เส้นด้วย ซึ่งในความเป็นจริงแล้วจะต้องมีสายข้อมูลอย่างน้อยอีก 2 เส้น ซึ่งได้แก่สาย DATA READY กับสายดิน

ลักษณะการโอนถ่ายข้อมูลแบบขนานจะเป็นดังรูป 2.2

2.2.2) การถ่ายโอนแบบอนุกรม

การส่งแบบอนุกรมนี้ ข้อมูลแต่ละบิทจะถูกส่งเรียงกันออกไปเป็นลำดับต่อเนื่องกันทีละบิทระหว่างจุดส่งและจุดรับ จากการส่งในลักษณะนี้จึงใช้สายส่งข้อมูลเพียงคู่เดียวเท่านั้น

การถ่ายเทข้อมูลแบบอนุกรมนี้แม้จะช้าแต่ก็เหมาะกับการส่งในระยะทางไกลๆ ทั้งนี้เป็นเพราะประหยัดสายส่งได้เป็นจำนวนมากได้ ลักษณะการโอนถ่ายข้อมูลจะเป็นดังรูป 2.3 .

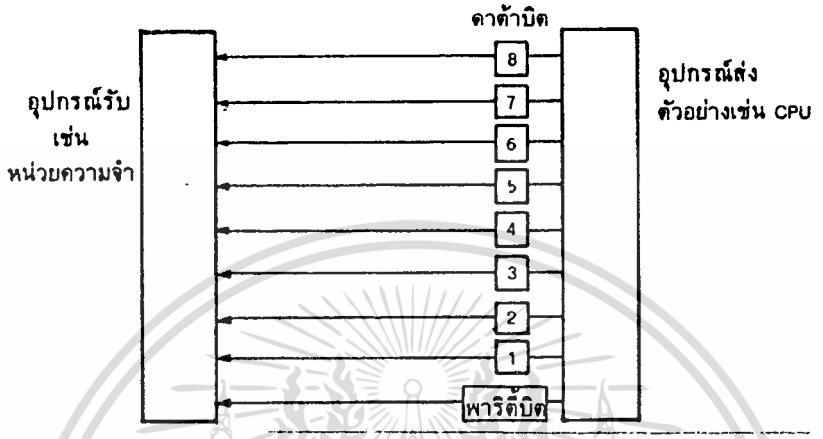
เนื่องจาก ความจริงแล้วภายในไมโครคอมพิวเตอร์มีการรับส่งข้อมูลแบบขนาน ดังนั้นจำเป็นต้องมีการแปลงกลับให้เหมาะสมเสียก่อน โดยการแปลงกลับสัญญาณให้เป็นแบบขนาน ซึ่งมีรูปแบบในการรับ (protocal)

ในโครงงานนี้เราใช้การติดต่อแบบอนุกรมเป็นหลัก ดังนั้นจะขอกล่าวถึงการติดต่อแบบนี้โดยละเอียด การสื่อสารข้อมูลแบบอนุกรมนี้แบ่งได้ออกเป็น 2 แบบ ด้วยกันคือ

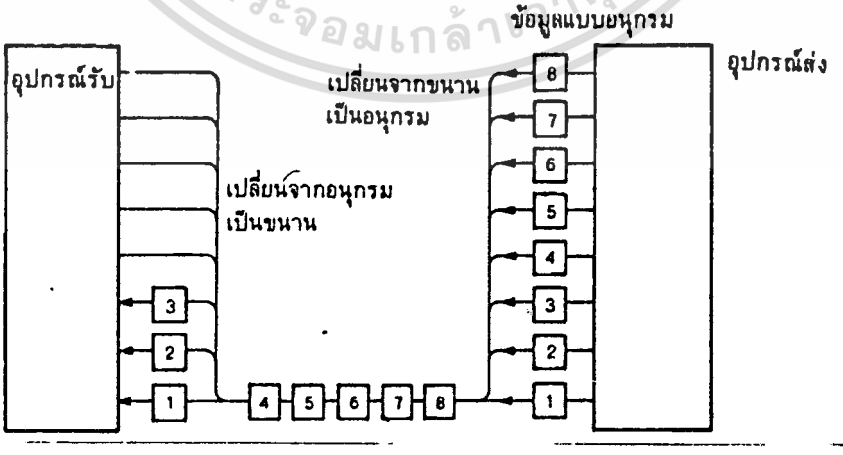
- การรับส่งแบบซิงโครนัส (synchronous protocol)
- การรับส่งแบบอะซิงโครนัส (asynchronous protocol)

<u>การรับส่งข้อมูลแบบอะซิงโครนัส (asynchronous protocol)</u> การส่งข้อมูลแบบนี้ข้อมูลที่ถูกส่งออกไปจะมีลักษณะเป็นบล็อกๆ ซึ่งในแต่ละบล็อกจะประกอบด้วยบิทเริ่มต้น(start bit) , ส่วนของข้อมูล , บิทสิ้นสุดของข้อมูล (stop bit) โดยบิทเริ่มต้นจะแสดงถึงการมาหรือการเริ่มต้นของข้อมูล แล้วตามด้วยส่วนของกลุ่มข้อมูล ในบางกรณีอาจจะมีการเพิ่มบิทพาริตี้เข้าไปเพื่อใช้ตรวจสอบความถูกต้องของข้อมูล ส่วนในบิทสุดท้ายนั้นจะเป็นการบอกว่าข้อมูลได้สิ้นสุดเพียงเท่านี้

- บิทเริ่มต้น ในโปรโตคอลของการส่งข้อมูลอนุกรมแบบอะซิงโครนัส กำหนดให้สถานะมาร์ค (marking state) เป็นสัญญาณลอจิก 1 เมื่อทางด้านส่งจะทำการส่งข้อมูลก็จะส่งบิทเริ่มต้น หรือสัญญาณโลจิก 0 ทำให้ทางด้านรับสามารถตรวจสถานะของสายส่งว่าในขณะนั้นกำลังจะมีข้อมูลส่งมา ปัญหาอันหนึ่งที่พบบ่อยๆในการส่งสัญญาณข้อมูลคือสัญญาณสไปรค์ (spike) ซึ่งทำให้สถานะลอจิกสั้นเกินไป ปัญหานี้แก้ได้โดยจะมีส่วนของวงจรสไปรค์ดีเทคชั่น (spike detection)



รูป 2.2 การโอนถ่ายข้อมูลแบบ ขนาน



รูป 2.3 การโอนถ่ายข้อมูลแบบ อนุกรม

- บิทข้อมูล หลังจากที่ด้านรับสามารถดีเท็กซ์สัญญาณบิทเริ่มต้นได้แล้ว ก็จะทำการเซ็ทสถานะของชิฟรีจิสเตอร์ ให้พร้อมที่จะรับบิทข้อมูลได้ โดยบิทข้อมูลจะมีจำนวนบิทเป็น 5,6,7 หรือ 8 บิท ขึ้นกับจำนวนคาร์แรกเตอร์ที่ใช้

- บิทพาริตี้ จะทำหน้าที่ในการบอกให้ส่วนรับข้อมูลทราบว่าข้อมูลที่รับเข้ามาผิด หรือไม่บิทพาริตี้นี้จะถูกส่งออกมาพร้อมกับบิทข้อมูล ซึ่งบิทนี้จะเป็น 1 หรือ 0 นั้น ขึ้นกับข้อมูลที่ส่งออกมาว่ามีจำนวนบิท 1 เป็นจำนวนคี่หรือคู่ และยังขึ้นกับอุปกรณ์รับส่งข้อมูลด้วยว่าถูกออกแบบ (โปรแกรม) ไว้ให้รับส่งบิทพาริตี้ในลักษณะพาริตี้คู่ หรือคี่อีกด้วย

การรับส่งข้อมูลแบบซิงโครนัส (synchronous protocol)

ในการส่งข้อมูลแบบซิงโครนัสนั้น ต่างจากการส่งแบบอะซิงโครนัส คือการส่งแบบนี้จะไม่มีการส่งสัญญาน start bit หรือ stop bit ออกไปซิงค์กับตัวรับ แต่จะมีการส่งชุดสถานะของสัญญานที่เรียกว่า sync character โดยจะส่งสัญญานนี้ออกไปเรื่อยๆจนกว่าฝ่ายรับจะ sync กับตัวส่งได้

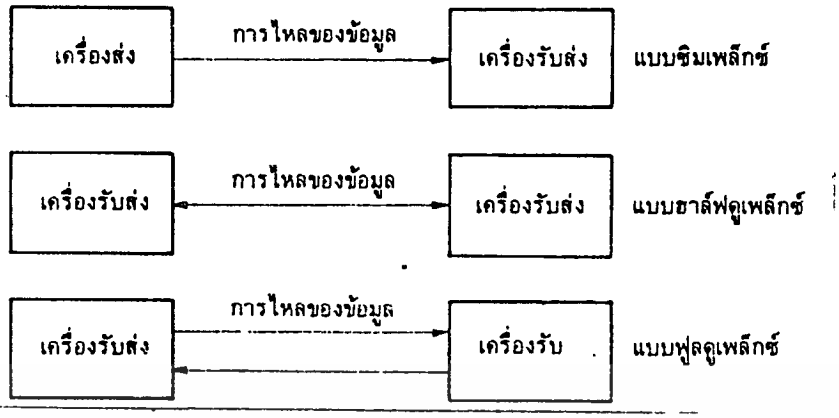
ดังนี้นจะเห็นได้ชัดว่า ในการส่งข้อมูลแบบซิงโครนัสนั้นจะมีความรวดเร็วกว่าการส่งแบบอะซิงโครนัสมากเพราะไม่จำเป็นต้องตรวจสอบบิทแบบในกรณีของการส่งแบบ อะซิงโครนัส แต่อย่างไรก็ตามความผิดพลาดจากการส่งแบบซิงโครนัสย่อมต้องมีมากกว่าทั้งนี้ก็เนื่องมาจากไม่มีการตรวจสอบบิทพาริตี้

สิ่งที่สำคัญมากสิ่งหนึ่งในการส่งข้อมูลแบบอนุกรม(ทั้งแบบซิงโครนัสและอะซิงโครนัส) ก็คือ baud rate ความถี่ที่ใช้ในการส่งข้อมูลซึ่งจะต้องสัมพันธ์กันระหว่างอุปกรณ์ที่ทำการรับและส่งข้อมูล ความถี่นี้เราเรียกว่า baud rate หรือ อาจกล่าวได้ว่า

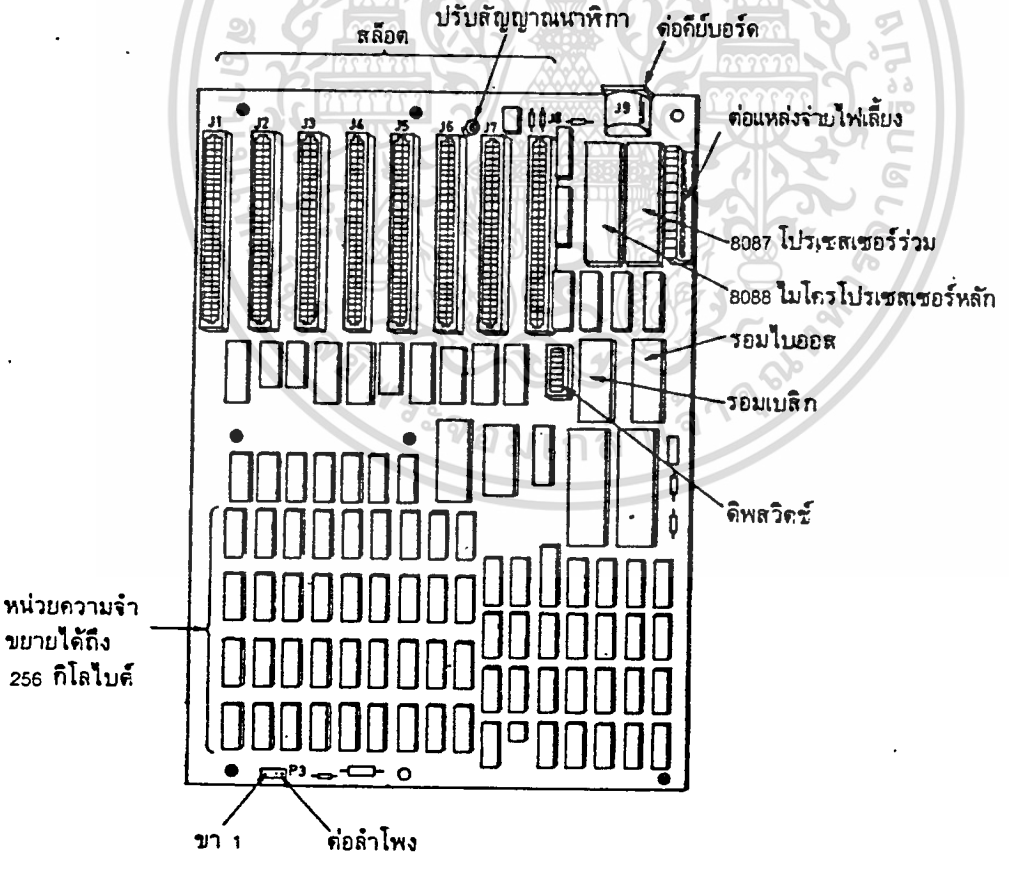
baud rate คือ " อัตราการรับส่งข้อมูลเป็นบิทต่อวินาที " ถ้าหากว่าเครื่องส่งใช้ buad rate ไม่สัมพันธ์กับเครื่องรับแล้ว ก็จะทำให้การรับส่งข้อมูลเกิดการผิดพลาดขึ้นได้

จากรูปแสดง จะเห็นว่ามีทิศทางของการส่งข้อมูลต่างๆ กัน 3 ลักษณะ คือ แบบทิศทางเดียวหรือซิมเพล็กซ์, แบบสองทางแต่ตอบโต้กันไม่ได้ หรือ แบบฮาล์ฟดูเพล็กซ์ อีกแบบที่เหลือคือการส่งแบบฟูลดูเพล็กซ์ ที่ทั้งสองทางสามารถตอบโต้กันได้

วิธีที่ง่ายที่สุดก็คือ การส่งข้อมูลแบบทางเดียว เช่น ส่งข้อมูลจากจุด a ไปจุด b ในกรณีนี้จุด a จะเป็นเครื่องส่งและ b จะเป็นเครื่องรับเท่านั้น



รูปแสดง การส่งข้อมูลแบบต่าง ๆ



รูปแสดงโครงสร้างพื้นฐานของ (Main board)

วิธีต่อมาคือการส่งข้อมูลแบบฮาล์ฟดูเพล็กซ์ การส่งแบบนี้จุด a และจุด b สามารถส่งข้อมูลถึงกันได้ คืออาจเป็นการส่งข้อมูลจาก a ไป b หรือจาก b ไป a แต่การส่งไปมานี้ต้องเป็นคนละเวลากัน

วิธีสุดท้ายคือการส่งข้อมูลแบบฟูลดูเพล็กซ์ กล่าวคือจุด a และจุด b สามารถที่จะตอบโต้กันได้โดยที่จุดใดจุดหนึ่งอาจเป็นจุดส่งหรือจุดรับในเวลาเดียวกัน

ตัวอย่างของระบบที่ใช้การติดต่อในลักษณะดังกล่าวนี้ได้แก่ ระบบซิมเพล็กซ์ ได้แก่การส่งกระจายเสียงวิทยุหรือโทรทัศน์ หรือ การส่งข้อมูลจากคอมพิวเตอร์ไปให้เครื่องพิมพ์

ระบบฟูลดูเพล็กซ์ได้แก่การสื่อสารทางโทรศัพท์และการสื่อสารคอมพิวเตอร์

บทที่ 3

ความรู้เบื้องต้นเกี่ยวกับส่วนประกอบของ IBM PC/XT

ในบทนี้จะกล่าวถึงความรู้ทั่วไปทางด้าน hardware ที่เกี่ยวข้องกับโครงงานนี้ โดยเฉพาะในเรื่องการควบคุมทางการสื่อสาร รวมถึงเรื่องเกี่ยวกับไบออส และ วิธี การ interupt โดยในรายละเอียดส่วนนี้จะสามารถใช้ได้กับเครื่องเทียบเท่า(compatible)ได้ด้วย

3.1 **mainboard** ถือว่าเป็นส่วนที่สำคัญที่สุดของเครื่อง ทั้งนี้เพราะเป็นที่อยู่ของหน่วยประมวลผลกลางและชิพซัพพอร์ทต่างๆ รวมทั้งหน่วยความจำ โดยได้ แสดงไว้ดังรูปที่ 3.1

3.2 **โครงสร้างหน่วยความจำ** การจัดการโครงสร้างหน่วยความจำของระบบเป็นสิ่งที่สำคัญมาก ในการเขียน software

ในรูปที่ 3.2แสดงการจัดสรรหน่วย ความจำของเครื่อง IBM PC/XT

3.3 **CHIP SUPPORT** นอกจากหน่วยประมวลผลกลางและหน่วยความจำแล้ว องค์ประกอบของเครื่องไมโครคอมพิวเตอร์ยังประกอบด้วย chip สนับสนุนมากมาย chip สนับสนุนเหล่านี้เราจะนำ chip ตัวที่มีบทบาทสำคัญมาอธิบายพอสังเขปดังต่อไปนี้

