ปีการศึกษา

ปริญญานิพนธ์เรื่อง

การออกแบบและพัฒนาโครงข่ายไมโครคอมพิวเตอร์

MCS - NETWORK SYSTEM AND APPLICATION SOFTWARE

โดย

นาย ธีรพร มณฑล 291073

นาย สุเมธ เลิศปัญญาพินิจกิจ 291255

อาจารย์ที่ปรึกษา

ดร. รัตติกร วรากูลศิริพันธุ์

ภาควิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปริญญานิพนธ์ปีการศึกษา 2532

เรื่อง การออกแบบและพัฒนาโครงข่ายไมโครคอมพิวเตอร์

MCS - NETWORK : System and Application Software

ผู้จัดทำ

1. นาย ธีรพร มณฑล 291073

2. นาย สุเมธ เลิศปัญญาพินิจกิจ 291255

.....อาจารย์ที่ปรึกษา

(ดร. รัตติกร วรากูลศิริพันธุ์)

ความสำคัญและที่มา

ปัจจุบันไมโครคอมพิวเตอร์ได้เข้ามาที่บทบาท ในขอบข่ายงานต่าง ๆ ซึ่งมีการใช้เครื่องไมโครคอมพิวเตอร์หลาย ๆ เครื่องมาต่อทำงานร่วมกันเป็นระบบโครงข่าย หรือนำเอามาต่อกับอุปกรณ์ภายนอกอื่น ๆ เข้าด้วยกันเป็นระบบงานที่ซับซ้อนขึ้น

เนื่องจากความต้องการใช้งานเครื่องไมโครคอมพิวเตอร์ เพิ่มมากขึ้น ยังผลให้เกิดความต้องการบุคคลากรที่มีความรู้ความสามารถในด้านนี้เพิ่มสูงขึ้นด้วย ทำให้คณะผู้จัดทำมีความคิดที่จะใช้เครื่องไมโครคอมพิวเตอร์มาทำเป็นระบบโครงข่ายเพื่อช่วยในเรื่องการเรียนการสอนขึ้น

ปัญหาที่สำคัญที่ตามมาก็คือ จะทำอย่างไรให้อาจารย์สามารถควบคุมเครื่องไมโครคอมพิวเตอร์ ที่ใช้ในการเรียนการสอนนี้ และจะทำอย่างไรให้นักเรียนทุกคนสามารถติดตามบทเรียนได้อย่างต่อเนื่องและมีประสิทธิภาพ เพื่อที่จะได้อำนวยความสะดวกระหว่างผู้เรียนกับผู้สอนโดยสมบูรณ์ ฉะนั้นจึงได้มีการนำเครื่องไมโครคอมพิวเตอร์หลาย ๆ เครื่องมาติดต่อกันเป็นระบบโครงข่ายเพื่อใช้ในงานนี้โดยเฉพาะ

การจัดโครงข่ายเครื่องไมโครคอมพิวเตอร์นั้นยังมีราคาสูงอยู่ และไม่เหมาะสมในการนำมาประยุกต์ใช้งาน ดังนั้น ปริญญานิพนธ์ฉบับนี้ได้กล่าวถึงหลักการออกแบบโครงข่ายไมโครคอมพิวเตอร์มุ่งที่จะแก้ปัญหาดังกล่าว โดยได้มีการออกแบบพัฒนาโครงข่ายไมโครคอมพิวเตอร์ใช้ชื่อว่า Master Control Slave Network (MCS-Network) เพื่อใช้งานกับเครื่องไมโครคอมพิวเตอร์ของ IBM PC XT หรือเครื่อง Compatible ต่าง ๆ ในงานการควบคุมการเรียนการสอนในชั้นเรียนให้มีประสิทธิภาพมากที่สุด โดยมีราคาถูกรวมทั้งสะดวกในการติดตั้งและขยายโครงข่าย ซึ่งยังสามารถนำไปประยุกต์ใช้งานอื่น ๆ ได้อีกด้วย เช่น การสาธิตในงานนิทรรศการ และแม้แต่ภายในสำนักงานเพื่อใช้ในการสื่อสารในแต่ละแผนก โดยการเลือกใช้งานโครงข่ายในลักษณะเทอร์มินอล เป็นต้น

ปริญญานิพนธ์ฉบับนี้เป็นการพัฒนาโครงข่าย Network แบบ MCS ของปริญญานิพนธ์ฉบับก่อน

ขอบเขตของโครงงาน

1. พัฒนารูปแบบของตัวอักษรจากอักษรภาษาอังกฤษล้วน มาเป็นแบบภาษาไทย 25 บรรทัด ซึ่งสามารถใช้ได้ทั้งภาษาไทยและอังกฤษ ซึ่งโครงงานนี้ใช้ Thai Card ของ Thai Champion

2. เครื่อง Master ส่งข้อความจาก Keyboard ไปทีละตัวอักษรไปปรากฏที่จอของ Slave ที่ตำแหน่งใดก็ได้ตามแต่ที่ Program จะเขียนไว้ ซึ่งการทำงานในลักษณะนี้ต้องทำการควบคุมไม่ให้เครื่อง Slave ใช้ Keyboard ของตนเองได้

3. เครื่อง Master ส่งข้อความจาก Script การเรียนการสอน ไปปรากฏที่จอของ Slave ที่ตำแหน่งใดก็ได้ตามแต่ที่ Program จะเขียนไว้

4. การเขียน Program จะใช้ทั้งภาษา C & Assembly ร่วมกัน ขึ้นอยู่กับความเหมาะสมของงาน

ประโยชน์ที่คาดว่าจะได้รับ

1. ใช้ภาษาไทยในการติดต่อสื่อสาร เพื่อความสะดวกในการใช้งานทั้ง USER ที่เป็นทั้ง Master & Slave ซึ่งเหมาะสำหรับคนไทยโดยทั่ว ๆ ไป เพราะมีภาษาไทยในการเรียนการสอนโดยเฉพาะ

2. มีมาตราฐานในการ Control Character โดยทั่ว ๆ ไป เพื่อความสะดวกในการควบคุมและการทำงานต่าง ๆ

3. ประโยชน์ที่ได้จากการเตรียม Script การสอนซึ่งเตรียมไว้ก่อนแล้วนั้น ทำให้การเรียนการสอนทำได้อย่างรวดเร็ว และคำพูดที่ใช้ได้ใจความและถูกต้องกระทัดรัดเพียงแค่ Master กด Key เพียงไม่กี่ตัวเท่านั้น

สารบัญ

บทที่ 1

บทนำ

เนื่องจาก ความต้องการที่จะใช้งานไมโครคอมพิวเตอร์ได้เพิ่มมากขึ้นในปัจจุบัน บุคคลากรทางงานด้านนี้ ก็ได้เริ่มนำเทคโนโลยีทางโครงข่าย (NETWORK) มาเพื่อที่จะใช้งานและเพื่อเพิ่มประสิทธิภาพในการส่งข้อมูลจาก หน่วยงานหนึ่งไปยังหน่วยงานอื่น ๆ ซึ่งยังผลให้บุคคลากรมีแนวโน้มที่ต้องการมากขึ้น แต่กำลังที่จะผลิตบุคคลากรที่มีความรู้ความสามารถในทางการใช้งานไมโครคอมพิวเตอร์ ยังอยู่ในเกณฑ์ต่ำ ซึ่งเราจะทำอย่างไรที่จะสามารถเพิ่มจำนวนบุคคลากรเหล่านี้ ให้เพิ่มสูงขึ้นได้ โดยที่เมื่อเทียบกับเวลาแล้วใช้เวลาน้อยที่สุด และเทคโนโลยีที่มาจากต่างประเทศ และใช้งานอยู่ในปัจจุบันนั้น มีราคาแพงไม่เหมาะกับการประยุกต์ใช้งาน จึงจำเป็นอย่างยิ่งที่จะต้องคิดค้นหรือดัดแปลงโครงข่าย (Communication Network) โดยการใช้อุปกรณ์ในประเทศเรา

ปริญญานิพนธ์ ฉบับนี้ได้กล่าวถึง อุปกรณ์ที่มีความสามารถในการควบคุมการติดต่อสื่อสารระหว่างเครื่องไมโครคอมพิวเตอร์ตัวแม่ (MASTER COMPUTER) กับไมโครคอมพิวเตอร์ตัวลูก (SLAVE COMPUTER) ในลักษณะ การติดต่อจากตัวแม่ไปตัวลูกแบบจุดต่อจุด (POINT TO POINT) แบบหลายจุด (MULTIPOINT) รวมทั้งรายละเอียดต่าง ๆ ที่ใช้ในการทำงานและติดต่อระหว่างไมโครคอมพิวเตอร์ตัวแม่และตัวลูก ซึ่งมีรายละเอียดแต่ละบทดังนี้

บทที่ 1 บทนำ แนะนำเนื้อหาและวัตถุประสงค์โดยย่อของโครงงานนี้

บทที่ 2 ทฤษฎีเบื้องต้นในการสื่อสารข้อมูล กล่าวถึงการสื่อสารเบื้องต้นและลักษณะการต่อสายการติดต่อสื่อสาร RS-232C แบบกว้าง ๆ เพื่อเป็นแนวทางในการออกแบบอุปกรณ์ควบคุมโครงข่าย (Communication Controler Box)

บทที่ 3 ได้กล่าวถึงหลักการทาง hardware อย่างกว้าง ๆ และรูปแบบการต่อโครงข่ายการสื่อสาร

บทที่ 4 และบทที่ 5 ได้อธิบายสรุปการใช้งาน โปรแกรมที่เปลี่ยนแนวไปจากเดิม รวมทั้งผลการทดลอง

บทที่ 6 จะเป็นการสรุปผล วิจารณ์ และแนวทางที่พัฒนาต่อ

ภาคผนวก กิตติกรรมประกาศ และหนังสืออ้างอิง

บทที่ 2

ทฤษฎีเบื้องต้นในการสื่อสารข้อมูล

2.1 การอินเทอร์เฟสไมโครโปรเซสเซอร์

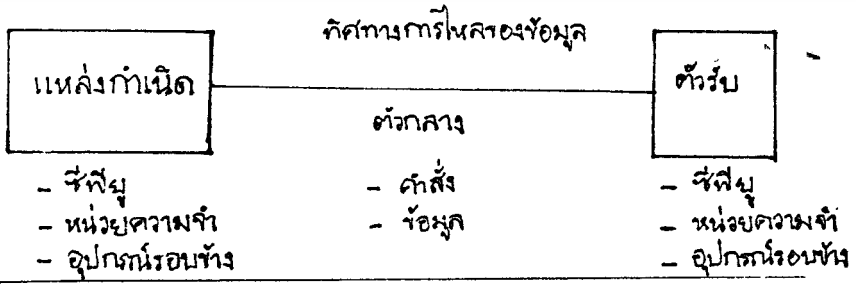
การอินเทอร์เฟสไมโครโปรเซสเซอร์ก็คือ การทำงานร่วมกันระหว่าง ซีพียู กับอุปกรณ์อื่น ๆ ในการโอนย้ายข้อมูลระหว่างอุปกรณ์ต่าง ๆ บนแผ่นวงจร ซีพียู จะต้องทำงานสอดคล้องกับ ROM, RAM แล้วยังต้องอินเทอร์เฟสเข้ากับ อุปกรณ์อินพุท, เอาท์พุท ต่าง ๆ อีกด้วย เพื่อเพิ่มประสิทธิภาพในการทำงานให้สมบูรณ์ยิ่งขึ้น ในขบวนการต่าง ๆ ของการทำงานร่วมกันของอุปกรณ์อิเลคทรอนิคส์ จะต่อเนื่องกันเป็นลูกโซ่

2.2 ส่วนประกอบเบื้องต้นในการสื่อสาร

ส่วนประกอบในการสื่อสารข้อมูลแบ่งได้เป็น 3 ส่วน คือ

- แหล่งกำเนิด (SOURCE)
- ตัวกลาง (MEDIUM)
- ตัวรับ RECEIVER)

แสดงความสัมพันธ์ของทั้ง 3 ส่วนได้ดังรูป 2.1 คือ

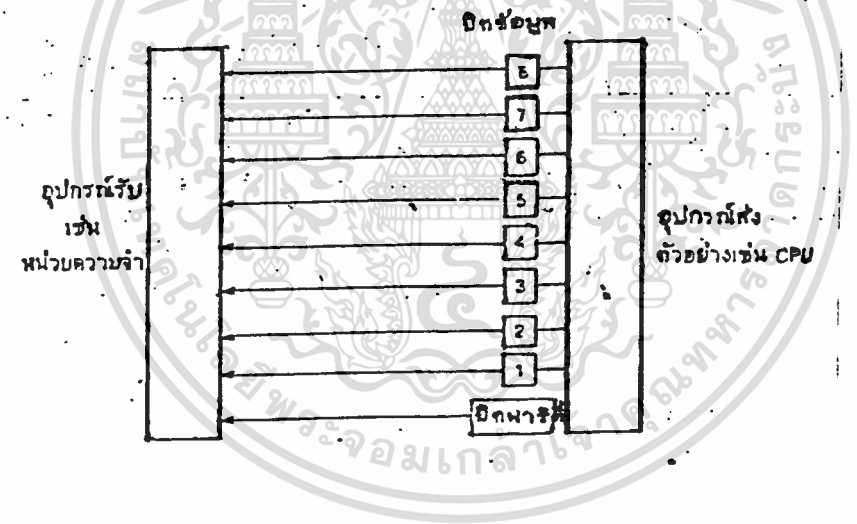


รูปที่ 2.1 แสดงความสัมพันธ์ขององค์ประกอบหลักในการสื่อสาร

2.3 วิธีถ่ายโอนข้อมูล

หลักใหญ่ ๆ ในการส่งข้อมูลในคอมพิวเตอร์ หรือระหว่างคอมพิวเตอร์ด้วยกันมีลักษณะของการส่งข้อมูลอยู่ 2 แบบ คือ ส่งแบบขนานและส่งแบบอนุกรม

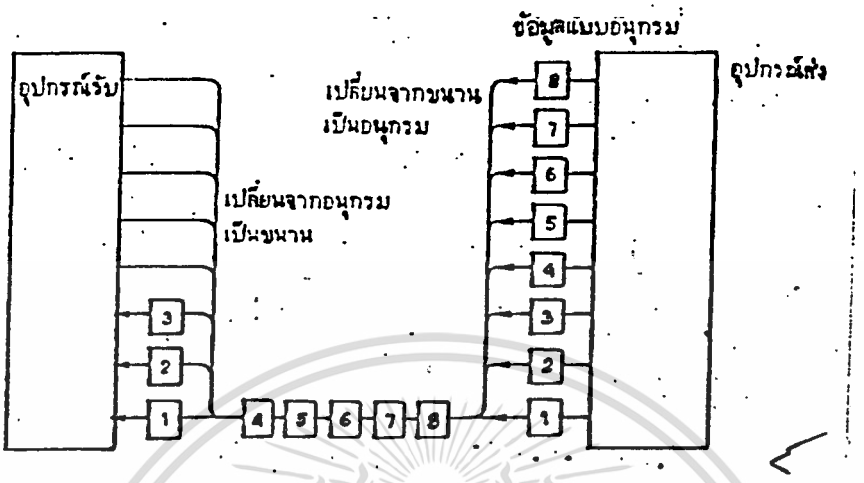
2.3.1 การโอนถ่ายข้อมูลแบบขนาน ทำได้โดยการส่งข้อมูลออกมาพร้อม ๆ กันทุกบิท ดังนั้นช่องทางที่ข้อมูลใช้ในการเดินทาง จึงมีจำนวนเท่ากับจำนวนบิทข้อมูลที่รับส่งกัน การถ่ายโอนแบบนี้มักทำในระยะทางใกล้ ๆ ทั้งนี้เพื่อกันปัญหาเรื่องระดับกราวด์ทางไฟฟ้าระหว่างจุดส่งและจุดรับต่างกันเนื่องจากความต้านทานของสาย ลักษณะการถ่ายโอนแบบนี้แสดงดังรูปที่ 2.2 อยู่หน้าถัดไป



รูปที่ 2.2 แสดงการถ่ายโอนแบบขนาน

2.3.2 การถ่ายโอนข้อมูลแบบอนุกรม ทำได้โดยการส่งสัญญาณข้อมูลออกมาทีละบิท เรียงลำดับกันระหว่างจุดส่งและจุดรับ ถึงแม้จะทำได้ช้าแต่มีจุดเด่นคือประหยัดสายกว่า เพราะใช้สายเพียงคู่เดียวเท่านั้น การถ่ายโอนแบบนี้ เหมาะสมกับการส่งในระยะทางไกล ๆ เพราะ

เป็นวิธีที่ประหยัดสายส่งได้เป็นจำนวนมาก ลักษณะการส่งดังแสดงดัง รูปที่ 2.3



รูปที่ 2.3 แสดงการถ่ายโอนแบบอนุกรม

การถ่ายโอนแบบนี้จำเป็นต้องได้รับการแปลงกลับอย่างเหมาะสม เป็นสัญญาณแบบขนาน เมื่อมาถึงทางด้านรับซึ่งมีโปรโตคอล (PROTOCOL) หลักที่จัดการในเรื่องนี้ 2 ประเภท คือ

1) ข้อกำหนดแบบซิงค์โครนัส (SYNCHRONOUS PROTOCOL)

2) ข้อกำหนดแบบอะซิงค์โครนัส (ASYMCHRONOUS PROTOCOL)

ข้อเปรียบเทียบระหว่างการส่งข้อมูลแบบขนานและแบบอนุกรม

	แบบขนาน	
1. ระยะทาง	ปกติจะน้อยกว่า 100 ฟุต	ส่งได้ตั้งแต่ระยะสั้น ๆ จนถึงระยะทางเป็นไมล์
2. ความเร็ว	อัตราความเร็วสูงมากในระยะที่ไม่ไกลมากนัก กำหนดได้เป็นจำนวนบิทต่อวินาที	อัตราความเร็วของข้อมูลที่ใช้กันอยู่ทั่วไปจะอยู่ในช่วง 0 ถึง 2 ล้านบิทต่อวินาที
3. ระดับของสัญญาณ	ในการอินเตอร์เฟสจะใช้ระดับสัญญาณที่ใช้กับอุปกรณ์ TTL คือ สัญญาณลอจิก 1 และ 0 จะแทนด้วยระดับแรงดัน +5V และ 0V ตามลำดับ	ใช้มาตรฐานของ EIA-RS 232C คือมีระดับสัญญาณไฟฟ้าขนาด 12V หรืออาจจะใช้มาตรฐาน 20 mA Current loop หรืออาจจะใช้ระดับสัญญาณของ TTL ก็ได้(ใช้กันน้อยมาก)
4. ความผิดพลาดของสัญญาณ	ถ้าส่งในระยะทางไกล ๆ ความผิดพลาดของข้อมูลจะเกิดขึ้นง่าย	การผิดพลาดของสัญญาณจะมีน้อยลง
5. ค่าใช้จ่าย	ถ้าส่งในระยะทางไกลๆ จะสิ้นเปลืองค่าใช้จ่ายมาก เพราะต้องใช้สายส่งสัญญาณหลายเส้น	สิ้นเปลืองน้อยกว่าหลายเท่า ถึงแม้ว่าจะใช้อุปกรณ์เปลี่ยนสัญญาณของข้อมูลจากแบบขนานไปเป็นแบบอนุกรม แล้วส่งผ่านสายส่งใช้อุปกรณ์ในการแปลงสัญญาณกลับมาเป็นแบบขนานอีก ก็ยังลงทุนน้อยกว่า

2.4 ทิศทางการส่งข้อมูล

ทิศทางของการรับส่งข้อมูลจำแนกได้ 3 ลักษณะ คือ

2.4.1 การส่งแบบทิศทางเดียว (SIMPLEX หรือ ONE-WAY TRANSMISSION)

การส่งแบบนี้ทิศทางการส่งจะคงที่ โดยการกำหนดในครั้งแรกว่า ฝ่ายใดเป็นฝ่ายส่งฝ่ายใดเป็นฝ่ายรับ การส่งแบบนี้ไม่คล่องตัวแต่ก็มีความเหมาะสมเป็นอย่างมากในการประยุกต์ใช้งานบางประเภท

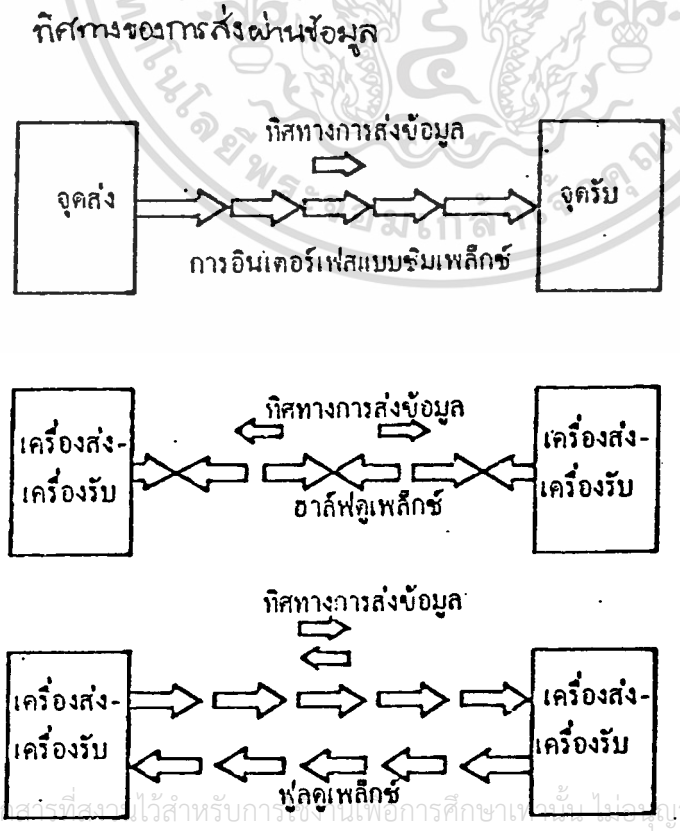
2.4.2 การส่งแบบฮาล์ฟดูเพล็กซ์ (HALF DUPLEX)

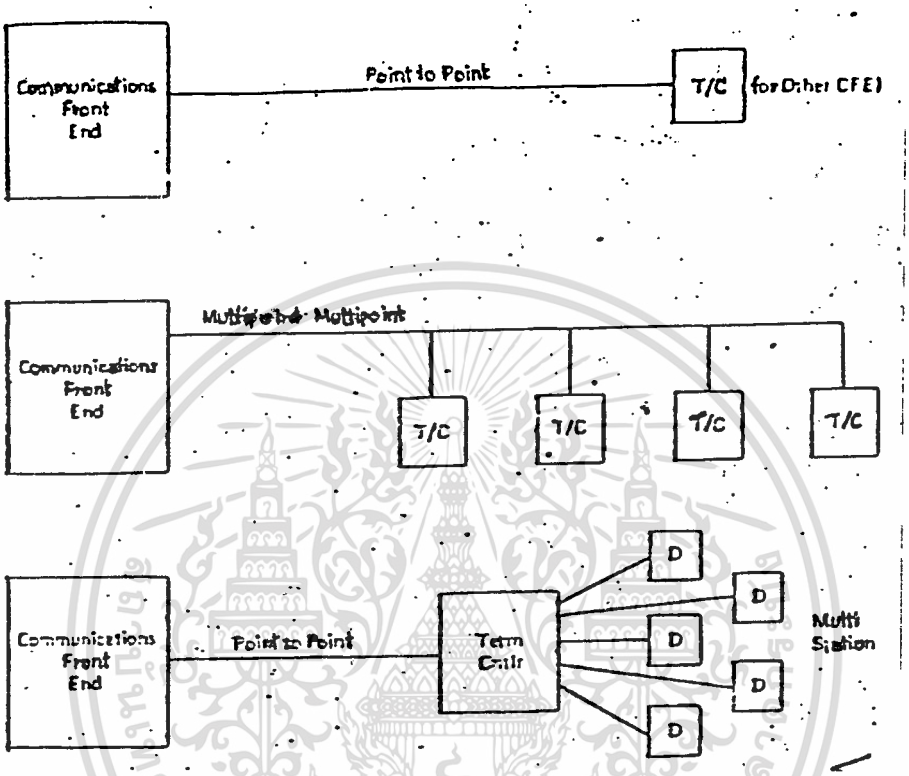
เป็นการส่งในทิศทางใดทิศทางหนึ่งในเวลาใดเวลาหนึ่ง กล่าวคือทั้ง 2 ฝ่าย หรือทั้ง 2 สถานีสามารถผลัดกันส่งได้ แต่จะส่งพร้อมกันไม่ได้ต้องผลัดกันรับผลัดกันส่ง

2.4.3 การส่งแบบฟูลดูเพล็กซ์ (FULL DUPLEX)

เป็นการส่งในทิศทางใดๆ ก็ได้ในเวลาหนึ่งๆ กล่าวคือทั้ง 2 สถานีสามารถรับ - ส่งพร้อมกันได้ในเวลาเดียวกัน

ลักษณะการรับ - ส่ง แบบต่างๆ ที่กล่าวมานั้นแสดงได้ดังรูปที่ 2.4





รูปที่ 2.5 แสดงลักษณะการต่อสายสื่อสาร

2.5 ลักษณะการต่อสายสำหรับการสื่อสาร

แบ่งได้เป็น 2 ลักษณะ คือ

2.5.1 แบบจุดต่อจุด (POINT TO POINT)

2.5.2 แบบหลายจุด (MULTIPOINT)

ดังแสดงในรูปที่ 2.5

2.6 โครงข่ายไมโครคอมพิวเตอร์

ลักษณะของการเชื่อมต่อไมโครคอมพิวเตอร์เข้าเป็นโครงข่าย ก็เพื่อให้เกิดความสะดวกรวดเร็ว และมีประสิทธิภาพเพิ่มขึ้น

ลักษณะโครงข่ายแบ่งตามระยะทางโปรเซสเซอร์ที่เชื่อมต่อกันดังตารางที่ 2.1

ระยะทางระหว่างโปรเซสเซอร์	ลักษณะที่ตั้งของโปรเซสเซอร์	ชื่อเรียกเนตเวอร์ค
0.1 เมตร	บอร์ดอิเล็กทรอนิกส์	เครื่องจักรชนิดสถาโฟลว์
1.0 เมตร	ระบบเดียวกัน	มัลติโปรเซสเซอร์
10.0 เมตร	ห้อง	
100.0 เมตร	ตัวอาคาร	โลคัลเนตเวอร์ค
1 กิโลเมตร	หน่วยงานเดียวกัน	
10 กิโลเมตร	เมือง	
100 กิโลเมตร	ประเทศ	เนตเวอร์ระยะไกล
1000 กิโลเมตร	ระหว่างประเทศ	
10000 กิโลเมตร	ระหว่างดวงดาว	เนตเวอร์คระยะไกลมาก

ตารางที่ 2.1 แสดงการแบ่งแยกลักษณะโครงข่ายคอมพิวเตอร์

บทที่ 3

ความรู้ทางด้านฮาร์ดแวร์ของ IBM PE/XT

ในการออกแบบและพัฒนาโครงข่าย MCS-Network ซึ่งผู้เขียนจะได้กล่าวถึงต่อไปส่วนที่ 2 ของปริญญานิพนธ์ฉบับนี้นั้น ใช้ความรู้ความเข้าใจในเรื่องส่วนประกอบทางฮาร์ดแวร์และ BIOS ของ IBM PC/XT เป็นอย่างมากในการเขียนโปรแกรมระบบจัดการโครงข่าย (Network Operating System : NOS)

ในบทนี้นั้นจึงจะได้กล่าวถึงความรู้ดังกล่าวในส่วนฮาร์ดแวร์ที่เกี่ยวข้องกับโครงงานโดยเฉพาะในเรื่องคาร์ดควบคุมการสื่อสาร (Communication adapter card) ส่วนเรื่อง HARDWARE ของเครื่อง IBM PC/Xt จะไม่ขอกล่าว ณ.ที่นี้

3.1 **คาร์ดควบคุมการสื่อสาร** ในการสื่อสารข้อมูลกันระหว่างเครื่องตระกูล IBM PC นั้นนิยมทำการติดต่อกันผ่านทางฟอร์ทสื่อสาร ตามมาตรฐาน RS-232C ของ EIA โดยมีการกำหนดเป็นฟอร์ท COM1 และ COM2 ซึ่งจะเลือกใช้ได้โดยการกำหนดค่าแก่ดิฟสวิทช์บนคาร์ดควบคุมการสื่อสาร

คาร์ดควบคุมการสื่อสารที่มีใช้กันอยู่มีทั้งของ IBM เอง และที่เทียบเท่า (compatible) เช่น Multi I/O card และ multifunction card

ในส่วนนี้นั้นจะได้อธิบายโดยยกตัวอย่างคาร์ดควบคุมการสื่อสารของบริษัท IBM เอง ส่วนรายละเอียดของคาร์ด multifunction และคาร์ด multi I/O นั้นศึกษาได้จากภาคผนวกท้ายเล่ม

3.1.1 <u>คาร์ดควบคุมการสื่อสาร</u> (IBM Asychronous Communication adepter)

ก. มีอัตราความเร็วของการรับส่งข้อมูล จาก 50 baud บอด ถึง 9600 บอด

ข. สามารถโปรแกรมการทำงานได้

ค. ใช้ควบคุมการสื่อสารแบบอะซิงโครนัส

หัวใจสำคัญของคาร์ด คือ ไอซี INS8250 LSI ซึ่งมีจุดเด่นดังต่อไปนี้

1. มีบัฟเฟอร์ (buffer) เป็น 2 เท่า ทำให้ตัดความจำเป็นในการซิงโครไนซ์ (synchronization) ที่ละเอียดถี่ถ้วนออกไป

2. มีสัญญาณนาฬิกา ที่เป็นอิสระของตัวเอง

3. มีฟังก์ชั่นการควบคุมต่าง ๆ ที่เกี่ยวกับ MODEM คือ

   Clear to Send (CTS)

   Request to Send (RTS)

   Data Set Ready (DSR)

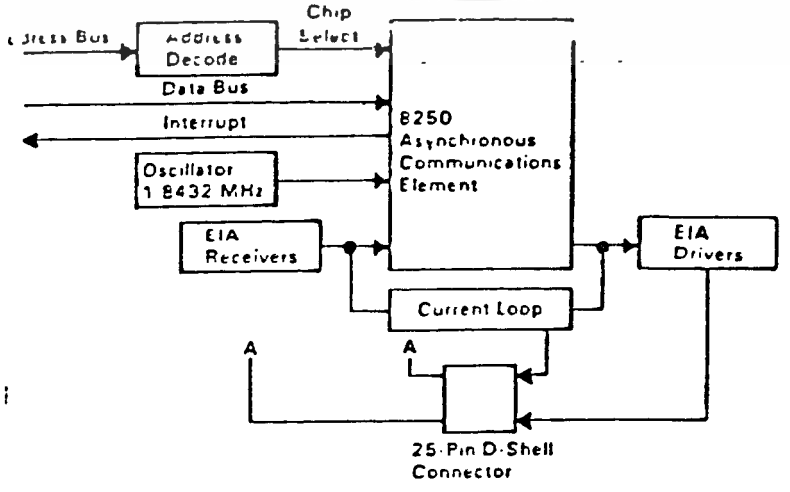
   Ring Indicator (RI)

   Carrier Detect

4. มีฟังก์ชั่นการตรวจสอบความผิดพลาดของบิทเริ่มต้น (start bit)

5. มีฟังก์ชั่นสร้างและตรวจจับสัญญาณ line break)

โปรแกรมการสื่อสารทั้งหมด (communication protocol) จะต้องโหลดเข้ามาในเครื่องเสียก่อนการใช้งานคาร์ดควบคุมการสื่อสารนี้



รูปที่ 3.1 แสดงแผนภูมิของคาร์ดควบคุมการสื่อสารแบบอะซิงโครนัสนี้

3.1.2 การทำงาน

ลักษณะต่าง ๆ ของการใช้งานทำโดยการโปรแกรม 8250 ด้วยวิธีการเลือกตำแหน่งที่อยู่ของอุปกรณ์อินพุทเอาท์พุท (3F8H ถึง 3FFH สำหรับ COM1 และ 2F8 ถึง 2FF สำหรับ COM2) แล้วเขียนค่าต่าง ๆ ให้แก่คาร์ด

ดังรูปที่ 3.2 การเลือกรีจีสเตอร์ต่าง ๆ เพื่อกำหนดลักษณะการสื่อสารทำโดยการกำหนดค่าแก่บิทที่ A0, A1, A2 ดังแสดงในรูปที่ 3.3

I/O Decode (in Hex)			
Pimary Adapter	Alterante Adepter	Register Selected	DLAB State
3F8	2F8	TX Buffer	DLAB = 0(Write)
3F8	2F8	RX Buffer	DLAB = 0(Read)
3F8	2F8	Divisor Latch LSB	DLAB = 1
3F9	2F9	Divsor Latch MSB	DLAB = 1
3F9	2F9	Interrupt Enable Registers	
3FA	3FA	Interrupt Identlication Registers	
3FB	2FB	Line Control Register	
3FD	2FC	Modem Control Register	
3FD	2FD	Line Status Register	
3FE	2FE	Modem Status Register	

รูปที่ 3.2 แสดงตำแหน่งที่อยู่ของรีจีสเตอร์ต่าง ๆ ของ 8250 ทั้งกรณี COM1 และ COM2

Hex Address 3F8 to 3FF and 2F8 to 2FF

A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	DLAB	Register
1	1 0	1	1	1	1	1	x	x	x		
							0	0	0	0	Receivt Buffer (terd) Transmit HoldingReg (write)
							0	0	1	0	Interrupt Enable
							0	1	0	x	Interrupt Indenufic-ation
							0	1	1	x	Line Control
							1	0	0	x	Modem Control
							1	0	1	x	LineStatus
							1	1	0	x	Modem Status

Hex Address 3F8 to 3FF and 2F8 to 2FF											
							1	1	1	x	None
							0	0	0	1	Divisor Latch(LSB)
							0	0	1	1	Divisor Latch(MSB)

Note : Bit 8 will logical 1 for the adapter designated as primary or a logical 0 for the adapter designated as alternate(as defined by the address jumper module on the address jumper module on the adapter)

A2.A1 and A0 bits are "don't cares" and are used to select the different register of the communication chip

รูปที่ 3.3 สรุปการกำหนดค่าแก่บิท A0, A1, A2 เพื่อเลือกรีจีสเตอร์ต่าง ๆ และกำหนดลักษณะการทำงาน

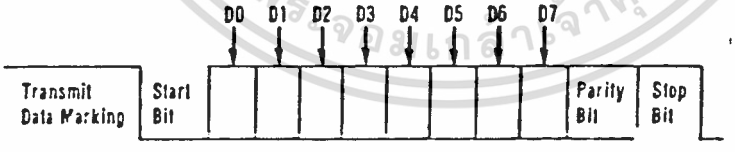
3.1.3 <u>การขัดจังหวะ</u>

ในการขอขัดจังหวะของอุปกรณ์ฮาร์ดแวร์นั้น ทำโดยการให้สัญญาณผ่านสายสัญญาณ IRQ (interrupt request line) โดยมีรายละเอียดสรุปได้ดังนี้

Port Configunation		I/O port	IRQ line
primary	COM1	3F8H-3FFH	IRQ4
alternate	COM2	2F8H-2FFH	IRQ3

โดยจะต้องกำหนดค่า "1" แก่บิทที่ 3 ของ modem control register เพื่ออนุญาตให้คาร์ดควบคุมการสื่อสารนี้ส่งสัญญาณขอขัดจังหวะไปยังระบบได้ จากนั้นการขัดจังหวะใดๆ ที่ได้รับอนุญาตจาก interrupt enable register แล้ว จะมีผลทำให้เกิดการขัดจังหวะแก่ระบบต่อไป

รูปแบบของการส่งข้อมูลแบบอะซิงโครนัสเป็นดังรูปที่ 3.4



รูปที่ 3.4 แสดงรูปแบบของการส่งข้อมูลแบบอะซิงค์โครนัส

บิทศูนย์ ของข้อมูลจะเป็น บิทแรกที่ถูกส่งออกไปหรือรับได้ โดยคาร์ดจะแทรกบิทเริ่มต้น พาริตี้บิท และบิทสุดท้ายให้โดยอัตโนมัติตามที่ได้กำหนด

3,1.4 <u>การเชื่อมต่อโดยผ่านหัวต่อ DB-25 ตามมาตราฐาน RS-232C E1A</u>

RS-232C เป็นการเชื่อมต่อโดยแรงดัน ตามมาตราฐานของ EIA ใช้หัวต่อแบบ DB-25 ขาสัญญาณควบคุมและข้อมูลเป็นดังต่อไปนี้

ขาสัญญาณ	ชื่อสัญญาณ
PIN 2	Transmitted data
PIN 3	Received Data
PIN 4	Request to Send
PIN 5	Clear to Send
PIN 6	Data Set Ready
PIN 7	Signal Ground
PIN 8	Carrier Detect
PIN 20	Data Terminal Ready
PIN 22	Ring Indicator

คาร์ดควบคุมจะทำการแปลงระดับสัญญาณเหล่านี้จากระดับ TTL ไปเป็นระดับแรงดันตามข้อกำหนดของ EIA ซึ่งซอฟแวร์ จะสามารถตรวจสอบเพื่อทราบสถานะของการเชื่อมต่อและอุปกรณ์รอบนอกได้

3.1.5 <u>การเปลี่ยนแปลงระดับแรงดัน</u>

จะถือว่าสัญญาณเป็น marking เมื่อแรงดันจากวงจรเปลี่ยนแปลงระดับแรงดัน (interchange circuit) วัดที่จุดต่อเป็นลบมากกว่า -3 V_{dc} เทียบกับระดับกราวด์และเป็น spaceing เมื่อเป็นบวกมากกว่า +3 V_{dc} เทียบกับระดับกราวด์ ขอบเขตระหว่าง +3

V_{dc} ถึง $-3\ V_{dc}$ จะถือว่าเป็นทรานซิชั่นรีเจียน (transition region) และไม่มีความหมาย สำหรับค่าแรงดันที่อยู่นอกเหนือค่าเหล่านี้ถือว่าไม่มีความหมายเช่นกัน

ในระหว่างการสื่อสารจะถือว่า marking คือ โลจิค "1" และ spacing คือ โลจิค "0"

วงจรควบคุมการเชื่อต่อจะ "ON" เมื่อระดับแรงดันมากกว่า $+3\ V_{dc}$ และ "OFF" เมื่อระดับแรงดันเป็นลบมากกว่า $-3\ V_{dc}$ เทียบกับระดับกราวด์

รูปที่ 3.5 แสดงรายละเอียดเกี่ยวกับระดับสัญญาณของคาร์ดควบคุมการสื่อสารในการสื่อสารตามมาตราฐาน RS-232C

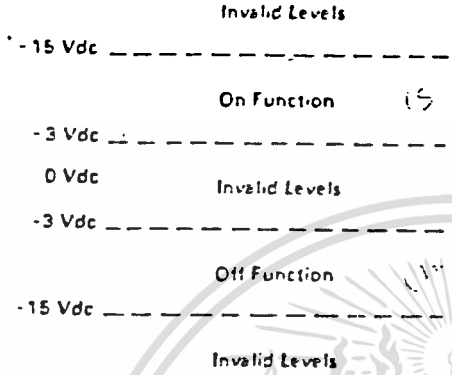
3.1.6 การโปรแกรม 8250

8250 มีรีจิสเตอร์ 10 ตัวที่สามารถควบคุมได้โดยผ่าน 8088 ตารางที่ 3.4 แสดง รีจีสเตอร์และตำแหน่งที่อยู่ของรีจีสเตอร์ต่าง ๆ ใน 8250

ตารางที่ 3.1 แสดงตำแหน่งที่อยู่ของรีจิสเตอร์ต่าง ๆ ใน 8250 กรณีเลือก COM21

I/O Port Address	Input or Output	Register Selected
3F8H*	Output	Transmitter Holding Register
3F8H*	Input	Receiver Data Register
3F8H*	Output	Baud-Rate Divisor (LSB)
3F9H*	Output	Baud-Rate Divisor (MSB)
3F9H*	Output	Interrupt-Enable Register
3FAH	Input	Interrupt-Identification Register
3FBH	Output	Line-Control Register
3FCH	Output	Modem-Control Register
3FDH	Input	Line-Status Register*
3FEH	Input	Modem-Status Register*

Interchange Voltage	Binary State	Signal Condition	Interface Control Function
Positive Voltage =	Binary (0)	= Spacing	= On
Negative Voltage =	Binary (1)	= Marking	= Off



รูปที่ 3.22

รูปที่ 3.5

ในการควบคุม buad rate diviser register นั้นเราจะต้องกำหนดค่าบิทสูงของ line control register เป็น "1" เสมอ และเป็น "0" ในเวลาอื่น ๆ

มีรีจิสเตอร์อยู่ 5 ตัวที่เราจะโปรแกรมได้โดยการใช้คำสั่ง OUT ในการกำหนดค่าเริ่มต้น (intial value) แก่ 8250 และจะไม่สนใจอีกต่อไปเมื่อผ่านการกำหนดค่าเริ่มต้นแก่ 8250 แล้ว คือ

1) Baud Rate Diviser (LSB)

2) Baud Rate Diviser (MSB)

3) Line Control Register

4) Modem Control Register

5) Interrupt Enable Register

ในระหว่างการเรียกใช้ NOS ของ MCS-Network นั้น line status register จะใช้ตัดสินว่าในขณะนั้นกำลังส่งหรือรับข้อมูลอยู่ คีย์สโตรกจะส่งออกไปยัง transmitler holding register โดยผ่านทาง transmitler shift register ก่อนและรับเข้าทางฝายสเลฟโดยผ่านทาง receiver shift register ไปยัง received data register เมื่อรับครบทุกบิทแล้ว

3.1.7 การกำหนดสภาวะเริ่มต้นแก่ 8250 โดยตรง(ไม่ผ่าน BIOS)

ก่อนที่จะใช้ 8250 ในการสื่อสารแบบอนุกรมได้นั้น จะต้องกำหนดค่าต่าง ๆ ให้ถูกต้องเหมาะสมเสียก่อน มีพารามิเตอร์อยู่หลายตัวที่ต้องกำหนดค่าเพื่อให้สอดคล้องกับพารามิเตอร์ควบคุมการสื่อสารของโครงข่าย MCS-Network

รายละเอียดมีดังต่อไปนี้

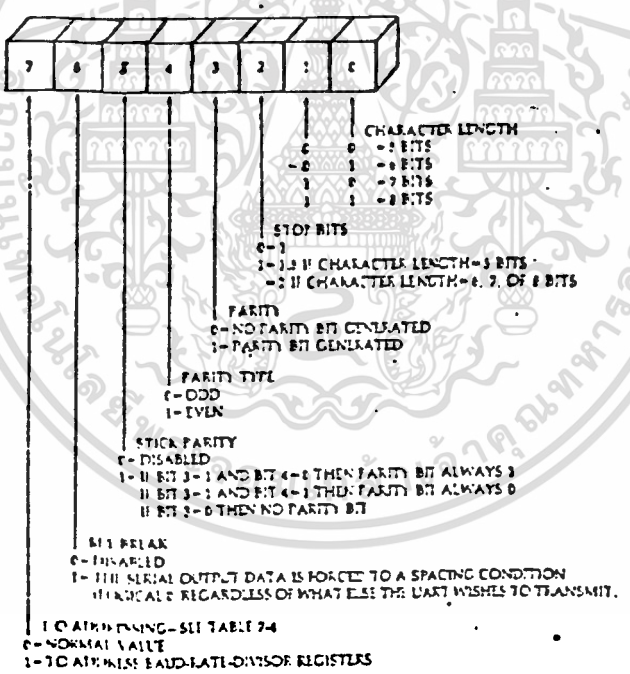
1. Baud rate diviser เป็นพารามิเตอร์ตัวแรกสุดที่ต้องทำการกำหนดค่าเริ่มต้นเพื่อกำหนดอัตราบอดทั้งฝายมาสเตอร์และสเลฟ ในตารางที่ 3.2 ได้สรุปค่าต่าง ๆ ที่ต้องกำหนดของ MSB และ LSB baud rate diviser สำหรับอัตราบอดต่าง ๆ ที่สามารถรับส่งได้ของคาร์ดควบคุมการสื่อสาร

Desired Baud Rate	Value for Baud-Rate-Divisor Registers	
	MSB	LSB
50	09H	00H
75	06H	00H
110	04H	17H
134.5	03H	59H
150	03H	00H
300	01H	80H
600	00H	C0H
1200	00H	60H
1800	00H	40H
2000	00H	3AH
2400	00H	30H
3600	00H	20H
4800	00H	18H
7200	00H	10H
9600	00H	0CH

ตารางที่ 3.2

ในโครงข่าย MCS-Network นี้กำหนดอัตราบอดของการสื่อสารเท่ากับ 9600 บอด วิธีการกำหนดค่าเริ่มต้นให้ baud rate deviser จะต้องกำหนดบิทอันดับสูงของ line control register เป็น "1" (80H) โดย OUT ค่าไปที่ตำแหน่ง 3 FBH (ใน MCS-Network ใช้ COM1) จากนั้นจึงกำหนดค่า LSB และ MSB ที่ถูกต้องให้ baud rate diviser ที่ตำแหน่ง 3F9H ได้ตามลำดับ

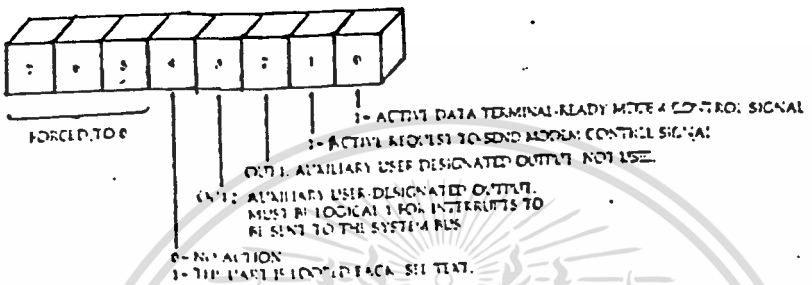
2) Line-control register จากนั้นจึงกำหนดค่าเริ่มต้นแก่ line control register เพื่อกำหนดความยาวของคาร์แรคเตอร์ (character), จำนวนบิทสุดท้าย (stop bit) และชนิดของบิทพาริตี้ รูปที่ 3.6 แสดงการกำหนดค่าเริ่มต้นให้รีจิสเตอร์ตัวนี้สำหรับพารามิเตอร์ในการส่งที่ต้องการ



รูปที่ 3.6 แสดงการกำหนดค่าเริ่มต้นให้แก่ line control register

3) Modem control register โดยปกติบิทอันสูง 3 บิท (high order

three bit) ของรีจิสเตอร์ตัวนี้จะถูกกำหนดให้เป็น "0" ในตอนเริ่มต้นเพื่อไม่ให้ baud rate diviser เปลี่ยนอัตราบอดได้อีก และบิท set break จะกำหนดให้เป็น "1" ก็ต่อเมื่อต้องการให้เกิดสภาวะเบรค (break condition) รูปที่ 3.7 แสดงวิธีกำหนดค่าเริ่มต้นให้รีจีสเตอร์ตัวนี้

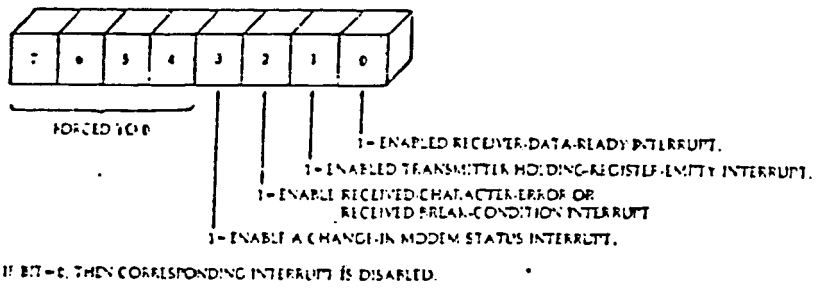


รูปที่ 3.7 แสดงการกำหนดค่าเริ่มต้นให้แก่ modem control register

ในโครงข่าย MCS-Network นี้ไม่ได้ออกแบบมาสำหรับการสื่อสารในระยะทางไกล (long hual communication) จึงไม่มีการใช้ MODEM ดังนั้นจึงกำหนดค่าเริ่มต้น OBH กล่าวคือให้ค่า "1" แก่บิท OUT2 เพื่อยอมให้เกิดการขัดจังหวะโดย 8250 ผ่านไปในบัสของเครื่องไปยัง 8259 interrupt controller

ในขณะที่พัฒนาโครงข่าย MCS-Network ผู้เขียนได้ใช้คุณสมบัติ loop back ของ 8250 โดยกำหนดค่า "1" ให้บิทที่ 4 ของรีจีสเตอร์ตัวนี้

4) Interrupt enable register รูปที่ 3.8 แสดงรีจีสเตอร์ตัวนี้และการกำหนดค่าให้บิทต่าง ๆ



รูปที่ 3.8 แสดงการกำหนดค่าแก่ interrupt enable register

ในทางฝ่ายมาสเตอร์นั้นเราให้ค่า 00H แก่ รีจีสเตอร์ตัวนี้ที่ตำแหน่งที่อยู่ของอุปกรณ์อินพุทเอาท์พุท 3F9H เพราะไม่ต้องการการขัดจังหวะเข้ามาทางพอร์ทสื่อสาร ซึ่งตรงข้ามกับทางฝ่ายสเลฟที่เราให้ค่า 03H เพื่ออนุญาตให้เกิดการขัดจังหวะตามที่เราต้องการ