8259 INTERRUPT CONTROLLER

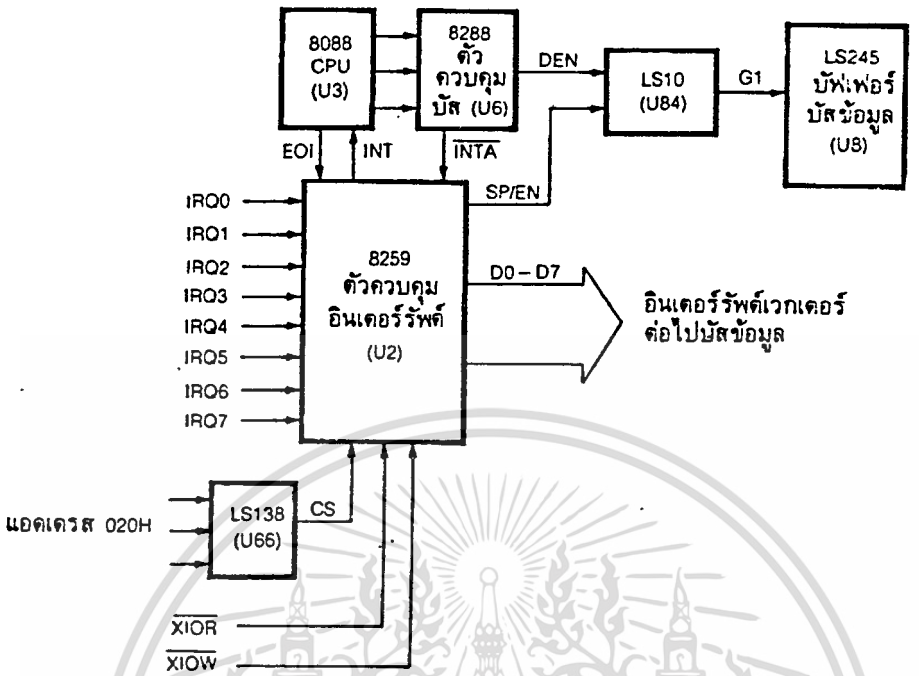
8259 เป็นตัวขัดจังหวะการทำงานของหน่วยประมวลผลกลางเพื่อให้มาทำ งานบางอย่างก่อน หลังจากนั้นจึงกลับมาทำงานที่ได้ค้างไว้ โดยก่อนที่จะทำการทำ งานใดๆนั้น ตัว 8259 จะทำการจัดลำดับความสำคัญก่อน จากนั้นจึงค่อยส่งสัญญาน ขัดจังหวะหน่วยประมวลผลกลางอีกต่อหนึ่ง ในการขัดจังหวะนั้นเมื่อ 8088 ได้รับการขัดจังหวะนั้น ตัว 8088 จะส่งสัญญาน s0-s2 ให้แก่ 8288 เพื่อสร้างสัญญาณ int ตอบสนองการขัดจังหวะ ของ 8259 จากนั้น 8259 จึงจะส่งสัญญาณ vector การขัดจังหวะ ให้แก่ 8088 ตาม รหัสตัวเลขที่ระบุหลังสาย IRQ (interupt request) ของอุปกรณ์ที่ขอขัดจังหวะ เมื่อ 8088 ได้รับการขัดจังหวะไปแล้วก็จะทราบได้ว่าต้องไปทำงาน ที่ตำแหน่งใด

Start Address Decimal.	Start Address Hex	Function
0	00000	128-256K Read/Write Memory on System Board
16K	04000	
32K	08000	
48K	0C000	
64K	10000	
80K	14000	
96K	18000	
112K	1C000	
128K	20000	
144K	24000	
160K	28000	
176K	2C000	
192K	30000	
208K	34000	
224K	38000	
240K	3C000	
256K	40000	384K R/W Memory Expansion in I/O Channel
272K	44000	
288K	48000	
304K	4C000	
320K	50000	
336K	54000	
352K	58000	
368K	5C000	
384K	60000	
400K	64000	
416K	68000	
432K	6C000	
448K	70000	
464K	74000	
480K	78000	
496K	7C000	
512K	80000	
528K	84000	
544K	88000	
560K	8C000	
576K	90000	
592K	94000	
608K	98000	
624K	9C000	

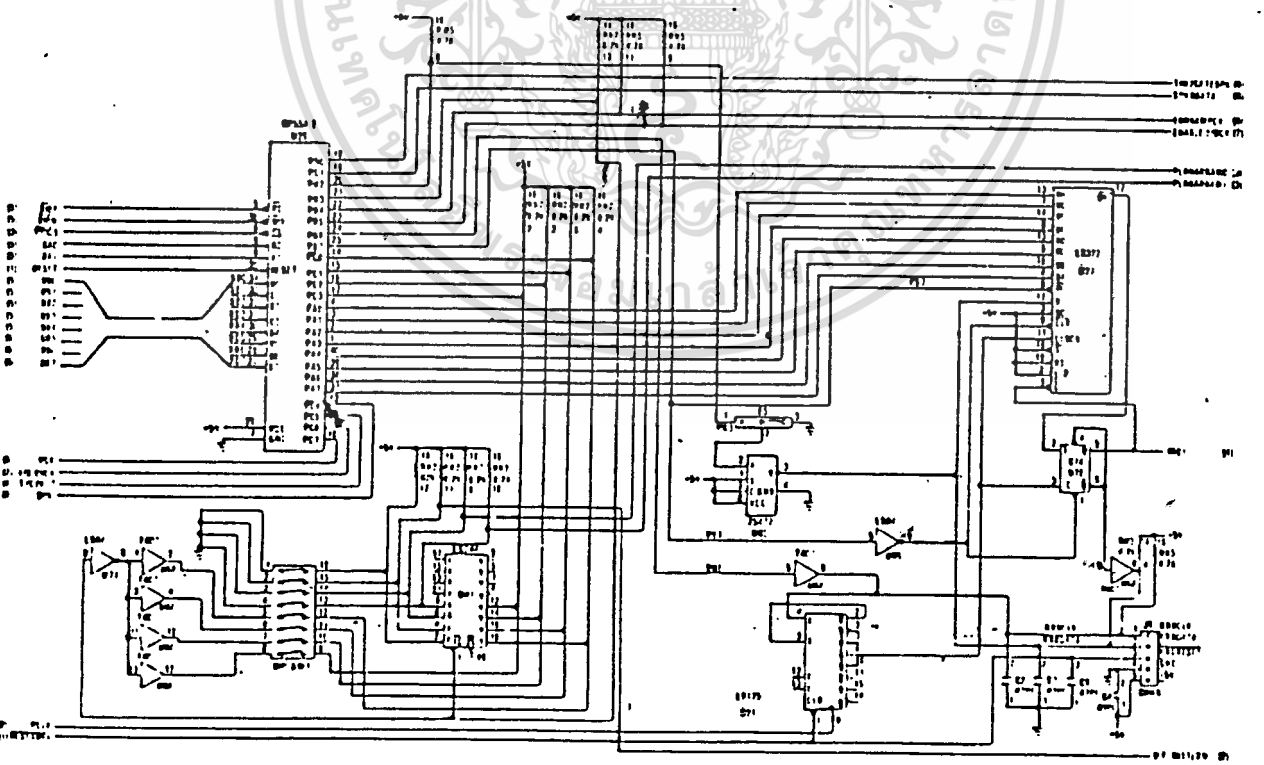
รูปแสดงโครงสร้างหน่วยความจำ ของ Main board.

Start Address Decimal	Start Address Hex	Function
640K	A0000	128K Reserved
656K	A4000	
672K	A8000	
688K	AC000	
704K	B0000	Monochrome
720K	B4000	
736K	B8000	Color/Graphics
752K	BC000	
768K	C0000	192K Read Only Memory Expansion and Control
784K	C4000	
800K	C8000	Fixed Disk Control
816K	CC000	
832K	D0000	
848K	D4000	
864K	D8000	
880K	DC000	
896K	E0000	
912K	E4000	
928K	E8000	
944K	EC000	
960K	F0000	64K Base System ROM BIOS and BASIC
976K	F4000	
992K	F8000	
1008K	FC000	

(ต่อ)



รูปแสดงการขัดจังหวะ ที่เชื่อมต่อกับ 8259



8253 programmable interval time processer

8253 เป็นอุปกรณ์ที่สำคัญอีกตัวหนึ่งที่อยู่บนเมนบอร์ด โดยทำหน้าที่เกี่ยวกับ (timing)ภายในตัวมันมีตัวนับ (counter) ที่ทำงานแยกกันอย่างอิสระอยู่ 3 ตัว

8255(LARGE SCALE INTEGRATED CIRCUIT)

8255 เป็นอุปกรณ์ LSI ชนิดหนึ่งที่บรรจุอยู่ใน package ขนาด 40 ขา แบบ DIP(DUAL IN LINE PACKAGE) จัดทำโดยบริษัท INTEL COOPERATION ผู้ผลิตไมโครโปรเซสเซอร์ 8080 ดังรูปแสดงบล็อกไดอะแกรมของ 8255

บล็อกกลุ่มแรกที่เราจะพูดถึงนี้ ได้แก่ บล็อกจำนวน 4 บล็อก ที่อยู่ทางด้านขวาของรูป ซึ่งจะเป็นส่วนที่เชื่อมต่อกับอุบกรณ์ภายนอกอื่นๆ โดยมีสาย PA0-PA7, PB0-PB7 และ PC0-PC7 เป็นทางผ่านของข้อมูลระหว่างอุปกรณ์ภายนอกกับ 8255 สายสI/O PORTS ได้แก่ พอร์ท A (PA), พอร์ท B (PB) และพอร์ท C (PC) พอร์ทเหล่านี้ทุกพอร์ทสามารถที่จะเป็นได้ทั้งพอร์ทอินร์พุทและ พอร์ทเอาพุท และแต่ละบล็อกจะมีสายสัญญาณเชื่อมเข้ากับบัสข้อมูลภายในของ 8255

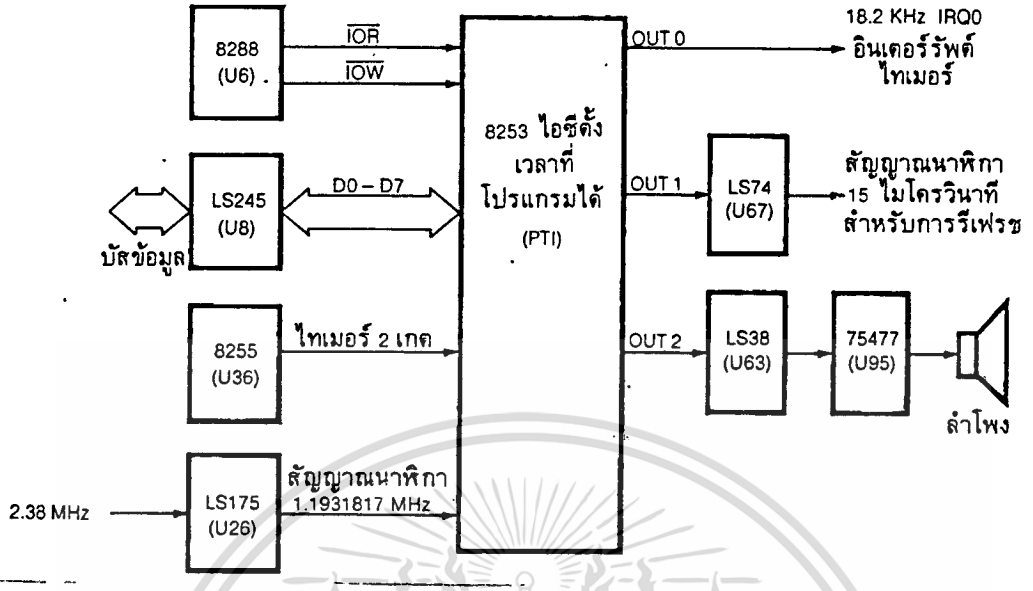
บล็อกกลุ่มถัดมาได้แก่ GROUP A CONTROL และ GROUP B CONTROL ซึ่งจะเป็นตัวหนดลักษณะการทำงานของทั้ง 3 I/O พอร์ท (8255 มีลักษณะการทำงานที่แตกต่าง กันอยู่ 3 โหมด สามารถกำหนดได้โดยการโปรแกรมส่ง CONTROL WORD ให้กับ 8255 จากรูปจะเห็นว่าพอร์ท c จะประกอบด้วยพอร์ทขนาน 4 บิท 2 พอร์ท โดยกลุ่มแรกจะถูกควบคุมโดย GROUP A CONTROL และอีกกลุ่มหนึ่งจะถูกควบคุมโดย GROUP B CONTROL

บล็อกกลุ่มสุดท้ายที่จะกล่าวถึงได้แก่ DATA BUS BRFFER และ READ/WRITE CONTROL LOGIC ซึ่งบล็อกเหล่านี้จะติดต่อกับ CPU ส่วน READ/WRITE CONTROL LOGIC จะเป็นส่วนที่ควบคุมให้ข้อมูลเข้าหรือออกจากรีจีสเตอร์ภายในตัวที่ถูกต้องและ ในเวลาที่เหมาะสม

<u>รายละเอียดการจัดเรียงของขาของ 8255</u>

ในส่วนนี้เราจะพิจารณาหน้าที่ของขาในแต่ละขาของ 8255 ซึ่งจะมีประโยชน์ต่อการเชื่อมต่อเข้ากับระบบบัสของ CPU สำหรับการจัดขาได้แสดงไว้ในรูปที่แล้วๆ รายละเอียดของขาแต่ละขามีดังต่อไปนี้คือ

D0-D7 : เป็นสายข้อมูลแบบ สองทิศทาง (BI-DIRECTIONAL BUS) เป็นทาง

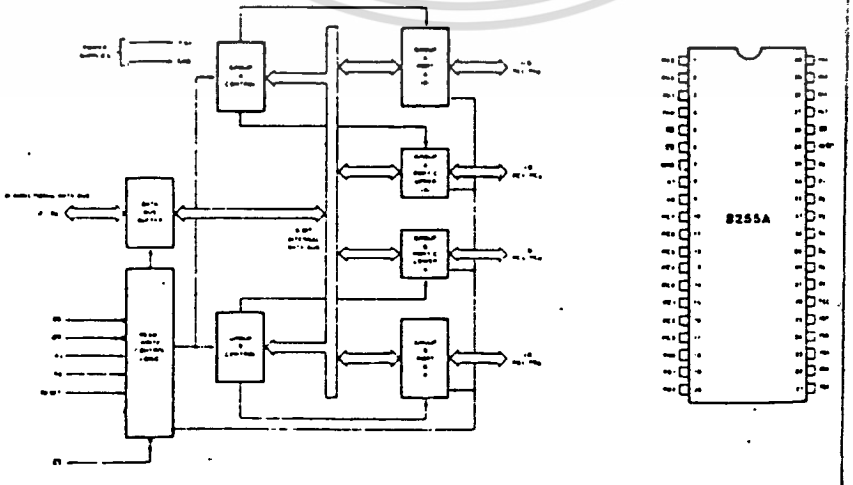


รูปแสดงการเชื่อมต่อ 8263 กับบัสข้อมูล

8255A/8255A-5
PROGRAMMABLE PERIPHERAL INTERFACE

- MCS-85™ Compatible 8255A-5
- 24 Programmable I/O Pins
- Completely TTL Compatible
- Fully Compatible with Intel® Microprocessor Families
- Improved Timing Characteristics
- Direct Bit Set/Reset Capability Easing Control Application Interface
- 40-Pin Dual In-Line Package
- Reduces System Package Count
- Improved DC Driving Capability

The Intel® 8255A is a general purpose programmable I/O device designed for use with Intel® microprocessors. It has 24 I/O pins which may be individually programmed in 2 groups of 12 and used in 3 major modes of operation. In the first mode (MODE 0), each group of 12 I/O pins may be programmed in sets of 4 to be input or output. In MODE 1, the second mode, each group may be programmed to have 8 lines of input or output. Of the remaining 4 pins, 3 are used for handshaking and interrupt control signals. The third mode of operation (MODE 2) is a bidirectional bus mode which uses 8 lines for a bidirectional bus, and 5 lines, borrowing one from the other group, for handshaking.