3.1.8 การสื่อสารด้วย 8250

เมื่อได้กำหนดค่าเริ่มต้นให้ 8250 แล้วก็จะสามารถใช้ 8250 ในการสื่อสารได้ เมื่อใดก็ตามที่ต้องการส่งข้อมูลออกไป ก็จะให้ค่าคาแรคเตอร์นั้น แก่ transmitter holding register ซึ่งถ้าว่าง 8250 ก็จะสามารถรับคาแรคเตอร์จาก reciever data register ได้

line control status register ใช้ในการบอกว่าเมื่อไรจะรับหรือส่งข้อมูล เมื่อต้องการส่งคาแรคเตอร์ออกไปจะต้องอ่านค่า line control register นี้ แล้วตรวจสอบบิทที่ 5 ว่าเป็น "1" หรือไม่ จึงจะสามารถให้ค่า คาแรคเตอร์แก่ transmitter holding register ได้ เมื่อส่งค่าคาแรคเตอร์ ออกไปแล้วบิทที่ 5 นี้จะถูกรีเซทเป็น "0" จนกว่า transimitter holding register พร้อมที่จะรับคาแรคเตอร์ตัวใหม่เข้ามา

เมื่อบิทที่ 0 ของ Line status register เป็น "1" 8250 จะรับคาแรคเตอร์ เข้ามาเก็บไว้ใน receiver data register จากนั้น NOS ของ MCS-Network

จึงจะรับค่าแรคเตอร์นี้ไป ก่อนที่ค่าแรคเตอร์ตัวใหม่จะเข้ามาก่อนการเกิด everrun error ขึ้น บิทศูนย์นี้จะเป็น "0" จนกว่า 8250 จะสามารถรับค่าแรคเตอร์ตัวใหม่ได้

นอกจากนั้นเรายังใช้ Line status register ตรวจสถานะการรับข้อมูลที่ผิดพลาด (received data error) หรือ ตรวจสถานะการรับสภาวะเบรค (received break condition) ได้อีกด้วย

MCS-Network ตรวจจับข้อผิดพลาดโดยการตรวจค่าใน line status register ว่าตรงกับ 1EH หรือไม่

บทที่ 4

การทำงานของโปรแกรม

จุดประสงค์ของการทำงานของโปรแกรมก็คือ ทำให้ USER ใช้งานโปรแกรมที่สร้างขึ้นที่ให้สะดวกและเข้าถึง USER ที่สุด โดยสร้างชุดคำสั่งให้ USER ใช้เอง โดยเปลี่ยนรูปแบบเกี่ยวกับหน้าจอ และการใช้โปรแกรมจากของเดิม ซึ่งของเดิมนั้นใช้ฟังก์ชันคีย์ให้ USER ใช้ ซึ่งไม่สะดวกนัก

ชุดคำสั่ง

Window (X_1,Y_1,X_2,Y_2)

CLEAR - WIN (X_1,Y_1,X_2,Y_2)

KEY - ON ( )

KEY - OFF ( )

CUR - ON ( )

CUR - OFF ( )

STRING (" ")

SET - CUR (X_1, Y_1)

END ( )

RECALL (X_1 ,Y_1, X_2, Y_2)

หลักการทำงาน

MASTER : คือ ทาง Master จะนำคำสั่งที่ USER ใช้ไปแปลงเป็นระหัสคำสั่งแล้วทำการส่งไปยังเครื่อ SLAVE

SLAVE : SLAVE จะทำการรับคำสั่งที่แปลงเป็นรหัสแล้วจาก Master แล้ว จากนั้นก็ไปทำงานตามคำสั่งที่ USER ใช้

การใช้โปรแกรม

เมื่อ USER เรียกโปรแกรม จะมี main menu ขึ้นมาโดยให้เลือกเป็น mode FILE, EDIT, RUN ในแถบบน และ mode THAI/ENLISH ในแถบล่างดังรูปผลการทดลองที่ 5.1

จากนั้นก็ เลือก mode ต่าง ๆ ถ้า FILE ก็จะปรากฏผลดังรูปที่ 5.2

สำหรับ mode EDIT ก็จะเป็นการแก้ไขโปรแกรมใน FILE

ส่วน RUN ก็จะแปลงคำสั่งแล้วส่งไปให้ SLAVE รับคำสั่งและทำงานตามคำสั่ง

การทำงานมีอยู่ 2 ลักษณะคือ

- รับข้อมูลที่ผู้ใช้ที่เป็นผู้สอน คีย์ ขึ้นขณะนั้นโดยไม่ได้เตรียมไว้ทำในรูปของกลุ่มคำสั่งที่เตรียมไว้ แล้ว RUN ที่ละคำสั่ง (แปลงเป็นระหัส) ส่งไปยัง SLAVE เพื่อให้ SLAVE ทำงานตามคำสั่ง
- ข้อมูลที่ Master จะส่งไปยัง SLAVE นั้นได้มีการเตรียมเน้น script ไว้ก่อนแล้ว ซึ่งก็เริ่มโดยเรียก script แล้วทำการ RUN ไปให้ SLAVE ทำงานตามคำสั่ง

การทำงาน 2 ลักษณะดังกล่าว แบบแรกที่เตรียมไว้ ใน mode FILE คือ write to ซึ่งใช้สำหรับแบบแรก ส่วน mode FILE ที่เป็น LOAD เตรียมไว้ใช้กับ script ซึ่งเป็นแบบที่ 2

ส่วน EDIT นั้นใช้ในการแก้ไขข้อมูล จากนั้นก็มีการ SAVE 1 FILE ต่าง ๆ ไว้

ส่วน OSSHELL ก็จะกลับไป DOS ชั่วคราว เมื่อพิมพ์ exit จะกลับมาสู่โปรแกรมของเรา

ส่วน TODOS ก็จะกลับไป DOS เลย

ส่วนความหมายของคำสั่งต่าง ๆ

Window ( ) ก็จะเป็นคำสั่ง สำหรับสร้าง Window

มีการกำหนดจุด coordinate ของ X_1, Y_1, X_2, Y_2 ซึ่งก็คือ COORDINATE ของกรอบหน้าต่าง มุมบนซ้าย (X_1, Y_1) และมุมบนขวา (X_2, Y_2)

CLEAR-WIN ( ) เป็นคำสั่งที่มีการกำหนด COORDINATE (X_1, Y_1, X_2, Y_2) ซึ่งเป็นการ clear window ในขอบเขตของ COORDINATE ดังกล่าว

KEY - ONE ( ) และ DYE - OFF ( ) เป็นการตัด KEY BOARD และต่อ KEY BOARD ของตัว SLAVE ก็เพื่อให้ตัวลูกใช้ DEYBOARD ไม่ได้ขณะที่ตัว master กำลังทำการสอน และเมื่อตัวแม่ (MASTER) เลิกทำการสอนก็ต่อ KEYBOARD คืนให้กับตัวลูก (SLAVE)

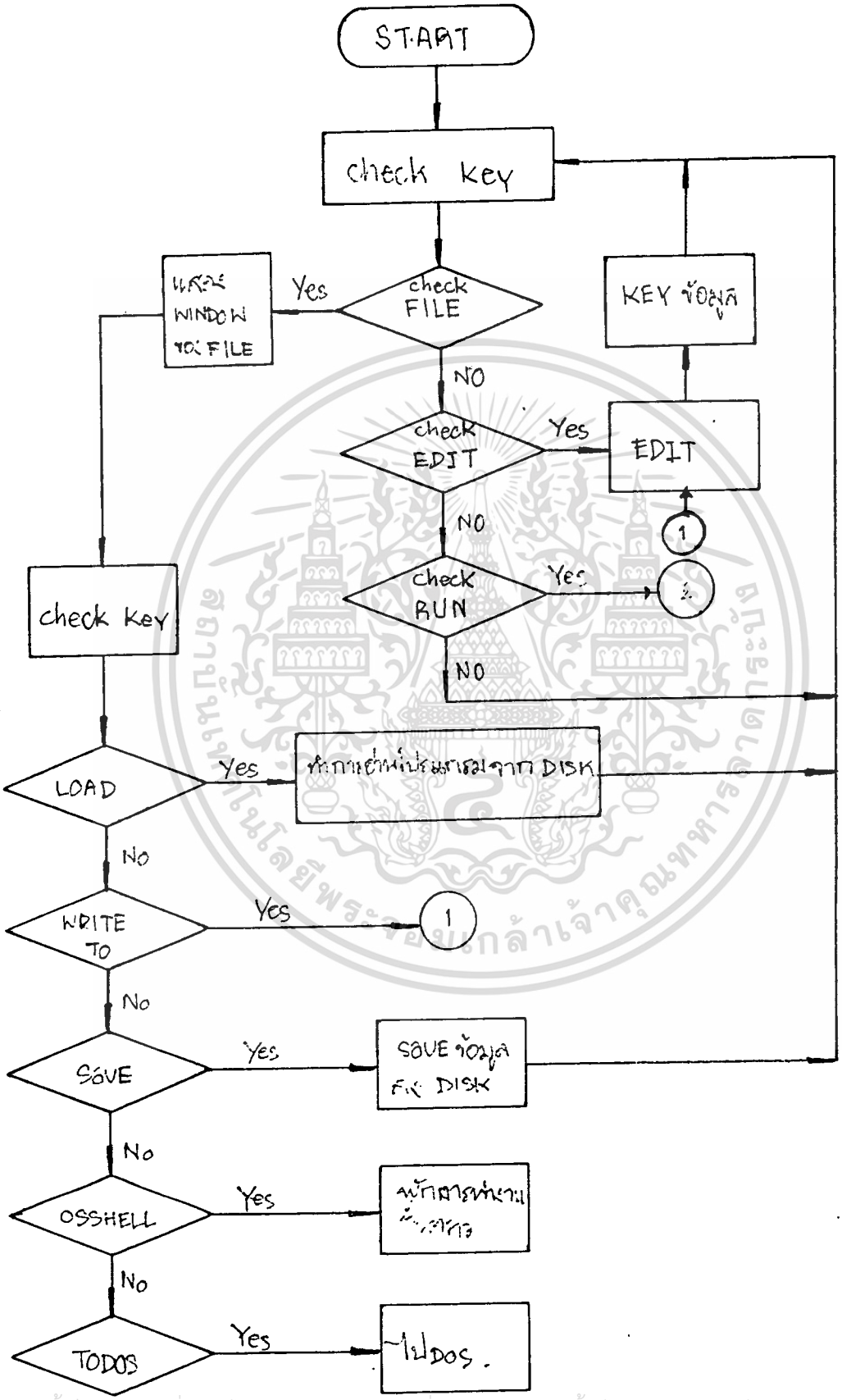
END ( ) จะใส่เมื่อจบโปรแกรมในชุดคำสั่งของเรา

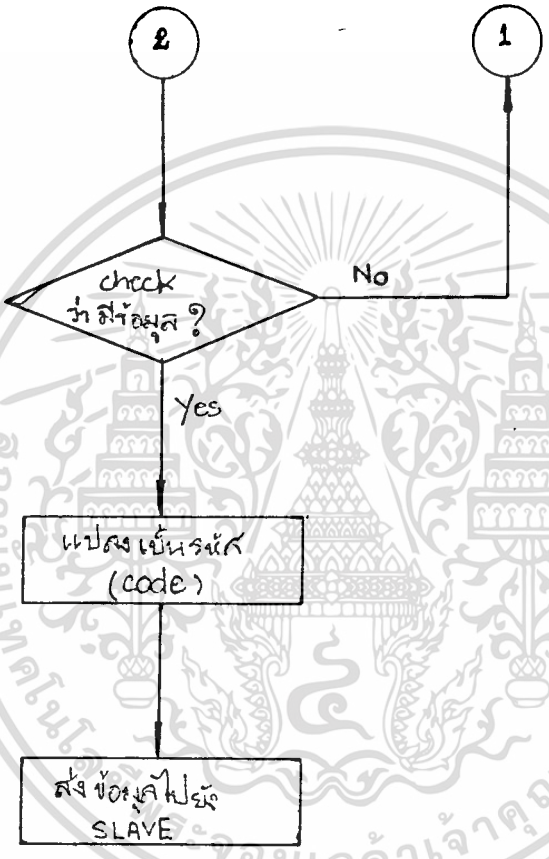
RECALL หมายถึง การเรียกข้อมูลที่ถูก window ตัวอื่นทับซ้อนอยู่ให้ปรากฏออกมา โดยมี COORDINATE X_1, Y_1, X_2, Y_2 เป็นตัวกำหนด

CUR - OFF ( ) และ CUR - ON ( ) และ SET - CUR (X_1, Y_1)

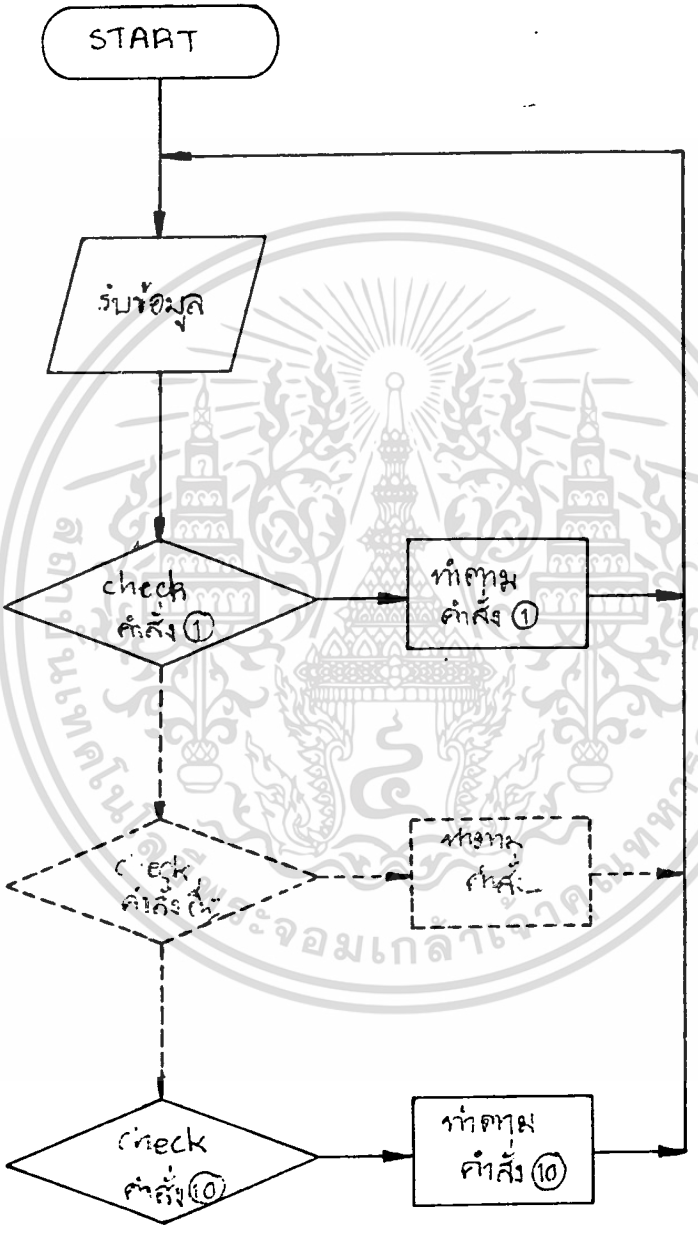
CUR - OFF จะหมายถึงการ OFF CURSOR คือเราไม่สนใจตำแหน่ง CURSOR ที่ปรากฏบนจอ ถ้า CUR-ON จะหมายถึง เราจะสนใจตำแหน่งของ CURSOR และ SET- CURSOR เมื่อเราสนใจในตำแหน่ง CURSOR คำสั่งนี้ก็สามารถเลื่อน CURSOR ไปตามตำแหน่งที่ต้องการ ซึ่งกำหนดโดย COORDINATE X_1, Y_1

MASTER





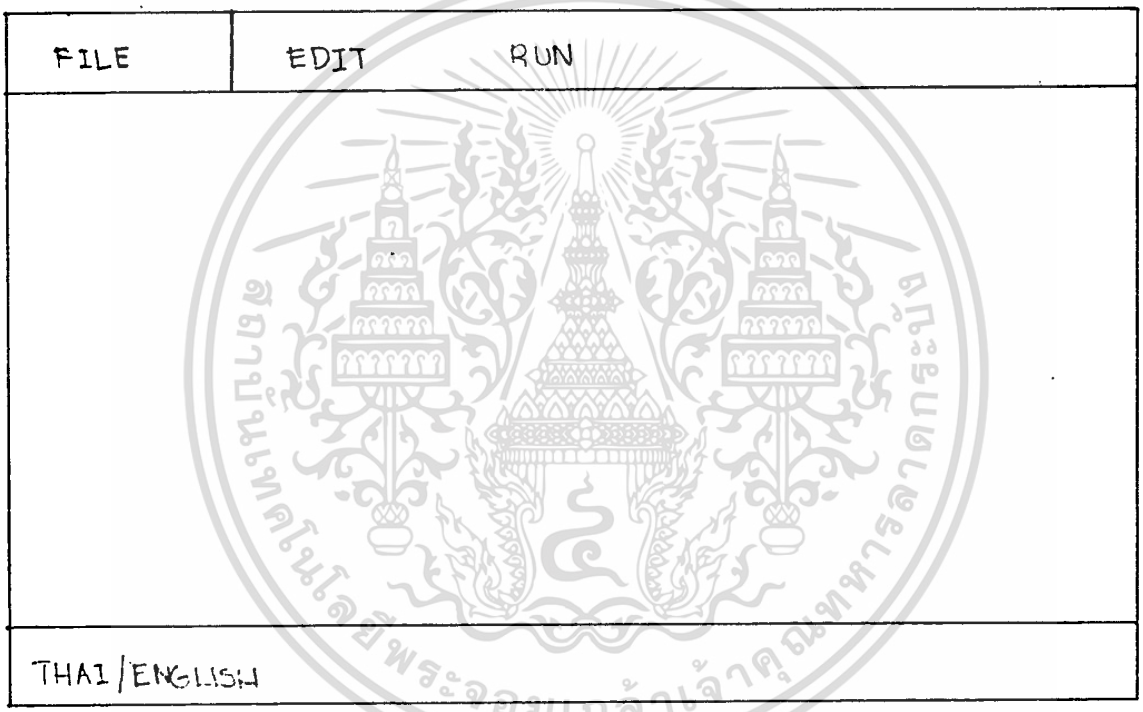
SLAVE



บทที่ 5

ผลการทดลอง

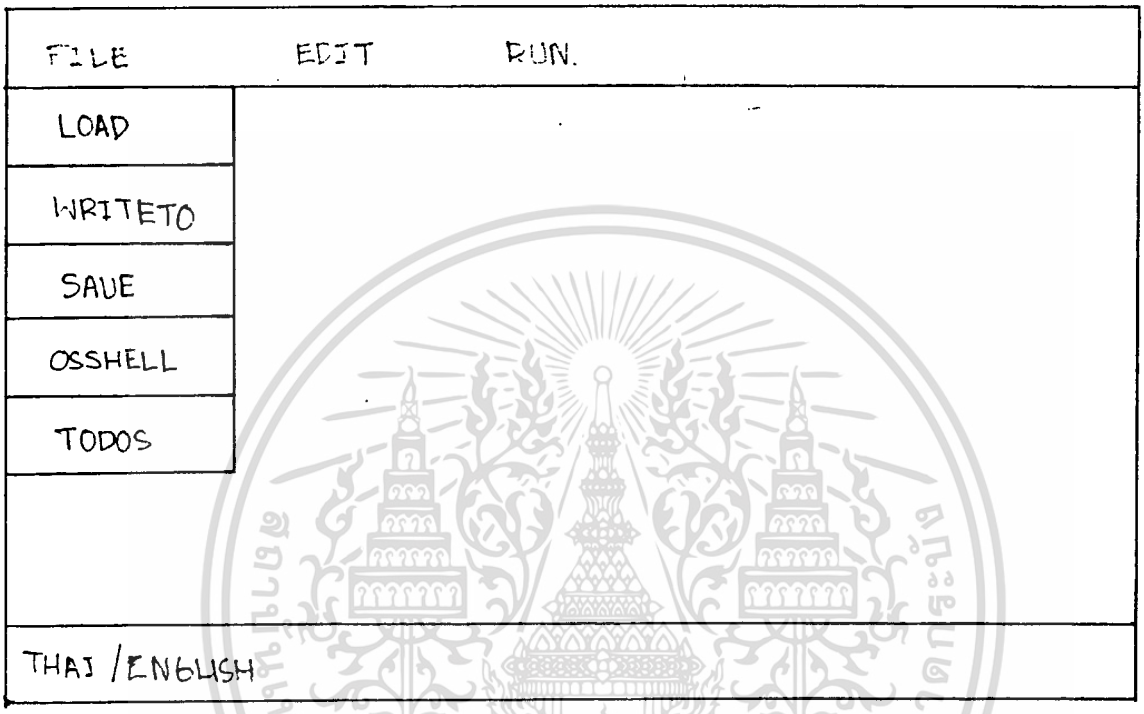
เมื่อโปรแกรม MASTER ถูกเรียก



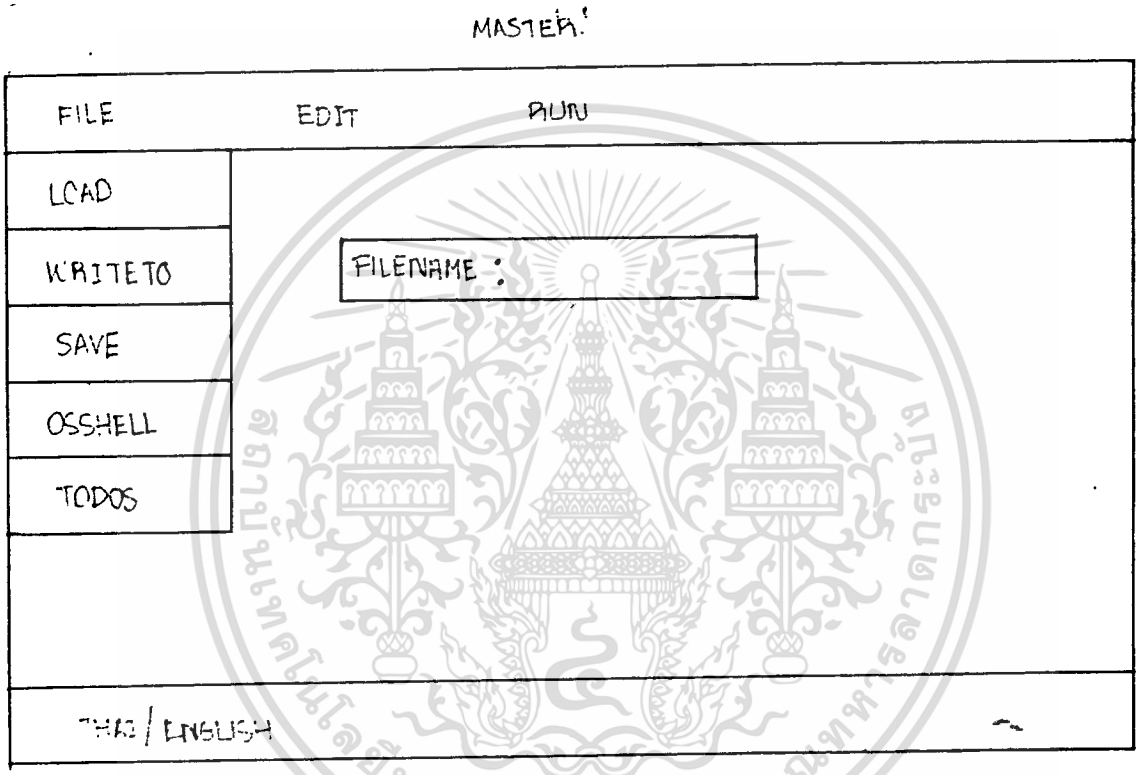
รูปที่ 5.1

กดคีย์ ลูกศร เลือก mode

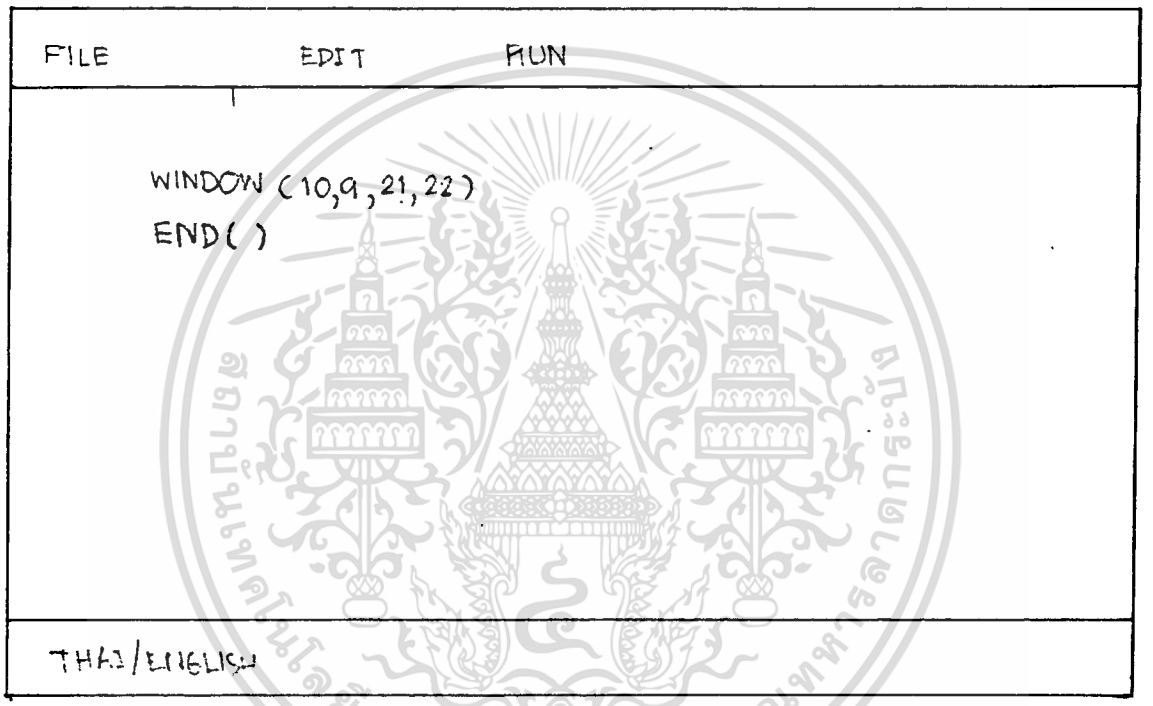
และเลือก ENGLISH/THAI



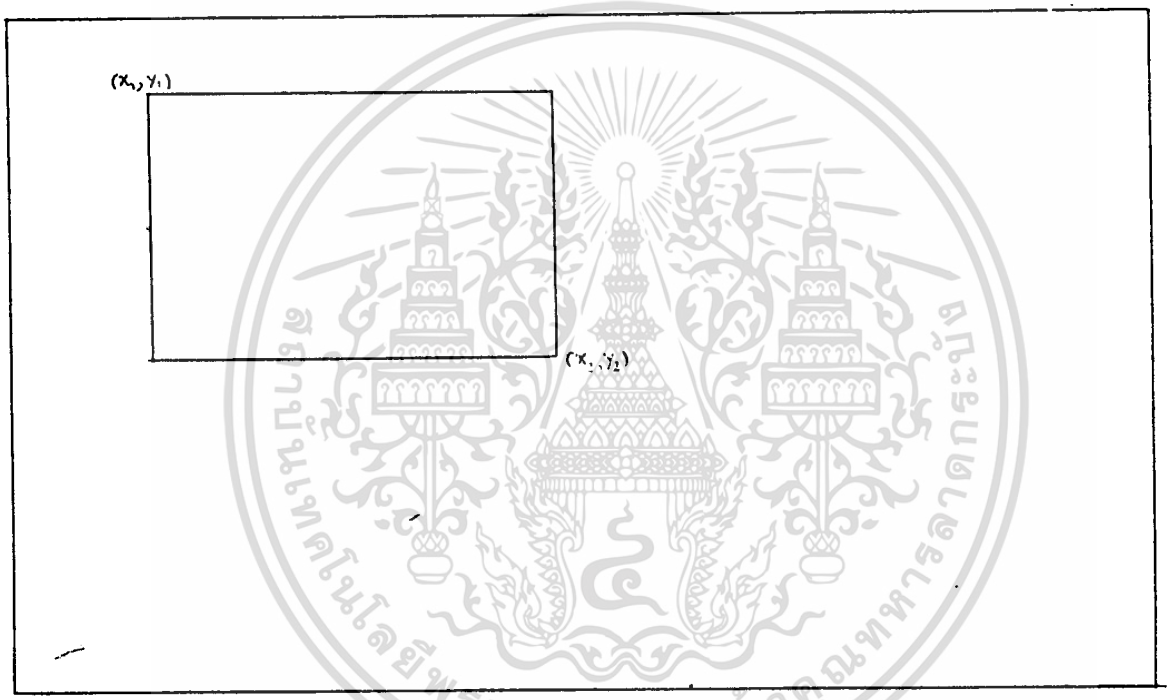
เลือก FILE รูป 5.2



รูป 5.3 เรียก LOAD ใส่ชื่อ FILE



รูปที่ 5.4 หลังจากใส่ชื่อ FILE ก็จะปรากฏตัวโปรแกรม
พอกด RUN ก็จะไปปรากฏผลที่ SLAVE



รูปที่ 5.5 รูปผลที่แสดงออกมาจากรูป 5.4

สำหรับส่วน WRITETO ก็จะปรากฏจากรูป 5.2 มาเป็นจอรูป 5.1 จากนั้นก็คีย์คำสั่งที่ USER ใช้ ดังรูปที่ 5.4 ผลก็จะออกมาในรูป 5.5

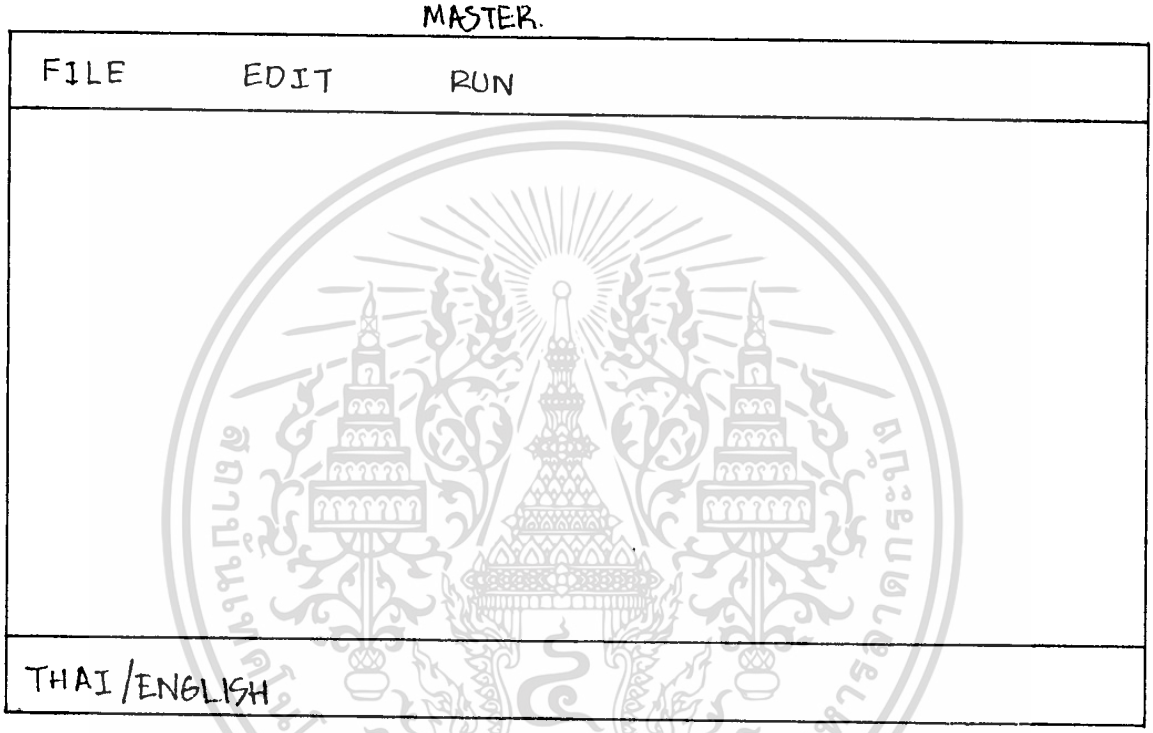
ส่วนคำสั่ง SAVE, EDIT เอาไว้เพื่อ SAVE หรือแก้ไขโปรแกรมที่ load ขึ้นมา

ส่วน OSSHELL ก็จะปรากฏ A> บนจอภาพ

รูปที่ 5.6

ถ้าคีย์คำว่า exit ก็จะกลับมาที่โปรแกรมอย่างเดิม

รูปที่ 5.7



รูปที่ 5.8 กลับมาที่ส่วนเดิม

ส่วน TODOS นั้นก็กลับไปที่ DOS เลย แล้วปรากฏดังรูป 5.6

บทที่ 6

สรุป วิจารณ์ และแนวทางในการพัฒนาต่อ

6.1 สรุปและวิจารณ์

6.1.1 รูปแบบที่เริ่มพัฒนาต่างไปจากรูปเดิมนั้น ยังขาด function ต่าง ๆ หลาย ๆ อย่าง ทั้งบางจุดยังมีข้อบกพร่องอยู่ทั้งลักษณะการใช้งานยังไม่สะดวกนัก ยังเกิดความล่าช้าระหว่างการติดต่อในขณะส่งข้อมูล

6.1.2 ซอฟแวร์ ที่ใช้เขียนขึ้นโดยใช้ภาษาแอสแซมบลี้ 8088/8086 (ASSEMBLY 8088/8086) ลักษณะการทำงานแบ่งเป็น -ทางฝ่ายมาสเตอร์ ซึ่งมีลักษณะเป็นโปรแกรมที่มีการจัดจังหวะด้วยการส่งคำสั่ง Key-on โดยผู้ใช้ทางด้านมาสเตอร์จะอยู่ในลักษณะที่ใช้คำสั่งอื่นทำการสอนและส่งข้อมูลมาทางฝ่ายสเลฟ จนกว่าจะ Key-off ให้ฝ่ายสเลฟใช้งานปกติ - ทางฝ่าย slave จะเป็นโปรแกรม ที่นำคำสั่งจากฝ่ายมาสเตอร์ที่แปลงแล้วมาทำงานตามคำสั่ง

6.1.3 โครงข่าย MCS-Network เป็นโครงข่ายที่มีประสิทธิภาพสูงโดยเฉพาะอย่างยิ่งเมื่อเทียบกับค่าใช้จ่ายในการจัดตั้งโครงข่าย การขยายโครงข่ายทำได้ไม่จำกัด นอกจากนี้ยังสามารถนำไปใช้กับงานอื่นได้ดังตัวอย่าง เช่น

- การควบคุมการเรียนการสอนในชั้นเรียน
- การควบคุมการสาธิตในงานนิทรรศการผ่านทางจอมอนิเตอร์ของไมโครคอมพิวเตอร์หลาย ๆ เครื่องพร้อมกัน
- การสั่งงานภายในสำนักงานโดยการใช้การทำงานในลักษณะเทอร์มินอล เป็นต้น

6.2 แนวทางการพัฒนาต่อ

6.2.1 ควรมีการเพิ่มและแก้ไขฟังก์ชันให้ดีขึ้น อย่างเช่นส่วน write to นั้นต้องใช้วิธี USER ทาง MASTER คีย์คำสั่งขณะนั้น ซึ่งมีผลทำให้เกิดความล่าช้าในกรณีที่โปรแกรมที่ USER ทาง MASTER ที่เขียนยาว ทำให้ SLAVE มีช่วงเวลาว่างรออยู่ นอกจากนั้น การเพิ่มฟังก์ชันเข้าไป เช่น ฟังก์ชันที่เกี่ยวกับทาง graphic

6.2.2 การพัฒนาการเรียนการสอนในลักษณะการโต้ตอบคำถาม เมื่อ MASTER ทำการสอนเสร็จ ซึ่งการตอบคำถามอาจมีรูปแบบ เช่น ทางเครื่องไมโครคอมพิวเตอร์ตัวแม่ (MASTER COMPUTER) สามารถเลือกตอบว่าจะตอบตัวไหน หรือจะตอบพร้อมกันหมดเลย พร้อมทั้งสามารถตรวจสอบได้ว่า เครื่องไมโครคอมพิวเตอร์ตัวลูก (SLAVE COMPUTER) ตัวใดเรียกหรือสงสัยในการสอน ซึ่งระหว่างที่มีคำถามนั้นอาจมีไมโครคอมพิวเตอร์ตัวลูก เรียกเข้ามาหลายตัว ซึ่งไม่สามารถตอบคำถามได้พร้อมกัน ดังนั้นทำเป็นสัญญาณที่เรียกจากตัวลูกเข้ามาให้แสดงสัญญาณที่เรียกมาบนหน้าจอของเครื่องไมโครคอมพิวเตอร์ตัวแม่แล้วจึงเลือกตอบกลับไป เป็นต้น

6.2.3 การพัฒนาต่อควรจะเน้นความรวดเร็ว และความสะดวกแก่ผู้ใช้ (USER)

6.2.4 ควรขยายโครงข่ายหรือคิดค้นแนวทางเพื่อทดสอบโครงข่ายได้ ซึ่งอาจทำให้ต่อกับตัวลูกจำนวนมาก ๆ ซึ่งเพื่อไว้ก่อน และพยายามตรวจสอบข้อบกพร่องหรือผิดพลาดในระบบใกล้เคียงจริง

ภาคผนวก

ในส่วนภาคผนวกประกอบด้วยรายละเอียดดังต่อไปนี้

ภาคผนวก ก แสดงซอสลิสติ้ง (SOURCE LISTING) ของโปรแกรมของคุมทางด้านมาสเตอร์

ภาคผนวก ข แสดงซอสลิสติ้ง (SOURCE LISTING) ของโปรแกรมของคุมทางด้านสเลฟ

ภาคผนวก ก แสดงซอสลิสติ้ง (SOURCE LISTING) ของโปรแกรมของคุมทางด้านมาสเตอร์

```
A>type menu.prj
menu.c
g.c
exec.c (twindow.h, keys.h)
load.c
notepad (twindow.h)
editor (twindow.h, keys.h)
entry (twindow.h, keys.h)
thelp (twindow.h, keys.h)
tmenu (twindow.h)
twindow (twindow.h, keys.h)
ibmpc
```

```
}
```

```
A>type keys.h
/*  ----------keys.h--------- */
#define HT          9
#define RUBOUT      8
#define BELL        7
#define ESC        27
#define SHIFT_HT 143
#define CTRL_T    20
#define CTRL_B     2
#define CTRL_D     4
#define ALT_D    160
#define F1       187
#define F2       188
#define F3       189
#define F4       190
#define F5       191
#define F6       192
#define F7       193
#define F8       194
#define F9       195
#define F10      196

#define HOME     199
#define UP       200
#define PGUP     201
#define BS       203
#define FWD      205
#define END      207
#define DN       208
#define PGDN     209
#define INS      210
#define DEL      211

#define CTRL_HOME    247
#define CTRL_BS      243
#define CTRL_FWD     244
#define CTRL_END     245
```

A>type twindow.h

```
/* ------------------twindow.h---------- */
/*       Uncomment this for stacked windows
 *       rather than layered windows.
 *
 *        #define FASTWINDOWS
 *
 */
/* ---------window colors-------------- */
#define RED     4
#define GREEN   2
#define BLUE    1
#define WHITE   (RED+GREEN+BLUE)
#define YELLOW  (RED+GREEN)
#define AQUA    (GREEN+BLUE)
#define MAGENTA (RED+BLUE)
#define BLACK   0
#define BRIGHT  8
#define DIM     0

#define BORDER  0
#define TITLE   1
#define ACCENT  2
#define NORMAL  3
#define ALL     4
#define TRUE    1
#define FALSE   0
#define ERROR   -1
#define OK      0
/* -------------WINDOW CONTROLLER ------------------------ */
typedef struct field{
    char *fmask;
    int  fprot;
    char *fbuff;
    int  ftype;
    int  frow;
    int  fcol;
    void (*fhelp)();
    char *fhwin;
    int  flx,fly;
    int  (*fvalid)();
    struct field * fnxt;
    struct field * fprv;
} FIELD;
typedef struct _wnd {
    int _wv;
    int _hd;
    char *_ws;
    char *_tl;
    int _wx;
    int _wy;
    int _ww;
    int _wh;
    int _wsp;
    int _sp;
    int _cr;
    int btype;
    int wcolor[4];
    int _pn;
    struct _wnd * nx;
```

```
    struct _wnd *_pv;
    FIELD *_fh;
    FIELD *_ft;
} WINDOW;
typedef struct w_menu{
    char *mname;
    char **mselcs;
    void (**func)();
} MENU;
#define SAV       (wnd->_ws)
#define WTITLE    (wnd->_tl)
#define COL       (wnd->_wx)
#define ROW       (wnd->_wy)
#define WIDTH     (wnd->_ww)
#define HEIGHT    (wnd->_wh)
#define SCROLL    (wnd->_wsp)
#define SELECT    (wnd->_sp)
#define WCURS     (wnd->_cr)
#define WBORDER   (wnd->wcolor[BORDER])
#define WTITLEC   (wnd->wcolor[TITLE])
#define WACCENT   (wnd->wcolor[ACCENT])
#define WNORMAL   (wnd->wcolor[NORMAL])
#define PNORMAL   (wnd->_pn)
#define BTYPE     (wnd->btype)
#define NEXT      (wnd->_nx)
#define PREV      (wnd->_pv)
#define WCOLOR    (wnd->wcolor)
#define VISIBLE   (wnd->_wv)
#define HIDDEN    (wnd->_hd)
#define FHEAD     (wnd->_fh)
#define FTAIL     (wnd->_ft)

#define NW        (wcs[wnd->btype].nw)
#define NE        (wcs[wnd->btype].ne)
#define SE        (wcs[wnd->btype].se)
#define SW        (wcs[wnd->btype].sw)
#define SIDE      (wcs[wnd->btype].side)
#define LINE      (wcs[wnd->btype].line)
/* ----------FUNTION PROTOTYPES------------------ */

/* ----- general-purpose functions and macros ----- */

void clear_screen(void);
int vmode(void);
void cursor(int, int);
void curr_cursor(int * ,int * );
int cursor_type(void);
void set_cursor_type(int);
int get_char(void);
int scroll_lock(void);
void vpoke(unsigned, unsigned, unsigned);
int vpeek(unsigned, unsigned);

/* ----------window funtion----------------- */
WINDOW *establish_window(int, int, int, int );
void set_border(WINDOW * ,int);
void set_colors(WINDOW * ,int, int, int, int);
void set_intensity(WINDOW * ,int);
void set_title(WINDOW * ,char *);
void display_window(WINDOW * );
void delete_window(WINDOW * ) ;
void clear_window (WINDOW * );
void hide_window(WINDOW * );
void wprintf(WINDOW * ,char *, ...);
void wputchar(WINDOW * ,int);
void close_all(void);
```

```
void wcursor(WINDOW * ,int x, int y);
void error_message(char *);
void clear_message(void);
int get_selection(WINDOW * , int, char *);

#define reverse_video(wnd)      wnd->wcolor[3]=wnd->wcolor[2]
#define normal_video(wnd)       wnd->wcolor[3]=wnd->_pn
#define rmove_window(wnd,x,y)   repos_wnd(wnd,x,y,0)
#define move_window(wnd,x,y)    repos_wnd(wnd,COL-x, ROW-y,0)
#define forefront(wnd)          repos_wnd(wnd,0,0,1)
#define rear_window(wnd)        repos_wnd(wnd,0,0,-1)

/* -------------internal to window processas---------------- */
void accent(WINDOW * );
void deaccent(WINDOW * );
void scroll(WINDOW * , int);
void repos_window(WINDOW * , int, int, int);
void acline(WINDOW * , int );
#define accent(wnd)     acline(wnd, WACCENT)
#define deaccent(wnd)   acline(wnd, WNORMAL)
#define clr(bg, fg,in)  ((fg)|(bg<<4)|(in))
#define vad(x,y)        ((y)* 160+(x)* 2)
#ifdef FASTWINDOWS
#define cht(ch,at)      (((ch)&255)|((at)<<8))
#define displ(w,x,y,c,a) vpoke(VSG,vad(x+COL,y+ROW),cht(c,a))
#define dget(w,x,y)     vpeek(VSG,vad(x+COL,y+ROW)
#define verify_wnd(w)   (*(w)=listtail)!=0
#else
void displ(WINDOW *wnd, int x, int y, int ch, int at);
#endif

/* --------editor----------------- */
void text_editor(WINDOW * ,char * , unsigned);

/* ----------menu--------------- */
void menu_select(char *name, MENU *sn);

/* ----------help---------------- */
void load_help(char *);
void set_help(char *, int, int);

/* ----------data entry---------- */
void init_template(WINDOW * );
FIELD *establish_field(WINDOW * ,int,int,char * , char * ,int);
void clear_template(WINDOW * );
void field_tally(WINDOW * );
int data_entry(WINDOW * );
void wprompt(WINDOW * , int, int, char * );
void error_message(char * );
void clear_notice(void);
void field_window(FIELD * , char * , int, int);
#define field_protect(f,s)      f->fprot=s
#define field_help(f,h)         f->fhelp=h
#define field_validate(f,v)     f->fvalid=v
```

```
A>type menu.c
/* ---------- menu.c ---------- */

#include "twindow.h"
void exec(void);

char notefile [] = "note.pad";

main()
{
    load_help("tcprogs.hlp");
    exec();
}
```

```
A>type g.c
char *command(void);
void  osshell(void);
void  quit(void);

char send[30] = "\x1b[";        /* string to send       */

char *command()
{
        char    *cmd, key[20];
        int     i, j, t1,t2,t3,t4;

        cmd = (char *) malloc(80);
        printf("Enter command: "); gets(cmd); strlwr(cmd);

        for(i=0; *cmd != '(';i++) {
                key[i] = *cmd++;
        }
        /* detect each keyword in <cmd> to <key>        */
        key[i] = '\0';
        if( strcmp(key,"window") == 0 ){
                sscanf(cmd,"(%d,%d,%d,%d)",&t1,&t2,&t3,&t4);
                strcat(send,"00");
                j = strlen(send);
                send[j++] = (char)t1;
                send[j++] = (char)t2;
                send[j++] = (char)t3;
                send[j++] = (char)t4;
                send[j] = '\0';
        } else if( strcmp(key,"clear_win") == 0 ){
                sscanf(cmd,"(%d,%d,%d,%d)",&t1,&t2,&t3,&t4);
                strcat(send,"01");
                j = strlen(send);
                send[j++] = (char)t1;
                send[j++] = (char)t2;
                send[j++] = (char)t3;
                send[j++] = (char)t4;
                send[j] = '\0';
        } else if( strcmp(key,"key_on") == 0 ){
                strcat(send,"02");
        } else if( strcmp(key,"key_off") == 0 ){
                strcat(send,"03");
        } else if( strcmp(key,"cur_on") == 0 ){
                strcat(send,"04");
        } else if( strcmp(key,"cur_off") == 0 ){
                strcat(send,"05");
        } else if( strcmp(key,"print_string") == 0 ){
                strcat(send,"06");
                for(i=0, cmd += 2, j=strlen(send); cmd[i] != '"'; i++,j++) {
                        send[j] = cmd[i];
                }
                send[j] = '\0';
        } else if( strcmp(key,"set_cur") == 0 ){
                sscanf(cmd,"(%d,%d)",&t1,&t2);
                strcat(send,"07");
                j = strlen(send);
                send[j++] = (char)t1;
                send[j++] = (char)t2;
                send[j] = '\0';
        } else if( strcmp(key,"end") == 0 ){
                strcat(send,"08");
```

```
                strcat(send, vb );
        } else {
                printf("Command error: %s", key);
                *send = '\0';
        }
        return send;
}

void quit( void )
{
        exit(0);
}

void osshell( void )
{
        system("");
}
```

```
A>type exec.c
/* ---------- exec.c -------- */
#include <stdio.h>
#include "twindow.h"
/* -------- local prototypes -------- */
void load(void);
void write(void);
void save(void);
void osshell(void);
void quit(void);
void notepad(void);
/* ---------- menu tables ---------- */
char *dselcs[] = {
    " load ",
    " write ",
    " save   ",
    " osshell ",
    " quit ",
    NULL
};
char *pselcs[] = {
    " notepad ",
          NULL
};
static void (*dfuncs[])()={load,write,save,osshell,quit };
static void (*pfuncs[])()={notepad};
static MENU tmn [] = {
    {" FILE ",    dselcs, dfuncs},
    {" EDIT ",  pselcs, pfuncs},
    {" RUN ",NULL,NULL},
    {NULL,NULL,NULL}
};

void exec()
{
    clrscr();
    menu_select(" MCS NETWORK ", tmn);
}
```

```
A>type load.c
/*---------------loads.c--------------------*/

#include <stdio.h>
#include <conio.h>

char *command(void);
void  osshell(void);
void  quit(void);

char send[30] = "\x1b[";        /* string to send       */

char jie;
FILE *f;
char line[300][100];
int i;

main()
{
        i = 0;
        f = fopen("a:jie.c", "r" );
        do
        {
                fscanf(f, "%s", line[i]);
                printf("%s\n",line[i]);
                i++;
        } while (! feof(f));
        fclose(f);
}

save(){}

char *command()
{
        char    *cmd, key[20];
        int     i, j, t1,t2,t3,t4;

        cmd = (char *) malloc(80);
        printf("Enter command: "); gets(cmd); strlwr(cmd);

        for(i=0; *cmd != '(';i++) {
                key[i] = *cmd++;
        }
        /* detect each keyword in <cmd> to <key>        */
        key[i] = '\0';
        if( strcmp(key,"window") == 0 ){
                sscanf(cmd,"(%d,%d,%d,%d)",&t1,&t2,&t3,&t4);
                strcat(send,"00");
                j = strlen(send);
                send[j++] = (char)t1;
                send[j++] = (char)t2;
                send[j++] = (char)t3;
                send[j++] = (char)t4;
                send[j] = '\0';
        } else if( strcmp(key,"clear_win") == 0 ){
                sscanf(cmd,"(%d,%d,%d,%d)",&t1,&t2,&t3,&t4);
                strcat(send,"01");
                j = strlen(send);
                send[j++] = (char)t1;
                send[j++] = (char)t2;
                send[j++] = (char)t3;
```

```
                send[j++] = (char)t4;
                send[j] = '\0';
        } else if( strcmp(key,"key_on") == 0 ){
                strcat(send,"02");
        } else if( strcmp(key,"key_off") == 0 ){
                strcat(send,"03");
        } else if( strcmp(key,"cur_on") == 0 ){
                strcat(send,"04");
        } else if( strcmp(key,"cur_off") == 0 ){
                strcat(send,"05");
        } else if( strcmp(key,"print_string") == 0 ){
                strcat(send,"06");
                for(i=0, cmd += 2, j=strlen(send); cmd[i] != '"'; i++,j++) {
                        send[j] = cmd[i];
                }
                send[j] = '\0';
        } else if( strcmp(key,"set_cur") == 0 ){
                sscanf(cmd,"(%d,%d)",&t1,&t2);
                strcat(send,"07");
                j = strlen(send);
                send[j++] = (char)t1;
                send[j++] = (char)t2;
                send[j] = '\0';
        } else if( strcmp(key,"end") == 0 ){
                strcat(send,"08");
        } else {
                printf("Command error: %s", key);
                *send = '\0';
        }
        return send;
}

void quit( void )
{
        exit(0);
}

void osshell( void )
{
        system("");
}
```

```
A>type notepad.c
/*----------------notepad.c---------------*/
#include <stdio.h>
#include <mem.h>
#include "twindow.h"

#define LWID 80
#define WHT  50
#define PADHT  20

char bf [PADHT] [LWID];
extern  char notefile[];

void notepad()
{
        WINDOW *wnd;
        FILE *fp, *fopen();
        int i, ctr = 0;

        set_help("notepad",0,0);
        setmem(bf, sizeof bf, ' ');
        if (( fp = fopen(notefile, "rt")) != NULL)  {
            while (fread(bf [ctr], LWID, 1, fp))
                ctr++;
            fclose(fp);
}
        wnd = establish_window
            ((80-(LWID+2))/2, (25-(WHT+2))/2, WHT+2, LWID+2);
        set_border(wnd, 3);
        set_title(wnd, "  Note pad  ");
        set_colors(wnd, ALL, BLUE, AQUA, BRIGHT);
        set_colors(wnd, ACCENT, WHITE, BLACK, DIM);
        display_window(wnd);
        text_editor(wnd, bf[0], (unsigned) LWID * PADHT);
        delete_window(wnd);
        ctr = PADHT;
        while (--ctr)  {
             for ( i = 0; i < LWID; i++)
                 if (bf [ctr] [i] != ' ')
                 break;
             if (i < LWID)
                 break;
        }
        fp = fopen(notefile, "w");
        for ( i = 0; i < ctr+1; i++)
              fwrite(bf[i], LWID, 1, fp);
              fclose(fp);
}
```