รูปแสดง บล็อกไดอะแกรม ของ 8255

ผ่านของข้อมูลต่างๆระหว่างพอร์ทต่างๆของ 8255 กับบัสข้อมูล

RESET : เมื่อสถานะนี้เป็น 1 8255 จะอยู่ในสภาวะรีเซ็ท ทุกๆพอร์ทของ 8251 จะถูกเซ็ทให้อยู่ในโหมดอินพุท

PA0-PA7,PB0-PB7 : ขาเหล่านี้ถูกใช้เป็นพอร์ท I/O ขนาด 8 บิทใช้ต่อกับอุปกรณ์ภายนอกอื่นๆ

PC0_0C7 : ขาสัญญาณเหล่านี้จะถูกใช้เป็นพอร์ท I/O ขนาด 8 บิทเช่นเดียวกับ PA0-PA7 และ PB0-PB7 แต่กลุ่มของขาสัญญาณเหล่านี้แบ่งออกได้เป็นสองกลุ่ม โดยแต่ละกลุ่มมีขนาด 4 บิท กลุ่มแรกใช้ควบคุม PB0-PB7 และกลุ่มที่ 2 ใช้ควบคุม PA0-PA7

WR (WRITE INPUT) : เมื่อขานี้เป็น 0 และ CS มีสถานะเป็น 0 ข้อมูลจากระบบข้อมูลจะถูกเขียนเข้าไปได้

RD (READ INPUT) : เมื่อขานี้เป็น 0 และสัญญาณ CS มีค่า 0 ข้อมูลจาก 8255 จะเข้าสู่บัสข้อมูล CPU ก็จะสามารถอ่านข้อมูลออกไปได้

A0-A1 : จะเป็นตัวกำหนดการเลือกใช้รีจีสเตอร์ภายในของ 8255

CS : เมื่อขานี้มีค่าเป็น 0 CPU จะสามารถอ่านหรือเขียนข้อมูลกับ 8255 ได้

3.4 คีย์บอร์ด เป็นหน่วย input ที่สามารถป้อนข้อมูลคอยสั่งงานให้ไมโครคอมพิวเตอร์ ดังนั้นอุปกรณ์คีย์บอร์ดจึงนับว่ามีความสำคัญมาก

คีบอร์ดต่างๆรวมแล้วมีทั้งหมด 83 คีย์ (ในปัจจุบันนี้มักใช้ 101 คีย์) ลักษณะการวางตัวจะวางตัวเหมือนเครื่องพิมพ์ แต่ได้เพิ่มฟังก์ชั่นคีย์เข้ามาต่างหากอีก รวมทั้งฟังก์ชั่นทางคณิตศาสตร์อีก

3.6 การ์ดควบคุมการสื่อสาร บนไมโครคอมพิวเตอร์กลุ่มที่ใช้ไมโครโปรเซสเซอร์ 8088 และ 8086 มีระบบการติดต่อสื่อสารข้อมูลแบบอนุกรม โดยกำหนดเป็นพอร์ต com1 กับ com2 โดยพอร์ตทั้งสองสามารถโปรแกรมตามความต้องการของผู้ใช้งาน โดยส่วนอินเตอร์เฟสกับภายนอกเป็นไปตามมาตรฐานของ RS-232

การติดต่อสื่อสารที่นิยมกันมากที่สุด ในการรับส่งแบบอนุกรมคือ การติดต่อแบบอะซิงโครนัสโดยมีบิตเริ่มต้น บิตสิ้นสุด บิตทั้งหมดจะประกอบกันเป็นเฟรมในขณะส่งหรือรับข้อมูล

สำหรับทาง hardware ที่จะทำให้เกิดการรับส่งข้อมูลแบบอนุกรมนี้ มักใช้ไอซีประเภท asynchronous reciever and transmitter แต่สำหรับบนเครื่องไอบีเอ็มนี้ได้ใช้

ชิพ 8250 เป็นตัวรับส่งข้อมูล ชิพ 8250 สามารถโปรแกรมให้ใช้งานตามต้องการได้ ซึ่งโดยปกติสามารถเลือกอัตราการส่งข้อมูลได้ตั้งแต่ 50 ถึง 9600 บอร์ดโดยมี หนึ่งครึ่ง หรือ สองบิต เฟรมของข้อมูลจะส่งแบบ 5 บิต 6 บิต 7 บิต หรือ 8 บิต ก็ได้

ขีดความสามารถของบอร์ดอะซิงโครนัสนี้ได้แก่

- การรับไม่ขึ้นกับสัญญาณนาฬิกาที่ป้อนเข้ามาทางอินพุต และไม่ต้องซิงโครไนซ์
- มีบัฟเฟอร์การรับส่งข้อมูล
- สามารถดีเทคการผิดพลาดในการรับส่งข้อมูลได้
- มีการควบคุมส่วนที่เป็นโมเด็ม

บนบอร์ดนี้ถ้าเป็น COM1จะมีหมายเลขพอร์ตเป็น F8H-3FEH แต่ถ้าโปรแกรมด้วยดิปสวิตช์ให้เป็น COM2 จะใช้หมายเลขพอร์ตเป็น 2F8H-2FEH

จากบนบอร์ดนี้จะมีสายอินเตอร์รัพต์ IRQ4 ซึ่งเป็นอินเตอร์รัพต์สำหรับ COM1 และ IRQ3 สำหรับ COM2 เราสามารถเลือกสายอินเตอร์รัพต์ตามต้องการได้

บทที่ 4

รูปแบบโครงข่ายไมโครคอมพิวเตอร์

ลักษณะและลักษณะการจัดโครงข่ายโทโปโรยี่ (topology) หมายถึงรูปร่างของโครงข่าย ที่พิจารณาจากการลากเส้นมาต่อร่วมกัน การเลือกรูปแบบโทโปโลยี่ที่เหมาะสมเป็นสิ่งสำคัญประการหนึ่งในการออกแบบโครงข่าย

เนื่องจากในการสื่อสารข้อมูลระหว่างจุดต่างๆ นั้นเราสายในการสื่อสารเป็นหลัก ดังนั้นจึงสามารถแบ่งการเชื่อมต่อสายถึงกันได้เป็น 2 แบบคือ

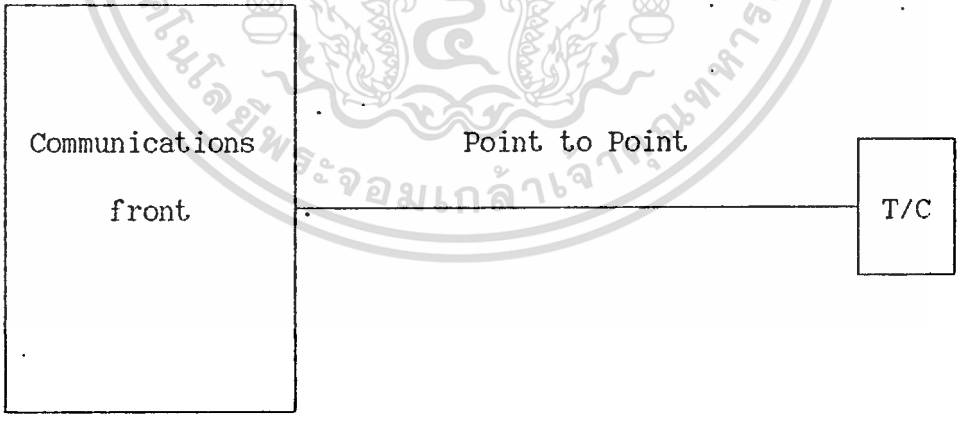
1) การเชื่อมต่อแบบจุดต่อจุด (point to point line)

2) การเชื่อมต่อสายแบบหลายจุด (multipoint or multidrop line)

ดังมีรายละเอียดดังต่อไปนี้

1) การเชื่อมต่อแบบจุดต่อจุด (point to point line)

ลักษณะการเชื่อมต่อแบบนี้เป็นลักษณะเบื้องต้นของการสื่อสารข้อมูล โดยใช้สายในการเชื่อมต่อระหว่างจุด 2 จุดแสดงได้ดังรูปข้างล่าง

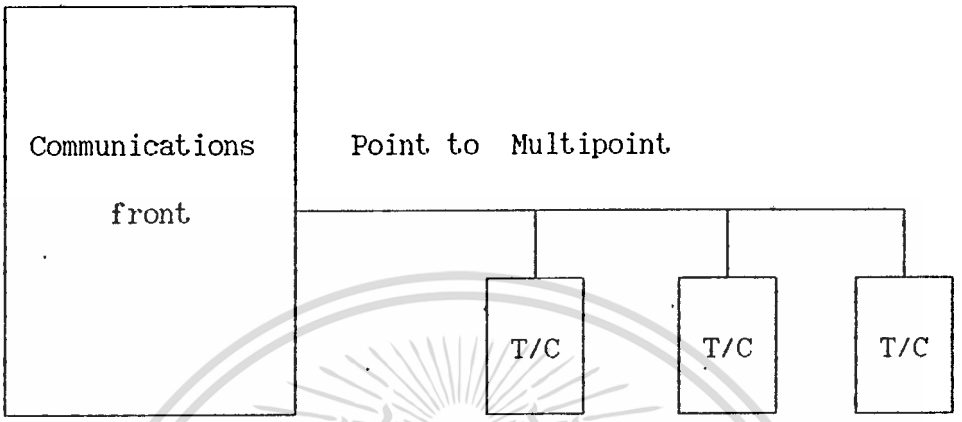


รูปที่ 4.1 แสดงการใช้สายแบบ point to point

2) การเชื่อมต่อสายแบบหลายจุด (multipoint or multidrop line)

ลักษณะการเชื่อมต่อแบบนี้ ประหยัดกว่าในแบบจุดต่อจุดมาก โดยเฉพาะอย่างยิ่งใน

กรณีที่มีจำนวนจุดในการสื่อสารมากขึ้นดังแสดงดังรูป



ลักษณะโดยทั่วไป ที่ใช้กันในโครงข่าย มีดังต่อไปนี้คือ

1) โครงข่ายแบบดาว (star network)

มีรูปแบบการติดต่อในลักษณะ ที่นำเอาเครือข่ายนำไปต่อเข้ากับ ศูนย์กลางการติดต่อ (host computer) หรือสวิทซ์ชิ่งเซนเตอร์ (switching center)

2) โครงข่ายแบบวงแหวน (ring network)

โทโปโรยี่แบบนี้ประกอบด้วยอุปกรณ์ทวนสัญญาณ 1 ตัว อยู่กับแต่ละสถานี มีลิงค์ต่อถึงกันเป็นวงรอบ อุปกรณ์สัญญาณเป็นอุปกรณ์ที่สามารถรับข้อมูลจากลิงค์ทางด้านหนึ่ง แล้วส่งด้วยความเร็วเท่ากับที่รับเข้ามา โดยที่ไม่มีการเก็บไว้ในบัฟเฟอร์ ของอุปกรณ์ทวนสัญญาณ ข้อมูลจะเคลื่อนที่ไปในทางทิศทางเดียว

3) โครงข่ายรูปต้นไม้ (tree network)

ลักษณะโทโปโรยี่แบบนี้ จะแยกสายส่งออกไปเป็นกิ่งก้านโดยไม่เป็นวงรอบ การรับ-ส่งจะผ่านเข้ายังตัวกลาง ไปยังตัวอื่นๆ

ข้อกำหนดที่ใช้ภายในโครงข่ายข้อกำหนดที่ใช้ภายในโครงข่าย มีความหมายถึงการกำหนดถึงข้อตกลงในเรื่องต่างๆสำหรับควบคุมการไหลของข้อมูล เพื่อสามารถที่จะสามารถติดต่อกันได้อย่างมีประสิทธิภาพและอย่างถูกต้อง

ISO (International Standards Organization) ได้แบ่งโปรโตคอล

ออกเป็น 7 ระดับดังต่อไปนี้คือ

1)Physical layal เป็นการบ่งบอกในระดับ hardware ยกตัวอย่าง เช่น RS-232

2)Data link layer เป็นการแปลงระดับการรับส่งข้อมูลที่ไม่แน่นอนให้แน่นอนยิ่งขึ้นโดยจัดรูปแบบเป็น(block),(frame)พร้อมทั้งมีการตรวจสอบข้อผิดพลาดเช่นตรวจสอบผลบวก (checksum), CRC เพื่อให้ทราบว่าข้อมูลที่ได้รับมานั้นถูกต้องหรือไม่

3)Network layer ทำการส่งข้อมูลแบบ (packet) เข้าไปในโครงข่าย แพกเกจอาจเดินทางโดยอิสระโดยกำหนดตำแหน่งที่อยู่ของผู้รับและผู้ส่ง วิธีนี้เรียกว่าdatagram หรือรับส่งข้อมูลไปตามเส้นทางที่กำหนดเป็นลำดับที่ละแพกเกจเรียกว่าvirtual circuit

4)Transport layer ทำหน้าที่ในการรับผิดชอบในการประกันคุณภาพของการบริการภายในโครงข่าย รวมทั้งมีหน้าที่ในการรับส่งข้อมูลระหว่าง 2 ด้าน

5)Session layer ทำหน้าที่ในการปฏิบัติการปฏิกริยาที่มีความสำคัญเสมอกันระหว่างขบวนการสื่อสารที่ร่วมกัน 2 ขบวนการ

6)Presentation layer ทำการแปลงรูปหรือ โคดที่จำเป็นสำหรับการส่งข่าวสารที่ร่วมกัน

7)Appication layer ทำหน้าที่โดยตรงกับโปรแกรมประยุกต์ใช้งานของผู้ใช้

บทที่ 5

แนวความคิดในการออกแบบ

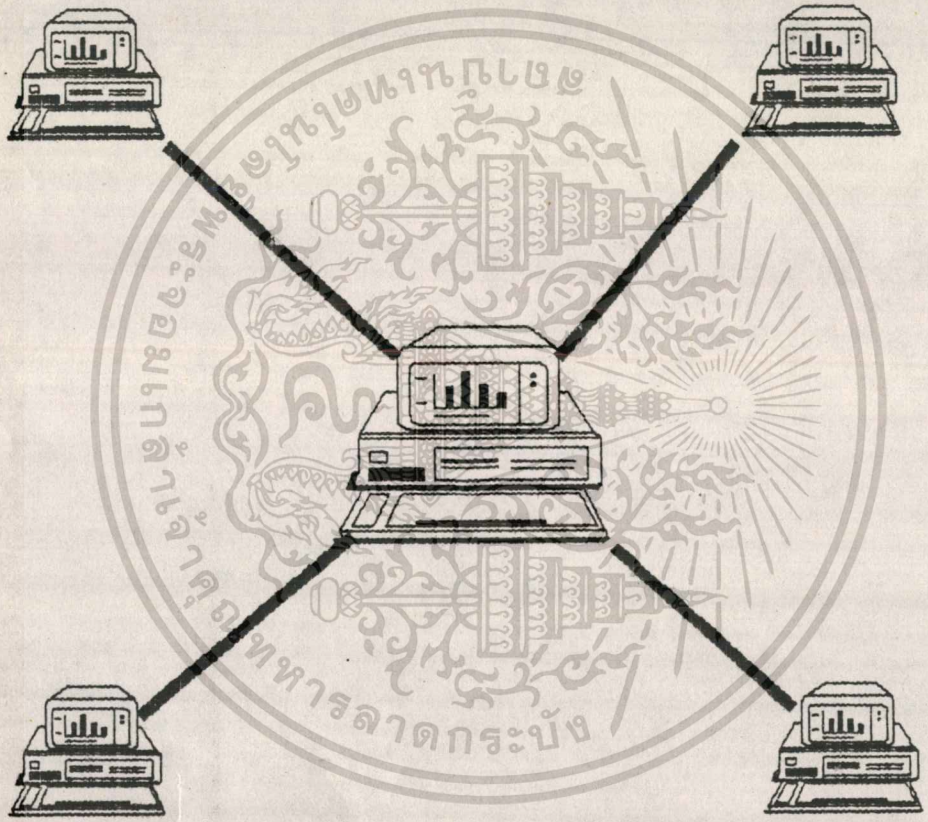
จากทฤษฎีและการรายละเอียดต่างๆ ดังที่กล่าวไว้แล้วตามบทข้างต้น เราจะนำทฤษฎีทางการสื่อสาร และรายละเอียดทางด้าน Hardware และ Software ของไมโครคอมพิวเตอร์ มาใช้ในการออกแบบเพื่อสร้างเป็นระบบการสื่อสารข้อมูลในเครือข่ายไมโครคอมพิวเตอร์ (Data Communication In Microcomputer Networks)

เราจะเริ่มต้นในการออกแบบระบบการสื่อสารข้อมูลในเครือข่ายไมโครคอมพิวเตอร์ด้วยการออกแบบและขอบเขตคุณสมบัติต่างๆโดยย่อดังนี้

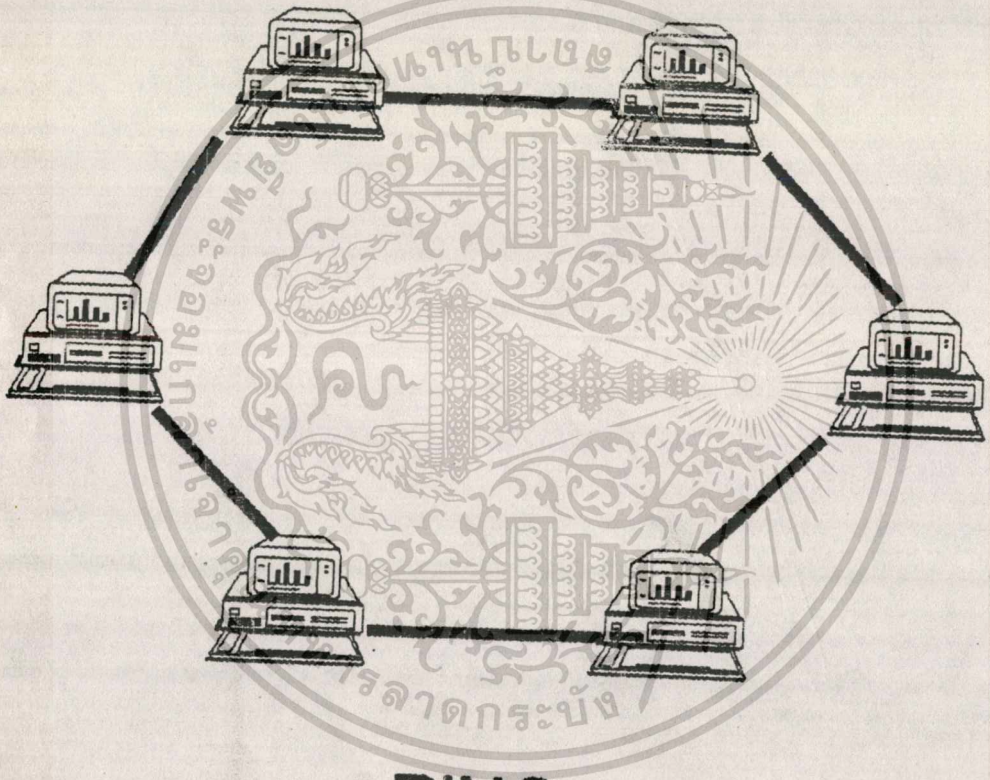
1. ผู้ใช้ไมโครคอมพิวเตอร์แต่ละจุดสามารถติดต่อกันได้อย่างอิสระ
2. สามารถขยายระบบเครือข่ายได้ง่าย

จากขอบเขตและคุณสมบัติข้างต้น เนื่องจากเราต้องการให้มีความสามารถในการขยายระบบได้ง่ายเราจึงต้องเลือกรูปแบบของการจัดโครงข่ายเป็นแบบบัส ที่มีข้อดีในด้านความสามารถในการขยายระบบดีกว่าแบบอื่นๆ