type editor.c

```
/*--------------------------- editor.c ---------------------------*/
#include <stdio.h>
#include <ctype.h>
#include <mem.h>
#include <conio.h>
#include <alloc.h>
#include "twindow.h"
#include "keys.h"

#define TRUE 1
#define FALSE 0
#define TAB 4
#define NEXTTAB (TAB-(x%TAB))
#define LASTTAB (((wwd-1)/TAB)*TAB)
#define PREVTAB (((x-1)%TAB)+1)
#define curr(x,y) (bfptr+(y)*wwd+(x))
#define lineno(y) ((int)(bfptr-topptr)/wwd+(y))

extern int VSG;
int last_x, last_y;
static int wht;
static int wwd;
static int wsz;
static char *topptr;
static char *bfptr;
static char *lstptr;
static int lines;
static char *endptr;
static int blkbeg;
static int blkend;
static int inserting;
static WINDOW *wnd;
static int do_display_text = 1;

/*----------------local function prototypes-------------------*/
void erase_buffer(int *x, int *y);
int lastword(int x, int y);
void last_char(int *x, int *y);
void test_para(int x, int y);
int trailing_spaces(int y);
int first_wordlen(int y);
void paraform(int x, int y);
int blankline(int line);
void delete_word(int x, int y);
void delete_line(int y);
void delete_block(void);
void copy_block(int y);
void move_block(int y);
mvblock(int y, int moving);
void findlast(void);
void find_end(int *x, int *y);
void carrtn(int *x, int *y, int insert);
void backspace(int *x, int *y);
void fore_word(int *x, int *y, char *bf);
int spaceup(int *x,int *y, char **bf);
void back_word(int *x, int *y, char *bf);
int spacedn(int *x, int *y, char **bf);
void forward(int *x, int *y);
int downward(int *y);
void upward(int *y);
```

```
void upward(int *y);
void display_text(void);
void disp_line(int y);
void insert_line(void);

/*-------------- Process text entry for a window. ---------------*/
void text_editor(WINDOW *wndl, char *bf, unsigned bsize)
{
      char *b, *buff;
      int depart = FALSE, i, c;
      int x, y, svx, svlw, tx, tabctr = 0;

      wnd = wndl;
      wht = HEIGHT-2;
      wwd = WIDTH-2;
      wsz = wwd * wht;
      topptr = bfptr = bf;
      lines = bsize / wwd;
      endptr = bf + wwd * lines;
      blkbeg = 0;
      blkend = 0;
      inserting = FALSE;
      x = 0;
      y = 0;
      display_text();
      /*------------ read in text from the keyboard ------------*/
      findlast();
      while (TRUE)    {
      last_x = COL + 1 + x;
      last_y = ROW + 1 + y;
      cursor(last_x, last_y);
      buff = curr(x, y);
      if (tabctr)     {
         --tabctr;
         c = ' ';
      }
      else     {
         c = get_char();
         clear_message();
      }
      switch (c)   {
         case '\r': carrtn(&x, &y, inserting);
                    break;
         case DN:   downward(&y);
                    break;
         case PGUP: y = 0;
                    for (i = 0; i < wht; i++)
                       upward(&y);
                    break;
         case PGDN: y = HEIGHT - 2;
                    for (i = 0; i < wht; i++)
                       downward(&y);
                    y = 0;
                    break;
         case '\t': if (x + NEXTTAB < wwd)  {
                       if (inserting)
                          tabctr = NEXTTAB;
                       else
                          x += NEXTTAB;
                    }
                    else
                       carrtn(&x, &y, inserting);
                    break;
          case SHIFT_HT:
                    if (x < TAB)    {
```

```
                x = LASTTAB;
            }
            else
                x -= PREVTAB;
            break;
        case CTRL_FWD:
            fore_word(&x, &y, buff);
            break;
        case CTRL_BS:
            back_word(&x, &y , buff);
            break;
        case CTRL_B:
            y = wht - 1;
            break;
        case CTRL_T:
            y = 0;
            break;
        case CTRL_HOME:
            x = y = 0;
            bfptr = topptr;
            display_text();
            break;
        case HOME: x = 0;
            break;
        case CTRL_END:
            find_end(&x, &y);
            display_text();
            break;
        case END: last_char(&x, &y);
            break;
        case UP:  upward(&y);
            break;
        case F2:
        case ESC: depart = TRUE;
            break;
        case '\b':
        case BS:  if (curr(x, y) == topptr)
                break;
            backspace(&x, &y);
            if (x == wwd - 1)
                last_char(&x, &y);
            if (c ==BS)
                break;
            buff = curr(x, y);
        case DEL: movmem(buff+1, buff, wwd-1-x);
            $(buff+wwd-1-x) = ' ';
            disp_line(y);
            test_para(x+1, y);
            break;
        case ALT_D:
            delete_line(y);
            break;
        case CTRL_D:
            delete_word(x, y);
            test_para(x, y);
            break;
        case INS: inserting = TRUE;
            insert_line();
            break;
        case F3:  erase_buffer(&x, &y);
            break;
        case F4:  paraform(0, y);
            break;
        case F5:  blkbeg = lineno(y) + 1;
```

```
                display_text();
                break;
        case F6:  blkend = lineno(y) + 1;
                if (blkend < blkbeg)
                    blkbeg = 1;
                display_text();
                break;
        case F7:  move_block(y);
                break;
        case F8:  copy_block(y);
                break;
        case F9:  delete_block();
                break;
        case F10: blkbeg = blkend = 0;
                display_text();
                break;
        case FWD: forward(&x, &y);
                break;
        default: /* if (lisprint(c))
                    break; */
                if (curr(x, y) == endptr-1 ||
                   (lineno(y)+1 >=lines && inserting
                        && *curr(wwd-2, y) != ' ')) {
                    error_message("End of Buffer ");
                    break;
                }
                if (inserting) {
                    buff = curr(x, y);
                    movmem(buff, buff + 1, wwd-1-x);
                }
                buff = curr(x, y);
                if (buff < endptr)  {
                    if (buff >= lstptr)
                        lstptr = buff + 1;
                    *buff = c;
                    disp_line(y);
                }
                buff = curr(wwd-1, y);
                if (endptr && *buff != ' ')  {
                    for (b = buff+1; b < endptr; b++)
                        if (*b==' ' && *(b + 1)==' ')
                            break;
                    movmem(buff+1, buff+2, b-buff-1);
                    *(buff+1) = ' ';
                    svx = x;
                    svlw = lastword(x, y);
                    x = wwd-1;
                    if (*(buff-1) != ' ')
                        back_word(&x, &y, buff);
                    tx = x;
                    carrtn(&x, &y, TRUE);
                    if (svlw)
                        x = svx-tx;
                    else    {
                        x = svx;
                        --y;
                    }
                }
                forward(&x, &y);
                break;
        }
        if (depart)
            break;
    }
    inserting = FALSE;
    insert_line();
```

```
/*--------------------- erase the buffer ----------------------*/
static void erase_buffer(int *x, int *y)
{
      int c = 0;
      WINDOW *swr;

      swr = establish_window(28, 11, 4, 24);
      set_colors(swr, ALL, RED, YELLOW, BRIGHT);
      display_window(swr);
      wprintf(swr, "Erase text window\n Are you sure? (y/n)");
      while (c != 'y' && c != 'n')   {
          c = get_char();
          c = tolower(c);
          if (c == 'y')   {
              lstptr = bfptr = topptr;
              *x = *y = 0;
              setmem(bfptr, lines * wwd, ' ');
              blkbeg = blkend = 0;
              display_text();
          }
       }
       delete_window(swr);
}

/*---------- see if a word is the last word on the line ------*/
static int lastword(int x, int y)
{
    char *bf = curr(x, y);

    while (x++ < wwd-1)
        if (*bf++ == ' ')
            return 0;
    return 1;
}

/*------ go to last dis playable character on the line ------*/
static void last_char(int *x, int *y)
{
     char *bf;

     *x = wwd-1;
     bf = curr(0, *y);
     while (*x && *(bf + *x) == ' ')
         --(*x);
     if (*x && *x < wwd - 1)
         (*x)++;
 }

 /*------ test to see if paragraph should be reformed -------*/
 static void test_para(int x , int y)
 {
     int ts, fw;

     if (!scroll_lock() && y < lines)    {
         ts = trailing_spaces(y);
         fw = first_wordlen(y+1);
         if (fw && ts > fw)
         paraform(x, y);
     }
 }
/*--------- count the trailing spaces on a lines -----------*/
static int trailing_spaces(int y)
{
```

```
    char *bf = curr(x, y);

    while (x >= 0)  {
        if (*(bf + x)  != ' ')
            break;
        --x;
        ct++;
    }
    return ct;
}

/*--------- count the length of the first word on a line --------*/
static int first_wordlen(int y)
{
    int ct = 0, x = 0;
    char *bf = curr(0, y);

    while (x < wwd-1 && *(bf+x) == ' ')
        x++;
    while (x+ct < wwd-1 && *(bf+x+ct)  != ' ')
        ct++;
    return ct;
}

/*-------------- form a paragraph --------------*/
static void paraform(int x, int y)
{
    char *cp1, *cp2, *cpend, *svcp;
    int x1;

    if (blankline(lineno(y)+1))
        return;
    if (!blkbeg)   {
        blkbeg = blkend = lineno(y)+1;
        blkend++;
        while (blkend < lines)   {
            if (blankline(blkend))
                break;
            blkend++;
        }
        --blkend;
    }
    if (lineno(y) != blkbeg-1)
        x = 0;
    x1 = x;
    cp1 = cp2 = topptr + (blkbeg - 1) * wwd + x;
    cpend = topptr + blkend * wwd;
    while (cp2 < cpend)  {
        while (*cp2 == ' ' && cp2 < cpend)
            cp2++;
        if ( cp2 == cpend)
            break;
        /* at a word */
        while (*cp2 != ' ' && cp2 < cpend)  {
            if (x1 >= wwd - 1)   {
                /* warp the word */
                svcp = cp1 + (wwd - x1);
                while (*--cp1 != ' ')   {
                    *cp1 = ' ';
                    --cp2;
                }
                x1 = 0;
                blkbeg++;
                cp1 = svcp;
            }
```

```
            x1++;
        }
        if (cp2 < cpend)    {
                *cp1++ = ' ';
                x1++;
        }
    }
    while (cp1 < cpend)
          *cp1++ = ' ';
    blkbeg++;
    if (blkbeg <= blkend)
        delete_block();
    blkbeg = blkend = 0;
    display_text();
    findlast();
}

/*------------ test for a blank line -------------*/
static int blankline(int line)
{
    char *cp;
    int x;

    cp = topptr + (line-1) * wwd;
    for (x = 0; x < wwd; x++)
        if (*(cp + x) != ' ')
            break;
    return (x == wwd);
}

/*------------ delete a word -----------------*/
static void delete_word(int x, int y)
{
    int wct = 0;
    char *cp1, *cp2;

    cp1 = cp2 = curr(x, y);
    if (*cp2 == ' ')
        while (*cp2 == ' ' && x + wct < wwd)   {
            wct++;
            cp2++;
        }
    else {
        while (*cp2 != ' ' && x + wct < wwd)    {
            wct++;
            cp2++;
        }
        while (*cp2 == ' ' && x + wct < wwd)   {
            wct++;
            cp2++;
        }
    }
    movmem(cp2, cp1, wwd - x - wct);
    setmem(cp1 + wwd - x - wct, wct, ' ');
    display_text();
    findlast();
}

/*------------ delete a line ----------------*/
static void delete_line(int y)
{
    char *cp1, *cp2;
    int len;
```

```
    if (cp1 < lstptr)    {
        len = endptr - cp2;
        movmem(cp2, cp1, len);
        lstptr -= wwd;
        setmem(endptr - wwd, wwd, ' ');
        display_text();
    }
}

/*------------ delete a block ---------------*/
static void delete_block()
{
    char *cp1, *cp2;
    int len;

    if (!blkbeg || !blkend)    {
        putchar(BELL);
        return;
    }
    cp1 = topptr + blkend * wwd;
    cp2 = topptr + (blkbeg - 1) * wwd;
    len = endptr - cp1;
    movmem(cp1, cp2, len);
    setmem(cp2 + len, endptr - (cp2 + len), ' ');
    blkbeg = blkend = 0;
    lstptr -= (cp1 - cp2);
    display_text();
}

/*--------------move and copy text blocks---------------*/
static void mvblock(int y, int moving)
{
    char *cp1, *cp2, *hd;
    int len;
    if (!blkbeg || !blkend) {
        putchar(BELL);
        return;
    }
    if (lineno(y) >= blkbeg-1 && lineno(y) <= blkend-1)   {
        error_message("Can't move/copy a block into itself");
        return;
    }
    len = (blkend - blkbeg + 1) * wwd;
    if ((hd = malloc(len)) == 0)
        return;
    cp1 = topptr +(blkbeg-1) * wwd;
    movmem(cp1, hd, len);
    cp2 = topptr + lineno(y) * wwd;
    if (moving)   {
        if (lineno(y) > blkbeg-1)
            cp2 -= len;
        do_display_text = 0;
        delete_block();
        do_display_text = 1;
    }
    if (cp2+len <= endptr)  {
        movmem(cp2, cp2 + len, endptr - cp2 - len);
        movmem(hd, cp2, len);
    }
    free(hd);
    blkbeg = blkend = 0;
    display_text();
}
```

```
{
   mvblock(y, FALSE);
   findlast();
}
/*----------------- move a block -----------------*/
static void move_block(int y)
{
    mvblock(y, TRUE);
}

/*----- find the last chracter in the buffer -----*/
static void findlast()
{
    register char *lp = endptr - 1;
    register char *tp = topptr;

    while (lp > tp && (*lp == ' ' || *lp == '\0'))
    {
    if (*lp == '\0')
        *lp = ' ';
    --lp;
    }
    if (*lp != ' ')
        lp++;
    lstptr = lp;
}

/*--- go to the end of the data in the buffer ----*/
static void find_end(int *x, int *y)
{
    int ct;

    bfptr = lstptr;
    ct = (lstptr - topptr) % wsz;
    bfptr -= ct;
    if (bfptr + wsz > endptr)
        bfptr = endptr - wsz;
    *y = (ct / wwd);
    *x = 0;
    downward(y);
}

/*---------------- carriage return --------------*/
static void carrtn(int *x, int *y, int insert)
{
    int insct;
    char *cp, *nl;
    int ctl = 2;

    cp = curr(*x, *y);
    nl = cp + ((cp - topptr) % wwd);
    if (lineno(*y) + 2 < lines)
        if (insert && nl < endptr)
        {
           insct = wwd - *x;
           while (ctl--)
           {
             if (endptr > cp + insct)

             {
                movmem(cp, cp+insct, endptr-insct-cp);
                setmem(cp, insct, ' ');
             }
             else if (ctl == 1)
```

```
                insct = *x;
              }
            }
        *x = 0;
        downward(y);
        if (insert)
        {
            test_para(*x, *y);
            display_text();
        }
        if (lineno(*y) + 2 < lines)
           if (insert)
               if ((lstptr + wwd) <= endptr)
                   if (lstptr > curr(*x, *y))
                       lstptr += wwd;
    }

    /*--- move the buffer offset back one position ---*/
    static void backspace(int *x, int *y)
    {
        if (*x == 0)
        {
            *x = wwd - 1;
            upward(y);
        }
        else
           --(*x);
    }

    /*---- move the buffer offset forward one word ---*/
    static void fore_word(int *x, int *y, char *bf)
    {
        while (*bf != ' ')
        {
           if (spaceup(x, y, &bf) == 0)
               return;
           if (*x == 0)
               break;
        }
        while (*bf == ' ')
           if (spaceup(x, y, &bf) == 0)
               return;
    }

    static int spaceup(int *x, int *y, char **bf)
    {
        if (*bf == lstptr)
            return 0;
        (*bf)++;
        forward(x, y);
        return 1;
    }

    /*--- move the buffer offset backward one word ---*/
    static void back_word(int *x, int *y, char *bf)
    {
        spacedn(x, y, &bf);
        while (*bf == ' ')
           if (spacedn(x, y, &bf) == 0)
               return;
        while (*bf != ' ')
        {
           if (*x == 0)
               return;
```

```
    }
    spaceup(x, y, &bf);
}

static int spacedn(int *x, int *y, char **bf)
{
    if (*bf == topptr)
        return 0;
    --(*bf);
    backspace(x, y);
    return 1;
}

/*-- move the buffer offset forward one position -*/
static void forward(int *x, int *y)
{
    int ww = wwd;

    (*x)++;
    if (*x == ww)
    {
        downward(y);
        *x = 0;
    }
}

/*--- move the buffer offset down one position ---*/
static int downward(int *y)
{
    if (*y < wht - 1)
    {
        (*y)++;
        return 1;
    }
    else if ((bfptr + wsz) < endptr)
    {
        bfptr += wwd;
        scroll(wnd, UP);
        disp_line(wht-1);
        return 1;
    }
    return 0;
}

/*---- move the buffer offset up one position ----*/
static void upward(int *y)
{
    if (*y)
        --(*y);
    else if ((topptr + wwd) <= bfptr)
    {
        bfptr -= wwd;
        scroll(wnd, DN);
        disp_line(0);
    }
}

/*------- display all the lines in a window ------*/
static void display_text()
{
    int y = 0;
    if (do_display_text)
        while (y < wht)
```

```
/*---------------- display a lines ---------------*/
static void disp_line(int y)
{
    int x = 0, atr = WNORMAL;

    if (blkbeg || blkend)
        if (lineno(y) >= blkbeg-1)
            if (lineno(y) <= blkend-1)
                atr = WACCENT;
    while (x < wwd)
    {
        displ(wnd, x+1, y+1, *(bfptr+y * wwd+x), atr);
        x++;
    }
}

/*------- set insert/exchange cursor shape -------*/
static void insert_line()
{
    set_cursor_type(inserting ? 0x0106 : 0x0607);
}


/*---------------------- end ---------------------*/
```

```
A>type entry.c
/* ---------------entry.c---------*/

#include <stdio.h>
#include <ctype.h>
#include <stdlib.h>
#include <alloc.h>
#include <mem.h>
#include <string.h>
#include "twindow.h"
#include "keys.h"

#define FIELDCHAR '_'
int insert_mode = FALSE;          /* insert mode , TRUE/FALSE  */
extern int helpkey;

/*--------- local prototypes-------*/
void addfield(WINDOW *wnd, FIELD *fld);
void disp_field(WINDOW *wnd, char *bf, char *msk);
void data_value(WINDOW *wnd, FIELD *fld);
void insert_status(void);
int read_field(WINDOW *wnd, FIELD *fld);
void right_justify(char *s);
void right_justify_zero_fill(char *s);
int validate_date(char *s);
int endstroke(int c);
int spaces(char *c);

/*---------- initialize a template ------*/
void init_template(WINDOW *wnd)
{
     FIELD *fld, *fl;

     fld = FHEAD;
     while (fld) {
         fl = fld->fnxt;
         free(fld);
         fld = fl;
     }
     FHEAD = NULL;
}
/*-----------establish a field in a template------*/
FIELD *establish_field(wnd, cl, rw, msk, bf, ty)
WINDOW *wnd;
int rw;
int cl;
char *msk;
char *bf;
int ty;
{
       FIELD *fld ;

       if ( (fld = malloc(sizeof(FIELD))) == NULL)
            return NULL;
       fld->fmask = msk;
       fld->frow = rw;
       fld->fcol = cl;
       fld->fbuff = bf;
```

```
        fld->fvalid = NULL ;
        fld->fhelp = NULL;
        fld->fhwin = NULL;
        fld->flx = fld->fly =0;
        addfield(wnd, fld);
        return fld;
}

/*--------add a field to the end of the list --------*/
static void  addfield(WINDOW *wnd, FIELD  *fld)
{
        if (FTAIL)   {
             fld->fprv = FTAIL;
             FTAIL->fnxt = fld ;
        }
        FTAIL = fld;
        if (!FHEAD )
            FHEAD = fld;
}

/*--------display a data field ----*/
static void disp_field(WINDOW *wnd, char *bf, char *msk)
{
      while (*msk)    {
          wputchar(wnd, *msk != FIELDCHAR ? *msk : *bf++ ) ;
          msk++;
          }

}


/*--------disply the datavalue in a field -----*/
static void data_value(WINDOW *wnd, FIELD *fld)
{
       wcursor(wnd, fld->fcol, fld->frow);
       disp_field(wnd, fld->fbuff, fld->fmask);
}

/*---------display all the fieldsin the window -----*/
void field_tally(WINDOW *wnd)
{
       FIELD *fld ;

       fld = FHEAD;
       while (fld != NULL ){
          data_value(wnd, fld);
          fld = fld->fnxt;
       }
}

/*-------set a field's help window -------*/
void field_window(FIELD *fld, char *hwin, int x, int y)
{
       fld->fhwin=hwin;
       fld->flx = x;
       fld->fly = y;
}

/*----- clear a template to all blanks ------*/
void clear_template(WINDOW *wnd)
{
       FIELD *fld ;
       char *bf, *msk;
```

```
        while (fld != NULL) {
            bf= fld->fbuff;
            msk = fld->fmask;
            while (*msk)   {
              if (*msk == FIELDCHAR)
                  *bf++ = ' ';
                  msk++;
            }
            fld = fld->fnxt;
        }
        field_tally(wnd);
}

/*---------set insert /exchange cursur ----------*/
static void insert_status()
{
      set_cursor_type(insert_mode ? 0x0106 : 0x0607);
}

/*--------read a field from  the keyboard -------*/
static int read_field(WINDOW *wnd, FIELD *fld)
{
        char *mask = fld->fmask, *buff = fld->fbuff;
        int  done = FALSE, c, column;

        column = fld->fcol;
        while (*mask !=  FIELDCHAR ) {
            column++;
            mask++;
        }
        while  (TRUE)     {
            wcursor(wnd, column, fld->frow);
            c = get_char();
            if (fld->ftype == 'A' )
               c=toupper(c);
            clear_message();
            switch (c)  {
                case  '\b':
                case BS:
                  if (buff == fld->fbuff) {
                      done = c == BS ;
                      break;
                  }
                  --buff;
                  do
                    {
                    --mask;
                    --column;
                    } while (*mask != FIELDCHAR);
                    if (c == BS )
                        break;
                case DEL:
                   movmem(buff+1, buff, strlen(buff));
                   *(buff+strlen(buff)) = ' ';
                   wcursor(wnd,column, fld->frow);
                   disp_field(wnd, buff, mask);
                   break;
                case  FWD:
                   do  {
                        column++;
                        mask++ ;
                   } while  (*mask && *mask !=      FIELDCHAR);
                   buff++;
                   break;
```

```
            insert_status();
            break;
        case '.':
            if (fld->ftype == 'C' ) {
                if (*mask++ && *buff == ' ') {
                    *buff++ = '0';
                    if (*mask++ && *buff == ' ')
                        *buff = '0';
                }
                right_justify(fld->fbuff);
                wcursor(wnd, fld->fcol, fld->frow);
                disp_field(wnd, fld->fbuff, fld->fmask);
                column = fld->fcol+strlen(fld->fmask)-2;
                mask = fld->fmask+strlen(fld->fmask)-2;
                buff = fld->fbuff+strlen(fld->fbuff)-2;
                break;
            }
        default:
            if (endstroke(c))  {
                done = TRUE;
                break;
            }
            if  (toupper(fld->ftype) != 'A'&&!isdigit(c))  {
                error_message("Numbers only ");
                break;
            }
            if (insert_mode)  {
                movmem(buff, buff+1, strlen(buff)-1);
                disp_field(wnd, buff, mask);
                wcursor(wnd,column, fld->frow);
            }
            *buff++ = c;
            wputchar(wnd, c );
            do  {
                    column++;
                    mask++ ;
                } while (*mask && *mask != FIELDCHAR);
                if ( !*mask)
                c = FWD;
                break;
    }
    if (!*mask)
        done = TRUE;
    if  (done)  {
        if (fld->ftype == 'D' &&
              c != ESC &&
                 validate_date(fld->fbuff) != OK)
            return  ERROR;
        break;
    }
}
if (c != ESC && toupper(fld->ftype)  != 'A' ) {
    if  (fld->ftype ==  'C' )  {
        if (*mask++ && *buff == ' ' )  {
            *buff++ = '0' ;
            if (*mask++ && *buff == ' ' )
                    *buff++ = '0';
        }
    }
    if  (fld->ftype == 'I' || fld->ftype == 'D' )
        right_justify_zero_fill(fld->fbuff);
    else
        right_justify(fld->fbuff);
    wcursor(wnd, fld->fcol, fld->frow);
```

```
            return c;
        }
/*-------test c for an ending keystroke -------*/
static int endstroke(int c)
{
            switch (c)   {
                case '\r' :
                case '\n' :
                case '\t' :
                case ESC :
                case F1 :
                case F2 :
                case F3 :
                case F4 :
                case F5 :
                case F6 :
                case F7 :
                case F8 :
                case F9 :
                case F10 :
                case PGUP :
                case PGDN :
                case HOME :
                case END :
                case UP :
                case DN :
                   return  TRUE ;
                default:
                   return  FALSE;
            }
        }
/*-----------right justify ,space fill ------*/
static void right_justify(char *s)
{
       int len;

       len =strlen(s);
       while (*s == ' '  || *s == '0' && len ) {
           len--;
           *s++ = ' ';
       }
       if (len)
          while (*(s+(len-1)) == ' ') {
              movmem(s, s+1, len-1);
              *s = ' ';
          }
       }
/*-------right justify,zero fill ------*/
static void right_justify_zero_fill(char *s)
{
    int len;

    if (spaces(s))
        return;
    len = strlen(s);
    while (*(s + len - 1) == ' ' ) {
        movmem(s, s + 1, len-1);
        *s = '0';
    }
}
/*--------test for spaces -------*/
int spaces(char *c)
{
   while (*c == ' ')
```

```
}
/*---------- validate a date ----------*/
static int validate_date(char *s )
{
        static int days [] =
           {31,28,31,30,31,30,31,31,30,31,30,31};
        char date [7];
        int mo;

        strcpy(date, s);
        if (spaces(date))
            return OK;
        days[1] = (atoi(date+4)%4) ? 28 :29 ;
        *(date + 4 ) ='\0';
        mo =atoi(date+2);
        *(date+2) = '\0';
        if (mo && mo<13 && atoi(date) && atoi(date)<=days[mo-1])
            return OK;
        error_message("Invalid date");
        return ERROR;
}
/*---------Process data entry for a screen template.------*/
int data_entry(WINDOW *wnd)
{
     FIELD *fld ;
     int exitcode, isvalid, done=FALSE, oldhelpkey=helpkey;
     field_tally(wnd);
     fld = FHEAD;
     /*------collect data from keyboard into screen ---*/
     while (fld != NULL && done == FALSE)  {
          set_help(fld->fhwin, fld->flx, fld->fly);
          helpkey =(fld->fhelp) ? 0 : oldhelpkey;
          wcursor(wnd, fld->fcol, fld->frow);
          if (fld->fprot == FALSE)  {
             reverse_video(wnd);
             data_value(wnd, fld);
             wcursor(wnd, fld->fcol, fld->frow);
             exitcode = read_field(wnd, fld);
             isvalid = (exitcode != ESC && fld->fvalid) ?
                     (*(fld->fvalid))(fld->fbuff) : OK ;
             }
             else  {
                 exitcode = FWD;
                 isvalid = OK;
             }
             if (isvalid == OK )  {
                normal_video(wnd);
                data_value(wnd, fld);
                switch (exitcode) { /* passes edit  */
                    case F1 : if (fld->fhelp) {
                                (*(fld->fhelp))(fld->fbuff);
                                data_value(wnd, fld);
                                }
                                break;
                    case DN :
                    case '\r':
                    case '\t' :
                    case FWD : fld = fld->fnxt;
                               if (fld == NULL )
                                   fld = FHEAD ;
                               break;
                    case UP:
                    case BS:   fld = fld->fprv;
                               if (fld == NULL )
```

```
                default:    done =endstroke(exitcode);
                            break;

            }
        }
    }
    helpkey = oldhelpkey;
    return (exitcode);
}
/*-------display a window prompt -----*/
void wprompt(WINDOW *wnd, int x, int y,char *s)
{
    wcursor(wnd, x, y);
    wprintf(wnd, s);
}
```