หลังจากเราได้รูปแบบตามโครงข่ายที่เราต้องการแล้ว เราจะกำหนดคุณสมบัติโดยละเอียดโดยจะแยกเป็นคุณสมบัติทางด้าน Hardware และคุณสมบัติทางด้าน Software ดังต่อไปนี้



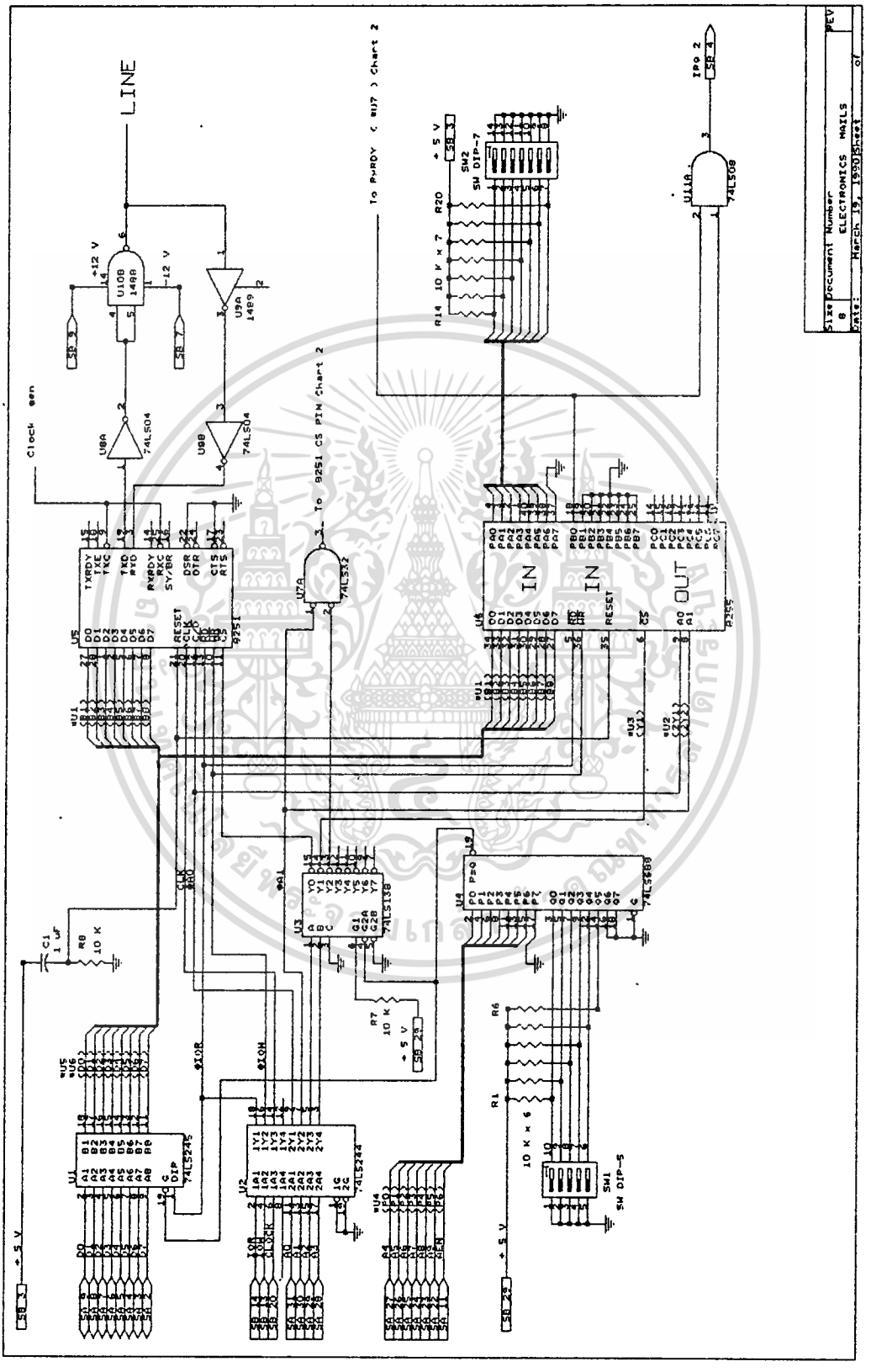
STAR

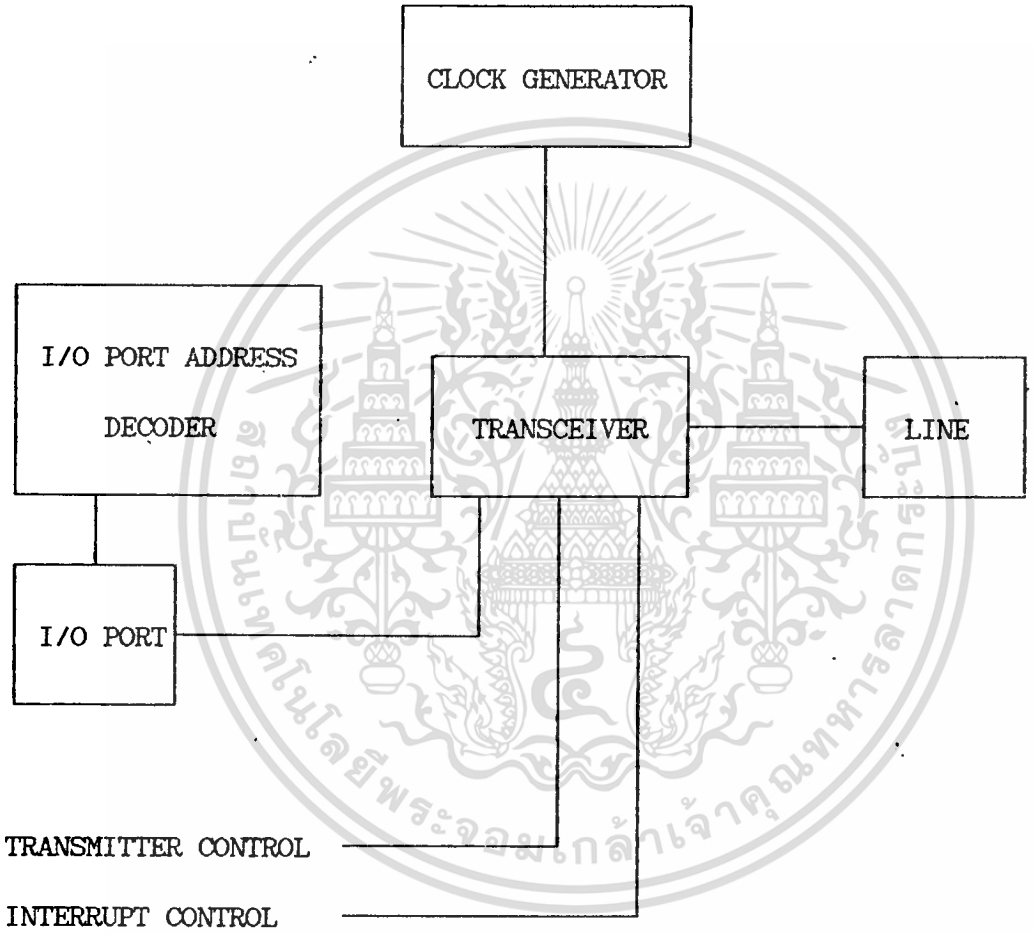


RING

ตารางแสดงคุณสมบัติที่ต้องการของระบบ

Hardware	Software
1. สามารถติดต่อกับไมโครคอมพิวเตอร์ของผู้ใช้ ได้โดยจะทำการเปลี่ยนข้อมูลจากสายส่งที่เป็นแบบอนุกรมมาเป็นข้อมูลแบบขนานเพื่อให้ไมโครคอมพิวเตอร์ของผู้ใช้ประมวลผลได้	1. สามารถทำงานร่วมกับHardware ได้ในการรับส่งข้อมูล
2. สามารถทำการ Interupt เพื่อบอกไมโครคอมพิวเตอร์ของผู้ใช้ได้ กรณีมีข้อมูลส่งมาจากที่อื่น	2. สามารถให้ผู้ควบคุมระบบกำหนดได้ว่าต้องการให้ผู้ใช้แต่ละคนติดต่อไปยังจุดใดบ้าง
3. สามารถทำการส่งข้อมูลพร้อมทั้งกำหนดที่หมายปลายทางได้ด้วย	3. สามารถกำหนดรหัสผ่านได้เพื่อป้องกันการเข้าระบบโดยไม่ได้รับอนุญาติ





BLOCK DIAGRAM

OF

DATA COMMUNICATION IN BUS TYPE

MICROCOMPUTER NETWORK ADAPTER CARD

รายละเอียดการทำงานของส่วนประกอบต่างๆบน (Data Communication In Bus Type Adapter Card) จาก block diagrame แสดงโครงสร้างของ Adapter Card สามารถแบ่งออกได้เป็น 6 ส่วนได้แก่

1. ส่วนถอดรหัสแอดเดรสอินพุทเอาท์พุทพอร์ท (I/O PORT ADDRESS DECODER)
2. ส่วนสร้างสัญญาณนาฬิกา (CLOCK GENERATOR)
3. ส่วนพอร์ทอินพุทและเอาร์พุทพอร์ท (I/O PORT)
4. ส่วนรับส่งข้อมูล (TRANSCEIVER)
5. ส่วนสัญญาณควบคุม (CONTROL SIGNAL)
6. ส่วนอื่นๆ

ส่วนถอดรหัสแอดเดรสอินพุทเอาท์พุทพอร์ท (I/O PORT ADDRESS DECODER)

เพราะการ Fixed Decoder อาจเกิดปัญหาการซ้ำซ้อน (OVERLAP) ของแอดเดรสที่ถอดรหัสไว้กับ ADAPTER CARD ที่เพิ่มเข้าไปในอนาคต จากวงจรเราเพียงแต่เปลี่ยนตำแหน่ง dip switch (sw1) เราก็จะสามารถเปลี่ยนช่วงรหัสการถอดแอดเดรสได้ สำหรับการทำงานของวงจรนั้น เราจะนำสัญญาณจาก (dip switch 5) ไปเปรียบเทียบสัญญาณแอดเดรสที่เข้ามาโดยใช้ไอซี 74LS638 โดยถ้าเท่ากัน เอาร์พุทจะเท่ากับ "0" ไปเอนาเบิ้ลไอซี 74LS138 (3-LINE TO 8-LINE DECODER)

ในการถอดรหัสแอดเดรสพอร์ท ต้องใช้สัญญาณ AEN ด้วยเพราะ AEN จะเป็นตัวชี้ว่า ขณะนี้สัญญาณต่างๆบนบัสเกิดจาก DMA หรือ CPU โดยถ้า AEN เป็น "1" จะหมายถึงช่วง DMA เพราะฉะนั้นช่วงนี้เราต้อง disable การถอดรหัสต่างๆไว้ มิฉะนั้นข้อมูลที่จะอ่านเขียนอาจเกิดการผิดพลาดได้

ข้อสำคัญในการถอดรหัสแอดเดรสพอร์ทใน

วงจรนี้ต้องมี delay time น้อยกว่า 92 nanoseconds มิฉะนั้นจะเกิดการเขียนข้อมูลลงบน address port ที่ผิด

IOW signal ต้องมี delay time น้อยกว่า 120 nanoseconds มิฉะนั้น, address port อาจจะถูกเขียนข้อมูลที่ไม่ถูกต้อง

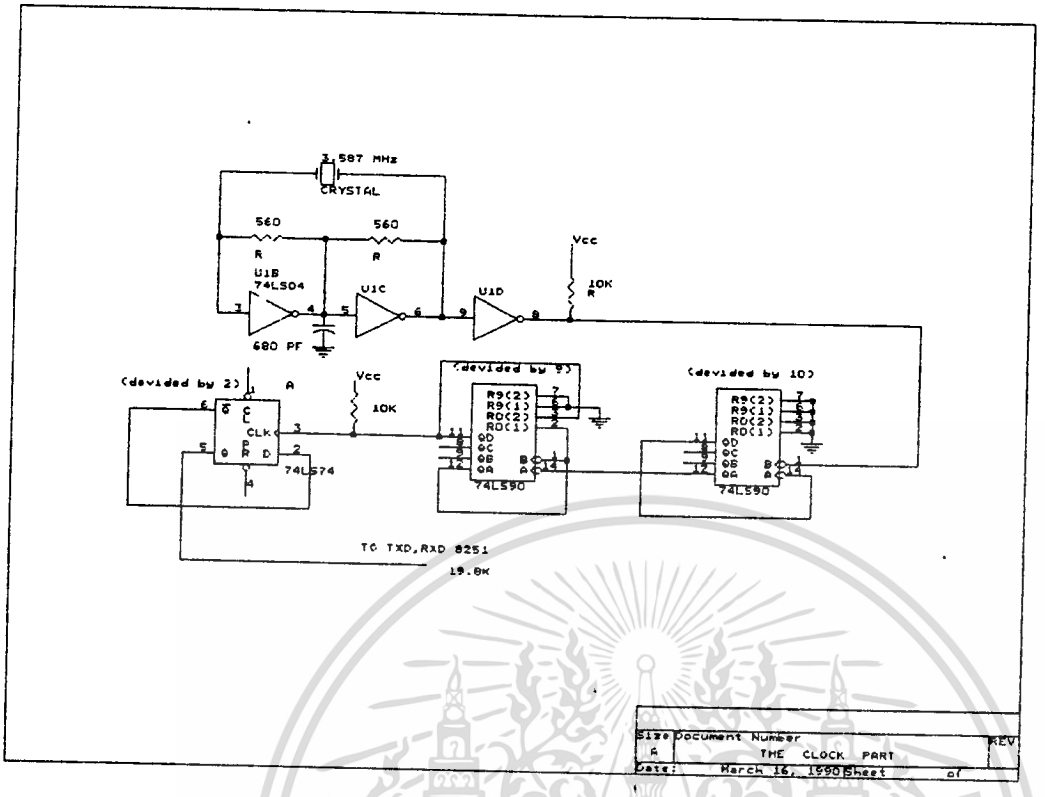
สัญญาณ A9 เป็นสัญญาณที่บ่งบอกว่า อยู่บน base board(A9="0") หรืออยู่บน system bus card slots (A9="1")

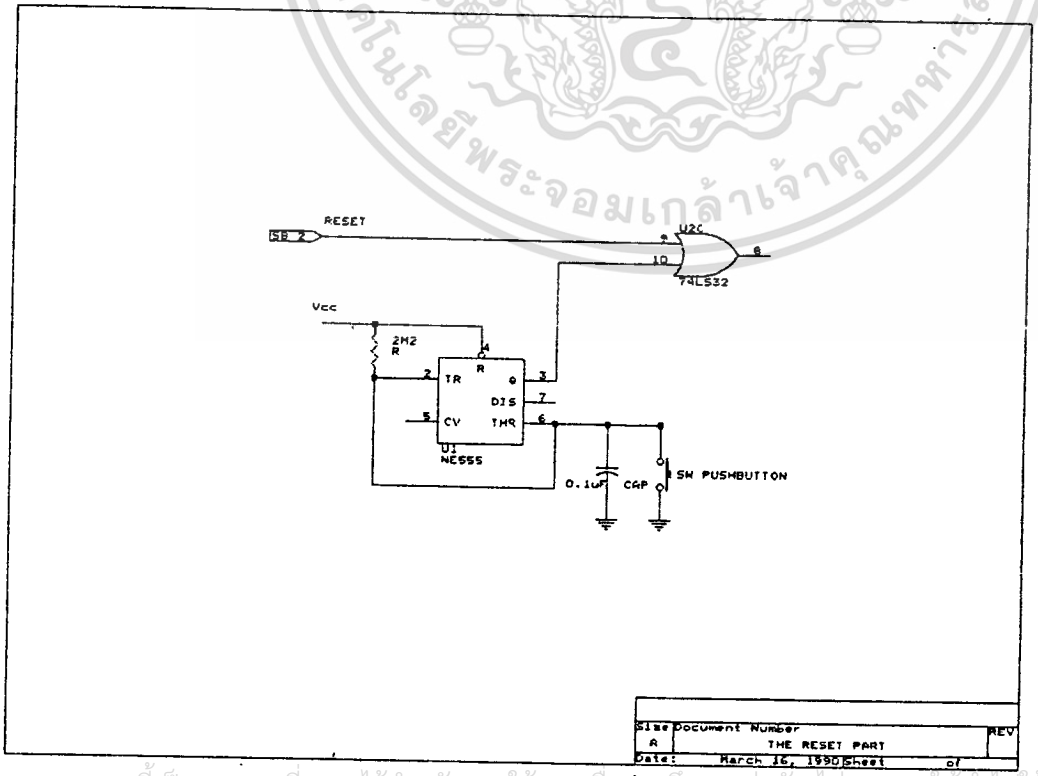
การเลือก address port นั้นควรเลือกให้ address port ที่ซ้ำซ้อนกับ address port ที่สงวนไว้แต่เฉพาะงาน (see the diagram card slot i/o port address usage in the IBM pc/xt reference)

สำหรับ data communication in bus type microcomputer networks adapter card ใช้ address port ที่อยู่ในช่วง 300h-309h หรือเขียนอยู่ให้อยู่ในรูป binary คือ 001100000000b-001100001001b พบว่า A9-A4 จะมีค่าไม่เปลี่ยนแปลง ดังนั้นจึงต่อ AEN เข้ากับขา P6-P0 ตามลำดับของไอซี 74LS688 ทำการตั้งค่า dip switch (sw-5) เท่ากับ A9-A4 คือ 110000b ตามลำดับและเนื่องจาก จะต้องทำการ disable ขณะ AEN ="1" และ A9="1" ดังนั้นที่ขา Q6 จึงต่อลงกราวด์และที่ขา Q5 จึงต่ออยู่กับ Vcc นำขา output ของ IC 74LS688 ไป enable IC 74LS138 เพื่อไปทำการ decoder address บิตต่ำคือ A3-A2 จากนั้นสามารถไปอินาเบิ้ล I/O port ต่อไปและใช้ IC 74LS32 ทำการดีโคดบิตที่หนึ่งต่อ

decoder address A1 ic 74LS138 (3-LINE TO 8-LINE DECODER)