A>type thelp.c

```
/* --------- thelp.c ----------- */

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "twindow.h"
#include "keys.h"

#define MAXHELPS 25
#define HBG WHITE
#define HFG BLACK
#define HINT DIM

#define TRUE 1
#define FALSE 0

static struct helps {
    char hname [9];
    int h, w;
    long hptr;
} hps [MAXHELPS+1];

static int hp = 0;
static int ch = 0;
static int hx, hy;
FILE *helpfp = NULL;
long ftell();
char *fgets();
void help();
char helpname[64];
void getline (char *lineh);

/* ----------- load the HELP! definition file ----------------- */
void load_help(char *hn)
{
    extern void (*helpfunc)();
    extern int helpkey;
    char lineh [80];

    if (strcmp(helpname, hn) == 0)
        return;
    helpfunc = help;
    helpkey = F1;
    hp = 0;
    strcpy(helpname, hn);
    if ((helpfp = fopen(helpname, "r")) == NULL)
        return;
    getline(lineh);
    while (1)  {
        if (hp == MAXHELPS)
            break;
        if (strncmp(lineh, "<end>", 5) == 0)
            break;
        if (*lineh != '<')
            continue;
        hps[hp].h = 3;
        hps[hp].w = 18;
        strncpy(hps[hp].hname, lineh+1,8);
        hps[hp].hptr = ftell(helpfp);
```

```
        getline(lineh);
        while (*lineh != '<')   {
            hps[hp].h++;
            hps[hp].w = max(hps[hp].w,strlen(lineh)+2);
            getline(lineh);
        }
        hp++;
    }
}

/* -------- get a line of text form the help file -------- */
static void getline(char *lineh)
{
    if (fgets(lineh, 80, helpfp) == NULL)
        strcpy(lineh, "<end>");
}
/*---------- set the current active help screen --------*/
void set_help(char *s, int x, int y)
{
for (ch = 0 ;ch < hp; ch++)
        if (strncmp(s, hps[ch].hname, 8) == 0)
           break;
        hx = x;
        hy = y;
}
/* ---------- display the current help window ---------- */
void help()
{
    char ln [80];
    int i, xx, yy;
    WINDOW *wnd;
    extern int helpkey;
    if (hp && ch !=hp) {
        curr_cursor(&xx, &yy);
        cursor(0, 25);
        wnd = establish_window(hx, hy, hps[ch].h, hps[ch].w);
        set_colors(wnd, ALL, HBG, HFG, HINT);
        set_title(wnd,"");
        display_window(wnd);
        fseek(helpfp, hps[ch].hptr, 0);
        for (i = 0; i < hps[ch].h-3; i++)   {
            getline(ln);
            wprintf(wnd, ln);
        }
        wprintf(wnd, " [Help] to return");
        while (get_char() != helpkey)
            putchar(BELL);
        delete_window(wnd);
        cursor(xx, yy);
    }
}
```

```
A>type tmenu.c
/* ---------- tmenu.c ---------- */

#include  <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include "keys.h"
#include "twindow.h"

extern int VSG;

WINDOW *open_menu(char *mnm, MENU *mn, int hsel);
int gethmenu(MENU *mn, WINDOW *hmenu, int hsel);
int getvmn(MENU *mn, WINDOW *hmenu, int *hsel, int vsel);
int haccent(MENU *mn, WINDOW *hmenu, int hsel, int vsel);
void dimension(char *sl[], int *ht, int *wd);
void light(MENU *mn, WINDOW *hmenu, int hsel, int d);

/* ------------- display & process a menu ----------- */
void menu_select(char *name, MENU *mn)
{
    WINDOW *open_menu();
    WINDOW *hmenu;
    int sx, sy;
    int hsel = 1, vsel;

    curr_cursor(&sx, &sy);
    cursor(0, 26);
    hmenu = open_menu(name, mn, hsel);
    while (hsel = gethmenu(mn, hmenu, hsel))    {
        vsel = 1;
        while (vsel = getvmn(mn, hmenu, &hsel, vsel))    {
            delete_window(hmenu);
            set_help("", 0, 0);
            (*(mn+hsel-1)->func [vsel-1])(hsel, vsel);
            hmenu = open_menu(name, mn, hsel);
        }
    }
    delete_window(hmenu);
    cursor(sx, sy);
}

/* ------- open a horizontal menu ------- */
static WINDOW *open_menu(char *mnm, MENU *mn, int hsel)
{
    int i = 0;
    WINDOW *hmenu;

    set_help("menu    ", 30, 10);
    hmenu = establish_window(0, 0, 3, 80);
    set_title(hmenu, mnm);
    set_colors(hmenu, ALL, BLUE, AQUA, BRIGHT);
    set_colors(hmenu, ACCENT, WHITE, BLACK, DIM);
    display_window(hmenu);
    while ((mn+i)->mname)
        wprintf(hmenu, " %-10.10s ", (mn+i++)->mname);
    light(mn, hmenu, hsel, 1);
    cursor(0, 26);
    return hmenu;
}
```

```
/*  ----    get a horizontal selection              */
static int gethmenu(MENU *mn, WINDOW *hmenu, int hsel)
{
    int sel;

    light(mn, hmenu, hsel, 1);
    while (TRUE)    {
        switch (sel = get_char())    {
            case FWD:
            case BS:    hsel = haccent(mn, hmenu, hsel, sel);
                        break;
            case ESC:   return 0;
            case '\r':  return hsel;
            default:    putchar(BELL);
                        break;
        }
    }
}

/* -------- pop down a vertical menu --------- */
static int getvmn(MENU *mn,WINDOW *hmenu,int *hsel,int vsel)
{
    WINDOW *vmenu;
    int ht = 10, wd = 20;
    char **mp;

    while (1)   {
        dimension((mn+*hsel-1)->mselcs, &ht, &wd);
        vmenu = establish_window(2+(*hsel-1)*12, 2, ht, wd);
        set_title(vmenu,"");
        set_colors(vmenu, ALL, BLUE, AQUA, BRIGHT);
        set_colors(vmenu, ACCENT, WHITE, BLACK, DIM);
        set_border(vmenu, 4);
        display_window(vmenu);
        mp = (mn+*hsel-1)->mselcs;
        while(*mp)
            wprintf(vmenu, "\n%s", *mp++);
        vsel = get_selection(vmenu, vsel, "");
        delete_window(vmenu);
        if (vsel == FWD || vsel == BS)    {
            *hsel = haccent(mn, hmenu, *hsel, vsel);
            vsel = 1;
        }
        else
            return vsel;
    }
}

/* ----- manage the horizontal menu selection accent ----- */
static int haccent(MENU *mn,WINDOW *hmenu,int hsel,int sel)
{
    switch (sel)    {
        case FWD:
            light(mn, hmenu, hsel, 0);
            if ((mn+hsel)->mname)
                hsel++;
            else
                hsel = 1;
            light(mn, hmenu, hsel, 1);
            break;
        case BS:
            light(mn, hmenu, hsel, 0);
            if (hsel == 1)
                while ((mn+hsel)->mname)
```

```
            --hsel;
            light(mn, hmenu, hsel, 1);
            break;
        default:
            break;
    }
    return hsel;
}

/* ---------- compute a menu's height & width --------- */
static void dimension(char *sl[], int *ht, int *wd)
{
    unsigned strlen(char *);

    *ht = *wd = 0;
    while (sl [*ht])    {
        *wd = max(*wd, strlen(sl [*ht]));
        (*ht)++;
    }
    *ht += 2;
    *wd += 2;
}

/* --------- accent a horizontal menu selection ---------- */
static void light(MENU *mn, WINDOW *hmenu, int hsel, int d)
{
    if (d)
        reverse_video(hmenu);
    wcursor(hmenu, (hsel-1)*12+2, 0);
    wprintf(hmenu, (mn+hsel-1)->mname);
    normal_video(hmenu);
    cursor(0, 26);
}
```

```
A>type twindow.c
/*----------------------twindow.c-----------------------*/
#include <stdio.h>
#include <ctype.h>
#include <stdarg.h>
#include <dos.h>
#include <alloc.h>
#include <stdlib.h>
#include <string.h>
#include "twindow.h"
#include "keys.h"

#define TABS     4
#define SCRNHT   25
#define SCRNWIDTH 80
#define ON       1
#define OFF      0
#define ERROR    -1
/*--------------------------local prototype-----------------------------*/

redraw(WINDOW *wnd);
wframe(WINDOW *wnd);
dtitle(WINDOW *wnd);
int *waddr(WINDOW *wnd, int x, int y);
vswap(WINDOW *wnd);
vsave(WINDOW *wnd);
vrstr(WINDOW *wnd);
add_list(WINDOW *wnd);
beg_list(WINDOW *wnd);
remove_list(WINDOW *wnd);
insert_list(WINDOW *w1,WINDOW *w2);
#ifndef FASTWINDOWS
int dget(WINDOW *wnd, int x, int y);
verify_wnd(WINDOW **w1);
#endif
/*----array of border character sets--------*/
struct {
        int nw, ne, se, sw, side, line;
} wcs[] = {
        {218,191,217,192,179,196},
        {201,187,188,200,186,205},
        {214,183,189,211,186,196},
        {213,184,190,212,179,205},
        {194,194,217,192,179,196}
};
/*--------window structure linked list head & tail -------------*/
WINDOW *listhead =NULL;
WINDOW *listtail =NULL;
int VSG; /*vedeo segment address */
/*----------------- establish a new window ----------------------*/
WINDOW *establish_window(x, y, h, w)
{
        WINDOW *wnd;
        VSG = (vmode() == 7 ? 0xb000 : 0xb800);
        if ((wnd = (WINDOW *) malloc(sizeof (WINDOW))) == NULL )
/*----adjust for out-of bounds parameter ----------------------*/
        WTITLE = "";
        HEIGHT = min(h, SCRNHT);
        WIDTH = min(w, SCRNWIDTH);
        COL = max(0, min(x, SCRNWIDTH-WIDTH));
        ROW = max(0, min(y, SCRNHT-HEIGHT));
```

```
        WCURS = 0;
        SCROLL = 0;
        SELECT = 1;
        BTYPE = 0;
        VISIBLE = HIDDEN = 0;
        PREV = NEXT = NULL;
        FHEAD = FTAIL = NULL;
        WBORDER = WNORMAL = PNORMAL = WTITLEC =
                                clr(BLACK, WHITE, BRIGHT);
        WACCENT = clr(WHITE, BLACK, DIM);
        if ((SAV = malloc(WIDTH * HEIGHT * 2)) == (char *) 0)
                return NULL;
        add_list(wnd);
#ifndef FASTWINDOWS
        clear_window(wnd);
        wframe(wnd);
#endif
        return wnd;
}

/*------------set the window's border-----------------------------------*/
void set_border(WINDOW *wnd, int btype)
{
        if (verify_wnd(&wnd)) {
                BTYPE = btype;
                redraw(wnd);
        }
}

/*----------------------set color-------------------------------------*/
void set_colors(WINDOW *wnd, int area, int bg, int fg, int inten)
{
        if (vmode() == 7) {
         if (bg != WHITE && bg != BLACK)
                return;
         if (fg != WHITE && fg != BLACK)
                return;
        }
        if (verify_wnd(&wnd)) {
         if (area == ALL)
          while (area)
                  WCOLOR [--area] = clr(bg, fg, inten);
         else
                WCOLOR [area] = clr(bg, fg, inten);
         redraw(wnd);
        }
}
/*------------set the intensity of a window----------------------------*/
void set_intensity(WINDOW *wnd, int inten)
{
        int area = ALL;
        if (verify_wnd(&whd)) {
         while (area) {
                WCOLOR [--area] &= ~BRIGHT;
                WCOLOR [area] |= inten;
         }
         redraw(wnd);
        }
}
/*---------------set title-------------------------------------------*/
void set_title(WINDOW *wnd, char *title)
{
        if (verify_wnd(&wnd)) {
```

```
        }
}

/*---------------redraw a window when an attribute changes---------------*/
static redraw(WINDOW *wnd)
{
#ifndef FASTWINDOWS
        int x, y, chat, atr;
        for (y = 1; y < HEIGHT-1; y++)
                for ( x = 1; x < WIDTH-1; x++) {
                  chat = dget(wnd, x, y);
                  atr = (((chat >>8)&255) ==
                        PNORMAL ? WNORMAL : WACCENT);
                  displ(wnd, x, y, chat&255, atr);
                }
        wframe(wnd);
#endif
        PNORMAL = WNORMAL;
}

/*------------display an established window-------------------------------*/
void display_window(WINDOW *wnd)
{
        if (verify_wnd(&wnd) && !VISIBLE) {
                VISIBLE = 1;
#ifdef FASTWINDOWS
        if (HIDDEN) {
         HIDDEN = 0;
         vrstr(wnd);
        }
        else {
          vsave(wnd);
          clear_window(wnd);
          wframe(wnd);
  }
#else
  vswap(wnd);
#endif
        }
}

/*-----------------close all windows-------------------------------------*/
void close_all()
{
 WINDOW *sav, *wnd = listtail;
 while (wnd) {
        sav = PREV;
        delete_window(wnd);
        wnd = sav;
 }
}
/*-----------------remove a window-----------------------------------*/
void delete_window(WINDOW *wnd)
{
        if (verify_wnd(&wnd)) {
         hide_window(wnd);
         free(SAV);
         remove_list(wnd);
         free(wnd);
        }
}
/*---------------hide a window ---------------------------------------*/
void hide_window(WINDOW *wnd)
{
```

```
                vswap(wnd);
#else
                vrstr(wnd);
#endif
                HIDDEN = 1;
                VISIBLE = 0;
        }
}
#ifndef FASTWINDOWS
/*----------reposition the window in its 3-axis plane----------------*/
void repos_wnd(WINDOW *wnd, int x, int y, int z)
{
        WINDOW *twnd;
        int x1, y1, chat;
        if (!verify_wnd(&wnd))
         return;
        twnd = establish_window(x+COL, y+ROW, HEIGHT, WIDTH);
        twnd->_tl = WTITLE;
        twnd->btype = BTYPE;
        twnd->wcolor[BORDER] = WBORDER;
        twnd->wcolor[TITLE] = WTITLEC;
        twnd->wcolor[ACCENT] = WACCENT;
        twnd->wcolor[NORMAL] = WNORMAL;
        twnd->_wsp = SCROLL;
        twnd->_cr = WCURS;
        if (z != 1) {
              remove_list(twnd);
              if (z == 0)
                    insert_list(twnd, wnd);
              else
                    beg_list(twnd);
        }
        for (y1 = 0; y1 < twnd->_wh; y1++)
              for (x1 = 0; x1 < twnd->_ww; x1++) {
                    chat = dget(wnd, x1, y1);
                    displ(twnd, x1, y1, chat&255, (chat>>8)&255);
              }
              twnd->_wv = 1;
              vswap(twnd);
              hide_window(wnd);
              free(SAV);
              remove_list(wnd);
              *wnd = *twnd;
              insert_list(wnd, twnd);
              remove_list(twnd);
              free(twnd);
}
#endif

/*----------------clear the window area---------------------------------*/
void clear_window(WINDOW *wnd)
{
       register int x1, y1;
       if (verify_wnd(&wnd))
        for(y1 = 1; y1 < HEIGHT-1; y1++)
              for (x1 = 1;x1 < WIDTH-1; x1++)
                displ(wnd, x1, y1,' ',WNORMAL);
}

/*---------------- draw the window frame ---------------------------------*/
static wframe(WINDOW *wnd)
{
 register int x1, y1;
  if (!verify_wnd(&wnd))
        return;
/*---------------- window title-----------------------------------------*/
```

```
        displ(wnd, 0, 0, NW, WBORDER);
        dtitle(wnd);
        displ(wnd, WIDTH-1, 0, NE, WBORDER);
/*---------------- window sides ----------------------------------------*/
for (y1 = 1; y1 < HEIGHT-1; y1++) {
        displ(wnd, 0, y1, SIDE, WBORDER);
        displ(wnd, WIDTH-1, y1,SIDE, WBORDER);
}
/*-----------------bottom of frame ---------------------------------*/
        displ(wnd, 0, y1, SW, WBORDER);
        for (x1 = 1; x1 < WIDTH-1; x1++)
                displ(wnd, x1, y1, LINE, WBORDER);
        displ(wnd, x1, y1, SE, WBORDER);
}
/*-------------------displ the window title-----------------------------*/
static dtitle(WINDOW *wnd)
{
        int x1 = 1, i, ln;
        char *s = WTITLE;
        if (!verify_wnd(&wnd))
         return;
        if (s) {
         ln = strlen(s);
         if (ln > WIDTH-2)
          i = 0;
         else
          i = ((WIDTH-2-ln) / 2);
         if (i > 0)
                while (i--)
                 displ(wnd, x1++, 0, LINE, WBORDER);
                while (*s && x1 <WIDTH-1)
                 displ(wnd, x1++, 0, *s++, WTITLEC);
        }
        while (x1 < WIDTH-1)
                displ(wnd, x1++, 0, LINE, WBORDER);
}
/*------------------window-oriented printf-----------------------------*/
void wprintf(WINDOW *wnd, char *ln, ...)
{
        char dlin [100], *dl = dlin;
        if (verify_wnd(&wnd)) {
         va_list ap;
         va_start(ap, ln);
         vsprintf(dlin, ln, ap);
         va_end(ap);
         while (*dl)
          wputchar(wnd, *dl++);
        }
}
/*------------write a character to the window---------*/
void wputchar(WINDOW *wnd, int c )
{
        if (! verify_wnd(&wnd))
         return;
        switch (c) {
        case '\n':
                if( SCROLL == HEIGHT-3)
                        scroll(wnd,UP);
                else
                        SCROLL++;
                WCURS =0;
                break;
        case '\t':
        do displ(wnd,(WCURS++)+3,SCROLL+1,' ',WNORMAL);
                while ((WCURS%TABS) && (WCURS+1) < WIDTH-1);
        break;
```

```
        break;
       default:
        if ((WCURS+1) <WIDTH-1) {
         displ(wnd, WCURS+1, SCROLL+1, c,WNORMAL);
         WCURS++;
        }
        break;
       }
 }
 /*---------set window cursor----------*/
void wcursor(WINDOW *wnd,int x, int y)
{
 if (verify_wnd(&wnd)&& x<WIDTH-1&& y< HEIGHT-1) {
       WCURS = x;
       SCROLL =y;
       cursor(COL+x+1, ROW+y+1);
       }
 }
/*----------------allow the user to make a window selection----*/
int get_selection(WINDOW *wnd, int s, char *keys)
{
 int c=0, ky;
 if (!verify_wnd(&wnd))
       return 0;
 SELECT = s;
 while (c != ESC && c != '\r' && c != BS && c !=FWD) {
  accent(wnd);
  c = get_char();
  deaccent(wnd);
  switch(c) {
       case UP: if (SELECT >1)
                    SELECT--;
             else
                    SELECT = SCROLL+1;
             break;
       case DN: if(SELECT < SCROLL+1)
                     SELECT++;
              else
                    SELECT = 1;
             break;
       case'\r':
       case ESC:
       case FWD:
       case BS: break;
       default: if (keys) {
                    ky = 0;
                    while (*(keys + ky)) {
                     if (*(keys + ky) == toupper(c) ||
                      *(keys + ky) == tolower(c))
                      return ky + 1;
                     ky++;
                    }
       }
       break;
       }
 }
 return c == '\r' ? SELECT : c == ESC ? 0 : c;
}

union REGS rg ;
/*---------------scroll a window's contents up or down ----------------*/
void scroll(WINDOW *wnd, int dir)
{
  int row = HEIGHT-1, col, chat;
  if (!verify_wnd(&wnd))
```

```
  if (next == NULL && height > 3 && visible) {
        rg.h.ah = dir == UP ? 6 : 7;
        rg.h.al = 1;
        rg.h.bh = WNORMAL;
        rg.h.cl = COL + 1;
        rg.h.ch = ROW + 1;
        rg.h.dl = COL + WIDTH - 2 ;
        rg.h.dh = ROW + HEIGHT - 2;
        int86(16, &rg, &rg);
        return;
  }
  if (dir == UP) {
        for (row = 2; row < HEIGHT-1; row++)
         for (col = 1 ; col < WIDTH-1; col++) {
               chat = dget(wnd, col, row);
               displ(wnd, col, row-1, chat&255,(chat>>8)&255);
               }
         for (col=1; col < WIDTH-1; col++)
               displ(wnd, col, row-1, '', WNORMAL);
         }
         else {
         for (row = HEIGHT-2; row > 1; --row)
               for (col = 1; col < WIDTH-1; col++) {
                chat = dget(wnd, col, row-1);
                displ(wnd, col, row, chat&255,(chat>>8)&255);
                }
               for (col = 1; col < WIDTH-1; col++)
                displ(wnd, col, row, '', WNORMAL);
         }
 }
#ifndef FASTWINDOWS
/*------------computer address of a window's display character----------*/
static int *waddr(WINDOW *wnd, int x, int y)
{
  WINDOW *nxt = NEXT;
  int *vp;
  if (!VISIBLE)
        return (int *) (SAV+y*(WIDTH*2)+x*2);
  x += COL;
  y += ROW;
  while (nxt) {
         if (nxt->_wv)
               if (x >= nxt->_wx && x <= nxt->_wx + nxt->_ww-1)
                if (y >= nxt->_wy &&
                       y <= nxt->_wy + nxt->_wh-1) {
                x -= nxt->_wx;
                y -= nxt->_wy;
                vp = (int *)
                      ((nxt->_ws) +y*(nxt->_ww*2)+x*2);
                return vp;
                }
               nxt = nxt->_nx;
         }
         return NULL;
  }
/*-------display a character to a window ------------*/
void displ(WINDOW *wnd, int x, int y, int ch, int at)
{
  int *vp;
  int vch = (ch&255)|(at<<8);
  if ((vp = waddr(wnd, x, y)) != NULL)
        *vp = vch;
  else
        vpoke(VSG,vad(x+COL,y+ROW),vch);
}
```

```
static int dget(WINDOW *wnd, int x, int y )
{
        int *vp;
        if ((vp = waddr(wnd, x, y)) != NULL)
         return *vp;
        return vpeek(VSG,vad(x+COL,y+ROW));
}
/*------------low-level video functions---------------------*/
/*-------swap the video image with the save buffer ---------*/
static vswap(WINDOW *wnd)
{
        int x, y, chat;
        int *bf = (int *) SAV;
        for (y = 0; y < HEIGHT ; y++)
         for (x = 0;x < WIDTH; x++) {
                chat = *bf;
                *bf++ = dget(wnd, x, y);
                displ(wnd, x, y, chat&255, (chat>>8)&255);
         }
}
#else
/*--------save video memory into the save buffer-----------*/
static vsave(WINDOW *wnd)
{
        int x, y;
        int *bf = (int *) SAV;
        for(y = 0; y < HEIGHT; y++)
         for(x = 0;x < WIDTH; x++)
                *bf++ = vpeek(VSG, vad( x+COL, y+ROW));
}
/*-----------restore video memory from the save buffer----*/
static vrstr(WINDOW *wnd)
{
        int x, y;
        int *bf = (int *) SAV;
        for (y = 0 ; y < HEIGHT; y++)
         for (x = 0; x < WIDTH; x++)
          vpoke(VSG, vad(x+COL, y+ROW), *bf++);
}
#endif
/*-------(de)accent the line where SELECT points----------*/
void acline(WINDOW *wnd, int set)
{
        int x, ch;
        if (!verify_wnd(&wnd))
         return;
        for ( x = 1; x < WIDTH-1; x++) {
         ch = dget(wnd, x, SELECT) & 255;
         displ(wnd, x, SELECT, ch, set);
        }
}
/*--------------linked list functions---------------------*/
/*-------add a window to the end of list-----------------*/
static add_list(WINDOW *wnd)
{
        if (listtail) {
        PREV = listtail;
        listtail->_nx = wnd;
        }
        listtail = wnd;
        if (!listhead)
         listhead = wnd;
}
/*--------add a window to the beginning of the list-------*/
```

```
        if (listhead) {
        NEXT = listhead;
        listhead->_pv = wnd;
        }
        listhead = wnd;
        if (!listtail)
         listtail = wnd;
}
/*---------remove the window from the list----------------*/
static remove_list(WINDOW *wnd)
{
        if (NEXT)
        NEXT->_pv = PREV;
        if (PREV)
        PREV->_nx = NEXT;
        if (listhead == wnd)
        listhead = NEXT;
        if (listtail == wnd)
        listtail = PREV;
        NEXT = PREV = NULL;
}
/*----------insert w1 after w2----------------------------*/
static insert_list(WINDOW *w1, WINDOW *w2)
{
        w1->_pv = w2;
        w1->_nx = w2->_nx;
        w2->_nx = w1;
        if (w1->_nx == NULL)
         listtail = w1;
        else
         w1->_nx->_pv = w1;
}
#ifndef FASTWINDOWS
/*-----------verify the presence of a window from the list--*/
static verify_wnd(WINDOW **w1)
{
        WINDOW *wnd;
        wnd = listhead;
        if (*w1 == NULL)
               *w1 = listtail;
        else {
          while (wnd != NULL) {
          if (*w1 == wnd)
                break;
          wnd = NEXT;
          }
        }
        return wnd != NULL;
}
#endif
WINDOW *ewnd = NULL;
/*--------------------error messages------------------*/
void error_message(char *s)
{
        ewnd = establish_window(50, 22, 3, max(10, strlen(s)+2));
        set_colors(ewnd, ALL, RED, YELLOW, BRIGHT);
        set_title(ewnd, "ERROR!");
        display_window(ewnd);
        wprintf(ewnd, s);
        putchar(BELL);
}
void clear_message()
{
        if (ewnd)
```

```
A>type ibmpc.c
#include <dos.h>

static union REGS rg;

void cursor(int x, int y)
{
        rg.x.ax = 0x0200;
        rg.x.bx = 0;
        rg.x.dx = ((y << 8) & 0xff00) + x;
        int86(16, &rg, &rg);
}

 /*------------ return the cursor position ----------*/
 void curr_cursor(int *x, int *y)
 {
        rg.x.ax = 0x0300;
        rg.x.bx = 0;
        int86(16 ,&rg, &rg);
        *x = rg.h.dl;
        *y = rg.h.dh;
 }

 /*----------------set cursor type-------------*/
 void set_cursor_type(int t)
 {
        rg.x.ax = 0x0100;
        rg.x.bx = 0;
        rg.x.cx = t;
        int86(16, &rg, &rg);
 }

char attrib =7;

/*-------------clear the screen -----------*/
void clear_screen()
{
        cursor(0, 0);
        rg.h.al =' ';
        rg.h.ah = 9;
        rg.x.bx = attrib;
        rg.x.cx = 2000;
        int86(16, &rg, &rg);
}

/*------------return the video -----------------*/
int vmode()
{
     rg.h.ah = 15;
     int86(16, &rg ,&rg);
     return  rg.h.al;
}

/*-------------test for scroll lock---------------*/
int scroll_lock()
{
     rg.x.ax = 0x0200;
     int86(0x16, &rg, &rg);
     return rg.h.al & 0x10;
}
```

```
int helpkey = 0;
int helping =0;

/*------------get a keyboard character -----------------*/

int get_char()
{
     int c;

     while (1)   {
       rg.h.ah = 1;
       intB6(0x16, &rg, &rg );
       if (rg.x.flags & 0x40)   {
           intB6(0x28, &rg, &rg);
           continue;
       }
       rg.h.ah = 0;
       intB6(0x16, &rg, &rg);
       if (rg.h.al == 0)
           c = rg.h.ah | 128;
       else
           c = rg.h.al;
       if (c == helpkey && helpfunc)  {
           if (!helping)   {
                helping = 1;
                (*helpfunc)();
                helping = 0;
                continue;
           }
       }
       break;
     }
     return c;
}

/*----insert a character and attibute into video ram -------*/
void vpoke(unsigned vseg, unsigned adr , unsigned chr)
{
       poke(vseg, adr, chr);
}

/*----------read a character and attribute from video RAM-----*/
int vpeek(unsigned vseg, unsigned adr)
{
       int ch, at;

         return peek(vseg,adr);
}
```