Each 3-bit address drives one output low all others stay high. This chapter has three enable inputs. When g2 is high, all output are high, all output are high . When g1 is low, all output are high. To enable chip, make g1 is high and g2 low. (note g2+g2a+g2b)





ส่วนสร้างสัญญาณนาฬิกา (clock generation)

วงจรสร้างสัญญาณนาฬิกาเป็น **วงจรอะสเตเบิล มัลติไวเบรเตอร์** (ASTABLE MULTIVIBRATOR) ที่ใช้อินเวอร์เตอร์แบบ TTL จำนวน 2 ตัว ตัวต้านทาน (resistor 560 ohm) ทำหน้าที่ เป็นตัวไบแอสให้กับอินเวอร์เตอร์ความถี่ของสัญญาณนาฬิกาถูกกำหนดจากคริสตอล (CRYSTAL=3.579545 Mhz)

วงจรหารความถี่ (IC 7490, IC 74LS74)

ใช้วงจรหาร 3 ชุด ได้แก่ วงจรหาร10, วงจรหาร9, วงจรหาร2 ตามลำดับ ความถี่สัญญาณนาฬิกาสุดท้ายจะเท่ากับ 1/180 เท่าของความถี่ 3.57945 Mhz หรือประมาณเท่ากับ 19.8 Khz สัญญาณนาฬิกาที่ได้นี้ทำการป้อนเข้าสู่ขา TXC, RXC (TRANSMIT BAUD RATE CLOCK, RECEIVE BAUD RATE CLOCK) ของ ไอซี8251A

ส่วนพอร์ตอินพุทเอาท์พุท (INPUT/OUTPUT PORT)

(U6 IC 8255)

เราโปรแกรมการทำงานของไอซี 8255 ในโหมด 0 ซึ่งจะทำหน้าที่เป็นพอร์ทอินพุทและเอาท์พุทพอร์ทอย่างเดียว

สำหรับ พอร์ทแอดเดรสของไอซี 8255 ที่ถอดรหัส (DECODE) ไว้คือ **พอร์ท 304H - 307H หน้าที่ของแต่ละพอร์ทมีดังต่อไปนี้**

พอร์ท A (PORT 304H) ทำหน้าที่เป็น อินพุทพอร์ท ซีพียูสามารถรู้ว่าการ์ดอะแดปเตอร์(ADAPTER CARD) ที่ติดตั้งเข้ากับระบบเป็นการ์ดเลขที่เท่าไร (NUMBER CARD) โดยการอ่านค่าเข้ามาทางพอร์ท A (ระบบสื่อสารข้อมูลที่สร้างขึ้นนี้สามารถทำการขยายระบบได้ถึง 256 การ์ด) เราสามารถทำการกำหนดเลขที่ของการ์ด โดยการเซ็ทค่าที่ดิฟ

สวิทช์ (DIP SWITCH)

พอร์ท B (PORT 305H) ทำหน้าที่เป็นอินพุทพอร์ท ซีพียูจะทำการอ่านค่าเข้ามาทางพอร์ทนี้ เพื่อตรวจสอบว่ามี HARDWARE INTERRUPT IRQ2 เกิดขึ้นหรือไม่

พอร์ท C (PORT 306H) ทำหน้าที่เป็นเอาท์พุทพอร์ท เป็นพอร์ทที่ ซีพียูใช้ในการควบคุมการปิดเปิดเส้นทาง (LINE) ฮาร์ดแวร์อินเทอร์รัพท์

สำหรับ PORT 307H เป็นพอร์ทที่รับรหัสควบคุม (CONTROL WORD) ในที่นี้คือ 92H

ส่วนส่งและรับข้อมูล (TRANSMITTER AND RECEIVER)

(IC 8251A)

ในระบบนี้เราใช้ไอซี 8251A 2 ตัว

ไอซี8251A ตัวแรก (PORT 300H - 301H) จะทำการรับส่งข้อมูลที่เป็นคำสั่งควบคุมการทำงานของระบบซึ่งมีความหนาแน่นของข้อมูลในสายส่งต่ำ และเราใช้สัญญาณขาRXRDY ของไอซี8251A ตัวแรกนี้เป็นสัญญาณที่ทำให้เกิดฮาร์ดแวร์อินเทอร์รัพท์ที่ IRQ2

ไอซี8251A ตัวที่สอง (PORT 308H - 309H) ใช้ในการส่งข้อมูลข่าวสารจะมีความหนาแน่นของข้อมูลในสายส่งสูง

รายละเอียดขาของไอซี 8251A ที่ใช้ในโครงงานนี้

กลุ่มที่ใช้ติดต่อกับซีพียู

D0-D7 ใช้ในการติดต่อกับ DATA BUS ของซีพียูโดยตรง ซึ่งจะทำหน้าที่ในการรับส่งข้อมูลและคำสั่งต่างๆระหว่าง ไอซี8251A กับ ซีพียู

RESET IC 8251A จะถูกรีเซ็ทเมื่อขานี้ได้รับลอจิก "1"

CLK ใช้ในการควบคุมช่วงเวลาการทำงานภายในของ IC 8251A สำหรับการใช้งานนั้นจะต่อเข้าโดยตรงกับระบบ **สัญญาณที่ขาCLK นี้ไม่เกี่ยวข้องกับอัตราการรับส่งข้อมูลแต่อย่างใด**

$\overline{RD}$ เมื่อขานี้ได้รับลอจิก "0" IC 8251A จะทำการส่งข้อมูลแบบขนานออกมาที่DATABUS เพื่อส่งให้กับซีพียู

$\overline{WR}$ เมื่อขานี้ได้รับลอจิก "0" IC 8251A จะทำการรับข้อมูลแบบขนาน จาก DATA BUSของระบบ

C\$\overline{D}$ (CONTROL\DATA) เมื่อขานี้ได้รับลอจิก "1" แสดงว่า ซีพียู ต้องการติดต่อกับ CONTROL REGISTER

เมื่อขานี้ได้รับลอจิก "0" แสดงว่า ซีพียูต้องการติดต่อกับ DATA REGISTER

$\overline{CS}$ ในกรณีที่ขานี้ได้รับลอจิก"0" ก็จะ ENABLE IC 8251A โดยสัญญาณนี้จะได้มาจากการถอดรหัสพอร์ทแอดเดรส

CTS (CLEAR TO SEND) เป็นขาอินพุทที่ใช้ในการทำให้ IC 8251A เริ่มทำการส่งข้อมูลได้ สิ่งที่ตองระวังในการใช้งานขานี้คือ เมื่อไม่ได้ใช้งานขานี้จะต้องถูกต่อเข้ากับลอจิก"0" ถ้าไม่เช่นนั้นIC 8251A จะทำการส่งข้อมูลไม่ได้

TXD (TRANSMIT DATA OUTPUT) เป็นขาที่ใช้ในการส่งข้อมูลไปตามสายส่ง

TXC (TRANSMIT BAUD RATE CLOCK) ขานี้เป็นขาที่ใช้ในการส่งสัญญาณ CLOCK ที่ใช้ในการส่งข้อมูล ซึ่งก็คือความถี่ที่ใช้นการกำหนด BAUD RATE นั่นเอง โดยปกติแล้วจะ ต้องมีความถี่น้อยกว่าความถี่สัญญาณนาฬิกาของระบบ (CLK) ไม่น้อยกว่า 30 เท่า

RXD (RECEIVER DATA INPUT) ใช้ในการรับข้อมูลแบบอนุกรมจากสายส่ง

RXC (RECEIVER BAUD RATE CLOCK) เป็นขาที่ใช้ในการรับสัญญาณ CLOCK ที่ใช้ในการรับข้อมูล โดยปกติแล้วจะทำการต่อเข้ากับขา TXC โดยตรง

RXDY (RECEIVER DATA READY) ใช้แสดงว่า IC 8251A พร้อมที่จะส่งข้อมูลให้กับซีพียู และขานี้เราใช้ในการขอฮาร์ดแวร์อินเตอร์รัพท์

การเชื่อมต่อกับสายส่งข้อมูล

การรับส่งข้อมูลแบบอนุกรมนั้น นิยมไปใช้ในการรับส่งข้อมูลในระยะทางไกล ระดับสัญญาณที่ใช้ในวงจรคือ +5 ,0 โวลต์ นั้นไม่สามารถจะส่งไปได้ไกลนัก ดังนั้นก่อนที่จะส่งข้อมูลไปในสายส่งเราทำการเปลี่ยนระดับแรงดันของสัญญาณใหม่ ใน PROJECT นี้จะใช้ระดับแรงดันในสายส่งประมาณ +/- 12 โวลต์ ซึ่งเราจะมีส่วนที่จะทำการดปลี่ยนระดับแรงดันของสัญญาณจากระดับสัญญาณที่ใช้กับอุปกรณ์พวก TTL ไปเป็น ระดับสัญญาณ +\- 12 โวลต์

จากวงจร การส่งข้อมูล เราใช้ IC MC1488 ทำหน้าที่เปลี่ยนระดับสัญญาณ TTL เป็น +\- 12โวลต์

การรับข้อมูล ใช้IC MC1489 ทำหน้าที่เปลี่ยนระดับสัญญาณ +\- 12 โวลต์ เป็นระดับสัญญาณ TTL

คุณสมบัติทางซอฟแวร์

จากส่วนประกอบของฮาร์ดแวร์ตามที่กล่าวข้างต้น จะทำให้เราได้ซอฟแวร์โดยเราจะแบ่งซอฟแวร์ตามลักษณะการทำงานในส่วนนี้ได้ 3 ส่วนคือ

1. ส่วนรับ
2. ส่วนส่ง
3. ส่วนที่ทำหน้าที่ทั่วไป

โดยมีรายละเอียดดังนี้

1. ส่วนรับ

จะได้รับการกระตุ้นผ่านฮาร์ดแวร์อินเทอร์รัพท์ IRQ2 โดยเมื่อเริ่มทำงานต้องทำการตัดสัญญาณอินเทอร์รัพท์ที่ได้ก่อนโดยผ่าน 8255 Port C ซึ่งได้ทำการติดตั้งเป็นเอาร์พุทพอร์ท (output port)

2. ส่วนส่ง

เนื่องจากวงจรได้ออกแบบให้มีทั้งการส่งทั้งแบบ สายควบคุม และ สายข้อมูล โปรแกรมในส่วนนี้จะเป็นลักษณะโปรแกรมย่อย ที่สามารถเรียกใช้ได้ผ่านซอฟแวร์อินเทอร์รัพท์ซึ่งใช้เป็นอินเตอร์รัพท์หมายเลข 66H ในส่วนนี้จะเป็นลักษณะของโปรแกรมที่ฝังตัวอยู่ในหน่วยความจำดังจะมีรายละเอียด เช่น

2.1 AH = 2H จะส่งค่า AL ไปทางสายควบคุม 1 อักขระ

2.2 AH = 82H จะส่งค่าใน AL ไปทางสายข้อมูล 1 อักขระ

2.3 AH = 85H

CX = จำนวนไบท์ที่จะส่ง

DX = เซกเมนต์ที่ข้อมูลอยู่

BX = ออฟเซตที่ข้อมูลอยู่

จะส่งข้อมูลจำนวน CX ไบต์ ทางสายควบคุมเป็นต้น

โดยรายละเอียดการใช้งานโปรแกรมในส่วนนี้สามารถอ่านรายละเอียดได้จากตัวโปรแกรม 51IN66.ASM โดยตรง

3. ส่วนที่ทำหน้าที่ทั่วไป

ในส่วนนี้ได้แก่ส่วนที่ติดตั้งข้อมูลของระบบทั้งหมดตั้งแต่การติดตั้งชื่อและรหัสผ่านของผู้ควบคุมระบบ (SUPIRVISOR) , การติดตั้งชื่อและรหัสผ่านของผู้ใช้ทั่วไป (user) ,การติดตั้งตู้ไปรษณีย์ของผู้ใช้แต่ละคน และ อื่นๆ ทั้งนี้ข้อมูลในส่วนนี้จะผ่านการเข้ารหัส (encode) เพื่อป้องกันและเพื่อความปลอดภัยของข้อมูลไว้ด้วย นอกจากนี้ยังได้มีการตั้งชื่อกลุ่มและเผื่อข้อมูลว่างไว้ส่วนหนึ่งเพื่อการขยายและพัฒนาระบบต่อไปในอนาคตไว้ด้วย และเนื่องจากโปรแกรมในส่วนนี้เขียนด้วยภาษา (pascal) จึงค่อนข้างยาวและประกอบด้วยโปรแกรมอรรถประโยชน์ (utility) ตลอดจนโปรแกรมย่อยต่างๆหลายๆโปรแกรมจึงจะขอนำมาลงไว้ในที่นี้เพียงบางส่วนเท่านั้น

โครงสร้างข้อมูลของระบบ

โครงสร้างของข้อมูลที่ใช้ในระบบจะใช้โดยมีการกำหนดรูปแบบของข้อมูลในแบบของภาษาปาสคาล (pascal) ในชื่อของข้อมูลแบบ Udata-type โดยมีรูปแบบดังนี้

```
Udatatype = Record
    Table_Encode : byte
       User_Name : string[30];
   User_Password : string[15];
    Gmember_Number : byte;
          Gnumber : Array[1..30] of byte;
end;
```

โดยจะมีรายละเอียดของฟิลด์ต่างๆดังนี้

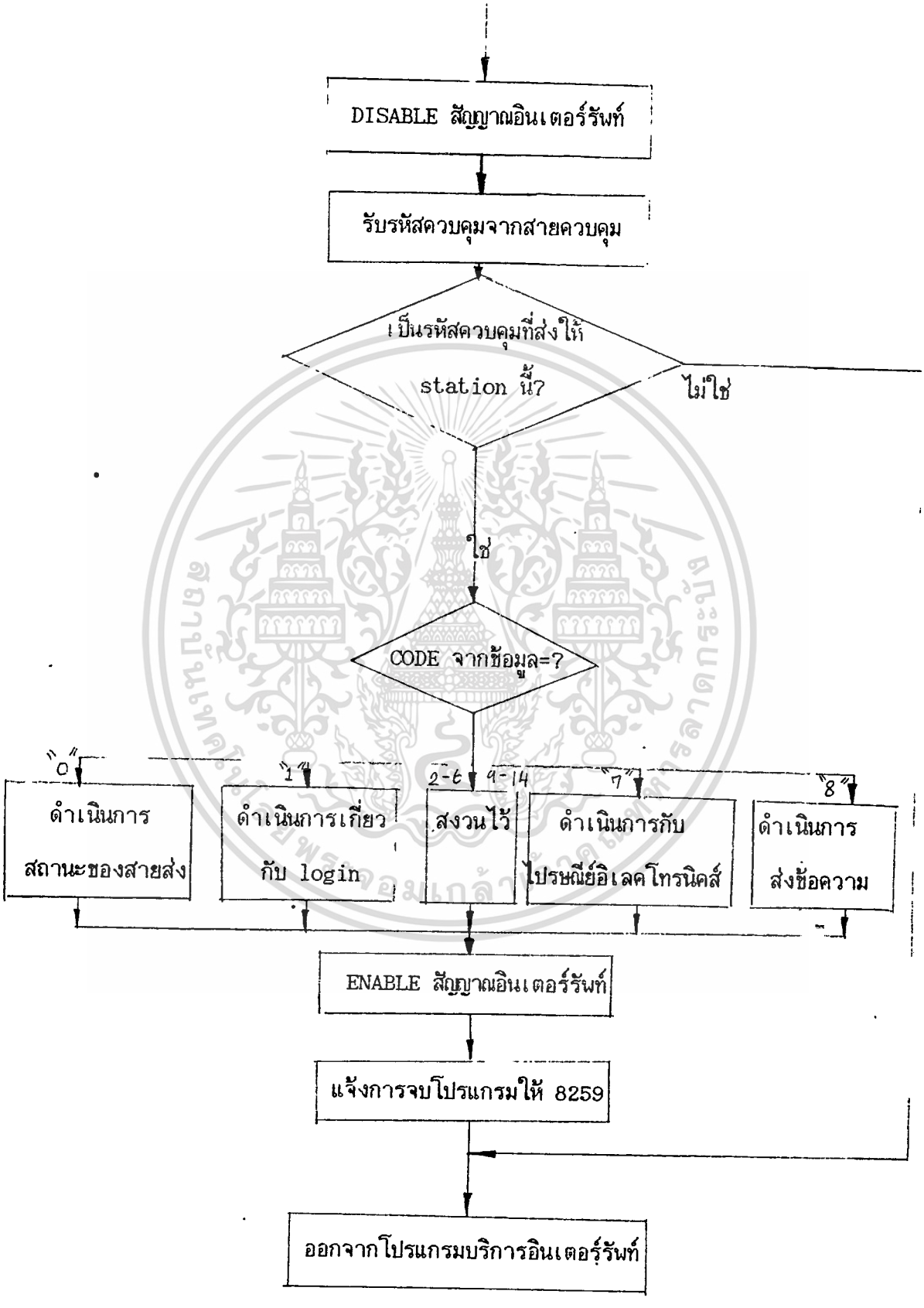
1.User_Number แสดงถึงลำดับที่ของผู้ใช้แต่ละคน

2.User_name แสดงถึงชื่อของผู้ใช้โดยกำหนดเป็นสตริงขนาด 30 ไบต์ซึ่งขนาดความยาวของชื่อนั้นสามารถเปลี่ยนแปลงได้ในค่าคงที่ชื่อ user_name_num_ch

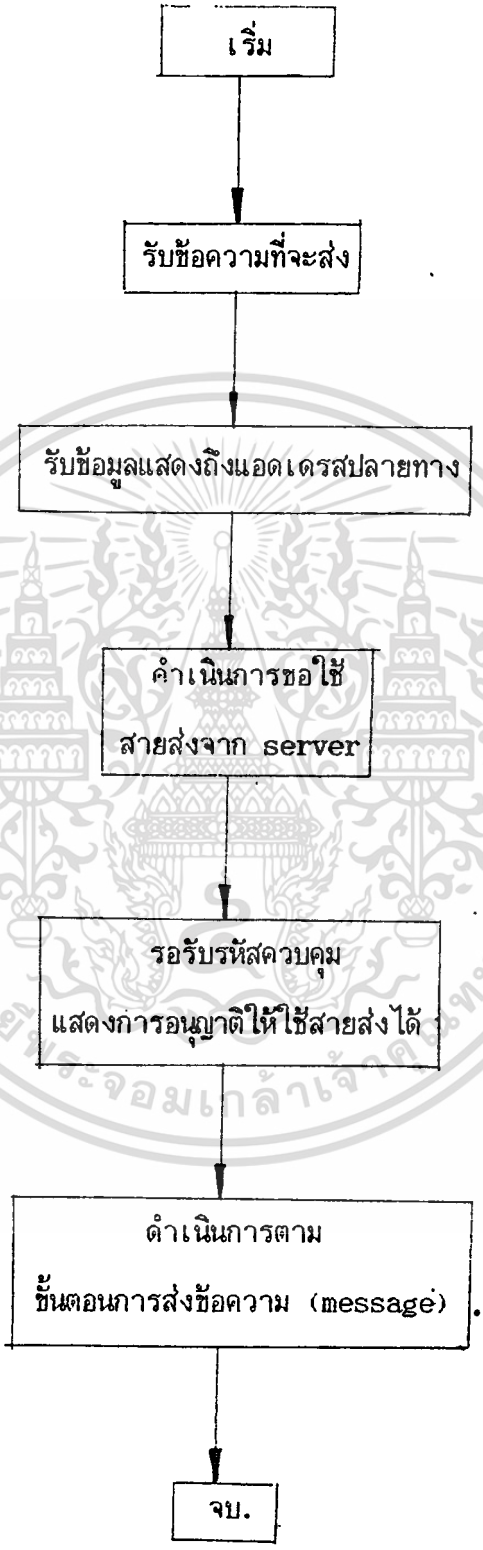
3.User_Password แสดงถึงรหัสผ่านของผู้ใช้นั้นๆกำหนดขนาดไว้จำนวน 15 ไบต์ซึ่งสามารถเปลี่ยนแปลงขนาดจากค่าคงที่ชื่อ user_pass_num_ch

4. Gmember_number และ Gnumber เป็นฟิลด์ที่กำหนดไว้เพื่อขยายความสามารถของระบบในอนาคตในการกำหนดชื่อกลุ่ม และ กำหนดความสามารถในการเข้าถึงข้อมูล (group level) ของแต่ละกลุ่มด้วย

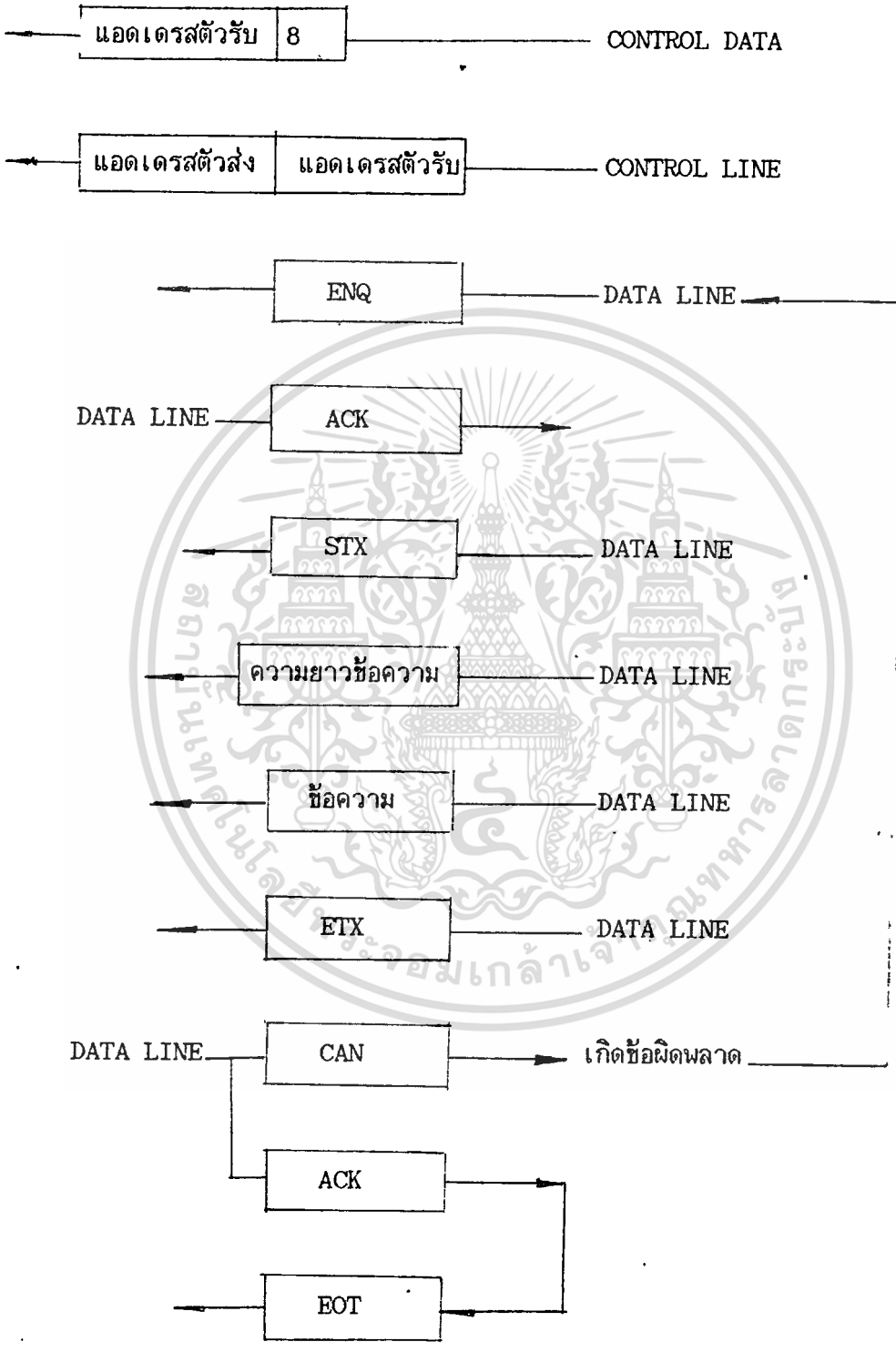
ผังการทำงานของ Interupt Service Routine (OAH)



ผังการทำงานของโปรแกรมในการส่งข้อความ



ขั้นตอนในการส่งข้อความ (Message)



หมายเหตุ

ENQ : ENQUIRY REQUEST

STATUS FROM REMOTE STATION

ACK : ACKNOWLEDGE

STX : START OF TEXT

ETX : END OF TEXT

CAN : CANCEL

EOT : END OF TRANSMISSION

บทที่ 6

บทสรุปและข้อเสนอแนะ

บทสรุป

ระบบสื่อสารข้อมูลโครงข่ายแบบบัสที่สร้างขึ้นนี้ประกอบด้วย 2 ส่วนใหญ่ๆคือ

1. ส่วนฮาร์ดแวร์(Hardware)

ในส่วนฮาร์ดแวร์เราสร้างเป็นการ์ดอินเตอร์เฟสที่เชื่อมกับ Slot ของเครื่องไมโครคอมพิวเตอร์ IBM PC/XT เพื่อใช้ในการสื่อสารข้อมูลโครงข่ายแบบบัส (Data Communication Network In Bus Type Interface Card)

เราถอดรหัสแอดเดรสอินพุทเอาร์พุทพอร์ท (Decode I/O Ports Address) ในช่วง 300H-309H

ระบบนี้มีการรับการตอบสนอง ฮาร์ดแวร์อินเตอรัพท์ผ่านทาง IRQ2 หรือ Int 0AH การ์ดสื่อสารที่สร้างขึ้นนี้มีความอ่อนตัวสูง คือสามารถนำมาประยุกต์ใช้งานทางด้านการสื่อสารข้อมูลในรูปแบบต่างๆ โดยเพียงแต่เขียน ซอฟแวร์ สนับสนุนการทำงานนั้นเพิ่มเข้าไปโดยไม่ต้องทำการแก้ไขในส่วน ฮาร์ดแวร์ หรือดัดแปลงก็เพียงแต่เล็กน้อย

สำหรับสายส่งรับข้อมูล เราใช้ 3 เส้น ในการเชื่อมต่อโครงข่าย

1. กราวด์
2. สายส่งรับข้อมูลที่เป็นรหัสควบคุม (Control Code)
3. สายส่งรับข้อมูลข่าวสาร

หมายเหตุ Interface Card แต่ละอันจะมี Number Card ของแต่ละการ์ด ซึ่งเราสามารถทำการเปลี่ยน Number Card โดยการตั้งค่าที่ Dip Switch การ์ดที่มี Number Card 00H จะกำหนดเป็น File server

2. ส่วนซอฟแวร์(Software)

ซอฟแวร์ที่เขียนขึ้นเพื่อควบคุมการทำงานของระบบสื่อสารข้อมูลให้เป็นไปตาม โปรโตคอลที่วางไว้ และได้พยายามทำให้เป็นระบบเปิด กล่าวคือ ให้เกิดความสะดวกต่อผู้ที่ต้องการพัฒนาโดยการเพิ่มเติม คุณลักษณะทางซอฟแวร์ เพื่อขยายขีดความสามารถการทำงานของระบบให้สมบูรณ์ยิ่งขึ้นในอนาคต เป็นต้นว่า เราเขียนโปรแกรมสำหรับการทำงานทางด้านการส่งรับข้อมูล , อ่านค่าสถานะต่างๆ เป็นโปรแกรม Interrupt Routine ที่เขียนขึ้นเอง คือ Int66H ซึ่งเป็นโปรแกรมที่ฝังตัวอยู่ในหน่วยความจำ Resident Programe .

For example ถ้าเราต้องการส่ง character ไปที่ data line จะเขียนโปรแกรมได้ดังนี้

```
mov al,character
mov ah,82h
int 66h
```

และเราเขียนโปรแกรมควบคุมการสื่อสาร เป็นแบบการขัดจังหวะการทำงานของ cpu (Interrupt) เมื่อมีข้อมูลเข้ามาที่ช่องการสื่อสาร วิธีเช่นนี้จะทำให้โปรแกรมหลักไม่ต้องพะวงถึงข้อมูลที่จะเข้ามาทางช่องทางการสื่อสาร เนื่องจากมี Interrupt Routine OAH คอยจัดการให้อยู่แล้ว (ทาง IRQ2)

ระบบการสื่อสารข้อมูลที่สร้างนี้ใช้การสื่อสารในโหมด Asynchronous

มีการใช้แรงดันในสายส่งรับข้อมูล +/- 12V

จำนวนบิทหยุด (Stop Bit) 2 บิท

จำนวนบิทข้อมูล(Data Bit) 8 บิท

การตรวจเช็คข้อผิดพลาดแบบพาริตี้คี่ (Odd parity)

อัตราเร็วในการสื่อสารข้อมูลมีหลายอัตรา ได้แก่ 19200/16 bauds, 19200/64 buads, 19200 buads ซึ่งสามารถเลือกอัตราเร็วในการสื่อสารข้อมูลได้จาก ซอฟแวร์

ในระบบสื่อสารข้อมูลที่สร้างขึ้นนี้จะมีเครื่องคอมพิวเตอร์ตัวหนึ่งจะทำหน้าที่เป็นตัวหลัก (File Server) สำหรับระบบที่ออกแบบไว้ตอนนี้สามารถมีการเชื่อมต่อของเครื่องไมโครคอมพิวเตอร์สูงสุด 16 เครื่อง (File Server 1 เครื่อง Station 15 เครื่อง)แต่ในกรณีที่ต้องการจำนวนเครื่องไมโครคอมพิวเตอร์ที่จะนำมาเชื่อมต่อในโครงข่ายมากๆ เราสามารถทำการแก้ไขในส่วน ซอฟแวร์ได้ โดยสามารถมีจำนวนเครื่องไมโครคอมพิวเตอร์ที่เชื่อมต่อในโครงข่ายมากที่สุด 256 เครื่อง(ในกรณีที่มีจำนวนเครื่องไมโครคอมพิวเตอร์ในโครงข่ายมากควรทำการออกแบบวงจรในส่วนเชื่อมต่อกับสายข้อมูล (Line Driver)ใหม่)

ความสามารถของระบบที่สร้างขึ้นนี้

สามารถจัดสรรค์พื้นที่ข้อมูลใน Disk ของแต่ละ User

มีการใช้ Password

สามารถรับส่งข้อมูลที่เป็นข้อความ, ไฟล์ข้อมูลในอัตราเร็วสูงสุด 19.2 Kbuads

สามารถทำงานในรูปแบบของไปรษณีย์อิเลคโทรนิคส์ (Electronics Mail)

ข้อเสนอแนะ

โครงงานที่ทำขึ้นนี้มีความซับซ้อนและมีลายละเอียดต่างๆอยู่มาก ดังนั้นถ้ามีผู้สนใจที่จะศึกษาพัฒนาระบบสื่อสารข้อมูลโครงข่ายแบบบัสนี้ให้ดียิ่งขึ้นในอนาคต ควรจะทำการพัฒนาขีดความสามารถของระบบทางด้านต่อไปนี้

1. ในการเชื่อมโยงการเป็นโครงข่ายสำหรับเครื่องไมโครคอมพิวเตอร์เพื่อใช้อุปกรณ์ที่มีราคาแพงร่วมกันเช่น Hard Disk และเครื่องพิมพ์ ดังนั้น ควรทำการจัดสรรพื้นที่ในฮาร์ดดีสค์ ที่ File Server ควบคุมอยู่ให้กับ User แต่ละราย และทำให้ไมโครคอมพิวเตอร์เครื่องอื่นมองเห็นเป็นแฟ้มข้อมูลของระบบงานน

2. File Sharing คือไมโครคอมพิวเตอร์หลายๆเครื่องสามารถติดต่อแฟ้มข้อมูลเดียวกันพร้อมๆกัน โดยที่มีระบบ Log Record ป้องกันการ Update ซ้อน

3. Print Spoling คือไมโครคอมพิวเตอร์ในโครงข่ายสามารถส่งงานมาพิมพ์ที่เครื่องพิมพ์กลางได้

4. Time Multiplexing คือเป็นการแบ่งเวลาการใช้สายส่งรับข้อมูลเพื่อความรวดเร็วในการทำงานของระบบ ตัวอย่างเช่นในกรณีที่ User A ต้องการอ่านไฟล์จากข้อมูลใน Hard disk ของ File Server เป็นเวลา 20 นาที ในขณะที่ User B ต้องการอ่านไฟล์จาก Hard disk เป็นเวลา 5 นาที แต่User Aได้ใช้สายรับส่งข้อมูลก่อน ดังนั้นถ้าไม่มีการแบ่งเวลาการใช้สายรับส่งข้อมูล User B ต้องรอ User A เลิกใช้สายรับส่งข้อมูล ดังนั้น User B จะใช้เวลาในการรับส่งข้อมูล เท่ากับ 25 นาที แทนที่จะเป็น 5 นาที ซึ่งเราสามารถพัฒนาจุดด้อยนี้โดยการใช้ Time Multiplexing กล่าวคือให้ระบบสามารถทำหน้าที่ผลัดกันทำการส่งรับข้อมูลให้กับ User ที่ขอใช้สายรับส่งข้อมูลในเวลาใกล้เคียงกันคือ ให้ User A รับส่งข้อมูล 100 Bytes และให้ User B 100 Bytes สลับกันไปเรื่อยๆ จนข้อมูลที่จะทำการรับส่งให้แก่ User B หมดลงจึงทำการรับส่งข้อมูล ให้ User A อย่างเดียวเป็นต้น ดังนั้นจึงดูเหมือนว่า User B และ User A สามารถใช้สายส่งข้อมูลได้ในเวลาพร้อมๆกัน