กิตติกรรมประกาศ

งานออกแบบระบบควบคุมการสื่อสารข้อมูลในเครือข่ายไมโครคอมพิวเตอร์ทางด้าน SOFTWARE สำเร็จลุล่วงได้เนื่องจากได้รับความช่วยเหลือจากหลายท่านด้วยกันคือ ดร. รัตติกร วรากูลศิริพันธุ์ อาจารย์ที่ปรึกษา ซึ่งได้กรุณาให้ความรู้ทางด้านเครือข่ายไมโครคอมพิวเตอร์ ตอลดจนคำแนะนำทางด้าน SOFTWARE ต่าง ๆ ซึ่งทำให้ผู้เขียนเข้าใจจในระบบการควบคุมการสื่อสารข้อมูลทางเครือข่าย ขอขอบคุณ คุณ สมศักดิ์ จันวัน ซึ่งเป็นนักศึกษาปริญญาโท ที่ให้การควบคุมให้งานนี้บรรลุเป้าหมาย และคุณ ธนา หงษ์สุวรรณ ที่ให้คำแนะนำต่าง ๆ ทางด้าน SOFTWARE

ผู้เขียนจึงขอบคุณเป็นอย่างยิ่ง มา ณ ที่นี้ด้วย

```
                        CUR_START       EQU     09h
                        CUR_STOP        EQU     0ah
                        CUR_ADDR        EQU     3000h
                        AR_LEFT         EQU     0F1H
                        AR_RIGHT        EQU     0F2H
                        AR_UP           EQU     0F3H
                        AR_DOWN         EQU     0F4H
                        COUNT_1         EQU     2048d

0000                    CODE            segment byte public
                                        assume  cs:code,ds:code

0000                    begin           proc    far
0000  8B ---- R                        mov     ax,code
0003  8E D8                            mov     ds,ax
0005  E9 0912 R                        jmp     start

0008  0800[             text            db      2048d dup (?)
        ??
                        ]

080B  00                pull_flag       db      00
0809  00                pull_ret        db      00
080A  0000              win_addr        dw      00
080C  0064[             win_area        db      100 dup (0)
          00
                        ]

0870  00                win_count       db      00
0871  00                pull_count      db      00
                                ; number of call count
0872  0000              pulldown_addr   dw      00
0874  0064[             pull_area       db      100 dup (0)
                                ; Number of call must less then
                         10
          00
                        ]

08D8  0000              Select_Addr     dw      00
08DA  05                No_Fld          db      05
                                ; Number of field
08DB  08                Fld_Len         db      08
08DC  70                Fld_Att         db      070h
                                ; Attribute scroll bar
08DD  70                Fld_Att1        db      070h
                                ; Attribute normal
08DE  00                sel_dir         db      00
                                ; 00 = left -> right
08DF  00                sel_ret         db      00
                                ; return from select
08E0  00                sel_code        db      00
                                ; return key from select

08E1  80 B5 EE F3 F7 FB EA      tab_look        db      080h,085h,0eeh,
```

```
                                0f3h,0f7h,0fbh,0eah
    45
    46 08E8  00                 mode_flag       db      00
                                        ; 00 = not compen att in print
                                string
    47 08E9  01                 win_flag        db      01
                                        ; 00 = not shadow window
    48 08EA  0000               pointer         dw      00
                                        ; pointer to print string
    49 08EC  00                 attrib          db      00
    50 08ED  00                 x_row           db      0
                                        ; use in locate
    51 08EE  00                 x_col           db      0
    52 08EF  00                 w_flag          db      00
                                        ; use in multi frame
    53 08F0  00                 lg_col          db      00
    54 08F1  00                 lg_row          db      00
    55 08F2  05                 upr             db      5
    56 08F3  05                 upc             db      5
    57 08F4  14                 lwr             db      20
    58 08F5  14                 lwc             db      20
    59 08F6  0000               buffer          dw      00
    60 08F8  0000               addr            dw      00
    61 08FA  0000               char            dw      00
    62 08FC  02                 edit_len        db      2
    63 08FD  0000               edit_addr       dw      00
    64 08FF  00                 count           db      00
    65 0900  092C R             jump_tab        dw offset win
    66 0902  094C R                             dw offset clear_win
    67 0904  0969 R                             dw offset key_on
    68 0906  096E R                             dw offset key_off
    69 0908  0C04 R                             dw offset cur_on
    70 090A  0C1E R                             dw offset cur_off
    71 090C  0C67 R                             dw offset print_string
    72 090E  0D6E R                             dw offset set_cursor
    73 0910  0B21 R                             dw offset recall
    74 0912                     start:
    75
    76 0912  80 3C 1B           zzz: cmp byte ptr[si],1bh
    77 0915  75 13                   jne not_command
    78 0917  80 7C 01 92             cmp byte ptr[si+1],92h
    79 091B  75 0D                   jne not_command
    80 091D  BB 0900 R               mov bx,offset jump_tab
    81 0920  02 5C 02                add bl,[si+2]
    82 0923  02 5C 02                add bl,[si+2]
    83 0926  FF 17                   call [bx]
    84 0928  EB E8                   jmp zzz
    85
    86 092A                     not_command:
    87 092A  CD 20                   int 20h
    88
    89 092C                     begin           endp
    90
    91 092C                     win  proc       near
```

```
 92
 93 092C  B3 C6 03                  add si,3
 94 092F  AC                        lodsb
 95 0930  A2 08F2 R                 mov upr,al
 96 0933  AC                        lodsb
 97 0934  A2 08F3 R                 mov upc,al
 98 0937  AC                        lodsb
 99 0938  A2 08F4 R                 mov lwr,al
100 093B  AC                        lodsb
101 093C  A2 08F5 R                 mov lwc,al
102 093F  E8 0973 R                 call window
103 0942  E8 0A9C R                 call save_frame
104 0945  E8 0BBA R                 call clear
105 0948  E8 09AB R                 call frame
106 094B  C3                        ret
107
108 094C                        win  endp
109
110 094C                        clear_win  proc     near
111
112 094C  B3 C6 03                  add si,3
113 094F  AC                        lodsb
114 0950  A2 08F2 R                 mov upr,al
115 0953  AC                        lodsb
116 0954  A2 08F3 R                 mov upc,al
117 0957  AC                        lodsb
118 0958  A2 08F4 R                 mov lwr,al
119 095B  AC                        lodsb
120 095C  A2 08F5 R                 mov lwc,al
121 095F  E8 0973 R                 call window
122 0962  E8 0BBA R                 call clear
123 0965  B3 C6 06                  add si,6
124 0968  C3                        ret
125
126 0969                        clear_win  endp
127
128 0969                        key_on  proc        near
129
130 0969  B3 C6 03                   add si,3
131 096C  CD 20                      int 20h
132
133 096E                        key_on  endp
134
135 096E                        key_off  proc       near
136
137 096E  B3 C6 03                   add si,3
138 0971  CD 20                      int 20h
139
140 0973                        key_off  endp
141
142                             ; to make window
143                             ;       parameter in : upr upc lwr lwc
144                             ; and
145                             ;       call window
```

```
                        ;       call save_window
                        ;       call clear
                        ; afterthat
                        ;       use this window to do anything
                        ;       call recall

0973                    window          proc    near
0973  50                               push    ax
0974  51                               push    cx
0975  33 C0                            xor     ax,ax
0977  A0 08F2 R                        mov     al,upr
097A  FE C8                            dec     al
097C  B1 A0                            mov     cl,160
097E  F6 E1                            mul     cl
0980  8B F0                            mov     si,ax
0982  33 C0                            xor     ax,ax
0984  A0 08F3 R                        mov     al,upc
0987  FE C8                            dec     al
0989  02 C0                            add     al,al
098B  03 C6                            add     ax,si
098D  A3 08F6 R                        mov     Buffer,ax

0990  A0 08F4 R                        mov     al,lwr
0993  2A 06 08F2 R                     sub     al,upr
0997  FE C8                            dec     al
0999  A2 08F1 R                        mov     lg_row,al
099C  A0 08F5 R                        mov     al,lwc
099F  2A 06 08F3 R                     sub     al,upc
09A3  FE C8                            dec     al
09A5  A2 08F0 R                        mov     lg_col,al
09A8  59                               pop     cx
09A9  58                               pop     ax
09AA  C3                               ret
09AB                    window          endp

09AB                    frame           proc    near
09AB  50                               push    ax
09AC  51                               push    cx
09AD  57                               push    di
09AE  06                               push    es
09AF  B8 B200                          mov     ax,0b200h
09B2  8E C0                            mov     es,ax
09B4  A1 08F6 R                        mov     ax,buffer
                                       ; get address of row &
                        column
09B7  8B F8                            mov     di,ax
09B9  8A 1E 08F0 R                     mov     bl,lg_col
                                       ; get length of column
09BD  B8 078A                          mov     ax,078ah
                                       ; write '' on corner
09C0  26: 89 05                        mov     es:[di],ax
09C3  B8 0711                          mov     ax,0711h
09C6  26: 89 85 4000                   mov     es:[di+4000h],a
                        x
```

```
194 09CB  47                                    inc     di
195 09CC  47                                    inc     di
196 09CD  B8 078F                               mov     ax,078fh
                                                ;
197 09D0  26: 89 05                     f10:    mov     es:[di],ax
                                                ;draw up ----- line
198 09D3  47                                    inc     di
199 09D4  47                                    inc     di
200 09D5  FE CB                                 dec     bl
201 09D7  75 F7                                 jnz     f10
202 09D9  B8 078B                               mov     ax,078bh
203 09DC  26: 89 05                             mov     es:[di],ax
204 09DF  B8 0711                               mov     ax,0711h
205 09E2  26: 89 85 4000                        mov     es:[di+4000h],a
                                        x
206 09E7  B8 078E                               mov     ax,078eh
207 09EA  8A 1E 08F1 R                          mov     bl,lg_row
208 09EE  FE C3                                 inc     bl
209 09F0  81 C7 00A0                    f20:    add     di,160
                                                ;draw right ; line
210 09F4  26: 89 05                             mov     es:[di],ax
211 09F7  26: 89 85 4000                        mov     es:[di+4000h],a
                                        x
212 09FC  80 3E 08E9 R 00                       cmp     win_flag,0
213 0A01  74 0D                                 jz      f21
214 0A03  26: C7 45 02 079A                     mov     word ptr es:[di
                                        +2],079ah
215 0A09  26: C7 85 4002 079A                   mov     word ptr es:[di
                                        +4002h],079ah
216 0A10  FE CB                         f21:    dec     bl
217 0A12  75 DC                                 jnz     f20
218 0A14  B8 078D                               mov     ax,078dh
219 0A17  26: 89 05                             mov     es:[di],ax
220 0A1A  B8 0710                               mov     ax,0710h
221 0A1D  26: 89 85 4000                        mov     es:[di+4000h],a
                                        x
222 0A22  80 3E 08E9 R 00                       cmp     win_flag,0
223 0A27  74 1B                                 jz      f31
224 0A29  26: C7 45 02 079A                     mov     word ptr es:[di
                                        +2],079ah
225 0A2F  26: C7 85 4002 079A                   mov     word ptr es:[di
                                        +4002h],079ah
226 0A36  26: C7 85 00A2 079A                   mov     word ptr es:[di
                                        +162],079ah
227 0A3D  26: C7 85 40A2 079A                   mov     word ptr es:[di
                                        +40a2h],079ah
228 0A44  B8 078F                       f31:    mov     ax,078fh
229 0A47  8A 1E 08F0 R                          mov     bl,lg_col
230 0A4B  FE C3                                 inc     bl
231 0A4D  4F                            f30:    dec     di
232 0A4E  4F                                    dec     di
233 0A4F  26: 89 05                             mov     es:[di],ax
                                                ;draw down ---- line
234 0A52  26: C7 85 4000 0720                   mov     word ptr es:[di
```

```
                                +4000h],0720h
  235 0A59  80 3E 08E9 R 00                     cmp     win_flag,0
  236 0A5E  74 0E                               jz      f41
  237 0A60  26: C7 85 00A2 079A                 mov     word ptr es:[di
                                +162],079ah
  238 0A67  26: C7 85 40A2 079A                 mov     word ptr es:[di
                                +40a2h],079ah
  239 0A6E  FE CB                       f41:    dec     bl
  240 0A70  75 DB                               jnz     f30
  241 0A72  BB 078C                             mov     ax,078ch
  242 0A75  26: 89 05                           mov     es:[di],ax
  243 0A78  B8 0710                             mov     ax,0710h
  244 0A7B  26: 89 85 4000                      mov     es:[di+4000h],a
                                x
  245 0A80  BB 078E                             mov     ax,078eh
  246 0A83  8A 1E 08F1 R                        mov     bl,lg_row
  247 0A87  81 EF 00A0                  f40:    sub     di,160
  248 0A8B  26: 89 05                           mov     es:[di],ax
                                                ;draw left ; line
  249 0A8E  26: 89 85 4000                      mov     es:[di+4000h],a
                                x
  250 0A93  FE CB                               dec     bl
  251 0A95  75 F0                               jnz     f40
  252 0A97  07                                  pop     es
  253 0A98  5F                                  pop     di
  254 0A99  59                                  pop     cx
  255 0A9A  58                                  pop     ax
  256 0A9B  C3                                  ret
  257 0A9C                              frame   endp
  258
  259 0A9C                              save_frame      proc    near
  260 0A9C  50                                  push    ax
  261 0A9D  53                                  push    bx
  262 0A9E  51                                  push    cx
  263 0A9F  56                                  push    si
  264 0AA0  57                                  push    di
  265 0AA1  1E                                  push    ds
  266 0AA2  06                                  push    es
  267 0AA3  8A 0E 08F1 R                        mov     cl,lg_row
  268 0AA7  80 C1 03                            add     cl,3
  269 0AAA  8A 2E 08F0 R                        mov     ch,lg_col
  270 0AAE  80 C5 03                            add     ch,3
  271 0AB1  8A FD                               mov     bh,ch
  272 0AB3  8B 36 08F6 R                        mov     si,buffer
  273 0AB7  8B 3E 080A R                        mov     di,win_addr
  274 0ABB  B8 B500                             mov     ax,0b500h
  275 0ABE  8E C0                               mov     es,ax
  276 0AC0  B8 B200                             mov     ax,0b200h
  277 0AC3  8E D8                               mov     ds,ax
  278 0AC5  8A EF                       s20:    mov     ch,bh
  279 0AC7  56                                  push    si
  280 0AC8  8A 04                       s10:    mov     al,[si]
                                ; data in page1
  281 0ACA  26: 88 05                           mov     es:[di],al
```

```
0ACD  8A B4 4000                        mov     al,[si+4000h]
                                ; data in page2
0AD1  26: 88 45 01                      mov     es:[di+1],al
0AD5  8A 44 01                          mov     al,[si+1]
                                ; attribute
0AD8  26: 88 45 02                      mov     es:[di+2],al
0ADC  46                                inc     si
0ADD  46                                inc     si
0ADE  83 C7 03                          add     di,3
0AE1  FE CD                             dec     ch
0AE3  75 E3                             jnz     s10
0AE5  5E                                pop     si
0AE6  81 C6 00A0                        add     si,160
0AEA  FE C9                             dec     cl
0AEC  75 D7                             jnz     s20
0AEE  07                                pop     es
0AEF  1F                                pop     ds
0AF0  A0 0B70 R                         mov     al,win_count
0AF3  B1 0A                             mov     cl,10
0AF5  F6 E1                             mul     cl
0AF7  BE 0B0C R                         mov     si,offset win_a
                        rea
0AFA  03 F0                             add     si,ax
0AFC  A1 0BF6 R                         mov     ax,buffer
0AFF  89 04                             mov     [si],ax
0B01  A1 0B0A R                         mov     ax,win_addr
0B04  89 44 02                          mov     [si+2],ax
0B07  A0 0BF1 R                         mov     al,lg_row
0B0A  88 44 04                          mov     [si+4],al
0B0D  A0 0BF0 R                         mov     al,lg_col
0B10  88 44 05                          mov     [si+5],al
0B13  FE 06 0B70 R                      inc     win_count
0B17  89 3E 0B0A R                      mov     win_addr,di
0B1B  5F                                pop     di
0B1C  5E                                pop     si
0B1D  59                                pop     cx
0B1E  5B                                pop     bx
0B1F  58                                pop     ax
0B20  C3                                ret
0B21                            save_frame      endp

0B21                            recall          proc    near
0B21  50                                push    ax
0B22  53                                push    bx
0B23  51                                push    cx
0B24  56                                push    si
0B25  57                                push    di
0B26  1E                                push    ds
0B27  06                                push    es
0B28  FE 0E 0B70 R                      dec     win_count
0B2C  A0 0B70 R                         mov     al,win_count
0B2F  B1 0A                             mov     cl,10
0B31  F6 E1                             mul     cl
0B33  BE 0B0C R                         mov     si,offset win_a
```

```
                                rea
0B36  03 F0                             add     si,ax
0B38  8B 04                             mov     ax,[si]
0B3A  A3 08F6 R                         mov     buffer,ax
0B3D  8B 44 02                          mov     ax,[si+2]
0B40  A3 080A R                         mov     win_addr,ax
0B43  8A 44 04                          mov     al,[si+4]
0B46  A2 08F1 R                         mov     lg_row,al
0B49  8A 44 05                          mov     al,[si+5]
0B4C  A2 08F0 R                         mov     lg_col,al
0B4F  8A 0E 08F1 R                      mov     cl,lg_row
0B53  80 C1 03                          add     cl,3
0B56  8A 2E 08F0 R                      mov     ch,lg_col
0B5A  80 C5 03                          add     ch,3
0B5D  8A FD                             mov     bh,ch
0B5F  8B 36 08F6 R                      mov     si,buffer
0B63  8B 3E 080A R                      mov     di,win_addr
0B67  B8 B500                           mov     ax,0b500h
0B6A  8E C0                             mov     es,ax
0B6C  B8 B200                           mov     ax,0b200h
0B6F  8E D8                             mov     ds,ax
0B71  8A EF                     r20:    mov     ch,bh
0B73  56                                push    si
0B74  26: 8A 05                 r10:    mov     al,es:[di]
                                        ; data in page1
0B77  88 04                             mov     [si],al
0B79  26: 8A 45 01                      mov     al,es:[di+1]
                                ; data in page2
0B7D  88 84 4000                        mov     [si+4000h],al
0B81  26: 8A 45 02                      mov     al,es:[di+2]
                                ; attribute
0B85  88 44 01                          mov     [si+1],al
0B88  88 84 4001                        mov     [si+4001h],al
0B8C  46                                inc     si
0B8D  46                                inc     si
0B8E  83 C7 03                          add     di,3
0B91  FE CD                             dec     ch
0B93  75 DF                             jnz     r10
0B95  5E                                pop     si