ภาคผนวก

ภาคผนวกนี้แสดงโปรแกรมควบคุมการทำงานการสื่อสารข้อมูลโครงข่ายไมโครคอมพิวเตอร์แบบบัส

```
UNIT  PCONSTU;

INTERFACE

USES    BOXUNIT;

Const   Mybox : Boxdesc =
              ( BorderAttr : $0F;
                  BoxType :  $5;     (* 0,1..8 $80, $81..$88 *)
                        W :  18;     (* Width of window  >=2 *)
                        H :  12;     (* Highth of window >=2 *)
                FillerChar : ' ';
                FillerAttr : $70;
                        X :   1;
                        Y :   2);

        MainMenuNum  = 5;            (* Number of Menu        >=1 *)
        MenuChoice = 5;              (* Choice for each menu >=1 *)
        Maxlevel   = 1;              (* Max Dimension for each menu *)
        FollowLength = 20;
        Aup = #24;
        Adown = #25;
        Aright = #26;
        Aleft = #27;
        TitleLine = 3;
        HeaderLine = 2;
        HeaderAttr = $70;
        AllScreen = 4000;
        Boxtype_of_writeinbox = $85;
        MaxBytemove = 4000;
        StartWLine  = 4;
        SystemDrive : String[5] = 'B:';
       PV_Lan_Dirst : String[30] ='\PVLAN';
            MailST : String[30] = '\MAIL';
```

```
        GroupDST : String[12] = '\GROUP1.DAT';
        USERDST  : String[12] = '\USER1.DAT';
        BodyMailST : String[12] = '\BODYMAIL.ENV';
        NEWMAILST  : String[12] = '\NEWMAIL.DAT';

        G_Name_Num_Ch = 30;
        Number_of_Dir = 30;
        User_Name_Num_Ch = 30;
        User_Pass_Num_Ch = 15;
        User_Have_G_Num  = 30;

        Normal306 = $40;
         Clear306 = $00;
       Disable306 = $C0;
       Cl_Disable306 = $80;


Type    S=Array[1..25,1..80] of record
                                  Ch1,Att:byte;
                                end;
      Scrkeep = ^ScrPntrType;
      ScrPntrType = Record
                      ScrSack : Array[1..4000] of byte;{ Array[1..3520] of by
                      Next    : Scrkeep;
                    end;
       SP         = ^S;
    (*New  Type  declaration *)
             Index_P   = ^Selected_Index_P;
      Selected_Index_P = Record
       DBackWard,DForward : Index_P;
```

```
        Menu_Number  : integer;    { Tell Number of choice that is sele
end;


    Dir_Data  =  Record
      Dir_Name : String[8];
   Dir_Surname : String[3];
     Dir_Reach : Array[1..8] of Boolean;
    End;


    GDatatype  =  Record
     Table_Encode : byte;
     Group_Number : integer;
     Group_Name   : String[G_Name_Num_Ch];
     All_Dir_IN   : Array[1..Number_of_Dir] of Dir_Data;
    End;


    UDataType  = Record
      Table_Encode : Byte;
       User_Number : Integer;
         User_Name : String[User_Name_Num_Ch];
     User_Password : String[User_Pass_Num_Ch];
    GMember_Number : byte;
           GNumber : Array[1..User_Have_G_Num] of byte;
    End;


    MainDataType  =  Record
              Can_Send : Boolean;
```

```
                Now_User : UDataType;
                Now_Group : GDataType;

        (* add  for  any  variable *)
            End;

            LetterNews = Record
                Name : String[12];
                Open : Boolean;
            End;

            System_Join_Data = Record
                Login_Now  :  Boolean;
                Login_Data : UDataType;
                Can_Send : Boolean;
                    End;

Var         MainDataPointer  : ^MainDataType  Absolute  $60:0;
                MainData  :  MainDataType;

IMPLEMENTATION
BEGIN
  MainDataPointer:=Ptr(Seg(Maindata),Ofs(Maindata));
  MainData.Can_Send:=True;
END.
```

```
Unit   Contacin;
Interface
Uses   Dos;

  Function      HiNibble(Num : byte) : byte;
  Function      LowNibble(Num : byte) : byte;
  Function      PackinByte(HiNib,LowNib : byte): byte;
  Function      ThisStation : byte;
  Procedure     SendChControl(Chb : byte);
  Procedure     SendStrData(Var  Str : string);
  Procedure     ReceiveControl(Var  Chb,Status : byte);

implementation

Function      HiNibble(Num : byte) : byte;
Begin
  Hinibble:=Num Shr 4;
End;

Function      LowNibble(Num : byte) : byte;
Begin
  Lownibble := Num and $0F;
End;

Function      PackinByte(HiNib,LowNib : byte): byte;
Begin
 PackinByte:=(Hinib Shl 4) or (LowNib And $0F);
End;

Function      ThisStation : byte;
```

```
Begin
 ThisStation:=Port[$304];
End;

Procedure    SendChControl(Chb : byte);
Var    Reg1 : registers;
Begin
  Reg1.AL:=Chb;
  Reg1.Ah:=$02;
  Intr($66,Reg1);
End;

Procedure    SendStrData(Var  Str : string);
Var    Reg2 : registers;
Begin
  Reg2.Ah:=$85;
  Reg2.Cx:=Length(Str);
  Reg2.Dx:=Seg(Str);
  Reg2.Bx:=Ofs(Str);
  Intr($66,Reg2);
End;

Procedure    ReceiveControl(Var  Chb,Status : byte);
Var    Reg3 : registers;
Begin
  Reg3.Ah:=03;
  Intr($66,Reg3);
  Chb:=Reg3.Al;
  Status:=Reg3.Ah;
End;
```

```
;
;  Procedure   Setlogic(Portnumber : integer; bit_number,logic_: byte);
;  23:36:57  9/14/1989

SetlogicFrame  Struc
Old_BP          Dw    ?
Retadr          DD    ?
logic_          Dw    ?
bit_number      Dw    ?
Portnumber      Dw    ?
SetlogicFrame  Ends

logicframe     Struc
Old_BP2        Dw   ?
Retaddr2       DD   ?
Bit_number2    Dw   ?
Portnumber2    Dw   ?
;FuncTret2     Dw   ?
logicframe     Ends


CODE        Segment  byte public
            Assume   Cs:CODE
            PUBLIC   Setlogic,logic

Setlogic    Proc  Far
            push  BP
            mov   BP,SP
            push  AX
            push  CX
```

```
            push  Dx
            pushf
            mov   Dx,[BP].Portnumber
            in    AL,DX                 ;Read value of Port
            mov   AH,01h
            mov   CL,byte ptr [BP].bit_number
            shl   AH,CL                 ;Adjust for operate  data
            mov   CL,[BP].byte ptr logic_
            or    CL,CL
            jnz   logic_1
            not   AH
            and   AL,AH                   ;set bit to 0
            jmp   Ready_for_Out
logic_1:    or    AL,AH                   ;set bit to 1
Ready_for_Out:  out   DX,AL
            popf
            pop   DX
            pop   CX
            pop   AX
            pop   BP
            ret 6
Setlogic    Endp

;
; Function logic(Portnumber:integer;bit_number:byte)
; 23:37:04  9/14/1989
;

logic       Proc  Far
            push  BP
```

```
        mov   BP,SP
        push  DX
        push  CX
        pushf
        mov   DX,[BP].Portnumber2
        in    AL,DX
        mov   CL,byte ptr [BP].bit_number2
        inc   CL
        shr   AL,CL
        jc    it_1
        mov   AX,0              ;Return  function value
        jmp   Exit1
it_1:   mov   AX,1
Exit1:  popf
        pop   CX
        pop   DX
        pop   BP
        ret   4
logic   Endp

CODE    Ends
        End
```

```
;-----------------------------------------------
; Program          :  8251  Routine
; Purpose          :
; Written  Date    :  21:12:04  11/26/1989
; Last   UpDate    :  17:13:19  3/11/1990.
; Written  By      :  Piyamit  Chatchartree
;
;---  INT  66H  ------------------------------------------------
;              BIT
;  (AH) = 7 6 5 4 3 2 1 0
;         X ----- Y -----
;
;   X  =  0   OPERATE  WITH  CONTROL  LINE
;   X  =  1   OPERATE  WITH  DATA  LINE
;
;   Y  =  1  INITIAL  8251
;         (DH)  =  COMMAND  WORD
;         (DL)  =  MODE  WORD
;
;   Y  =  2  SEND  CHARACTER  IN  (AL)  OVER  THE  LINE
;         (AL)  =  CHARACTER
;
;   Y  =  3  RECEIVE  CHARACTER  FROM  LINE  AND  RETURN TO (AX)
;         (AH)  =  CHARACTER
;         (AL)  =  STATUS
;
;   Y  =  4  READ  STATUS  TO  (AL)
;         (AL)  =  STATUS
;
;   Y  =  5  SEND  BLOCK  OF  CHARACTER  TO  LINE
```

```
;       (CX) = NUMBER OF BLOCK IN BYTE
;       (DX) = SEGMENT OF DATA
;       (BX) = OFFSET OF DATA
;-------------------------------------------------------------------
Page        30,132
Title                       ; <Title>

Codesg      Segment     Public 'Code'
            Org         0100h
            Assume  Cs:Codesg,Ds:Codesg

CTRLPORT8255 =  307H
CTRL8255W    =   92H

CTRL8251D    =  300H
CTRL8251C    =  301H

DAtA8251D    =  308H
DATA8251C    =  309H

Baud19k      =  0DDH
Baud1200     =  0DEH
Baud300      =  0DFH

MODE8251W    =   15H

COMMUINT     =   66H

CTRL_8251    =  300H
DATA_8251    =  308H
```

```
;***  Main Program  Go Below  ***

Start:        Jmp    InitResident_IO

Active_Port   Dw   (?)

Support_Commu    Proc   Near
                 Assume    Cs:Codesg,Ds:Codesg,Ss:Codesg
                 Sti
                 Push    Dx
                 Push    Cx
                 Push    Es
                 Push    Ds
                 Push    Cs
                 Pop     Ds
                 Push    Ax
                 Cmp     Ah,80H
                 JAE     more_than_80
                 Mov     Ax,Ctrl_8251
                 jmp     Continue1
more_than_80:
                 mov     Ax,Data_8251
Continue1:       Mov     Active_Port,Ax
                 Pop     Ax
                 And     Ah,07FH
                 Dec     Ah
                 Jnz     Not_EQ_1
                 Jmp     Init_8251
Not_Eq_1:        Dec     Ah
```

```
                Jnz     NOt_Eq_2
                jmp     Send1
Not_Eq_2:       Dec     Ah
                Jnz     Not_Eq_3
                Jmp     Receive
Not_Eq_3:       Dec     Ah
                Jnz     Not_Eq_4
                Jmp     Read_Status
Not_Eq_4:       Dec     Ah
                Jnz     Return1
                Jmp     Send_Many
Return1:        Pop     Ds
                Pop     Es
                Pop     Cx
                Pop     Dx
                iret

Init_8251:      Push    Dx
                Mov     Dx,Active_Port
                Inc     Dx
                Mov     Al,55H
                Out     Dx,Al           ;Internal  Reset
                Pop     Ax
                Out     Dx,Al
                Mov     Al,Ah
                Out     Dx,Al
                Jmp     Return1

Send1:          Push    Dx
                Mov     Dx,Active_Port
```

```
                Inc     Dx
                Mov     Ah,Al
Send1loop:      in      Al,Dx
                Test    Al,01H
                Jz      Send1loop
                mov     Al,Ah
                Dec     Dx
                Out     Dx,Al
                Pop     Dx
                Jmp     Return1

Receive:        Push    Dx
                Mov     Dx,Active_Port
                Inc     Dx
Wait_For_Receive:
                in      Al,Dx
                Test    Al,02H
                Jz      Wait_For_Receive
                Mov     Ah,Al
                Dec     Dx
                in      Al,Dx
                Pop     Dx
                Jmp     Return1

Read_Status:    Push    Dx
                mov     Dx,Active_Port
                inc     Dx
                in      Al,Dx
                Pop     Dx
                Jmp     Return1
```

```
Send_many:      Push    Ax
                Mov     Es,Dx  ; Dx  contain Segment of data to send
                Mov     Dx,Active_Port
Loop1_Many:     Inc     Dx
Wait_For_many:  in      Al,Dx
                Test    Al,01H
                Jz      Wait_For_many
                Dec     Dx
                Mov     Al,ES:[BX]
                Out     Dx,Al
                Inc     Bx
                Jnz     Not_Add
                Push    Ax
                Mov     Ax,Es
                Add     Ax,1000H
                Mov     Es,Ax
                Pop     Ax
Not_Add:        Loop    Loop1_Many
                Pop     Ax
                Jmp     Return1

Support_Commu   EndP
InitResident_IO:
                Mov     Dx,CTRLPORT8255
                Mov     AL,CTRL8255W
                Out     Dx,AL           ; Initialize  8255
                Mov     Dx,CTRL8251C
                Mov     AL,Baud19K
                Out     Dx,AL
```

```
        Mov     Al,MODE8251W
        Out     Dx,AL           ; Initialize  8251 CONTROL UNIT
        Mov     Dx,DATA8251C
        Mov     AL,Baud19K
        Out     Dx,AL
        Mov     Al,MODE8251W
        Out     Dx,AL           ; Initialize  8251  DATA  UNIT
        Mov     Dx,Offset   Support_Commu
        Mov     Al,66H
        Mov     Ah,25H
        Int     21H
        Mov     Dx,Offset  InitResident_IO
        Int     27H

;***  Others Sub rountine Go Below  ***

;***  Define Constant  ***


Codesg      Ends
            End     Start
```

```
PROGRAM  INITIAL_SYSTEM_AND_SUPERVISOR;

Uses     BoxUnit,Crt,Objmenu,Dos,ReadCHU,PCONSTU,Table,StringU,GLOBALPR;

Const    HeadMenu1 : HXY =

         ( HeadName : 'INITIAL';

                  X : 4;

                  Y : 4    );

          MENU1 : Menu = (

                Menu_number  : 1;    (*  Reset  By  Menu.INIT  *)

           When_Left   : nil; { Pointer  to  other menu  when  left pressed }

          When_Right   : nil;

            When_ESC   : nil;

          RealChoice   : 1;    (*  Reset  By  Menu.INIT  *)

          LastChoice   : 1;

          HeadNameXY   : nil;     { Head  Of  menu ; = nil when none }

            CoXofBox   : 10;

            CoYofBox   : 9;

            BoxWidth   : 1;

        BoxTypeLocal   : $87;

       Respectively    : True;

         ChoiceMaxL    : 0;    (*  Reset  By  Menu.INIT  *)

       HaveFollowSt    : False;

            FollStL    : 10;       { Max  Length  of  follow string }

                                   (*  Reset  By  Menu.INIT  *)

               Body    :

              (

              ( ChoiceStr : ' SUPERVISOR ';
```

```
    FollowSt : nil;
  Follst_Pos : Right;
        Skip : false;
  When_Enter : nil),

( ChoiceStr : ' OPTION     ';
    FollowSt : nil;
  Follst_Pos : Right;
        Skip : false;
  When_Enter : nil),

( ChoiceStr : ' EXECUTE    ';
    FollowSt : nil;
  Follst_Pos : Right;
        Skip : True;
  When_Enter : nil  ),

( ChoiceStr : ' QUIT       ';
    FollowSt : nil;
  Follst_Pos : Right;
        Skip : false;
  When_Enter : nil),

( ChoiceStr : '';
    FollowSt : nil;
  Follst_Pos : Right;
```

```
            Skip : false;
       When_Enter : nil    )     ));


     HeadMenu2 : HXY =
( HeadName : 'SUPERVISOR';
           X : 40 - Length('SUPERVISOR') div 2;
           Y : 4     );

   MENU2 : Menu = (
Menu_number   : 1;     (* Reset By Menu.INIT *)
  When_Left   : nil; { Pointer to other menu when left pressed }
 When_Right   : nil;
   When_ESC   : nil;
 RealChoice   : 2;     (* Reset By Menu.INIT *)
 LastChoice   : 1;
 HeadNameXY   : nil;     { Head Of menu ; = nil when none }
   CoXofBox   : 20;
   CoYofBox   : 13;
   BoxWidth   : 1;
BoxTypeLocal  : $86;
Respectively  : True;
  ChoiceMaxL  : 0;     (* Reset By Menu.INIT *)
HaveFollowSt  : True;
     FollStL  : 0;        { Max Length of follow string }
                          (* Reset By Menu.INIT *)
```

```
   Body  :
(
( ChoiceStr : ' NAME      ';
   FollowSt : nil;
 Follst_Pos : Right;
       Skip : false;
 When_Enter : nil),

( ChoiceStr : ' PASSWORD  ';
   FollowSt : nil;
 Follst_Pos : Right;
       Skip : false;
 When_Enter : nil),

( ChoiceStr : '';
   FollowSt : nil;
 Follst_Pos : Right;
       Skip : false;
 When_Enter : nil  ),

( ChoiceStr : '';
   FollowSt : nil;
 Follst_Pos : Right;
       Skip : false;
 When_Enter : nil),
```

```
    ( ChoiceStr : '';
        FollowSt : nil;
     Follst_Pos : Right;
           Skip : false;
     When_Enter : nil    )    ));

    HeadMenu3 : HXY =
( HeadName : 'OPTION';
         X : 40 - Length('OPTION') div 2;
         Y : 4     );

  MENU3 : Menu = (
Menu_number   : 2;     (* Reset By Menu.INIT *)
  When_Left   : nil; { Pointer to other menu when left pressed }
 When_Right   : nil;
   When_ESC   : nil;
 RealChoice   : 2;     (* Reset By Menu.INIT *)
 LastChoice   : 1;
 HeadNameXY   : nil;     { Head Of menu ; = nil when none }
   CoXofBox   : 14;
   CoYofBox   : 14;
   BoxWidth   : 1;     (* Reset By Menu.INIT *)
BoxTypeLocal  : $86;
Respectively  : True;
  ChoiceMaxL  : 0;     (* Reset By Menu.INIT *)
HaveFollowSt  : True;
```

```
FollStL  : 20;        { Max  Length  of  follow string }
  (*  Reset  By  Menu.INIT  if HaveFollost=false*)
   Body  :
(
( ChoiceStr : ' RESET GROUP DATA ';
   FollowSt : nil;
 Follst_Pos : Middle;
       Skip : false;
 When_Enter : nil),

( ChoiceStr : ' RESET USERS DATA ';
   FollowSt : nil;
 Follst_Pos : Middle;
       Skip : false;
 When_Enter : nil),

( ChoiceStr : '';
   FollowSt : nil;
 Follst_Pos : Right;
       Skip : false;
 When_Enter : nil  ),

( ChoiceStr : '';
   FollowSt : nil;
 Follst_Pos : Right;
       Skip : false;
```

```
            When_Enter : nil),


           ( ChoiceStr : '';
             FollowSt : nil;
            Follst_Pos : Right;
                 Skip : false;
            When_Enter : nil   )    ));


        Header = 'INITIAL   SYSTEM   UTILITY';
       Version = 'Version 1.2';
   MaxBytemove = 4000;
   StartWLine  = 4;
   YesSt = 'YES';
   NoSt = 'NO';

   Have_SV_Name : Boolean = false;
   Have_SV_Pass : Boolean = false;
   SKIP_Not_Sel_Attr : Array[False..True] of byte =($07,$0F);
   SKIP_Sel_Attr : Array[False..True] of byte =($70,$0F);

Var              p,j ,OCurX,OCurY  : byte;
                OcurStart,OcurStop : byte;
  (*                  P_Menu       : MenuP;
      S_Choice_P , Head_S_Choice_P : Index_P;    *)
         { Tell us which choice is selected }
                SV_NAME, SV_PASS   : STRING;
```

```
                    FollMENU2Name : Str40;
                    FollMENU2PASS : Str40;
                  FollMENU3GROUP : Str40;
                    FollMENU3USER : Str40;
                    PathStrAccess : String;
                            AccFi : File;
                         Dirinfo1 : SearchRec;
    Have_Group_Dat , Have_User_Dat : Boolean;

     Have_PV_LAN_Dir,Have_Mail_Dir : Boolean;

       IN_PV_LAN_DIR,PahtStrAccess : String;
                             Fi3   : file of Letternews;

{ ------------------------------------------------------------------ }
{ --  initMaxL   To   Set  Array of maxL  to length of max string -- }
{ ------------------------------------------------------------------ }

Procedure   InitPointer_Of_AllMenu;
begin
  FillChar(FollMENU2Name, SizeOf(FollMENU2Name),'*');
    FollMENU2PASS := FollMENU2Name;
 FollMENU2Name[0] := Char(User_Name_Num_Ch);  { set  length of string }
 FollMENU2PASS[0] := Char(User_Pass_Num_Ch);
    FollMENU2Name := FollMENU2Name + ' ';
```

```
New(S_Choice_P);
Head_S_Choice_P:=S_Choice_P;
S_Choice_P^.DBackWard:=Nil;
S_CHoice_P^.DForward:=Nil;
P_Menu:=@Menu1;

Menu1.Body[1].When_Enter := @MENU2;
Menu1.Body[2].When_Enter := @MENU3;
        Menu1.HeadNameXy := @HeadMenu1;

Menu2.Body[1].Followst := @FollMENU2Name;
Menu2.Body[2].Followst := @FollMENU2PASS;
      Menu2.HeadNameXy := @HeadMenu2;
        Menu2.When_Esc := @Menu1;

Menu3.Body[1].Followst := @FollMenu3GROUP;
Menu3.Body[2].Followst := @FollMenu3USER;
      Menu3.HeadNameXy := @HeadMenu3;
        Menu3.When_Esc := @Menu1;


(* INIT  CONSTANT *)
StartSave:=Ptr(Seg(Vram^),Ofs(Vram^)); {  +160*Titleline);  }
OCurX:=whereX;
OCurY:=WhereY;
OCurStop:=Mem[$40:$60];
```

```
OCurStart:=Mem[$40:$61];
ByteMove := 160 * 25;
WaitChar := '*';
Textattr:=$07;
Tempscr:=nil;
TempCreate:=nil;

Menu1.Init(1);
Menu2.Init(1);
Menu3.Init(2);

end;

Procedure   CheckStatusOfdata;
begin
 In_Pv_Lan_Dir:=SystemDrive+PV_Lan_DirST;
 FindFirst(IN_PV_LAN_DIR,Directory,DIrinfo1);
 If  DosError = 0 then
   begin
     Have_PV_Lan_Dir:=True;
     FindFirst(In_PV_LAn_Dir+Mailst,Directory,DirInfo1);
     if  DosError = 0  then  Have_Mail_Dir:=True
                       Else  Have_Mail_Dir:=False;
     FindFirst(IN_PV_LAN_DIR + GroupDST,Anyfile,DIrInfo1);
     if  DosError = 0 then Have_Group_Dat:=true
```

```
                    Else Have_group_Dat:=False;
    FindFirst(IN_PV_LAN_DIR + UserDST,AnyFile,Dirinfo1);
    If  DosError = 0 then  Have_User_Dat:=true
                    Else  Have_User_Dat:=False;
  End
Else
  Begin
    Have_PV_Lan_Dir:=False;
    Have_Group_Dat:=False;
    Have_Mail_Dir:=False;
    Have_User_Dat:=False;
  End;
If  Have_Group_Dat  then
begin
   FollMenu3Group:=Nost;
   Menu3.Body[1].SKip:=False;
 end
Else
  Begin
   FollMenu3Group:=YesST;
   Menu3.Body[1].Skip:=True;
  end;

If  Have_USer_dat then
 Begin
  Follmenu3User:=NoST;
```

```
    Menu3.Body[2].Skip:=False;
   end
  Else
   begin
     Follmenu3USer:=YesST;
     Menu3.Body[2].Skip:=True;
   end;
end;

Function   ThisStation:byte;
begin
  ThisStation:=Port[$306];
end;

{ ---------------------  SHOW  MENU  ---------------------- }
{              Show  Menu  And  Choice  in  Menu  .         }
{ ------------------------------------------------------- }

Procedure  TiTle;
Var  i : integer;
begin
 for i:=2 to 79 do  writefast(' ',$70,i,Headerline);
 writefast(Header,$70,40-length(Header) div 2,Headerline);
 writefast(Version,$70,77-length(version),Headerline);
 MyBox.W:=80;
 MyBox.H:=3;
```

```
 MyBox.X:=1;
 MyBox.Y:=1;
 box(Mybox);
 MyBox.H:=20;
 MyBox.Y:=StartWLine+1;
 mybox.Boxtype:=mybox.Boxtype or $85;
 MyBox.FillerAttr:=$07;
 MyBox.FillerChar:=Char(176);
 box(Mybox);
end;

Procedure   SuperVisorPROC;

Procedure   SVName;
CONST   X = 34;
        Y = 15;
Begin
  space11:=User_Name_Num_Ch;
  WriteinBox(X,Y,$87,$70,' '+space1+' ',' NAME ');
  GotoXY(X+1,Y);
  SetCursize(OCurstart,OCurstop);
  readChar(SV_NAME,Space11,Show);
  if SV_NAME[1] <> #27 then
   Begin
    FollMENU2Name:=Cut_Space_Two_Side(Bigchar(SV_NAME));
     Have_SV_Name:=True;
```

```
   End;
  SetCursize(7,6);
end;


Procedure  SVPassword;
CONST   X = 39;
        Y = 15;
begin
  space11:=User_Pass_Num_Ch;
  WriteinBox(X,Y,$87,$70,' '+space1+' ',' PASSWORD ');
  GotoXY(X+1,Y);
  SetCursize(OCurstart,OCurstop);
  readChar(SV_PASS,Space11,Notshow);   (* SHOW  or  NOT  SHOW  When  READ *)
  if SV_PASS[1] <> #27 then
    begin
     FollMENU2PASS:=Cut_Space_Two_Side(BIGCHAR(SV_PASS));
      Have_SV_Pass:=True;
    end;
   SetCursize(7,6);
End;


Begin   (* SuperVisorPROC *)
 S_Choice_P:=S_Choice_P^.Dforward;
 SaveThisScreen(StartSave,ByteMove);
```

```
Case  S_Choice_P^.Menu_Number of

 1:SVName;

 2:SVPassword;

end;

if Have_SV_Name and Have_SV_Pass then  Menu1.Body[3].Skip:=False;

ReturnLastscreen(StartSave,ByteMove);

End;    (* SuperVisorPROC *)


Procedure   OptionPROC;

Procedure   ChangeOfGroup;

begin

  if Menu3.Body[1].FollowST^=YesST  then  Menu3.Body[1].FollowST^:=NoST

                                    Else  Menu3.Body[1].FollowST^:=YesSt;

end;


Procedure   ChangeOfUser;

begin

  if Menu3.Body[2].FollowSt^=YesST  then  Menu3.Body[2].FollowST^:=NoST

                                    Else  Menu3.Body[2].FollowST^:=YesSt;

end;


begin

 S_Choice_P:=S_Choice_P^.Dforward;

 SaveThisScreen(StartSave,ByteMove);

 Case  S_Choice_P^.Menu_Number of

  1:ChangeOfGroup;
```

```
  2:ChangeOfUser;
 end;
 ReturnLastscreen(StartSave,ByteMove);

end;

Procedure  TellAndReadCh(x,y,SAttr,HAttr : Byte ;St,Headbox : String);
begin
  SaveThisScreen(StartSave,ByteMove);
  writeinbox(X,Y,SAttr,HAttr,St,HeadBox);
  Ch:=ReadKey;
  ReturnLastscreen(StartSave,ByteMove);
End;

Procedure  ExecutePROC;
Const    FiDataAttr = {Hidden + Readonly} Archive;
Var       i : integer;
begin
  {$I-}
  PathStrAccess:=SystemDrive + PV_LAN_DIRST;
  if  Not Have_PV_LAN_Dir then
   Begin    (*3*)
     MkDir(SystemDrive+PV_Lan_DirSt);
     if IOResult <> 0 then
     Begin
      TellAndReadCh(13,15,$07,$70,'Cannot create directory','Error');
```

```
    {$I+}
    Exit;
   End;
   Have_PV_LAN_Dir :=True;
   MkDir(SystemDrive + PV_Lan_DirSt+MailST);
   if IOResult <> 0 then
     Begin
      TellAndReadCh(13,15,$07,$70,' Cannot create mail directory ',' Error '
      {$I+}
      Exit;
     End;
   Have_Mail_Dir:=True;
End;   (*3*)
if Not Have_Mail_Dir then
    begin
     MkDir(SystemDrive + PV_Lan_DirSt + MailST);
     if IOResult <> 0 then
      Begin
        TellAndReadCh(13,15,$07,$70,' Cannot create mail directory ',' Erro
        {$I+}
       .Exit;
      End;
     Have_Mail_Dir:=True;
    End;

If  (FollMenu3User = YesST) or (Not Have_User_Dat) then
```

```
begin  (*4*)
  Assign(UFi,PathStrAccess+UserDST);
  if  Have_User_Dat  then
     Begin
      SetFAttr(UFi,Archive);
      Erase(UFi);
     End;
  Rewrite(UFi);
  if IOResult <> 0 then
    Begin
      TellAndReadCh(13,15,$07,$70,' Cannot create User file ',' Error ');
      {$I+}
      Exit;
    End;
  Close(UFi);
  SetFAttr(UFi,FiDataAttr);

  Assign(Accfi,PathStrAccess+MailST+'\*.USR');
  {$I-}
  Erase(Accfi);

 End;

if (FollMENU3GROUP=YesST) or (Not Have_Group_Dat) Then
  begin  (*1*)
    Assign(GFi,PathSTRAccess+GroupDST);
```

```
      if Have_Group_Dat then
       Begin
        SetFAttr(GFi,Archive);
        Erase(GFi);
       End;
      Rewrite(GFi);
(*      if IOResult <> 0 then
        Begin
          TellAndReadCh(13,15,$07,$70,' Cannot create Group file ',' Error ');
          {$I+}
          Exit;
        End;     *)
      Close(GFi);
      SetFAttr(GFi,FiDataAttr);
      Assign(Gfi,PathStrAccess+MailST+'\*.GRP');
      {$I-}
      Erase(Gfi);
    End;
  {$I-}
  Assign(Ufi,PathStrAccess+UserDST);
  Reset(UFi);
  Seek(Ufi,0);
  With accudata do
   begin
    Randomize;
    Table_Encode:=Random(10)+1;
```

```
      User_Number:=1;
      User_Name:=Encryp(Table_Encode,FollMENU2NAME);
      User_PassWord:=Encryp(Table_Encode,FollMENU2PASS);
      GMember_Number:=0;
      FillChar(GNumber,User_Have_G_Num,Chr(0));
    end;
    FindFirst(In_PV_LAn_Dir+Mailst+'\00000001.USR',Directory,DirInfo1);
    if  DosError <> 0 then
     begin
      MkDir(In_PV_Lan_dir+MailSt+'\00000001.USR');
     end;
     write(UFi,AccUdata);
     Assign(Fi3,systemdrive+PV_Lan_Dirst+MailSt+'\00000001.USR\NEWS');
     Rewrite(Fi3);
     Close(Fi3);
     Close(UFi);
   TellAndReadCh(14,15,$0F,$70,' INITIAL  ALL  RIGHT  / Stike  any  key  when
end;


begin
   CheckStatusOfdata;
   InitPointer_Of_AllMenu;
   writeln;
   SaveThisScreen(StartSave,ByteMove);
   SetCursize(7,6);
```

```
Clrscr;
Title;
Move(Vram^,ScreenRoot,MaxByteMove);
if Thisstation = 0 then
  begin  (* STATION = 0 *)
    Menu1.Init(1);
    Menu1.show(Mybox,HeaderAttr);
    Repeat
     CheckPress1(S_Choice_P,P_Menu);
     S_Choice_P:=Head_S_Choice_P;
     Case S_Choice_P^.Menu_Number of
      1:SuperVisorPROC;
      2:OptionPROC;
      3:ExecutePROC;
     End;
     Change_TO_NOW;
    Until S_Choice_P^.Menu_Number = 4;
End   (* STATION = 0 *)
Else
 Begin
   WriteinBox(15,12,$07,$07,' Can'+#39+'t  run  initial  process  on  this
 end;
ReturnLastscreen(StartSave,ByteMove);
SetCursize(OCurstart,OCurstop);
gotoxy(OCurX,OCurY);
end.
```

```
Unit Contacin;
Interface
Uses   Dos;
  Function     HiNibble(Num : byte) : byte;
  Function     LowNibble(Num : byte) : byte;
  Function     PackinByte(HiNib,LowNib : byte) :byte;
  Function     FlagsError(Status : byte) : Boolean;
  Procedure    SendChControl(Chb : byte);
  Procedure    SendStrData(Var Str : String);
  Procedure    SendChData(Chb : byte);
  Procedure    ReceiveControl(Var  Chb,Status : byte);
  Procedure    ReceiveData(Var Chb,Status : byte);


Implementation
Function     HiNibble(Num : byte) : byte;
Begin
  Hinibble:= Num Shr 4;
End;

Function     LowNibble(Num : byte) : byte;
Begin
  LowNibble :=Num and $0F;
End;

Function     PackinByte(HiNib,LowNib : byte) :byte;
Begin
  PackinByte:=(HiNib Shl 4) or (LowNib and $0F);
End;

Function     FlagsError(Status : byte) : Boolean;
```

```
Var       AccB : boolean;
Begin
  AccB:=(Status and $38) <> 0;
  if  AccB  then  Port[$301]:=$15;
  FlagsError:=AccB;
End;

Procedure   SendChControl(Chb : byte);
Var     Reg1 : Registers;
Begin
   Reg1.AL:=Chb;
   Reg1.Ah:=$02;
   Intr($66,Reg1);
End;

Procedure   SendStrData(Var Str : String);
Var     Reg2 : Registers;
Begin
   Reg2.AH:=$85;
   Reg2.Cx:=Length(Str);
   Reg2.Dx:=Seg(Str);
   Reg2.Bx:=Ofs(Str);
   Intr($66,Reg2);
End;

Procedure   SendChData(Chb : byte);
Var   Reg3 : Registers;
Begin
   Reg3.Ah:=$82;
   Reg3.Al:=Chb;
   Intr($66,Reg3);
```

```
End;

Procedure   ReceiveControl(Var  Chb,Status :byte);
Var   Reg4 : Registers;
Begin
   Reg4.AH:=$03;
   Intr($66,Reg4);
   Chb:=Reg4.Al;
   Status:=Reg4.AH;
End;

Procedure   ReceiveData(Var Chb,Status : byte);
Var   Reg5 : Registers;
Begin
   Reg5.AH:=$83;
   Intr($66,Reg5);
   Chb:=Reg5.Al;
   Status:=Reg5.AH;
End;

Begin
END.
```