0B96  81 C6 00A0                        add     si,160
0B9A  FE C9                             dec     cl
0B9C  75 D3                             jnz     r20

0B9E  07                                pop     es
0B9F  1F                                pop     ds
0BA0  5F                                pop     di
0BA1  5E                                pop     si
0BA2  59                                pop     cx
0BA3  5B                                pop     bx
0BA4  58                                pop     ax
0BA5  C3                                ret
0BA6                    recall          endp
0BA6                    cls             proc    near
```

```
0BA6  C7 06 08F6 R 0000                 mov     buffer,0
0BAC  C6 06 08F1 R 18                   mov     lg_row,24
0BB1  C6 06 08F0 R 4F                   mov     lg_col,79
0BB6  E8 0BBA R                         call    clear
0BB9  C3                                ret
0BBA                            cls     endp
0BBA                            clear   proc    near
0BBA  50                                push    ax
0BBB  51                                push    cx
0BBC  57                                push    di
0BBD  06                                push    es
0BBE  B8 B200                           mov     ax,0b200h
0BC1  8E C0                             mov     es,ax
0BC3  8B 3E 08F6 R                      mov     di,buffer
0BC7  B8 0720                           mov     ax,0720h
0BCA  8A 0E 08F1 R                      mov     cl,lg_row
0BCE  FE C1                             inc     cl
0BD0  57                        c20:    push    di
0BD1  8A 2E 08F0 R                      mov     ch,lg_col
0BD5  FE C5                             inc     ch
0BD7  26: 89 05                 c10:    mov     es:[di],ax
0BDA  26: 89 85 4000                    mov     es:[di+4000h],a
                                x
0BDF  47                                inc     di
0BE0  47                                inc     di
0BE1  FE CD                             dec     ch
0BE3  75 F2                             jnz     c10
0BE5  5F                                pop     di
0BE6  81 C7 00A0                        add     di,160
0BEA  FE C9                             dec     cl
0BEC  75 E2                             jnz     c20
0BEE  07                                pop     es
0BEF  5F                                pop     di
0BF0  59                                pop     cx
0BF1  58                                pop     ax
0BF2  C3                                ret
0BF3                            clear   endp
0BF3                            chk_len proc    near
0BF3  56                                push    si
0BF4  2B D2                             sub     dx,dx
0BF6  80 3C 00                  chk_again:      cmp     byte ptr [si],0
0BF9  74 04                             je      chk_ok
0BFB  46                                inc     si
0BFC  42                                inc     dx
0BFD  EB F7                             jmp     short chk_again
0BFF  83 C2 04                  chk_ok:         add     dx,4
0C02  5E                                pop     si
0C03  C3                                ret
0C04                            chk_len endp
```

```
                                        public  cur_on
0C04                            cur_on  proc    near
0C04  83 C6 03                          add     si,3
0C07  50                                push    ax
0C08  52                                push    dx
0C09  BA 03B4                           mov     dx,3b4h
0C0C  B0 0A                             mov     al,0ah
0C0E  EE                                out     dx,al
0C0F  42                                inc     dx
0C10  B0 14                             mov     al,0bh + CUR_ST
                                ART     ; cursor start
0C12  EE                                out     dx,al
0C13  4A                                dec     dx
0C14  B0 0B                             mov     al,0bh
0C16  EE                                out     dx,al
0C17  42                                inc     dx
0C18  B0 16                             mov     al,0ch + CUR_ST
                                OP      ; cursor stop
0C1A  EE                                out     dx,al
0C1B  5A                                pop     dx
0C1C  58                                pop     ax
0C1D  C3                                ret
0C1E                            cur_on  endp
                                
                                        public  cur_off
0C1E                            cur_off proc    near
0C1E  83 C6 03                          add     si,3
0C21  50                                push    ax
0C22  52                                push    dx
0C23  BA 03B4                           mov     dx,3b4h
0C26  B0 0A                             mov     al,0ah
0C28  EE                                out     dx,al
0C29  42                                inc     dx
0C2A  B0 2F                             mov     al,2fh
0C2C  EE                                out     dx,al
0C2D  5A                                pop     dx
0C2E  58                                pop     ax
0C2F  C3                                ret
0C30                            cur_off endp
                                
                                        public  locate
0C30                            locate  proc    near
0C30  50                                push    ax
0C31  51                                push    cx
0C32  52                                push    dx
0C33  33 C0                             xor     ax,ax
0C35  A0 0BED R                         mov     al,x_row
                                ; get row
0C38  B1 50                             mov     cl,80
0C3A  F6 E1                             mul     cl
0C3C  33 C9                             xor     cx,cx
0C3E  8A 0E 0BEE R                      mov     cl,x_col
0C42  03 C1                             add     ax,cx   ; get
                                col
```

```
0C44  05 3000                          add     ax,CUR_ADDR
0C47  8B CB                            mov     cx,ax
0C49  B4 0E                            mov     ah,14
0C4B  E8 0C52 R                        call    p6845
0C4E  5A                               pop     dx
0C4F  59                               pop     cx
0C50  58                               pop     ax
0C51  C3                               ret
0C52                           locate  endp
0C52                           p6845   proc    near
0C52  BA 03B4                          mov     DX,3b4h
0C55  8A C4                            mov     AL,AH
0C57  EE                               out     DX,AL
0C58  42                               inc     DX
0C59  BA C5                            mov     AL,CH
0C5B  EE                               out     DX,AL
0C5C  4A                               dec     DX
0C5D  BA C4                            mov     AL,AH
0C5F  FE C0                            inc     AL
0C61  EE                               out     DX,AL
0C62  42                               inc     DX
0C63  BA C1                            mov     AL,CL
0C65  EE                               out     DX,AL
0C66  C3                               ret
0C67                           p6845   endp

                               ; function print_string use to print th
                               ai sentense to direct screen
                               ; pointer must keep address of string t
                               o print in following format
                               ;
                               ;       row,col,"string",00,attribute

                                       public  print_string
0C67                           print_string    proc    near
0C67  50                               push    ax
0C68  51                               push    cx
0C69  52                               push    dx
0C6A  56                               push    si
0C6B  57                               push    di
0C6C  06                               push    es
0C6D  8B 36 08EA R                     mov     si,pointer
0C71  2B C0                            sub     ax,ax
0C73  8A 04                            mov     al,[si]
                                   ; get row
0C75  46                               inc     si
0C76  48                               dec     ax
0C77  B9 00A0                          mov     cx,160
0C7A  F7 E1                            mul     cx
0C7C  2B D2                            sub     dx,dx
0C7E  8A 14                            mov     dl,[si]
                                   ; get col
0C80  46                               inc     si
```

```
0CB1  4A                        dec     dx
0CB2  D1 E2                     shl     dx,1
0CB4  03 C2                     add     ax,dx
0CB6  8B F8                     mov     di,ax
                        ;  real  offset of scr_mem
0CB8  57                        push    di
0CB9  56                        push    si
0CBA  8A 26 08EC R              mov     ah,attrib
                        ;  get attbu
0CBE  B9 0000                   mov     cx,00
0CC1  B0 00                     mov     al,00
0CC3  56                        push    si
0CC4                    d1:
0CC4  46                        inc     si
0CC5  41                        inc     cx
0CC6  38 04                     cmp     [si],al
0CC8  75 FA                     jnz     d1
0CCA  5E                        pop     si
0CCB                    d2:
0CCB  B0 D6                     mov     al,0d6h
0CCD  38 04                     cmp     [si],al
0CCF  76 39                     jbe     d5
0CD1  B0 E5                     mov     al,0e5h
0CD3  38 04                     cmp     [si],al
0CD5  73 33                     jae     d5
0CD7  BB B600                   mov     bx,0b600h
0CDA  8E C3                     mov     es,bx
0CDC  26: 8A 45 FE              mov     al,es:[di-2]
0CE0  3C 20                     cmp     al,20h
0CE2  75 05                     jnz     d3
0CE4  8A 04                     mov     al,[si]
0CE6  EB 1A 90                  jmp     d4
0CE9                    d3:
0CE9  8A 5C FF                  mov     bl,[si-1]
0CEC  B7 00                     mov     bh,00h
0CEE  81 EB 00D7                sub     bx,0d7h
0CF2  1E                        push    ds
0CF3  56                        push    si
0CF4  BE 08E1 R                 mov     si,offset tab_1
                        ook
0CF7  8A 10                     mov     dl,[si+bx]
0CF9  80 EA E0                  sub     dl,0e0h
0CFC  5E                        pop     si
0CFD  1F                        pop     ds
0CFE  02 14                     add     dl,[si]
0D00  8A C2                     mov     al,dl
0D02                    d4:
0D02  26: 88 45 FE              mov     es:[di-2],al
0D06  46                        inc     si
0D07  EB 1B 90                  jmp     d6
0D0A                    d5:
0D0A  BB B200                   mov     bx,0b200h
0D0D  8E C3                     mov     es,bx
0D0F  8A 04                     mov     al,[si]
```

```
0CE1  26: 89 05                         mov     es:[di],ax
0CE4  BB B600                           mov     bx,0b600h
0CE7  8E C3                             mov     es,bx
0CE9  B0 20                             mov     al,20h
0CEB  26: 89 05                         mov     es:[di],ax
0CEE  47                                inc     di
0CEF  47                                inc     di
0CF0  46                                inc     si
0CF1                            d6:
0CF1  E2 A8                             loop    d2
0CF3  5E                                pop     si
0CF4  5F                                pop     di
0CF5  80 3E 0BE8 R 00                   cmp     mode_flag,0
0CFA  74 1D                             je      d33
0CFC  E8 0BF3 R                         call    chk_len
0CFF  83 EA 04                          sub     dx,4
0D02  8B CA                             mov     cx,dx
0D04  B8 B200                           mov     ax,0b200h
0D07  8E C0                             mov     es,ax
0D09  A0 0BEC R                         mov     al,attrib
0D0C  26: 88 45 01              d24:    mov     es:[di+1],al
0D10  26: 88 85 4001                    mov     es:[di+4001h],a
                                        l
0D15  47                                inc     di
0D16  47                                inc     di
0D17  E2 F3                             loop    d24

0D19  07                        d33:    pop     es
0D1A  5F                                pop     di
0D1B  5E                                pop     si
0D1C  5A                                pop     dx
0D1D  59                                pop     cx
0D1E  58                                pop     ax
0D1F  C3                                ret
0D20                            print_string    endp
0D20
0D20                            upcase          proc    near
0D20  3C 61                             cmp     al,'a'
0D22  72 04                             jb      up_case
0D24  2C 61                             sub     al,'a'
0D26  04 41                             add     al,'A'
0D28                            up_case:
0D28  C3                                ret
0D29                            upcase          endp
0D29
0D29                            backword        proc    near
0D29  50                                push    ax
0D2A  83 EF 02                          sub     di,2
0D2D  B8 7020                           mov     ax,7020h
0D30  AB                                stosw
0D31  FE CA                             dec     dl
0D33  4E                                dec     si
0D34  E8 0D6E R                         call    set_cursor
0D37  FE 0E 0BFF R                      dec     count
```

```
636 0D3B  83 EF 02                        sub     di,2
637 0D3E  58                              pop     ax
638 0D3F  C3                              ret
639 0D40                          backword        endp
640
641 0D40                          advance         proc    near
642 0D40  50                              push    ax
643 0D41  FE C2                           inc     dl
644 0D43  46                              inc     si
645 0D44  E8 0D6E R                       call    set_cursor
646 0D47  FE 06 08FF R                    inc     count
647 0D4B  58                              pop     ax
648 0D4C  C3                              ret
649 0D4D                          advance         endp
650
651                               ;-------------------------------------
                                  -----
652                               ; Routine to fill attribute
653                               ; di : address of screen
654                               ; cx : length to fill
655                               ;-------------------------------------
                                  -----
656
657 0D4D                          fill_att        proc    near
658 0D4D  06                              push    es
659 0D4E  57                              push    di
660 0D4F  56                              push    si
661 0D50  51                              push    cx
662 0D51  50                              push    ax
663 0D52  51                              push    cx
664 0D53  57                              push    di
665 0D54  8A 04                   fi10:           mov     al,[si]
666 0D56  B4 70                           mov     ah,70h
667 0D58  AB                              stosw
668 0D59  46                              inc     si
669 0D5A  E2 F8                           loop    fi10
670 0D5C  5F                              pop     di
671 0D5D  81 C7 4000                      add     di,4000h
672 0D61  59                              pop     cx
673 0D62  B0 20                   fi20:           mov     al,20h
674 0D64  AB                              stosw
675 0D65  46                              inc     si
676 0D66  E2 FA                           loop    fi20
677 0D68  58                              pop     ax
678 0D69  59                              pop     cx
679 0D6A  5E                              pop     si
680 0D6B  5F                              pop     di
681 0D6C  07                              pop     es
682 0D6D  C3                              ret
683 0D6E                          fill_att        endp
684
685 0D6E                          set_cursor      proc    near
686
687 0D6E  50                              push    ax
```

```
0D6F  83 C6 03                          add     si,3
0D72  AC                                lodsb
0D73  A2 08ED R                         mov     x_row,al
0D76  AC                                lodsb
0D77  A2 08EE R                         mov     x_col,al
0D7A  E8 0C30 R                         call    locate
0D7D  58                                pop     ax
0D7E  C3                                ret

0D7F                            set_cursor      endp


0D7F                                    code    ends
                                        end
```

Segments and Groups:

Name	Length	Align	Combine Class
CODE	0D7F	BYTE	PUBLIC

Symbols:

Name	Type	Value	Attr	
ADDR	L WORD	08F8	CODE	
ADVANCE	N PROC	0D40	CODE	Length = 000D
AR_DOWN	NUMBER	00F4		
AR_LEFT	NUMBER	00F1		
AR_RIGHT	NUMBER	00F2		
AR_UP	NUMBER	00F3		
ATTRIB	L BYTE	08EC	CODE	
BACKWORD	N PROC	0D29	CODE	Length = 0017
BEGIN	F PROC	0000	CODE	Length = 092C
BUFFER	L WORD	08F6	CODE	
C10	L NEAR	0BD7	CODE	
C20	L NEAR	0BD0	CODE	
CHAR	L WORD	08FA	CODE	
CHK_AGAIN	L NEAR	0BF6	CODE	
CHK_LEN	N PROC	0BF3	CODE	Length = 0011
CHK_OK	L NEAR	0BFF	CODE	
CLEAR	N PROC	0BBA	CODE	Length = 0039
CLEAR_WIN	N PROC	094C	CODE	Length = 001D
CLS	N PROC	0BA6	CODE	Length = 0014
COUNT	L BYTE	08FF	CODE	
COUNT_1	NUMBER	0800		
CUR_ADDR	NUMBER	3000		
CUR_OFF	N PROC	0C1E	CODE	Global Length = 0012
CUR_ON	N PROC	0C04	CODE	Global Length = 001A
CUR_START	NUMBER	0009		
CUR_STOP	NUMBER	000A		
D1	L NEAR	0C94	CODE	
D2	L NEAR	0C9B	CODE	
D24	L NEAR	0D0C	CODE	
D3	L NEAR	0CB9	CODE	
D33	L NEAR	0D19	CODE	
D4	L NEAR	0CD2	CODE	
D5	L NEAR	0CDA	CODE	
D6	L NEAR	0CF1	CODE	
EDIT_ADDR	L WORD	08FD	CODE	
EDIT_LEN	L BYTE	08FC	CODE	
F10	L NEAR	09D0	CODE	
F20	L NEAR	09F0	CODE	
F21	L NEAR	0A10	CODE	

```
F30 . . . . . L NEAR  0A4D   CODE
F31 . . . . . L NEAR  0A44   CODE
F40 . . . . . L NEAR  0A87   CODE
F41 . . . . . L NEAR  0A6E   CODE
F110 . . . . . L NEAR  0D54   CODE
F120 . . . . . L NEAR  0D62   CODE
FILL_ATT . . . . . N PROC  0D4D   CODE   Length = 0021
FLD_ATT . . . . . L BYTE  08DC   CODE
FLD_ATT1 . . . . . L BYTE  08DD   CODE
FLD_LEN . . . . . L BYTE  08DB   CODE
FRAME . . . . . N PROC  09AB   CODE   Length = 00F1

JUMP_TAB . . . . . L WORD  0900   CODE

KEY_OFF . . . . . N PROC  096E   CODE   Length = 0005
KEY_ON . . . . . N PROC  0969   CODE   Length = 0005

LG_COL . . . . . L BYTE  08F0   CODE
LG_ROW . . . . . L BYTE  08F1   CODE
LOCATE . . . . . N PROC  0C30   CODE   Global  Length = 0022
LWC . . . . . L BYTE  08F5   CODE
LWR . . . . . L BYTE  08F4   CODE

MODE_FLAG . . . . . L BYTE  08E8   CODE

NOT_COMMAND . . . . . L NEAR  092A   CODE
NO_FLD . . . . . L BYTE  08DA   CODE

P6845 . . . . . N PROC  0C52   CODE   Length = 0015
POINTER . . . . . L WORD  08EA   CODE
PRINT_STRING . . . . . N PROC  0C67   CODE   Global  Length = 00B9
PULLDOWN_ADDR . . . . . L WORD  0872   CODE
PULL_AREA . . . . . L BYTE  0874   CODE   Length = 0064
PULL_COUNT . . . . . L BYTE  0871   CODE
PULL_FLAG . . . . . L BYTE  0808   CODE
PULL_RET . . . . . L BYTE  0809   CODE

R10 . . . . . L NEAR  0B74   CODE
R20 . . . . . L NEAR  0B71   CODE
RECALL . . . . . N PROC  0B21   CODE   Length = 00B5

S10 . . . . . L NEAR  0AC8   CODE
S20 . . . . . L NEAR  0AC5   CODE
SAVE_FRAME . . . . . N PROC  0A9C   CODE   Length = 00B5
SELECT_ADDR . . . . . L WORD  08D8   CODE
SEL_CODE . . . . . L BYTE  08E0   CODE
SEL_DIR . . . . . L BYTE  08DE   CODE
SEL_RET . . . . . L BYTE  08DF   CODE
SET_CURSOR . . . . . N PROC  0D6E   CODE   Length = 0011
START . . . . . L NEAR  0912   CODE

TAB_LOOK . . . . . L BYTE  08E1   CODE
TEXT . . . . . L BYTE  0008   CODE   Length = 0800
```

```
UPC . . . . . . . . . . . . . .     L BYTE  08F3    CODE
UPCASE . . . . . . . . . . . .     N PROC  0D20    CODE    Length = 0009
UPR . . . . . . . . . . . . . .     L BYTE  08F2    CODE
UP_CASE . . . . . . . . . . . .     L NEAR  0D28    CODE

WIN . . . . . . . . . . . . . .     N PROC  092C    CODE    Length = 0020
WINDOW . . . . . . . . . . . .     N PROC  0973    CODE    Length = 0038
WIN_ADDR . . . . . . . . . . .     L WORD  080A    CODE
WIN_AREA . . . . . . . . . . .     L BYTE  080C    CODE    Length = 0064
WIN_COUNT . . . . . . . . . . .     L BYTE  0870    CODE
WIN_FLAG . . . . . . . . . . .     L BYTE  08E9    CODE
W_FLAG . . . . . . . . . . . .     L BYTE  08EF    CODE

X_COL . . . . . . . . . . . . .     L BYTE  08EE    CODE
X_ROW . . . . . . . . . . . . .     L BYTE  08ED    CODE

ZZZ . . . . . . . . . . . . . .     L NEAR  0912    CODE

@FILENAME . . . . . . . . . . .     TEXT  b:sumeti


    692 Source  Lines
    692 Total   Lines
    100 Symbols

  50452 + 405548 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors
```

ภาคผนวก ข แสดงซอสลิสติ้ง (SOURCE LISTING) ของโปรแกรมของคุมทางด้านสเลฟ

บรรณานุกรม

1. การสื่อสารข้อมูลและไมโครคอมพิวเตอร์เนตเวอร์ค ยืน ภู่วรวรรณ, ไพศาล สงวนหมู่ กรุงเทพมหานคร : บริษัท ซีเอ็ดยูเคชั่น จำกัด, 2528

2. IBM Corporation, Technaical Reference Personel Computer XT System. IBM Computer Hardware Reference Library, 6936808

3. Turbo C : Memory - Resident Utilties, Screen I/O And Programming Techniques

4. การสื่อสารข้อมูล ชูชัย ธนสารตั้งเจริญ B.Eng (KMIT - LADKRABANG) และ ทินกร ดุ๊ก B.ENG (KMIT - LADKRABANG) PHYSICS CENTER การพิมพ์ ,2532

5. ASSEMBLY LANGUAGE PRIMER for the IBM PC & XT by Robert Latore, A Plume / Waite Book NEW Aerican Library NEW York and Scar Borough Ontario ,1984

6. THE 8086 BOOK includes the 8088 : Russell Rector - Geeorge Alexy , OSBORNE / McGrow-Hill Berkeley, California. 1988