บรรณานุกรม

1.น.ต.ดร. ไพศาล สงวนหมู่ และ รศ. ยืน ภู่วรวรรณ ,**การสื่อสารข้อมูล และ ไมโครคอมพิวเตอร์เนตเวอร์ค** กรุงเทพมหานคร : บริษัท ซีเอ็ดยูเคชั่น จำกัด,2528.

2. IBM Corporation, Technical Reference Personal Computer XT system. IBM Computer Hardware Reference Library, 6936808.

3. Joseph J.Carr, Microcomputer Interfacing Handbook : A/D&D/A. TAB BOOKS Inc, 350p., 1981

4. Microsoft Corporation, Microsoft Macro Assembler V4.0 for The MS-DOS Operation System: User's guide. Microsoft press, 1985

5. Texas Instrument Incorporated, The Bipolar Digital Integrated Circuits Data Book For Design Engineers, 1978

กิตติกรรมประกาศ

งานออกแบบระบบสื่อสารข้อมูลในเครือข่ายไมโครคอมพิวเตอร์แบบบัสนี้ ทางคณะผู้จัดทำขอขอบคุณ ผศ.ดร. รัตติกร วรากูลศิริพันธุ์ อาจารย์ที่ปรึกษา ซึ่งท่านกรุณาให้ความรู้และคำแนะนำซึ่งเป็นประโยชน์ต่อการทำงานเป็นอย่างยิ่ง และ คณะรุ่นพี่ปริญญาโทที่ได้ให้ความรู้ และ อำนวยความสะดวกในด้านต่างๆพร้อมทั้งให้คำปรึกษา นอกจากนี้ผู้จัดทำขอขอบคุณภาควิชาอิเลคโทรนิคส์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังที่ให้ความสะดวกในการใช้เครื่องมือทดลอง